



EBook Gratis

APRENDIZAJE CSS

Free unaffiliated eBook created from
Stack Overflow contributors.

#css

Tabla de contenido

Acerca de.....	1
Capítulo 1: Empezando con CSS	2
Observaciones.....	2
Versiones.....	2
Examples.....	2
Hoja de estilo externa.....	2
Ejemplo.....	2
Estilos internos.....	4
Estilos en linea.....	4
CSS @import rule (uno de CSS at-rule).....	5
Cómo usar @import.....	5
Cambiando CSS con JavaScript.....	5
JavaScript puro.....	5
jQuery.....	6
Ver también.....	6
Listas de estilo con CSS.....	6
Capítulo 2: Actuación.....	8
Examples.....	8
Utilice la transformación y la opacidad para evitar el diseño de disparo.....	8
NO HACER.....	8
HACER.....	9
Capítulo 3: Ajuste y colocación de objetos.....	10
Observaciones.....	10
Examples.....	10
ajuste de objeto.....	10
Capítulo 4: Animaciones.....	13
Sintaxis.....	13
Parámetros.....	13
Examples.....	13

Animaciones con la propiedad de transición.....	13
Ejemplo.....	13
Compatibilidad entre navegadores.....	14
Incrementando el rendimiento de la animación usando el atributo `will-change`	15
Animaciones con fotogramas clave.....	15
Ejemplo básico.....	15
Compatibilidad entre navegadores.....	17
Ejemplos de sintaxis.....	17
Capítulo 5: Antecedentes.....	19
Introducción.....	19
Sintaxis.....	19
Observaciones.....	19
Examples.....	19
Color de fondo.....	19
Nombres de colores.....	20
Códigos de color hexadecimales.....	20
RGB / RGBa.....	20
HSL / HSLa.....	21
Interacción con imagen de fondo.....	21
Imagen de fondo.....	22
Gradientes de fondo.....	23
gradiente lineal().....	23
gradiente radial ().....	24
Gradientes de repetición.....	24
Taquigrafía de fondo.....	25
Sintaxis.....	26
Ejemplos.....	26
Posición de fondo.....	26
Propiedades de posición de fondo a largo plazo.....	27
Adjunto de fondo.....	27
Ejemplos.....	28

fondo adjunto: desplazamiento.....	28
fondo adjunto: fijo.....	28
Fondo adjunto: local.....	28
Repetición de fondo.....	28
Color de fondo con opacidad.....	29
Imagen de fondo múltiple.....	30
La propiedad de origen-fondo.....	30
Clip de fondo.....	32
Tamaño de fondo.....	33
Visión general	33
Manteniendo la relación de aspecto.	35
Plantas de huevo para contain y cover.....	35
contain.....	36
cover.....	36
Demostración con código actual.....	37
Propiedad de modo de mezcla de fondo.....	38
Capítulo 6: Cascada y especificidad	39
Observaciones.....	39
Examples.....	39
En cascada.....	39
CSS orden de carga	39
¿Cómo se resuelven los conflictos?	39
Ejemplo 1 - Reglas de especificidad.....	39
Ejemplo 2 - Reglas de cascada con selectores idénticos.....	40
Ejemplo 3 - Reglas en cascada después de las reglas de especificidad.....	40
Una nota final.....	41
La! Declaración importante.....	41
Cálculo de la especificidad del selector.....	41
Ejemplo 1: Especificidad de varias secuencias de selección.....	42
Ejemplo 2: Cómo se usa la especificidad por el navegador.....	42
Ejemplo 3: Cómo manipular la especificidad.....	43
Declaraciones de estilo !important y en línea.....	44

Una nota final.....	44
Ejemplo de especificidad más compleja.....	45
Capítulo 7: Centrado.....	47
Examples.....	47
Usando la transformación CSS.....	47
COMPATIBILIDAD DE NAVEGADOR CRUZADO.....	47
MÁS INFORMACIÓN.....	48
Usando flexbox.....	48
Posición de uso: absoluta.....	49
Técnica del elemento fantasma (hack de Micha Czernow).....	50
Usando text-align.....	50
Centrado en relación a otro elemento.....	51
Alinear verticalmente cualquier cosa con 3 líneas de código.....	52
Alinear verticalmente una imagen dentro de div.....	52
Centrado horizontal y vertical utilizando el diseño de la mesa.....	53
Utilizando calc ().....	54
Alinear verticalmente elementos dinámicos de altura.....	54
Usando la línea de altura.....	55
Centrado vertical y horizontalmente sin preocuparse por la altura o el ancho.....	55
El contenedor exterior.....	55
El contenedor interior.....	55
El cuadro de contenido.....	55
Manifestación.....	56
Centrado con tamaño fijo.....	56
Centrado horizontal con un solo ancho fijo.....	57
Centrado vertical con altura fija.....	57
Usando el margen: 0 auto;.....	58
Capítulo 8: Centrado vertical.....	60
Observaciones.....	60
Examples.....	60
Centrado con display: mesa.....	60
Centrado con transformar.....	60

