

CSS

Apuntes para Profesionales

Chapter 4: Selectors

CSS selectors identify specific HTML elements as targets for CSS styles. This topic covers how CSS HTML elements. Selectors use a wide range of over 50 selection methods offered by the CSS to select elements, classes, IDs, pseudo-elements and pseudo-classes, and patterns.

Section 4.1: Basic selectors

Selector	Description
*	Universal selector (all elements)
div	Tag selector (all <code>div</code> elements)
.blue	Class selector (all elements with class <code>blue</code>)
#blue	ID selector (the element with "id" attribute set to <code>blue</code>)
div.red	All elements with class <code>blue</code> and <code>red</code> (a type of Compound selector)
div#blue	ID selector (the element with "id" attribute set to <code>blue</code>)
div.class	All elements with pseudo-class
div:element	Element that matches pseudo-element
div:lang(en)	Element that matches lang declaration, for example <code>span.lang="en"</code>
div > p	child selector

Note: The value of an ID must be unique in a web page. It is a violation of the ID value of an ID more than once in the same document tree.

Section 4.2: Attribute Selectors

Attribute selectors can be used with various types of operators that change the select an element using the presence of a given attribute or attribute value.

Selector(s)	Matched element	Selects elements...
[attr]	<code>div attr</code>	Where attribute <code>attr</code> has value
[attr="val"]	<code>div attr="val"</code>	Where <code>val</code> appears in the whitespace-separated list of values
[attr~="val"]	<code>div attr="val val2 val3"</code>	Where <code>attr</code> 's value begins with <code>val</code>
[attr ="val"]	<code>div attr="val"</code>	Where <code>attr</code> 's value ends with <code>val</code>
[attr^="val"]	<code>div attr="val"</code>	Where <code>attr</code> contains <code>val</code>
[attr\$="val"]	<code>div attr="val"</code>	Where <code>attr</code> 's value is equal or starts with <code>val</code> and is followed by <code>val</code> (or <code>val2</code>)
[attr*="val"]	<code>div attr="val-oth etc"</code>	Where <code>attr</code> has value containing <code>val</code> 's letter case

Notes:
1. The attribute value can be surrounded by either single-quotes or double-quotes. No quotes at all may also work but it's not valid according to the CSS standard, and is discouraged.

Chapter 24: Grid

Grid layout is a new and powerful CSS layout system that allows to divide a web page content into rows and columns in an easy way.

Section 24.1: Basic Example

Property Possible Values
`display: grid / inline-grid`

The CSS Grid is defined as a display property. It applies to a parent element and its immediate children only.

Consider the following markup:

```
<section class="container">
  <div class="item">...</div>
  <div class="item">...</div>
  <div class="item">...</div>
  <div class="item">...</div>
  <div class="item">...</div>
  <div class="item">...</div>
</section>
```

The easiest way to define the markup structure above as a grid is to simply set its `display` property to `grid`:

```
.container {
  display: grid;
}
```

However, doing this will invariably cause all the child elements to collapse on top of one another. This is because the children do not currently know how to position themselves within the grid. But we can explicitly tell them.

First we need to tell the grid element `.container` how many rows and columns will make up its structure and we can do this using the `grid-columns` and `grid-rows` properties (note the pluralisation):

```
.container {
  display: grid;
  grid-columns: 1fr 1fr 1fr;
  grid-rows: 50px 50px;
}
```

However, this still doesn't help us much because we need to give an order to each child element. We can do this by specifying the `grid-row` and `grid-column` values which will tell it where it sits in the grid:

```
.container .item1 {
  grid-column: 1;
  grid-row: 1;
}
.container .item2 {
  grid-column: 2;
  grid-row: 1;
}
.container .item3 {
  grid-column: 1;
  grid-row: 2;
}
.container .item4 {
  grid-column: 2;
  grid-row: 2;
}
.container .item5 {
  grid-column: 3;
  grid-row: 2;
}
```

CSS Notes for Professionals

Chapter 27: Animations

Transition

Parameter	Details
property	Either the CSS property to transition on, or <code>*</code> , which specifies all transitionable properties.
duration	Transition time, either in seconds or milliseconds.
timing-function	Specifies a function to define how intermediate values for properties are computed. Common values are <code>ease</code> , <code>linear</code> , and <code>step-end</code> . Check out the timing-function cheat sheet for more.
delay	Amount of time, in seconds or milliseconds, to wait before playing the animation.

Keyframes

[from | to | **percentage**] You can either specify a set time with a percentage value, or two percentage values, in 10%, 50%, for a period of time where the keyframe's set attributes are set. Any amount of CSS attributes for the keyframe.

Section 27.1: Animations with keyframes

For multi-stage CSS animations, you can create CSS keyframes. Keyframes allow you to define multiple animation points, called a keyframe, to define more complex animation.

Basic Example

In this example, we'll make a basic background animation that cycles between all colors.

```
@keyframes rainbow-background {
  0% { background-color: #FF0000; }
  16.66% { background-color: #FF00FF; }
  33.33% { background-color: #0000FF; }
  50.00% { background-color: #00FF00; }
  66.67% { background-color: #00FF00; }
  83.33% { background-color: #0000FF; }
  100.00% { background-color: #FF0000; }
}
```

```
.rainbow-background {
  animation: rainbow-background 10s infinite;
}
```

Look About

There's a few different things to note here. First, the actual keyframes syntax.

Keyframes: `rainbow-background`

This sets the name of the animation to `rainbow-background`.

CSS Notes for Professionals

Traducido por:



200+ páginas
de consejos y trucos profesionales

Contenidos

Acerca de	1
Capítulo 1: Introducción a CSS	2
Sección 1.1: Hoja de estilos externa.....	2
Sección 1.2: Estilos internos	3
Sección 1.3: Regla CSS @import (una de las reglas CSS).....	3
Sección 1.4: Estilos en línea	4
Sección 1.5: Modificar CSS con JavaScript	4
Sección 1.6: Listas con estilo CSS	5
Capítulo 2: Estructura y formato de una Regla CSS.....	6
Sección 2.1: Lista de propiedades.....	6
Sección 2.2: Selectores múltiples.....	6
Sección 2.3: Reglas, selectores y bloques de declaración	6
Capítulo 3: Comentarios	7
Sección 3.1: Línea única.....	7
Sección 3.2: Línea múltiple	7
Capítulo 4: Selectores	8
Sección 4.1: Selectores básicos	8
Sección 4.2: Selectores de atributos.....	8
Sección 4.3: Combinadores	11
Sección 4.4: Pseudoclases	12
Sección 4.5: Pseudoclase hijo	14
Sección 4.6: Selectores de nombres de clases.....	14
Sección 4.7: Seleccionar elemento usando su ID sin la alta especificidad del selector ID.....	15
Sección 4.8: El selector :last-of-type.....	15
Sección 4.9: Ejemplo de selector CSS3 :in-range	16
Sección 4.10: El ejemplo de pseudoclase :not y pseudoclase CSS :focus-within.....	16
Sección 4.11: Booleano global con checkbox:checked y ~ (combinador general de hermanos).....	17
Sección 4.12: Selectores ID	18
Sección 4.13: Cómo aplicar estilo a un input Range	18
Sección 4.14: Ejemplo de selector pseudoclase :only-child.....	19
Capítulo 5: Background	20
Sección 5.1: Color del fondo	20
Sección 5.2: Gradientes como fondo.....	21
Sección 5.3: Imagen como fondo	23
Sección 5.4: Abreviatura background.....	24
Sección 5.5: Tamaño del fondo	25

Sección 5.6: Posición de fondo.....	29
Sección 5.7: La propiedad background-origin.....	29
Sección 5.8: Imagen de fondo múltiple.....	31
Sección 5.9: background-attachment.....	32
Sección 5.10: background-clip.....	33
Sección 5.11: background-repeat.....	34
Sección 5.12: La propiedad background-blend-mode.....	35
Sección 5.13: Color de fondo con opacidad.....	35
Capítulo 6: Centrado.....	37
Sección 6.1: Uso de Flexbox.....	37
Sección 6.2: Uso de transform.....	39
Sección 6.3: Uso de margin: 0 auto;.....	39
Sección 6.4: Uso de la alineación de texto.....	41
Sección 6.5: Uso de position: absolute.....	42
Sección 6.6: Uso de calc().....	42
Sección 6.7: Uso de line-height.....	43
Sección 6.8: Alinear verticalmente cualquier cosa con 3 líneas de código.....	43
Sección 6.9: Centrado en relación con otro elemento.....	43
Sección 6.10: Técnica del elemento fantasma (hack de Michał Czernow).....	44
Sección 6.11: Centrado vertical y horizontal sin preocuparse por la altura o la anchura.....	45
Sección 6.12: Alinear verticalmente una imagen dentro de div.....	47
Sección 6.13: Centrado con tamaño fijo.....	47
Sección 6.14: Alinear verticalmente elementos de altura dinámica.....	49
Sección 6.15: Centrado horizontal y vertical mediante diseño de tabla.....	49
Capítulo 7: El modelo de caja.....	51
Sección 7.1: ¿Qué es el modelo de caja?.....	51
Sección 7.2: box-sizing.....	52
Capítulo 8: Margin.....	54
Sección 8.1: Colapso del margen.....	54
Sección 8.2: Aplicar el margen en un lado determinado.....	56
Sección 8.3: Simplificación de la propiedad margin.....	57
Sección 8.4: Centrar horizontalmente los elementos de una página utilizando margin.....	57
Sección 8.5: Ejemplo 1:.....	58
Sección 8.6: Márgenes negativos.....	58
Capítulo 9: Padding.....	59
Sección 9.1: Abreviatura padding.....	59
Sección 9.2: Relleno en un lado determinado.....	60
Capítulo 10: Border.....	61

Sección 10.1: border-radius.....	61
Sección 10.2: border-style.....	62
Sección 10.3: Bordes múltiples.....	63
Sección 10.4: border (abreviatura).....	64
Sección 10.5: border-collapse.....	64
Sección 10.6: border-image.....	64
Sección 10.7: Creación de un borde multicolor mediante border-image.....	65
Sección 10.8: border-[left right top bottom].....	66
Capítulo 11: Outline.....	67
Sección 11.1: Resumen.....	67
Sección 11.2: outline-style.....	67
Capítulo 12: Overflow.....	69
Sección 12.1: overflow-wrap.....	69
Sección 12.2: overflow-x y overflow-y.....	70
Sección 12.3: overflow-scroll.....	70
Sección 12.4: overflow: visible.....	71
Sección 12.5: Contexto de formato de bloque creado con overflow.....	71
Capítulo 13: Media Queries.....	73
Sección 13.1: Terminología y estructura.....	73
Sección 13.2: Ejemplo básico.....	74
Sección 13.3: mediatype.....	74
Sección 13.4: Media Queries para pantallas retina y no retina.....	75
Sección 13.5: Width vs Viewport.....	75
Sección 13.6: Uso de Media Queries para distintos tamaños de pantalla.....	76
Sección 13.7: Uso de la etiqueta link.....	76
Sección 13.8: Media Queries e IE8.....	76
Capítulo 14: Float.....	78
Sección 14.1: Hacer flotar una imagen dentro del texto.....	78
Sección 14.2: Propiedad clear.....	78
Sección 14.3: Clearfix.....	79
Sección 14.4: DIV en línea usando float.....	80
Sección 14.5: Uso de la propiedad overflow para borrar floats.....	81
Sección 14.6: Diseño simple de dos columnas de ancho fijo.....	81
Sección 14.7: Diseño simple de tres columnas de ancho fijo.....	82
Sección 14.8: Layout de dos columnas Vago/Codicioso.....	83
Capítulo 15: Tipografía.....	85
Sección 15.1: La abreviatura font.....	85
Sección 15.2: quotes.....	86

Sección 15.3: font-size.....	86
Sección 15.4: Dirección del texto	86
Sección 15.5: Pilas de fuentes tipográficas	87
Sección 15.6: text-overflow.....	87
Sección 15.7: text-shadow	87
Sección 15.8: text-transform	88
Sección 15.9: letter-spacing.....	88
Sección 15.10: text-indent.....	89
Sección 15.11: text-decoration	89
Sección 15.12: word-spacing	89
Sección 15.13: font-variant	90
Capítulo 16: Diseño de caja flexible (Flexbox)	91
Sección 16.1: Centrado dinámico vertical y horizontal (align-items, justify-content)	91
Sección 16.2: Pie de página sticky de altura variable	97
Sección 16.3: Ajustar óptimamente los elementos a su contenedor	98
Sección 16.4: El Santo Grial del diseño con Flexbox.....	99
Sección 16.5: Botones perfectamente alineados dentro de tarjetas con flexbox	100
Sección 16.6: Misma altura en los contenedores anidados	102
Capítulo 17: Cascada y especificidad	103
Sección 17.1: Cálculo de la especificidad del selector	103
Sección 17.2: La declaración !important.....	105
Sección 17.3: En cascada	105
Sección 17.4: Ejemplo de especificidad más complejo	107
Capítulo 18: Colores.....	108
Sección 18.1: currentColor.....	108
Sección 18.2: Palabras clave de color	109
Sección 18.3: Valor hexadecimal.....	114
Sección 18.4: Notación rgb().....	114
Sección 18.5: Notación rgba().....	115
Sección 18.6: Notación hsl().....	115
Sección 18.7: Notación hsla().....	116
Capítulo 19: Opacidad.....	118
Sección 19.1: Propiedad opacity	118
Sección 19.2: Compatibilidad IE para 'opacity'.....	118
Capítulo 20: Unidades de longitud	119
Sección 20.1: Creación de elementos escalables mediante rems y ems	119
Sección 20.2: Tamaño de letra con rem.....	120
Sección 20.3: vmin y vmax.....	121

Sección 20.4: vh y vw	121
Sección 20.5: usando el porcentaje %	121
Capítulo 21: Pseudoelementos	123
Sección 21.1: Pseudoelementos	123
Sección 21.2: Pseudoelementos en listas.....	123
Capítulo 22: Posicionamiento	125
Sección 22.1: Elementos superpuestos con z-index.....	125
Sección 22.2: Posición absoluta.....	127
Sección 22.3: Posición fija	127
Sección 22.4: Posición relativa.....	127
Sección 22.5: Posición estática.....	128
Capítulo 23: Control del diseño.....	129
Sección 23.1: La propiedad display.....	129
Sección 23.2: Para obtener la estructura de la tabla antigua utilizando div.....	131
Capítulo 24: Grid.....	133
Sección 24.1: Ejemplo básico.....	133
Capítulo 25: Tablas.....	135
Sección 25.1: table-layout	135
Sección 25.2: empty-cells.....	135
Sección 25.3: border-collapse.....	135
Sección 25.4: border-spacing.....	136
Sección 25.5: caption-side.....	136
Capítulo 26: Transiciones	137
Sección 26.1: Abreviatura transition.....	137
Sección 26.2: cubic-bezier.....	137
Sección 26.3: transition (no abreviado).....	138
Capítulo 27: Animaciones	140
Sección 27.1: Animaciones con keyframes	140
Sección 27.2: Animaciones con la propiedad transition	141
Sección 27.3: Ejemplos de sintaxis	142
Sección 27.4: Aumentar el rendimiento de la animación con el atributo ‘will-change’	142
Capítulo 28: Transformaciones 2D	143
Sección 28.1: rotate	143
Sección 28.2: scale.....	143
Sección 28.3: skew	144
Sección 28.4: Múltiples transformaciones	144
Sección 28.5: translate.....	145
Sección 28.6: transform-origin	145

Capítulo 29: Transformaciones 3D	147
Sección 29.1: Forma de aguja o puntero de brújula mediante transformaciones 3D.....	147
Sección 29.2: Efecto de texto 3D con sombra.....	148
Sección 29.3: backface-visibility.....	150
Sección 29.4: Cubo 3D.....	150
Capítulo 30: Propiedad filter	153
Sección 30.1: blur.....	153
Sección 30.2: Sombra (usa box-shadow en su lugar si es posible).....	154
Sección 30.3: hue-rotate	154
Sección 30.4: Múltiples valores de filter.....	154
Sección 30.5: Invertir color	155
Capítulo 31: Estilo del cursor	156
Sección 31.1: Cambiar el tipo de cursor.....	156
Sección 31.2: pointer-events.....	156
Sección 31.3: caret-color	157
Capítulo 32: box-shadow	158
Sección 32.1: sombra paralela sólo en la parte inferior utilizando un pseudoelemento.....	158
Sección 32.2: sombra paralela.....	159
Sección 32.3: sombra interior	159
Sección 32.4: sombras múltiples.....	160
Capítulo 33: Formas de float	161
Sección 33.1: Forma exterior con forma básica - circle().....	161
Sección 33.2: Margen de forma.....	162
Capítulo 34: Estilos de lista	164
Sección 34.1: Posición de la viñeta.....	164
Sección 34.2: Eliminar viñetas / números.....	164
Sección 34.3: Tipo de viñeta o numeración.....	165
Capítulo 35: Contadores	166
Sección 35.1: Aplicación del estilo de números romanos al contador	166
Sección 35.2: Numere cada elemento con el contador CSS.....	167
Sección 35.3: Numeración multinivel con contadores CSS.....	167
Capítulo 36: Funciones	169
Sección 36.1: Función calc().....	169
Sección 36.2: Función attr().....	169
Sección 36.3: Función var().....	169
Sección 36.4: Función radial-gradient().....	170
Sección 36.5: Función linear-gradient().....	170
Capítulo 37: Propiedades personalizadas (variables)	171

Sección 37.1: Color variable	171
Sección 37.2: Dimensiones variables	171
Sección 37.3: Variables en cascada	171
Sección 37.4: Válidos/Invalidos.....	172
Sección 37.5: Con media queries	173
Capítulo 38: Formas de un solo elemento.....	176
Sección 38.1: Trapecio	176
Sección 38.2: Triángulos.....	176
Sección 38.3: Círculos y elipses	179
Sección 38.4: Ráfagas	180
Sección 38.5: Cuadrado	181
Sección 38.6: Cubo.....	182
Sección 38.7: Pirámide	182
Capítulo 39: Columnas.....	184
Sección 39.1: Ejemplo sencillo (recuento de columnas).....	184
Sección 39.2: Ancho de columna.....	184
Capítulo 40: Múltiple columnas	186
Sección 40.1: Crear múltiples columnas	186
Sección 40.2: Ejemplo básico.....	186
Capítulo 41: Diseño inline-block	187
Sección 41.1: Barra de navegación justificada.....	187
Capítulo 42: Herencia	188
Sección 42.1: Herencia automática.....	188
Sección 42.2: Herencia forzosa.....	188
Capítulo 43: Sprites de imágenes CSS	189
Sección 43.1: Una implementación básica	189
Capítulo 44: Recortar y enmascarar	190
Sección 44.1: Recortar y enmascarar: Generalidades y diferencias.....	190
Sección 44.2: Máscara simple que desvanece una imagen de sólido a transparente	191
Sección 44.3: Recorte (circle).....	192
Sección 44.4: Recorte (polygon).....	193
Sección 44.5: Uso de máscaras para cortar un agujero en el centro de una imagen	194
Sección 44.6: Uso de máscaras para crear imágenes con formas irregulares irregulares	195
Capítulo 45: Fragmentación	196
Sección 45.1: @media print page-break	196
Capítulo 46: Modelo de objetos CSS (CSSOM).....	197
Sección 46.1: Añadir una regla de background-image a través de CSSOM.....	197
Sección 46.2: Introducción.....	197

Capítulo 47: Característica de las Queries	198
Sección 47.1: Uso básico de @supports	198
Sección 47.2: Encadenamiento de detecciones de características	198
Capítulo 48: Contexto de apilamiento	199
Sección 48.1: Contexto de apilamiento	199
Capítulo 49: Contextos de formato de bloque	203
Sección 49.1: Usando la propiedad overflow con un valor distinto de visible	203
Capítulo 50: Centrado vertical.....	204
Sección 50.1: Centrado con display: table	204
Sección 50.2: Centrado con Flexbox	204
Sección 50.3: Centrado con transform	205
Sección 50.4: Centrar texto con line-height	205
Sección 50.5: Centrado con position: absolute	206
Sección 50.6: Centrado con pseudoelemento	206
Capítulo 51: Ajuste y colocación de objetos	208
Sección 51.1: object-fit	208
Capítulo 52: Patrones de diseño CSS	211
Sección 52.1: BEM	211
Capítulo 53: Soporte de navegadores y prefijos	213
Sección 53.1: Transiciones.....	213
Sección 53.2: transform	213
Capítulo 54: Normalización de los estilos del navegador	214
Sección 54.1: normalize.css	214
Sección 54.2: Enfoques y ejemplos	214
Capítulo 55: Trucos para Internet Explorer	216
Sección 55.1: Añadir soporte para inline-block a IE6 e IE7.....	216
Sección 55.2: Modo de alto contraste en Internet Explorer 10 y superior	216
Sección 55.3: Sólo Internet Explorer 6 e Internet Explorer 7.....	216
Sección 55.4: Sólo Internet Explorer 8	217
Capítulo 56: Rendimiento	218
Sección 56.1: Utiliza transform y opacity para evitar el diseño desencadenante	218
Créditos	221

Acerca de

Este libro ha sido traducido por rortegag.com

Si desea descargar el libro original, puede descargarlo desde:

<https://goalkicker.com/CSSBook>

Si desea contribuir con una donación, hazlo desde:

<https://www.buymeacoffee.com/GoalKickerBooks>

Por favor, siéntase libre de compartir este PDF con cualquier persona de forma gratuita, la última versión de este libro se puede descargar desde:

<https://goalkicker.com/CSSBook>

Este libro CSS Apuntes para Profesionales está compilado a partir de la [Documentación de Stack Overflow](#), el contenido está escrito por la hermosa gente de Stack Overflow. El contenido del texto está liberado bajo Creative Commons BY-SA, ver los créditos al final de este libro quién contribuyó a los distintos capítulos. Las imágenes pueden ser copyright de sus respectivos propietarios a menos que se especifique lo contrario.

Este es un libro no oficial gratuito creado con fines educativos y no está afiliado con los grupo(s) o empresa(s) oficiales de CSS ni Stack Overflow. Todas las marcas comerciales y marcas registradas son propiedad de sus respectivos propietarios de la empresa.

No se garantiza que la información presentada en este libro sea correcta ni exacta. Utilícelo bajo su propia responsabilidad.

Envíe sus comentarios y correcciones a web@petercv.com

Capítulo 1: Introducción a CSS

Versión	Fecha de publicación
1	17-12-1996
2	12-05-1998
3	13-10-2015

Sección 1.1: Hoja de estilos externa

Una hoja de estilos CSS externa puede aplicarse a cualquier número de documentos HTML colocando un elemento `<link>` en cada documento HTML.

El atributo `rel` de la etiqueta `<link>` debe establecerse en `"stylesheet"`, y el atributo `href` en la ruta relativa o absoluta a la hoja de estilos. Aunque el uso de rutas URL relativas suele considerarse una buena práctica, también pueden utilizarse rutas absolutas. En HTML5 [puede omitirse](#) el atributo `type`.

Se recomienda colocar la etiqueta `<link>` en la etiqueta `<head>` del archivo HTML para que los estilos se carguen antes que los elementos a los que aplican el estilo. De lo contrario, [los usuarios verán un destello de contenido sin estilo](#).

Ejemplo

hola-mundo.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1> ¡Hola mundo! </h1>
    <p> I ❤️ CSS </p>
  </body>
</html>
```

style.css

```
h1 {
  color: green;
  text-decoration: underline;
}

p {
  font-size: 25px;
  font-family: 'Trebuchet MS', sans-serif;
}
```

Asegúrate de incluir la ruta correcta a tu archivo CSS en el `href`. Si el archivo CSS está en la misma carpeta que el archivo HTML, no es necesaria ninguna ruta (como en el ejemplo anterior), pero si está guardado en una carpeta, especificálo así `href="nombrecarpeta/style.css"`.

```
<link rel="stylesheet" type="text/css" href="nombrecarpeta/style.css">
```

Las hojas de estilo externas se consideran la mejor forma de manejar tu CSS. Hay una razón muy simple para ello: cuando gestionas un sitio de, digamos, 100 páginas, todas controladas por una única hoja de estilo, y quieres cambiar los colores de los enlaces de azul a verde, es mucho más fácil hacer el cambio en tu archivo CSS y dejar que los cambios "cascada" a través de las 100 páginas que ir a 100 páginas separadas y hacer el mismo cambio 100 veces. Una vez más, si desea cambiar por completo el aspecto de su sitio web, sólo tiene que actualizar este archivo.

Puede cargar tantos archivos CSS en su página HTML como necesite.

```
<link rel="stylesheet" type="text/css" href="main.css">
<link rel="stylesheet" type="text/css" href="invalidar.css">
```

Las reglas CSS se aplican con algunas reglas básicas, y el orden sí importa. Por ejemplo, si tiene un archivo main.css con algún código en él:

```
p.green { color: #00FF00; }
```

Todos tus párrafos con la clase 'verde' se escribirán en verde claro, pero puedes anular esto con otro archivo .css simplemente incluyéndolo *después* de main.css. Puedes tener invalidar.css con el siguiente código siga el main.css, por ejemplo:

```
p.green { color: #006600; }
```

Ahora todos tus párrafos con la clase 'verde' se escribirán en verde más oscuro en lugar de verde claro.

Se aplican otros principios, como la regla `!important`, la especificidad y la herencia.

Cuando alguien visita por primera vez su sitio web, su navegador descarga el HTML de la página actual más el archivo CSS enlazado. Luego, cuando navegan a otra página, su navegador sólo necesita descargar el HTML de esa página; el archivo CSS se almacena en caché, por lo que no es necesario descargarlo de nuevo. Dado que los navegadores almacenan en caché la hoja de estilos externa, sus páginas se cargan más rápido.

Sección 1.2: Estilos internos

El CSS encerrado en etiquetas `<style></style>` dentro de un documento HTML funciona como una hoja de estilos externa, salvo que vive en el documento HTML al que aplica el estilo en lugar de en un archivo independiente y, por tanto, sólo puede aplicarse al documento en el que se encuentra. Tenga en cuenta que este elemento *debe* estar dentro del elemento `<head>` para la validación HTML (aunque funcionará en todos los navegadores actuales si se coloca en body).

```
<head>
  <style>
    h1 {
      color: green;
      text-decoration: underline;
    }
    p {
      font-size: 25px;
      font-family: 'Trebuchet MS', sans-serif;
    }
  </style>
</head>
<body>
  <h1> ¡Hola mundo! </h1>
  <p> I ❤️ CSS </p>
</body>
```

Sección 1.3: Regla CSS @import (una de las reglas CSS)

La regla @import CSS se utiliza para importar reglas de estilo de otras hojas de estilo. Estas reglas deben preceder a todos los demás tipos de reglas, excepto a las reglas @charset; como no es una sentencia anidada, @import no puede utilizarse dentro de reglas condicionales dentro de reglas at condicionales. [@import](#).

Cómo utilizar @import

Puede utilizar la regla @import de las siguientes formas:

A. Con etiqueta de estilo interna

```
<style>
  @import url('/css/styles.css');
</style>
```

B. Con hoja de estilo externa

La siguiente línea importa un archivo CSS llamado `additional-styles.css` del directorio raíz al archivo CSS en el que aparece:

```
@import '/additional-styles.css';
```

También es posible importar CSS externo. Un caso de uso común son los archivos de fuentes de letra.

```
@import 'https://fonts.googleapis.com/css?family=Lato';
```

El segundo argumento opcional de la regla `@import` es una lista de consultas de medios:

```
@import '/print-styles.css' print;
@import url('landscape.css') screen and (orientation:landscape);
```

Sección 1.4: Estilos en línea

Utiliza los estilos en línea para aplicar estilo a un elemento específico. Ten en cuenta que esto **no** es lo óptimo. Se recomienda colocar las reglas de estilo en una etiqueta `<style>` o en un archivo CSS externo para mantener la distinción entre contenido y presentación.

Los estilos en línea anulan cualquier CSS de una etiqueta `<style>` o de una hoja de estilos externa. Aunque esto puede ser útil en algunas circunstancias, la mayoría de las veces reduce la mantenibilidad de un proyecto.

Los estilos del siguiente ejemplo se aplican directamente a los elementos a los que se adjuntan.

```
<h1 style="color: green; text-decoration: underline;"> ¡Hola mundo! </h1>
<p style="font-size: 25px; font-family: 'Trebuchet MS';"> I ♥ CSS </p>
```

Los estilos en línea suelen ser la forma más segura de garantizar la compatibilidad de representación en varios clientes de correo electrónico, programas y dispositivos, pero pueden llevar mucho tiempo y ser un poco difíciles de gestionar.

Sección 1.5: Modificar CSS con JavaScript

JavaScript puro

Es posible añadir, eliminar o cambiar valores de propiedades CSS con JavaScript a través de la propiedad `style` de un elemento.

```
var el = document.getElementById("elemento");
el.style.opacity = 0.5;
el.style.fontFamily = 'sans-serif';
```

Tenga en cuenta que las propiedades de estilo se nombran en minúsculas. En el ejemplo se ve que la propiedad CSS `font-family` se convierte en `fontFamily` en JavaScript.

Como alternativa a trabajar directamente sobre los elementos, puede crear un elemento `<style>` o `<link>` en JavaScript y añadirlo al `<body>` o `<head>` del documento HTML.

jQuery

Modificar las propiedades CSS con jQuery es aún más sencillo.

```
$('#elemento').css('margin', '5px');
```

Si necesitas cambiar más de una regla de estilo:

```
$('#elemento').css({
    margin: "5px",
    padding: "10px",
    color: "black"
});
```

jQuery incluye dos formas de cambiar las reglas CSS que contienen guiones (por ejemplo, `font-size`). Puede ponerlos entre comillas el nombre de la regla de estilo.

```
$('.ejemplo-clase').css({
    "background-color": "blue",
    fontSize: "10px"
});
```

Vea también

- Documentación de JavaScript - Lectura y modificación de estilos CSS
- [Documentación de jQuery - Manipulación CSS](#)

Sección 1.6: Listas con estilo CSS

Existen tres propiedades diferentes para dar estilo a los elementos de lista: `list-style-type`, `list-style-image` y `list-style-position`, que deben declararse en ese orden. Los valores por defecto son `disc`, `outside` y `none`, respectivamente. Cada propiedad puede declararse por separado, o utilizando la propiedad abreviada `list-style`.

`list-style-type` define la forma o tipo de viñeta utilizada para cada elemento de la lista.

Algunos de los valores aceptables para `list-style-type`:

- disc
- circle
- square
- decimal
- lower-roman
- upper-roman
- none

(Para obtener una lista exhaustiva, consulte la [wiki de especificaciones del W3C](#))

Para utilizar viñetas cuadradas para cada elemento de la lista, por ejemplo, utilizaría el siguiente propiedad-valor:

```
li {
    list-style-type: square;
}
```

La propiedad `list-style-image` determina si el icono del elemento de lista se establece con una imagen, y acepta un valor de `none` o una URL que apunte a una imagen.

```
li {
    list-style-image: url(imagenes/bala.png);
}
```

La propiedad `list-style-position` define dónde colocar el marcador de elemento de lista, y acepta uno de dos valores: `inside` o `outside`.

```
li {
    list-style-position: inside;
}
```

Capítulo 2: Estructura y formato de una Regla CSS

Sección 2.1: Lista de propiedades

Algunas propiedades pueden tomar múltiples valores, conocidos colectivamente como lista de propiedades.

```
/* Dos valores en la lista de propiedades */
span {
    text-shadow: yellow 0 0 3px, green 4px 4px 10px;
}
/* Formato alternativo */
span {
    text-shadow:
    yellow 0 0 3px,
    green 4px 4px 10px;
}
```

Sección 2.2: Selectores múltiples

Cuando agrupas selectores CSS, aplicas los mismos estilos a varios elementos diferentes sin repetir los estilos en tu hoja de estilos. Utilice una coma para separar varios selectores agrupados.

```
div, p { color: blue }
```

Así que el color azul se aplica a todos los elementos `<div>` y a todos los elementos `<p>`. Sin la coma sólo serían rojos los elementos `<p>` hijos de un `<div>`.

Esto también se aplica a todos los tipos de selectores.

```
p, .azul, #primero, div span{ color : blue }
```

Esta regla se aplica a:

- `<p>`
- elementos de la clase `azul`
- elemento con el ID `primero`
- cada `` dentro de un `<div>`

Sección 2.3: Reglas, selectores y bloques de declaración

Una **regla** CSS consta de un **selector** (por ejemplo, `h1`) y un **bloque de declaración** (`{}`).

```
h1 {}
```

Capítulo 3: Comentarios

Sección 3.1: Línea única

```
/* Esto es un comentario CSS */  
div {  
    color: red; /* Esto es un comentario CSS */  
}
```

Sección 3.2: Línea múltiple

```
/*  
Esto  
es  
un  
comentario  
CSS */  
div {  
    color: red;  
}
```

Capítulo 4: Selectores

Los selectores CSS identifican elementos HTML específicos como objetivos de los estilos CSS. Este tema trata sobre cómo los selectores CSS elementos HTML. Los selectores utilizan una amplia gama de los más de 50 métodos de selección que ofrece el lenguaje CSS, incluidos elementos, clases, IDs, pseudoelementos y pseudoclases, y patrones.

Sección 4.1: Selectores básicos

Selector	Descripción
*	Selector universal (todos los elementos)
div	Selector de etiquetas (todos los elementos <code><div></code>)
.azul	Selector de clase (todos los elementos con clase azul)
.azul.rojo	Todos los elementos con clase azul y rojo (un tipo de selector compuesto)
#titular	Selector ID (el elemento con el atributo "id" definido como titular)
:pseudoclase	Todos los elementos con pseudoclase
::pseudoelemento	Elemento que coincide con el pseudoelemento
:lang(en)	Elemento que coincide con la declaración :lang, por ejemplo <code></code>
div > p	Selector hijo

Nota: El valor de un ID debe ser único en una página web. Es una violación del [estándar HTML](#) utilizar el valor de un ID más de una vez en el mismo árbol de documentos.

Sección 4.2: Selectores de atributos

Resumen

Los selectores de atributos pueden utilizarse con varios tipos de operadores que modifican los criterios de selección en consecuencia. Seleccionan un elemento utilizando la presencia de un atributo o valor de atributo determinado.

Selector (1)	Elemento coincidente	Selecciona elementos...	Versión CSS
[attr]	<code><div attr></code>	Con el atributo attr	2
[attr='val']	<code><div attr="val"></code>	Donde el atributo attr tiene valor val	2
[attr~='val']	<code><div attr="val val2 val3"></code>	Donde val aparece en la lista separada por espacios en blanco de attr	2
[attr^='val']	<code><div attr="val1 val2"></code>	Cuando el valor de attr empieza por val	3
[attr\$='val']	<code><div attr="sth aval"></code>	Cuando el valor de attr termina en val	3
[attr*='val']	<code><div attr="somevalhere"></code>	Donde attr contiene val en cualquier lugar	3
[attr ='val']	<code><div attr="val-sth etc"></code>	Cuando el valor de attr es exactamente val, o empieza por val e inmediatamente seguido de - (U+002D)	3
[attr='val' i]	<code><div attr="val"></code>	Donde attr tiene valor val, ignorando si es minúscula o mayúscula de val.	4(2)

Notas:

1. El valor del atributo puede ir entre comillas simples o dobles. También puede funcionar sin comillas, pero no es válido según el estándar CSS y se desaconseja.
2. No existe una especificación CSS4 única e integrada, ya que está dividida en módulos independientes. Sin embargo, existen módulos de "nivel 4". Consulta la [compatibilidad con navegadores](#).

Detalles

[atributo]

Selecciona elementos con el atributo dado.

```
div[data-color] {
  color: red;
}
<div data-color="red">Esto será rojo </div>
<div data-color="green">Esto será rojo </div>
<div data-background="red">Esto NO será rojo </div>
```

[Demostración en JSBin](#)

[atributo="valor"]

Selecciona elementos con el atributo y el valor dados.

```
div[data-color="red"] {
  color: red;
}
<div data-color="red">Esto será rojo </div>
<div data-color="green">This will NOT be red</div>
<div data-color="blue">Esto NO será rojo </div>
```

[Demostración en JSBin](#)

[atributo*="valor"]

Selecciona los elementos con el atributo y el valor dados cuando el atributo dado contiene el valor dado en cualquier lugar (como una subcadena de caracteres).

```
[class*="foo"] {
  color: red;
}
<div class="foo-123">Esto será rojo </div>
<div class="foo123">Esto será rojo </div>
<div class="bar123foo">Esto será rojo </div>
<div class="barfoo0123">Esto será rojo </div>
<div class="barfo0">Esto NO será rojo </div>
```

[Demostración en JSBin](#)

[atributo~="valor"]

Selecciona elementos con el atributo y el valor dados cuando el valor dado aparece en una lista separada por espacios en blanco.

```
[class~="color-red"] {
  color: red;
}
<div class="color-red foo-bar the-div">Esto será rojo </div>
<div class="color-blue foo-bar the-div">Esto NO será rojo </div>
```

[Demostración en JSBin](#)

[atributo^="valor"]

Selecciona elementos con el atributo y valor dados donde el atributo dado comienza con el valor.

```
[class^="foo-"] {
    color: red;
}
<div class="foo-123"> Esto será rojo </div>
<div class="foo-234"> Esto será rojo </div>
<div class="bar-123"> Esto NO será rojo </div>
```

[Demostración en JSBin](#)

[atributo\$="valor"]

Selecciona elementos con el atributo y el valor dados en los que el atributo termina con el valor dado.

```
[class$="file"] {
    color: red;
}
<div class="foobar-file"> Esto será rojo </div>
<div class="foobar-file"> Esto será rojo </div>
<div class="foobar-input"> Esto NO será rojo </div>
```

[Demostración en JSBin](#)

[atributo]="valor"]

Selecciona elementos con un atributo y un valor dados en los que el valor del atributo es exactamente el valor dado o es exactamente el valor dado seguido de - (U+002D)

```
[lang]="ES"] {
    color: red;
}
<div lang="ES-es"> Esto será rojo </div>
<div lang="ES-ar"> Esto será rojo </div>
<div lang="PT-pt"> Esto NO será rojo </div>
```

[Demostración en JSBin](#)

[atributo="valor" i]

Selecciona elementos con un atributo y un valor dados, donde el valor del atributo puede representarse como Valor, VALOR, vALOr o cualquier otra posibilidad que no distinga entre mayúsculas y minúsculas.

```
[lang="ES" i] {
    color: red;
}
<div lang="ES"> Esto será rojo </div>
<div lang="es"> Esto será rojo </div>
<div lang="PT"> Esto NO será rojo </div>
```

[Demostración en JSBin](#)

Especificidad de los selectores de atributos

0-1-0

Igual que el selector de clase y la pseudoclase.

```
*[type=checkbox] // 0-1-0
```

Tenga en cuenta que esto significa que se puede utilizar un selector de atributo para seleccionar un elemento por su ID con un nivel de especificidad menor que si se seleccionara con un selector de ID: `[id="mi-ID"]` se dirige al mismo elemento que `#mi-ID` pero con menor especificidad.

Consulta la sección Sintaxis para obtener más información.

Sección 4.3: Combinadores

Resumen

Selector	Descripción
<code>div span</code>	Selector descendiente (todos los <code></code> s que son descendientes de un <code><div></code>)
<code>div > span</code>	Selector de hijo (todos los <code></code> s que son hijos directos de un <code><div></code>).
<code>a ~ span</code>	Selector general de hermanos (todos los <code></code> s que son hermanos después de un <code><a></code>)
<code>a + span</code>	Selector de hermanos adyacentes (todos los <code></code> s que están inmediatamente después de un <code><a></code>)

Nota: Los selectores hermanos se dirigen a los elementos que les siguen en el documento fuente. CSS, por su naturaleza (funciona en cascada), no puede dirigirse a elementos *anteriores* o *padres*. Sin embargo, utilizando la propiedad `flex order`, [se puede simular un selector de hermanos anterior puede simularse en medios visuales](#).

Combinador descendiente: `selector selector`

Un combinador descendiente, representado por al menos un carácter de espacio (), selecciona los elementos que son descendientes de del elemento definido. Este combinador selecciona **todos** los descendientes del elemento (desde los elementos hijos hacia abajo).

```
div p {
  color:red;
}
<div>
  <p> Mi texto es rojo </p>
  <section>
    <p> Mi texto es rojo </p>
  </section>
</div>
<p> Mi texto no es rojo </p>
```

[Demostración en JSBin](#)

En el ejemplo anterior, se seleccionan los dos primeros elementos `<p>` ya que ambos son descendientes del `<div>`.

Combinador hijo: `selector > selector`

El combinador hijo (>) se utiliza para seleccionar elementos que son **hijos**, o **descendientes directos**, del elemento especificado.

```
div > p {
  color:red;
}
<div>
  <p> Mi texto es rojo </p>
  <section>
    <p> Mi texto no es rojo </p>
  </section>
</div>
```

[Demostración en JSBin](#)

El CSS anterior selecciona sólo el primer elemento `<p>`, ya que es el único párrafo que desciende directamente de un `<div>`.

El segundo elemento `<p>` no se selecciona porque no es hijo directo del `<div>`.

Combinador de hermanos adyacentes: selector + selector

El combinador de hermanos adyacentes (+) selecciona un elemento hermano que sigue inmediatamente a un elemento especificado.

```
p + p {
    color: red;
}
<p> Mi texto no es rojo </p>
<p> Mi texto es rojo </p>
<p> Mi texto es rojo </p>
<hr>
<p> Mi texto no es rojo </p>
```

[Demostración en JSBin](#)

El ejemplo anterior sólo selecciona los elementos `<p>` que están *directamente precedidos* por otro elemento `<p>`.

Combinador general de hermanos: selector ~ selector

El combinador general de hermanos (~) selecciona todos los hermanos que siguen al elemento especificado.

```
p ~ p {
    color: red;
}
<p> Mi texto es no rojo </p>
<p> Mi texto es rojo </p>
<hr>
<h1> Y ahora un título </h1>
<p> Mi texto es rojo </p>
```

[Demostración en JSBin](#)

El ejemplo anterior selecciona todos los elementos `<p>` precedidos por otro elemento `<p>`, sean o no inmediatamente adyacentes.

Sección 4.4: Pseudoclases

Las [pseudoclases](#) son **palabras clave** que permiten la selección basada en información que se encuentra fuera del árbol del documento o que no puede ser expresada por otros selectores o combinadores. Esta información puede asociarse a un determinado estado (pseudoclases [estado](#) y [dinámico](#)), a localización (pseudoclases [estructurales](#) y [target](#)), a negaciones de lo anterior (pseudoclase [not](#)) o a idiomas (pseudoclase [lang](#)). Por ejemplo, si se ha seguido o no un enlace ([:visited](#)), si el ratón está sobre un elemento ([:hover](#)), si una casilla está marcada ([:checked](#)), etc.

Síntaxis

```
selector:pseudoclase{
    propiedad: VALOR;
}
```

Lista de pseudoclases

Nombre	Descripción
:active	Se aplica a cualquier elemento activado (es decir, pulsado) por el usuario.
:is (antes <code>:any</code>)	Le permite crear conjuntos de selectores relacionados mediante la creación de grupos con los que coincidirán los elementos incluidos. Se trata de una alternativa a la repetición de un selector completo.
:target	Selecciona el elemento <code>#noticias</code> activo en ese momento (al hacer clic en una URL que contenga ese nombre de anclaje)
:checked	Se aplica a los elementos de radio, casilla de verificación u opción que están marcados o activados.
:default	Representa cualquier elemento de la interfaz de usuario que sea el predeterminado entre un grupo de elementos similares.
:disabled	Se aplica a cualquier elemento de la interfaz de usuario que se encuentre en estado desactivado.
:empty	Se aplica a cualquier elemento que no tenga hijos.
:enabled	Se aplica a cualquier elemento de la interfaz de usuario que se encuentre en estado activado.
:first	Utilizada junto con la regla <code>@page</code> , selecciona la primera página de un documento impreso. documento impreso.
:first-child	Representa cualquier elemento que es el primer elemento hijo de su padre.
:first-of-type	Se aplica cuando un elemento es el primero del tipo de elemento seleccionado dentro de su padre. Puede ser o no el primer hijo.
:focus	Se aplica a cualquier elemento que tenga el foco del usuario. Esto puede ser dado por el teclado del usuario, eventos del ratón u otras formas de entrada.
:focus-within	Puede utilizarse para resaltar toda una sección cuando se enfoca un elemento dentro de ella. Coincide con cualquier elemento con el que coincida la pseudoclase <code>:focus</code> o que tenga un descendiente enfocado.
:fullscreen	Se aplica a cualquier elemento visualizado en modo de pantalla completa. Selecciona toda la pila de elementos y no sólo el elemento de nivel superior.
:hover	Se aplica a cualquier elemento sobre el que pase el puntero del usuario, pero no se activa.
:indeterminate	Se aplica a los elementos de la interfaz de usuario que no están marcados ni desmarcados, sino que se encuentran en un estado indeterminado. Esto puede deberse a un o a la manipulación del DOM.
:in-range	La pseudoclase CSS <code>:in-range</code> coincide cuando un elemento tiene su atributo <code>value</code> dentro de las limitaciones de rango especificadas para este elemento. Permite que la página indique que el valor definido actualmente utilizando el elemento está dentro de los límites del rango.
:invalid	Se aplica a los elementos <code><input></code> cuyos valores no son válidos según el tipo especificado en el atributo <code>type=</code> .
:lang	Se aplica a cualquier elemento cuyo elemento envolvente <code><body></code> tenga un atributo correctamente designado. Para que la pseudoclase sea válida, debe contener un código de idioma válido de dos o tres letras .
:last-child	Representa cualquier elemento que sea el último elemento hijo de su padre.
:last-of-type	Se aplica cuando un elemento es el último del tipo de elemento seleccionado dentro de de su padre. Este puede o no ser el último hijo.
:left	Utilizada junto con la regla <code>@page</code> , selecciona todas las páginas izquierdas de un documento impreso.
:link	Se aplica a cualquier enlace que no haya sido visitado por el usuario.
:not()	Se aplica a todos los elementos que no coinciden con el valor pasado a <code>:not(p)</code> o <code>:not(. nombre-clase)</code> por ejemplo. Debe tener un valor para ser válido y sólo puede contener un selector. Sin embargo, puede encadenar varios selectores <code>:not</code> entre sí.
:nth-child	Se aplica cuando un elemento es el n-ésimo elemento de su padre, donde n puede ser un número entero, una expresión matemática o las palabras clave <code>par</code> o <code>impar</code> .

:nth-of-type	Se aplica cuando un elemento es el n-ésimo elemento de su padre del mismo tipo de elemento, donde n puede ser un entero, una expresión (por ejemplo, <code>n+3</code>) o las palabras clave par o impar.
:only-child	La pseudoclase CSS <code>:only-child</code> representa cualquier elemento que es el único hijo de su padre. Es lo mismo que <code>:first-child:last-child</code> o <code>:nth-child(1):nth-last-child(1)</code> , pero con una especificidad menor.
:optional	La pseudoclase CSS <code>:optional</code> representa cualquier elemento que no tiene el atributo requerido. Esto permite a los formularios indicar fácilmente los campos opcionales y aplicarles el estilo correspondiente.
:out-of-range	La pseudoclase CSS <code>:out-of-range</code> coincide cuando un elemento tiene su atributo <code>value</code> fuera de las limitaciones de rango especificadas para este elemento. Permite a la página informar de que el valor definido actualmente mediante el elemento está fuera de los límites del intervalo. Un valor puede estar fuera de un intervalo si es menor o mayor que los valores máximos y mínimos establecidos.
:placeholder-shown	Experimental. Se aplica a cualquier elemento de formulario que muestre texto de marcador de posición.
:read-only	Se aplica a cualquier elemento que no sea editable por el usuario.
:read-write	Se aplica a cualquier elemento editable por un usuario, como los elementos <code><input></code> .
:right	Utilizada junto con la regla <code>@page</code> , selecciona todas las páginas correctas de un documento impreso.
:root	Coincide con el elemento raíz de un árbol que representa el documento.
:scope	La pseudoclase CSS coincide con los elementos que son un punto de referencia para los selectores.
:target	Selecciona el elemento <code>#noticias</code> activo en ese momento (al hacer clic en una URL que contenga ese nombre de anclaje)
:visited	Se aplica a todos los enlaces visitados por el usuario.

La pseudoclase `:visited` ya no se puede utilizar para la mayoría de los estilos en muchos navegadores modernos porque es un agujero de seguridad. Consulta este [enlace](#) como referencia.

Sección 4.5: Pseudoclase hijo

"La pseudo-clase CSS `:nth-child(an+b)` coincide con un elemento que tiene `an+b-1` hermanos antes que él en el árbol del documento, para un valor positivo o nulo de `n`". - [MDN :nth-child](#)

pseudo-selector	1	2	3	4	5	6	7	8	9	10
<code>:first-child</code>	✓									
<code>:nth-child(3)</code>			✓							
<code>:nth-child(n+3)</code>			✓	✓	✓	✓	✓	✓	✓	✓
<code>:nth-child(3n)</code>			✓			✓			✓	
<code>:nth-child(3n+1)</code>	✓			✓			✓			✓
<code>:nth-child(-n+3)</code>	✓	✓	✓							
<code>:nth-child(odd)</code>	✓		✓		✓		✓		✓	
<code>:nth-child(even)</code>		✓		✓		✓		✓		✓
<code>:last-child</code>										✓
<code>:nth-last-child(3)</code>							✓			

Sección 4.6: Selectores de nombres de clases

El selector de nombre de clase selecciona todos los elementos con el nombre de clase seleccionado. Por ejemplo, el nombre de clase `.advertencia` seleccionaría el siguiente elemento `<div>`:

```
<div class="advertencia">
  <p> Esto sería una copia de advertencia. </p>
</div>
```

También puede combinar nombres de clases para orientar los elementos de forma más específica. Basándose en el ejemplo anterior para una selección de clases más complicada.

CSS

```
.importante {
  color: orange;
}
.advertencia {
  color: blue;
}
.advertencia.importante {
  color: red;
}
```

HTML

```
<div class="advertencia">
  <p> Esto sería una copia de advertencia. </p>
</div>
<div class="importante advertencia">
  <p class="importante"> Esta es una copia de advertencia realmente importante. </p>
</div>
```

En este ejemplo, todos los elementos con la clase `.advertencia` tendrán un color de texto azul, los elementos con la clase `.importante` tendrán un color de texto naranja, y todos los elementos que tengan tanto el nombre de clase `.importante` como `.advertencia` tendrán un color de texto rojo.

Observe que en el CSS, la declaración `.advertencia.importante` no tiene ningún espacio entre los dos nombres de clase de clase. Esto significa que sólo encontrará elementos que contengan tanto nombres de clase `advertencia` como `importante` en su atributo `class`. Esos nombres de clase pueden estar en cualquier orden en el elemento.

Si se incluyera un espacio entre las dos clases en la declaración CSS, sólo seleccionaría los elementos que tuvieran elementos padre con nombres de clase `.advertencia` y elementos hijo con nombres de clase `.importante`.

Sección 4.7: Seleccionar elemento usando su ID sin la alta especificidad del selector ID

Este truco le ayuda a seleccionar un elemento utilizando el ID como valor de un selector de atributo para evitar la alta especificidad del selector ID.

HTML

```
<div id="elemento">...</div>
```

CSS

```
#elemento { ... } /* La alta especificidad anulará muchos selectores */
[id="elemento"] { ... } /* Baja especificidad, se puede anular fácilmente */
```

Sección 4.8: El selector `:last-of-type`

La opción `:last-of-type` selecciona el elemento que es el último hijo, de un tipo determinado, de su padre. En el siguiente ejemplo el CSS selecciona el último párrafo y el último título `h1`.