Centrado con Flexbox.....	61
Centrado de texto con altura de línea.....	61
Centrado con Posición: absoluta.....	62
Centrado con pseudo elemento.....	63
Capítulo 9: Colores.....	64
Sintaxis.....	64
Examples.....	64
Palabras clave de color.....	64
Palabras clave de color.....	64
Valor hexadecimal.....	71
Fondo.....	71
Sintaxis.....	72
rgb () Notación.....	72
Sintaxis.....	73
hsl () Notación.....	73
Sintaxis.....	73
Notas.....	74
color actual.....	74
Utilizar en el mismo elemento.....	74
Heredado del elemento padre.....	74
rgba () Notación.....	75
Sintaxis.....	76
hsla () Notación.....	76
Sintaxis.....	76
Capítulo 10: Columnas.....	77
Sintaxis.....	77
Examples.....	77
Ejemplo simple (conteo de columnas).....	77
Ancho de columna.....	78
Capítulo 11: Columnas multiples.....	80
Introducción.....	80

Observaciones.....	80
Examples.....	80
Ejemplo basico.....	80
Crear múltiples columnas.....	81
Capítulo 12: Comentarios.....	82
Sintaxis.....	82
Observaciones.....	82
Examples.....	82
Linea sola.....	82
Línea múltiple.....	82
Capítulo 13: Consultas de características.....	83
Sintaxis.....	83
Parámetros.....	83
Observaciones.....	83
Examples.....	83
Uso básico de los soportes.....	83
Encadenamiento de detecciones de características.....	83
Capítulo 14: Contadores.....	85
Sintaxis.....	85
Parámetros.....	85
Observaciones.....	85
Examples.....	86
Aplicando un estilo de números romanos a la salida del contador.....	86
CSS.....	86
HTML.....	86
Numera cada ítem usando el Contador CSS.....	86
CSS.....	86
HTML.....	87
Implementación de numeración multinivel utilizando contadores CSS.....	87
CSS.....	87
HTML.....	87

Capítulo 15: Contexto de apilamiento	89
Examples	89
Contexto de apilamiento	89
Capítulo 16: Contextos de formato de bloque	93
Observaciones	93
Examples	93
Usando la propiedad de desbordamiento con un valor diferente a visible	93
Capítulo 17: Contornos	95
Sintaxis	95
Parámetros	95
Observaciones	95
Examples	96
Visión general	96
esquema de estilo	96
Capítulo 18: Control de diseño	98
Sintaxis	98
Parámetros	98
Examples	99
La propiedad de visualización	99
En línea	99
Bloquear	99
Bloque en linea	99
ninguna	101
Para obtener la estructura de la tabla de edad utilizando div	102
Capítulo 19: Cuadrícula	103
Introducción	103
Observaciones	103
Examples	103
Ejemplo básico	103
Capítulo 20: Diseño de caja flexible (Flexbox)	105
Introducción	105

Sintaxis.....	105
Observaciones.....	105
Prefijos de Vendor.....	105
Recursos.....	105
Examples.....	106
Pie de página variable pegajoso.....	106
Disposición del Santo Grial usando Flexbox.....	107
Botones perfectamente alineados dentro de tarjetas con flexbox.....	108
Centrado dinámico vertical y horizontal (alinear elementos, justificar contenido).....	110
Ejemplo simple (centrando un solo elemento).....	110
HTML.....	110
CSS.....	110
Razonamiento.....	110
Ejemplos de propiedad individual.....	111
Ejemplo: justify justify-content: center en un flexbox horizontal.....	111
Ejemplo: justify justify-content: center en un flexbox vertical.....	112
Ejemplo: align-content: center en un flexbox horizontal.....	113
Ejemplo: align-content: center en un flexbox vertical.....	114
Ejemplo: combinación para centrar ambos en flexbox horizontal.....	115
Ejemplo: combinación para centrar ambos en flexbox vertical.....	116
Misma altura en contenedores anidados.....	117
Ajustar los elementos de forma óptima a su contenedor.....	118
Capítulo 21: Disposición de bloques en línea.....	120
Examples.....	120
Barra de navegación justificada.....	120
HTML.....	120
CSS.....	120
Notas.....	120
Capítulo 22: El modelo de caja.....	122
Sintaxis.....	122
Parámetros.....	122

Observaciones.....	122
Acerca de la caja de relleno.....	122
Examples.....	122
¿Qué es el modelo de caja?.....	122
Los bordes.....	122
Ejemplo.....	123
tamaño de caja.....	125
Capítulo 23: Estilo del cursor.....	127
Sintaxis.....	127
Examples.....	127
Cambiando el tipo de cursor.....	127
eventos punteros.....	128
color caret.....	128
Capítulo 24: Estructura y formato de una regla CSS.....	129
Observaciones.....	129
Bueno.....	129
Malo.....	129
Un trazador de líneas.....	129
Examples.....	129
Reglas, selectores y bloques de declaración.....	129
Listas de propiedades.....	129
Selectores multiples.....	130
Capítulo 25: Flotadores.....	131
Sintaxis.....	131
Observaciones.....	131
Examples.....	131
Flotar una imagen dentro del texto.....	131
Diseño simple de dos columnas de ancho fijo.....	132
Diseño simple de tres columnas de ancho fijo.....	133
Disposición perezosa / codiciosa de dos columnas.....	134
claro propiedad.....	135

Clearfix.....	136
Clearfix (con el colapso del margen superior de los flotadores contenidos todavía ocurren.....	136
Clearfix también evita el colapso del margen superior de los flotadores contenidos.....	136
Clearfix con soporte de navegadores obsoletos IE6 y IE7.....	137
DIV en línea usando flotador.....	137
Uso de la propiedad de desbordamiento para eliminar flotadores.....	139
Capítulo 26: Formas de un solo elemento.....	140
Examples.....	140
Cuadrado.....	140
triangulos.....	140
Ráfagas.....	144
Círculos y elipses.....	145
Círculo.....	145
Elipse.....	146
Trapecio.....	146
Cubo.....	147
Pirámide.....	148
Capítulo 27: Formas para flotadores.....	150
Sintaxis.....	150
Parámetros.....	150
Observaciones.....	150
Examples.....	150
Forma exterior con forma básica - círculo ().....	150
Margen de forma.....	152
Capítulo 28: Fragmentación.....	154
Sintaxis.....	154
Parámetros.....	154
Observaciones.....	154
Examples.....	154
Impresión de medios de página.....	154
Capítulo 29: Frontera.....	156