```

p:last-of-type {
    background: #C5CAE9;
}
h1:last-of-type {
    background: #CDDC39;
}
<div class="contenedor">
  <p> Primer párrafo </p>
  <p> Segundo párrafo </p>
  <p> Último párrafo </p>
  <h1>Encabezado 1</h1>
  <h2> Primer encabezamiento 2</h2>
  <h2> Primer encabezamiento 2</h2>
</div>

```

Primer párrafo

Segundo párrafo

Último párrafo

Encabezado 1

Primer encabezamiento 2

Primer encabezamiento 2

Sección 4.9: Ejemplo de selector CSS3 :in-range

```

<style>
  input:in-range {
    border: 1px solid blue;
  }
</style>
<input type="number" min="10" max="20" value="15">
<p> El borde de este valor será azul </p>

```

La pseudoclase CSS `:in-range` coincide cuando un elemento tiene su atributo `value` dentro del rango especificado para este elemento. Permite que la página indique que el valor definido actualmente mediante el elemento está dentro de los límites del rango [1].

Sección 4.10: El ejemplo de pseudoclase :not y pseudoclase CSS :focus-within

A. La sintaxis se presenta a continuación

El siguiente selector coincide con todos los elementos `<input>` de un documento HTML que no estén desactivados y no tengan la clase `.ejemplo`:

HTML

```

<form>
  Teléfono: <input type="tel" class="ejemplo">
  Correo electrónico: <input type="email" disabled="disabled">
  Contraseña: <input type="password">
</form>

```

CSS

```
input:not([disabled]):not(.ejemplo){
  background-color: #ccc;
}
```

La pseudoclase `:not()` también admitirá selectores separados por comas en el nivel 4 de selectores

[Demostración en JSBin](#)

B. La pseudoclase CSS `:focus-within`

HTML

```
<h3>El fondo es azul si la entrada está enfocada. </h3>
<div>
  <input type="text">
</div>
```

CSS

```
div {
  height: 80px;
}
input{
  margin:30px;
}
div:focus-within {
  background-color: #1565C0;
}
```

```
div {
  height: 80px;
}
input{
  margin:30px;
}
div:focus-within {
  background-color: #1565C0;
}
```

Background is blue if the input is focused .



Consulte la [compatibilidad entre navegadores](#).

Sección 4.11: Booleano global con `checkbox:checked` y `~` (combinador general de hermanos)

Con el selector `~`, puede implementar fácilmente un booleano global accesible sin utilizar JavaScript.

Añadir booleano como casilla de verificación

Al principio de su documento, añada tantos booleanos como desee con un identificador único y el atributo `hidden`:

```
<input type="checkbox" id="sidebarShown" hidden />
<input type="checkbox" id="darkThemeUsed" hidden />
<!-- aquí comienza el contenido real, por ejemplo: -->
<div id="container">
  <div id="sidebar">
    <!-- Menú, Buscar, ... -->
  </div>
  <!-- Más contenido ... -->
</div>
<div id="footer">
  <!-- ... -->
</div>
```

Cambiar el valor del booleano

Puede alternar el booleano añadiendo un `label` con el atributo `for` activado:

```
<label for="sidebarShown"> ¡Mostrar/Ocultar la barra lateral! </label>
```

Acceso a valores booleanos con CSS

El selector normal (como `.color-red`) especifica las propiedades por defecto. Pueden anularse mediante los siguientes selectores `true` / `false`:

```
/* verdadero: */  
<checkbox>:checked ~ [hermano de checkbox & padre de target] <target>  
/* falso: */  
<checkbox>:not(:checked) ~ [hermano de checkbox & padre de target] <target>
```

Tenga en cuenta que `<checkbox>`, `[hermano ...]` y `<target>` deben sustituirse por los selectores adecuados. `[hermano ...]` puede ser un selector específico, (a menudo si eres perezoso) simplemente `*` o nada si el objetivo ya es un hermano de la casilla de verificación.

Ejemplos para la estructura HTML anterior serían:

```
#sidebarShown:checked ~ #container #sidebar {  
    margin-left: 300px;  
}  
#darkThemeUsed:checked ~ #container,  
#darkThemeUsed:checked ~ #footer {  
    background: #333;  
}
```

Sección 4.12: Selectores ID

Los selectores de ID seleccionan elementos DOM con el ID deseado. Para seleccionar un elemento por un ID específico en CSS, se utiliza el prefijo `#`.

Por ejemplo, el siguiente elemento `div` de HTML...

```
<div id="ejemploID">  
    <p>Ejemplo</p>  
</div>
```

...puede seleccionarse mediante `#ejemploID` en CSS, como se muestra a continuación:

```
#ejemploID {  
    width: 20px;  
}
```

Nota: Las especificaciones HTML no permiten múltiples elementos con el mismo ID

Sección 4.13: Cómo aplicar estilo a un input Range

HTML

```
<input type="range"></input>
```

CSS

Efecto	Pseudoselector
Thumb	<code>input[type=range]::-webkit-slider-thumb, input[type=range]::-moz-range-thumb, input[type=range]::-ms-thumb</code>
Track	<code>input[type=range]::-webkit-slider-runnable-track, input[type=range]::-moz-range-track, input[type=range]::-ms-track</code>
OnFocus	<code>input[type=range]:focus</code>
Parte inferior	<code>input[type=range]::-moz-range-progress, input[type=range]::-ms-fill-lower</code> (actualmente no es posible en navegadores WebKit actualmente - se necesita JS)

Sección 4.14: Ejemplo de selector pseudoclase :only-child

La pseudoclase CSS :only-child representa cualquier elemento que sea el único hijo de su padre.

HTML

```
<div>
  <p> Este párrafo es el único hijo del div, tendrá el color azul </p>
</div>
<div>
  <p> Este párrafo es uno de los dos hijos del div </p>
  <p> Este párrafo es uno de los dos hijos de su padre </p>
</div>
```

CSS

```
p:only-child {
  color: blue;
}
```

El ejemplo anterior selecciona el elemento `<p>` que es el único hijo de su padre, en este caso un `<div>`.

[Demostración en JSBin](#)

Capítulo 5: Background

Con CSS puedes establecer colores, degradados e imágenes como fondo de un elemento.

Es posible especificar varias combinaciones de imágenes, colores y degradados, así como ajustar el tamaño, la posición y la repetición (entre otros) de los mismos.

Sección 5.1: Color del fondo

La propiedad `background-color` establece el color de fondo de un elemento utilizando un valor de color o mediante palabras clave como `transparent`, `inherit` o `initial`.

- **transparent**, especifica que el color de fondo debe ser transparente. Esto es por defecto.
- **inherit**, hereda esta propiedad de su elemento padre.
- **initial**, establece esta propiedad a su valor por defecto.

Puede aplicarse a todos los elementos y a los pseudoelementos `::first-letter/::first-line`.

Los colores en CSS pueden especificarse por diferentes métodos.

Nombres de los colores

CSS

```
div {  
    background-color: red; /* rojo */  
}
```

HTML

```
<div> Esto tendrá un fondo rojo </div>
```

El ejemplo anterior es una de las muchas formas que tiene CSS de representar un solo color.

Códigos hexadecimales de color

El código hexadecimal se utiliza para denotar los componentes RGB de un color en notación hexadecimal de base-16. `#ff0000`, por ejemplo, es rojo brillante, donde el componente rojo del color es 256 bits (ff) y las correspondientes porciones verde y azul de del color es 0 (00). Si los dos valores de cada uno de los tres emparejamientos RGB (R, G y B) son iguales, entonces el código de color se puede acortar en tres caracteres (el primer dígito de cada emparejamiento). `#ff0000` se puede acortar a `#f00`, y `#ffffff` se puede acortado a `#fff`.

La notación hexadecimal no distingue entre mayúsculas y minúsculas.

```
body {  
    background-color: #de1205; /* rojo */  
}  
.main {  
    background-color: #00f; /* azul */  
}
```

RGB / RGBA

Otra forma de declarar un color es utilizar RGB o RGBA.

RGB son las siglas de Red (rojo), Green (verde) y Blue (azul), y requiere tres valores separados entre 0 y 255, puestos entre paréntesis, que se corresponden con los valores decimales de color para el rojo, el verde y el azul, respectivamente.

RGBA permite añadir un parámetro alfa adicional entre 0,0 y 1,0 para definir la opacidad.

```

header {
    background-color: rgb(0, 0, 0); /* negro */
}
footer {
    background-color: rgba(0, 0, 0, 0.5); /* negro con 50% de opacidad */
}

```

HSL / HSLa

Otra forma de declarar un color es utilizar HSL o HSLa y es similar a RGB y RGBA.

HSL son las siglas de Hue (tono), Saturation (saturación) y Lightness (luminosidad), y también suele denominarse HLS:

- El matiz es un grado de la rueda cromática (de 0 a 360).
- La saturación es un porcentaje entre 0% y 100%.
- La luminosidad también es un porcentaje entre 0% y 100%.

HSLa permite añadir un parámetro alfa adicional entre 0,0 y 1,0 para definir la opacidad.

```

li a {
    background-color: hsl(120, 100%, 50%); /* verde */
}
#p1 {
    background-color: hsla(120, 100%, 50%, .3); /* verde con 30% de opacidad */
}

```

Interacción con background-image

Las siguientes afirmaciones son todas equivalentes:

```

body {
    background: red;
    background-image: url(parcialmentetransparente.png);
}
body {
    background-color: red;
    background-image: url(parcialmentetransparente.png);
}
body {
    background-image: url(parcialmentetransparente.png);
    background-color: red;
}
body {
    background: red url(parcialmentetransparente.png);
}

```

Todos ellos harán que el color rojo se muestre debajo de la imagen, que las partes de la imagen sean transparentes o que la imagen no se muestre (quizás como resultado de `background-repeat`).

Tenga en cuenta que lo siguiente no es equivalente:

```

body {
    background-image: url(parcialmentetransparente.png);
    background: red;
}

```

Aquí, el valor de `background` anula su `background-image`.

Para obtener más información sobre la propiedad `background`, consulta Abreviatura `background`.

Sección 5.2: Gradientes como fondo

Los degradados son nuevos tipos de imagen, añadidos en CSS3. Como una imagen, los degradados se establecen con la propiedad `background-image` o la abreviatura `background`.

Existen dos tipos de funciones de gradiente: lineal y radial. Cada tipo tiene una variante no repetitiva y una variante repetitiva:

- `linear-gradient()`
- `repeating-linear-gradient()`
- `radial-gradient()`
- `repeating-radial-gradient()`

linear-gradient()

Un `linear-gradient` tiene la siguiente sintaxis

background: `linear-gradient(<dirección>?, <color-1>, <color-2>, ...);`

Valor

<dirección>

Significado

Puede ser un argumento como `to top`, `to bottom`, `to right` o `to left`; o un **ángulo** como `0deg`, `90deg`... . El ángulo empieza desde arriba y gira en el sentido de las agujas del reloj. Puede especificarse en **deg** (grados), **grad** (gradianes), **rad** (radianes), o **turn** (giros). Si se omite, el gradiente fluye de arriba a abajo

<lista-de-colores>

Lista de colores, opcionalmente seguido cada uno por un porcentaje o **longitud** para mostrarlo. Por ejemplo, `yellow 10%`, `rgba(0,0,0,.5) 40px`, `#fff 100%`...

Por ejemplo, esto crea un degradado lineal que empieza por la derecha y pasa del rojo al azul.

```
.gradiente-lineal {  
  background: linear-gradient(to left, red, blue); /* también se puede utilizar 270deg */  
}
```

Puede crear un gradiente **diagonal** declarando una posición inicial horizontal y otra vertical.

```
.gradiente-lineal-diagonal {  
  background: linear-gradient(to left top, red, yellow 10%);  
}
```

Es posible especificar cualquier número de paradas de color en un degradado separándolas con comas. Los siguientes ejemplos crearán un degradado con 8 paradas de color.

```
.gradiente-lineal-arcoiris {  
  background: linear-gradient(to left, red, orange, yellow, green, blue, indigo, violet)  
}
```

radial-gradient()

```
.gradiente-radial-simple {  
  background: radial-gradient(red, blue);  
}  
.gradiente-radial {  
  background: radial-gradient(circle farthest-corner at top left, red, blue);  
}
```

Valor

circle

Significado

Forma del gradiente. Los valores son **circle** o **ellipse**, por defecto es **ellipse**.

farthest-corner

Palabras clave que describen el tamaño de la forma final. Los valores son: **closest-side**, **farthest-side**, **closest-corner**, **farthest-corner**.

top left

Establece la posición del centro del degradado, del mismo modo que **background-position**.

Gradientes repetidos

Las funciones de gradiente de repetición toman los mismos argumentos que los ejemplos anteriores, pero mosaico el gradiente a través del fondo del elemento.

```
.diana {  
    background: repeating-radial-gradient(red, red 10%, white 10%, white 20%);  
}  
.advertencia {  
    background: repeating-linear-gradient(-45deg, yellow, yellow 10%, black 10%, black 20% );  
}
```

Valor	Significado
<code>-45deg</code>	En ángulos . El ángulo empieza desde arriba y gira en el sentido de las agujas del reloj. Puede especificarse en deg (grados), grad (gradianes), rad (radianes), o turn (giros).
<code>to left</code>	Dirección del gradiente, por defecto es <code>to bottom</code> . Sintaxis: <code>to [y-axis(top 0 bottom)] [x-axis(left 0 right)]</code> es decir, <code>to top right</code> .
<code>yellow 10%</code>	Color, opcionalmente seguido de un porcentaje o longitud para mostrarlo. Se repite dos o más veces.

Tenga en cuenta que se pueden utilizar códigos de color HEX, RGB, RGBA, HSL y HSLa en lugar de nombres de color. Los nombres de los colores se usan con fines ilustrativos. Observe también que la sintaxis del degradado radial es mucho más compleja que la del degradado lineal, y aquí se muestra una versión simplificada. Para una explicación completa y especificaciones, consulte [Documentación MDN](#).

Sección 5.3: Imagen como fondo

La propiedad `background-image` se utiliza para especificar una imagen de fondo que se aplicará a todos los elementos coincidentes. Por defecto, esta imagen se coloca en mosaico para cubrir todo el elemento, excluyendo el margen.

```
.miClase {  
    background-image: url('/ruta/hacia/imagen.jpg');  
}
```

Para utilizar varias imágenes como `background-image`, defina `url()` separadas por comas.

```
.miClase {  
    background-image: url('/ruta/hacia/imagen.jpg'),  
                    url('/ruta/hacia/imagen2.jpg');  
}
```

Las imágenes se apilarán según su orden, con la primera imagen declarada encima de las demás y así sucesivamente.

Valor	Resultado
<code>url('/ruta/hacia/imagen.jpg')</code>	Especifica la(s) ruta(s) de la imagen de fondo o un recurso de imagen especificado con el esquema URI (pueden omitirse los apóstrofes), separe los múltiples comas
<code>none</code>	Ninguna imagen como fondo
<code>initial</code>	Valor por defecto
<code>inherit</code>	Hereda el valor del padre

Más CSS para la imagen de fondo

Los siguientes atributos son muy útiles y casi esenciales.

```
background-size: xpx ypx | x% y%;  
background-repeat: no-repeat | repeat | repeat-x | repeat-y;  
background-position: left offset (px/%) right offset (px/%) | center center | left top | right bottom;
```

Sección 5.4: Abreviatura background

La propiedad `background` puede utilizarse para establecer una o más propiedades relacionadas con el fondo:

Valor	Descripción	Versión CSS
<code>background-image</code>	Imagen de fondo a utilizar	1+
<code>background-color</code>	Color de fondo a aplicar	1+
<code>background-position</code>	Posición de la imagen de fondo	1+
<code>background-size</code>	Tamaño de la imagen de fondo	3+
<code>background-repeat</code>	Repetir la imagen de fondo	1+
<code>background-origin</code>	Cómo se posiciona el fondo (ignorado cuando <code>background-attachment</code> es <code>fixed</code>)	3+
<code>background-clip</code>	Cómo se pinta el fondo en relación con la <code>content-box</code> , <code>border-box</code> o <code>padding-box</code> .	3+
<code>background-attachment</code>	Cómo se comporta la imagen de fondo, si se desplaza junto con el bloque que la contiene o si tiene una posición fija dentro de la ventana gráfica.	1+
<code>initial</code>	Establece el valor por defecto de la propiedad	3+
<code>inherit</code>	Hereda el valor de la propiedad del padre	2+

El orden de los valores no importa y cada valor es opcional.

Sintaxis

La sintaxis de la declaración abreviada de fondo es:

```
background: [<background-image>] [<background-color>] [<background-position>]/[<background-size>] [<background-repeat>] [<background-origin>] [<background-clip>] [<background-attachment>] [<initial|inherit>];
```

Ejemplos

```
background: red;
```

Simplemente estableciendo un `background-color` con el valor `red`.

```
background: border-box red;
```

Establecer un `background-clip` a `border-box` y un `background-color` a rojo.

```
background: no-repeat center url("algunpng.jpg");
```

Establece `background-repeat` a `no-repeat`, `background-origin` a `center` y `background-image` una imagen.

```
background: url('patron.png') green;
```

En este ejemplo, el `background-color` del elemento se establecería en `green` con `patron.png`, si está disponible, superpuesto sobre el color, repitiéndose tantas veces como sea necesario para llenar el elemento. Si `patron.png` incluye alguna transparencia, el color `green` será visible detrás de ella.

```
background: #000000 url("imagen.png") top left / 600px auto no-repeat;
```

En este ejemplo tenemos un fondo negro con una imagen 'imagen.png' encima, la imagen no se repite en ninguno de los dos ejes y está posicionada en la esquina superior izquierda. El `/` después de la posición es para poder incluir el tamaño de la imagen de fondo que en este caso se establece como `600px` de ancho y `auto` para la altura. Este ejemplo podría funcionar bien con una imagen característica que pueda desvanecerse en un color sólido.

NOTA: El uso de la propiedad abreviada `background` restablece todos los valores de la propiedad `background` establecidos previamente, incluso si no se indica un valor. Si sólo desea modificar el valor de una propiedad de fondo previamente establecida, utiliza en su lugar una propiedad larga y no la abreviada.

Sección 5.5: Tamaño del fondo

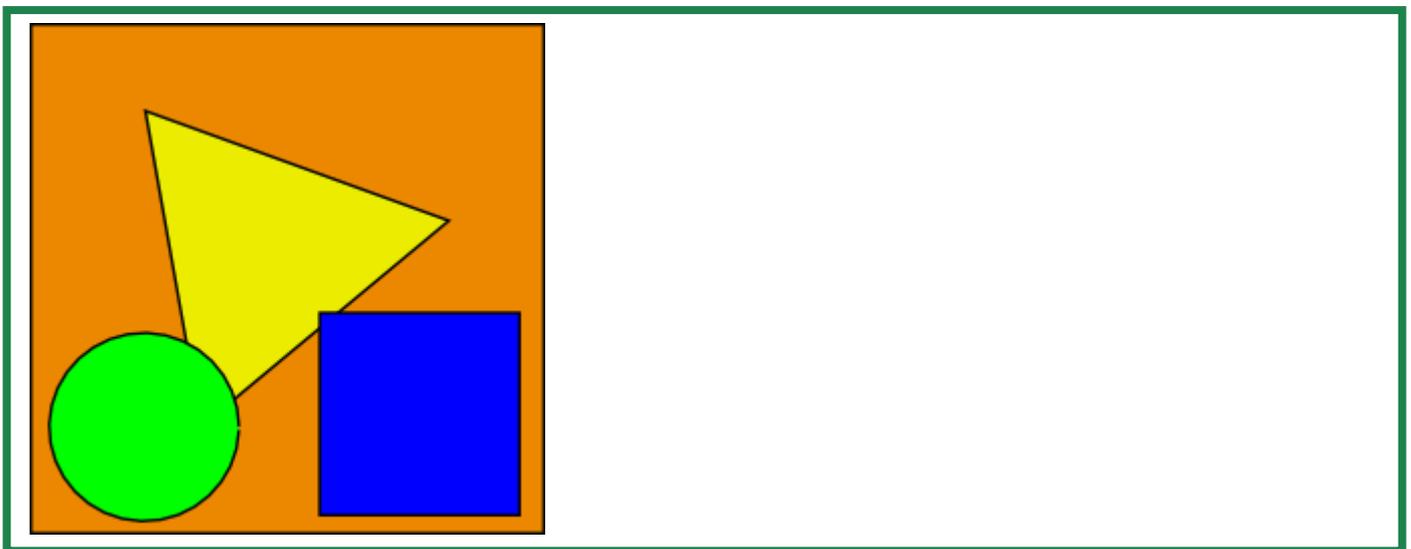
Visión general

La propiedad `background-size` permite controlar la escala del `background-image`. Toma hasta dos valores, que determinan la escala/tamaño de la imagen resultante en dirección vertical y horizontal. Si la propiedad falta, se considerará `auto` tanto en `width` (anchura) como en `height` (altura).

`auto` mantendrá la relación de aspecto de la imagen, si se puede determinar. La altura es opcional y puede considerarse `auto`. Por lo tanto, en una imagen de 256px x 256px, todos los ajustes de `background-size` siguientes producirían una imagen con altura y anchura de 50px:

```
background-size: 50px;  
background-size: 50px auto; /* igual que arriba */  
background-size: auto 50px;  
background-size: 50px 50px;
```

Así, si partimos de la siguiente imagen (que tiene el tamaño mencionado de 256px × 256px),

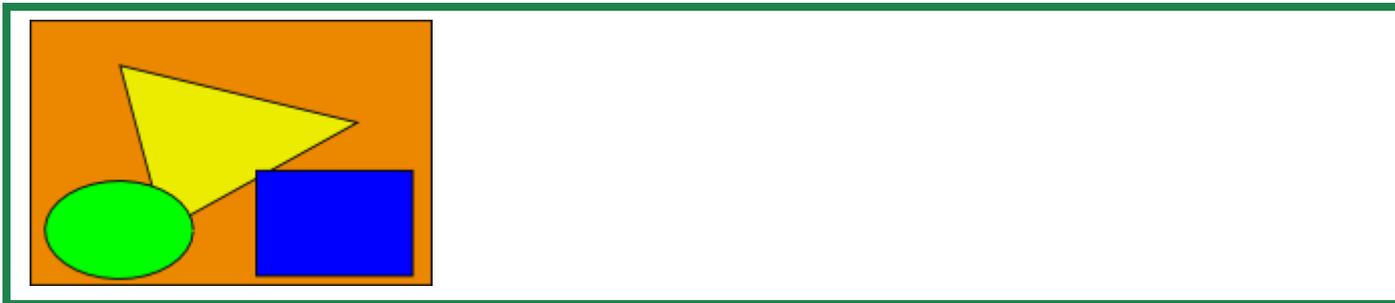


acabaremos con un 50px × 50px en la pantalla del usuario, contenido en el fondo de nuestro elemento:



También se pueden utilizar valores porcentuales para escalar la imagen con respecto al elemento. El siguiente ejemplo una imagen dibujada de 200px × 133px:

```
#confondo {  
  background-image: url(hacia/algun/fondo.png);  
  background-size: 100% 66%;  
  width: 200px;  
  height: 200px;  
  padding: 0;  
  margin: 0;  
}
```



El comportamiento depende del `background-origin`.

Mantener la relación de aspecto

El último ejemplo de la sección anterior perdió su relación de aspecto original. El círculo se convirtió en una elipse, el cuadrado en un rectángulo, el triángulo en otro triángulo.

El enfoque de la longitud o el porcentaje no es lo suficientemente flexible como para mantener la relación de aspecto en todo momento. `auto` no ayuda, ya que es posible que no sepas qué dimensión de tu elemento será mayor. Sin embargo, para cubrir ciertas áreas con una imagen (y la relación de aspecto correcta) completamente o para contener una imagen con la relación de aspecto correcta completamente en un área de fondo, los valores `contain` y `cover` proporcionan la funcionalidad adicional.

Eggspanación para `contain` y `cover`

Perdón por el mal juego de palabras, pero vamos a utilizar una [foto del día de Biswarup Ganguly](#) para la demostración. Digamos que esta es tu pantalla, y el área gris está fuera de tu pantalla visible. Para la demostración, vamos a suponer una relación de 16 x 9.



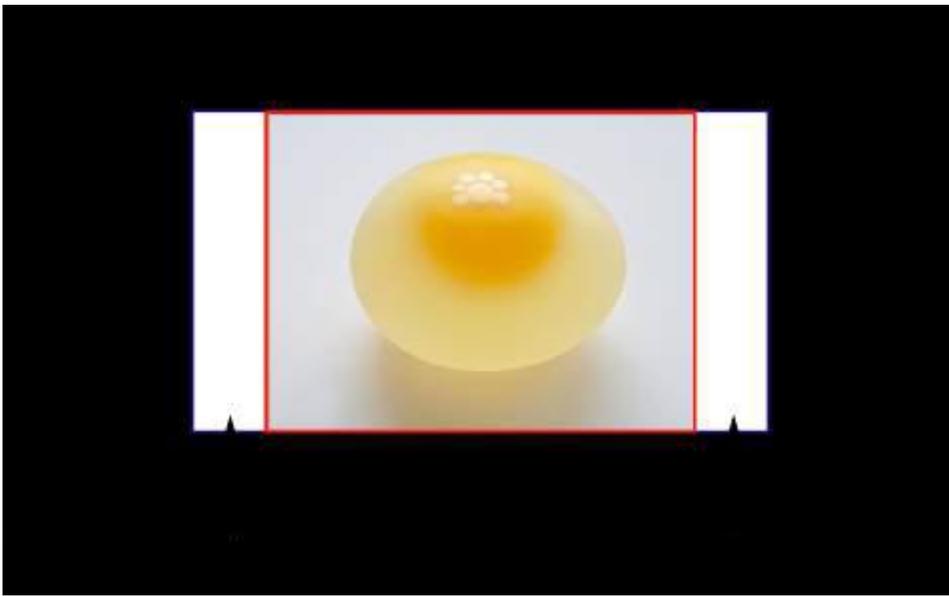
Queremos utilizar la mencionada foto del día como fondo. Sin embargo, recortamos la imagen a 4x3 por alguna razón. Podríamos establecer la propiedad `background-size` a alguna longitud fija, pero nos centraremos en `contain` y `cover`. Ten en cuenta que también asumo que no hemos manipulado la anchura y/o la altura del `body`.

`contain`

`contain`

Escala la imagen, conservando su relación de aspecto intrínseca (si la tiene), al tamaño más grande que permita que tanto su anchura como su altura quepan dentro del área de posicionamiento del fondo.

Esto asegura que la imagen de fondo está siempre completamente contenida en el área de posicionamiento de fondo, sin embargo, podría haber algún espacio vacío lleno de su `background-color` en este caso:

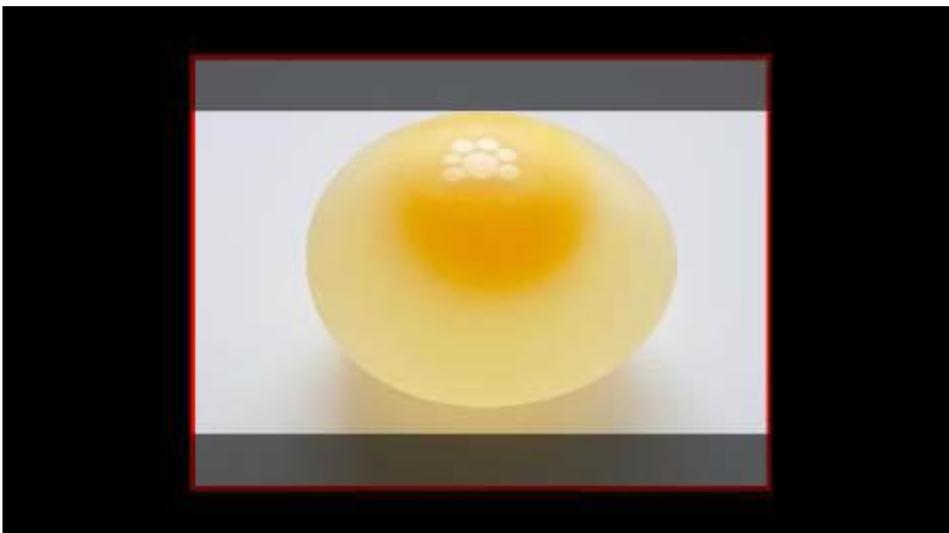


`cover`

`cover`

Escale la imagen, conservando su relación de aspecto intrínseca (si la tiene), al tamaño más pequeño tal que tanto su anchura como su altura puedan cubrir completamente el área de posicionamiento del fondo.

Esto asegura que la imagen de fondo cubra todo. No habrá un `background-color` visible, pero dependiendo de la proporción de la pantalla, una gran parte de la imagen podría quedar cortada:



Demostración con código actual

```
div > div {
  background-image: url(http://i.stack.imgur.com/r5CAq.jpg);
  background-repeat: no-repeat;
  background-position: center center;
  background-color: #ccc;
  border: 1px solid;
  width: 20em;
  height: 10em;
}
div.contain {
  background-size: contain;
}
div.cover {
  background-size: cover;
}
/*****
Estilos adicionales para los cuadros de explicación
*****/
div > div {
  margin: 0 1ex 1ex 0;
  float: left;
}
div + div {
  clear: both;
  border-top: 1px dashed silver;
  padding-top: 1ex;
}
div > div::after {
  background-color: #000;
  color: #fefefe;
  margin: 1ex;
  padding: 1ex;
  opacity: 0.8;
  display: block;
  width: 10ex;
  font-size: 0.7em;
  content: attr(class);
}
<div>
  <div class="contain"></div>
  <p> Fíjese en el fondo gris. La imagen no cubre toda la región, pero está totalmente
  <em>contenida</em>.</p>
</div>
<div>
  <div class="cover"></div>
  <p> Fíjate en los patos/gansos de la parte inferior de la imagen. La mayor parte del agua
  está cortada, así como una parte del cielo. Ya no se ve la imagen completa, pero tampoco se ve
  ningún color de fondo; la imagen <em>cubre</em> todo el <code>&lt;div&gt;</code>.</p>
</div>
```



Fíjese en el fondo gris. La imagen no cubre toda la región, pero está totalmente *contenida*.



Fíjate en los patos/gansos de la parte inferior de la imagen. La mayor parte del agua está cortada, así como una parte del cielo. Ya no se ve la imagen completa, pero tampoco se ve ningún color de fondo; la imagen *cubre* todo el <div>.

Sección 5.6: Posición de fondo

La propiedad `background-position` se utiliza para especificar la posición inicial de una imagen de fondo o un degradado.

```
.miClase {
  background-image: url('ruta/hacia/imagen.jpg');
  background-position: 50% 50%;
}
```

La posición se establece utilizando una coordenada **X** e **Y** y puede establecerse utilizando cualquiera de las unidades utilizadas en CSS.

Unidad	Descripción
<code>valor% valor%</code>	Un porcentaje para el desplazamiento horizontal es relativo a (<i>anchura del área de posicionamiento del fondo - anchura de la imagen de fondo</i>). Un porcentaje para el desplazamiento vertical es relativo a (<i>altura del área de posicionamiento del fondo - altura de la imagen de fondo</i>). El tamaño de la imagen es el tamaño dado por <code>background-size</code> .
<code>valorpx valorpx</code>	Desplaza la imagen de fondo una longitud dada en píxeles con respecto a la parte superior izquierda del fondo. área de posicionamiento

Las unidades en CSS pueden especificarse mediante diferentes métodos (véase aquí).

Propiedades largas de la posición de fondo

Además de la propiedad abreviada anterior, también se pueden utilizar las propiedades largas de `background-position-x` y `background-position-y`. Permiten controlar las posiciones x o y por separado.

NOTA: Esto es compatible con todos los navegadores excepto Firefox (versiones 31-48) ². Firefox 49, que se lanzará en septiembre de 2016, *soportará* estas propiedades. Hasta entonces, [hay un hack de Firefox dentro de este Stack Overflow de respuesta](#).

Sección 5.7: La propiedad `background-origin`

La propiedad `background-origin` especifica dónde se posiciona la imagen de fondo.

Nota: Si la propiedad `background-attachment` está establecida en `fixed`, esta propiedad no tiene efecto.

Valor por defecto: `padding-box`

Valores posibles:

- `padding-box` - La posición es relativa al cuadro de relleno
- `border-box` - La posición es relativa al cuadro del borde
- `content-box` - La posición es relativa a la caja de contenido
- `initial`
- `inherit`

CSS

```
.ejemplo {
  width: 300px;
  border: 20px solid black;
  padding: 50px;
  background: url(https://static.pexels.com/photos/6440/magazines-desk-work-workspace-medium.jpg);
  background-repeat: no-repeat;
}
.ejemplo1 {}
.ejemplo2 { background-origin: border-box; }
.ejemplo3 { background-origin: content-box; }
```

HTML

`<p>` Sin `background-origin` (`padding-box` por defecto): `</p>`

```
<div class="ejemplo ejemplo1">
```

```
<h2>Lorem Ipsum Dolor</h2>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. </p>
```

```
<p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. </p>
```

```
</div>
```

```
<p>background-origin: border-box:</p>
```

```
<div class="ejemplo ejemplo2">
```

```
<h2>Lorem Ipsum Dolor</h2>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. </p>
```

```
<p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. </p>
```

```
</div>
```

```
<p>background-origin: content-box:</p>
```

```
<div class="ejemplo ejemplo3">
```

```
<h2>Lorem Ipsum Dolor</h2>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. </p>
```

```
<p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. </p>
```

```
</div>
```

Resultado

Sin background-origin (padding-box por defecto):



background-origin: border-box:



background-origin: content-box:



Mas:

<https://www.w3.org/TR/css-backgrounds-3/#the-background-origin>

<https://developer.mozilla.org/es/docs/Web/CSS/background-origin>

Sección 5.8: Imagen de fondo múltiple

En CSS3, podemos apilar varios fondos en el mismo elemento.

```
#midiv {
  background-image: url(img_1.png), /* imagen superior */
                  url(img_2.png), /* imagen central */
                  url(img_3.png); /* imagen inferior */
  background-position: right bottom,
                    left top,
                    right top;
  background-repeat: no-repeat,
                  repeat,
                  no-repeat;
}
```

Las imágenes se apilarán unas sobre otras con el primer fondo en la parte superior y el último fondo en la parte posterior. `img_1` estará en la parte superior, el `img_2` y `img_3` está en la parte inferior.

Para ello, también podemos utilizar la propiedad abreviada `background`:

```
#midiv {
  background: url(img_1.png) right bottom no-repeat,
            url(img_2.png) left top repeat,
            url(img_3.png) right top no-repeat;
}
```

También podemos apilar imágenes y gradientes:

```
#midiv {
  background: url(image.png) right bottom no-repeat,
            linear-gradient(to bottom, #fff 0%, #000 100%);
}
```

Sección 5.9: background-attachment

La propiedad `background-attachment` establece si una imagen de fondo es fija o se desplaza con el resto de la página.

```
body {
  background-image: url('img.jpg');
  background-attachment: fixed;
}
```

Valor	Descripción
<code>scroll</code>	El fondo se desplaza junto con el elemento. Esto es por defecto.
<code>fixed</code>	El fondo se fija con respecto a la ventana gráfica.
<code>local</code>	El fondo se desplaza junto con el contenido del elemento.
<code>initial</code>	Establece esta propiedad a su valor por defecto.
<code>inherit</code>	Hereda esta propiedad de su elemento padre.

Ejemplos

background-attachment: scroll

El comportamiento por defecto, cuando el cuerpo se desplaza el fondo se desplaza con él:

```
body {
  background-image: url('imagen.jpg');
  background-attachment: scroll;
}
```

background-attachment: fixed

La imagen de fondo será fija y no se moverá cuando se desplace el cuerpo:

```
body {  
    background-image: url('imagen.jpg');  
    background-attachment: fixed;  
}
```

background-attachment: local

La imagen de fondo del div se desplazará cuando se desplace el contenido del div.

```
div {  
    background-image: url('image.jpg');  
    background-attachment: local;  
}
```

Sección 5.10: background-clip

Definición y uso: La propiedad `background-clip` especifica el área de pintura del fondo.

Valor por defecto: `border-box`

Valores

- `border-box` es el valor por defecto. Esto permite que el fondo se extienda hasta el borde exterior del elemento.
- `padding-box` recorta el fondo en el borde exterior del relleno del elemento y no deja que se extienda dentro del borde.
- `content-box` recorta el fondo en el borde de la caja de contenido.
- `inherit` aplica la configuración del padre al elemento seleccionado.

CSS

```
.ejemplo {  
    width: 300px;  
    border: 20px solid black;  
    padding: 50px;  
    background: url(https://static.pexels.com/photos/6440/magazines-desk-work-workspace-medium.jpg);  
    background-repeat: no-repeat;  
}  
.ejemplo1 {}  
.ejemplo2 { background-origin: border-box; }  
.ejemplo3 { background-origin: content-box; }
```

HTML

```
<p>Sin background-origin (padding-box por defecto): </p>
<div class="ejemplo ejemplo1">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod
  tincidunt ut laoreet dolore magna aliquam erat volutpat. </p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
  nisl ut aliquip ex ea commodo consequat. </p>
</div>
<p>background-origin: border-box:</p>
<div class="ejemplo ejemplo2">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod
  tincidunt ut laoreet dolore magna aliquam erat volutpat. </p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
  nisl ut aliquip ex ea commodo consequat. </p>
</div>
<p>background-origin: content-box:</p>
<div class="ejemplo ejemplo3">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod
  tincidunt ut laoreet dolore magna aliquam erat volutpat. </p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
  nisl ut aliquip ex ea commodo consequat. </p>
</div>
```

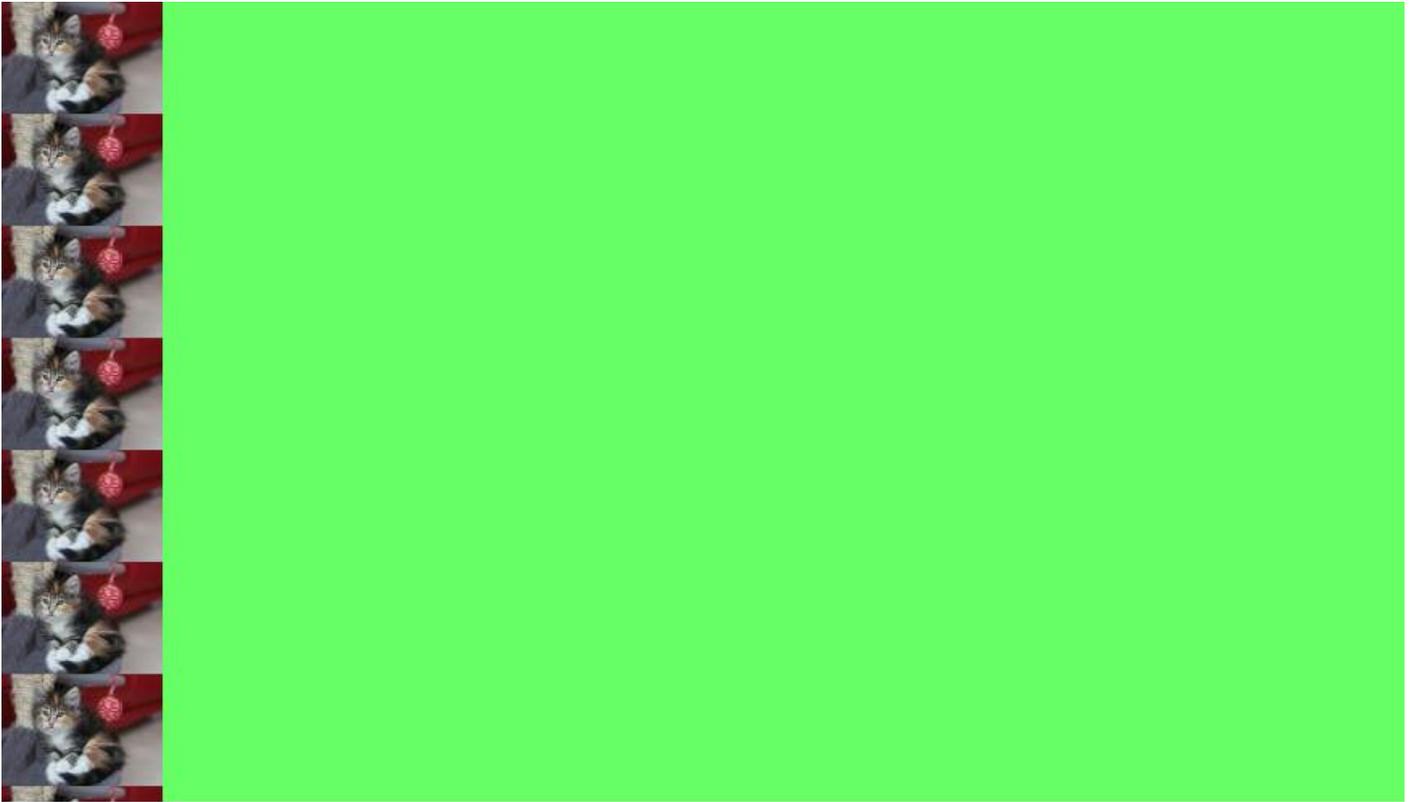
Sección 5.11: background-repeat

La propiedad `background-repeat` establece si/cómo se repetirá una imagen de fondo.

Por defecto, una imagen de fondo se repite tanto vertical como horizontalmente.

```
div {
  background-image: url("img.jpg");
  background-repeat: repeat-y;
}
```

Así se ve un `background-repeat: repeat-y`:

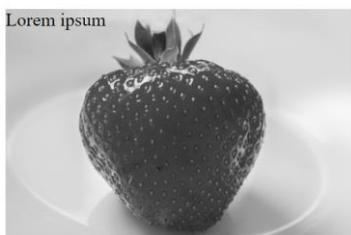


Sección 5.12: La propiedad background-blend-mode

```
.mi-div {  
  width: 300px;  
  height: 200px;  
  background-size: 100%;  
  background-repeat: no-repeat;  
  background-image: linear-gradient(to right, black 0%,white 100%),  
  url('https://static.pexels.com/photos/54624/strawberry-fruit-red-sweet-54624-medium.jpeg');  
  background-blend-mode: saturation;  
}
```

<div class="mi-div">Lorem ipsum</div>

Resultado



Sintaxis CSS: background-blend-mode: normal | multiply | screen | overlay | darken | lighten | color-dodge | saturation | color | luminosity;

Sección 5.13: Color de fondo con opacidad

Si fija la `opacity` en un elemento, afectará a todos sus elementos hijos. Para establecer una opacidad sólo en el fondo de un tendrá que utilizar colores RGBA. El siguiente ejemplo tendrá un fondo negro con una opacidad de 0,6.

```
/* Fallback para navegadores que no soportan RGBA */  
background-color: rgb(0, 0, 0);  
/* RGBA con 0,6 de opacidad */  
background-color: rgba(0, 0, 0, 0.6);  
/* Para IE 5.5 - 7*/  
filter: progid:DXImageTransform.Microsoft.gradient(startColorstr=#99000000,  
endColorstr=#99000000);  
/* Para IE 8*/  
-ms-filter: "progid:DXImageTransform.Microsoft.gradient(startColorstr=#99000000,  
endColorstr=#99000000)";
```

Capítulo 6: Centrado

Sección 6.1: Uso de Flexbox

HTML

```
<div class="container">  
    
</div>
```

CSS

```
html, body, .container {  
  height: 100%;  
}  
.container {  
  display: flex;  
  justify-content: center; /* centro horizontal */  
}  
img {  
  align-self: center; /* centro vertical */  
}
```

Resultado



HTML

```

```

CSS

```
html, body {  
    height: 100%;  
}  
body {  
    display: flex;  
    justify-content: center; /* centro horizontal */  
    align-items: center; /* centro vertical */  
}
```

Resultado



Consulte Centrado dinámico vertical y horizontal en la documentación de Flexbox para obtener más detalles sobre flexbox y qué significan estos estilos.

Compatibilidad con navegadores

Flexbox es compatible con los principales navegadores, [excepto las versiones de IE anteriores a la 10](#).

Algunas versiones recientes de navegadores, como Safari 8 e IE10, requieren [prefijos de proveedor](#).

Para generar prefijos de forma rápida existe [Autoprefixer](#), una herramienta de terceros.

Para navegadores más antiguos (como IE 8 y 9) [existe un Polyfill](#).

Para obtener información más detallada sobre la compatibilidad de flexbox con los navegadores, consulte [esta respuesta](#).

Sección 6.2: Uso de transform

Las transformaciones CSS se basan en el tamaño de los elementos, por lo que, si no sabes lo alto o ancho que es tu elemento, puedes posicionarlo absolutamente al 50% de la parte superior e izquierda de un contenedor relativo y trasladarlo un 50% hacia la izquierda y hacia arriba para centrarlo vertical y horizontalmente.

Tenga en cuenta que, con esta técnica, el elemento podría terminar siendo renderizado en un límite de píxeles no entero, haciendo que se vea borroso. Vea [esta respuesta en SO](#) para una solución.

HTML

```
<div class="container">
  <div class="elemento"></div>
</div>
```

CSS

```
.container {
  position: relative;
}
.elemento {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
```

Compatibilidad entre navegadores

La propiedad transform necesita prefijos para ser soportada por navegadores antiguos. Se necesitan prefijos para Chrome<=35, Safari<=8, Opera<=22, Android Browser<=4.4.4 e IE9. Las transformaciones CSS no son compatibles con IE8 y versiones anteriores.

A continuación, se muestra una declaración de transformación común para el ejemplo anterior:

```
-webkit-transform: translate(-50%, -50%); /* Chrome, Safari, Opera, Android */
-ms-transform: translate(-50%, -50%); /* IE 9 */
transform: translate(-50%, -50%);
```

Para más información, vea [CanIUse](#).

Más información

El elemento se posiciona según el primer padre no estático (**position**: `relative`, `absolute` o `fixed`).

Para centrar sólo horizontalmente, utilice **left**: `50%` y **transform**: `translateX(-50%)`. Lo mismo ocurre con vertical-only con **top**: `50%` y **transform**: `translateY(-50%)`.

El uso de elementos de anchura/altura no estáticos con este método de centrado puede hacer que el elemento centrado aparezca aplastado. Esto ocurre sobre todo con elementos que contienen texto, y puede solucionarse añadiendo: **margin-right**: `-50%`; y **margin-bottom**: `-50%`;

Sección 6.3: Uso de margin: 0 auto;

Los objetos se pueden centrar utilizando `margin: 0 auto`; si son elementos de bloque y tienen una anchura definida.

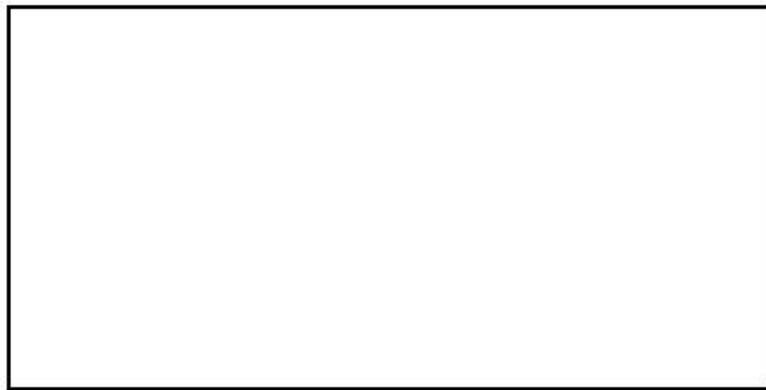
HTML

```
<div class="contenedorDiv">
  <div id="divCentrado"></div>
</div>
<div class="contenedorDiv">
  <p id="parrafoCentrado">Este es un párrafo centrado. </p>
</div>
<div class="contenedorDiv">
  
</div>
```

CSS

```
body {
  background: lightgrey;
}
.contenedorDiv {
  width: 100%;
  height: 100px;
  padding-bottom: 40px;
}
#divCentrado {
  margin: 0 auto;
  width: 200px;
  height: 100px;
  border: 1px solid #000;
}
#parrafoCentrado {
  width: 200px;
  margin: 0 auto;
}
#imagenCentrado {
  display: block;
  width: 200px;
  margin: 0 auto;
}
```

Resultado



Este es un párrafo centrado.



Sección 6.4: Uso de la alineación de texto

El tipo de centrado más común y sencillo es el de las líneas de texto de un elemento. CSS tiene la regla `text-align: center` para este propósito:

HTML

```
<p>Lorem ipsum</p>
```

CSS

```
p {  
    text-align: center;  
}
```

Esto no funciona para centrar elementos de bloque completos. `text-align` sólo controla la alineación del contenido en línea, como el texto en su elemento de bloque padre.

Más información sobre `text-align` en la sección Tipografía.

Sección 6.5: Uso de `position: absolute`

Funciona en navegadores antiguos (IE >= 8)

Los márgenes automáticos, junto con valores de cero para los desplazamientos `left` y `right` o `top` y `bottom`, centrarán un elemento de posición absoluta dentro de su padre.

HTML

```
<div class="padre">  
      
</div>
```

CSS

```
.padre {  
    position: relative;  
    height: 500px;  
}  
.centrar {  
    position: absolute;  
    margin: auto;  
    top: 0;  
    right: 0;  
    bottom: 0;  
    left: 0;  
}
```

Los elementos que no tienen su propia anchura y altura implícitas, como las imágenes, necesitarán que se definan esos valores.

Otros recursos: [Centrado absoluto en CSS](#)

Sección 6.6: Uso de `calc()`

La función `calc()` es la parte de una nueva sintaxis en CSS3 en la que puedes calcular (matemáticamente) qué tamaño/posición ocupa tu elemento utilizando una variedad de valores como píxeles, porcentajes, etc. Nota: Siempre que utilice esta función, tenga en cuenta el espacio entre dos valores `calc(100% - 80px)`.

CSS

```
.centrar {  
    position: absolute;  
    height: 50px;  
    width: 50px;  
    background: red;  
    top: calc(50% - 50px / 2); /* altura dividida por 2 */  
    left: calc(50% - 50px / 2); /* anchura dividida por 2 */  
}
```

HTML

```
<div class="centrar"></div>
```

Sección 6.7: Uso de line-height

También puede utilizar `line-height` para centrar verticalmente una sola línea de texto dentro de un contenedor:

CSS

```
div {  
    height: 200px;  
    line-height: 200px;  
}
```

Eso es bastante feo, pero puede ser útil dentro de un elemento `<input />`. La propiedad `line-height` sólo funciona cuando el texto a centrar abarca una sola línea. Si el texto se envuelve en varias líneas, la salida resultante no estará centrada.

Sección 6.8: Alinear verticalmente cualquier cosa con 3 líneas de código

[Compatible con IE11+](#)

Utilice estas 3 líneas para alinear verticalmente prácticamente todo. Sólo asegúrese de que el div/imagen al que aplica el código tiene un con una altura.

CSS

```
div.vertical {  
    position: relative;  
    top: 50%;  
    transform: translateY(-50%);  
}
```

HTML

```
<div class="vertical"> ¡Texto alineado verticalmente! </div>
```

Sección 6.9: Centrado en relación con otro elemento

Veremos cómo centrar el contenido en función de la altura de un elemento cercano.

Compatibilidad: IE8+, el resto de navegadores modernos.

HTML

```
<div class="contenido">  
    <div class="posicion-contenido">  
        <div class="thumb">  
              
        </div>  
        <div class="detalles">  
            <p class="banner-titulo">texto 1</p>  
            <p class="banner-texto">contenido contenido contenido contenido contenido  
contenido contenido contenido contenido contenido contenido contenido  
contenido</p>  
            <button class="btn">boton</button>  
        </div>  
    </div>  
</div>
```

CSS

```
.contenido * {
  box-sizing: border-box;
}
.contenido .posicion-contenido {
  display: table;
}
.contenido .detalles {
  display: table-cell;
  vertical-align: middle;
  width: 33.333333%;
  padding: 30px;
  font-size: 17px;
  text-align: center;
}
.contenido .thumb {
  width: 100%;
}
.contenido .thumb img {
  width: 100%;
}
```

Resultado



texto 1

contenido contenido contenido
contenido contenido contenido
contenido contenido contenido
contenido contenido

boton

Los puntos principales son los 3 contenedores `.thumb`, `.detalles` y `.posicion-contenedor`:

- El `.posicion-contenedor` debe tener `display: table`.
- El `.detalles` debe tener el ancho real establecido `width:` y `display: table-cell`, `vertical-align: middle`.
- El `.thumb` debe tener `width: 100%` si quieres que ocupe todo el espacio restante y sea influenciado por el ancho de `.detalles`.
- La imagen (si tiene una imagen) dentro de `.thumb` debe tener `width: 100%`, pero no es necesario si tiene proporciones correctas.

Sección 6.10: Técnica del elemento fantasma (hack de Michał Czernow)

Esta técnica funciona incluso cuando se desconocen las dimensiones del contenedor.

Configure un elemento "fantasma" dentro del contenedor a centrar que tenga una altura del 100% y, a continuación, utiliza `vertical-align: middle` tanto en éste como en el elemento a centrar.

CSS