Sintaxis.....	156
Observaciones.....	156
Examples.....	158
radio del borde.....	158
estilo de borde.....	159
frontera (taquigrafía).....	160
imagen de borde.....	160
borde- [izquierda derecha arriba abajo].....	161
colapso de la frontera.....	161
Múltiples fronteras.....	161
Creando un borde multicolor usando una imagen de borde.....	163
CSS.....	163
HTML.....	163
Capítulo 30: Funciones.....	166
Sintaxis.....	166
Observaciones.....	166
Examples.....	166
función calc ().....	166
función attr ().....	167
función de gradiente lineal ().....	167
función de gradiente radial ().....	167
función var ().....	167
Capítulo 31: Hacks de Internet Explorer.....	169
Observaciones.....	169
Examples.....	169
Modo de alto contraste en Internet Explorer 10 y superior.....	169
Ejemplos.....	169
Más información:.....	169
Sólo Internet Explorer 6 e Internet Explorer 7.....	170
Sólo Internet Explorer 8.....	170
Adición de soporte de bloque en línea a IE6 y IE7.....	170
Capítulo 32: Herencia.....	171

Sintaxis.....	171
Examples.....	171
Herencia automatica.....	171
Herencia forzada.....	171
Capítulo 33: Imagen de CSS Sprites.....	173
Sintaxis.....	173
Observaciones.....	173
Examples.....	173
Una implementación básica.....	173
Capítulo 34: Listar estilos.....	175
Sintaxis.....	175
Parámetros.....	175
Observaciones.....	175
Examples.....	175
Tipo de viñeta o numeración.....	175
Posición de la bala.....	176
Eliminando Balas / Números.....	176
Capítulo 35: Márgenes.....	178
Sintaxis.....	178
Parámetros.....	178
Observaciones.....	178
Examples.....	178
Aplicar margen en un lado dado.....	178
Propiedades específicas de la dirección.....	178
Especificando la dirección usando la propiedad abreviada.....	179
Colapso de margen.....	180
Centrar horizontalmente los elementos en una página utilizando el margen.....	182
Simplificación de propiedades de margen.....	182
Márgenes negativos.....	183
Ejemplo 1:.....	183
Capítulo 36: Mesas.....	185

Sintaxis.....	185
Observaciones.....	185
Examples.....	185
diseño de la mesa.....	185
colapso de la frontera.....	186
espaciado de la frontera.....	186
celdas vacías.....	187
lado del título.....	187
Capítulo 37: Modelo de objetos CSS (CSSOM).....	189
Observaciones.....	189
Examples.....	189
Introducción.....	189
Agregando una regla de imagen de fondo a través del CSSOM.....	189
Capítulo 38: Normalizar los estilos del navegador.....	191
Introducción.....	191
Observaciones.....	191
Examples.....	191
normalize.css.....	191
Qué hace.....	191
Diferencia a reset.css.....	192
Enfoques y ejemplos.....	192
Capítulo 39: Opacidad.....	194
Sintaxis.....	194
Observaciones.....	194
Examples.....	194
Propiedad de opacidad.....	194
Compatibilidad de IE para la 'opacidad'.....	194
Capítulo 40: Patrones de diseño CSS.....	196
Introducción.....	196
Observaciones.....	196
Examples.....	196
BEM.....	196

Ejemplo de código	197
Capítulo 41: Posicionamiento	198
Sintaxis	198
Parámetros	198
Observaciones	198
Examples	198
Posición fija	198
Elementos superpuestos con índice z	199
Ejemplo	199
HTML	199
CSS	199
Sintaxis	200
Observaciones	200
Posición relativa	201
Posición absoluta	201
Posicionamiento estático	202
Capítulo 42: Preguntas de los medios	203
Sintaxis	203
Parámetros	203
Observaciones	204
Examples	205
Ejemplo básico	205
Usar en la etiqueta de enlace	205
tipo de medio	206
Uso de consultas de medios para identificar diferentes tamaños de pantalla	207
Ancho vs Viewport	207
Consultas de medios para pantallas de retina y no retina	208
Terminología y estructura	209
Estructura general de una consulta de medios	209
Una consulta de medios que contiene un tipo de medio	209
Una consulta de medios que contiene un tipo de medio y una característica de medios	209

Una consulta de medios que contiene una característica de medios (y un tipo de medios impl.) 209

Consultas de medios e IE8.....	210
Una solución basada en Javascript.....	210
La alternativa.....	210
Capítulo 43: Propiedad de filtro.....	211
Sintaxis.....	211
Parámetros.....	211
Observaciones.....	212
Examples.....	212
Sombra paralela (usa box-shadow si es posible).....	212
Valores de filtro múltiples.....	212
Hue Rotate.....	213
Invertir color.....	214
Difuminar.....	214
Capítulo 44: Propiedades personalizadas (variables).....	216
Introducción.....	216
Sintaxis.....	216
Observaciones.....	216
APOYO / COMPATIBILIDAD DEL NAVEGADOR.....	216
Examples.....	217
Color variable.....	217
Dimensiones variables.....	217
Cascada variable.....	217
Válido / Inválido.....	218
Con consultas de medios.....	219
Capítulo 45: Pseudoelementos.....	222
Introducción.....	222
Sintaxis.....	222
Parámetros.....	222
Observaciones.....	223
Examples.....	223

Pseudoelementos.....	223
Pseudo-elementos en las listas.....	223
Capítulo 46: Rebosar.....	225
Sintaxis.....	225
Parámetros.....	225
Observaciones.....	225
Examples.....	225
desbordamiento: desplazamiento.....	226
envoltura de desbordamiento.....	226
desbordamiento: visible.....	227
Contexto de formato de bloque creado con desbordamiento.....	228
overflow-x y overflow-y.....	229
Capítulo 47: Recorte y enmascaramiento.....	231
Sintaxis.....	231
Parámetros.....	231
Observaciones.....	232
Mascarillas.....	232
Recorrido del clip:.....	233
Examples.....	233
Recorte (polígono).....	233
CSS:.....	233
HTML:.....	233
Recorte (círculo).....	234
CSS:.....	234
HTML:.....	234
Recorte y enmascaramiento: Resumen y diferencia.....	235
Recorte.....	235
Enmascaramiento.....	235
Máscara simple que desvanece una imagen de sólido a transparente.....	236
CSS.....	236
HTML.....	236

Usando máscaras para cortar un agujero en el medio de una imagen.....	237
CSS.....	237
HTML.....	238
Usando máscaras para crear imágenes con formas irregulares.....	238
CSS.....	238
HTML.....	239
Capítulo 48: Relleno.....	240
Sintaxis.....	240
Observaciones.....	240
Examples.....	240
Relleno en un lado dado.....	240
Relleno de taquigrafía.....	241
Capítulo 49: Selectores.....	243
Introducción.....	243
Sintaxis.....	243
Observaciones.....	243
Examples.....	243
Selectores de atributos.....	243
Visión general.....	243
Detalles.....	244
[attribute].....	245
[attribute="value"].....	245
[attribute*="value"].....	245
[attribute~= "value"].....	245
[attribute^="value"].....	246
[attribute\$="value"].....	246
[attribute = "value"].....	246
[attribute="value" i].....	247
Especificidad de los selectores de atributos.....	247
0-1-0.....	247
Combinadores.....	247

Visión general	247
Combinador Descendiente: selector selector	248
Combinador infantil: selector > selector	248
Combinador de hermanos adyacente: selector + selector	249
Combinador general de hermanos: selector ~ selector	249
Selectores de nombre de clase	250
Selectores de ID	251
Pseudo-clases	251
Sintaxis	251
Lista de pseudo-clases:	251
Selectores basicos	254
Cómo diseñar una entrada de rango	255
Global booleano con casilla de verificación: marcado y ~ (combinador general de hermanos)	255
Añadir booleano como casilla de verificación	256
Cambiar el valor del booleano	256
Accediendo al valor booleano con CSS	256
En acción	257
CSS3: ejemplo de selector de rango	257
Pseudo clase infantil	257
Seleccione el elemento utilizando su ID sin la alta especificidad del selector de ID	258
A. El ejemplo: no pseudo-clase & B.: focus-within CSS pseudo-class	258
El ejemplo de selector de pseudo-clase de hijo único	260
El: selector de último tipo	261
Capítulo 50: sombra de la caja	262
Sintaxis	262
Parámetros	262
Observaciones	262
Examples	262
sombra paralela	262
sombra interior	263
Sombra de la parte inferior sólo con un pseudo-elemento	263

múltiples sombras.....	264
Capítulo 51: Soporte de navegador y prefijos.....	266
Parámetros.....	266
Observaciones.....	266
Examples.....	266
Transiciones.....	267
Transformar.....	267
Capítulo 52: Tipografía.....	268
Sintaxis.....	268
Parámetros.....	268
Observaciones.....	269
Examples.....	269
Tamaño de fuente.....	269
La taquigrafía de la fuente.....	269
Pilas de fuentes.....	270
Espaciado de letras.....	271
Transformación de texto.....	271
Guion de texto.....	272
Decoracion de texto.....	272
Desbordamiento de texto.....	272
Espaciado de palabras.....	273
Dirección del texto.....	274
Variante de fuente.....	274
Citas.....	275
Sombra de texto.....	275
Sombra sin radio borroso.....	275
Sombra con radio borroso.....	275
Sombras multiples.....	276
Capítulo 53: Transformaciones 2D.....	277
Sintaxis.....	277
Parámetros.....	277
Observaciones.....	278

Sistema coordinador 2D	278
Soporte de navegador y prefijos.....	279
Ejemplo de transformación prefijada:.....	279
Examples.....	279
Girar.....	279
Escala.....	279
Traducir.....	280
Sesgar.....	280
Transformaciones multiples.....	281
Transformar origen.....	283
Capítulo 54: Transformaciones 3D.....	285
Observaciones.....	285
Sistema coordinado.....	285
Examples.....	285
Cubo 3D.....	286
visibilidad de la cara posterior.....	287
Brújula con forma de puntero o aguja mediante transformaciones 3D.....	288
CSS.....	288
HTML.....	288
Efecto de texto 3D con sombra.....	289
Capítulo 55: Transiciones.....	292
Sintaxis.....	292
Parámetros.....	292
Observaciones.....	292
Examples.....	292
Taquigrafía de transición.....	293
Transición (a mano).....	293
CSS.....	293
HTML.....	293
cúbico-bezier.....	294
Capítulo 56: Unidades de longitud.....	296