```
/* Este padre puede tener cualquier anchura y altura */
.bloque {
    text-align: center;
    /* Puede ser conveniente hacerlo si existe el riesgo de que el contenedor sea más estrecho
    que el elemento que contiene. */
    white-space: nowrap;
}
/* El elemento fantasma */
.bloque:before {
    content: '';
    display: inline-block;
    height: 100%;
    vertical-align: middle;
    /* Hay un espacio entre el elemento fantasma y .centrado, causado por el carácter de espacio
    renderizado. Podría eliminarse desplazando .centrado (la distancia de desplazamiento depende
    de la familia de fuentes), o poniendo a cero el tamaño de fuente en .padre y restableciéndolo
    (probablemente a 1rem) en .centrado. */
    margin-right: -0.25em;
}
/* El elemento a centrar, también puede ser de cualquier anchura y altura */
.centrado {
    display: inline-block;
    vertical-align: middle;
    width: 300px;
    white-space: normal; /* Restablecer el comportamiento nowrap heredado */
}
```

HTML

```
<div class="bloque">
    <div class="centrado"></div>
</div>
```

Sección 6.11: Centrado vertical y horizontal sin preocuparse por la altura o la anchura

La siguiente técnica le permite añadir su contenido a un elemento HTML y centrarlo tanto horizontal como verticalmente **sin preocuparse de su altura o anchura**.

El contenedor exterior

- debe tener `display: table;`

El contenedor interior

- debe tener `display: table-cell;`
- debe tener `vertical-align: middle;`
- debe tener `text-align: center;`

El cuadro de contenido

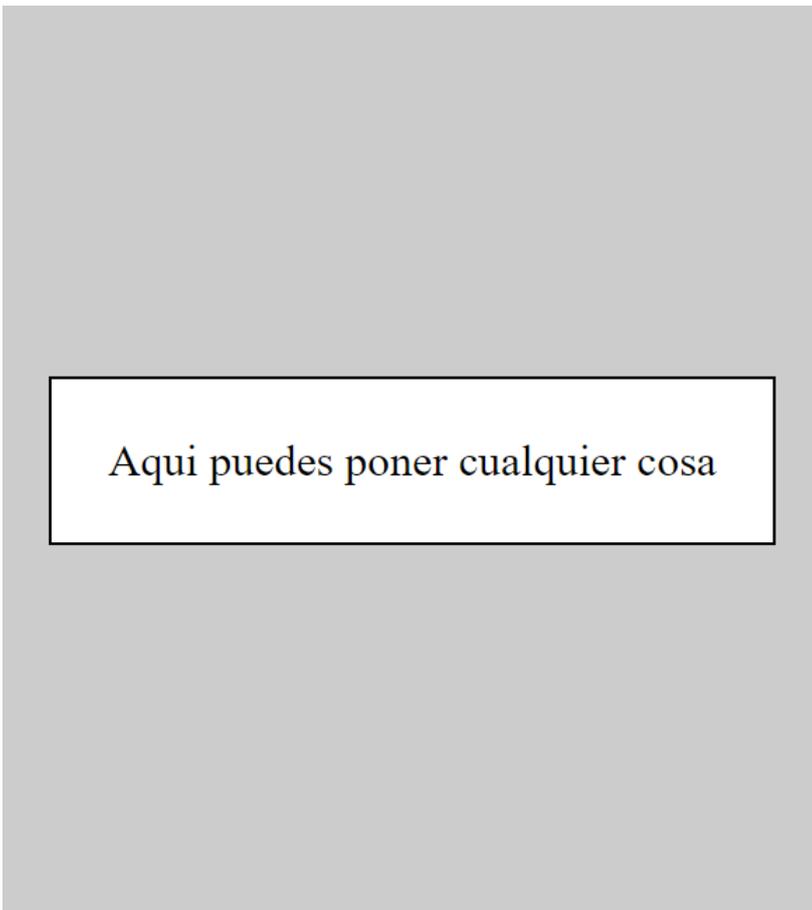
- debe tener `display: inline-block;`
- debe reajustar la alineación horizontal del texto a, por ejemplo, `text-align: left;` o `text-align: right;`, a menos que texto esté centrado.

HTML

```
<div class="contenedor-exterior">
  <div class="contenedor-interior">
    <div class="contenido-centrado">
      Aquí puedes poner cualquier cosa
    </div>
  </div>
</div>
```

CSS

```
body {
  margin : 0;
}
.contenedor-exterior {
  position : absolute;
  display: table;
  width: 100%; /* Puede ser de CUALQUIER anchura */
  height: 100%; /* Esto podría ser CUALQUIER altura */
  background: #ccc;
}
.contenedor-interior {
  display: table-cell;
  vertical-align: middle;
  text-align: center;
}
.contenido-centrado {
  display: inline-block;
  text-align: left;
  background: #fff;
  padding: 20px;
  border: 1px solid #000;
}
```



Sección 6.12: Alinear verticalmente una imagen dentro de div

HTML

```
<div class="wrap">  
    
</div>
```

CSS

```
.wrap {  
  height: 50px; /* altura máxima de la imagen */  
  width: 100px;  
  border: 1px solid blue;  
  text-align: center;  
}  
.wrap:before {  
  content: "";  
  display: inline-block;  
  height: 100%;  
  vertical-align: middle;  
  width: 1px;  
}  
img {  
  vertical-align: middle;  
}
```

Sección 6.13: Centrado con tamaño fijo

Si el tamaño de su contenido es fijo, puede utilizar el posicionamiento absoluto al 50% con un `margin` que reduzca a la mitad la anchura y la altura de su contenido. anchura y altura de su contenido:

HTML

```
<div class="centrar">
  Centrar vertical y horizontalmente
</div>
```

CSS

```
.centrar {
  position: absolute;
  background: #ccc;
  left: 50%;
  width: 150px;
  margin-left: -75px; /* width * -0.5 */
  top: 50%;
  height: 200px;
  margin-top: -100px; /* height * -0.5 */
}
```

Centrado horizontal sólo con anchura fija

Puede centrar el elemento horizontalmente, aunque no conozca la altura del contenido:

HTML

```
<div class="centrar">
  Centrar sólo horizontalmente
</div>
```

CSS

```
.centrar {
  position: absolute;
  background: #ccc;
  left: 50%;
  width: 150px;
  margin-left: -75px; /* anchura * -0,5 */
}
```

Centrado vertical con altura fija

Puede centrar el elemento verticalmente si conoce su altura:

HTML

```
<div class="centrar">
  Centrar sólo verticalmente
</div>
```

CSS

```
.centrar {
  position: absolute;
  background: #ccc;
  top: 50%;
  height: 200px;
  margin-top: -100px; /* width * -0.5 */
}
```

Sección 6.14: Alinear verticalmente elementos de altura dinámica

Aplicar CSS intuitivamente no produce los resultados deseados porque

- `vertical-align: middle` **no es aplicable a los elementos de nivel de bloque**
- `margin-top: auto` y `margin-bottom: auto` los valores utilizados se computarían como **cero**
- `margin-top: -50%` los valores de margen basados en porcentajes se calculan en relación con la **anchura del bloque contenedor**

Para una mayor compatibilidad con los navegadores, una solución con elementos de ayuda:

HTML

```
<div class="vcentrar--contenedor">
  <div class="vcentrar--ayudante">
    <div class="vcentrar--contenido">
      <!-- cosas -->
    </div>
  </div>
</div>
```

CSS

```
.vcentrar--contenido {
  display: table;
  height: 100%;
  position: absolute;
  overflow: hidden;
  width: 100%;
}
.vcentrar--ayudante {
  display: table-cell;
  vertical-align: middle;
}
.vcentrar--contenido {
  margin: 0 auto;
  width: 200px;
}
```

[Pregunta original](#). Este método:

- funciona con elementos dinámicos de altura
- respeta el flujo de los contenidos
- es compatible con los navegadores antiguos

Sección 6.15: Centrado horizontal y vertical mediante diseño de tabla

Se puede centrar fácilmente un elemento hijo utilizando la propiedad `display: table;`.

HTML

```
<div class="wrapper">
  <div class="padre">
    <div class="hijo"></div>
  </div>
</div>
```

CSS

```
.wrapper {  
  display: table;  
  vertical-align: center;  
  width: 200px;  
  height: 200px;  
  background-color: #9e9e9e;  
}  
.padre {  
  display: table-cell;  
  vertical-align: middle;  
  text-align: center;  
}  
.hijo {  
  display: inline-block;  
  vertical-align: middle;  
  text-align: center;  
  width: 100px;  
  height: 100px;  
  background-color: teal;  
}
```

Capítulo 7: El modelo de caja

Parámetro	Detalle
<code>content-box</code>	La anchura y la altura del elemento sólo incluyen el área de contenido.
<code>padding-box</code>	La anchura y la altura del elemento incluyen el contenido y el relleno.
<code>border-box</code>	La anchura y la altura del elemento incluyen el contenido, el relleno y el borde.
<code>initial</code>	Pone el modelo de caja en su estado por defecto.
<code>inherit</code>	Hereda el modelo de caja del elemento padre.

Sección 7.1: ¿Qué es el modelo de caja?

Los bordes

El navegador crea un rectángulo para cada elemento del documento HTML. El modelo de caja describe cómo se añaden el relleno, el borde y el margen al contenido para crear este rectángulo.

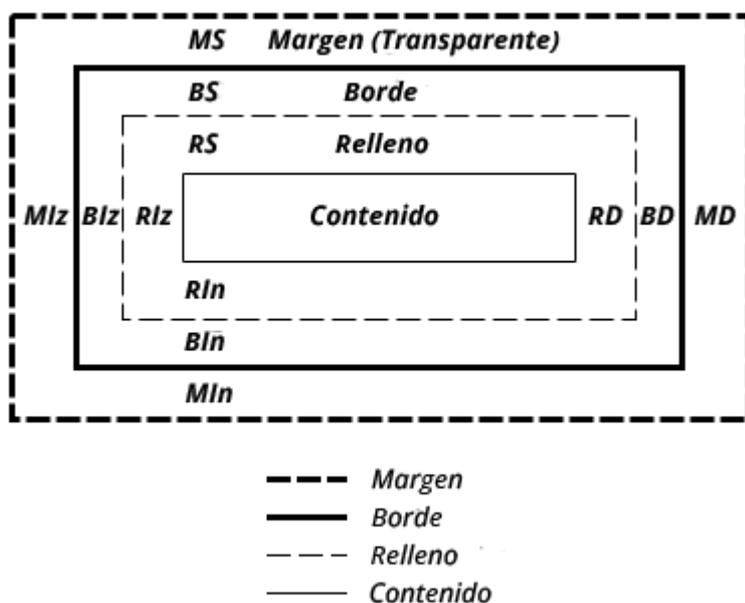


Diagrama de [CSS2.2 Working Draft](#)

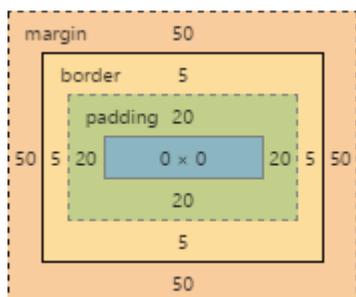
El perímetro de cada una de las cuatro áreas se denomina *arista*. Cada arista define una caja.

- El rectángulo más interior es la **caja de contenido**. Su anchura y altura dependen del contenido renderizado del elemento (texto, imágenes y cualquier elemento hijo que pueda tener).
- Lo siguiente es el **cuadro de relleno**, definido por la propiedad `padding`. Si no hay anchura de `padding` definida, el borde de relleno es igual al borde de contenido.
- Luego tenemos la **caja de borde**, definida por la propiedad `border`. Si no hay anchura de `border` definida, el borde es igual al borde de relleno.
- El rectángulo más exterior es la **caja de margen**, definida por la propiedad `margin`. Si no hay anchura de `margin` definido, el borde del margen es igual al borde del borde.

Ejemplo

```
div {  
  border: 5px solid red;  
  margin: 50px;  
  padding: 20px;  
}
```

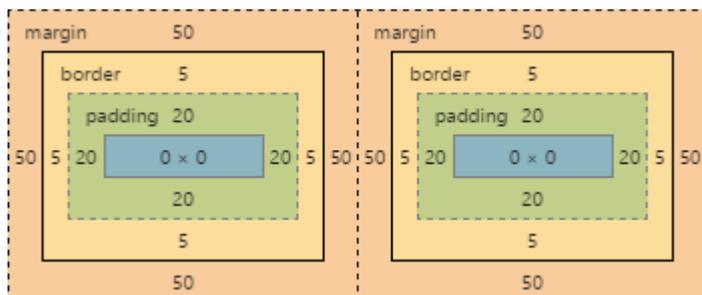
Este CSS estiliza todos los elementos `div` para que tengan un borde superior, derecho, inferior e izquierdo de `5px` de ancho; un margen superior, derecho, inferior e izquierdo de `50px`; y un relleno superior, derecho, inferior e izquierdo de `20px`.



Captura de pantalla del panel Estilos de elementos de Google Chrome

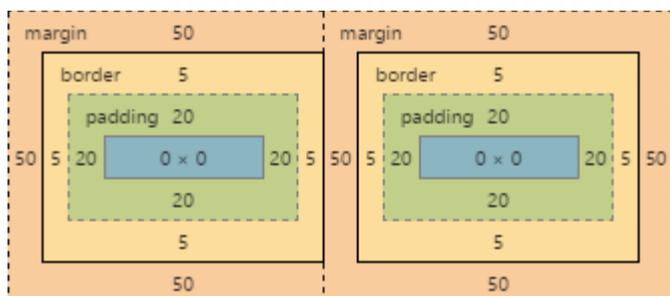
- Como no hay contenido, la región de contenido (la caja azul del centro) no tiene altura ni anchura (0px por 0px).
- La caja de relleno por defecto es del mismo tamaño que la caja de contenido, más los 20px de ancho en los cuatro bordes que estamos arriba con la propiedad `padding` (40px por 40px).
- La caja de borde es del mismo tamaño que la caja de relleno, más los 5px de ancho que definimos arriba con la propiedad `border` (50px por 50px).
- Por último, la caja de margen es del mismo tamaño que la caja de borde, más los 50px de ancho que definimos arriba con la propiedad de margen (dando a nuestro elemento un tamaño total de 150px por 150px).

Ahora vamos a darle a nuestro elemento un hermano con el mismo estilo. El navegador examina el modelo de caja de ambos elementos para determinar en qué lugar del contenido del elemento anterior debe situarse el nuevo elemento:



El contenido de cada uno de los elementos está separado por un espacio de 150px, pero las cajas de los dos elementos se tocan.

Si entonces modificamos nuestro primer elemento para que no tenga margen derecho, el borde del margen derecho estaría en la misma posición que el borde del borde derecho, y nuestros dos elementos tendrían ahora este aspecto:



Sección 7.2: box-sizing

El modelo de caja por defecto (`content-box`) puede ser contraintuitivo, ya que la `width` / `height` de un elemento no representará su anchura o altura real en pantalla tan pronto como empieces a añadir estilos de `padding` y `border` al elemento.

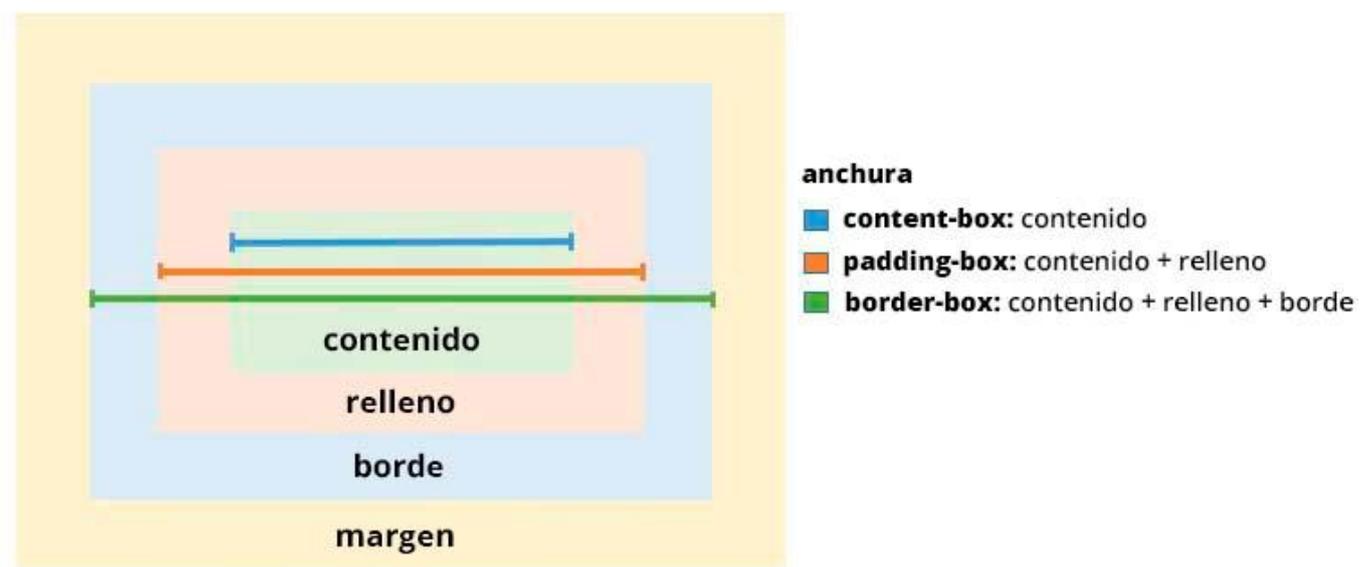
El siguiente ejemplo demuestra este problema potencial con `content-box`:

```
textarea {  
  width: 100%;  
  padding: 3px;  
  box-sizing: content-box; /* valor por defecto */  
}
```

Dado que el relleno se añadirá a la anchura del `textarea`, el elemento resultante es un `textarea` más ancho que 100%.

Afortunadamente, CSS nos permite cambiar el modelo de caja con la propiedad `box-sizing` de un elemento. Hay tres valores diferentes para la propiedad disponible:

- `content-box`: El modelo de caja común: la anchura y la altura sólo incluyen el contenido, no el relleno ni el borde.
- `padding-box`: La anchura y la altura incluyen el contenido y el relleno, pero no el borde.
- `border-box`: La anchura y la altura incluyen el contenido, el relleno y el borde.



Para resolver el problema del `textarea`, puedes cambiar la propiedad `box-sizing` a `padding-box` o `border-box`. `border-box` es la más utilizada.

```
textarea {  
  width: 100%;  
  padding: 3px;  
  box-sizing: border-box;  
}
```

Para aplicar un modelo de caja específico a cada elemento de la página, utilice el siguiente fragmento:

```
html {  
  box-sizing: border-box;  
}  
*, *:before, *:after {  
  box-sizing: inherit;  
}
```

En esta codificación `box-sizing: border-box`; no se aplica directamente a `*`, por lo que puede sobrescribir fácilmente esta propiedad en elementos individuales.

Capítulo 8: Margin

Parámetro

0
auto
unidades (por ejemplo, px)
inherit
initial

Detalles

establecer el margen en ninguno
utilizado para el centrado, ajustando uniformemente los valores a cada lado
consulte la sección de parámetros en Unidades para obtener una lista de unidades válidas
heredar el valor del margen del elemento padre
restablecer el valor inicial

Sección 8.1: Colapso del margen

Cuando dos márgenes se tocan verticalmente, se colapsan. Cuando dos márgenes se tocan horizontalmente no se colapsan.

Ejemplo de márgenes verticales adyacentes:

Considere los siguientes estilos y marcas:

```
div{
  margin: 10px;
}
<div>
  algún contenido
</div>
<div>
  algo más de contenido
</div>
```

Estarán separados 10px ya que los márgenes verticales se colapsan sobre uno y otro. (El espaciado no será la suma de dos márgenes).

Ejemplo de márgenes horizontales adyacentes:

Considere los siguientes estilos y marcas:

```
span{
  margin: 10px;
}
<span>algún</span><span>contenido</span>
```

Estarán separados 20px ya que los márgenes horizontales no se colapsan sobre uno y otro. (El espaciado será la suma de dos márgenes).

Superposición con diferentes tamaños

```
.superior{
  margin: 10px;
}
.inferior{
  margin: 15px;
}
<div class="superior">
  algún contenido
</div>
<div class="inferior">
  algo más de contenido
</div>
```

Estos elementos tendrán una separación vertical de 15px. Los márgenes se solapan todo lo que pueden, pero el mayor margen determinará el espaciado entre los elementos.

Superposición con diferentes tamaños

```
.superior{
  margin: 10px;
}
.inferior{
  margin: 15px;
}
<div class="superior">
  algún contenido
</div>
<div class="inferior">
  Algo más de contenido
</div>
```

Estos elementos tendrán una separación vertical de 15px. Los márgenes se solapan todo lo que pueden, pero el mayor margen determinará el espaciado entre los elementos.

Problema del margen superpuesto

```
.exterior-superior{
  margin: 10px;
}
.interior-superior{
  margin: 15px;
}
.exterior-inferior{
  margin: 20px;
}
.interior-inferior{
  margin: 25px;
}
<div class="exterior-superior">
  <div class="interior-superior">
    algún contenido
  </div>
</div>
<div class="exterior-inferior">
  <div class="interior-inferior">
    Algo más de contenido
  </div>
</div>
```

¿Cuál será el espaciado entre los dos textos? (pasa el ratón por encima para ver la respuesta)

El espaciado será de 25px. Dado que los cuatro márgenes se tocan entre sí, se contraerán, utilizando así el margen más grande de los cuatro.

Ahora, ¿qué tal si añadimos algunos bordes al marcado anterior?

```
div{
  border: 1px solid red;
}
```

¿Cuál será el espaciado entre los dos textos? (pasa el ratón por encima para ver la respuesta)

¡El espaciado será de 59px! Ahora sólo los márgenes de `.exterior-superior` y `.exterior-inferior` se tocan entre sí, y son los únicos márgenes colapsados. Los márgenes restantes están separados por los bordes. Así que tenemos $1\text{px} + 10\text{px} + 1\text{px} + 15\text{px} + 20\text{px} + 1\text{px} + 25\text{px} + 1\text{px}$. (Los 1px son los bordes...)

Colapso de márgenes entre elementos padre e hijo:

HTML

```
<h1>Título</h1>
<div>
  <p>Párrafo</p>
</div>
```

CSS

```
h1 {
  margin: 0;
  background: #cff;
}
div {
  margin: 50px 0 0 0;
  background: #cfc;
}
p {
  margin: 25px 0 0 0;
  background: #cf9;
}
```

En el ejemplo anterior, sólo se aplica el margen mayor. Es posible que haya esperado que el párrafo se situara a 60px del h1 (ya que el elemento `div` tiene un `margin-top` de 40px y el `p` tiene un `margin-top` de 20px). Esto no ocurre porque los márgenes se contraen para formar uno solo.

Sección 8.2: Aplicar el margen en un lado determinado

Propiedades específicas de la dirección

CSS permite especificar un lado determinado al que aplicar el margen. Las cuatro propiedades previstas para este fin son:

- `margin-left`
- `margin-right`
- `margin-top`
- `margin-bottom`

El siguiente código aplicaría un margen de 30 píxeles al lado izquierdo del `div` seleccionado.

HTML

```
<div id="miDiv"></div>
```

CSS

```
#miDiv {
  margin-left: 30px;
  height: 40px;
  width: 40px;
  background-color: red;
}
```

Parámetro

`margin-left`
30px

Detalles

La dirección en la que debe aplicarse el margen.
La anchura del margen.

Especificación de la dirección mediante la propiedad abreviada

La propiedad `margin` estándar puede ampliarse para especificar diferentes anchuras a cada lado de los elementos seleccionados. La sintaxis para hacerlo es la siguiente:

margin: <top> <right> <bottom> <left>;

El siguiente ejemplo aplica un margen de ancho cero a la parte superior del `div`, un margen de 10px al lado derecho, un margen de 50px al lado izquierdo y un margen de 100px al lado izquierdo.

HTML

```
<div id="miDiv"></div>
```

CSS

```
#miDiv {  
  margin: 0 10px 50px 100px;  
  height: 40px;  
  width: 40px;  
  background-color: red;  
}
```

Sección 8.3: Simplificación de la propiedad margin

```
p {  
  margin:1px; /* 1px de margen en todas las direcciones */  
  /* igual a:*/  
  margin:1px 1px;  
  /* igual a:*/  
  margin:1px 1px 1px;  
  /* igual a:*/  
  margin:1px 1px 1px 1px;  
}
```

Otro ejemplo:

```
p {  
  margin:10px 15px; /* 10px margin-top y bottom y 15px margin-right y left */  
  /* igual a:*/  
  margin:10px 15px 10px 15px;  
  /* igual a:*/  
  margin:10px 15px 10px;  
  /* el margen izquierdo se calculará a partir del valor del margen derecho (=15px) */  
}
```

Sección 8.4: Centrar horizontalmente los elementos de una página utilizando margin

Siempre que el elemento sea un **bloque**, y tenga un **valor de anchura explícitamente establecido**, los márgenes pueden utilizarse para centrar elementos de bloque en una página horizontalmente.

Añadimos un valor de anchura inferior a la anchura de la ventana y la propiedad `auto` de `margin` distribuye entonces el espacio restante a la izquierda y a la derecha:

```
#miDiv {  
  width:80%;  
  margin:0 auto;  
}
```

En el ejemplo anterior usamos la declaración abreviada de `margin` para establecer primero `0` a los valores de margen superior e inferior (aunque podría ser cualquier valor) y luego usamos `auto` para dejar que el navegador asigne el espacio automáticamente a los valores de margen izquierdo y derecho.

En el ejemplo anterior, el elemento `#miDiv` tiene un ancho del 80%, lo que deja un 20% de sobra. El navegador distribuye este valor a los lados restantes así:

$(100\% - 80\%) / 2 = 10\%$

Sección 8.5: Ejemplo 1:

Es obvio suponer que el valor porcentual de `margin` a `margin-left` y `margin-right` sería relativo a su elemento padre.

```
.padre {
  width : 500px;
  height: 300px;
}
.hijo {
  width : 100px;
  height: 100px;
  margin-left: 10%; /* (anchuraPadre * 10/100) => 50px */
}
```

Pero no es el caso cuando se trata de `margin-top` y `margin-bottom`. Ambas propiedades, en porcentajes, no son relativas a la altura del contenedor padre, sino a la **anchura** del contenedor padre.

Entonces..,

```
.padre {
  width : 500px;
  height: 300px;
}
.hijo {
  width : 100px;
  height: 100px;
  margin-left: 10%; /* (anchuraPadre * 10/100) => 50px */
  margin-top: 20%; /* (anchuraPadre * 20/100) => 100px */
}
```

Sección 8.6: Márgenes negativos

Margen es una de las pocas propiedades CSS que pueden establecerse con valores negativos. Esta propiedad puede utilizarse para **solapar elementos sin posicionamiento absoluto**.

```
div{
  display: inline;
}
#sobre{
  margin-left: -20px;
}
<div> Div base </div>
<div id="sobre"> Div superpuesto </div>
```

Capítulo 9: Padding

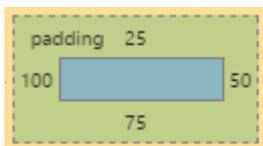
Sección 9.1: Abreviatura padding

La propiedad `padding` establece el espacio de relleno en todos los lados de un elemento. El área de relleno es el espacio entre el contenido del elemento y su borde. No se admiten valores negativos.

Para no tener que añadir relleno a cada lado individualmente (usando `padding-top`, `padding-left`, etc.), puede escribirlo como una abreviatura como se indica a continuación:

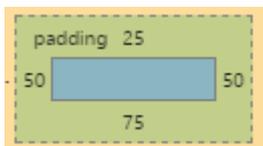
Cuatro valores:

```
<style>
.miDiv {
    padding: 25px 50px 75px 100px; /* top right bottom left; */
}
</style>
<div class="miDiv"></div>
```



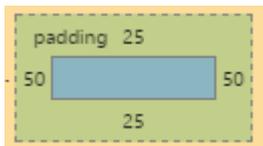
Tres valores:

```
<style>
.miDiv {
    padding: 25px 50px 75px; /* top left/right bottom */
}
</style>
<div class="miDiv"></div>
```



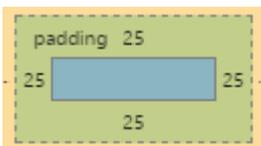
Dos valores:

```
<style>
.miDiv {
    padding: 25px 50px; /* top/bottom left/right */
}
</style>
<div class="miDiv"></div>
```



Un valor:

```
<style>
.miDiv {
    padding: 25px; /* top/right/bottom/left */
}
</style>
<div class="miDiv"></div>
```



Sección 9.2: Relleno en un lado determinado

La propiedad `padding` establece el espacio de relleno en todos los lados de un elemento. El área de relleno es el espacio entre el contenido del elemento y su borde. No se admiten valores negativos.

Puedes especificar un lado individualmente:

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

El siguiente código añadiría un relleno de `5px` a la parte superior del `div`:

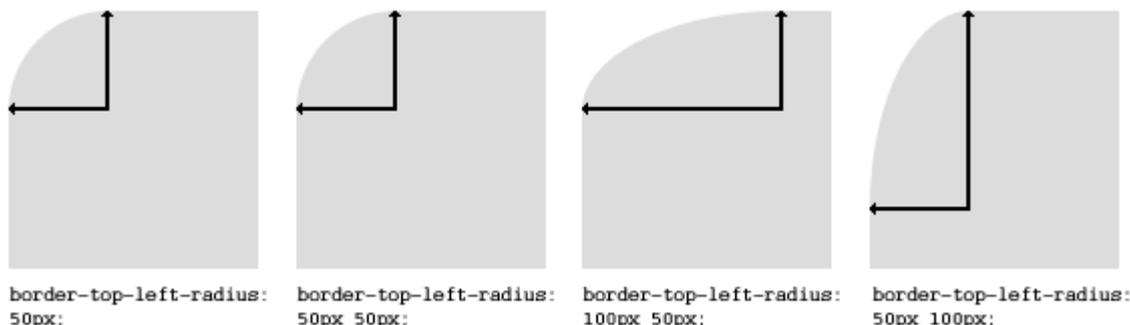
```
<style>
.miClase {
    padding-top: 5px;
}
</style>
<div class="miClase"></div>
```

Capítulo 10: Border

Sección 10.1: border-radius

La propiedad `border-radius` permite cambiar la forma del modelo de caja básico.

Cada esquina de un elemento puede tener hasta dos valores, para el radio vertical y horizontal de esa esquina (para un máximo de 8 valores).



El primer conjunto de valores define el radio horizontal. El segundo conjunto opcional de valores, precedido por un `'`, define el radio vertical. Si sólo se suministra un conjunto de valores, se utiliza tanto para el radio vertical como para el horizontal.

```
border-radius: 10px 5% / 20px 25em 30px 35em;
```

Los `10px` son el radio horizontal de la parte superior izquierda e inferior derecha. Y el `5%` es el radio horizontal de la parte superior derecha e inferior izquierda. Los otros cuatro valores después de `'` son los radios verticales para arriba-izquierda, arriba-derecha, abajo-derecha y abajo-izquierda.

Como ocurre con muchas propiedades CSS, se pueden utilizar abreviaturas para cualquiera o todos los valores posibles. Por lo tanto, puede especificar entre uno y ocho valores. La siguiente abreviatura le permite establecer el radio horizontal y vertical de cada esquina al mismo valor:

HTML

```
<div class='caja'></div>
```

CSS

```
.caja {  
  width: 250px;  
  height: 250px;  
  background-color: black;  
  border-radius: 10px;  
}
```

`border-radius` se utiliza normalmente para convertir elementos de caja en círculos. Al establecer el radio del borde en la mitad de la longitud de un elemento cuadrado, se crea un elemento circular:

```
.circulo {  
  width: 200px;  
  height: 200px;  
  border-radius: 100px;  
}
```

Dado que `border-radius` acepta porcentajes, es habitual utilizar `50%` para evitar calcular manualmente el valor de `border-radius`:

```
.circulo {
  width: 150px;
  height: 150px;
  border-radius: 50%;
}
```

Si las propiedades de anchura y altura no son iguales, la forma resultante será un óvalo en lugar de un círculo.

Ejemplo de `border-radius` específico del navegador:

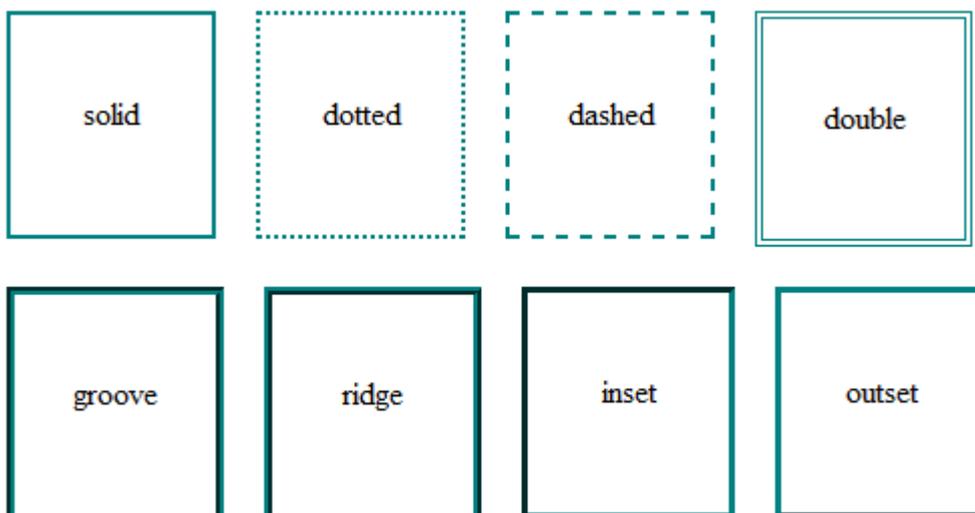
```
-webkit-border-top-right-radius: 4px;
-webkit-border-bottom-right-radius: 4px;
-webkit-border-bottom-left-radius: 0;
-webkit-border-top-left-radius: 0;
-moz-border-radius-topright: 4px;
-moz-border-radius-bottomright: 4px;
-moz-border-radius-bottomleft: 0;
-moz-border-radius-topleft: 0;
border-top-right-radius: 4px;
border-bottom-right-radius: 4px;
border-bottom-left-radius: 0;
border-top-left-radius: 0;
```

Sección 10.2: border-style

La propiedad `border-style` establece el estilo del borde de un elemento. Esta propiedad puede tener de uno a cuatro valores (por cada lado del elemento un valor).

Ejemplos:

```
border-style: dotted;
border-style: dotted solid double dashed;
```



`border-style` también puede tener los valores `none` y `hidden`. tienen el mismo efecto, excepto que `hidden` funciona para resolución de conflictos de bordes para elementos `<table>`. En una `<table>` con múltiples bordes, `none` tiene la prioridad más baja (lo que significa que, en un conflicto, el borde se mostraría), y `hidden` tiene la prioridad más alta (lo que significa que, en un conflicto, él no se mostraría el borde).

Sección 10.3: Bordos múltiples

Uso de `outline`:

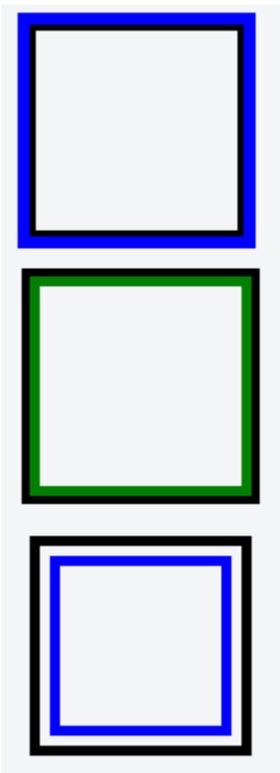
```
.div1{  
  border: 3px solid black;  
  outline: 6px solid blue;  
  width: 100px;  
  height: 100px;  
  margin: 20px;  
}
```

Uso de `box-shadow`:

```
.div2{  
  border: 5px solid green;  
  box-shadow: 0px 0px 0px 4px #000;  
  width: 100px;  
  height: 100px;  
  margin: 20px;  
}
```

Uso de un pseudoelemento:

```
.div3 {  
  position: relative;  
  border: 5px solid #000;  
  width: 100px;  
  height: 100px;  
  margin: 20px;  
}  
.div3:before {  
  content: " ";  
  position: absolute;  
  border: 5px solid blue;  
  z-index: -1;  
  top: 5px;  
  left: 5px;  
  right: 5px;  
  bottom: 5px;  
}
```



Sección 10.4: border (abreviatura)

En la mayoría de los casos querrá definir varias propiedades de `border` (`border-width`, `border-style` y `border-color`) para todos los lados de un elemento.

En vez de escribir:

```
border-width: 1px;
border-style: solid;
border-color: #000;
```

Puedes simplemente escribir:

```
border: 1px solid #000;
```

Estas abreviaturas también están disponibles para cada lado de un elemento: `border-top`, `border-left`, `border-right` y `border-bottom`. Así que puedes hacer:

```
border-top: 2px double #aaaaaa;
```

Sección 10.5: border-collapse

La propiedad `border-collapse` sólo se aplica a `table` (y elementos mostrados como `display: table` o `inline-table`) y establece si los bordes de la tabla se contraen en un único borde o se separan como en HTML estándar.

```
table {
  border-collapse: separate; /* por defecto */
  border-spacing: 2px; /* Sólo funciona si border-collapse está separado */
}
```

Véase también la entrada de documentación [Tablas – border-collapse](#)

Sección 10.6: border-image

Con la propiedad `border-image` tiene la posibilidad de establecer una imagen que se utilizará en lugar de los estilos de borde normales.

Una `border-image` consiste esencialmente en un:

- `border-image-source`: La ruta de la imagen que se va a utilizar.
- `border-image-slice`: Especifica el desplazamiento que se utiliza para dividir la imagen en **nueve regiones** (cuatro **esquinas**, cuatro **bordes** y un **centro**).
- `border-image-repeat`: Especifica cómo se escalan las imágenes de los lados y del centro de la imagen del borde.

Considere el siguiente ejemplo en el que `border.png` es una imagen de 90x90 píxeles:

```
border-image: url("border.png") 30 stretch;
```

La imagen se dividirá en nueve regiones de 30x30 píxeles. Los bordes se utilizarán como esquinas del borde mientras que el lateral se utilizará en el medio. Si el elemento es más alto / ancho que 30px esta parte de la imagen será **estirada**. La parte central de la imagen es transparente por defecto.

Sección 10.7: Creación de un borde multicolor mediante `border-image`

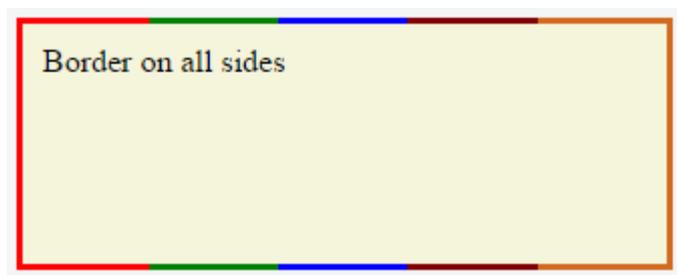
CSS

```
.bordeado {  
  border-image: linear-gradient(to right, red 20%, green 20%, green 40%, blue 40%, blue 60%,  
    maroon 60%, maroon 80%, chocolate 80%); /* degradado con los colores requeridos */  
  border-image-slice: 1;  
}
```

HTML

```
<div class='bordeado'> Bordes en todos los lados </div>
```

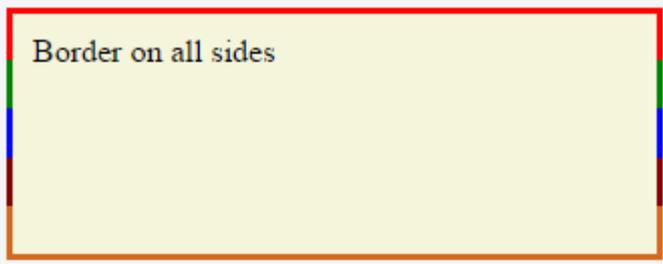
El ejemplo anterior produciría un borde compuesto por 5 colores diferentes. Los colores se definen mediante un `linear-gradient` (puede encontrar más información sobre gradientes en la documentación). Puede encontrar más información sobre la propiedad `border-image-slice` en el ejemplo `border-image` de la misma página.



(Nota: Se han añadido propiedades adicionales al elemento a efectos de presentación).

Se habrá dado cuenta de que el borde izquierdo sólo tiene un color (el color de inicio del degradado), mientras que el borde derecho también tiene un solo color (el color final del degradado). Esto se debe a la forma en que funciona la propiedad de imagen de borde. Es como si el degradado se aplicara a toda la caja y luego se enmascararían los colores de las áreas de relleno y contenido, haciendo así que parezca que sólo el borde tiene el degradado.

Qué borde(s) tienen un solo color depende de la definición del gradiente. Si el degradado es `to right` a izquierda, el borde izquierdo será el color inicial del degradado y el borde derecho será el color final. Si fuera un degradado es `to bottom` a arriba, el borde superior sería el color inicial del degradado y el borde inferior sería el color final. A continuación, la salida de un gradiente `to bottom` color de 5.



Si el borde sólo es necesario en determinados lados del elemento, la propiedad `border-width` puede utilizarse igual que con cualquier otro borde normal. Por ejemplo, si se añade el código siguiente, sólo se creará un borde en la parte superior del elemento.

```
border-width: 5px 0px 0px 0px;
```



Tenga en cuenta que, cualquier elemento que tenga la propiedad `border-image` **no respetará** el `border-radius` (es decir, el borde no se curvase). Esto se basa en la siguiente afirmación de la especificación:

Los fondos de una caja, pero no su imagen de borde, se recortan a la curva apropiada (determinada por `background-clip`).

Sección 10.8: `border-[left|right|top|bottom]`

La propiedad `border-[left|right|top|bottom]` se utiliza para añadir un borde a un lado específico de un elemento.

Por ejemplo, si quieres añadir un borde a la izquierda de un elemento, puedes hacerlo:

```
#elemento {  
    border-left: 1px solid black;  
}
```

Capítulo 11: Outline

Parámetro	Detalles
dotted	Contorno punteado
dashed	Contorno discontinuo
solid	Contorno sólido
double	Doble contorno
groove	Contorno acanalado 3D, depende del valor del color del contorno
ridge	Contorno estriado 3D, depende del valor del color del contorno
inset	Contorno de inserción 3D, depende del valor del color del contorno
outset	Contorno de inicio 3D, depende del valor del color del contorno
none	Sin contorno
hidden	Contorno oculto

Sección 11.1: Resumen

El contorno es una línea que rodea el elemento, fuera del borde. A diferencia de `border`, el contorno no ocupa espacio en el modelo de caja. Por lo tanto, añadir un contorno a un elemento no afecta a la posición del elemento ni a la de otros elementos.

Además, los contornos pueden no ser rectangulares en algunos navegadores. Esto puede ocurrir si se aplica el `outline` a un elemento `span` que contiene texto con diferentes propiedades de `font-size`. A diferencia de los bordes, los contornos *no pueden* tener esquinas redondeadas.

Las partes esenciales de `outline` son `outline-color`, `outline-style` y `outline-width`.

La definición de un contorno es equivalente a la definición de un borde:

Un contorno es una línea alrededor de un elemento. Se muestra alrededor del margen del elemento. Sin embargo, es diferente de la propiedad borde.

```
outline: 1px solid black;
```

Sección 11.2: outline-style

La propiedad `outline-style` se utiliza para establecer el estilo del contorno de un elemento.

HTML

```
<p class="p1"> Un contorno punteado </p>
<p class="p2"> Un contorno discontinuo </p>
<p class="p3"> Un contorno sólido </p>
<p class="p4"> Un doble contorno </p>
<p class="p5"> Un contorno acanalado </p>
<p class="p6"> Un contorno estriado</p>
<p class="p7"> Un contorno inserto </p>
<p class="p8"> Un contorno inicial </p>
```

CSS

```
p {  
  border: 1px solid black;  
  outline-color:blue;  
  line-height:30px;  
}  
.p1{  
  outline-style: dotted;  
}  
.p2{  
  outline-style: dashed;  
}  
.p3{  
  outline-style: solid;  
}  
.p4{  
  outline-style: double;  
}  
.p5{  
  outline-style: groove;  
}  
.p6{  
  outline-style: ridge;  
}  
.p7{  
  outline-style: inset;  
}  
.p8{  
  outline-style: outset;  
}
```

Un contorno punteado

Un contorno discontinuo

Un contorno solido

Un doble contorno

Un contorno acanalado

Un contorno estriado

Un contorno inserto

Un contorno inicial

Capítulo 12: Overflow

Valor overflow

visible
scroll
hidden

auto

inherit

Detalles

Muestra todo el contenido desbordado fuera del elemento

Oculta el contenido desbordado y añade una barra de desplazamiento

Oculta el contenido desbordado, ambas barras de desplazamiento desaparecen y la página queda fija

Igual que `scroll` si el contenido desborda, pero no añade barra de desplazamiento si el contenido cabe

Hereda el valor de esta propiedad del elemento padre

Sección 12.1: overflow-wrap

`overflow-wrap` indica a un navegador que puede romper una línea de texto dentro de un elemento de destino en varias líneas en un lugar que, de otro modo, sería irrompible. Útil para evitar que una cadena de caracteres larga de texto cause problemas de diseño debido a que desborda su contenedor.

CSS

```
div {  
  width:100px;  
  outline: 1px dashed #bbb;  
}  
#div1 {  
  overflow-wrap:normal;  
}  
#div2 {  
  overflow-wrap:break-word;  
}
```

HTML

```
<div id="div1">  
  <strong>#div1</strong>: Las palabras pequeñas se muestran normalmente, pero una palabra  
  larga como <span style="color: red;">supercalifragilisticexpialidoso</span> es demasiado  
  largo, por lo que sobrepasará el borde del salto de línea.  
</div>  
<div id="div2">  
  <strong>#div2</strong>: Las palabras pequeñas se muestran normalmente, pero una palabra  
  larga como <span style="color: red;">supercalifragilisticexpialidoso</span> se dividirá en  
  el salto de línea y continuará en la línea siguiente.  
</div>
```

#div1: Las palabras pequeñas se muestran normalmente, pero una palabra larga como **supercalifragilisticexpialidoso** es demasiado largo, por lo que sobrepasará el borde del salto de línea.

#div2: Las palabras pequeñas se muestran normalmente, pero una palabra larga como **supercalifragilisticexpialidoso** se dividirá en el salto de línea y continuará en la línea siguiente.

overflow-wrap - Valor

normal
break-word
inherit

Detalles

Permite que una palabra se desborde si es más larga que la línea
Dividirá una palabra en varias líneas, si es necesario
Hereda el valor del elemento padre para esta propiedad

Sección 12.2: overflow-x y overflow-y

Estas dos propiedades funcionan de forma similar a la propiedad `overflow` y aceptan los mismos valores. El parámetro `overflow-x` sólo funciona en el eje x o de izquierda a derecha. El `overflow-y` funciona en el eje y o de arriba a abajo.

HTML

`<div id="div-x">` Si este div es demasiado pequeño para mostrar su contenido, el contenido de la izquierda y la derecha se recortará. `</div>`

`<div id="div-y">` Si este div es demasiado pequeño para mostrar su contenido, el contenido de la parte superior e inferior se recortará. `</div>`

CSS

```
div {  
  width: 200px;  
  height: 200px;  
}  
#div-x {  
  overflow-x: hidden;  
}  
#div-y {  
  overflow-y: hidden;  
}
```

Sección 12.3: overflow-scroll

HTML

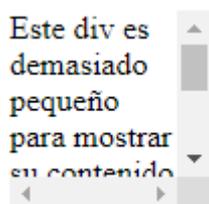
`<div>`

Este div es demasiado pequeño para mostrar su contenido para mostrar los efectos de la propiedad de `overflow`. `</div>`

CSS

```
div {  
  width:100px;  
  height:100px;  
  overflow:scroll;  
}
```

Resultado



El contenido de arriba está recortado en una caja de 100px por 100px, con desplazamiento disponible para ver el contenido desbordado.

La mayoría de los navegadores de escritorio muestran barras de desplazamiento horizontales y verticales, independientemente de que el contenido esté recortado o no. Esto puede evitar problemas con las barras de

desplazamiento que aparecen y desaparecen en un entorno dinámico. Las impresoras pueden imprimir contenido desbordante.

Sección 12.4: overflow: visible

HTML

```
<div> Aunque este div sea demasiado pequeño para mostrar su contenido, éste no se recorta.
</div>
```

CSS

```
div {
  width:50px;
  height:50px;
  overflow:visible;
}
```

Resultado

```
Aunque
este div
sea
demasiado
pequeño
para
mostrar
su
contenido,
éste no
se
recorta.
```

El contenido no se recorta y se mostrará fuera de la caja de contenido si supera el tamaño de su contenedor.

Sección 12.5: Contexto de formato de bloque creado con overflow

El uso de la propiedad `overflow` con un valor distinto de `visible` creará un nuevo **contexto de formato de bloque**. Esto es útil para alinear un elemento de bloque junto a un elemento flotante.

CSS

```
img {
  float:left;
  margin-right: 10px;
}
div {
  overflow:hidden; /* crea un contexto de formato de bloque */
}
```

HTML

```

<div>
<p> Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia. </p>
<p> Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitis sea. </p>
</div>
```

Resultado



Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia.

Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitis sea.

Este ejemplo muestra cómo los párrafos dentro de un `div` con la propiedad `overflow` establecida interactuarán con una imagen flotante.

Capítulo 13: Media Queries

Parámetro

`mediatype`

`not`

`media feature`

Detalles

(Opcional) Este es el tipo de medio. Puede ser cualquier cosa en el rango de todo a la pantalla.

(Opcional) No aplica el CSS para este tipo de medio en particular y lo aplica para todo lo demás.

Lógica para identificar el caso de uso de CSS. Las opciones se describen a continuación.

Características

`aspect-ratio`

`color`

`color-index`

`grid`

`height`

`max-width`

`min-width`

`max-height`

`min-height`

`monochrome`

`orientation`

`resolution`

`scan`

`width`

Detalles

Describe la relación de aspecto del área de visualización objetivo del dispositivo de salida.

Indica el número de bits por componente de color del dispositivo de salida. Si el dispositivo no es de color, este valor es cero.

Indica el número de entradas en la tabla de búsqueda de color para el dispositivo de salida.

Determina si el dispositivo de salida es un dispositivo de cuadrícula o un dispositivo de mapa de bits.

La característica multimedia altura describe la altura de la superficie de representación del dispositivo de salida.

CSS no se aplicará en un ancho de pantalla mayor que el especificado.

CSS no se aplicará en un ancho de pantalla inferior al especificado.

CSS no se aplicará en una altura de pantalla superior a la especificada.

CSS no se aplicará en una altura de pantalla inferior a la especificada.

Indica el número de bits por píxel en un dispositivo monocromo (escala de grises).

CSS sólo se mostrará si el dispositivo está utilizando la orientación especificada.

Consulte las observaciones para obtener más detalles.

Indica la resolución (densidad de píxeles) del dispositivo de salida.

Describe el proceso de escaneado de los dispositivos de salida de televisión.

La característica de anchura del soporte describe la anchura de la superficie de representación del dispositivo de salida (como la anchura de la ventana del documento, o la anchura de la caja de página en una impresora).

Características obsoletas

`device-aspect-ratio`

`max-device-width`

`min-device-width`

`max-device-height`

`min-device-height`

Detalles

Obsoleto. El CSS sólo se mostrará en dispositivos cuya relación altura/anchura coincida con la relación especificada. Se trata de una función obsoleta cuyo funcionamiento no está garantizado.

Obsoleto. Igual que `max-width` pero mide la anchura física de la pantalla, en lugar de la del navegador.

Obsoleto. Igual que `min-width` pero mide la anchura física de la pantalla, en lugar de la del navegador.

Obsoleto. Igual que `max-height` pero mide la anchura física de la pantalla, en lugar de la del navegador.

Obsoleto. Igual que `min-height` pero mide la anchura física de la pantalla, en lugar de la del navegador.

Sección 13.1: Terminología y estructura

Las **Media queries** permiten aplicar reglas CSS basadas en el tipo de dispositivo / medio (por ejemplo, pantalla, impresión o dispositivo portátil) denominado **tipo de media**, los aspectos adicionales del dispositivo se describen con **características de media** como la disponibilidad de color o las dimensiones de la ventana gráfica.

Estructura general de una Media Query

```
@media [...] {  
    /* Una o más reglas CSS que se aplicarán cuando se cumpla la consulta */  
}
```

Una Media Query que contiene un tipo de media

```
@media print {  
    /* Una o más reglas CSS que se aplicarán cuando se cumpla la consulta */  
}
```

Una Media Query que contiene un tipo de media y una característica media

```
@media screen and (max-width: 600px) {  
    /* Una o más reglas CSS que se aplicarán cuando se cumpla la consulta */  
}
```

Una Media Query que contiene una característica media (y un tipo de media implícito de «todos»)

```
@media (orientation: portrait) {  
    /* Una o más reglas CSS que se aplicarán cuando se cumpla la consulta */  
}
```

Sección 13.2: Ejemplo básico

```
@media screen and (min-width: 720px) {  
    body {  
        background-color: skyblue;  
    }  
}
```

La Media Query anterior especifica dos condiciones:

1. La página debe verse en una pantalla normal (no en una página impresa, proyector, etc.).
2. El ancho del puerto de vista del usuario debe ser de al menos 720 píxeles.

Si se cumplen estas condiciones, los estilos dentro de la Media Query estarán activos y el color de fondo de la página será azul cielo.

Las Media Querys se aplican dinámicamente. Si en la carga de página se cumplen las condiciones especificadas en la Media Query, el CSS se aplicará, pero se desactivará inmediatamente si se dejan de cumplir las condiciones. Por el contrario, si inicialmente no se cumplen las condiciones, el CSS no se aplicará hasta que se cumplan las condiciones especificadas.

En nuestro ejemplo, si el ancho del puerto de vista del usuario es inicialmente superior a 720 píxeles, pero el usuario reduce el ancho, el color de fondo dejará de ser azul cielo tan pronto como el usuario haya redimensionado el puerto de vista a menos de 720 píxeles de ancho.

Sección 13.3: mediatype

Las consultas multimedia tienen un parámetro de tipo medio opcional. Este parámetro se coloca directamente después de @media declaración (@media mediatype), por ejemplo:

```
@media print {  
    html {  
        background-color: white;  
    }  
}
```

El código CSS anterior dará al elemento `HTML` DOM un color de fondo blanco al ser impreso.

El parámetro `mediatype` tiene un prefijo opcional `not` o `only` que aplicará los estilos a todo excepto el tipo de medio especificado o sólo el tipo de medio especificado, respectivamente. Por ejemplo, el siguiente ejemplo de código aplicar el estilo a todos los tipos de medios excepto `print`.

```
@media not print {
  html {
    background-color: green;
  }
}
```

Y de la misma manera, para sólo mostrarlo en la pantalla, esto se puede utilizar:

```
@media only screen {
  .desvanecimiento {
    display: block;
  }
}
```

La lista de `mediatype` puede entenderse mejor con la siguiente tabla:

Sección 13.4: Media Queries para pantallas retina y no retina

Aunque esto funciona sólo para navegadores basados en WebKit, esto es útil:

```
/* ----- Pantallas no retina ----- */
@media screen
  and (min-width: 1200px)
  and (max-width: 1600px)
  and (-webkit-min-device-pixel-ratio: 1) {
}
/* ----- Pantallas retina ----- */
@media screen
  and (min-width: 1200px)
  and (max-width: 1600px)
  and (-webkit-min-device-pixel-ratio: 2)
  and (min-resolution: 192dpi) {
}
```

Información previa

Hay dos tipos de píxeles en la pantalla. Uno son los píxeles lógicos y el otro son los píxeles físicos. Sobre todo, los píxeles físicos siempre permanecen iguales, porque es el mismo para todos los dispositivos de visualización. Los píxeles lógicos cambian basado en la resolución de los dispositivos para mostrar píxeles de mayor calidad. La relación de píxeles del dispositivo es la relación entre píxeles físicos y lógicos. Por ejemplo, el MacBook Pro Retina, iPhone 4 y superiores reportan un píxel del dispositivo relación de 2, porque la resolución física lineal es el doble de la resolución lógica.

La razón por la que esto funciona sólo con navegadores basados en WebKit es debido a:

- El prefijo del proveedor `-webkit-` antes de la regla.
- Esto no se ha implementado en motores que no sean WebKit y Blink.

Sección 13.5: Width vs Viewport

Cuando usamos "width" con Media Queries es importante configurar la metaetiqueta correctamente. La metaetiqueta básica tiene este aspecto y debe colocarse dentro de la etiqueta `<head>`.

```
<meta name="viewport" content="width=device-width,initial-scale=1">
```

¿Por qué es importante?

Según la definición de MDN, "width" es

La característica `width` describe la anchura de la superficie de representación del dispositivo de salida (como la anchura de la ventana del documento, o la anchura de la caja de página en una impresora).

¿Qué significa eso?

El puerto de visualización es la anchura del propio dispositivo. Si la resolución de su pantalla dice que la resolución es 1280x720, su puerto de vista es "1280px".

A menudo, muchos dispositivos asignan diferentes cantidades de píxeles para mostrar un píxel. Por ejemplo, el iPhone 6 Plus tiene 1242x2208 de resolución. Pero la anchura y la altura reales de la ventana son 414x736. Esto significa que se para crear 1 píxel.

Pero si no ha configurado la etiqueta `meta` correctamente, intentará mostrar su página web con su resolución nativa, lo que se traduce en una vista ampliada (textos e imágenes más pequeños). una vista ampliada (textos e imágenes más pequeños).

Sección 13.6: Uso de Media Queries para distintos tamaños de pantalla

A menudo, el diseño web adaptativo implica media queries, que son bloques CSS que sólo se ejecutan si se cumple una condición. Esto es útil para el diseño web responsivo, ya que puede utilizar media queries para especificar diferentes estilos CSS para la versión móvil de su sitio web frente a la versión de escritorio.

```
@media only screen and (min-width: 300px) and (max-width: 767px) {
  .site-title {
    font-size: 80%;
  }
  /* Los estilos de este bloque sólo se aplican si el tamaño de la pantalla es de al menos
  300px de ancho, pero no superior a 767px. */
}
@media only screen and (min-width: 768px) and (max-width: 1023px) {
  .site-title {
    font-size: 90%;
  }
  /* Los estilos de este bloque sólo se aplican si el tamaño de la pantalla es de al menos
  768px de ancho, pero no superior a 1023px. */
}
@media only screen and (min-width: 1024px) {
  .site-title {
    font-size: 120%;
  }
  /* Los estilos de este bloque sólo se aplican si el tamaño de la pantalla es superior a
  1024px de ancho. */
}
```

Sección 13.7: Uso de la etiqueta link

```
<link rel="stylesheet" media="min-width: 600px" href="ejemplo.css" />
```

Esta hoja de estilo se sigue descargando, pero sólo se aplica en dispositivos con un ancho de pantalla superior a 600px.

Sección 13.8: Media Queries e IE8

Las Media Queries no son compatibles en absoluto con IE8 y versiones inferiores.

Una solución basada en Javascript

Para añadir compatibilidad con IE8, puede utilizar una de varias soluciones JS. Por ejemplo, [Respond](#) puede añadir soporte Media Query sólo para IE8 con el siguiente código:

```
<!--[if lt IE 9]>
<script
src="respond.min.js">
</script>
<![endif]-->
```

Las Media Queries es otra biblioteca que hace lo mismo. El código para añadir esa librería a tu HTML sería idéntico:

```
<!--[if lt IE 9]>
<script
src="css3-mediaqueries.js">
</script>
<![endif]-->
```

La alternativa

Si no te gusta una solución basada en JS, también deberías considerar añadir una hoja de estilo sólo para IE<9 donde ajustar tu estilo específico para IE<9. Para ello, debe agregar el siguiente HTML a su código:

```
<!--[if lt IE 9]>
<link rel="stylesheet" type="text/css" media="all" href="style-ielt9.css" />
<![endif]-->
```

Nota:

Técnicamente es una alternativa más: usar [hacks de CSS](#) para apuntar a IE<9. Tiene el mismo impacto que una hoja de estilo sólo para IE<9 pero no necesita una hoja de estilo separada para eso. Sin embargo, no recomiendo esta opción, ya que producen código CSS no válido (que no es más que una de las muchas razones por las que el uso de CSS hacks está mal visto hoy en día).

Capítulo 14: Float

Sección 14.1: Hacer flotar una imagen dentro del texto

El uso más básico de una carroza es hacer que el texto se envuelva alrededor de una imagen. El código siguiente producirá dos párrafos y una imagen, con el segundo párrafo fluyendo alrededor de la imagen. Observe que siempre es el contenido situado después del elemento que fluye alrededor del elemento flotante.

HTML

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. </p>
```

```

```

```
<p>Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet. </p>
```

CSS

```
img {  
    float:left;  
    margin-right:1rem;  
}
```

Este sería la salida

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla.



Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet.

Sección 14.2: Propiedad clear

La propiedad `clear` está directamente relacionada con floats. Valores de la propiedad:

- `none` - Por defecto. Permite elementos flotantes en ambos lados.
- `left` - No se permiten elementos flotantes en el lado izquierdo.
- `right` - No se permiten elementos flotantes en el lado derecho.
- `both` - No se permiten elementos flotantes ni en el lado izquierdo ni en el derecho.
- `initial` - Establece esta propiedad a su valor por defecto. Más información sobre `initial`.
- `inherit` - Hereda esta propiedad de su elemento padre. Más información sobre `inherit`.

```

<html>
  <head>
    <style>
      img {
        float: left;
      }
      p.clear {
        clear: both;
      }
    </style>
  </head>
  <body>
    
    <p>Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem
    ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum </p>
    <p class="clear">Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum
    Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum
    </p>
  </body>
</html>

```

Sección 14.3: Clearfix

El hack clearfix es una forma popular de contener flotadores (N. Gallagher aka @necolas)

No debe confundirse con la propiedad `clear`, clearfix es un concepto (que también está relacionado con floats, de ahí la posible confusión). Para *contener flotantes*, tienes que añadir la clase `.cf` o `.clearfix` en el contenedor (**el padre**) y aplicar estilo a esta clase con algunas reglas que se describen a continuación.

3 versiones con efectos ligeramente diferentes (fuentes: [Un nuevo micro clearfix hack](#) de N. Gallagher y [clearfix reloaded](#) de T. J. Koblenz):

Clearfix (se sigue produciendo el colapso del margen superior del contenido flotante)

```

.cf:after {
  content: "";
  display: table;
}
.cf:after {
  clear: both;
}

```

Clearfix también impide el colapso del margen superior del contenido flotante

```

/**
 * Para los navegadores modernos
 * 1. El contenido de espacio es una manera de evitar un error de Opera cuando el contenteditable se
 * incluye en cualquier otra parte del documento De lo contrario, provoca la aparición de espacios
 * en la parte superior e inferior de los elementos que están clearfixed.
 * 2. El uso de `table` en lugar de `block` sólo es necesario si se utiliza `:before` para contener
 * los márgenes superiores de los elementos hijos.*/
.cf:before,
.cf:after {
  content: " "; /* 1 */
  display: table; /* 2 */
}
.cf:after {
  clear: both;
}

```

Clearfix compatible con los navegadores obsoletos IE6 e IE7

```
.cf:before,  
.cf:after {  
    content: " ";  
    display: table;  
}  
.cf:after {  
    clear: both;  
}  
/**  
 * Únicamente para IE 6/7  
 * Incluye esta regla para activar hasLayout y contener floats.  
 */  
.cf {  
    *zoom: 1;  
}
```

[Codepen mostrando el efecto clearfix](#)

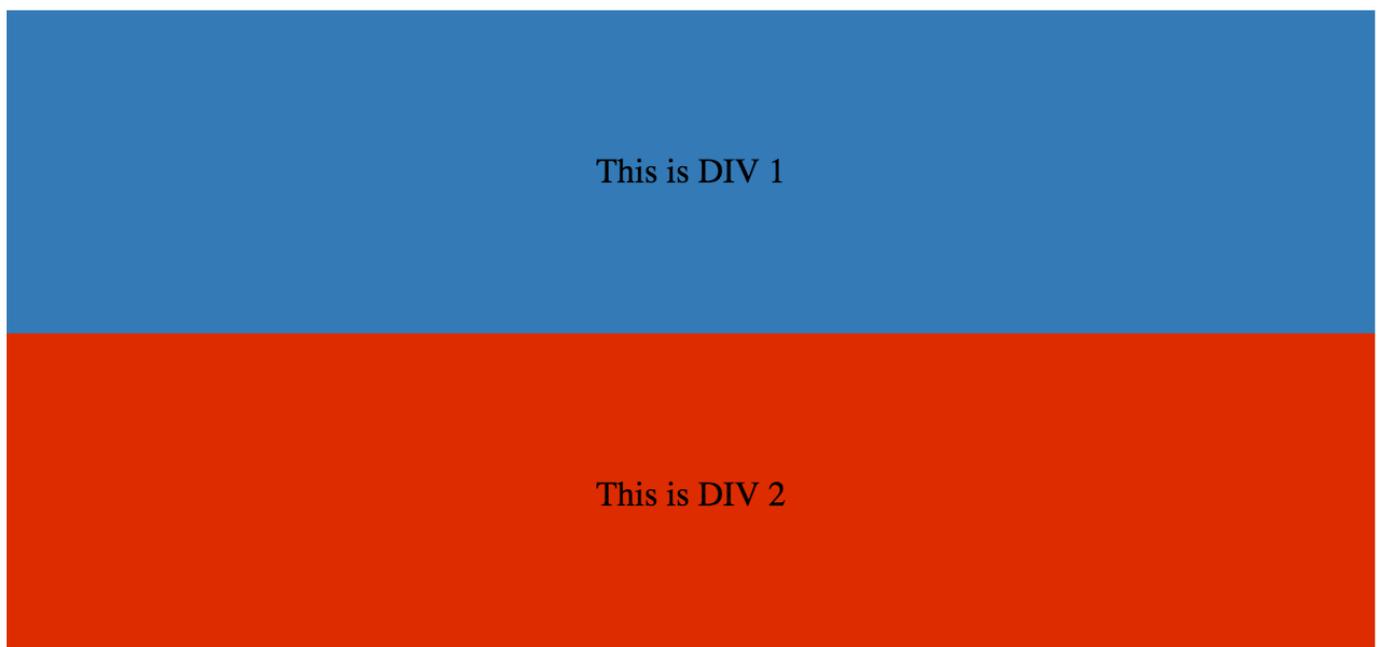
Otro recurso: [Todo lo que sabes sobre clearfix es erróneo](#) (clearfix y BFC - Block Formatting Context mientras que hasLayout se refiere a los navegadores obsoletos IE6 tal vez 7)

Sección 14.4: DIV en línea usando float

El `div` es un elemento de nivel de bloque, es decir, ocupa todo el ancho de la página y sus hermanos se colocan uno debajo del otro independientemente de su anchura.

```
<div>  
    <p>Esto es un DIV 1</p>  
</div>  
<div>  
    <p>Esto es un DIV 2</p>  
</div>
```

La salida del siguiente código será



Podemos hacerlos en línea añadiendo una propiedad CSS `float` al `div`.

HTML

```
<div class="div-exterior">
  <div class="div1-interior">
    <p>Esto es un DIV 1</p>
  </div>
  <div class="div2-interior">
    <p>Esto es un DIV 2</p>
  </div>
</div>
```

CSS

```
.div1-interior {
  width: 50%;
  margin-right: 0px;
  float: left;
  background: #337ab7;
  padding: 50px 0px;
}
.div2-interior {
  width: 50%;
  margin-right: 0px;
  float: left;
  background: #dd2c00;
  padding: 50px 0px;
}
p {
  text-align: center;
}
```



[Enlace al Codepen](#)

Sección 14.5: Uso de la propiedad overflow para borrar floats

Establecer el valor de `overflow` a `hidden`, `auto` o `scroll` a un elemento, borrará todos los flotadores dentro de ese elemento.

Nota: el uso de `overflow: scroll` siempre mostrará el cuadro de desplazamiento.

Sección 14.6: Diseño simple de dos columnas de ancho fijo

Un diseño sencillo de dos columnas consiste en dos elementos flotantes de ancho fijo. Tenga en cuenta que la barra lateral y el área de contenido no tienen la misma altura en este ejemplo. Esta es una de las partes complicadas de los diseños de varias columnas que utilizan flotadores, y requiere soluciones para hacer que varias columnas parezcan tener la misma altura.

HTML

```
<div class="envoltorio">
  <div class="barralateral">
    <h2>Barra lateral</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. </p>
  </div>
  <div class="content">
    <h1>Contenido</h1>
    <p>Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet. </p>
  </div>
</div>
```

CSS

```
.envoltorio {
  width:600px;
  padding:20px;
  background-color:pink;
  /* Los elementos flotantes no utilizan altura. Añadir "overflow:hidden;" obliga al elemento padre a expandirse para contener a sus hijos flotantes. */
  overflow:hidden;
}
.barralateral {
  width:150px;
  float:left;
  background-color:blue;
}
.contenido {
  width:450px;
  float:right;
  background-color:yellow;
}
```

Sección 14.7: Diseño simple de tres columnas de ancho fijo

HTML

```
<div class="envoltorio">
  <div class="barralateralizquierda">
    <h1>Barra lateral Izquierda</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
  </div>
  <div class="contenido">
    <h1>Contenido</h1>
    <p>Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. </p>
  </div>
  <div class="barralateralderecha">
    <h1>Barra lateral Derecha</h1>
    <p>Fusce ac turpis quis ligula lacinia aliquet.</p>
  </div>
</div>
```

CSS

```
.envoltorio {
  width:600px;
  background-color:pink;
  padding:20px;
  /* Los elementos flotantes no utilizan altura. Añadir "overflow:hidden;" obliga al elemento
  padre a expandirse para contener a sus hijos flotantes. */
  overflow:hidden;
}
.barralateralizquierda {
  width:150px;
  background-color:blue;
  float:left;
}
.contenido {
  width:300px;
  background-color:yellow;
  float:left;
}
.berralateralderecha {
  width:150px;
  background-color:green;
  float:right;
}
```

Sección 14.8: Layout de dos columnas Vago/Codicioso

Este diseño utiliza una columna flotante para crear un diseño de dos columnas sin anchuras definidas. En este ejemplo, la barra lateral izquierda es "vaga", en el sentido de que sólo ocupa el espacio que necesita. Otra forma de decir esto es que la barra lateral izquierda está "encogida". La columna de contenido de la derecha es "codiciosa", en el sentido de que ocupa todo el espacio restante.

HTML

```
<div class="barralateral">
  <h1>Sidebar</h1>
  
</div>
<div class="contenido">
  <h1>Content</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. </p>
  <p>Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet. Mauris ipsum. Nulla metus metus, ullamcorper vel, tincidunt sed, euismod in, nibh. </p>
</div>
```

CSS

```
.barralateral {  
  /* `display:table;` encoge la columna */  
  display:table;  
  float:left;  
  background-color:blue;  
}  
.contenido {  
  /* `overflow:hidden;` impide que `.content` pase por debajo de `.sidebar`. */  
  overflow:hidden;  
  background-color:yellow;  
}
```

Capítulo 15: Tipografía

Parámetro

font-style
font-variant
font-weight
font-size

line-height

font-family
color

font-stretch

text-align
text-decoration

Detalles

italics o oblique

normal o small-caps

normal, bold o numérico de 100 a 900.

El tamaño de la fuente en %, px, em o cualquier otra medida CSS válida.

La altura de la línea dada en %, px, em, o cualquier otra medida CSS válida.

Sirve para definir el nombre de la fuente tipográfica.

Cualquier representación de color CSS válida, como red, #00FF00, hsl(240, 100%, 50%), etc.

Utilización o no de una fuente confinada o ampliada. Los valores válidos son normal, ultracondensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded o ultra-expanded

start, end, left, right, center, justify, match-parent

none, underline, overline, line-through, initial, inherit

Sección 15.1: La abreviatura font

Con la sintaxis:

```
elemento {  
  font: [font-style] [font-variant] [font-weight] [font-size/line-height] [font-family];  
}
```

Puede tener todos los estilos relacionados con la fuente en una sola declaración con la abreviatura font. Simplemente utilice la propiedad font y ponga sus valores en el orden correcto.

Por ejemplo, para hacer que todos los elementos p estén en negrita con un tamaño de fuente de 20px y utilizando Arial como familia de fuentes típicamente usted código de la siguiente manera:

```
p {  
  font-weight: bold;  
  font-size: 20px;  
  font-family: Arial, sans-serif;  
}
```

Sin embargo, con la taquigrafía de fuentes se puede condensar de la siguiente manera:

```
p {  
  font: bold 20px Arial, sans-serif;  
}
```

Nota: dado que font-style, font-variant, font-weight y line-height son opcionales, los tres se omiten en este ejemplo. omitidos en este ejemplo. Es importante tener en cuenta que al utilizar el acceso directo se restablecen los demás atributos no indicados. Otro punto importante es que los dos atributos necesarios para que funcione el atajo de fuente son font-size y font-family. Si no se incluyen ambos, el acceso directo se ignora.

Valor inicial para cada una de las propiedades:

- `font-style: normal;`
- `font-variant: normal;`
- `font-weight: normal;`
- `font-stretch: normal;`
- `font-size: medium;`
- `line-height: normal;`
- `font-family` - depende del agente de usuario

Sección 15.2: quotes

La propiedad `quotes` se utiliza para personalizar las comillas de apertura y cierre de la etiqueta `<q>`.

```
q {  
  quotes: "«" "»";  
}
```

Sección 15.3: font-size

HTML

```
<div id="elemento-uno">Hola soy un texto. </div>  
<div id="elemento-dos">Hola soy un texto más pequeño. </div>
```

CSS

```
#element-uno {  
  font-size: 30px;  
}  
#element-dos {  
  font-size: 10px;  
}
```

El texto dentro de `#element-uno` tendrá un tamaño de `30px`, mientras que el texto en `#element-dos` tendrá un tamaño de `10px`.

Sección 15.4: Dirección del texto

```
div {  
  direction: ltr; /* Por defecto, el texto se lee de izquierda a derecha */  
}  
.ex {  
  direction: rtl; /* el texto se lee de derecha a izquierda */  
}  
.horizontal-tb {  
  writing-mode: horizontal-tb; /* Por defecto, el texto se lee de izquierda a derecha y de arriba abajo. */  
}  
.vertical-rtl {  
  writing-mode: vertical-rl; /* texto leído de derecha a izquierda y de arriba abajo */  
}  
.vertical-ltr {  
  writing-mode: vertical-rl; /* texto leído de izquierda a derecha y de arriba abajo */  
}
```

La propiedad `direction` se utiliza para cambiar la dirección horizontal del texto de un elemento.

Sintaxis: `direction: ltr | rtl | initial | inherit;`

La propiedad `writing-mode` cambia la alineación del texto para que pueda leerse de arriba a abajo o de izquierda a derecha, dependiendo del idioma.

Sintaxis: **direction**: `horizontal-tb` | `vertical-rl` | `vertical-lr`;

Sección 15.5: Pilas de fuentes tipográficas

font-family: `'Segoe UI'`, `Tahoma`, `sans-serif`;

El navegador intentará aplicar la fuente "Segoe UI" a los caracteres de los elementos a los que se aplica la propiedad anterior. Si esta fuente no está disponible, o no contiene un glifo para el carácter requerido, el navegador recurrirá a Tahoma y, si es necesario, a cualquier fuente sans-serif del ordenador del usuario. Tenga en cuenta que los nombres de fuentes con más de una palabra, como "Segoe UI", deben ir entre comillas simples o dobles.

font-family: `Consolas`, `'Courier New'`, `monospace`;

El navegador intentará aplicar la fuente "Consolas" a los caracteres de los elementos a los que se aplica la propiedad anterior. Si esta fuente no está disponible, o no contiene un glifo para el carácter requerido, el navegador recurrirá a "Courier New" y, si es necesario, a cualquier fuente monoespaciada del ordenador del usuario.

Sección 15.6: text-overflow

La propiedad `text-overflow` trata de cómo debe señalarse a los usuarios el contenido desbordado. En este ejemplo, la `ellipsis` representa texto recortado.

```
.text {  
  overflow: hidden;  
  text-overflow: ellipsis;  
}
```

Por desgracia, `text-overflow: ellipsis` sólo funciona en una única línea de texto. No hay manera de soportar elipsis en la última línea en CSS estándar, pero se puede lograr con la implementación no estándar webkit-sólo de flexboxes.

```
.dameEllipsis {  
  overflow: hidden;  
  text-overflow: ellipsis;  
  display: -webkit-box;  
  -webkit-box-orient: vertical;  
  -webkit-line-clamp: N; /* número de líneas a mostrar */  
  line-height: X; /* alternativa */  
  max-height: X*N; /* alternativa */  
}
```

Sección 15.7: text-shadow

Para añadir sombras al texto, utilice la propiedad `text-shadow`. La sintaxis es la siguiente:

text-shadow: `horizontal-offset vertical-offset blur color`;

Sombra sin radio de desenfoque

```
h1 {  
  text-shadow: 2px 2px #0000FF;  
}
```

Esto crea un efecto de sombra azul alrededor de un título.

Sombra con radio de desenfoque

Para añadir un efecto de desenfoque, añada una opción `blur radius` como argumento.

```
h1 {
  text-shadow: 2px 2px 10px #0000FF;
}
```

Múltiples sombras

Para dar varias sombras a un elemento, sepárelas con comas

```
h1 {
  text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;
}
```

Sección 15.8: text-transform

La propiedad `text-transform` permite cambiar las mayúsculas del texto. Los valores válidos son: `uppercase`, `capitalize`, `lowercase`, `initial`, `inherit` y `none`

HTML

```
<p class="ejemplo1">
  todas las letras en mayúsculas <!-- " TODAS LAS LETRAS EN MAYÚSCULAS " -->
</p>
<p class="ejemplo2">
  todas las letras en mayúsculas <!-- " Todas las letras en mayúscula (mayúsculas y
minúsculas)" -->
</p>
<p class="ejemplo3">
  todas las letras en minúsculas <!-- " todas las letras en minúsculas " -->
</p>
```

CSS

```
.ejemplo1 {
  text-transform: uppercase;
}
.ejemplo2 {
  text-transform: capitalize;
}
.ejemplo3 {
  text-transform: lowercase;
}
```

Sección 15.9: letter-spacing

```
h2 {
  /* añade un espacio horizontal de 1px entre cada letra; también conocido como tracking */
  letter-spacing: 1px;
}
```

La propiedad `letter-spacing` se utiliza para especificar el espacio entre los caracteres de un texto.

`letter-spacing` también admite valores negativos:

```
p {
  letter-spacing: -1px;
}
```

Recursos: <https://developer.mozilla.org/en-US/docs/Web/CSS/letter-spacing>

Sección 15.10: text-indent

```
p {  
  text-indent: 50px;  
}
```

La propiedad `text-indent` especifica cuánto espacio horizontal debe desplazarse el texto antes del comienzo de la primera línea del contenido de texto de un elemento.

Recursos:

- <https://stackoverflow.com/questions/5856952/indenting-only-the-first-line-of-text-in-a-paragraph>
- <https://www.w3.org/TR/CSS21/text.html#propdef-text-indent>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/text-indent>

Sección 15.11: text-decoration

La propiedad `text-decoration` se utiliza para establecer o eliminar decoraciones del texto.

```
h1 { text-decoration: none; }  
h2 { text-decoration: overline; }  
h3 { text-decoration: line-through; }  
h4 { text-decoration: underline; }
```

`text-decoration` puede utilizarse en combinación con `text-decoration-style` y `text-decoration-color` como propiedad abreviada:

```
.titulo { text-decoration: underline dotted blue; }
```

Se trata de una versión abreviada de

```
.titulo {  
  text-decoration-style: dotted;  
  text-decoration-line: underline;  
  text-decoration-color: blue;  
}
```

Cabe señalar que las siguientes propiedades sólo son compatibles con Firefox

```
text-decoration-color  
text-decoration-line  
text-decoration-style  
text-decoration-skip
```

Sección 15.12: word-spacing

La propiedad `word-spacing` especifica el comportamiento del espaciado entre etiquetas y palabras.

Valores posibles

- una *longitud* positiva o negativa (utilizando `em` `px` `vh` `cm` etc.) o un *porcentaje* (utilizando `%`).
- la palabra clave `normal` utiliza el espaciado entre palabras por defecto de la fuente.
- la palabra clave `inherit` toma el valor del elemento padre.

HTML

```
<p>
  <span class="normal">Este es un ejemplo que muestra el efecto del "espaciado entre
  palabras".</span><br>
  <span class="estrecho">Este es un ejemplo que muestra el efecto del "espaciado entre
  palabras".</span><br>
  <span class="extenso">Este es un ejemplo que muestra el efecto del "espaciado entre
  palabras".</span><br>
</p>
```

CSS

```
.normal { word-spacing: normal; }
.estrecho { word-spacing: -3px; }
.extenso { word-spacing: 10px; }
```

Más información:

[word-spacing - MDN](#)

Sección 15.13: font-variant

Atributos:

normal

Atributo por defecto de los tipos de letra.

small-caps

Pone todas las letras en mayúsculas, pero hace que las minúsculas (del texto original) sean de menor tamaño que las letras que originalmente estaban en mayúsculas.

HTML

```
<p class="smallcaps"> Documentación sobre fuentes CSS <br>
Y EjEmpLo
</p>
```

CSS

```
.smallcaps{
  font-variant: small-caps;
}
```

Salida

DOCUMENTACIÓN SOBRE FUENTES CSS
Y EJEMPLO

Nota: La propiedad `font-variant` es una abreviatura de las propiedades: `font-variant-caps`, `font-variant-numeric`, `font-variant-alternates`, `font-variant-ligatures` y `font-variant-east-asian`.

Capítulo 16: Diseño de caja flexible (Flexbox)

El módulo de caja flexible, o "flexbox" para abreviar, es un modelo de caja diseñado para interfaces de usuario que permite a los usuarios alinear y distribuir el espacio entre los elementos de un contenedor de modo que los elementos se comporten de forma predecible cuando el diseño de la página deba adaptarse a distintos tamaños de pantalla desconocidos. Un contenedor flexible expande los artículos para llenar el espacio disponible y los encoge para evitar el desbordamiento.

Sección 16.1: Centrado dinámico vertical y horizontal (align-items, justify-content)

Ejemplo sencillo (centrar un solo elemento)

HTML

```
<div class="alineador">
  <div class="alineador-elemento">...</div>
</div>
```

CSS

```
.alineador {
  display: flex;
  align-items: center;
  justify-content: center;
}
.alineador-elemento {
  max-width: 50%; /*para la demostración. Utiliza la anchura real en su lugar. */
}
```

Aquí hay una [demostración](#)

Razonamiento

Propiedad	Valor	Descripción
align-items	center	Esto centra los elementos a lo largo del eje distinto del especificado por flex-direction, es decir, centrado vertical para un flexbox horizontal y centrado horizontal para un vertical.
justify-content	center	Esto centra los elementos a lo largo del eje especificado por flex-direction. Es decir, para un flexbox horizontal (flex-direction: row), se centra horizontalmente, y para un flexbox vertical flexbox (flex-direction: column) flexbox vertical, se centra verticalmente)

Ejemplos de propiedades individuales

Todos los estilos siguientes se aplican a este sencillo diseño:

```
<div id="contenedor">
  <div></div>
  <div></div>
  <div></div>
</div>
```

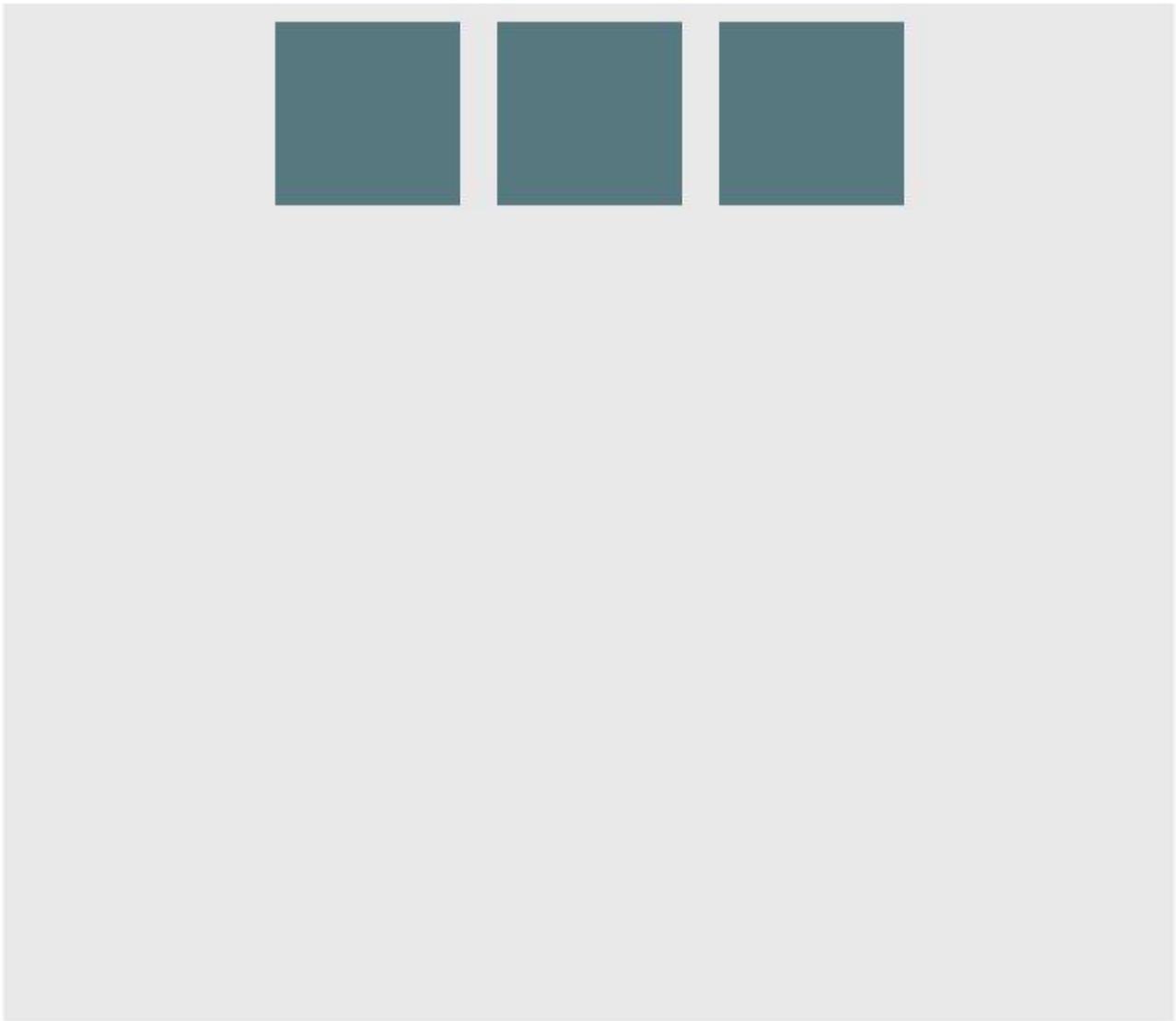
donde #container es el flex-box.

Ejemplo justify-content: center en un flexbox horizontal

CSS

```
div#contenedor {  
  display: flex;  
  flex-direction: row;  
  justify-content: center;  
}
```

Resultado



Ejemplo: justify-content: center en un flexbox vertical

CSS

```
div#contenedor {  
  display: flex;  
  flex-direction: column;  
  justify-content: center;  
}
```

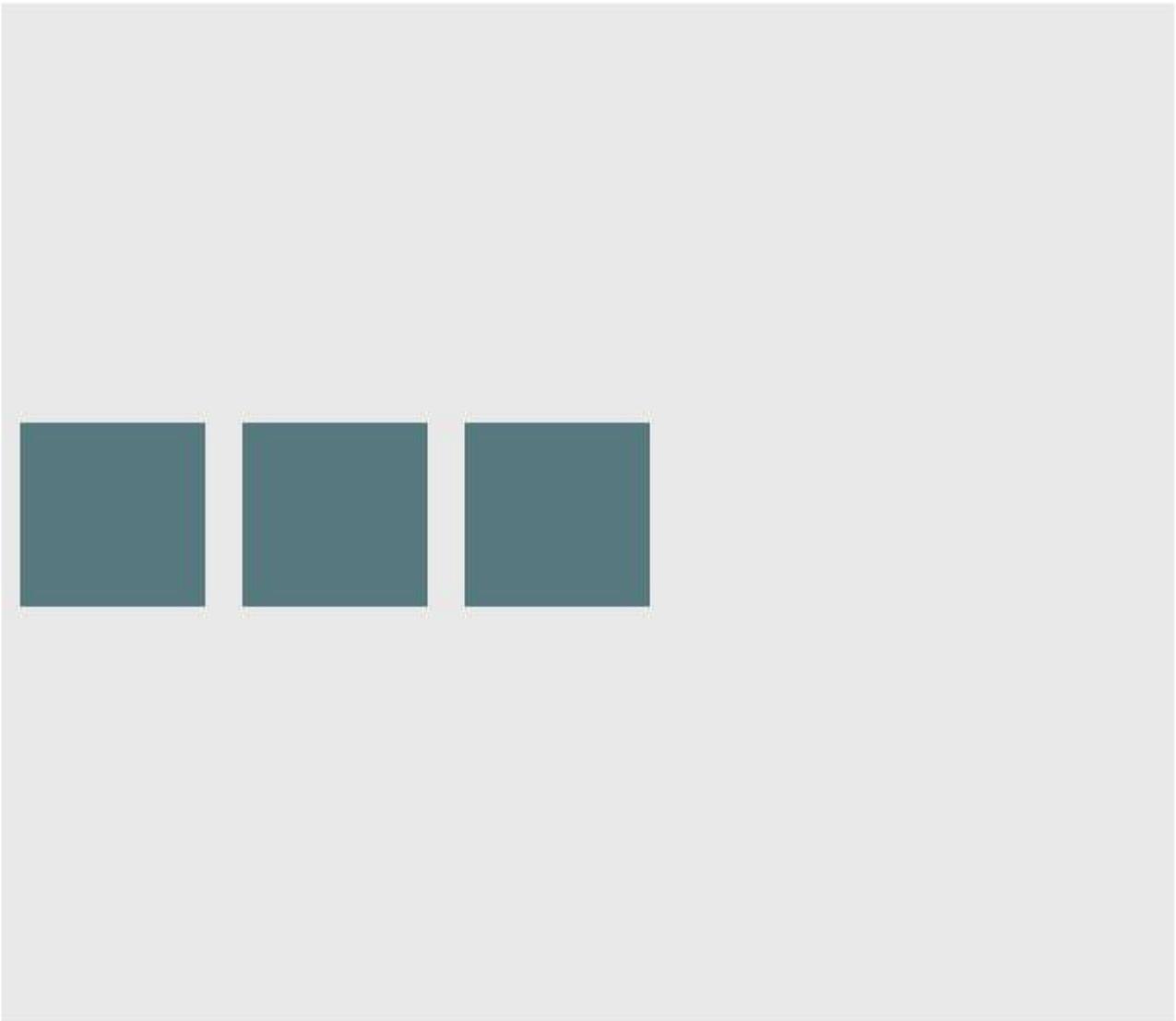
Resultado



Ejemplo: `align-content: center` en un flexbox horizontal

CSS

```
div#contenedor {  
  display: flex;  
  flex-direction: row;  
  align-items: center;  
}
```



Ejemplo: `align-content: center` en un flexbox vertical

CSS

```
div#contenedor {  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
}
```



Ejemplo: Combinación para centrar ambos en flexbox horizontal

```
div#contenedor {  
  display: flex;  
  flex-direction: row;  
  justify-content: center;  
  align-items: center;  
}
```



Ejemplo: Combinación para centrar ambos en flexbox vertical

```
div#contenedor {  
  display: flex;  
  flex-direction: column;  
  justify-content: center;  
  align-items: center;  
}
```



Sección 16.2: Pie de página sticky de altura variable

Este código crea un pie de página sticky. Cuando el contenido no llega al final de la ventana gráfica (viewport), el pie de página se pega a la parte inferior de la ventana. Cuando el contenido se extiende más allá de la parte inferior de la ventana gráfica, el pie de página también es empujado fuera de la ventana gráfica.

HTML

```
<div class="cabecera">  
  <h2>Cabecera</h2>  
</div>  
<div class="contenido">  
  <h1>Contenido</h1>  
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. </p>  
</div>  
<div class="pie-pagina">  
  <h4>Pie de página</h4>  
</div>
```

CSS

```
html, body {
    height: 100%;
}
body {
    display: flex;
    flex-direction: column;
}
.contenido {
    /* Incluye `0 auto` para una mayor compatibilidad con los navegadores. */
    flex: 1 0 auto;
}
.cabecera, .pie-pagina {
    background-color: grey;
    color: white;
    flex: none;
}
```

Sección 16.3: Ajustar óptimamente los elementos a su contenedor

Una de las mejores características de flexbox es permitir ajustar de forma óptima los contenedores a su elemento padre.

HTML

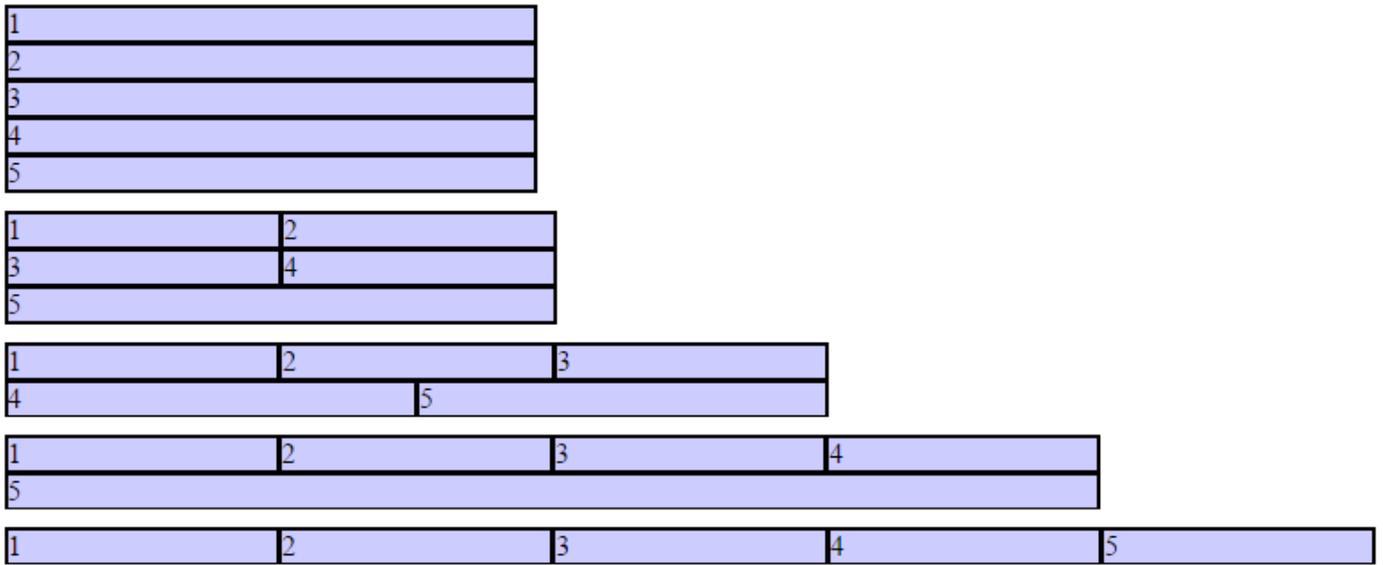
```
<div class="contenedor-flex">
  <div class="elemento-flex">1</div>
  <div class="elemento-flex">2</div>
  <div class="elemento-flex">3</div>
  <div class="elemento-flex">4</div>
  <div class="elemento-flex">5</div>
</div>
```

CSS

```
.contenedor-flex {
    background-color: #000;
    height: 100%;
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: flex-start;
    align-content: stretch;
    align-items: stretch;
}
.elemento-flex {
    background-color: #ccf;
    margin: 0.1em;
    flex-grow: 1;
    flex-shrink: 0;
    flex-basis: 200px; /* o % podría utilizarse para garantizar un diseño específico */
}
```

Resultado

Las columnas se adaptan al cambio de tamaño de la pantalla.



Sección 16.4: El Santo Grial del diseño con Flexbox

El diseño del Santo Grial es un diseño con una cabecera y un pie de página de altura fija, y un centro con 3 columnas. Las 3 columnas incluyen una navegación lateral de ancho fijo, un centro fluido, y una columna para otros contenidos como anuncios (el centro fluido aparece primero en el marcado). CSS Flexbox se puede utilizar para lograr esto con un marcado muy simple:

Marcado HTML

```
<div class="contenedor">
  <header class="cabecera">Cabecera</header>
  <div class="cuerpo-contenido">
    <main class="contenido">Contenido</main>
    <nav class="nav-lateral">Nav-Lateral</nav>
    <aside class="anuncios">Anuncios</aside>
  </div>
  <footer class="pie-paagina">Pie de página</footer>
</div>
```

CSS

```
body {  
  margin: 0;  
  padding: 0;  
}  
.contenedor {  
  display: flex;  
  flex-direction: column;  
  height: 100vh;  
}  
.cabecera {  
  flex: 0 0 50px;  
}  
.cuerpo-contenido {  
  flex: 1 1 auto;  
  display: flex;  
  flex-direction: row;  
}  
.cuerpo-contenido .contenido {  
  flex: 1 1 auto;  
  overflow: auto;  
}  
.cuerpo-contenido .nav-lateral {  
  order: -1;  
  flex: 0 0 100px;  
  overflow: auto;  
}  
.cuerpo-contenido .anuncios {  
  flex: 0 0 100px;  
  overflow: auto;  
}  
.pie-pagina {  
  flex: 0 0 50px;  
}
```

Sección 16.5: Botones perfectamente alineados dentro de tarjetas con flexbox

Es un patrón regular en el diseño de estos días para alinear verticalmente **llamada a la acción** dentro de sus tarjetas que contienen así:



Esto puede lograrse utilizando un truco especial con `flexbox`

HTML

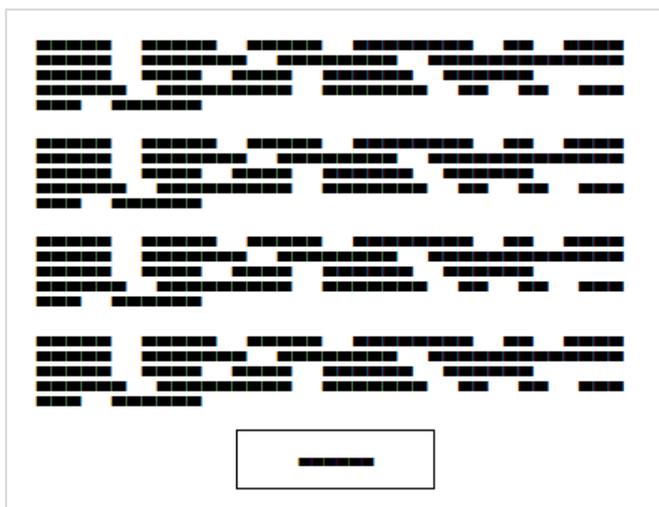
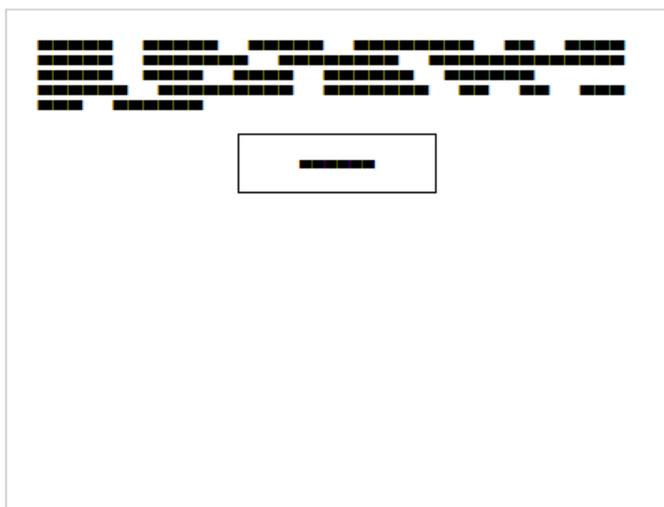
```
<div class="tarjetas">
  <div class="tarjeta">
    <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa ese
    enim mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
    <p><button>Acción</button></p>
  </div>
  <div class="tarjeta">
    <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa ese
    enim mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
    <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa ese
    enim mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
    <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa esse
    enim mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
    <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa esse
    enim mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
    <p><button>Acción</button></p>
  </div>
</div>
```

En primer lugar, utilizamos CSS para aplicar `display: flex`; al contenedor. Esto creará 2 columnas iguales en altura con el contenido fluya de forma natural en su interior.

CSS

```
.tarjetas {
  display: flex;
}
.tarjeta {
  border: 1px solid #ccc;
  margin: 10px 10px;
  padding: 0 20px;
}
button {
  height: 40px;
  background: #fff;
  padding: 0 40px;
  border: 1px solid #000;
}
p:last-child {
  text-align: center;
}
```

El diseño cambiará y quedará así:



Para mover los botones a la parte inferior del bloque, tenemos que aplicar `display: flex`; a la propia tarjeta con la dirección en `column`. Después de eso, debemos seleccionar el último elemento dentro de la tarjeta y establecer el `margin-top` a `auto`. Esto empujará el último párrafo a la parte inferior de la tarjeta y conseguirá el resultado deseado.

CSS definitivo

```
.tarjetas {
  display: flex;
}
.tarjeta {
  border: 1px solid #ccc;
  margin: 10px 10px;
  padding: 0 20px;
  display: flex;
  flex-direction: column;
}
button {
  height: 40px;
  background: #fff;
  padding: 0 40px;
  border: 1px solid #000;
}
p:last-child {
  text-align: center;
  margin-top: auto;
}
```

Sección 16.6: Misma altura en los contenedores anidados

Este código asegura que todos los contenedores anidados tengan siempre la misma altura. Esto se hace asegurándose de que todos los elementos anidados tengan la misma altura que el div principal que los contiene.

Este efecto se consigue gracias a que la propiedad `align-items` está establecida por defecto en `stretch`.

HTML

```
<div class="contenedor">
  <div style="background-color: red">
    Algunos <br />
    datos <br />
    para hacer<br />
    una altura <br />
  </div>
  <div style="background-color: blue">
    Menos <br />
    líneas <br />
  </div>
</div>
```

CSS

```
.contenedor {
  display: flex;
  align-items: stretch; // valor por defecto
}
```

Nota: [No funciona en versiones de IE inferiores a 10](#)

Capítulo 17: Cascada y especificidad

Sección 17.1: Cálculo de la especificidad del selector

Cada selector CSS individual tiene su propio valor de especificidad. Cada selector en una secuencia aumenta la especificidad global de la secuencia. Los selectores pertenecen a uno de estos tres grupos de especificidad: A, B y C. Cuando varias secuencias de selectores seleccionan un elemento determinado, el navegador utiliza los estilos aplicados por la secuencia con mayor especificidad.