Introducción.....	296
Sintaxis.....	296
Parámetros.....	296
Observaciones.....	297
Examples.....	297
Tamaño de letra con rem.....	297
Creando elementos escalables usando rems y ems.....	298
vh y vw.....	299
vmin y vmax.....	299
usando porcentaje%.....	299
Creditos.....	301

Acerca de

You can share this PDF with anyone you feel could benefit from it, download the latest version from: [css](#)

It is an unofficial and free CSS ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official CSS.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con CSS

Observaciones

Los estilos se pueden crear de varias maneras, lo que permite diversos grados de reutilización y alcance cuando se especifican en un documento HTML de origen. *Las hojas de estilo externas* se pueden reutilizar en documentos HTML. *Las hojas de estilo incrustadas* se aplican a todo el documento en el que se especifican. *Los estilos en línea* se aplican solo al elemento HTML individual en el que se especifican.

Versiones

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

Examples

Hoja de estilo externa

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

El atributo `rel` de la etiqueta `<link>` se debe establecer en `"stylesheet"`, y el atributo `href` en la ruta relativa o absoluta de la hoja de estilo. Si bien el uso de rutas de URL relativas generalmente se considera una buena práctica, también se pueden usar rutas absolutas. En HTML5 [se puede omitir el atributo de `type`](#).

Se recomienda que la etiqueta `<link>` se coloque en la etiqueta `<head>` del archivo HTML para que los estilos se carguen antes de los elementos que diseñan. 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>Hello world!</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;
}
```

Asegúrese de incluir la ruta correcta a su archivo CSS en el href. Si el archivo CSS está en la misma carpeta que su archivo HTML, entonces no se requiere una ruta (como en el ejemplo anterior), pero si está guardado en una carpeta, especifíquelo así `href="filename/style.css"`.

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

Las hojas de estilo externas se consideran la mejor manera de manejar tu CSS. Hay una razón muy simple para esto: cuando administra un sitio de, por ejemplo, 100 páginas, todo controlado por una sola hoja de estilo, y desea cambiar los colores de sus enlaces de azul a verde, es mucho más fácil hacer el cambio. en su archivo CSS y deje que los cambios "caigan en cascada" en las 100 páginas de lo que es ir a 100 páginas separadas y hacer el mismo cambio 100 veces.

Nuevamente, si desea cambiar completamente el aspecto de su sitio web, solo necesita actualizar este archivo.

Puede cargar tantos archivos CSS en su página HTML como sea necesario.

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

Las reglas de 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 los párrafos con la clase 'verde' se escribirán en verde claro, pero puede anular esto con otro archivo .css simplemente incluyéndolo *después* de main.css. Puede tener override.css con el siguiente código, siga main.css, por ejemplo:

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

Ahora todos los párrafos con la clase "verde" se escribirán en verde más oscuro en lugar de verde

claro.

Se aplican otros principios, como la regla '! Importante', la especificidad y la herencia.

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

Estilos internos

CSS encerrado en etiquetas `<style></style>` dentro de un documento HTML funciona como una hoja de estilo externa, excepto que vive en el documento HTML que diseña en lugar de en un archivo separado, y por lo tanto solo se puede aplicar al documento en el que está vive. Tenga en cuenta que este elemento *debe* estar dentro del elemento `<head>` para la validación de HTML (aunque funcionará en todos los navegadores actuales si se coloca en el `body`).

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

Estilos en linea

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

Los estilos en línea anulan cualquier CSS en una etiqueta `<style>` o en una hoja de estilo externa. Si bien esto puede ser útil en algunas circunstancias, este hecho a menudo reduce la capacidad de mantenimiento de un proyecto.

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

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

Los estilos en línea son generalmente la forma más segura de garantizar la compatibilidad de la

representación en varios clientes, programas y dispositivos de correo electrónico, pero pueden requerir mucho tiempo de escritura y ser un poco difíciles de administrar.

CSS @import rule (uno de CSS at-rule)

El @import CSS at-rule se utiliza para importar reglas de estilo de otras hojas de estilo. Estas reglas deben preceder a todos los otros tipos de reglas, excepto las reglas de @charset; Como no es una declaración anidada, @import no se puede usar dentro de reglas condicionales de grupo. [@import](#) .

Cómo usar @import

Puede usar la regla @import de las siguientes maneras:

A. Con etiqueta de estilo interno

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

B. Con hoja de estilo externa

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

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

Importar CSS externo también es posible. Un caso de uso común son los archivos de fuentes.

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

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

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

Cambiando CSS con JavaScript

JavaScript puro

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

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

Tenga en cuenta que las propiedades de estilo se nombran en un estilo de caja de camello inferior. En el ejemplo, ve que la `font-family` propiedad css se convierte en `fontFamily` en javascript.

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

jQuery

Modificar las propiedades CSS con jQuery es aún más simple.

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

Si necesita cambiar más de una regla de estilo:

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

jQuery incluye dos formas de cambiar las reglas de css que tienen guiones (es decir `font-size`). Puede ponerlos entre comillas o en camello el nombre de la regla de estilo.

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

Ver también

- [Documentación de JavaScript - Leer y cambiar el estilo CSS](#) .
- [Documentación jQuery - Manipulación CSS](#)

Listas de estilo con CSS

Hay tres propiedades diferentes para el estilo de elementos de `list-style-type` : `list-style-type` `list-style-image` `list-style-position` , `list-style-image` `list-style-position` , que se deben declarar en ese orden. Los valores predeterminados son `disco`, `fuera` y `ninguno`, respectivamente. Cada propiedad se puede declarar por separado, o usando la propiedad abreviada de `list-style` .

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

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

- `Dto`
- `circulo`

- cuadrado
- decimal
- bajo romano
- alto romano
- ninguna

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

Para usar puntos con viñetas cuadradas para cada elemento de lista, por ejemplo, usaría el siguiente par de valor de propiedad:

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

La propiedad `list-style-image` determina si el ícono de lista de elementos está configurado con una imagen y acepta un valor de `none` o una URL que apunta a una imagen.

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

La propiedad de `list-style-position` define dónde colocar el marcador de elemento de lista y acepta uno de dos valores: "dentro" o "fuera".

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

Lea Empezando con CSS en línea: <https://riptutorial.com/es/css/topic/293/empezando-con-css>

Capítulo 2: Actuación

Examples

Utilice la transformación y la opacidad para evitar el diseño de disparo

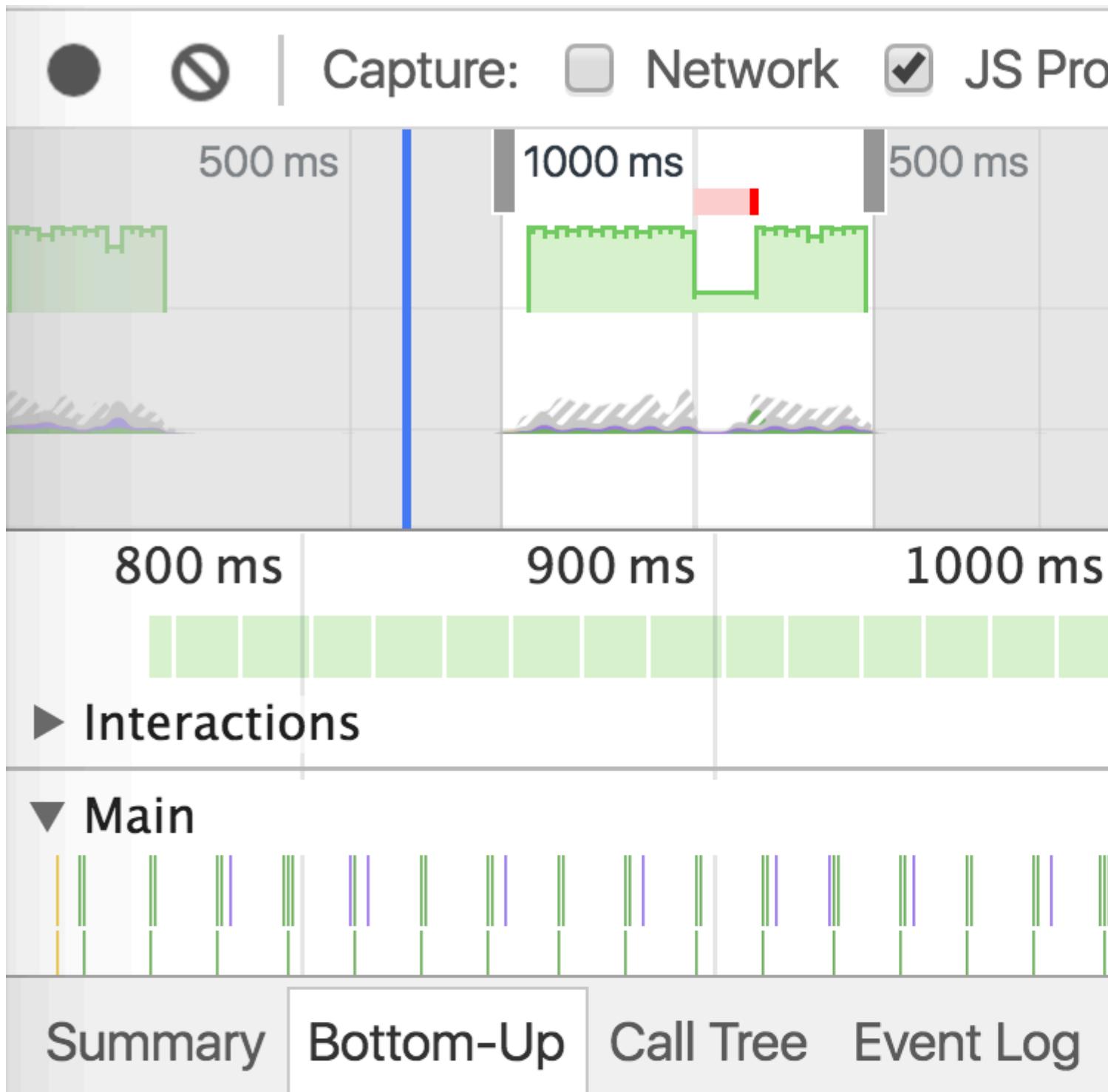
Cambiar algún atributo de CSS activará el navegador para que calcule el estilo y el diseño de forma sincrónica, lo cual es algo malo cuando necesitas animar a 60 fps.

NO HACER

Animar con diseño de gatillo `left` y `top`.