Grupo	Se compone de	Ejemplos
A	selectores id	<code>#foo</code>
B	selectores de clase selectores de atributos pseudoclases	<code>.bar</code> <code>[title]</code> , <code>[colspan="2"]</code> <code>:hover</code> , <code>:nth-child(2)</code>
C	selectores de tipo pseudoelementos	<code>div</code> , <code>li</code> <code>::before</code> , <code>::first-letter</code>

El Grupo A es el más específico, seguido del Grupo B y, por último, el Grupo C.

El selector universal (*) y los combinadores (como > y ~) no tienen especificidad.

Ejemplo 1: Especificidad de varias secuencias selectoras

```
#foo #baz {} /* a=2, b=0, c=0 */
#foo.bar {} /* a=1, b=1, c=0 */
#foo {} /* a=1, b=0, c=0 */
.bar:hover {} /* a=0, b=2, c=0 */
div.bar {} /* a=0, b=1, c=1 */
:hover {} /* a=0, b=1, c=0 */
[title] {} /* a=0, b=1, c=0 */
.bar {} /* a=0, b=1, c=0 */
div ul + li {} /* a=0, b=0, c=3 */
p::after {} /* a=0, b=0, c=2 */
*::before {} /* a=0, b=0, c=1 */
::before {} /* a=0, b=0, c=1 */
div {} /* a=0, b=0, c=1 */
* {} /* a=0, b=0, c=0 */
```

Ejemplo 2: Cómo utiliza la especificidad el navegador

Imagina la siguiente implementación de CSS:

```
#foo {
  color: blue;
}
.bar {
  color: red;
  background: black;
}
```

Aquí tenemos un selector de ID que declara el `color` como *azul*, y un selector de clase que declara el `color` como *rojo* y el `background` como *negro*.

Un elemento con un ID de `#foo` y una clase de `.bar` será seleccionado por ambas declaraciones. Los selectores de ID tienen una especificidad de Grupo A y los selectores de clase tienen una especificidad del Grupo B. Un selector de ID supera a cualquier número de selectores de clase. Debido a esto, `color: blue;` del selector `#foo` y `background: black;` del selector `.bar` serán se aplicarán al elemento. La mayor especificidad del selector ID hará que el navegador ignore la declaración de `color` del selector `.bar`.

Ahora imagina una implementación CSS diferente:

```
.bar {
  color: red;
  background: black;
}
.baz {
  background: white;
}
```

Aquí tenemos dos selectores de clase; uno de los cuales declara el `color` como *rojo* y el `background` como *negro*, y el otro declara el `background` como *blanco*.

Un elemento con las clases `.bar` y `.baz` se verá afectado por ambas declaraciones, sin embargo el problema que tenemos ahora es que tanto `.bar` como `.baz` tienen una especificidad idéntica del Grupo B. La naturaleza en cascada de CSS resuelve esto para nosotros: como `.baz` se define *después* de `.bar`, nuestro elemento termina con el color *rojo* de `.bar` pero el *blanco* `background` de `.baz`.

Ejemplo 3: Cómo manipular la especificidad

El último fragmento del Ejemplo 2 puede manipularse para garantizar que se utiliza la declaración de color de nuestro selector de clase `.bar` en lugar de la del selector de clase `.baz`. en lugar de la del selector de clase `.baz`.

```
.bar {} /* a=0, b=1, c=0 */
.baz {} /* a=0, b=1, c=0 */
```

La forma más habitual de conseguirlo sería averiguar qué otros selectores se pueden aplicar a la secuencia del selector `.bar`. Por ejemplo, si la clase `.bar` sólo se aplicara a elementos `span`, podríamos modificar el selector `.bar` por `span.bar`. Esto le daría una nueva especificidad de Grupo C, que anularía la falta del selector `.baz`:

```
span.bar {} /* a=0, b=1, c=1 */
.baz {} /* a=0, b=1, c=0 */
```

Sin embargo, no siempre es posible encontrar otro selector común que sea compartido por cualquier elemento que utilice la clase `.bar`. Por ello, CSS nos permite duplicar selectores para aumentar la especificidad. En lugar de simplemente `.bar`, podemos utilizar en su lugar `.bar.bar` (Véase [La gramática de los selectores, Recomendación del W3C](#)). Esto sigue seleccionando cualquier elemento con una clase `.bar`, pero ahora tiene el doble de especificidad del Grupo B:

```
.bar.bar {} /* a=0, b=2, c=0 */
.baz {} /* a=0, b=1, c=0 */
```

!important y declaraciones de estilo en línea

Se considera que el indicador `!important` de una declaración de estilo y los estilos declarados mediante el atributo `style` de HTML tienen una mayor especificidad que cualquier selector. Si existen, la declaración de estilo a la que afecten anulará otras declaraciones independientemente de su especificidad. Es decir, a menos que tenga más de una declaración que contenga un indicador `!important` para la misma propiedad que se apliquen al mismo elemento. Entonces, las reglas normales de especificidad se aplicarán a esas propiedades en referencia a las demás.

Debido a que anulan completamente la especificidad, el uso de `!important` está mal visto en la mayoría de los casos de uso. Hay que utilizarlo lo menos posible. Para mantener el código CSS eficiente y mantenible a largo plazo, casi siempre es mejor aumentar la especificidad del selector circundante que utilizar `!important`.

Una de esas raras excepciones en las que `!important` no está mal visto, es cuando se implementan clases ayudantes genéricas como una clase `.hidden` o `.background-yellow` que se supone que siempre anulan una o más propiedades dondequiera que se encuentren. Y, aun así, tienes que saber lo que haces. Lo último que quieres, cuando escribes CSS mantenible, es tener banderas `!important` en todo tu CSS.

Nota final

Un error común sobre la especificidad CSS es que los valores de los grupos *A*, *B* y *C* deben combinarse entre sí ($a=1, b=5, c=1 \Rightarrow 151$). Este **no** es el caso. Si este fuera el caso, tener 20 de un selector del grupo *B* o *C* sería suficiente para anular un único selector del grupo *A* o *B* respectivamente. Los tres grupos deben considerarse niveles individuales de especificidad. La especificidad no puede representarse con un único valor.

Al crear su hoja de estilo CSS, debe mantener la menor especificidad posible. Si necesitas hacer la especificidad un poco más alta para sobrescribir otro método, hazla más alta pero lo más baja posible para hacerla más alta. Usted no debería necesitar un selector como este:

```
body.pagina header.contenedor nav div#nav-principal li a {}
```

Esto dificulta futuros cambios y contamina esa página CSS.

Puede calcular la especificidad de su selector [aquí](#).

Sección 17.2: La declaración !important

La declaración **!important** se utiliza para anular la especificidad habitual en una hoja de estilo dando mayor prioridad a una regla. Su uso es: `property : value !important;`

```
#midiv {
    font-weight: bold !important; /* Esta propiedad no será anulada por la siguiente regla */
}
#exteriordiv #midiv {
    font-weight: normal; /* #midiv font-weight no se ajustará a normal aunque tenga una
especificidad mayor debido a la declaración !important anterior */
}
```

Se recomienda encarecidamente evitar el uso de **!important** (a menos que sea absolutamente necesario), ya que perturbará el flujo natural de las reglas CSS, lo que puede generar incertidumbre en su hoja de estilo. También es importante tener en cuenta que cuando se aplican varias declaraciones **!important** a la misma regla sobre un determinado elemento, la de mayor especificidad será la que se aplique.

He aquí algunos ejemplos en los que el uso de la declaración **!important** puede estar justificado:

- Si sus reglas no deben ser anuladas por ningún estilo en línea del elemento que está escrito dentro del atributo `style` del elemento html.
- Para dar al usuario más control sobre la accesibilidad web, como aumentar o disminuir el tamaño de la fuente, mediante anulando el estilo de autor mediante **!important**.
- Para pruebas y depuración utilizando el elemento `inspect`.

Vea también:

- [W3C - 6 Asignación de valores de propiedades, cascada y herencia -- 6.4.2 Reglas !important](#)

Sección 17.3: En cascada

La cascada y la especificidad se utilizan conjuntamente para determinar el valor final de una propiedad de estilo CSS. También definen los mecanismos para resolver conflictos en los conjuntos de reglas CSS.

Orden de carga de CSS

Los estilos se leen en las siguientes fuentes, por este orden:

1. Hoja de estilo del agente de usuario (Los estilos suministrados por el proveedor del navegador)
2. Hoja de estilo del usuario (El estilo adicional que un usuario ha establecido en su navegador)
3. Hoja de estilo del autor (Autor significa aquí el creador de la página web/sitio web)
 - Puede que uno o varios archivos `.css`
 - En el elemento `<style>` del documento HTML
4. Estilos en línea (en el atributo `style` de un elemento HTML)

El navegador buscará el/los estilo(s) correspondiente(s) al renderizar un elemento.

¿Cómo se resuelven los conflictos?

Cuando sólo un conjunto de reglas CSS intenta establecer un estilo para un elemento, entonces no hay conflicto, y se utiliza ese conjunto de reglas.

Cuando se encuentran varios conjuntos de reglas con configuraciones contradictorias, se utilizan primero las reglas de Especificidad y luego las reglas de Cascada para determinar qué estilo utilizar.

Ejemplo 1 - Reglas de especificidad

```
.miestilo { color: blue; } /* especificidad: 0, 0, 1, 0 */
div { color: red; } /* especificidad: 0, 0, 0, 1 */
<div class="miestilo">Hola Mundo</div>
```

¿De qué color será el texto?

azul

Primero se aplican las reglas de especificidad, y "gana" el que tenga la especificidad más alta.

Ejemplo 2 - Reglas en cascada con selectores idénticos

Archivo CSS externo

```
.clase {
  background: #FFF;
}
```

CSS interno (en el archivo HTML)

```
<style>
.clase {
  background: #000;
}
</style>
```

En este caso, donde tienes selectores idénticos, la cascada entra en acción, y determina que el último cargado "gana".

Ejemplo 3 - Reglas en cascada tras las reglas de especificidad

```
body > .miestilo { background-color: blue; } /* especificidad: 0, 0, 1, 1 */
.otroestilo > div { background-color: red; } /* especificidad: 0, 0, 1, 1 */
<body class="otroestilo">
  <div class="miestilo">Hello World</div>
</body>
```

¿De qué color será el fondo?

rojo

Tras aplicar las reglas de especificidad, sigue habiendo conflicto entre el azul y el rojo, por lo que se aplican las reglas en cascada sobre las reglas de especificidad. La cascada mira el orden de carga de las reglas, ya sea dentro del mismo archivo .css o en la colección de fuentes de estilo. La última cargada anula las anteriores. En este caso, la regla `.otherstyle > div` "gana".

Nota final

- La especificidad del selector siempre tiene prioridad.
- El orden de las hojas de estilo desempata.
- Los estilos en línea se imponen a todo.

Sección 17.4: Ejemplo de especificidad más complejo

```
div {
  font-size: 7px;
  border: 3px dotted pink;
  background-color: yellow;
  color: purple;
}
body.miestilo > div.miotroestilo {
  font-size: 11px;
  background-color: green;
}
#elemento1 {
  font-size: 24px;
  border-color: red;
}
.miestilo .miotroestilo {
  font-size: 16px;
  background-color: black;
  color: red;
}
<body class="miestilo">
  <div id="elemento1" class="miotroestilo">
    ¡Hola, mundo!
  </div>
</body>
```

¿Qué bordes, colores y tamaños de letra tendrá el texto?

font-size

font-size: 24;, ya que el conjunto de reglas **#elemento1** tiene la mayor especificidad para el **<div>** en cuestión, cada propiedad aquí se establece.

border

border: 3px dotted red; El color **red** del borde se toma del conjunto de reglas **#elemento1**, ya que tiene la mayor especificidad. Las otras propiedades del borde, grosor del borde y estilo del borde provienen del conjunto de reglas **div**.

background-color

background-color: green; El **background-color** se establece en los conjuntos de reglas **div**, **body.miestilo > div.myotherstyle**, y **.miestilo .myotherstyle**. Las especificidades son (0, 0, 1) frente a (0, 2, 2) frente a (0, 2, 0), por lo que la del medio gana".

color

color: red; El color se establece en los conjuntos de reglas **div** y **.miestilo .myotherstyle**. Este último tiene la mayor especificidad de (0, 2, 0) y "gana".

Capítulo 18: Colores

Sección 18.1: currentColor

`currentColor` devuelve el valor calculado del color del elemento actual.

Uso en el mismo elemento

Aquí `currentColor` evalúa a rojo ya que la propiedad `color` está establecida a `red`:

```
div {  
  color: red;  
  border: 5px solid currentColor;  
  box-shadow: 0 0 5px currentColor;  
}
```

En este caso, especificar `currentColor` para el borde es probablemente redundante porque omitirlo debería producir resultados idénticos. Sólo use `currentColor` dentro de la propiedad `border` dentro del mismo elemento si sería sobrescrito de otro modo debido a un selector más específico.

Dado que es el color calculado, el borde será verde en el siguiente ejemplo debido a que la segunda regla prevalece sobre la primera:

```
div {  
  color: blue;  
  border: 3px solid currentColor;  
  color: green;  
}
```

Hereda del elemento padre

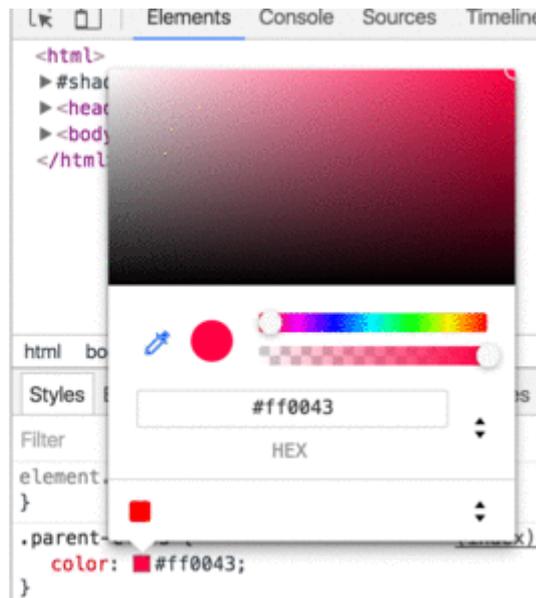
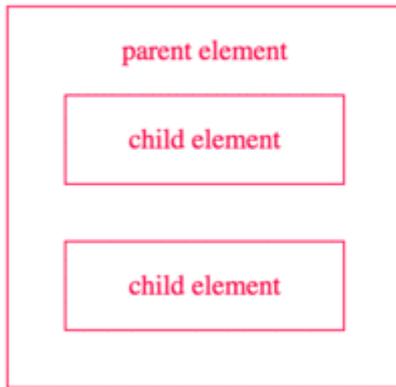
El color del elemento padre es heredado, aquí `currentColor` se evalúa a 'blue', haciendo que el color del borde del elemento hijo sea azul.

```
.padre-clase {  
  color: blue;  
}  
.padre-clase .hijo-clase {  
  border-color: currentColor;  
}
```

`currentColor` también puede ser utilizado por otras reglas que normalmente no heredarían de la propiedad `color`, tales como `background-color`. El ejemplo siguiente muestra a los hijos utilizando el color establecido en el padre como fondo:

```
.padre-clase {  
  color: blue;  
}  
.padre-clase .hijo-clase {  
  background-color: currentColor;  
}
```

Posible resultado:



Sección 18.2: Palabras clave de color

La mayoría de los navegadores admiten el uso de palabras clave de color para especificar un color. Por ejemplo, para establecer el color de un elemento en azul, utilice la palabra clave `blue`:

```
.alguna-clase {  
  color: blue;  
}
```

Las palabras clave CSS no distinguen entre mayúsculas y minúsculas: `blue`, `Blue` y `BLUE` darán como resultado `#0000FF`.

Palabras clave en color

Nombre del color	Valor hexadecimal	Valores RGB	Color
AliceBlue	#F0F8FF	rgb(240,248,255)	
AntiqueWhite	#FAEBD7	rgb(250,235,215)	
Aqua	#00FFFF	rgb(0,255,255)	
Aquamarine	#7FFFD4	rgb(127,255,212)	
Azure	#F0FFFF	rgb(240,255,255)	
Beige	#F5F5DC	rgb(245,245,220)	
Bisque	#FFE4C4	rgb(255,228,196)	
Black	#000000	rgb(0,0,0)	
BlanchedAlmond	#FFEBCD	rgb(255,235,205)	
Blue	#0000FF	rgb(0,0,255)	
BlueViolet	#8A2BE2	rgb(138,43,226)	
Brown	#A52A2A	rgb(165,42,42)	
BurlyWood	#DEB887	rgb(222,184,135)	
CadetBlue	#5F9EA0	rgb(95,158,160)	

Chartreuse	#7FFF00	rgb(127,255,0)	
Chocolate	#D2691E	rgb(210,105,30)	
Coral	#FF7F50	rgb(255,127,80)	
CornflowerBlue	#6495ED	rgb(100,149,237)	
Cornsilk	#FFF8DC	rgb(255,248,220)	
Crimson	#DC143C	rgb(220,20,60)	
Cyan	#00FFFF	rgb(0,255,255)	
DarkBlue	#00008B	rgb(0,0,139)	
DarkCyan	#008B8B	rgb(0,139,139)	
DarkGoldenRod	#B8860B	rgb(184,134,11)	
DarkGray	#A9A9A9	rgb(169,169,169)	
DarkGrey	#A9A9A9	rgb(169,169,169)	
DarkGreen	#006400	rgb(0,100,0)	
DarkKhaki	#BDB76B	rgb(189,183,107)	
DarkMagenta	#8B008B	rgb(139,0,139)	
DarkOliveGreen	#556B2F	rgb(85,107,47)	
DarkOrange	#FF8C00	rgb(255,140,0)	
DarkOrchid	#9932CC	rgb(153,50,204)	
DarkRed	#8B0000	rgb(139,0,0)	
DarkSalmon	#E9967A	rgb(233,150,122)	
DarkSeaGreen	#8FBC8F	rgb(143,188,143)	
DarkSlateBlue	#483D8B	rgb(72,61,139)	
DarkSlateGray	#2F4F4F	rgb(47,79,79)	
DarkSlateGrey	#2F4F4F	rgb(47,79,79)	
DarkTurquoise	#00CED1	rgb(0,206,209)	
DarkViolet	#9400D3	rgb(148,0,211)	
DeepPink	#FF1493	rgb(255,20,147)	
DeepSkyBlue	#00BFFF	rgb(0,191,255)	
DimGray	#696969	rgb(105,105,105)	
DimGrey	#696969	rgb(105,105,105)	
DodgerBlue	#1E90FF	rgb(30,144,255)	
FireBrick	#B22222	rgb(178,34,34)	
FloralWhite	#FFFAF0	rgb(255,250,240)	
ForestGreen	#228B22	rgb(34,139,34)	

Fuchsia	#FF00FF	rgb(255,0,255)	
Gainsboro	#DCDCDC	rgb(220,220,220)	
GhostWhite	#F8F8FF	rgb(248,248,255)	
Gold	#FFD700	rgb(255,215,0)	
GoldenRod	#DAA520	rgb(218,165,32)	
Gray	#808080	rgb(128,128,128)	
Grey	#808080	rgb(128,128,128)	
Green	#008000	rgb(0,128,0)	
GreenYellow	#ADFF2F	rgb(173,255,47)	
HoneyDew	#F0FFF0	rgb(240,255,240)	
HotPink	#FF69B4	rgb(255,105,180)	
IndianRed	#CD5C5C	rgb(205,92,92)	
Indigo	#4B0082	rgb(75,0,130)	
Ivory	#FFFFFF0	rgb(255,255,240)	
Khaki	#F0E68C	rgb(240,230,140)	
Lavender	#E6E6FA	rgb(230,230,250)	
LavenderBlush	#FFF0F5	rgb(255,240,245)	
LawnGreen	#7CFC00	rgb(124,252,0)	
LemonChiffon	#FFFACD	rgb(255,250,205)	
LightBlue	#ADD8E6	rgb(173,216,230)	
LightCoral	#F08080	rgb(240,128,128)	
LightCyan	#E0FFFF	rgb(224,255,255)	
LightGoldenRodYellow	#FAFAD2	rgb(250,250,210)	
LightGray	#D3D3D3	rgb(211,211,211)	
LightGrey	#D3D3D3	rgb(211,211,211)	
LightGreen	#90EE90	rgb(144,238,144)	
LightPink	#FFB6C1	rgb(255,182,193)	
LightSalmon	#FFA07A	rgb(255,160,122)	
LightSeaGreen	#20B2AA	rgb(32,178,170)	
LightSkyBlue	#87CEFA	rgb(135,206,250)	
LightSlateGray	#778899	rgb(119,136,153)	
LightSlateGrey	#778899	rgb(119,136,153)	
LightSteelBlue	#B0C4DE	rgb(176,196,222)	
LightYellow	#FFFFE0	rgb(255,255,224)	

Lime	#00FF00	rgb(0,255,0)	
LimeGreen	#32CD32	rgb(50,205,50)	
Linen	#FAF0E6	rgb(250,240,230)	
Magenta	#FF00FF	rgb(255,0,255)	
Maroon	#800000	rgb(128,0,0)	
MediumAquaMarine	#66CDAA	rgb(102,205,170)	
MediumBlue	#0000CD	rgb(0,0,205)	
MediumOrchid	#BA55D3	rgb(186,85,211)	
MediumPurple	#9370DB	rgb(147,112,219)	
MediumSeaGreen	#3CB371	rgb(60,179,113)	
MediumSlateBlue	#7B68EE	rgb(123,104,238)	
MediumSpringGreen	#00FA9A	rgb(0,250,154)	
MediumTurquoise	#48D1CC	rgb(72,209,204)	
MediumVioletRed	#C71585	rgb(199,21,133)	
MidnightBlue	#191970	rgb(25,25,112)	
MintCream	#F5FFFA	rgb(245,255,250)	
MistyRose	#FFE4E1	rgb(255,228,225)	
Moccasin	#FFE4B5	rgb(255,228,181)	
NavajoWhite	#FFDEAD	rgb(255,222,173)	
Navy	#000080	rgb(0,0,128)	
OldLace	#FDF5E6	rgb(253,245,230)	
Olive	#808000	rgb(128,128,0)	
OliveDrab	#6B8E23	rgb(107,142,35)	
Orange	#FFA500	rgb(255,165,0)	
OrangeRed	#FF4500	rgb(255,69,0)	
Orchid	#DA70D6	rgb(218,112,214)	
PaleGoldenRod	#EEE8AA	rgb(238,232,170)	
PaleGreen	#98FB98	rgb(152,251,152)	
PaleTurquoise	#AFEEEE	rgb(175,238,238)	
PaleVioletRed	#DB7093	rgb(219,112,147)	
PapayaWhip	#FFEFD5	rgb(255,239,213)	
PeachPuff	#FFDAB9	rgb(255,218,185)	
Peru	#CD853F	rgb(205,133,63)	
Pink	#FFC0CB	rgb(255,192,203)	

Plum	#DDA0DD	rgb(221,160,221)	
PowderBlue	#B0E0E6	rgb(176,224,230)	
Purple	#800080	rgb(128,0,128)	
RebeccaPurple	#663399	rgb(102,51,153)	
Red	#FF0000	rgb(255,0,0)	
RosyBrown	#BC8F8F	rgb(188,143,143)	
RoyalBlue	#4169E1	rgb(65,105,225)	
SaddleBrown	#8B4513	rgb(139,69,19)	
Salmon	#FA8072	rgb(250,128,114)	
SandyBrown	#F4A460	rgb(244,164,96)	
SeaGreen	#2E8B57	rgb(46,139,87)	
SeaShell	#FFF5EE	rgb(255,245,238)	
Sienna	#A0522D	rgb(160,82,45)	
Silver	#C0C0C0	rgb(192,192,192)	
SkyBlue	#87CEEB	rgb(135,206,235)	
SlateBlue	#6A5ACD	rgb(106,90,205)	
SlateGray	#708090	rgb(112,128,144)	
SlateGrey	#708090	rgb(112,128,144)	
Snow	#FFFAFA	rgb(255,250,250)	
SpringGreen	#00FF7F	rgb(0,255,127)	
SteelBlue	#4682B4	rgb(70,130,180)	
Tan	#D2B48C	rgb(210,180,140)	
Teal	#008080	rgb(0,128,128)	
Thistle	#D8BFD8	rgb(216,191,216)	
Tomato	#FF6347	rgb(255,99,71)	
Turquoise	#40E0D0	rgb(64,224,208)	
Violet	#EE82EE	rgb(238,130,238)	
Wheat	#F5DEB3	rgb(245,222,179)	
White	#FFFFFF	rgb(255,255,255)	
WhiteSmoke	#F5F5F5	rgb(245,245,245)	
Yellow	#FFFF00	rgb(255,255,0)	
YellowGreen	#9ACD32	rgb(154,205,50)	

Además de los colores con nombre, también existe la palabra clave `transparent`, que representa un negro totalmente transparente: `rgba(0, 0, 0, 0)`.

Sección 18.3: Valor hexadecimal

Fondo

Los colores CSS también pueden representarse como un triplete hexadecimal, en el que los miembros representan los componentes rojo, verde y azul de un color. Cada uno de estos valores representa un número en el rango de 00 a FF, o de 0 a 255 en notación decimal. Pueden utilizarse valores hexadecimales en mayúsculas y/o minúsculas (es decir, `#3fc` = `#3FC` = `#33ffCC`). El navegador interpreta `#369` como `#336699`. Si no es eso lo que pretendía, sino que quería `#306090`, deberá especificarlo de forma explícitamente.

El número total de colores que pueden representarse con notación hexadecimal es 256^3 o 16.777.216.

Sintaxis

`color: #rrggbb;`

`color: #rgb;`

Valor	Descripción
rr	00 - FF para la cantidad de rojo
gg	00 - FF para la cantidad de verde
bb	00 - FF para la cantidad de azul

```
.alguna-clase {
  /* Esto equivale a utilizar la palabra clave de color "blue". */
  color: #0000FF;
}
.tambien-azul {
  /* Si desea especificar cada valor de rango con un único número, ¡puede hacerlo! Esto
  equivale a '#0000FF' (y 'azul') */
  color: #00F;
}
```

La [notación hexadecimal](#) se utiliza para especificar valores de color en el formato de color RGB, de acuerdo con la guía ['Valores de colores numéricos' del W3C](#).

Hay muchas herramientas disponibles en Internet para buscar valores de color hexadecimales (o simplemente hexadecimales).

Busca **"paleta de colores hexadecimales"** o **"selector de colores hexadecimales"** en tu navegador favorito y encontrarás un montón de opciones.

Los valores hexadecimales empiezan siempre por el símbolo almohadilla (#), tienen un máximo de seis "dígitos" y no distinguen entre mayúsculas y minúsculas. `#FFC125` y `#ffc125` son del mismo color.

Sección 18.4: Notación rgb()

RGB es un modelo de color aditivo que representa los colores como mezclas de luz roja, verde y azul. En esencia, la representación RGB es el equivalente decimal de la Notación Hexadecimal. En hexadecimal, cada número oscila entre 00 y FF, lo que equivale a 0-255 en decimal y 0%-100% en porcentajes.

```
.alguna-clase {
  /* Escalar RGB, equivalente a "azul" */
  color: rgb(0, 0, 255);
}
.tambien-azul {
  /* Valores percentiles RGB */
  color: rgb(0%, 0%, 100%);
}
```

Sintaxis

`rgb(<red>, <green>, <blue>)`

Valor	Descripción
<code><red></code>	un número entero de 0 - 255 o un porcentaje de 0 - 100%.
<code><green></code>	un número entero de 0 - 255 o un porcentaje de 0 - 100%.
<code><blue></code>	un número entero de 0 - 255 o un porcentaje de 0 - 100%.

Sección 18.5: Notación rgba()

Similar a la notación rgb(), pero con un valor alfa (opacidad) adicional.

```
.rojo {  
  /* Rojo opaco */  
  color: rgba(255, 0, 0, 1);  
}  
.rojo-50p {  
  /* Rojo semitranslúcido */  
  color: rgba(255, 0, 0, .5);  
}
```

Sintaxis

`rgba(<red>, <green>, <blue>, <alpha>);`

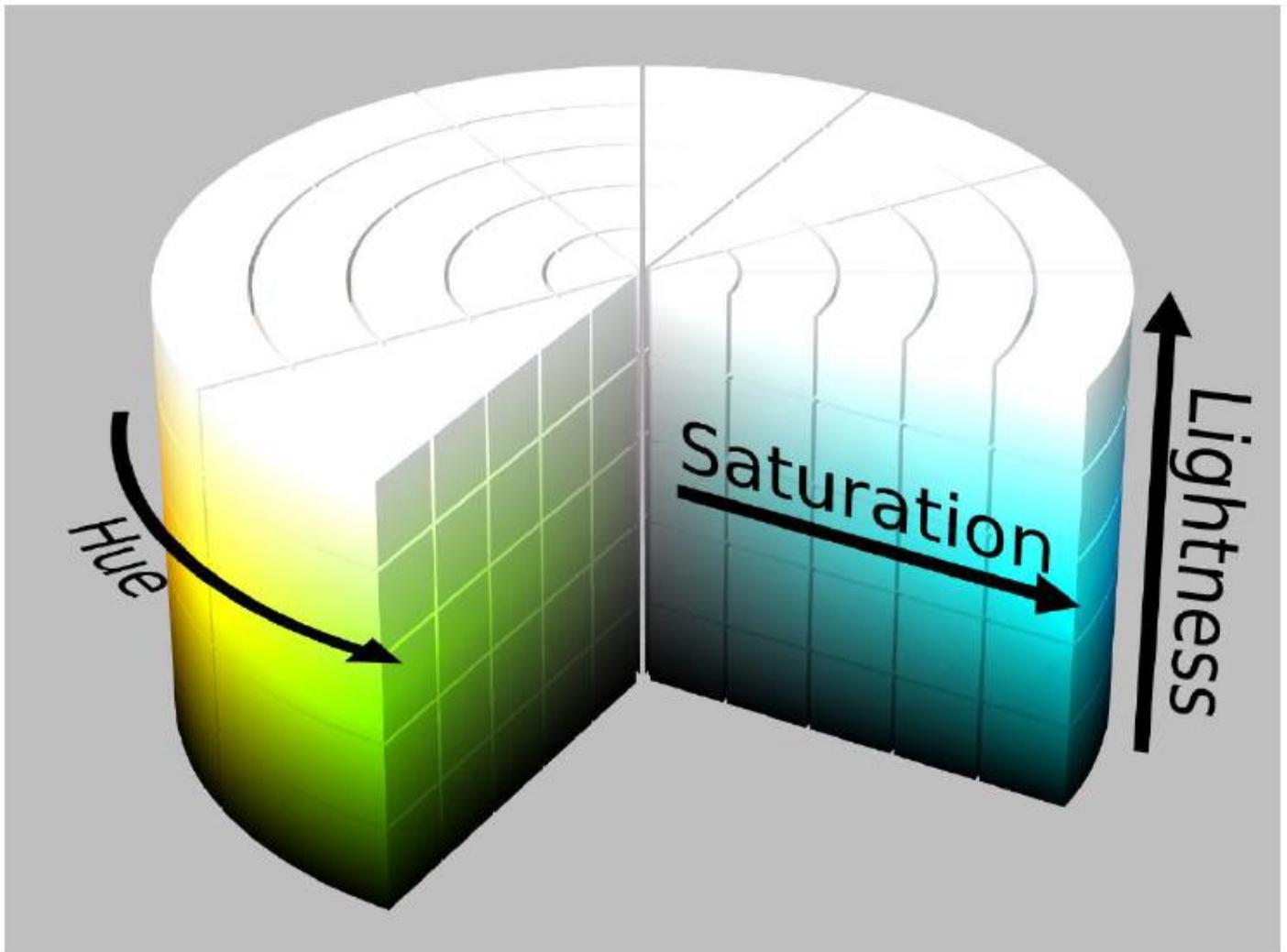
Valor	Descripción
<code><red></code>	un número entero de 0 - 255 o un porcentaje de 0 - 100%.
<code><green></code>	un número entero de 0 - 255 o un porcentaje de 0 - 100%.
<code><blue></code>	un número entero de 0 - 255 o un porcentaje de 0 - 100%.
<code><alpha></code>	un número entre 0 y 1, donde 0,0 es totalmente transparente y 1,0 totalmente opaco

Sección 18.6: Notación hsl()

HSL significa **hue**(tono) ("qué color"), **saturation**(saturación) ("cuánto color") y **lightness**(luminosidad) ("cuánto blanco").

El tono se representa como un ángulo de 0° a 360° (sin unidades), mientras que la saturación y la luminosidad se representan como porcentajes.

```
p {  
  color: hsl(240, 100%, 50%); /* Azul */  
}
```



Sintaxis

color: **hsl**(<hue>, <saturation>%, <lightness>%);

Valor	Descripción
<hue>	especificado en grados alrededor de la rueda de color (sin unidades), donde 0° es rojo, 60° es amarillo, 120° es verde, 180° es cian, 240° es azul, 300° es magenta y 360° es rojo
<saturation>	especificado en porcentaje, donde 0% es totalmente desaturado (escala de grises) y 100% es totalmente saturado (colores vivos)
<lightness>	especificado en porcentaje donde 0% es totalmente negro y 100% es totalmente blanco

Notas

- Una saturación del 0% siempre produce un color en escala de grises; cambiar el tono no tiene ningún efecto.
- Una luminosidad del 0% siempre produce negro, y del 100% siempre produce blanco; cambiar el tono o la saturación no tiene ningún efecto.

Sección 18.7: Notación hsla()

Similar a la notación hsl(), pero con un valor alfa (opacidad) añadido.

```
hsla(240, 100%, 50%, 0) /* transparente */
hsla(240, 100%, 50%, 0.5) /* azul semitranslúcido */
hsla(240, 100%, 50%, 1) /* azul totalmente opaco */
```

Sintaxis

hsla(<hue>, <saturation>%, <lightness>%, <alpha>);

Valor	Descripción
<hue>	especificado en grados alrededor de la rueda de color (sin unidades), donde 0° es rojo, 60° es amarillo, 120° es verde, 180° es cian, 240° es azul, 300° es magenta y 360° es rojo.
<saturation>	porcentaje en el que 0% es totalmente desaturado (escala de grises) y 100% es totalmente saturado (colores vivos).
<lightness>	porcentaje donde 0% es totalmente negro y 100% es totalmente blanco
<alpha>	un número de 0 a 1, donde 0 es totalmente transparente y 1 totalmente opaco

Capítulo 19: Opacidad

Sección 19.1: Propiedad opacity

La opacidad de un elemento puede establecerse utilizando la propiedad `opacity`. pueden oscilar entre `0.0` (transparente) y `1.0` (opaco).

Ejemplo de uso

```
<div style="opacity:0.8;">
  Es un elemento parcialmente transparente
</div>
```

Valor de la propiedad	Transparencia
<code>opacity: 1.0;</code>	Opaco
<code>opacity: 0.75;</code>	25% transparente (75% opaco)
<code>opacity: 0.5;</code>	50% transparente (50% opaco)
<code>opacity: 0.25;</code>	75% transparente (25% opaco)
<code>opacity: 0.0;</code>	Transparente

Sección 19.2: Compatibilidad IE para 'opacity'.

Para utilizar la opacidad en todas las versiones de IE, la orden es:

```
.elemento-transparente {
  /* para IE 8 & 9 */
  -ms-filter:"progid:DXImageTransform.Microsoft.Alpha(Opacity=60)"; // IE8
  /* funciona también en IE 8 y 9, pero también 5, 6, 7 */
  filter: alpha(opacity=60); // IE 5-7
  /* Navegadores modernos */
  opacity: 0.6;
}
```

Capítulo 20: Unidades de longitud

Unidad	Descripción
%	Definir los tamaños en función de los objetos padre o del objeto actual en función de la propiedad
em	Relativo al tamaño de la fuente del elemento (2em significa 2 veces el tamaño de la fuente actual)
rem	Relativo al tamaño de fuente del elemento raíz
vw	Relativo al 1% de la anchura de la ventana*
vh	Relativo al 1% de la altura de la ventana*
vmin	En relación con el 1% de la dimensión menor* de la ventana
vmax	En relación con el 1% de la dimensión mayor* de la ventana
cm	centímetros
mm	milímetros
in	pulgadas (1in = 96px = 2,54cm)
px	píxeles (1px = 1/96 de 1 pulgada)
pt	puntos (1pt = 1/72 de 1 pulgada)
pc	picas (1pc = 12 pt)
s	segundos (utilizados para animaciones y transiciones)
ms	milisegundos (utilizado para animaciones y transiciones)
ex	Relativo a la altura x de la fuente actual
ch	En función de la anchura del carácter cero (0)
fr	unidad fraccionaria (utilizada para CSS Grid Layout)

Una medida de distancia CSS es un número seguido inmediatamente de una unidad de longitud (px, em, pc, in, ...)

CSS admite varias unidades de medida de longitud. Pueden ser absolutas o relativas.

Sección 20.1: Creación de elementos escalables mediante rems y ems

Version \geq 3

Puedes usar `rem` definido por el `font-size` de tu etiqueta `html` para estilizar elementos estableciendo su `font-size` a un valor de `rem` y usar `em` dentro del elemento para crear elementos que escalen con tu `font-size` global.

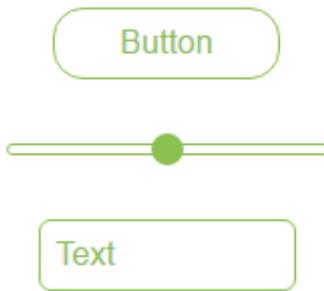
HTML

```
<input type="button" value="Botón">
<input type="range">
<input type="text" value="Texto">
```

CSS correspondiente

```
html {
  font-size: 16px;
}
input[type="button"] {
  font-size: 1rem;
  padding: 0.5em 2em;
}
input[type="range"] {
  font-size: 1rem;
  width: 10em;
}
input[type="text"] {
  font-size: 1rem;
  padding: 0.5em;
}
```

Posible resultado



Sección 20.2: Tamaño de letra con rem

CSS3 introduce algunas unidades nuevas, entre ellas la unidad `rem`, que significa "raíz em". Veamos cómo funciona `rem`.

En primer lugar, veamos las diferencias entre `em` y `rem`.

- `em`: Relativo al tamaño de fuente del padre. Esto causa el problema compuesto.
- `rem`: Relativo al tamaño de fuente de la raíz o del elemento `<html>`. Esto significa que es posible declarar un único tamaño de fuente para el elemento `html` y definir que todas las unidades `rem` sean un porcentaje del mismo.

El principal problema de utilizar `rem` para el tamaño de las fuentes es que los valores son algo difíciles de utilizar. He aquí un ejemplo de algunos tamaños de fuente comunes expresados en unidades `rem`, suponiendo que el tamaño base es 16px:

- 10px = 0.625rem
- 12px = 0.75rem
- 14px = 0.875rem
- 16px = 1rem (base)
- 18px = 1.125rem
- 20px = 1.25rem
- 24px = 1.5rem
- 30px = 1.875rem
- 32px = 2rem

CÓDIGO

Version \geq 3

```
html {  
    font-size: 16px;  
}  
h1 {  
    font-size: 2rem; /* 32px */  
}  
p {  
    font-size: 1rem; /* 16px */  
}  
li {  
    font-size: 1.5em; /* 24px */  
}
```

Sección 20.3: vmin y vmax

- **vmin**: Relativo al 1 por ciento de la dimensión más pequeña de la ventana.
- **vmax**: Relativo al 1 por ciento de la dimensión mayor de la ventana.

En otras palabras, 1 **vmin** es igual al menor de 1 *vh* y 1 *vw*

1 **vmax** es igual al mayor de 1 *vh* y 1 *vw*

Nota: **vmax** no está soportado en:

- cualquier versión de Internet Explorer
- Safari antes de la versión 6.1

Sección 20.4: vh y vw

CSS3 introdujo dos unidades para representar el tamaño.

- **vh**, que significa **viewport height** (altura de la ventana) es relativa al 1% de la altura de la ventana.
- **vw**, que significa **viewport width** (anchura de la ventana) es relativa al 1% de la anchura de la ventana.

Version \geq 3

```
div {  
    width: 20vw;  
    height: 20vh;  
}
```

Arriba, el tamaño del div ocupa el 20% de la anchura y la altura de la ventana.

Sección 20.5: usando el porcentaje %

Una de las unidades útiles a la hora de crear una aplicación responsive.

Su tamaño depende de su contenedor padre.

Ecuación

$$(\text{Anchura del contenedor padre}) * (\text{Porcentaje (\%)}) = \text{Salida}$$

Por ejemplo

El *Padre* tiene **100px** de ancho mientras que el *Hijo* tiene **50%**.

En la salida, el ancho del hijo será la mitad (50%) del ancho del padre, que es 50px.

HTML

```
<div class="padre">
  PADRE
  <div class="hijo">
    HIJO
  </div>
</div>
```

CSS

```
<style>
*{
  color: #CCC;
}
.padre{
  background-color: blue;
  width: 100px;
}
.hijo{
  background-color: green;
  width: 50%;
}
</style>
```

SALIDA



Capítulo 21: Pseudoelementos

pseudoelemento	Descripción
<code>::after</code>	Insertar contenido después del contenido de un elemento
<code>::before</code>	Insertar contenido antes del contenido de un elemento
<code>::first-letter</code>	Selecciona la primera letra de cada elemento
<code>::first-line</code>	Selecciona la primera línea de cada elemento
<code>::selection</code>	Coincide con la parte de un elemento seleccionada por un usuario
<code>::backdrop</code>	Se utiliza para crear un telón de fondo que oculta el documento subyacente de un elemento de la capa superior
<code>::placeholder</code>	Permite dar estilo al texto del marcador de posición de un elemento de formulario (Experimental)
<code>::marker</code>	Para aplicar atributos de estilo de lista a un elemento determinado (Experimental)
<code>::spelling-error</code>	Representa un segmento de texto que el navegador ha marcado como mal escrito (Experimental)
<code>::grammar-error</code>	Representa un segmento de texto que el navegador ha marcado como gramaticalmente incorrecto. (Experimental)

Los pseudoelementos, al igual que las pseudoclases, se añaden a los selectores CSS, pero en lugar de describir un estado especial, permiten definir el alcance y el estilo de determinadas partes de un elemento html.

Por ejemplo, el pseudoelemento `::first-letter` sólo se dirige a la primera letra de un elemento de bloque especificado por el selector.

Sección 21.1: Pseudoelementos

Los pseudoelementos se añaden a los selectores, pero en lugar de describir un estado especial, permiten aplicar estilo a determinadas partes de un documento.

El atributo `content` es necesario para que los pseudoelementos se muestren; sin embargo, el atributo puede tener un valor vacío (por ejemplo, `content: ""`).

```
div::after {
  content: 'after';
  color: red;
  border: 1px solid red;
}
div {
  color: black;
  border: 1px solid black;
  padding: 1px;
}
div::before {
  content: 'before';
  color: green;
  border: 1px solid green;
}
```



Sección 21.2: Pseudoelementos en listas

Los pseudoelementos se utilizan a menudo para cambiar el aspecto de las listas (sobre todo para las listas desordenadas, ul).

El primer paso es eliminar las viñetas de la lista por defecto:

```
ul {  
    list-style-type: none;  
}
```

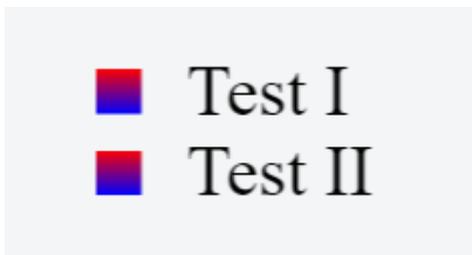
A continuación, añadida el estilo personalizado. En este ejemplo, crearemos cajas de gradiente para las viñetas.

```
li:before {  
    content: "";  
    display: inline-block;  
    margin-right: 10px;  
    height: 10px;  
    width: 10px;  
    background: linear-gradient(red, blue);  
}
```

HTML

```
<ul>  
    <li>Test I</li>  
    <li>Test II</li>  
</ul>
```

Resultado



Capítulo 22: Posicionamiento

Parámetro	Detalles
<code>static</code>	Valor por defecto. Los elementos se muestran en orden, tal y como aparecen en el flujo del documento. Las propiedades <code>top</code> , <code>right</code> , <code>bottom</code> , <code>left</code> y <code>z-index</code> no se aplican
<code>relative</code>	El elemento se posiciona en relación con su posición normal, por lo que <code>left:20px</code> añade 20 píxeles a la posición IZQUIERDA
<code>fixed</code>	El elemento se posiciona respecto a la ventana del navegador
<code>absolute</code>	El elemento se posiciona respecto a su primer elemento antepasado posicionado (no estático)
<code>initial</code>	Establece esta propiedad a su valor por defecto.
<code>inherit</code>	Hereda esta propiedad de su elemento padre.
<code>sticky</code>	Característica experimental. Se comporta como <code>position: static</code> dentro de su padre hasta que se alcanza un determinado umbral de desplazamiento y entonces actúa como <code>position: fixed</code> .
<code>unset</code>	Combinación de inicial y heredar. Más información aquí .

Sección 22.1: Elementos superpuestos con z-index

Para cambiar el [orden de apilamiento](#) por defecto de los elementos posicionados (propiedad `position` establecida como `relative`, `absolute` o `fixed`), utilice la propiedad `z-index`.

Cuanto mayor sea el índice z, más arriba se colocará en el contexto de apilamiento (en el eje z).

Ejemplo

En el ejemplo siguiente, un valor de `z-index` de 3 coloca el verde en la parte superior, un `z-index` de 2 coloca el rojo justo debajo, y un `z-index` de 1 pone el azul debajo.

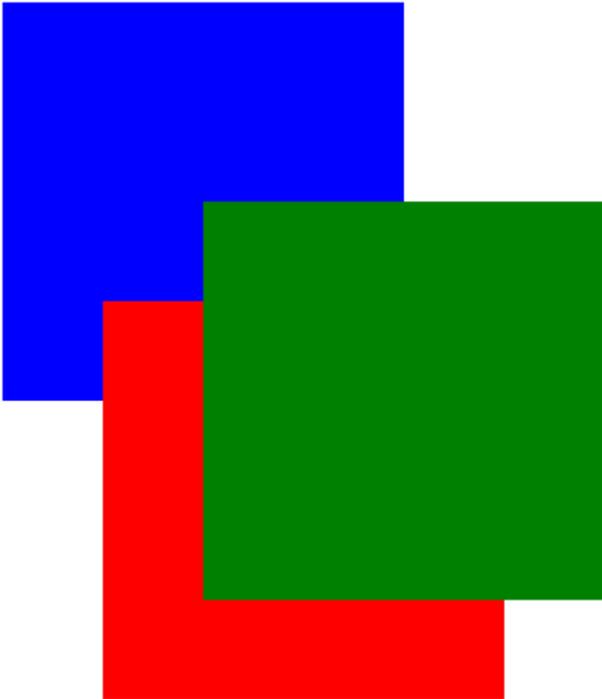
HTML

```
<div id="div1"></div>
<div id="div2"></div>
<div id="div3"></div>
```

CSS

```
div {  
  position: absolute;  
  height: 200px;  
  width: 200px;  
}  
div#div1 {  
  z-index: 1;  
  left: 0px;  
  top: 0px;  
  background-color: blue;  
}  
div#div2 {  
  z-index: 3;  
  left: 100px;  
  top: 100px;  
  background-color: green;  
}  
div#div3 {  
  z-index: 2;  
  left: 50px;  
  top: 150px;  
  background-color: red;  
}
```

Esto crea el siguiente efecto:



Sintaxis

`z-index: [number] | auto;`

Parámetro

`number`

`auto`

Detalles

Un valor entero. Un número más alto está más arriba en la pila de `z-index`. `0` es el valor por defecto. Los valores negativos están permitidos.

Da al elemento el mismo contexto de apilamiento que su padre. **(Por defecto)**

Observaciones

Todos los elementos se disponen en un eje 3D en CSS, incluido un eje de profundidad, medido por la propiedad `z-index`. `z-index` sólo funciona en elementos posicionados: (ver: [¿Por qué z-index necesita una posición definida para funcionar?](#)). El único valor en el que se ignora es el valor por defecto, `static`.

Más información sobre la propiedad `z-index` y los contextos de apilamiento en la [Especificación CSS](#) sobre la presentación en capas y en la [Red de desarrolladores de Mozilla](#).

Sección 22.2: Posición absoluta

Cuando se utiliza el posicionamiento absoluto la caja del elemento deseado se saca del *Flujo Normal* y ya no afecta a la posición de los demás elementos de la página. Propiedades de desplazamiento:

1. `top`
2. `left`
3. `right`
4. `bottom`

especifica que el elemento debe aparecer en relación con su siguiente elemento contenedor no estático.

```
.abspos{
  position: absolute;
  top: 0px;
  left: 500px;
}
```

Este código moverá la caja que contiene el elemento con el atributo `class="abspos"` hacia abajo 0px y hacia la derecha 500px en relación a su elemento contenedor.

Sección 22.3: Posición fija

Definiendo la posición como fija podemos sacar un elemento del flujo del documento y fijar su posición relativa a la ventana del navegador. Un uso obvio es cuando queremos que algo sea visible cuando nos desplazamos al final de una página larga.

```
#stickyDiv {
  position: fixed;
  top: 10px;
  left: 10px;
}
```

Sección 22.4: Posición relativa

El posicionamiento relativo desplaza el elemento en relación con el lugar que ocuparía en flujo normal. Propiedades de desplazamiento:

1. `top`
2. `left`
3. `right`
4. `bottom`

se utilizan para indicar la distancia a la que hay que desplazar el elemento desde donde se encontraría en flujo normal.

```
.relpos{
  position: relative;
  top: 20px;
  left: 30px;
}
```

Este código moverá la caja que contiene el elemento con atributo `class="relpos"` 20px hacia abajo y 30px hacia la derecha desde donde habría estado en flujo normal.

Sección 22.5: Posición estática

La posición por defecto de un elemento es `static`. Citando a [MDN](#):

Esta palabra clave permite que el elemento utilice el comportamiento normal, es decir, que se disponga en su posición actual en el flujo. Las propiedades `top`, `right`, `bottom`, `left` y `z-index` no se aplican.

```
.elemento {  
    position:static;  
}
```

Capítulo 23: Control del diseño

Valor	Efecto
<code>none</code>	Oculto el elemento y evita que ocupe espacio
<code>block</code>	Elemento de bloque, ocupa el 100% de la anchura disponible, se rompe después del elemento
<code>inline</code>	Elemento en línea, no ocupa ancho, no hay ruptura después del elemento
<code>inline-block</code>	Tomando propiedades especiales tanto de elementos en línea como de bloque, sin ruptura, pero puede tener anchura
<code>inline-flex</code>	Muestra un elemento como contenedor flex de nivel inline
<code>inline-table</code>	El elemento se muestra como una tabla en línea
<code>grid</code>	Se comporta como un elemento de bloque y dispone su contenido según el modelo de rejilla
<code>flex</code>	Se comporta como un elemento de bloque y distribuye su contenido según el modelo flexbox
<code>inherit</code>	Hereda el valor del elemento padre
<code>initial</code>	Restablece el valor predeterminado tomado de los comportamientos descritos en las especificaciones HTML o de la hoja de estilos por defecto del navegador/usuario
<code>table</code>	Se comporta como el elemento <code>table</code> de HTML
<code>table-cell</code>	Que el elemento se comporte como un elemento <code><td></code>
<code>table-column</code>	Que el elemento se comporte como un elemento <code><col></code>
<code>table-row</code>	Deja que el elemento se comporte como un elemento <code><tr></code>
<code>list-item</code>	Deja que el elemento se comporte como un elemento <code></code>

Sección 23.1: La propiedad display

La propiedad CSS `display` es fundamental para controlar el diseño y el flujo de un documento HTML. La mayoría de los elementos tienen un valor `display` predeterminado de `block` o `inline` (aunque algunos elementos tienen otros valores predeterminados).