```
#box {  
    left: 0;  
    top: 0;  
    transition: left 0.5s, top 0.5s;  
    position: absolute;  
    width: 50px;  
    height: 50px;  
    background-color: gray;  
}  
  
.box.active {  
    left: 100px;  
    top: 100px;  
}
```

La demo tomó **11.7ms** para renderizar, **9.8ms** para pintar.



Group by Category ▼

Self Time	Total Time	Activ
11.7 ms 54.2 %	11.7 ms 54.2 %	▼
4.2 ms 19.5 %	4.2 ms 19.5 %	
3.9 ms 18.2 %	3.9 ms 18.2 %	
https://riptutorial.com/es/home 1.8 ms 8.3 %	1.8 ms 8.3 %	9

Capítulo 3: Ajuste y colocación de objetos

Observaciones

Internet Explorer no admite las propiedades `object-fit object-position`.

Examples

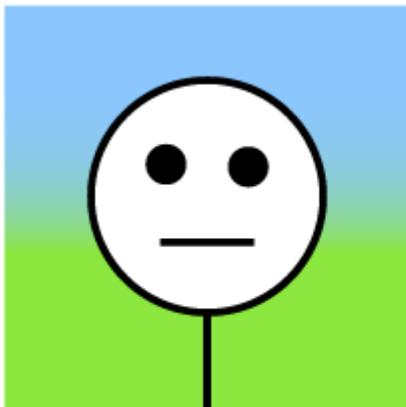
ajuste de objeto

La propiedad de **ajuste de objeto** definirá cómo un elemento encarájará en un cuadro con una altura y anchura establecidas. Generalmente aplicado a una imagen o video, Object-fit acepta los siguientes cinco valores:

LLENAR

```
object-fit:fill;
```

original image



object-fit: fill;

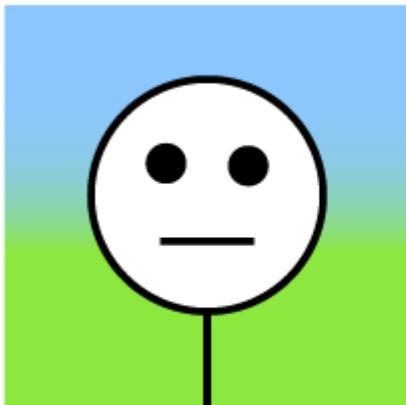


Relleno extiende la imagen para que se ajuste al cuadro de contenido sin importar la relación de aspecto original de la imagen.

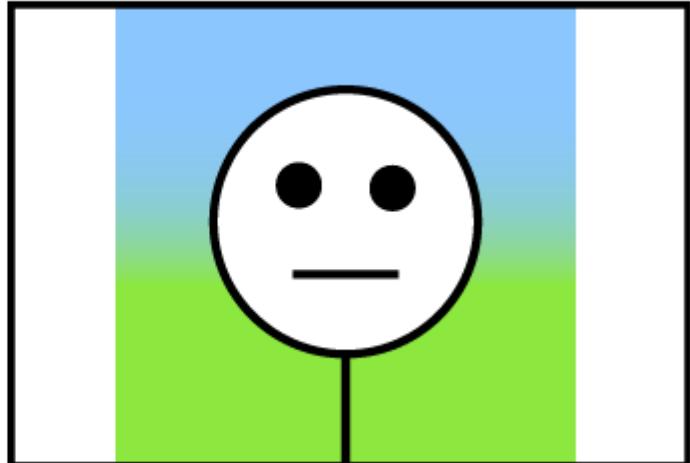
CONTIENE