Inline

Un elemento `inline` sólo ocupa el ancho necesario. Se apila horizontalmente con otros elementos del mismo tipo y no puede contener otros elementos no alineados.

```
<span> ¡Menudo texto en <b>negrita</b>! </span>
```

¡Menudo texto en **negrita**!

Como se ha demostrado anteriormente, dos elementos `inline`, `` y ``, están en línea (de ahí el nombre) y no rompen el flujo del texto.

Block

Un elemento `block` ocupa la anchura máxima disponible de su elemento padre. Comienza con una nueva línea y, a diferencia de los elementos `inline`, no restringe el tipo de elementos que puede contener.

```
<div>¡Hola Mundo!</div><div>¡Esto es un ejemplo!</div>
```

¡Hola Mundo!
¡Esto es un ejemplo!

El elemento `div` es de nivel de bloque por defecto, y como se muestra arriba, los dos elementos de `block` se apilan verticalmente y, a diferencia de los elementos `inline`, el flujo del texto se interrumpe.

Inline Block

El valor `inline-block` nos ofrece lo mejor de los dos mundos: integra el elemento en el flujo del texto al tiempo que nos permite utilizar el `padding`, `margin`, `height` y propiedades similares que no tienen efecto visible en los elementos `inline`.

Los elementos con este valor de visualización actúan como si fueran texto normal y, en consecuencia, se ven afectados por las reglas que controlan el flujo de texto, como `text-align`. Por defecto, también se reducen al tamaño más pequeño posible para acomodar su contenido.

```
<!-- Inline: lista desordenada -->
<style>
li {
  display: inline;
  background: lightblue;
  padding:10px;
  border-width:2px;
  border-color:black;
  border-style:solid;
}
</style>
<ul>
  <li> Primer Elemento </li>
  <li> Segundo Elemento </li>
  <li> Tercer Elemento </li>
</ul>
```



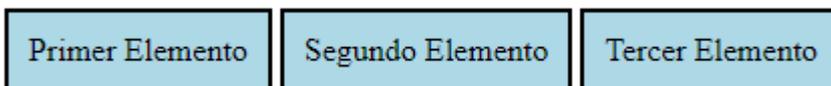
```
<!-- bloque: lista desordenada -->
<style>
li {
  display: block;
  background: lightblue;
  padding:10px;
  border-width:2px;
  border-color:black;
  border-style:solid;
}
</style>
<ul>
  <li> Primer Elemento </li>
  <li> Segundo Elemento </li>
  <li> Tercer Elemento </li>
</ul>
```



```

<!-- Inline-block: lista desordenada -->
<style>
li {
    display: inline-block;
    background: lightblue;
    padding:10px;
    border-width:2px;
    border-color:black;
    border-style:solid;
}
</style>
<ul>
    <li> Primer Elemento </li>
    <li> Segundo Elemento </li>
    <li> Tercer Elemento </li>
</ul>

```



none

Un elemento al que se le haya dado el valor none a su propiedad `display` no se mostrará en absoluto.

Por ejemplo, vamos a crear un elemento `div` con el id `miDiv`:

```
<div id="miDiv"></div>
```

Ahora puede marcarse como no mostrada mediante la siguiente regla CSS:

```
#miDiv {
    display: none;
}
```

Cuando un elemento ha sido configurado para ser `display:none`, el navegador ignora cualquier otra propiedad de diseño para ese elemento específico (tanto `position` como `float`). No se mostrará ninguna caja para ese elemento y su existencia en html no afecta a la posición de los elementos siguientes.

Tenga en cuenta que esto es diferente de establecer la propiedad de `visibility` en `hidden`. Establecer `visibility: hidden`; para un elemento no mostraría el elemento en la página, pero el elemento seguiría ocupando el espacio en el proceso de renderizado como si fuera visible. Por lo tanto, esto afectará a cómo se muestran los siguientes elementos en la página.

El valor `none` de la propiedad `display` se utiliza habitualmente junto con JavaScript para mostrar u ocultar elementos a voluntad, eliminando la necesidad de borrarlos y volver a crearlos.

Sección 23.2: Para obtener la estructura de la tabla antigua utilizando div

Ésta es la estructura normal de una tabla HTML.

```

<style>
    table {
        width: 100%;
    }
</style>
<table>
    <tr>
        <td>
            Soy una tabla
        </td>
    </tr>
</table>

```

Puede realizar la misma implementación de la siguiente manera.

```
<style>
  .table-div {
    display: table;
  }
  .table-row-div {
    display: table-row;
  }
  .table-cell-div {
    display: table-cell;
  }
</style>
<div class="table-div">
  <div class="table-row-div">
    <div class="table-cell-div">
      Ahora me porto como una tabla
    </div>
  </div>
</div>
```

Capítulo 24: Grid

El diseño Grid es un nuevo y potente sistema de diseño CSS que permite dividir el contenido de una página web en filas y columnas.

Sección 24.1: Ejemplo básico

Propiedad	Valores posibles
<code>display</code>	<code>grid</code> / <code>inline-grid</code>

La rejilla CSS se define como una propiedad de visualización. Se aplica únicamente a un elemento padre y a sus hijos inmediatos.

Considere el siguiente marcado:

```
<section class="contenedor">
  <div class="item1">item1</div>
  <div class="item2">item2</div>
  <div class="item3">item3</div>
  <div class="item4">item4</div>
</section>
```

La forma más sencilla de definir la estructura de marcado anterior como una cuadrícula es simplemente establecer su propiedad `display` en `grid`:

```
.contenedor {
  display: grid;
}
```

Sin embargo, esto provocará invariablemente que todos los elementos hijos se colapsen unos sobre otros. Esto se debe a que los niños no saben actualmente cómo colocarse dentro de la cuadrícula. Pero podemos decírselo explícitamente.

En primer lugar, tenemos que indicar al elemento `.contenedor` de la rejilla cuántas filas y columnas formarán su estructura y podemos hacerlo utilizando las propiedades `grid-columns` y `grid-rows` (nótese la pluralización):

```
.contenedor {
  display: grid;
  grid-columns: 50px 50px 50px;
  grid-rows: 50px 50px;
}
```

Sin embargo, eso no nos ayuda mucho porque necesitamos dar un orden a cada elemento hijo. Podemos hacerlo especificando los valores `grid-row` y `grid-column` que le dirán dónde se sitúa en la cuadrícula:

```
.contenedor .item1 {
  grid-column: 1;
  grid-row: 1;
}
.contenedor .item2 {
  grid-column: 2;
  grid-row: 1;
}
.contenedor .item3 {
  grid-column: 1;
  grid-row: 2;
}
.contenedor .item4 {
  grid-column: 2;
  grid-row: 2;
}
```

Al dar a cada elemento un valor de columna y fila, se identifica el orden de los elementos dentro del contenedor.

Vea un ejemplo funcional en [JSFiddle](#). Necesitarás ver esto en IE10, IE11 o Edge para que funcione ya que estos son actualmente los únicos navegadores que soportan el diseño Grid (con prefijo de proveedor `-ms-`) o habilitar un flag en Chrome, Opera y Firefox según [CanIUse](#) para poder probar con ellos.

Capítulo 25: Tablas

Sección 25.1: table-layout

La propiedad `table-layout` cambia el algoritmo que se utiliza para la disposición de una tabla.

A continuación, se muestra un ejemplo de dos tablas, ambas con un `width: 150px`:

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

La tabla de la izquierda tiene `table-layout: auto` mientras que la de la derecha tiene `table-layout: fixed`. La primera es más ancha que la especificada (210px en lugar de 150px), pero el contenido cabe. Este último toma el ancho definido de 150px, independientemente de si el contenido se desborda o no.

Valor	Descripción
<code>auto</code>	Este es el valor por defecto. Define el diseño de la tabla en función del contenido de sus celdas.
<code>fixed</code>	Este valor establece que el diseño de la tabla esté determinado por la propiedad <code>width</code> proporcionada a la tabla. Si el contenido de una celda supera esta anchura, la celda no se redimensionará, sino que dejará que el contenido se desborde.

Sección 25.2: empty-cells

La propiedad `empty-cells` determina si las celdas sin contenido deben mostrarse o no. Esto no tiene ningún efecto a menos que `border-collapse` esté ajustado a `separate`.

A continuación, se muestra un ejemplo con dos tablas con diferentes valores en la propiedad `empty-cells`:

First name	Last name	Homeworld
Luke		Tatooine
Leia	Organa	

First name	Last name	Homeworld
Luke		Tatooine
Leia	Organa	

La tabla de la izquierda tiene `empty-cells: show` mientras que la de la derecha tiene `empty-cells: hide`. El primero muestra las celdas vacías, mientras que el segundo no.

Valor	Descripción
<code>show</code>	Este es el valor por defecto. Muestra las celdas, aunque estén vacías.
<code>hide</code>	Este valor oculta completamente una celda si no hay contenido en ella.

Más información:

- <https://www.w3.org/TR/CSS21/tables.html#empty-cells>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/empty-cells>
- <https://codepen.io/SitePoint/pen/vYGJVY>
- <https://css-tricks.com/almanac/properties/e/empty-cells/>

Sección 25.3: border-collapse

La propiedad `border-collapse` determina si los bordes de las tablas deben separarse o fusionarse.

A continuación, un ejemplo de dos tablas con diferentes valores para la propiedad `border-collapse`:

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

La tabla de la izquierda tiene **border-collapse**: `separate` mientras que la de la derecha tiene **border-collapse**: `collapse`.

Valor	Descripción
<code>separate</code>	Este es el valor por defecto. Hace que los bordes de la tabla estén separados entre sí.
<code>collapse</code>	Este valor establece que los bordes de la tabla se fusionen, en lugar de ser distintos.

Sección 25.4: border-spacing

La propiedad **border-spacing** determina el espaciado entre celdas. Esto no tiene efecto a menos que **border-collapse** sea `separate`.

A continuación, se muestra un ejemplo de dos tablas con diferentes valores de la propiedad **border-spacing**:

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

La tabla de la izquierda tiene **border-spacing**: `2px` (por defecto) mientras que la de la derecha tiene **border-spacing**: `8px`.

Valor	Descripción
<code><longitud></code>	Este es el comportamiento por defecto, aunque el valor exacto puede variar entre navegadores.
<code><longitud> <longitud></code>	Esta sintaxis permite especificar valores horizontales y verticales por separado, respectivamente.

Sección 25.5: caption-side

La propiedad **caption-side** determina la posición vertical del elemento `<caption>` dentro de una tabla. Esto no tiene efecto si dicho elemento no existe.

A continuación, se muestra un ejemplo con dos tablas con diferentes valores establecidos en la propiedad **caption-side**:

Star Wars figures		
First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan
Star Wars figures		

La tabla de la izquierda tiene **caption-side**: `top` mientras que la de la derecha tiene **caption-side**: `bottom`.

Valor	Descripción
<code>top</code>	Este es el valor por defecto. Sitúa la leyenda encima de la tabla.
<code>bottom</code>	Este valor coloca el título debajo de la tabla.

Capítulo 26: Transiciones

Parámetro	Detalles
<code>transition-property</code>	La propiedad CSS específica cuyo cambio de valor debe ser objeto de transición (o <code>all</code> , si todas las propiedades transicionables necesitan una transición.
<code>transition-duration</code>	La duración (o período) en segundos (s) o milisegundos (ms) durante la cual la transición debe tener lugar.
<code>transition-timing-function</code>	Función que describe cómo se calculan los valores intermedios durante la transición. Los valores más comunes son <code>ease</code> , <code>ease-in</code> , <code>ease-out</code> , <code>ease-in-out</code> , <code>linear</code> , <code>cubic-bezier()</code> , <code>steps()</code> . Para más información sobre las distintas funciones de sincronización en las especificaciones del W3C .
<code>transition-delay</code>	Tiempo que debe haber transcurrido para que se inicie la transición. Puede especificarse en segundos (s) o milisegundos (ms)

Sección 26.1: Abreviatura transition

HTML

```
<div></div>
```

CSS

```
div{  
  width: 150px;  
  height:150px;  
  background-color: red;  
  transition: background-color 1s;  
}  
div:hover{  
  background-color: green;  
}
```

Este ejemplo va a cambiar el color de fondo cuando el `div` se pasa el cambio de color de fondo tendrá una duración de 1 segundo.

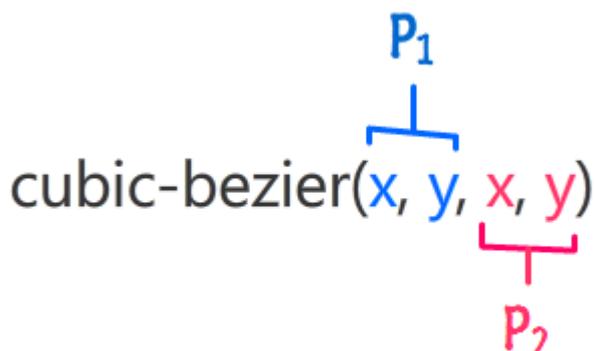
Sección 26.2: cubic-bezier

La función `cubic-bezier` es una función de temporización de transiciones que suele utilizarse para transiciones personalizadas y suaves.

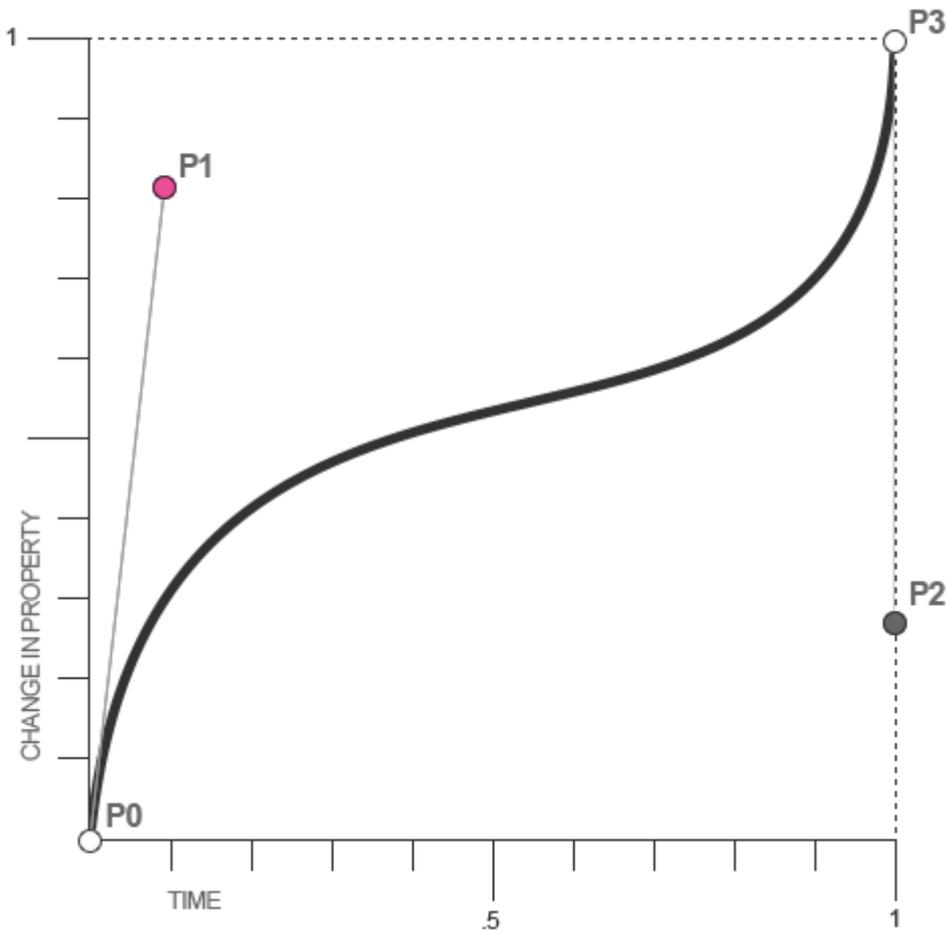
`transition-timing-function: cubic-bezier(0.1, 0.7, 1.0, 0.1);`

La función toma cuatro parámetros:

`cubic-bezier(P1_x, P1_y, P2_x, P2_y)`



Estos parámetros se asignarán a puntos que formen parte de una curva de Bézier:



Para las curvas CSS Bézier, P0 y P3 están siempre en el mismo punto. P0 está en (0,0) y P3 está en (1,1), lo que significa que los parámetros pasados a la función `cubic-bezier` sólo pueden estar entre 0 y 1.

Si pasas parámetros que no están en este intervalo, la función adoptará por defecto una transición `linear`.

Desde `cubic-bezier` es la transición más flexible en CSS, puede traducir todas las demás funciones de tiempo de transición a funciones `cubic-bezier`:

```
linear: cubic-bezier(0,0,1,1)
ease-in: cubic-bezier(0.42, 0.0, 1.0, 1.0)
ease-out: cubic-bezier(0.0, 0.0, 0.58, 1.0)
ease-in-out: cubic-bezier(0.42, 0.0, 0.58, 1.0)
```

Sección 26.3: transition (no abreviado)

CSS

```
div {
  height: 100px;
  width: 100px;
  border: 1px solid;
  transition-property: height, width;
  transition-duration: 1s, 500ms;
  transition-timing-function: linear;
  transition-delay: 0s, 1s;
}
div:hover {
  height: 200px;
  width: 200px;
}
```

HTML

`<div></div>`

- **transition-property:** Especifica las propiedades CSS para las que es el efecto de transición. En este caso, el `div` se expandirá horizontal y verticalmente al pasar el ratón por encima.
- **transition-duration:** Especifica el tiempo que tarda en completarse una transición. En el ejemplo anterior, las transiciones de altura y anchura tardarán 1 segundo y 500 milisegundos respectivamente.
- **transition-timing-function:** Especifica la curva de velocidad del efecto de transición. Un valor *linear* indica que la transición tendrá la misma velocidad de principio a fin.
- **transition-delay:** Especifica el tiempo que hay que esperar antes de que se inicie el efecto de transición. En este caso, la altura iniciará la transición inmediatamente, mientras que la anchura esperará 1 segundo.

Capítulo 27: Animaciones

Sección 27.1: Animaciones con keyframes

Para animaciones CSS de varias etapas, puede crear `@keyframes` CSS. Los keyframes permiten definir múltiples puntos de animación, llamado keyframe, para definir animaciones más complejas.

Ejemplo básico

En este ejemplo, vamos a hacer una animación de fondo básico que los ciclos entre todos los colores.

```
@keyframes fondo-arcoiris {
  0% { background-color: #ff0000; }
  8.333% { background-color: #ff8000; }
  16.667% { background-color: #ffff00; }
  25.000% { background-color: #80ff00; }
  33.333% { background-color: #00ff00; }
  41.667% { background-color: #00ff80; }
  50.000% { background-color: #00ffff; }
  58.333% { background-color: #0080ff; }
  66.667% { background-color: #0000ff; }
  75.000% { background-color: #8000ff; }
  83.333% { background-color: #ff00ff; }
  91.667% { background-color: #ff0080; }
  100.00% { background-color: #ff0000; }
}
.FondoArcoiris {
  animation: rainbow-background 5s infinite;
}
```

Aquí hay que tener en cuenta varias cosas. En primer lugar, la sintaxis `@keyframes` real.

```
@keyframes fondo-arcoiris {
```

Esto establece el nombre de la animación a `fondo-arcoiris`.

```
0% { background-color: #ff0000; }
```

Esta es la definición de un fotograma clave dentro de la animación. La primera parte, el `0%` en el caso, define dónde se encuentra el fotograma clave durante la animación. El `0%` implica que es el `0%` del tiempo total de animación desde el principio.

La animación pasará automáticamente de un fotograma clave a otro. Así, estableciendo el siguiente color de fondo en `8.333%`, la animación tomará suavemente `8.333%` del tiempo de transición entre esos fotogramas clave.

```
.FondoArcoiris {
  animation: fondo-arcoiris 5s infinite;
}
```

Este código adjunta nuestra animación a todos los elementos que tienen la clase `.FondoArcoiris`.

La propiedad de animación real toma los siguientes argumentos.

- **animation-name:** El nombre de nuestra animación. En este caso, `fondo-arcoiris`
- **animation-duration:** Cuánto durará la animación, en este caso 5 segundos.
- **animation-iteration-count (Opcional):** El número de veces que se repetirá la animación. En este caso, la animación continuará indefinidamente. Por defecto, la animación se reproducirá una vez.
- **animation-delay (Opcional):** Especifica el tiempo de espera antes de que comience la animación. Por defecto es 0 segundos, y puede tomar valores negativos. Por ejemplo, `-2s` iniciaría la animación a los 2 segundos de su bucle.
- **animation-timing-function (Opcional):** Especifica la curva de velocidad de la animación. Por defecto es `ease`, donde la animación comienza lenta, se acelera y termina lenta.

En este ejemplo concreto, los keyframes de `0%` y `100%` especifican `{ background-color: #ff0000; }`. Cuando dos o más keyframes compartan un estado, se pueden especificar en una única sentencia. En este caso, las dos líneas `0%` y `100%` podrían sustituirse por esta única línea:

```
0%, 100% { background-color: #ff0000; }
```

Compatibilidad entre navegadores

Para los navegadores más antiguos basados en WebKit, deberá utilizar el prefijo `vendor` tanto en la declaración `@keyframes` como en la propiedad `animation`, como se indica a continuación:

```
@-webkit-keyframes {}  
-webkit-animation: ...
```

Sección 27.2: Animaciones con la propiedad `transition`

Útil para animaciones simples, la propiedad `transition` CSS permite que las propiedades CSS basadas en números animen entre estados.

Ejemplo

```
.Ejemplo {  
  height: 100px;  
  background: #fff;  
}  
.Ejemplo:hover {  
  height: 120px;  
  background: #ff0000;  
}
```

Por defecto, al pasar el puntero del ratón sobre un elemento con la clase `.Ejemplo`, la altura del elemento salta inmediatamente a `120px` y su color de fondo pasa a ser rojo (`#ff0000`).

Añadiendo la propiedad `transition`, podemos hacer que estos cambios se produzcan a lo largo del tiempo:

```
.Ejemplo {  
  ...  
  transition: all 400ms ease;  
}
```

El valor `all` aplica la transición a todas las propiedades compatibles (basadas en números). Cualquier nombre de propiedad compatible (como `height` o `top`) puede sustituir a esta palabra clave.

`400ms` especifica el tiempo que dura la transición. En este caso, el cambio de altura del elemento tardará 400 milisegundos en completarse.

Por último, el valor `ease` es la función de animación, que determina cómo se reproduce la animación. `ease` significa empezar lento, acelerar y volver a terminar lento. Otros valores son `linear`, `ease-out` y `ease-in`.

Compatibilidad entre navegadores

En general, la propiedad `transition` está bien soportada por los principales navegadores, excepto IE 9. Para versiones anteriores de Firefox y navegadores basados en Webkit, utilice prefijos de proveedor como este:

```
.Ejemplo {  
  transition: all 400ms ease;  
  -moz-transition: all 400ms ease;  
  -webkit-transition: all 400ms ease;  
}
```

Nota: La propiedad `transition` puede animar cambios entre dos valores numéricos cualesquiera, independientemente de la unidad. También puede pasar de una unidad a otra, por ejemplo, de `100px` a `50vh`. Sin embargo, no puede realizar una transición entre un número y un valor predeterminado o automático, como la transición de la altura de un elemento de `100px` a `auto`.

Sección 27.3: Ejemplos de sintaxis

El primer ejemplo de sintaxis muestra la propiedad abreviada de animación utilizando todas las propiedades/parámetros disponibles:

```
animation: 3s ease-in 1s 2 reverse both paused slidein;  
duration | timing-function | delay | iteration-count | direction | fill-mode | playstate | name
```

El segundo ejemplo es un poco más sencillo y muestra que algunas propiedades pueden omitirse:

```
animation: 3s linear 1s slidein;  
duration | timing-function | delay | name
```

Nuestro tercer ejemplo muestra la declaración más mínima. Tenga en cuenta que `animation-name` y `animation-duration` deben ser declarados:

```
animation: 3s slidein;  
duration | name
```

También vale la pena mencionar que cuando se utiliza la abreviatura `animation` el orden de las propiedades hace una diferencia. Obviamente, el navegador puede confundir su duración con su retraso.

Si la brevedad no es lo tuyo, también puedes omitir la propiedad abreviada y escribir cada propiedad individualmente:

```
animation-duration: 3s;  
animation-timing-function: ease-in;  
animation-delay: 1s;  
animation-iteration-count: 2;  
animation-direction: reverse;  
animation-fill-mode: both;  
animation-play-state: paused;  
animation-name: slidein;
```

Sección 27.4: Aumentar el rendimiento de la animación con el atributo 'will-change'

Al crear animaciones y otras acciones que requieran un uso intensivo de la GPU, es importante conocer el atributo `will-change`.

Tanto los fotogramas clave CSS como la propiedad de transición utilizan la aceleración de la GPU. El rendimiento aumenta al descargar cálculos a la GPU del dispositivo. Para ello, se crean capas de pintura (partes de la página que se renderizan individualmente) que se descargan en la GPU para su cálculo. La propiedad `will-change` indica al navegador lo que va a animar, lo que le permite crear áreas de pintura más pequeñas, aumentando así el rendimiento.

La propiedad `will-change` acepta una lista separada por comas de propiedades a animar. Por ejemplo, si planea transformar un objeto y cambiar su opacidad, especificaría:

```
.Ejemplo {  
  ...  
  will-change: transform, opacity;  
}
```

Nota: utilice el cambio de testamento con moderación. La configuración de `will-change` para cada elemento de una página puede causar problemas de rendimiento de rendimiento, ya que el navegador puede intentar crear capas de pintura para cada elemento, aumentando significativamente la cantidad de procesamiento realizado por la GPU.

Capítulo 28: Transformaciones 2D

Función/Parámetro	Detalles
<code>rotate(x)</code>	Define una transformación que mueve el elemento alrededor de un punto fijo en el eje Z
<code>translate(x, y)</code>	Desplaza la posición del elemento en los ejes X e Y
<code>translateX(x)</code>	Desplaza la posición del elemento en el eje X
<code>translateY(y)</code>	Desplaza la posición del elemento en el eje Y
<code>scale(x, y)</code>	Modifica el tamaño del elemento en los ejes X e Y
<code>scaleX(x)</code>	Modifica el tamaño del elemento en el eje X
<code>scaleY(y)</code>	Modifica el tamaño del elemento en el eje Y
<code>skew(x, y)</code>	Cartografía de cizalladura, o transvección, que distorsiona cada punto de un elemento un ángulo determinado en cada dirección.
<code>skewX(x)</code>	Mapeado de cizalladura horizontal que distorsiona cada punto de un elemento un ángulo determinado en la dirección horizontal.
<code>skewY(y)</code>	Cartografía de cizalladura vertical que distorsiona cada punto de un elemento un ángulo determinado en la vertical.
<code>matrix()</code>	Define una transformación 2D en forma de matriz de transformación.
ángulo	El ángulo en el que el elemento debe girarse o sesgarse (dependiendo de la función con la que se utilice). El ángulo puede indicarse en grados (deg), gradianes (grad), radianes (rad) o vueltas (turn). En la función <code>skew()</code> , el segundo ángulo es opcional. Si no se proporciona, no habrá ninguna desviación (0) en el eje Y.
longitud o porcentaje	La distancia expresada en longitud o en porcentaje a la que debe ser trasladar. En la función <code>translate()</code> , la segunda longitud o porcentaje es opcional. Si no se proporciona, no habrá traslación (0) en el eje Y.
factor de escala	Un número que define cuántas veces debe escalarse el elemento en el eje especificado. En la función <code>scale()</code> , el segundo factor de escala es opcional. Si no se indica, el primer factor de escala se aplicará también al eje Y.

Sección 28.1: rotate

HTML

```
<div class="rotate"></div>
```

CSS

```
.rotate {  
  width: 100px;  
  height: 100px;  
  background: teal;  
  transform: rotate(45deg);  
}
```

Este ejemplo girará el `div` 45 grados en el sentido de las agujas del reloj. El centro de rotación está en el centro del `div`, 50% desde la izquierda y 50% desde arriba. Puede cambiar el centro de rotación estableciendo la propiedad `transform-origin`.

```
transform-origin: 100% 50%;
```

El ejemplo anterior fijará el centro de rotación en la mitad del extremo derecho.

Sección 28.2: scale

HTML

```
<div class="scale"></div>
```

CSS

```
.scale {  
  width: 100px;  
  height: 100px;  
  background: teal;  
  transform: scale(0.5, 1.3);  
}
```

Este ejemplo escalará el `div` a $100\text{px} * 0.5 = 50\text{px}$ en el eje X y a $100\text{px} * 1.3 = 130\text{px}$ en el eje Y.

El centro de la transformación está en el centro del `div`, a `50%` de la izquierda y `50%` de la parte superior.

Sección 28.3: skew

HTML

```
<div class="skew"></div>
```

CSS

```
.skew {  
  width: 100px;  
  height: 100px;  
  background: teal;  
  transform: skew(20deg, -30deg);  
}
```

Este ejemplo sesgará el `div` 20 grados en el eje X y -30 grados en el eje Y. El centro de la transformación está en el centro del `div`, a `50%` de la izquierda y `50%` de la parte superior.

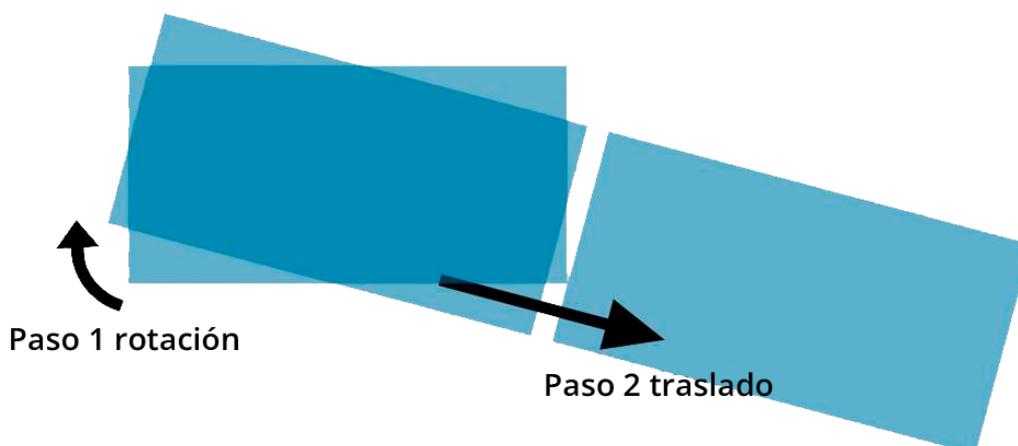
Sección 28.4: Múltiples transformaciones

Se pueden aplicar múltiples transformaciones a un elemento en una propiedad de esta forma:

```
transform: rotate(15deg) translateX(200px);
```

Esto rotará el elemento 15 grados en el sentido de las agujas del reloj y lo desplazará 200px a la derecha.

En las transformaciones encadenadas, **el sistema de coordenadas se desliza con el elemento**. Esto significa que la traslación no será horizontal, sino sobre un eje que girará 15 grados en el sentido de las agujas del reloj, como se muestra en la siguiente imagen:

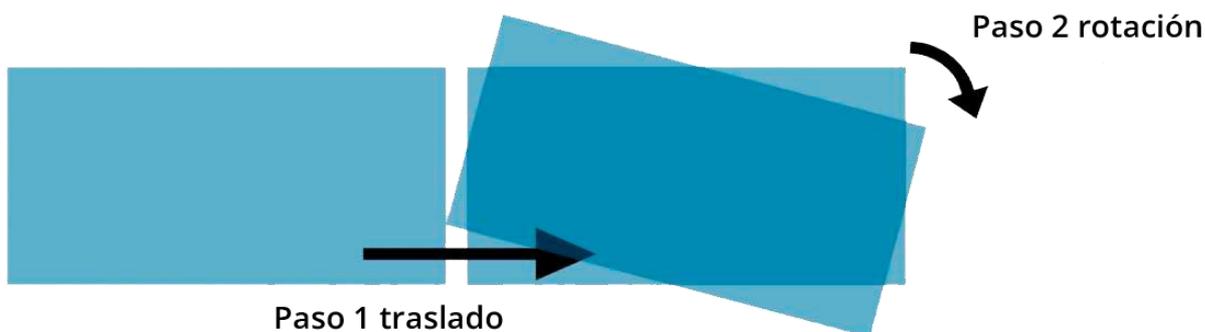


Si cambia el orden de las transformaciones, cambiará el resultado. El primer ejemplo será diferente a

```
transform: translateX(200px) rotate(15deg);
```

```
<div class="transform"></div>  
.transform {  
  transform: rotate(15deg) translateX(200px);  
}
```

Como se muestra en esta imagen:



Sección 28.5: translate

HTML

```
<div class="translate"></div>
```

CSS

```
.translate {  
  width: 100px;  
  height: 100px;  
  background: teal;  
  transform: translate(200px, 50%);  
}
```

Este ejemplo moverá el div 200px en el eje X y $100px * 50\% = 50px$ en el eje Y.

También puede especificar traslaciones en un solo eje.

En el eje X:

```
.translate {  
  transform: translateX(200px);  
}
```

En el eje Y:

```
.translate {  
  transform: translateY(50%);  
}
```

Sección 28.6: transform-origin

Las transformaciones se realizan con respecto a un punto definido por la propiedad `transform-origin`.

La propiedad toma 2 valores: `transform-origin: X Y;`

En el siguiente ejemplo el primer div (.tl) se gira alrededor de la esquina superior izquierda con `transform-origin: 0 0`; y el segundo (.tr) se transforma alrededor de su esquina superior derecha con `transform-origin: 100% 0`. La rotación se aplica al pasar el ratón **por encima**:

HTML

```
<div class="transform origin1"></div>
<div class="transform origin2"></div>
```

CSS

```
.transform {
  display: inline-block;
  width: 200px;
  height: 100px;
  background: teal;
  transition: transform 1s;
}
.origin1 {
  transform-origin: 0 0;
}
.origin2 {
  transform-origin: 100% 0;
}
.transform:hover {
  transform: rotate(30deg);
}
```

El valor por defecto de la propiedad `transform-origin` es `50% 50%` que es el centro del elemento.

Capítulo 29: Transformaciones 3D

Sección 29.1: Forma de aguja o puntero de brújula mediante transformaciones 3D

HTML

```
<div class='aguja'></div>
```

CSS

```
div.aguja {  
    margin: 100px;  
    height: 150px;  
    width: 150px;  
    transform: rotateY(85deg) rotateZ(45deg);  
    /* presentación */  
    background-image: linear-gradient(to top left, #555 0%, #555 40%, #444 50%, #333 97%);  
    box-shadow: inset 6px 6px 22px 8px #272727;  
}
```

En el ejemplo anterior, se crea una forma de aguja o puntero de brújula utilizando transformaciones 3D. Generalmente cuando aplicamos la transformación `rotate` en un elemento, la rotación ocurre sólo en el eje Z y en el mejor de los casos terminaremos con sólo formas de diamante. Pero cuando se añade una transformación `rotateY` encima, el elemento se aprieta en el eje Y, por lo tanto, termina pareciéndose a una aguja. Cuanto mayor sea la rotación del eje Y, más apretado parecerá el elemento.

El resultado del ejemplo anterior sería una aguja apoyada en su punta. Para crear una aguja que se apoya en su base, la rotación debe hacerse a lo largo del eje X en lugar de a lo largo del eje Y. Así que el valor de la propiedad `transform` ser algo como `rotateX(85deg) rotateZ(45deg);`.

[Este ejemplo](#) utiliza un enfoque similar para crear algo que recuerda al logotipo de Safari o a la esfera de una brújula.

Captura de pantalla de elemento sin transformación



Captura de pantalla de elemento con sólo transformación 2D



Captura de pantalla de elemento con transformación 3D



Sección 29.2: Efecto de texto 3D con sombra

HTML

```
<div id="titulo">  
  <h1 data-content="PASAR">PASAR</h1>  
</div>
```

CSS

```
* {
  margin:0;padding:0;
}
html,body {
  height:100%;width:100%;overflow:hidden;background:#0099CC;
}
#titulo {
  position:absolute;
  top:50%; left:50%;
  transform:translate(-50%,-50%);
  perspective-origin:50% 50%;
  perspective:300px;
}
h1 {
  text-align:center;
  font-size:12vmin;
  font-family: 'Open Sans', sans-serif;
  color:rgba(0,0,0,0.8);
  line-height:1em;
  transform:rotateY(50deg);
  perspective:150px;
  perspective-origin:0% 50%;
}
h1:after {
  content:attr(data-content);
  position:absolute;
  left:0;top:0;
  transform-origin:50% 100%;
  transform:rotateX(-90deg);
  color:#0099CC;
}
#titulo:before {
  content:'';
  position:absolute;
  top:-150%; left:-25%;
  width:180%; height:328%;
  background:rgba(255,255,255,0.7);
  transform-origin:0 100%;
  transform: translatez(-200px) rotate(40deg) skewX(35deg);
  border-radius:0 0 100% 0;
}
```

Ejemplo de vista con efecto `hover` adicional



En este ejemplo, el texto se transforma para que parezca que entra en la pantalla lejos del usuario.

La sombra se transforma en consecuencia para que siga al texto. Como está hecho con un pseudoelemento y el atributo data, hereda las transformaciones de su padre (la etiqueta H1).

La "luz" blanca se hace con un pseudoelemento sobre el elemento `#title`. Está sesgado y utiliza `border-radius` para la esquina redondeada.

Sección 29.3: backface-visibility

La propiedad `backface-visibility` está relacionada con las transformaciones 3D.

Con las transformaciones 3D y la propiedad `backface-visibility`, puede girar un elemento de modo que la cara frontal original de un elemento ya no esté orientada hacia la pantalla.

Por ejemplo, esto voltearía un elemento lejos de la pantalla:

```
<div class="flip">Loren ipsum</div>
<div class="flip back">Lorem ipsum</div>
.flip {
  -webkit-transform: rotateY(180deg);
  -moz-transform: rotateY(180deg);
  -ms-transform: rotateY(180deg);
  -webkit-backface-visibility: visible;
  -moz-backface-visibility: visible;
  -ms-backface-visibility: visible;
}
.flip.back {
  -webkit-backface-visibility: hidden;
  -moz-backface-visibility: hidden;
  -ms-backface-visibility: hidden;
}
```

Firefox 10+ e IE 10+ admiten `backface-visibility` sin prefijo. Opera, Chrome, Safari, iOS y Android necesitan `-webkit-backface-visibility`.

Tiene 4 valores:

1. **visible** (por defecto) - el elemento será siempre visible incluso cuando no esté orientado hacia la pantalla.
2. **hidden** - el elemento no es visible cuando no está frente a la pantalla.
3. **inherit** - la propiedad obtendrá su valor del elemento padre.
4. **initial** - establece la propiedad a su valor por defecto, que es visible.

Sección 29.4: Cubo 3D

Las transformaciones 3D se pueden utilizar para crear muchas formas 3D. He aquí un sencillo ejemplo de cubo 3D CSS:

CSS

```
body {
  perspective-origin: 50% 100%;
  perspective: 1500px;
  overflow: hidden;
}
.cubo {
  position: relative;
  padding-bottom: 20%;
  transform-style: preserve-3d;
  transform-origin: 50% 100%;
  transform: rotateY(45deg) rotateX(0);
}
.caraCubo {
  position: absolute;
  top: 0;
  left: 40%;
  width: 20%;
  height: 100%;
  margin: 0 auto;
  transform-style: inherit;
  background: #C52329;
  box-shadow: inset 0 0 0 5px #333;
  transform-origin: 50% 50%;
  transform: rotateX(90deg);
  backface-visibility: hidden;
}
.cara2 {
  transform-origin: 50% 50%;
  transform: rotateZ(90deg) translateX(100%) rotateY(90deg);
}
.caraCubo:before, .caraCubo:after {
  content: '';
  position: absolute;
  width: 100%;
  height: 100%;
  transform-origin: 0 0;
  background: inherit;
  box-shadow: inherit;
  backface-visibility: inherit;
}
.caraCubo:before {
  top: 100%;
  left: 0;
  transform: rotateX(-90deg);
}
.caraCubo:after {
  top: 0;
  left: 100%;
  transform: rotateY(90deg);
}
```

HTML

```
<div class="cube">
  <div class="caraCubo"></div>
  <div class="caraCubo cara2"></div>
</div>
```

[Ver este ejemplo](#)

En la demostración se añade un estilo adicional y se aplica una transformación al pasar el ratón por encima para ver las 6 caras del cubo.

Debe tenerse en cuenta que:

- 4 caras se hacen con pseudoelementos.
- se aplican transformaciones encadenadas.

Capítulo 30: Propiedad filter

Valor

`blur(x)`

`brightness(x)`

`contrast(x)`

`drop-shadow(h, v, x, y, z)`

`greyscale(x)`

`hue-rotate(x)`

`invert(x)`

`opacity(x)`

`saturate(x)`

`sepia(x)`

Descripción

Desenfoca la imagen x píxeles.

Aclara la imagen en cualquier valor por encima de 1,0 o 100%. Por debajo, la imagen se oscurecida.

Proporciona más contraste a la imagen en cualquier valor por encima de 1,0 o 100%. Por debajo de ese valor, la imagen se satura menos.

Da a la imagen una sombra. H y V pueden tener valores negativos. X, Y y Z son opcionales.

Muestra la imagen en escala de grises, con un valor máximo de 1,0 o 100%.

Aplica una rotación de tono a la imagen.

Invierte el color de la imagen con un valor máximo de 1,0 o 100%.

Establece el grado de opacidad/transparencia de la imagen con un valor máximo de 1,0 o 100%.

Satura la imagen en cualquier valor por encima de 1,0 o 100%. Por debajo, la imagen empezará a desaturar.

Convierte la imagen a sepia con un valor máximo de 1,0 o 100%.

Sección 30.1: blur

HTML

```
<img src='pato-donald.png' alt='Pato Donald' title='Pato Donald' />
```

CSS

```
img {  
  -webkit-filter: blur(1px);  
  filter: blur(1px);  
}
```

Resultado



Te dan ganas de frotarte las gafas.

Sección 30.2: Sombra (usa box-shadow en su lugar si es posible)

HTML

```
<p> Mi sombra siempre me sigue. </p>
```

CSS

```
p {  
  -webkit-filter: drop-shadow(10px 10px 1px green);  
  filter: drop-shadow(10px 10px 1px green);  
}
```

Resultado

Mi sombra siempre me sigue.
Mi sombra siempre me sigue.

Sección 30.3: hue-rotate

HTML

```
<img src='donald-duck.png' alt='Donald Duck' title='Donald Duck' />
```

CSS

```
img {  
  -webkit-filter: hue-rotate(120deg);  
  filter: hue-rotate(120deg);  
}
```

Resultado



Sección 30.4: Múltiples valores de filter

Para utilizar varios filtros, separe cada valor con un espacio.

HTML

```
<img src='pato-donald.png' alt='Pato Donald' title='Pato Donald' />
```

CSS

```
img {  
  -webkit-filter: brightness(200%) grayscale(100%) sepia(100%) invert(100%);  
  filter: brightness(200%) grayscale(100%) sepia(100%) invert(100%);  
}
```

Resultado



Sección 30.5: Invertir color

HTML

```
<div></div>
```

CSS

```
div {  
  width: 100px;  
  height: 100px;  
  background-color: white;  
  -webkit-filter: invert(100%);  
  filter: invert(100%);  
}
```

Resultado



Pasa de blanco a negro.

Capítulo 31: Estilo del cursor

Sección 31.1: Cambiar el tipo de cursor

cursor: *valor*;

	default		n-resize		not-allowed
	crosshair		ne-resize		no-drop
	hand		e-resize		vertical-text
	pointer		se-resize		all-scroll
	Cross browser		s-resize		col-resize
	move		sw-resize		row-resize
	text		w-resize		
	wait		nw-resize		
	help		progress		

Ejemplos:

Valor	Descripción
none	No se muestra ningún cursor para el elemento
auto	Por defecto. El navegador establece un cursor
help	El cursor indica que hay ayuda disponible
wait	El cursor indica que el programa está ocupado
move	El cursor indica que hay que mover algo
pointer	El cursor es un puntero e indica un enlace

Sección 31.2: pointer-events

La propiedad `pointer-events` permite controlar cómo responden los elementos HTML a los eventos de ratón/toque.

```
.disabled {  
    pointer-events: none;  
}
```

En este ejemplo,

'none' impide todas las opciones de clic, estado y cursor en el elemento HTML especificado [1]

Otros valores válidos para los elementos HTML son:

- auto
- inherit

1. <https://css-tricks.com/almanac/properties/p/pointer-events/>

Otros recursos:

- <https://developer.mozilla.org/en-US/docs/Web/CSS/pointer-events>
- <https://davidwalsh.name/pointer-events>

Sección 31.3: caret-color

La propiedad CSS `caret-color` especifica el color del caret, el indicador visible del punto de inserción en un elemento en el que se inserta texto y otros contenidos al escribir o editar el usuario.

HTML

```
<input id="ejemplo" />
```

CSS

```
#ejemplo {  
    caret-color: red;  
}
```

Recursos:

- <https://developer.mozilla.org/es/docs/Web/CSS/caret-color>

Capítulo 32: box-shadow

Parámetros	Detalles
<code>inset</code>	Por defecto, la sombra se trata como una sombra paralela. La palabra clave <code>inset</code> dibuja la sombra dentro del marco/borde
<code>offset-x</code>	la distancia horizontal
<code>offset-y</code>	la distancia vertical
<code>blur-radius</code>	0 por defecto. El valor no puede ser negativo. Cuanto mayor sea el valor, mayor y más clara será la sombra.
<code>spread-radius</code>	0 por defecto. Los valores positivos harán que la sombra se expanda. Los valores negativos harán que la sombra se reduzca
<code>color</code>	puede tener varias notaciones: una palabra clave de color, hexadecimal, <code>rgb()</code> , <code>rgba()</code> , <code>hsl()</code> , <code>hsla()</code>

Sección 32.1: sombra paralela sólo en la parte inferior utilizando un pseudoelemento

HTML

```
<div class="box_shadow"></div>
```

CSS

```
.box_shadow {  
  background-color: #1C90F3;  
  width: 200px;  
  height: 100px;  
  margin: 50px;  
}  
.box_shadow:after {  
  content: "";  
  width: 190px;  
  height: 1px;  
  margin-top: 98px;  
  margin-left: 5px;  
  display: block;  
  position: absolute;  
  z-index: -1;  
  -webkit-box-shadow: 0px 0px 8px 2px #444444;  
  -moz-box-shadow: 0px 0px 8px 2px #444444;  
  box-shadow: 0px 0px 8px 2px #444444;  
}
```



Sección 32.2: sombra paralela

HTML

```
<div class="box_shadow"></div>
```

CSS

```
.box_shadow {  
    -webkit-box-shadow: 0px 0px 10px -1px #444444;  
    -moz-box-shadow: 0px 0px 10px -1px #444444;  
    box-shadow: 0px 0px 10px -1px #444444;  
}
```

Sección 32.3: sombra interior

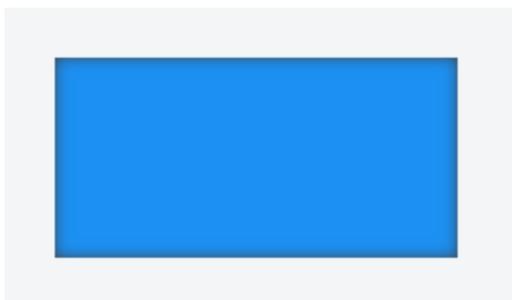
HTML

```
<div class="box_shadow"></div>
```

CSS

```
.box_shadow {  
    background-color: #1C90F3;  
    width: 200px;  
    height: 100px;  
    margin: 50px;  
    -webkit-box-shadow: inset 0px 0px 10px 0px #444444;  
    -moz-box-shadow: inset 0px 0px 10px 0px #444444;  
    box-shadow: inset 0px 0px 10px 0px #444444;  
}
```

Resultado



Sección 32.4: sombras múltiples

HTML

```
<div class="box_shadow"></div>
```

CSS

```
.box_shadow {  
  width: 100px;  
  height: 100px;  
  margin: 100px;  
  box-shadow:  
    -52px -52px 0px 0px #f65314,  
    52px -52px 0px 0px #7cbb00,  
    -52px 52px 0px 0px #00a1f1,  
    52px 52px 0px 0px #ffbb00;  
}
```



Capítulo 33: Formas de float

Parámetro	Detalles
<code>none</code>	Un valor de <code>none</code> significa que el área flotante (el área que se utiliza para envolver el contenido alrededor de un elemento flotante) no se ve afectada. Este es el valor por defecto/inicial.
<code>basic-shape</code>	Hace referencia a una de las funciones <code>inset()</code> , <code>circle()</code> , <code>ellipse()</code> o <code>polygon()</code> . Utilizando una de estas funciones y sus valores se define la forma.
<code>shape-box</code>	Se refiere a uno entre <code>margin-box</code> , <code>border-box</code> , <code>padding-box</code> , <code>content-box</code> . Cuando sólo se proporciona <code><shape-box></code> (sin <code><basic-shape></code>) esta caja es la forma. Cuando se utiliza junto con <code><basic-shape></code> , éste actúa como cuadro de referencia.
<code>image</code>	Cuando se proporciona una imagen como valor, la forma se calcula basándose en el canal alfa de la imagen especificada.

Sección 33.1: Forma exterior con forma básica - circle()

Con la propiedad CSS `shape-outside` se pueden definir valores de forma para el área flotante de modo que el contenido en línea se envuelva alrededor de la forma en lugar de la caja del flotador.

HTML

```


<p>Algún párrafo cuyo contenido de texto deba envolverse de forma que siga la curva del círculo
a ambos lados. Y luego hay texto de relleno para que el texto sea lo suficientemente largo.
Lorem Ipsum Dolor Sit Amet....</p>
```

CSS

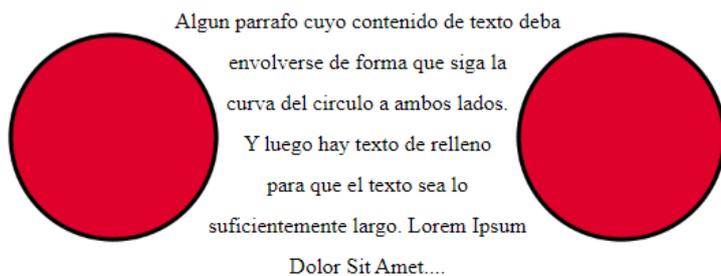
```
img:nth-of-type(1) {
  shape-outside: circle(80px at 50% 50%);
  float: left;
  width: 200px;
}
img:nth-of-type(2) {
  shape-outside: circle(80px at 50% 50%);
  float: right;
  width: 200px;
}
p {
  text-align: center;
  line-height: 30px; /* únicamente para demostración */
}
```

En el ejemplo anterior, ambas imágenes son en realidad imágenes cuadradas y cuando el texto se coloca sin la propiedad `shape-outside` no fluirá alrededor del círculo en ninguno de sus lados. Fluirá alrededor de la caja de contención de la imagen únicamente. Con `shape-outside` el área flotante se redefine como un *círculo* y el contenido se hace fluir alrededor de este *círculo imaginario* creado con `shape-outside`.

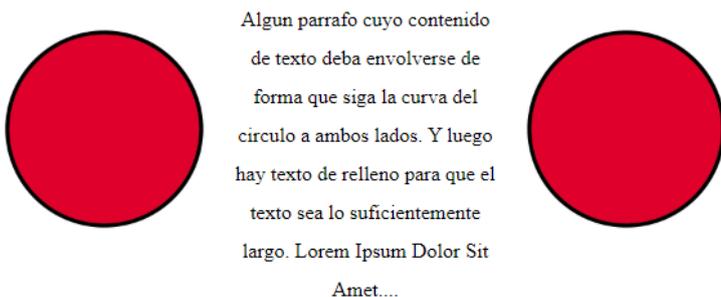
El *círculo imaginario* que se utiliza para redefinir el área flotante es un círculo con un radio de 80px trazado desde el punto medio-central del cuadro de referencia de la imagen.

A continuación, se muestran un par de capturas de pantalla para ilustrar cómo el contenido se envolvería alrededor cuando se utiliza `shape-outside` y cuando no se utiliza.

Salida con shape-outside



Salida sin shape-outside



Sección 33.2: Margen de forma

La propiedad CSS `shape-margin` añade un margen a `shape-outside`.

HTML

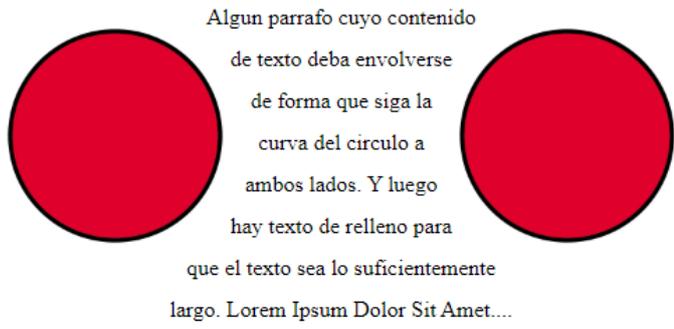
```
  
  
<p>Algún párrafo cuyo contenido de texto deba envolverse de forma que siga la curva del círculo a ambos lados. Y luego hay texto de relleno para que el texto sea lo suficientemente largo. Lorem Ipsum Dolor Sit Amet....</p>
```

CSS

```
img:nth-of-type(1) {  
    shape-outside: circle(80px at 50% 50%);  
    shape-margin: 10px;  
    float: left;  
    width: 200px;  
}  
img:nth-of-type(2) {  
    shape-outside: circle(80px at 50% 50%);  
    shape-margin: 10px;  
    float: right;  
    width: 200px;  
}  
p {  
    text-align: center;  
    line-height: 30px; /* unicamente para demostración */  
}
```

En este ejemplo, se añade un margen de 10px alrededor de la **forma** utilizando `shape-margin`. Esto crea un poco más de espacio entre el *círculo imaginario* que define el área flotante y el contenido real que fluye alrededor.

Resultado



Capítulo 34: Estilos de lista

Valor	Descripción
<code>list-style-type</code>	el tipo de marcador de elemento de lista.
<code>list-style-position</code>	especifica dónde colocar el marcador
<code>list-style-image</code>	especifica el tipo de marcador de elemento de lista
<code>initial</code>	establece esta propiedad en su valor por defecto
<code>inherit</code>	hereda esta propiedad de su elemento padre

Sección 34.1: Posición de la viñeta

Una lista está formada por elementos `` dentro de un elemento contenedor (`` o ``). Tanto los elementos de la lista como el contenedor pueden tener márgenes y rellenos que influyen en la posición exacta del contenido del elemento de la lista en el documento. Los valores por defecto para el margen y el relleno pueden ser diferentes para cada navegador. Con el fin de obtener el mismo diseño entre navegadores, estos deben ser establecidos específicamente.

Cada elemento de la lista recibe una "caja de marcador", que contiene el marcador de viñeta. Este cuadro puede colocarse dentro o fuera del cuadro de elementos de la lista.

`list-style-position`: `inside`;

coloca la viñeta dentro del elemento ``, empujando el contenido hacia la derecha según sea necesario.

`list-style-position`: `outside`;

coloca la viñeta a la izquierda del elemento ``. Si no hay espacio suficiente en el relleno del elemento contenedor, el cuadro de marcador se extenderá hacia la izquierda aunque se saliera de la página.

Sección 34.2: Eliminar viñetas / números

A veces, una lista no debe mostrar viñetas ni números. En ese caso, recuerde especificar el margen y el relleno.

HTML

```
<ul>
  <li>primer item</li>
  <li>segundo item</li>
</ul>
```

CSS

```
ul {
  list-style-type: none;
}
li {
  margin: 0;
  padding: 0;
}
```

Sección 34.3: Tipo de viñeta o numeración

Específico para etiquetas **** dentro de una lista desordenada (****):

```
list-style: disc; /* Un círculo relleno (por defecto) */  
list-style: circle; /* Un círculo hueco */  
list-style: square; /* Un cuadrado relleno */  
list-style: '-'; /* cualquier cadena de caracteres */
```

Específico para etiquetas **** dentro de una lista ordenada (****):

```
list-style: decimal; /* Números decimales empezando por 1 (por defecto) */  
list-style: decimal-leading-zero; /* Números decimales con ceros iniciales (01, 02, 03, ... 10) */  
list-style: lower-roman; /* Números romanos en minúsculas (i., ii., iii., iv., ...) */  
list-style: upper-roman; /* Uppercase roman numerals (I., II., III., IV., ...) */  
list-style-type: lower-greek; /* Números romanos en mayúsculas (I., II., III., IV., ...) */  
list-style-type: lower-alpha; /* Letras minúsculas (a., b., c., d., ...) */  
list-style-type: lower-latin; /* Letras minúsculas (a., b., c., d., ...) */  
list-style-type: upper-alpha; /* Letras mayúsculas (A., B., C., D., ...) */  
list-style-type: upper-latin; /* Letras mayúsculas (A., B., C., D., ...) */
```

No específico:

```
list-style: none; /* No hay marcador de lista visible */  
list-style: inherit; /* Hereda de padre */
```

Capítulo 35: Contadores

Parámetro	Detalles
<code>counter-name</code>	Este es el nombre del contador que necesita ser creado o incrementado o impreso. Puede ser cualquier nombre personalizado que desee el desarrollador
<code>integer</code>	Este entero es un valor opcional que cuando se proporciona junto al nombre del contador representará el valor inicial del contador (en las propiedades <code>counter-set</code> , <code>counter-reset</code>) o el valor por el que el contador (en <code>counter-increment</code>)
<code>none</code>	Este es el valor inicial para las 3 propiedades del <code>counter-*</code> . Cuando se utiliza este valor para el <code>counter-increment</code> , el valor de ninguno de los contadores se ve afectado. Cuando se utiliza para los otros dos, no se crea ningún contador
<code>counter-style</code>	Especifica el estilo en el que debe mostrarse el valor del contador. Admite todos los valores admitidos por la propiedad <code>list-style-type</code> . Si no se utiliza <code>none</code> , el valor del contador no se imprime
<code>connector-string</code>	Representa la cadena de caracteres que debe colocarse entre los valores de dos niveles de contador diferentes (como la "." en "2.1.1")

Sección 35.1: Aplicación del estilo de números romanos al contador

HTML

```
<div class='item'>Item N°: 1</div>
<div class='item'>Item N°: 2</div>
<div class='item'>Item N°: 3</div>
```

CSS

```
body {
    counter-reset: item-counter;
}
.item {
    counter-increment: item-counter;
}
.item:before {
    content: counter(item-counter, upper-roman) ". "; /* especificando el estilo mayúscula-romana la salida sería en números romanos */
}
```

En el ejemplo anterior, la salida del contador se mostraría como I, II, III (números romanos) en lugar de los habituales 1, 2, 3, ya que el desarrollador ha especificado explícitamente el estilo del contador.

Sección 35.2: Numere cada elemento con el contador CSS

HTML

```
<div class='item'>
  <div class='item-encabezado'>Item 1 Encabezado</div>
  <div class='item-contenido'>Lorem Ipsum Dolor Sit Amet...</div>
</div>
<div class='item'>
  <div class='item-encabezado'>Item 2 Encabezado</div>
  <div class='item-contenido'>Lorem Ipsum Dolor Sit Amet...</div>
</div>
<div class='item'>
  <div class='item-encabezado'>Item 3 Encabezado</div>
  <div class='item-contenido'>Lorem Ipsum Dolor Sit Amet...</div>
</div>
```

CSS

```
body {
  counter-reset: item-counter; /* crea el contador */
}
.item {
  counter-increment: item-counter; /* incrementar el contador cada vez que se encuentre un
  elemento de la clase "item" */
}
.item-header:before {
  content: counter(item-counter) ". "; /* imprimir el valor del contador antes de la cabecera
  y añádele un ". */
}
/* unicamente para demostración */
.item {
  border: 1px solid;
  height: 100px;
  margin-bottom: 10px;
}
.item-encabezado {
  border-bottom: 1px solid;
  height: 40px;
  line-height: 40px;
  padding: 5px;
}
.item-contenido {
  padding: 8px;
}
```

Sección 35.3: Numeración multinivel con contadores CSS

CSS

```
ul {
  list-style: none;
  counter-reset: list-item-number; /* contador autoanidado ya que el nombre es el mismo para
  todos los niveles */
}
li {
  counter-increment: list-item-number;
}
li:before {
  content: counters(list-item-number, ".") " "; /* uso de la función counters() significa que
  el valor de los contadores en todos los niveles superiores se combinan antes de imprimir */
}
```

HTML

```
<ul>
  <li>Nivel 1
    <ul>
      <li>Nivel 1.1
        <ul>
          <li>Nivel 1.1.1</li>
        </ul>
      </li>
    </ul>
  </li>
  <li>Nivel 2
    <ul>
      <li>Nivel 2.1
        <ul>
          <li>Nivel 2.1.1</li>
          <li>Nivel 2.1.2</li>
        </ul>
      </li>
    </ul>
  </li>
  <li>Nivel 3</li>
</ul>
```

Este es un ejemplo de numeración multinivel mediante contadores CSS. Utiliza el concepto de **auto-anidación** de los contadores. La auto-anidación es un concepto en el que si un elemento ya tiene un contador con el nombre dado, pero tiene que crear otro, entonces lo crea como hijo del contador existente. Aquí, el segundo nivel `ul` ya hereda el contador `list-item-number` de su padre, pero luego tiene que crear su propio `list-item-number` (para sus hijos `li`) y así crea `list-item-number [1]` (contador para el segundo nivel) y lo anida bajo `list-item-number [0]` (contador de primer nivel).

La salida se imprime utilizando la función `counters()` en lugar de la función `counter()` porque la función `counters()` está diseñada para anteponer el valor de todos los contadores de nivel superior (padre) al imprimir la salida.

Capítulo 36: Funciones

Sección 36.1: Función calc()

Acepta una expresión matemática y devuelve un valor numérico.

Es especialmente útil cuando se trabaja con diferentes tipos de unidades (por ejemplo, restar un valor px de un porcentaje) para calcular el valor de un atributo.

Se pueden utilizar los operadores `+`, `-`, `/` y `*`, y se pueden añadir paréntesis para especificar el orden de las operaciones si es necesario.

Utilice `calc()` para calcular la anchura de un elemento `div`:

```
#div1 {  
  position: absolute;  
  left: 50px;  
  width: calc(100% - 100px);  
  border: 1px solid black;  
  background-color: yellow;  
  padding: 5px;  
  text-align: center;  
}
```

Utilice `calc()` para determinar la posición de una imagen de fondo:

```
background-position: calc(50% + 17px) calc(50% + 10px), 50% 50%;
```

Utilice `calc()` para determinar la altura de un elemento:

```
height: calc(100% - 20px);
```

Sección 36.2: Función attr()

Devuelve el valor de un atributo del elemento seleccionado.

A continuación se muestra un elemento `blockquote` que contiene un carácter dentro de un atributo `data-*` que CSS puede utilizar (por ejemplo, dentro de del pseudo-elemento `::before` y `::after`) usando esta función.

```
<blockquote data-mark=""></blockquote>
```

En el siguiente bloque CSS, el carácter se añade antes y después del texto dentro del elemento:

```
blockquote[data-mark]::before,  
blockquote[data-mark]::after {  
  content: attr(data-mark);  
}
```

Sección 36.3: Función var()

La función `var()` permite acceder a variables CSS.

```
:root {  
  --primary-color: blue; /* establecer una variable */  
}  
  
selector {  
  color: var(--primary-color); /* acceder a la variable */  
}
```

Esta función está actualmente en desarrollo. Consulte caniuse.com para conocer la compatibilidad más reciente con navegadores.

Sección 36.4: Función radial-gradient()

Crea una imagen que representa un gradiente de colores que irradia desde el centro del gradiente.

```
radial-gradient(red, orange, yellow) /* Un degradado que sale del centro del degradado, rojo en el centro, luego naranja, hasta que finalmente es amarillo en los bordes. */
```

Sección 36.5: Función linear-gradient()

Crea una imagen que representa un gradiente lineal de colores.

```
linear-gradient( 0deg, red, yellow 50%, blue);
```

Esto crea un gradiente que va de abajo hacia arriba, con colores que comienzan en rojo, luego amarillo al 50%, y terminando en azul.

Capítulo 37: Propiedades personalizadas (variables)

Las Variables CSS permite crear valores reutilizables que pueden usarse en todo un documento CSS.

Por ejemplo, en CSS es habitual reutilizar un único color en todo el documento. Antes de las Variables CSS, esto significaba reutilizar el mismo valor de color muchas veces a lo largo de un documento. Con las Variables CSS el valor del color puede ser asignado a una variable y referenciado en múltiples lugares. Esto facilita el cambio de valores y es más semántico que utilizar valores CSS tradicionales.

Sección 37.1: Color variable

```
:root {
  --rojo: #b00;
  --azul: #4679bd;
  --gris: #ddd;
}
.Bx1 {
  color: var(--rojo);
  background: var(--gris);
  border: 1px solid var(--rojo);
}
```

Sección 37.2: Dimensiones variables

```
:root {
  --W200: 200px;
  --W10: 10px;
}
.Bx2 {
  width: var(--W200);
  height: var(--W200);
  margin: var(--W10);
}
```

Sección 37.3: Variables en cascada

Las variables CSS funcionan en cascada de la misma manera que otras propiedades, y se pueden reformular de forma segura.

Puede definir variables varias veces y sólo la definición con mayor especificidad se aplicará al elemento seleccionado.

Suponiendo que este HTML:

```
<a class="boton">Botón Verde</a>
<a class="boton boton_rojo">Botón Rojo</a>
<a class="boton">Botón activado</a>
```

Podemos escribir este CSS:

```
.boton {
  --color: green;
  padding: .5rem;
  border: 1px solid var(--color);
  color: var(--color);
}
.boton:hover {
  --color: blue;
}
.boton_rojo {
  --color: red;
}
```

Y obtener este resultado:



Sección 37.4: Válidos/Invalidos

Nomenclatura Al nombrar las variables CSS, sólo contiene letras y guiones al igual que otras propiedades CSS (por ejemplo: `line-height`, `-moz-box-sizing`), pero debe comenzar con guiones dobles (--).

```
// Estos son los nombres de las variables Inválidos
--123color: blue;
--#color: red;
--bg_color: yellow
--$width: 100px;
```

```
// Nombres de variables válidos
--color: red;
--bg-color: yellow
--width: 100px;
```

Las variables CSS distinguen entre mayúsculas y minúsculas.

```
/* Los nombres de las variables que figuran a continuación son todas variables diferentes */
--pcolor: ;
--Pcolor: ;
--pColor: ;
```

Vacío Vs Espacio

```
/* Inválido */
--color:;
/* Válido */
--color: ; /* espacio asignado */
```

Concatenaciones

```
/* Inválido - CSS no admite concatenación */
.logo{
  --logo-url: 'logo';
  background: url('assets/img/' var(--logo-url) '.png');
}
/* Inválido - Bug CSS */
.logo{
  --logo-url: 'assets/img/logo.png';
  background: url(var(--logo-url));
}
/* Válido */
.logo{
  --logo-url: url('assets/img/logo.png');
  background: var(--logo-url);
}
```

Cuidado al utilizar las unidades

```
/* Inválido */
--width: 10;
width: var(--width)px;
/* Válido */
--width: 10px;
width: var(--width);
/* Válido */
--width: 10;
width: calc(1px * var(--width)); /* multiplicar por 1 unidad para convertir */
width: calc(1em * var(--width));
```

Sección 37.5: Con media queries

Puedes reajustar las variables dentro de las consultas de medios y hacer que esos nuevos valores se reproduzcan en cascada dondequiera que se utilicen, algo que no es posible con las variables del pre-procesador.

Aquí, una consulta de medios cambia las variables utilizadas para configurar una cuadrícula muy simple:

HTML

```
<div></div>
<div></div>
<div></div>
<div></div>
```

CSS

```
:root{
  --width: 25%;
  --content: 'Esto es el escritorio';
}
@media only screen and (max-width: 767px){
  :root{
    --width:50%;
    --content: 'Esto es móvil';
  }
}
@media only screen and (max-width: 480px){
  :root{
    --width:100%;
  }
}
div{
  width: calc(var(--width) - 20px);
  height: 100px;
}
div:before{
  content: var(--content);
}
/* Otros estilos */
body {
  padding: 10px;
}
div{
  display: flex;
  align-items: center;
  justify-content: center;
  font-weight:bold;
  float:left;
  margin: 10px;
  border: 4px solid black;
  background: red;
}
```

Puede probar a cambiar el tamaño de la ventana en esta [demostración de CodePen](#).

Aquí tienes una captura de pantalla animada del cambio de tamaño en acción:



This is desktop

This is desktop

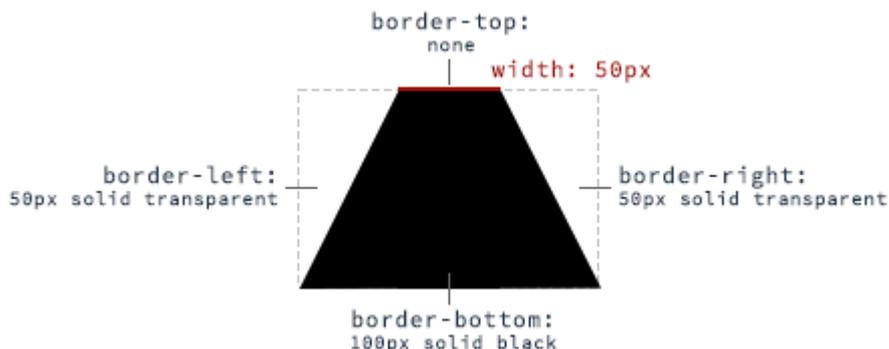
This is desktop

This is desktop

Capítulo 38: Formas de un solo elemento

Sección 38.1: Trapecio

Un trapecio puede estar formado por un elemento de bloque con altura cero (altura de 0px), una anchura mayor que cero y un borde transparente excepto por un lado:



HTML

```
<div class="trapecio"></div>
```

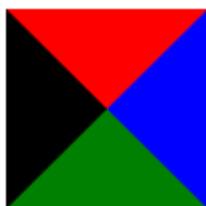
CSS

```
.trapecio {  
  width: 50px;  
  height: 0;  
  border-left: 50px solid transparent;  
  border-right: 50px solid transparent;  
  border-bottom: 100px solid black;  
}
```

Cambiando los lados del borde, se puede ajustar la orientación del trapecio.

Sección 38.2: Triángulos

Para crear un triángulo CSS, defina un elemento con una anchura y una altura de 0 píxeles. La forma triangular se formará utilizando las propiedades del borde. Para un elemento con altura y anchura 0, los 4 bordes (superior, derecho, inferior, izquierdo) forman cada uno un triángulo. Aquí hay un elemento con 0 altura/anchura y 4 bordes de diferentes colores.



Configurando algunos bordes como transparentes y otros como de color podemos crear varios triángulos. Por ejemplo, en el triángulo Arriba, establecemos el borde inferior en el color deseado y, a continuación, establecemos los bordes izquierdo y derecho en transparente. He aquí una imagen con los bordes izquierdo y derecho ligeramente sombreados para mostrar cómo se forma el triángulo.



Las dimensiones del triángulo pueden modificarse cambiando la anchura de los bordes: más alto, más bajo, inclinado, etc. Todos los ejemplos siguientes muestran un triángulo de 50x50 píxeles.

Triángulo - Apuntando hacia arriba



```
<div class="triangulo-arriba"></div>
.triangulo-arriba {
  width: 0;
  height: 0;
  border-left: 25px solid transparent;
  border-right: 25px solid transparent;
  border-bottom: 50px solid rgb(246, 156, 85);
}
```

Triángulo - Apuntando hacia abajo



```
<div class="triangulo-abajo"></div>
.triangulo-abajo {
  width: 0;
  height: 0;
  border-left: 25px solid transparent;
  border-right: 25px solid transparent;
  border-top: 50px solid rgb(246, 156, 85);
}
```

Triángulo - Apuntando a la derecha



```
<div class="triangulo-derecha"></div>
.triangulo-derecha {
  width: 0;
  height: 0;
  border-top: 25px solid transparent;
  border-bottom: 25px solid transparent;
  border-left: 50px solid rgb(246, 156, 85);
}
```

Triángulo - Apuntando a la izquierda



```
<div class="triangulo-izquierda"></div>
.triangulo-izquierda {
  width: 0;
  height: 0;
  border-top: 25px solid transparent;
  border-bottom: 25px solid transparent;
  border-right: 50px solid rgb(246, 156, 85);
}
```

Triángulo - Apuntando arriba/derecha



```
<div class="triangulo-arriba-derecha"></div>
.triangulo-arriba-derecha {
  width: 0;
  height: 0;
  border-top: 50px solid rgb(246, 156, 85);
  border-left: 50px solid transparent;
}
```

Triángulo - Hacia arriba/izquierda



```
<div class="triangulo-arriba-izquierda"></div>
.triangulo-arriba-izquierda {
  width: 0;
  height: 0;
  border-top: 50px solid rgb(246, 156, 85);
  border-right: 50px solid transparent;
}
```

Triángulo - Apuntando hacia abajo/derecha



```
<div class="triangulo-abajo-derecha"></div>
.triangulo-abajo-derecha {
  width: 0;
  height: 0;
  border-bottom: 50px solid rgb(246, 156, 85);
  border-left: 50px solid transparent;
}
```

Triángulo - Apuntando hacia abajo/izquierda



```
<div class="triangulo-abajo-izquierda"></div>
.triangulo-abajo-izquierda {
  width: 0;
  height: 0;
  border-bottom: 50px solid rgb(246, 156, 85);
  border-right: 50px solid transparent;
}
```

Sección 38.3: Círculos y elipses

Círculo

Para crear un **círculo**, defina un elemento con el mismo `width` y `height` (un cuadrado) y establece la propiedad `border-radius` de este elemento al `50%`.



HTML

```
<div class="circulo"></div>
```

CSS

```
.circulo {
  width: 50px;
  height: 50px;
  background: rgb(246, 156, 85);
  border-radius: 50%;
}
```

Elipse

Una **elipse** es similar a un círculo, pero con valores diferentes para la `width` y la `height`.



HTML

```
<div class="ovalado"></div>
```

CSS

```
.ovalado {  
    width: 50px;  
    height: 80px;  
    background: rgb(246, 156, 85);  
    border-radius: 50%;  
}
```

Sección 38.4: Ráfagas

Una ráfaga es similar a una estrella, pero con las puntas a menor distancia del cuerpo. Piense en una forma de ráfaga como un cuadrado con cuadrados adicionales, ligeramente girados, superpuestos.

Los cuadrados adicionales se crean utilizando los elementos `::before` y `::after`.

Ráfaga de 8 puntos

Una ráfaga de 8 puntos son cuadrados de 2 capas. El cuadrado inferior es el propio elemento, el cuadrado adicional se crea utilizando el pseudoelemento `:before`. La parte inferior está girada 20°, el cuadrado superior está girado 135°.



```
<div class="burst-8"></div>  
.burst-8 {  
    background: rgb(246, 156, 85);  
    width: 40px;  
    height: 40px;  
    position: relative;  
    text-align: center;  
    -ms-transform: rotate(20deg);  
    transform: rotate(20deg);  
}  
.burst-8::before {  
    content: "";  
    position: absolute;  
    top: 0;  
    left: 0;  
    height: 40px;  
    width: 40px;  
    background: rgb(246, 156, 85);  
    -ms-transform: rotate(135deg);  
    transform: rotate(135deg);  
}
```

Ráfaga de 12 puntos

Una ráfaga de 12 puntos son 3 cuadrados estratificados. El recuadro inferior es el propio elemento, los recuadros adicionales se crean utilizando los pseudoelementos `:before` y `:after`. El cuadrado inferior gira 0°, el cuadrado siguiente gira 30° y el cuadrado superior gira 60°.



```
<div class="burst-12"></div>
.burst-12 {
  width: 40px;
  height: 40px;
  position: relative;
  text-align: center;
  background: rgb(246, 156, 85);
}
.burst-12::before, .burst-12::after {
  content: "";
  position: absolute;
  top: 0;
  left: 0;
  height: 40px;
  width: 40px;
  background: rgb(246, 156, 85);
}
.burst-12::before {
  -ms-transform: rotate(30deg);
  transform: rotate(30deg);
}
.burst-12::after {
  -ms-transform: rotate(60deg);
  transform: rotate(60deg);
}
```

Sección 38.5: Cuadrado

Para crear un cuadrado, defina un elemento con anchura y altura. En el ejemplo siguiente, tenemos un elemento con un `width` y un `height` de 100 píxeles cada uno.



```
<div class="cuadrado"></div>
.cuadrado {
  width: 100px;
  height: 100px;
  background: rgb(246, 156, 85);
}
```

Sección 38.6: Cubo

Este ejemplo muestra cómo crear un cubo utilizando los métodos de transformación 2D `skewX()` y `skewY()` en pseudoelementos.



HTML

```
<div class="cubo"></div>
```

CSS

```
.cubo {  
    background: #dc2e2e;  
    width: 100px;  
    height: 100px;  
    position: relative;  
    margin: 50px;  
}  
.cubo::before {  
    content: '';  
    display: inline-block;  
    background: #f15757;  
    width: 100px;  
    height: 20px;  
    transform: skewX(-40deg);  
    position: absolute;  
    top: -20px;  
    left: 8px;  
}  
.cubo::after {  
    content: '';  
    display: inline-block;  
    background: #9e1515;  
    width: 16px;  
    height: 100px;  
    transform: skewY(-50deg);  
    position: absolute;  
    top: -10px;  
    left: 100%;  
}
```

Sección 38.7: Pirámide

Este ejemplo muestra cómo crear una **pirámide** utilizando bordes y métodos de transformación 2D `skewY()` y `rotate()` en pseudoelementos.



HTML

```
<div class="piramide"></div>
```

CSS

```
.piramide {  
    width: 100px;  
    height: 200px;  
    position: relative;  
    margin: 50px;  
}  
.piramide::before, .piramide::after {  
    content: '';  
    display: inline-block;  
    width: 0;  
    height: 0;  
    border: 50px solid;  
    position: absolute;  
}  
.piramide::before {  
    border-color: transparent transparent #ff5656 transparent;  
    transform: scaleY(2) skewY(-40deg) rotate(45deg);  
}  
.piramide::after {  
    border-color: transparent transparent #d64444 transparent;  
    transform: scaleY(2) skewY(40deg) rotate(-45deg);  
}
```

Capítulo 39: Columnas

Sección 39.1: Ejemplo sencillo (recuento de columnas)

El diseño CSS multicolumna facilita la creación de varias columnas de texto.

Código

```
<div id="multi-columnas">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum</div>
.multi-columnas {
  -moz-column-count: 2;
  -webkit-column-count: 2;
  column-count: 2;
}
```

Resultado

Stack Overflow is a privately held website, the flagship site of the Stack Exchange Network,[4][5][6] created in 2008 by Jeff Atwood and Joel Spolsky. [7][8] It was created to be a more open alternative to earlier Q&A sites such as Experts-Exchange. The name for the website was chosen by voting in April 2008 by readers of Coding Horror, Atwood's popular programming blog.

The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and

answers up or down and edit questions and answers in a fashion similar to a wiki or Digg.[13] Users of Stack Overflow can earn reputation points and "badges"; for example, a person is awarded 10 reputation points for receiving an "up" vote on an answer given to a question, and can receive badges for their valued contributions, [14] which represents a kind of gamification of the traditional Q&A site or forum. All user-generated content is licensed under a Creative Commons Attribute-ShareAlike license.

Sección 39.2: Ancho de columna

La propiedad `column-width` establece el ancho mínimo de la columna. Si no se define `column-count` el navegador hará tantas columnas como quepan en el ancho disponible.

Código

```
<div id="multi-columnas">Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do  
eiusmod tempor incididunt tu labore et dolore magna aliqua. Ut enim ad minim veniam, quis  
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure  
dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur  
sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est  
laborum</div>  
.multi-columnas {  
  -moz-column-width: 100px;  
  -webkit-column-width: 100px;  
  column-width: 100px;  
}
```

Resultado

Stack Overflow is a privately held website, the flagship site of the Stack Exchange Network,[4][5] [6] created in 2008 by Jeff Atwood and Joel Spolsky. [7][8] It was created to be a more open alternative to earlier Q&A sites such as Experts-	Exchange. The name for the website was chosen by voting in April 2008 by readers of Coding Horror, Atwood's popular programming blog. The website serves as a platform for users to ask and answer	questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers in a fashion similar to a wiki or Digg.[13] Users of Stack Overflow can earn reputation	points and "badges"; for example, a person is awarded 10 reputation points for receiving an "up" vote on an answer given to a question, and can receive badges for their valued contributions, [14] which represents a kind of	gamification of the traditional Q&A site or forum. All user- generated content is licensed under a Creative Commons Attribute- ShareAlike license.
---	---	--	--	--

Capítulo 40: Múltiple columnas

CSS permite definir que los contenidos de los elementos se envuelvan en varias columnas con espacios y reglas entre ellas.

Sección 40.1: Crear múltiples columnas

HTML

```
<div class="contenido"> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. </div>
```

CSS

```
.contenido {  
  -webkit-column-count: 3; /* Chrome, Safari, Opera */  
  -moz-column-count: 3; /* Firefox */  
  column-count: 3;  
}
```

Sección 40.2: Ejemplo básico

Considere el siguiente código HTML:

```
<section>  
<p>Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.</p>  
<p>Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.</p>  
<p>Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.</p>  
</section>
```

Con el siguiente CSS aplicado, el contenido se divide en tres columnas separadas por una regla de columna gris de dos píxeles.

```
section {  
  columns: 3;  
  column-gap: 40px;  
  column-rule: 2px solid gray;  
}
```

Capítulo 41: Diseño inline-block

Sección 41.1: Barra de navegación justificada

La barra de navegación (menú) justificada horizontalmente tiene un cierto número de elementos que deben justificarse. El primer elemento (izquierda) no tiene margen izquierdo dentro del contenedor, el último elemento (derecha) no tiene margen derecho dentro del contenedor. La distancia entre los artículos es la misma, independientemente de la anchura de cada artículo.

HTML

```
<nav>
  <ul>
    <li>abc</li>
    <li>abcdefghijkl</li>
    <li>abcdef</li>
  </ul>
</nav>
```

CSS

```
nav {
  width: 100%;
  line-height: 1.4em;
}
ul {
  list-style: none;
  display: block;
  width: 100%;
  margin: 0;
  padding: 0;
  text-align: justify;
  margin-bottom: -1.4em;
}
ul:after {
  content: "";
  display: inline-block;
  width: 100%;
}
li {
  display: inline-block;
}
```

Notas

- Las etiquetas `nav`, `ul` y `li` se eligieron por su significado semántico de "lista de elementos de navegación (menú)". Por supuesto, también pueden utilizarse otras etiquetas por supuesto.
- El pseudoelemento `:after` causa una 'línea' extra en el `ul` y por lo tanto una altura extra y vacía de este bloque, empujando otros contenidos hacia abajo. Esto se soluciona con el `margin-bottom` negativo, que tiene que tener la misma magnitud que el `line-height` (pero negativa).
- Si la página es demasiado estrecha para que quepan todos los elementos, éstos se dividirán en una nueva línea (empezando por la derecha) y se justificarán en esta línea. La altura total del menú crecerá según sea necesario.

Capítulo 42: Herencia

Sección 42.1: Herencia automática

La herencia es un mecanismo fundamental de CSS por el que los valores calculados de algunas propiedades de un elemento se aplican a sus hijos. Esto es particularmente útil cuando se desea establecer un estilo global a sus elementos en lugar de tener que establecer dichas propiedades a cada elemento en su marcado.

Las propiedades comunes que se heredan automáticamente son: `font`, `color`, `text-align`, `line-height`.

Supongamos la siguiente hoja de estilo:

```
#miContenedor {  
    color: red;  
    padding: 5px;  
}
```

Esto aplicará `color: red` no sólo al elemento `<div>` sino también a los elementos `<h3>` y `<p>`. Sin embargo, debido a la naturaleza del `padding`, su valor **no** se heredará a esos elementos.

```
<div id="miContenedor">  
    <h3>Algún encabezamiento</h3>  
    <p>Algún párrafo</p>  
</div>
```

Sección 42.2: Herencia forzosa

Algunas propiedades no se heredan automáticamente de un elemento a sus hijos. Esto se debe a que normalmente se desea que estas propiedades sean exclusivas del elemento (o selección de elementos) al que se aplica la propiedad. Las propiedades más comunes son `margin`, `padding`, `background`, `display`, etc.

Sin embargo, a veces se desea heredar de todos modos. Para ello, podemos aplicar el valor `inherit` a la propiedad que debe ser heredada. El valor `inherit` puede aplicarse a *cualquier* propiedad CSS y a *cualquier* elemento HTML.

Supongamos la siguiente hoja de estilo:

```
#miContenedor {  
    color: red;  
    padding: 5px;  
}  
#miContenedor p {  
    padding: inherit;  
}
```

Esto aplicará `color: red` tanto a los elementos `<h3>` como `<p>` debido a la naturaleza hereditaria de la propiedad `color`. Sin embargo, el elemento `<p>` también heredará el valor del `padding` de su padre porque así se ha especificado.

```
<div id="miContenedor">  
    <h3>Algún encabezamiento</h3>  
    <p>Algún párrafo</p>  
</div>
```

Capítulo 43: Sprites de imágenes CSS

Sección 43.1: Una implementación básica

¿Qué es un sprite?

Un sprite de imagen es un único valor situado dentro de una hoja de sprite de imagen. Una hoja de sprite de imagen es un archivo de imagen que contiene más de un valor que puede extraerse de él. Es como si fuera un array de pequeñas imágenes dentro del mismo archivo de imagen.

Por ejemplo:



La imagen de arriba es una hoja de sprites de imagen, y cada una de esas estrellas es un sprite dentro de la hoja de sprites. Estas hojas de sprite son útiles porque mejoran el rendimiento al reducir el número de peticiones HTTP de un navegador.

¿Cómo se hace? Aquí tienes un código de ejemplo.

HTML

```
<div class="icono icono1"></div>
<div class="icono icono2"></div>
<div class="icono icono3"></div>
```

CSS

```
.icono {
    background: url("icons-sprite.png");
    display: inline-block;
    height: 20px;
    width: 20px;
}
.icono1 {
    background-position: 0px 0px;
}
.icono2 {
    background-position: -20px 0px;
}
.icono3 {
    background-position: -40px 0px;
}
```

Estableciendo la anchura y la altura del sprite y utilizando la propiedad `background-position` en CSS (con un valor x e y) puedes extraer fácilmente sprites de una hoja de sprites utilizando CSS.

Capítulo 44: Recortar y enmascarar

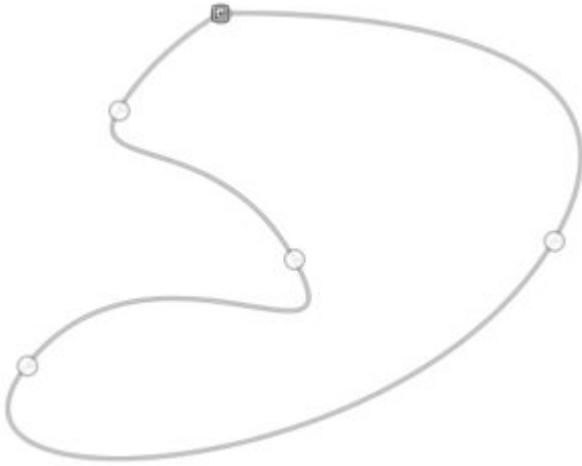
Parámetro	Detalles
<code>clip-source</code>	Una URL que puede apuntar a un elemento SVG en línea (o) a un elemento SVG en un archivo externo que contiene la definición de la ruta del clip.
<code>basic-shape</code>	Hace referencia a una de las funciones <code>inset()</code> , <code>circle()</code> , <code>ellipse()</code> o <code>polygon()</code> . Mediante una de estas funciones se define la trayectoria de recorte. Estas funciones de forma funcionan exactamente igual que en Formas para floats.
<code>clip-geometry-box</code>	Puede tener uno de los siguientes valores: <code>content-box</code> , <code>padding-box</code> , <code>border-box</code> , <code>margin-box</code> , <code>fill-box</code> , <code>stroke-box</code> , <code>view-box</code> como valores. Cuando se proporciona sin ningún valor para <code><basic-shape></code> , los bordes de la caja correspondiente se utiliza como la ruta para el recorte. Cuando se utiliza con una <code><basic-shape></code> , actúa como cuadro de referencia de la forma.
<code>mask-reference</code>	Puede ser <code>none</code> , una imagen o una URL de referencia a una fuente de imagen de máscara.
<code>repeat-style</code>	Especifica cómo debe repetirse o embaldosarse la máscara en los ejes X e Y. Los valores son <code>repeat-x</code> , <code>repeat-y</code> , <code>repeat</code> , <code>space</code> , <code>round</code> , <code>no-repeat</code> .
<code>mask-mode</code>	Puede ser <code>alpha</code> , <code>luminance</code> o <code>auto</code> e indica si la máscara debe tratarse como una máscara alfa o una máscara de luminancia. máscara alfa o de luminancia. Si no se proporciona ningún valor y la referencia de máscara es una imagen directa, se considerará como máscara alfa (o) si la máscara de referencia es una URL, se considerará como máscara de luminancia.
<code>position</code>	Especifica la posición de cada capa de máscara y su comportamiento es similar al de la propiedad <code>background-position</code> . El valor puede proporcionarse en sintaxis de 1 valor (como <code>top</code> , <code>10%</code>) o en sintaxis de 2 valores (como <code>top right</code> , <code>50% 50%</code>).
<code>geometry-box</code>	Especifica el cuadro al que debe recortarse la máscara (<i>área de pintura de la máscara</i>) o el cuadro que debe utilizarse como referencia para el origen de la máscara (<i>área de posicionamiento de la máscara</i>) dependiendo de la propiedad. La lista de valores posibles son <code>content-box</code> , <code>padding-box</code> , <code>border-box</code> , <code>margin-box</code> , <code>fill-box</code> , <code>stroke-box</code> , <code>view-box</code> . En la especificación del W3C valores está disponible en la especificación del W3C .
<code>bg-size</code>	Representa el tamaño de cada capa de máscara-imagen y tiene la misma sintaxis que <code>background-size</code> . El valor puede ser una longitud o porcentaje o <code>auto</code> o <code>cover</code> o <code>contain</code> . Longitud, porcentaje y <code>auto</code> pueden proporcionarse como un único valor o como uno para cada eje.
<code>compositing-operator</code>	Puede ser uno cualquiera entre <code>add</code> , <code>subtract</code> , <code>exclude</code> , <code>multiply</code> por capa y define el tipo de operación de composición que debe utilizarse para esta capa con las que están por debajo de ella. Encontrará explicaciones detalladas sobre cada valor en las especificaciones del W3C .

Sección 44.1: Recortar y enmascarar: Generalidades y diferencias

Con **Recortar** y **Enmascarar** puede hacer transparentes u opacas algunas partes específicas de los elementos. Ambos pueden a cualquier elemento HTML.

Recortar

Los clips son trazados vectoriales. Fuera de esta ruta el elemento será transparente, dentro es opaco. Por lo tanto, puede definir una propiedad `clip-path` en los elementos. Cada elemento gráfico que también existe en SVG se puede utilizar aquí como una para definir la ruta. Algunos ejemplos son `circle()`, `polygon()` o `ellipse()`.



Ejemplo

```
clip-path: circle(100px at center);
```

El elemento sólo será visible dentro de este círculo, que se sitúa en el centro del elemento y tiene un radio de 100px. radio de 100px.

Enmascaramiento

Las máscaras son similares a los Clips, pero en lugar de definir una ruta se define una máscara que capas sobre el elemento. Puede imaginar esta máscara como una imagen compuesta principalmente por dos colores: blanco y negro.

Máscara de luminosidad: El negro significa que la región es opaca, y el blanco que es transparente, pero también hay una zona gris que es semitransparente, para que puedas hacer transiciones suaves.

Máscara Alfa: Sólo en las zonas transparentes de la máscara el elemento será opaco.



Esta imagen, por ejemplo, se puede utilizar como una máscara de luminancia para hacer para un elemento una transición muy suave de derecha a izquierda y de opaco a transparente.

La propiedad `mask` permite especificar el tipo de máscara y la imagen que se utilizará como capa.

Ejemplo

```
mask: url(mascaras.svg#rectangulo) luminance;
```

Un elemento llamado `rectangulo` definido en `mascaras.svg` se utilizará como **máscara de luminosidad** sobre el elemento.

Sección 44.2: Máscara simple que desvanece una imagen de sólido a transparente

HTML

```
<div></div>
```

CSS

```
div {  
    height: 200px;  
    width: 200px;  
    background: url(https://picsum.photos/200/200);  
    mask-image: linear-gradient(to right, white, transparent);  
}
```

En el ejemplo anterior hay un elemento con una imagen como fondo. La máscara que se aplica a la imagen (mediante CSS) hace que parezca que se desvanece de izquierda a derecha.

El enmascaramiento se consigue utilizando un `linear-gradient` que va de blanco (a la izquierda) a transparente (a la derecha) como máscara. Como se trata de una máscara alfa, la imagen se vuelve transparente donde la máscara es transparente.

Salida sin la máscara:



Salida con la máscara:



Nota: Como se menciona en las observaciones, el ejemplo anterior funcionaría en Chrome, Safari y Opera sólo cuando se utiliza con el prefijo `-webkit`. Este ejemplo (con un `linear-gradient` como imagen de máscara) aún no es compatible con Firefox.

Sección 44.3: Recorte (circle)

HTML

```
<div></div>
```

CSS

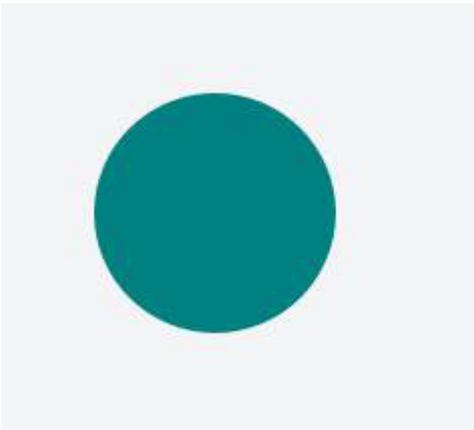
```
div{  
  width: 200px;  
  height: 200px;  
  background: teal;  
  clip-path: circle(30% at 50% 50%); /* consulte las observaciones antes del uso */  
}
```

Este ejemplo muestra cómo recortar un `div` a un círculo. El elemento se recorta en un círculo cuyo radio es del 30% en función de las dimensiones de la caja de referencia con su punto central en el centro de la caja de referencia. En este caso, dado que no se proporciona `<clip-geometry-box>` (en otras palabras, caja de referencia), se utilizará el `border-box` del elemento como la cuadro de referencia.

El círculo debe tener un radio y un centro con coordenadas (x, y):

```
circle(radius at x y)
```

Salida:



Sección 44.4: Recorte (polygon)

HTML

```
<div></div>
```

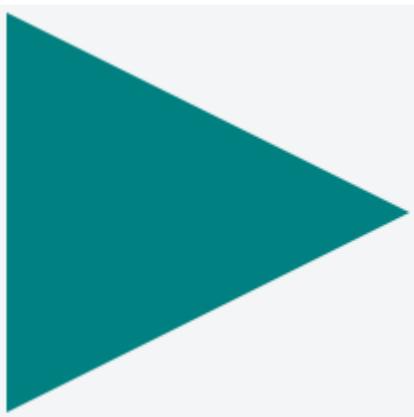
CSS

```
div{  
  width:200px;  
  height:200px;  
  background:teal;  
  clip-path: polygon(0 0, 0 100%, 100% 50%); /* consulte las observaciones antes del uso */  
}
```

En el ejemplo anterior, se utiliza una ruta de recorte **poligonal** para recortar el elemento cuadrado (200 x 200) en forma de triángulo. La forma de salida es un triángulo porque el camino empieza en (es decir, las primeras coordenadas están en) `0 0` - que es la esquina superior izquierda de la caja, luego va a `0 100%` - que es la esquina inferior izquierda de la caja y finalmente a `100% 50%` que es nada más que el punto medio derecho de la caja. Estas trayectorias se cierran solas (es decir, el punto inicial será el punto final), por lo que la forma final es la de un triángulo. punto final), por lo que la forma final es la de un triángulo.

También se puede utilizar en un elemento con una imagen o un degradado como fondo.

Salida:



Sección 44.5: Uso de máscaras para cortar un agujero en el centro de una imagen

HTML

```
<div></div>
```

CSS

```
div {  
  width: 200px;  
  height: 200px;  
  background: url(https://picsum.photos/200/200);  
  mask-image: radial-gradient(circle farthest-side at center, transparent 49%, white 50%); /*  
  consulte observaciones antes de usar */  
}
```

Imagen sin máscara:



Imagen con máscara:



Sección 44.6: Uso de máscaras para crear imágenes con formas irregulares irregulares

HTML

```
<div></div>
```

CSS

```
div { /* comprueba las observaciones antes de usar */  
  height: 200px;  
  width: 400px;  
  background-image: url(https://picsum.photos/200/200);  
  mask-image: linear-gradient(to top right, transparent 49.5%, white 50.5%), linear-  
  gradient(to top left, transparent 49.5%, white 50.5%), linear-gradient(white, white);  
  mask-size: 75% 25%, 25% 25%, 100% 75%;  
  mask-position: bottom left, bottom right, top left;  
  mask-repeat: no-repeat;  
}
```

En el ejemplo anterior, tres imágenes con degradado lineal (que colocadas en sus posiciones apropiadas cubrirían el 100% x 100% del tamaño del contenedor) se utilizan como máscaras para producir un corte triangular transparente en la parte inferior de la imagen. triangular transparente en la parte inferior de la imagen.

Imagen sin la máscara:



Imagen con la máscara:



Capítulo 45: Fragmentación

Valor	Descripción
<code>auto</code>	Por defecto. Saltos de página automáticos
<code>always</code>	Insertar siempre un salto de página
<code>avoid</code>	Evitar el salto de página (si es posible)
<code>left</code>	Insertar saltos de página para que la página siguiente tenga formato de página izquierda
<code>right</code>	Insertar saltos de página para que la página siguiente tenga formato de página derecha
<code>initial</code>	Establece esta propiedad a su valor por defecto
<code>inherit</code>	Hereda esta propiedad de su elemento padre

Sección 45.1: @media print page-break

```
@media print {  
  p {  
    page-break-inside: avoid;  
  }  
  h1 {  
    page-break-before: always;  
  }  
  h2 {  
    page-break-after: avoid;  
  }  
}
```

Este código hace 3 cosas:

- impide un salto de página dentro de cualquier etiqueta `p`, lo que significa que un párrafo nunca se romperá en dos páginas, si es posible.
- fuerza un salto de página antes de todos los encabezados `h1`, lo que significa que antes de cada aparición de `h1`, habrá un salto de página.
- evita los saltos de página justo después de cualquier `h2`.

Capítulo 46: Modelo de objetos CSS (CSSOM)

Sección 46.1: Añadir una regla de background-image a través de CSSOM

Para añadir una regla de imagen de fondo a través de CSSOM, primero obtenga una referencia a las reglas de la primera hoja de estilos:

```
var stylesheet = document.styleSheets[0].cssRules;
```

A continuación, obtener una referencia al final de la hoja de estilos:

```
var end = stylesheet.length - 1;
```

Por último, inserte una regla de imagen de fondo para el elemento body al final de la hoja de estilo:

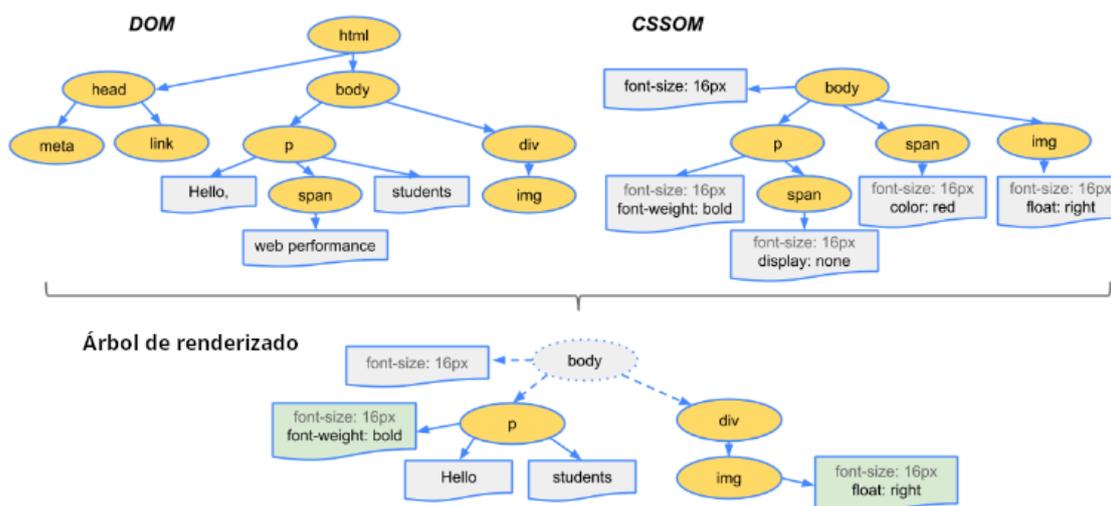
```
stylesheet.insertRule("body { background-image: url('http://cdn.sstatic.net/Sites/stackoverflow/img/favicon.ico'); }", end);
```

Sección 46.2: Introducción

El navegador identifica los tokens de la hoja de estilo y los convierte en nodos que se enlazan en una estructura de árbol. El mapa completo de todos los nodos con sus estilos asociados de una página sería el Modelo de Objetos CSS.

Para visualizar la página web, el navegador sigue los siguientes pasos.

1. El navegador web examina tu HTML y construye el DOM (Document Object Model).
2. El navegador web examina tu CSS y construye el CSSOM (CSS Object Model).
3. El navegador web combina el DOM y el CSSOM para crear un árbol de renderizado. El navegador muestra su página web.



Capítulo 47: Característica de las Queries

Parámetro	Detalles
<code>(property: value)</code>	Evalúa true si el navegador puede manejar la regla CSS. Los paréntesis alrededor de la regla son necesarios.
<code>and</code>	Devuelve verdadero sólo si las condiciones anterior y siguiente son verdaderas.
<code>not</code>	Anula la siguiente condición
<code>or</code>	Devuelve verdadero si la condición anterior o la siguiente son verdaderas.
<code>(...)</code>	Grupos condiciones

Sección 47.1: Uso básico de @supports

```
@supports (display: flex) {  
  /* Flexbox está disponible, así que úsalo */  
  .my-container {  
    display: flex;  
  }  
}
```

En términos de sintaxis, `@supports` es muy similar a `@media`, pero en lugar de detectar el tamaño y la orientación de la pantalla, `@supports` detectará si el navegador puede manejar una regla CSS determinada.

En lugar de hacer algo como `@supports (flex)`, observe que la regla es `@supports (display: flex)`.

Sección 47.2: Encadenamiento de detecciones de características

Para detectar varias características a la vez, utilice el operador `and`.

```
@supports (transform: translateZ(1px)) and (transform-style: preserve-3d) and (perspective: 1px)  
{  
  /* Probablemente hacer algunas cosas 3d de lujo aquí */  
}
```

También existe un operador `or` y un operador `not`:

```
@supports (display: flex) or (display: table-cell) {  
  /* Se utilizará si el navegador admite flexbox o display: table-cell */  
}  
@supports not (-webkit-transform: translate(0, 0, 0)) {  
  /* *No* se utilizará si el navegador admite -webkit-transform: translate(...) */  
}
```

Para disfrutar al máximo de `@supports`, pruebe a agrupar expresiones lógicas con paréntesis:

```
@supports ((display: block) and (zoom: 1)) or ((display: flex) and (not (display: table-cell)))  
or (transform: translateX(1px)) {  
  /* ... */  
}
```

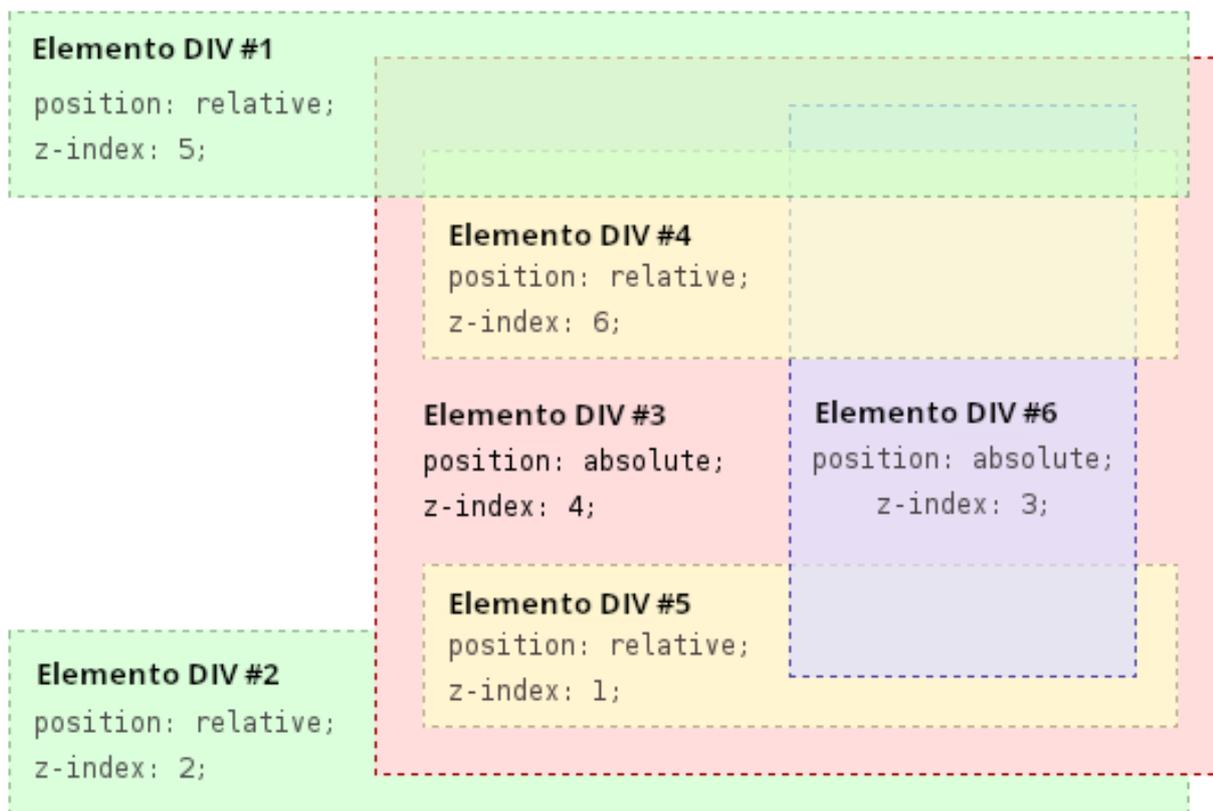
Esto funcionará si el navegador

1. Soporta `display: block` Y `zoom: 1`, o
2. Soporta `display: flex` Y NO `display: table-cell`, o
3. Soporta `transform: translateX(1px)`.

Capítulo 48: Contexto de apilamiento

Sección 48.1: Contexto de apilamiento

En este ejemplo, cada elemento posicionado crea su propio contexto de apilamiento, debido a su posicionamiento y a sus valores z-index de cada elemento. La jerarquía de contextos de apilamiento se organiza del siguiente modo:



- Raíz
 - DIV #1
 - DIV #2
 - DIV #3
 - DIV #4
 - DIV #5
 - DIV #6

Es importante notar que DIV #4, DIV #5 y DIV #6 son hijos de DIV #3, por lo que el apilamiento de esos elementos está completamente resuelto dentro del DIV#3. Una vez completado el apilamiento y renderizado dentro del DIV #3, todo el elemento DIV #3 es pasado para su apilamiento en el elemento raíz con respecto al DIV de su hermano.

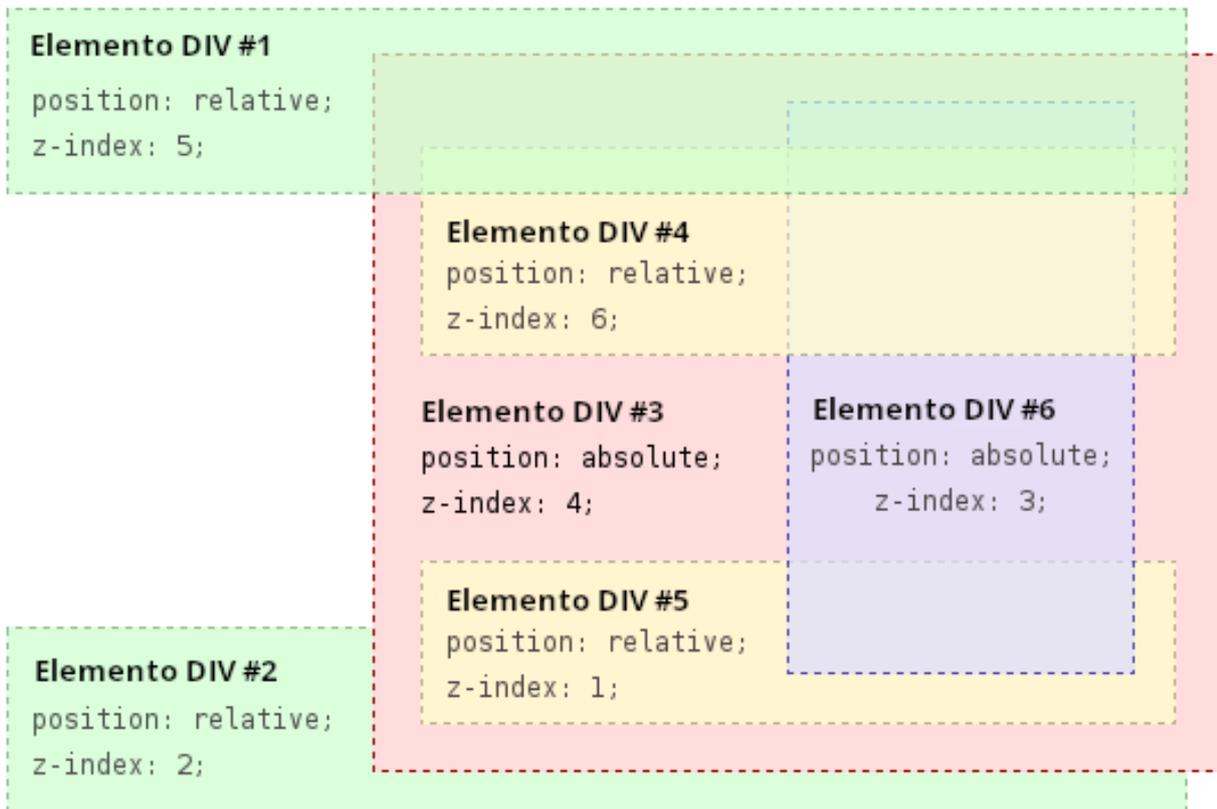
HTML

```
<div id="div1">
  <h1>Elemento DIV #1</h1>
  <code>position: relative;<br/> z-index: 5;</code>
</div>
<div id="div2">
  <h1>Elemento DIV #2</h1>
  <code>position: relative;<br/> z-index: 2;</code>
</div>
<div id="div3">
  <div id="div4">
    <h1>Elemento DIV #4</h1>
    <code>position: relative;<br/> z-index: 6;</code>
  </div>
  <h1>Elemento DIV #3</h1>
  <code>position: absolute;<br/> z-index: 4;</code>
  <div id="div5">
    <h1>Elemento DIV #5</h1>
    <code>position: relative;<br/> z-index: 1;</code>
  </div>
  <div id="div6">
    <h1>Elemento DIV #6</h1>
    <code>position: absolute;<br/> z-index: 3;</code>
  </div>
</div>
```

CSS

```
* { margin: 0; }
html {
  padding: 20px;
  font: 12px/20px Arial, sans-serif;
}
div {
  opacity: 0.7;
  position: relative;
}
h1 {
  font: inherit;
  font-weight: bold;
}
#div1, #div2 {
  border: 1px dashed #696;
  padding: 10px;
  background-color: #cfc;
}
#div1 {
  z-index: 5;
  margin-bottom: 190px;
}
#div2 {
  z-index: 2;
}
#div3 {
  z-index: 4;
  opacity: 1;
  position: absolute;
  top: 40px;
  left: 180px;
  width: 330px;
  border: 1px dashed #900;
  background-color: #fdd;
  padding: 40px 20px 20px;
}
#div4, #div5 {
  border: 1px dashed #996;
  background-color: #ffc;
}
#div4 {
  z-index: 6;
  margin-bottom: 15px;
  padding: 25px 10px 5px;
}
#div5 {
  z-index: 1;
  margin-top: 15px;
  padding: 5px 10px;
}
#div6 {
  z-index: 3;
  position: absolute;
  top: 20px;
  left: 180px;
  width: 150px;
  height: 125px;
  border: 1px dashed #009;
  padding-top: 125px;
  background-color: #ddf;
  text-align: center;
}
```

Resultado



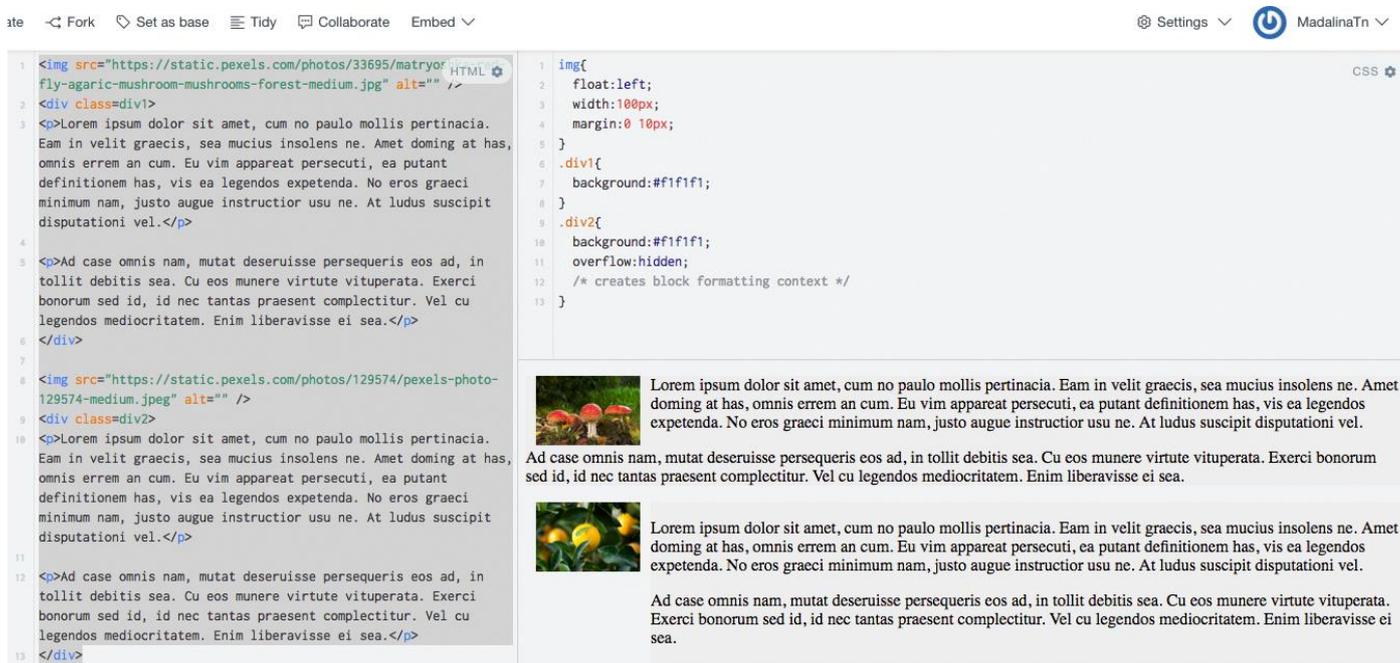
Fuente:

https://developer.mozilla.org/es/docs/Web/CSS/CSS_Positioning/Understanding_z_index/The_stacking_context

Capítulo 49: Contextos de formato de bloque

Sección 49.1: Usando la propiedad overflow con un valor distinto de visible

```
img{
  float:left;
  width:100px;
  margin:0 10px;
}
.div1{
  background:#f1f1f1;
  /* no crea contexto de formato de bloque */
}
.div2{
  background:#f1f1f1;
  overflow:hidden;
  /* crea un contexto de formato de bloque */
}
```



The screenshot shows a web editor interface with two main panels. The left panel displays HTML code, and the right panel displays CSS code. Below the code, there is a visual preview of the rendered page.

HTML Code (Left Panel):

```
1 
2 <div class=div1>
3 <p>Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia. Eam in velit graecis, sea mucus insolens ne. Amet doming at has, omnis errem an cum. Eu vim appareat persecuti, ea putant definitionem has, vis ea legendos expetenda. No eros graeci minimum nam, justo augue instructor usu ne. At ludus suscipit disputationi vel.</p>
4
5 <p>Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitis sea. Cu eos munere virtute vituperata. Exerci bonorum sed id, id nec tantas praesent complectitur. Vel cu legendos mediocritatem. Enim liberavisse ei sea.</p>
6 </div>
7
8 
9 <div class=div2>
10 <p>Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia. Eam in velit graecis, sea mucus insolens ne. Amet doming at has, omnis errem an cum. Eu vim appareat persecuti, ea putant definitionem has, vis ea legendos expetenda. No eros graeci minimum nam, justo augue instructor usu ne. At ludus suscipit disputationi vel.</p>
11
12 <p>Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitis sea. Cu eos munere virtute vituperata. Exerci bonorum sed id, id nec tantas praesent complectitur. Vel cu legendos mediocritatem. Enim liberavisse ei sea.</p>
13 </div>
```

CSS Code (Right Panel):

```
1 img{
2   float:left;
3   width:100px;
4   margin:0 10px;
5 }
6 .div1{
7   background:#f1f1f1;
8 }
9 .div2{
10  background:#f1f1f1;
11  overflow:hidden;
12  /* creates block formatting context */
13 }
```

Visual Preview (Bottom Panel):

The preview shows a page with two columns of text. The first column contains a floating image of mushrooms (from the first `` tag) and two paragraphs of Lorem Ipsum text. The second column contains a floating image of lemons (from the second `` tag) and two paragraphs of Lorem Ipsum text. The text in the second column is contained within a light gray background, demonstrating the effect of the `overflow: hidden` property on the `.div2` context.

Este ejemplo que muestra cómo varios párrafos interactuarán con una imagen flotante es similar a [este ejemplo](#), en [css-tricks.com](#).

MDN: <https://developer.mozilla.org/es/docs/Web/CSS/overflow>

Capítulo 50: Centrado vertical

Sección 50.1: Centrado con display: table

HTML

```
<div class="wrapper">
  <div class="exterior">
    <div class="interior">
      centrado
    </div>
  </div>
</div>
```

CSS

```
.wrapper {
  height: 600px;
  text-align: center;
}
.exterior {
  display: table;
  height: 100%;
  width: 100%;
}
.exterior .interior {
  display: table-cell;
  text-align: center;
  vertical-align: middle;
}
```

Sección 50.2: Centrado con Flexbox

HTML

```
<div class="contenedor">
  <div class="hijo"></div>
</div>
```

CSS

```
.contenedor {
  height: 500px;
  width: 500px;
  display: flex; // Usa Flexbox
  align-items: center; // Esto centra a los hijos verticalmente en el padre.
  justify-content: center; // Esto centra a los hijos horizontalmente.
  background: white;
}
.hijo {
  width: 100px;
  height: 100px;
  background: blue;
}
```

Sección 50.3: Centrado con transform

HTML

```
<div class="wrapper">
  <div class="centrado">
    centrado
  </div>
</div>
```

CSS

```
.wrapper {
  position: relative;
  height: 600px;
}
.centrado {
  position: absolute;
  z-index: 999;
  transform: translate(-50%, -50%);
  top: 50%;
  left: 50%;
}
```

Sección 50.4: Centrar texto con line-height

HTML

```
<div class="contenedor">
  <span>centrado verticalmente</span>
</div>
```

CSS

```
.contenedor{
  height: 50px; /* establece altura */
  line-height: 50px; /* ajustar line-height igual que la altura */
  vertical-align: middle; /* funciona sin esta regla, pero es bueno tenerla explícitamente establecida */
}
```

Nota: Este método sólo centrará verticalmente una *única línea de texto*. No centrará correctamente los elementos del bloque y si el texto se rompe en una nueva línea, tendrá dos líneas de texto muy altas.

Sección 50.5: Centrado con position: absolute

HTML

```
<div class="wrapper">

</div>
```

CSS

```
.wrapper{
  position:relative;
  height: 600px;
}
.wrapper img {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  margin: auto;
}
```

Si quieres centrar otro elemento que no sean imágenes, entonces debes dar altura y anchura a ese elemento.

HTML

```
<div class="wrapper">
  <div class="hijo">
    haz que me centre
  </div>
</div>
```

CSS

```
.wrapper{
  position:relative;
  height: 600px;
}
.wrapper .hijo {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  margin: auto;
  width: 200px;
  height: 30px;
  border: 1px solid #f00;
}
```

Sección 50.6: Centrado con pseudoelemento

HTML

```
<div class="wrapper">
  <div class="contenido"></div>
</div>
```

CSS

```
.wrapper{
  min-height: 600px;
}
.wrapper:before{
  content: "";
  display: inline-block;
  height: 100%;
  vertical-align: middle;
}
.contenido {
  display: inline-block;
  height: 80px;
  vertical-align: middle;
}
```

Este método se utiliza mejor en los casos en los que se tiene un `.contenido` de altura variable centrado dentro de `.wrapper`; y se desea que la altura de `.wrapper` se expanda cuando la altura de `.contenido` supere la altura mínima de `.wrapper`.

Capítulo 51: Ajuste y colocación de objetos

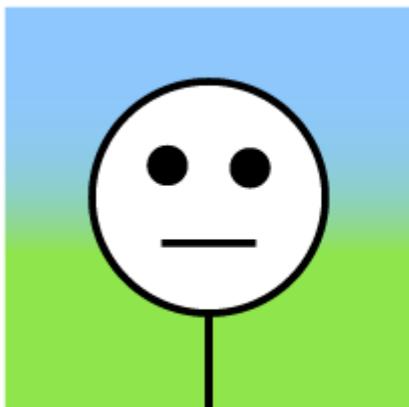
Sección 51.1: object-fit

La propiedad **object-fit** definirá cómo encajará un elemento en una caja con una altura y anchura establecidas. Normalmente aplicado a una imagen o vídeo, **object-fit** acepta los cinco valores siguientes:

FILL (RELLENAR)

`object-fit: fill;`

original image



object-fit: fill;

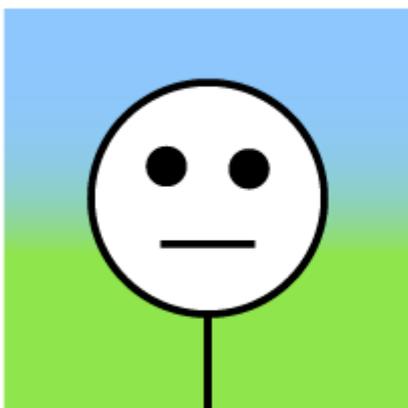


Rellenar estira la imagen para ajustarla a la caja de contenido sin tener en cuenta la relación de aspecto original de la imagen.

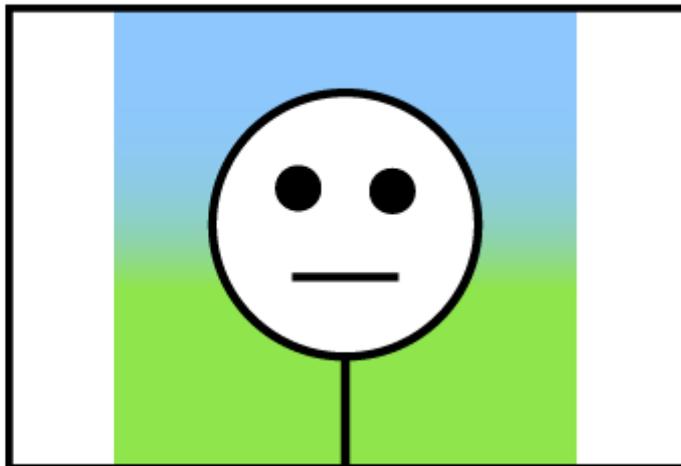
CONTAIN (CONTENER)

`object-fit: contain;`

original image



object-fit: contain;



Contain ajusta la imagen a la altura o anchura de la caja manteniendo la relación de aspecto de la imagen.

COVER (CUBRIR)

`object-fit: cover;`

original image



`object-fit: cover;`

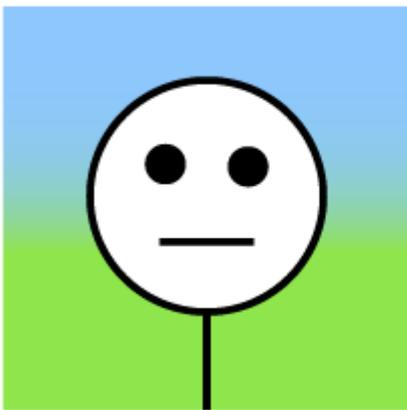


Cubrir rellena toda la caja con la imagen. La relación de aspecto de la imagen se mantiene, pero la imagen se recorta a las dimensiones de la caja.

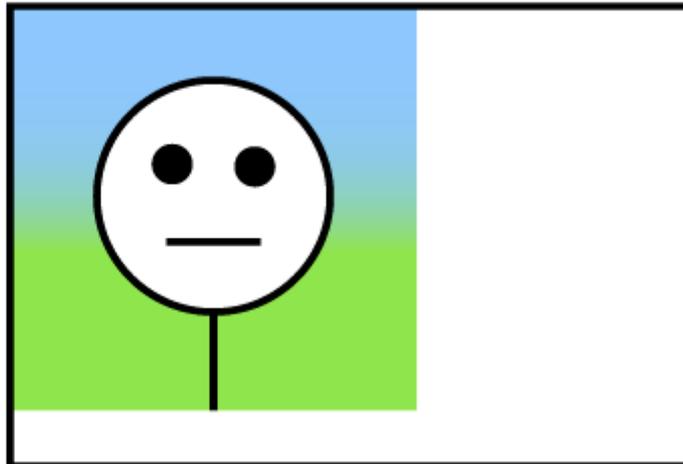
NONE (NADA)

`object-fit: none;`

original image



`object-fit: none;`



Ninguno ignora el tamaño de la caja y no se redimensiona.

SCALE-DOWN (REDUCIR-ESCALA)

`object-fit: scale-down;`

Reducir el tamaño del objeto como `none` o como `contain`. Muestra la opción que resulte en una imagen de menor tamaño.

original image



object-fit: scale-down;



Capítulo 52: Patrones de diseño CSS

Estos ejemplos sirven para documentar patrones de diseño específicos de CSS como [BEM](#), [OOCSS](#) y [SMACSS](#).

Estos ejemplos NO son para documentar frameworks CSS como [Bootstrap](#) o [Foundation](#).

Sección 52.1: BEM

BEM son las siglas de **Bloques**, **Elementos** y **Modificadores**. Se trata de una metodología concebida inicialmente por la empresa tecnológica rusa [Yandex](#), pero que también ha ganado adeptos entre los desarrolladores web estadounidenses y europeos occidentales.

Como la misma implica, la metodología BEM se trata de componentization de su código HTML y CSS en tres tipos

de componentes:

Bloques: entidades independientes que tienen sentido por sí solas.

Ejemplos: `header`, `container`, `menu`, `checkbox` & `textbox`

Elementos: Parte de los bloques que no tienen significado autónomo y están semánticamente ligados a sus bloques.

Ejemplos: `menu item`, `list item`, `checkbox caption` & `header title`

Modificadores: Banderas en un bloque o elemento, utilizadas para cambiar la apariencia o el comportamiento.

Ejemplos: `disabled`, `highlighted`, `checked`, `fixed`, `size big` & `color yellow`

El objetivo de BEM es seguir optimizando la legibilidad, el mantenimiento y la flexibilidad de su código CSS. La forma de conseguirlo es aplicar las siguientes reglas.

- Los estilos de bloque nunca dependen de otros elementos de la página.
- Los bloques deben tener un nombre sencillo y corto y evitar los caracteres `_` o `-`.
- Al aplicar estilos a los elementos, utilice selectores del formato `nombrebloque_nombredelelemento`
- Al aplicar modificadores de estilo, utilice selectores con el formato `nombrebloque--nombremodificador` y `nombrebloque_nombredelelemento_nombremodificador`.
- Los elementos o bloques que tienen modificadores deben heredar todo del bloque o elemento que modifican excepto las propiedades que el modificador debe modificar.

Ejemplo de código

Si aplicas BEM a tus elementos de formulario, tus selectores CSS deberían tener este aspecto:

```
.form { } // Bloque
.form--tema-navidad { } // Bloque + modificador
.form--simple { } // Bloque + modificador
.form__input { } // Bloque > elemento
.form__submit { } // Bloque > elemento
.form__submit--disabled { } // Bloque > elemento + modificador
```

El HTML correspondiente debería tener este aspecto:

```
<form class="form form--tema-navidad form--simple">  
  <input class="form__input" type="text" />  
  <input class="form__submit form__submit--disabled" type="submit" />  
</form>
```

Capítulo 53: Soporte de navegadores y prefijos

Prefijo

-webkit-

-moz-

-ms-

-o-, -xv-

-khtml-

Navegador(es)

Google Chrome, Safari, versiones más recientes de Opera 12 y superiores, navegadores Android, Blackberry y UC.

Mozilla Firefox

Internet Explorer, Edge

Opera hasta la versión 12

Konquerer

Sección 53.1: Transiciones

```
div {  
  -webkit-transition: all 4s ease;  
  -moz-transition: all 4s ease;  
  -o-transition: all 4s ease;  
  transition: all 4s ease;  
}
```

Sección 53.2: transform

```
div {  
  -webkit-transform: rotate(45deg);  
  -moz-transform: rotate(45deg);  
  -ms-transform: rotate(45deg);  
  -o-transform: rotate(45deg);  
  transform: rotate(45deg);  
}
```

Capítulo 54: Normalización de los estilos del navegador

Cada navegador tiene un conjunto predeterminado de estilos CSS que utiliza para representar los elementos. Estos estilos por defecto pueden no ser consistentes en todos los navegadores porque: las especificaciones del lenguaje no son claras por lo que los estilos base están sujetos a interpretación, los navegadores pueden no seguir las especificaciones que se dan, o los navegadores pueden no tener estilos por defecto para nuevos elementos HTML. En consecuencia, es posible que se desee normalizar los estilos por defecto en tantos navegadores como sea posible. Navegadores.

Sección 54.1: normalize.css

Los navegadores tienen un conjunto predeterminado de estilos CSS que utilizan para representar los elementos. Algunos de estos estilos pueden incluso personalizarse utilizando la configuración del navegador para cambiar, por ejemplo, las definiciones predeterminadas de tipo y tamaño de letra. Los estilos contienen, entre otras cosas, la definición de los elementos que deben aparecer en bloque o en línea.

Dado que las especificaciones del lenguaje dejan cierto margen de maniobra a estos estilos por defecto y que los navegadores pueden no seguir las especificaciones correctamente, pueden diferir de un navegador a otro.

Aquí es donde [normalize.css](#) entra en juego. Anula las incoherencias más comunes y corrige errores conocidos.

¿Para qué sirve?

- Conserva los valores predeterminados útiles, a diferencia de muchos restablecimientos de CSS.
- Normaliza los estilos de una amplia gama de elementos.
- Corrige errores e incoherencias comunes de los navegadores.
- Mejora la usabilidad con sutiles modificaciones.
- Explica lo que hace el código mediante comentarios detallados.

Por lo tanto, al incluir `normalize.css` en su proyecto su diseño se verá más parecido y coherente en diferentes navegadores.

Diferencia con reset.css

Es posible que hayas oído hablar de `reset.css`. ¿Cuál es la diferencia entre ambos?

Mientras que `normalize.css` proporciona coherencia estableciendo diferentes propiedades a valores predeterminados unificados, `reset.css` logra la coherencia mediante la **eliminación** de todos los estilos básicos que el navegador puede aplicar. Aunque esto pueda parecer una buena idea al principio, en realidad significa que tienes que escribir todas las reglas tú mismo, lo que va en contra de tener una norma sólida.

Sección 54.2: Enfoques y ejemplos

Los reajustes de CSS adoptan enfoques distintos a los predeterminados del navegador. El Reset CSS de Eric Meyer existe desde hace tiempo. Su enfoque anula de raíz muchos de los elementos del navegador conocidos por causar problemas. El siguiente es de su versión (v2.0 | 20110126) CSS Reset.

```
html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a,
abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small, strike,
strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form, label,
legend, table, caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup, menu, nav, output, ruby, section, summary, time,
mark, audio, video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
```

[Reset CSS de Eric Meyer](#)

Normalice CSS por otro, se ocupa de muchos de ellos por separado. A continuación se muestra un ejemplo de la versión (v4.2.0) del código.

```
/**
 * 1. Change the default font family in all browsers (opinionated).
 * 2. Correct the line height in all browsers.
 * 3. Prevent adjustments of font size after orientation changes in IE and iOS.
 */
/* Document
===== */
html {
    font-family: sans-serif; /* 1 */
    line-height: 1.15; /* 2 */
    -ms-text-size-adjust: 100%; /* 3 */
    -webkit-text-size-adjust: 100%; /* 3 */
}
/* Sections
===== */
/**
 * Remove the margin in all browsers (opinionated).
 */
body {
    margin: 0;
}
/**
 * Add the correct display in IE 9-.
 */
article,
aside,
footer,
header,
nav,
section {
    display: block;
}
/**
 * Correct the font size and margin on `h1` elements within `section` and
 * `article` contexts in Chrome, Firefox, and Safari.
 */
h1 {
    font-size: 2em;
    margin: 0.67em 0;
}
```

[Normalize CSS](#)

Capítulo 55: Trucos para Internet Explorer

Sección 55.1: Añadir soporte para inline-block a IE6 e IE7

```
display: inline-block;
```

La propiedad `display` con el valor `inline-block` no es compatible con Internet Explorer 6 y 7.

```
zoom: 1;  
*display: inline;
```

La propiedad `zoom` activa la función `hasLayout` de los elementos, y sólo está disponible en Internet Explorer. El `*display` se asegura de que la propiedad inválida se ejecute sólo en los navegadores afectados. Otros navegadores simplemente ignorarán la regla.

Sección 55.2: Modo de alto contraste en Internet Explorer 10 y superior

En Internet Explorer 10+ y Edge, Microsoft proporciona el selector de medios `-ms-high-contrast` para exponer la configuración de "Contraste alto" desde el navegador, lo que permite al programador ajustar los estilos de su sitio en consecuencia.

El selector `-ms-high-contrast` tiene 3 estados: `active`, `black-on-white` y `white-on-black`. En IE10+ también tenía un estado `none`, pero eso ya no se admite en Edge en adelante.

Ejemplos

```
@media screen and (-ms-high-contrast: active), (-ms-high-contrast: black-on-white) {  
  .encabezado {  
    background: #fff;  
    color: #000;  
  }  
}
```

Esto cambiará el fondo del encabezado a blanco y el color del texto a negro cuando el modo de alto contraste esté activo y está en modo `black-on-white`.

```
@media screen and (-ms-high-contrast: white-on-black) {  
  .encabezado {  
    background: #000;  
    color: #fff;  
  }  
}
```

Similar al primer ejemplo, pero este selecciona específicamente el estado `white-on-black` solamente, e invierte los colores del encabezado a un fondo negro con texto blanco.

Más información:

[Documentación de Microsoft](#) sobre `-ms-high-contrast`

Sección 55.3: Sólo Internet Explorer 6 e Internet Explorer 7

Para dirigirse a Internet Explorer 6 e Internet Explorer 7, inicie sus propiedades con `*`:

```
.ocultar-en-ie6-y-ie7 {  
  *display: none; // Esta línea sólo se procesa en IE6 e IE7  
}
```

Los prefijos no alfanuméricos (distintos de guiones y guiones bajos) son ignorados en IE6 e IE7, por lo que este truco funciona para cualquier propiedad sin el par `propiedad: valor`.

Sección 55.4: Sólo Internet Explorer 8

Para Internet Explorer 8, envuelva sus selectores dentro de `@media \0 screen { }`:

```
@media \0 screen {  
    .ocultar-en-ie8 {  
        display : none;  
    }  
}
```

Todo lo que hay entre `@media \0 screen { }` es procesado sólo por IE.

Capítulo 56: Rendimiento

Sección 56.1: Utiliza transform y opacity para evitar el diseño desencadenante

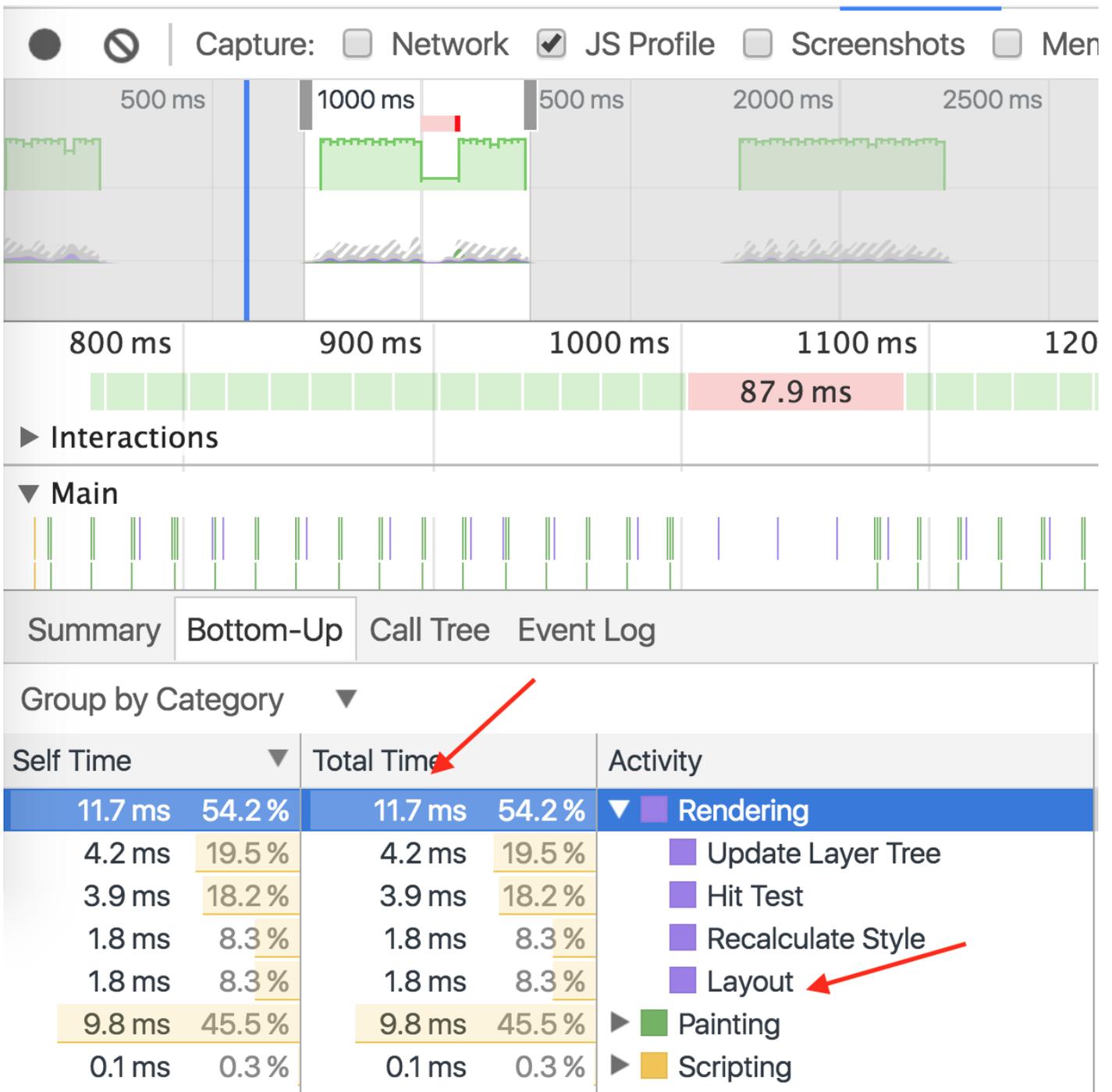
Cambiar algún atributo CSS hará que el navegador calcule de forma sincrónica el estilo y el diseño, lo que es malo cuando necesitas animar a 60 fps.

NO

Animación con diseño desencadenante `left` y `top`.

```
#caja {  
  left: 0;  
  top: 0;  
  transition: left 0.5s, top 0.5s;  
  position: absolute;  
  width: 50px;  
  height: 50px;  
  background-color: gray;  
}  
#caja.active {  
  left: 100px;  
  top: 100px;  
}
```

Tarda **11,7 ms** en renderizar, **9,8 ms** en pintar.

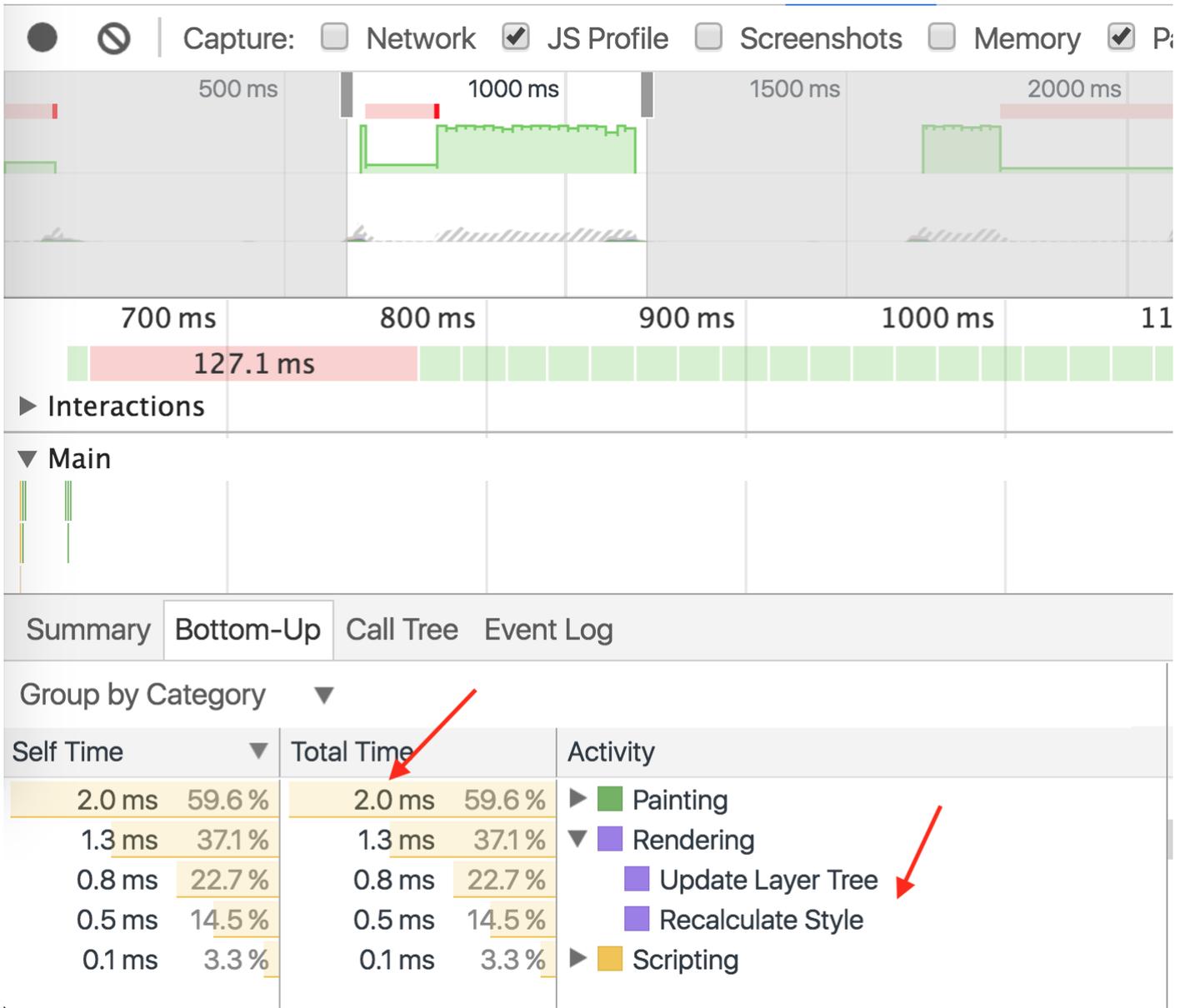


HAZ

Animar con `transform` con la misma animación.

```
#caja {
  left: 0;
  top: 0;
  position: absolute;
  width: 50px;
  height: 50px;
  background-color: gray;
  transition: transform 0.5s;
  transform: translate3d(0, 0, 0);
}
#caja.active {
  transform: translate3d(100px, 100px, 0);
}
```

Misma animación, tarda **1.3 ms** para renderizar, **2.0 ms** para pintar.



Créditos

Muchas gracias a todas las personas de Stack Overflow Documentation que ayudaron a proporcionar este contenido, más cambios pueden ser enviados a web@petercv.com para que el nuevo contenido sea publicado o actualizado.

Traductor al español

[rortegag](#)

A B	Capítulo 20
A.J	Capítulo 4
Aaron	Capítulo 4
abaracedo	Capítulo 4
Abhishek Singh	Capítulo 22
adamboro	Capítulo 1
Aeolingamenfel	Capítulos 27 y 55
Ahmad Alfy	Capítulos 4, 5 y 16
Alohci	Capítulo 15
amflare	Capítulos 13 y 17
Andre Lopes	Capítulo 44
andre mcgruder	Capítulo 54
andreas	Capítulos 15 y 38
Andrew	Capítulos 12, 19 y 53
Andrew Myers	Capítulo 47
Anil	Capítulo 4
animuson	Capítulos 4, 50 y 53
apaul	Capítulos 6 y 27
Araknid	Capítulo 4
Arif	Capítulo 11
Arjan Einbu	Capítulos 4, 7, 8, 15 y 17
Ashwin Ramaswami	Capítulos 1 y 4
Asim K T	Capítulos 5 y 16
AVAVT	Capítulo 50
awe	Capítulo 1
bdkopen	Capítulo 3
Ben Rhys	Capítulo 5
Bipon	Capítulo 40
BiscuitBaker	Capítulo 7
Boris	Capítulo 5
Boysenb3rry	Capítulo 1
brandaemon	Capítulo 17
Brett DeWoody	Capítulos 18, 38 y 39
CalvT	Capítulos 5 y 9
Casey	Capítulo 11
Cassidy Williams	Capítulos 10 y 22
cdm	Capítulos 5 y 8
Charlie H	Capítulos 4 y 28
Chathuranga Jayanath	Capítulos 11, 13 y 23
Chiller	Capítulo 38
Chris	Capítulos 1, 4, 23, 25, 42 y 50
Chris Spittles	Capítulos 8 y 24
Christiaan Maks	Capítulo 28
CocoaBean	Capítulo 5
coderfin	Capítulo 3

cone56	Capítulos 31 y 36
CPHPython	Capítulo 4
csx.cc	Capítulo 1
cuervoo	Capítulo 18
Daniel G. Blázquez	Capítulo 5
Daniel Käfer	Capítulo 6
Daniel Stradowski	Capítulo 5
DarkAjax	Capítulo 17
darrylyeo	Capítulos 2, 13 y 18
Darthstroke	Capítulo 5
Dave Everitt	Capítulo 4
David Fullerton	Capítulo 4
Demeter Dimitri	Capítulo 4
demonofthemist	Capítulo 14
designcise	Capítulos 4, 5 y 18
Devid Farinelli	Capítulos 4 y 6
Devon Bernard	Capítulo 4
Dex Star	Capítulo 27
Diego V	Capítulo 6
Dinidu Hewage	Capítulo 4
dippas	Capítulos 4, 17 y 21
doctorsherlock	Capítulo 10
dodopok	Capítulos 13, 36 y 45
Elegant.Scripting	Capítulo 43
Eliran Malka	Capítulo 6
Emanuele Parisio	Capítulo 6
Evgeny	Capítulo 15
Farzad YZ	Capítulo 6
fcalderan	Capítulo 5
feeela	Capítulos 46 y 55
FelipeAls	Capítulos 1, 5, 10, 11, 14, 16, 24 y 25
Felix A J	Capítulo 15
Felix Edelmann	Capítulo 4
Felix Schütz	Capítulo 4
Forty	Capítulo 4
fracz	Capítulo 4
fzzylogic	Capítulo 16
G	Capítulos 1 y 17
Gabriel R.	Capítulo 1
gandreadis	Capítulo 4
geek1011	Capítulo 21
geeksal	Capítulo 17
Gerardas	Capítulo 1
Gnietschow	Capítulo 10
GoatsWearHats	Capítulo 1
Gofilord	Capítulo 21
Grant Palin	Capítulo 54
H. Pauwelyn	Capítulos 4, 18 y 36
HansCz	Capítulo 4
Harish Gyanani	Capítulo 1
Harry	Capítulos 10, 26, 28, 29, 33, 35 y 44
henry	Capítulo 4
Horst Jahns	Capítulo 5
Hristo	Capítulo 32
Hugo Buff	Capítulo 4
Hynes	Capítulos 4, 5 y 15

insertusernamehere	Capítulo 15
JAtkin	Capítulos 1 y 4
JF	Capítulos 4 y 20
Jacob Gray	Capítulos 4, 5 y 22
James Donnelly	Capítulos 7 y 17
James Taylor	Capítulo 5
jaredsk	Capítulos 10, 36 y 50
JedaiCoder	Capítulo 6
Jef	Capítulo 16
Jeffery Tang	Capítulo 30
jehna1	Capítulo 6
jgh	Capítulo 12
JHS	Capítulo 25
Jmh2013	Capítulos 13, 23 y 43
joejoe31b	Capítulos 4 y 13
JoelBonetR	Capítulo 4
joe_young	Capítulos 1 y 15
John Slegers	Capítulos 4, 5, 6, 13, 17, 18, 28, 52 y 55
Jon Chan	Capítulos 5 y 15
Jonathan Argentiero	Capítulo 6
Jonathan Lam	Capítulos 1, 6, 7, 16 y 22
Jonathan Zúñiga	Capítulo 5
Jose Gomez	Capítulo 1
Just a student	Capítulo 1
Kevin Katzke	Capítulo 23
kingcobra1986	Capítulo 17
Kuhan	Capítulo 18
Kyle Ratliff	Capítulo 6
leo_ap	Capítulo 50
LiLacTac	Capítulo 55
Luka Kerr	Capítulo 29
Luke Taylor	Capítulo 28
Madalina Taina	Capítulos 4, 5, 6, 8, 9, 10, 11, 12, 14, 15, 19, 25, 29, 31, 32, 34, 39, 45 y 49
Marc	Capítulo 20
Marcatectura	Capítulo 21
Marjorie Pickard	Capítulo 2
Mark Perera	Capítulo 4
Marten Koetsier	Capítulos 34 y 41
Matas Vaitkevicius	Capítulos 4 y 13
Mattia Astorino	Capítulo 22
Maximillian Laumeister	Capítulos 5 y 13
Maxouhell	Capítulo 6
Michael Moriarty	Capítulos 5, 15 y 18
Michael_B	Capítulos 4 y 6
Mifeet	Capítulo 6
Mike McCaughan	Capítulo 24
Miles	Capítulos 12 y 51
Miro	Capítulo 18
Mmachinegun	Capítulo 54
mmativ	Capítulo 50
Mod Proxy	Capítulo 6
Mr. Alien	Capítulo 5
Mr. Meeseeks	Capítulo 29
Mr_Green	Capítulo 8
Muthu Kumaran	Capítulo 37
Naeem Shaikh	Capítulo 4

Nate	Capítulo 5
Nathan Arthur	Capítulos 1, 4, 6, 8, 13, 14, 15 y 16
Nemanja Trifunovic	Capítulo 48
Niek Brouwer	Capítulo 23
niyasc	Capítulo 18
Nobal Mohan	Capítulo 10
o.v.	Capítulo 6
Obsidian	Capítulos 37 y 53
Ortomala Lokni	Capítulos 6, 7 y 17
Pat	Capítulo 21
patelarpan	Capítulos 1 y 50
Paul Kozlovitch	Capítulo 6
Paul Sweatte	Capítulo 46
Persijn	Capítulos 4 y 5
Phil	Capítulo 50
pixelbandito	Capítulo 9
Praveen Kumar	Capítulos 4, 6, 13, 15, 26, 28, 50 y 55
Qaz	Capítulo 12
Rahul Nanwani	Capítulo 22
RamenChef	Capítulo 43
rdans	Capítulo 4
RedRiderX	Capítulo 37
rejev	Capítulo 8
Richard Hamilton	Capítulos 4, 5, 15, 18, 20 y 27
Rion Williams	Capítulo 4
rishabh dev	Capítulo 46
rmondasilva	Capítulos 15 y 20
Robotnicka	Capítulo 20
Rocket Risa	Capítulo 1
Sandeep Tuniki	Capítulo 6
Saroj Sasmal	Capítulo 1
ScientiaEtVeritas	Capítulos 1, 4, 6, 7, 10, 11, 18, 20, 21, 23, 26, 31, 33, 44 y 53
Sebastian Zartner	Capítulo 40
SeinopSys	Capítulos 18, 23, 36 y 54
Sergey Denisov	Capítulos 5 y 26
Shaggy	Capítulos 5, 21 y 53
Siavas	Capítulo 6
Someone	Capítulo 6
Sourav Ghosh	Capítulos 5 y 22
Squazz	Capítulos 16 y 31
srikarg	Capítulo 13
StefanBob	Capítulo 9
Stewartside	Capítulos 4, 5, 6, 18, 20 y 21
Stratboy	Capítulo 5
sudo bangbang	Capítulo 4
Sumner Evans	Capítulo 4
Sun Qingyao	Capítulo 8
Sunnyok	Capítulos 4, 6 y 8
Sverri M. Olsen	Capítulo 1
takeradi	Capítulo 16
Taylor	Capítulo 6
Ted Goas	Capítulos 12, 15, 34 y 43
Teo Dragovic	Capítulos 1 y 13
ThatWeirdo	Capítulo 4
TheGenie OfTruth	Capítulo 36
Theodore K.	Capítulo 22

think123	Capítulo 5
Timothy Miller	Capítulo 55
Toby	Capítulos 15 y 20
Todd	Capítulo 1
ToniB	Capítulo 15
Tot Zam	Capítulo 8
Trevor Clarke	Capítulos 5, 8, 10 y 15
TrungDQ	Capítulo 56
TylerH	Capítulos 1, 4, 5, 36 y 53
Ulrich Schwarz	Capítulo 43
user007	Capítulo 18
user2622348	Capítulo 20
vishak	Capítulo 14
vkopio	Capítulo 7
Vlusion	Capítulo 15
Volker E.	Capítulo 15
web	Capítulos 6, 26, 28, 29 y 44
Will DiFruscio	Capítulo 9
Wolfgang	Capítulo 18
X	Capítulo 18
Xinyang Li	Capítulo 1
xpy	Capítulo 4
Yury Fedorov	Capítulo 4
Zac	Capítulos 5 and 12
Zaffy	Capítulo 4
Zakaria Acharki	Capítulo 20
Zaz	Capítulo 4
Ze Rubeus	Capítulo 4
zeel	Capítulo 6
zer00ne	Capítulo 20
Zeta	Capítulo 5
Zze	Capítulo 5