```
object-fit:contain;
```

original image



object-fit: contain;

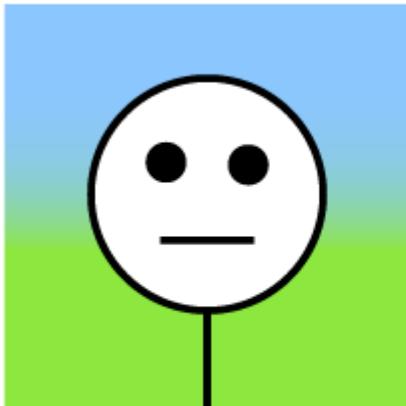


Contener ajusta la imagen a la altura o al ancho del cuadro, manteniendo la relación de aspecto de la imagen.

CUBRIR

object-fit: cover;

original image



object-fit: cover;

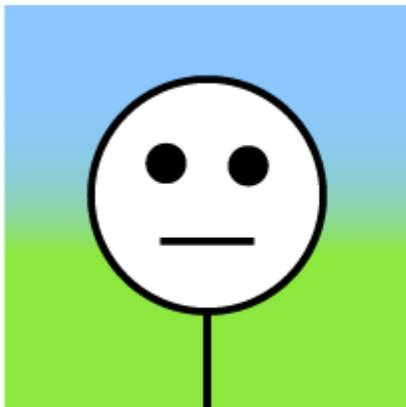


La cubierta llena todo el cuadro con la imagen. La relación de aspecto de la imagen se conserva, pero la imagen se recorta a las dimensiones del cuadro.

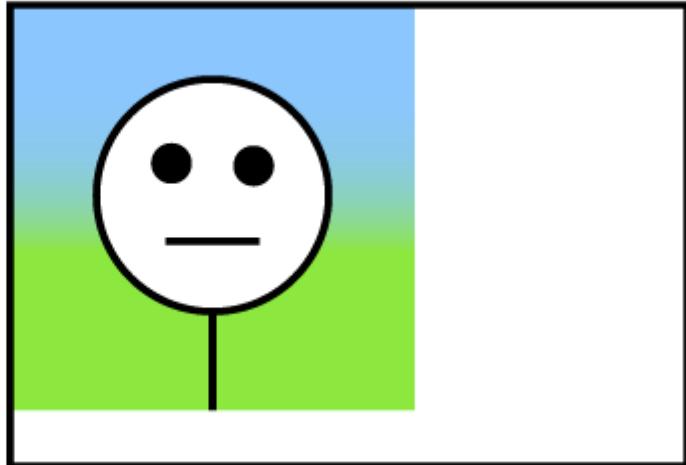
NINGUNA

object-fit: none;

original image



object-fit: none;



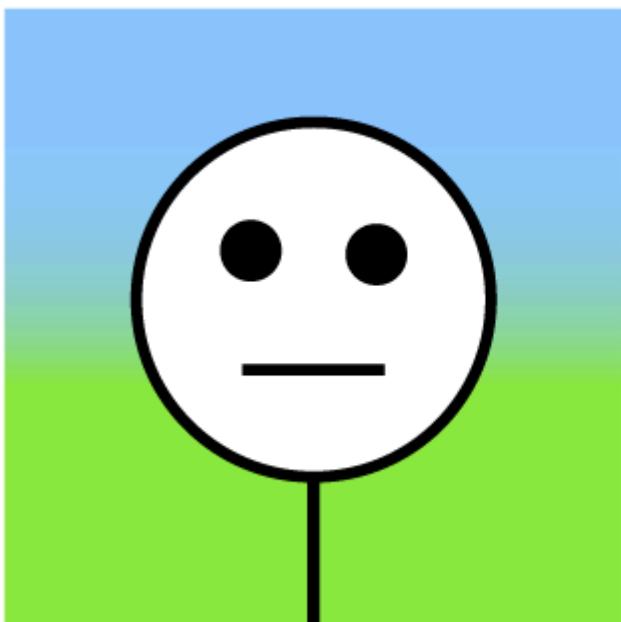
Ninguno ignora el tamaño de la caja y no se redimensiona.

ESCALA ABAJO

object-fit: scale-down;

Reducir o reducir el tamaño del objeto como `none` o como `contain`. Muestra cualquier opción que resulte en un tamaño de imagen más pequeño.

original image



object-fit: scale-down;



Lea Ajuste y colocación de objetos en línea: <https://riptutorial.com/es/css/topic/5520/ajuste-y-colocacion-de-objetos>

Capítulo 4: Animaciones

Sintaxis

- `transition: <property> <duration> <timing-function> <delay>;`
- `@keyframes <identifier>`
- `[[from | to | <percentage>] [, from | to | <percentage>]* block]*`

Parámetros

Transición	
Parámetro	Detalles
propiedad	Ya sea la propiedad CSS para la transición, o <code>all</code> , que especifica todas las propiedades de transición.
duración	Tiempo de transición, ya sea en segundos o milisegundos.
función de tiempo	Especifica una función para definir cómo se calculan los valores intermedios de las propiedades. Los valores comunes son la <code>ease</code> , los <code>linear</code> y <code>step-end</code> . Echa un vistazo a la función de aceleración cheat-sheet para más.
retrasar	Cantidad de tiempo, en segundos o milisegundos, para esperar antes de reproducir la animación.
@keyframes	
<code>[desde a <percentage>]</code>	Puede especificar un tiempo establecido con un valor de porcentaje, o dos valores de porcentaje, es decir, <code>10%, 20%</code> , durante un período de tiempo en el que se establecen los atributos establecidos del fotograma clave.
block	Cualquier cantidad de atributos CSS para el fotograma clave.

Examples

Animaciones con la propiedad de transición.

Útil para animaciones simples, la propiedad de `transition` CSS permite que las propiedades CSS basadas en números se animen entre estados.

Ejemplo

```
.Example{  
    height: 100px;  
    background: #fff;  
}  
  
.Example:hover{  
    height: 120px;  
    background: #ff0000;  
}
```

[Ver resultado](#)

De forma predeterminada, al pasar el cursor sobre un elemento con la clase `.Example`, la altura del elemento saltará a `120px` y el color de fondo a rojo (`#ff0000`).

Al agregar la propiedad de `transition`, podemos hacer que estos cambios ocurran con el tiempo:

```
.Example{  
    ...  
    transition: all 400ms ease;  
}
```

[Ver resultado](#)

`all` valor aplica la transición a todas las propiedades compatibles (basadas en números). Cualquier nombre de propiedad compatible (como `height` o `top`) puede sustituirse por esta palabra clave.

`400ms` especifica la cantidad de tiempo que tarda la transición. En este caso, el cambio de altura del elemento tardará 400 milisegundos en completarse.

Finalmente, la `ease` valor es la función de animación, que determina cómo se reproduce la animación. `ease` significa comenzar lento, acelerar, luego terminar lento nuevamente. Otros valores son `linear`, `de ease-out` y `de ease-in`.

Compatibilidad entre navegadores

La propiedad de `transition` general está bien soportada en todos los navegadores principales, excepto en IE 9. Para versiones anteriores de Firefox y navegadores basados en Webkit, use los prefijos de los proveedores, como por ejemplo:

```
.Example{  
    transition: all 400ms ease;  
    -moz-transition: all 400ms ease;  
    -webkit-transition: all 400ms ease;  
}
```

Nota: La propiedad de `transition` puede animar cambios entre dos valores numéricos cualquiera, independientemente de la unidad. También puede pasar entre las unidades, tal como `100px` a `50vh`. Sin embargo, no puede realizar la transición entre un número y un valor predeterminado o automático, como la transición de la altura de un elemento de `100px` a `auto`.

Incrementando el rendimiento de la animación usando el atributo `will-change`

Al crear animaciones y otras acciones pesadas de GPU, es importante entender el atributo `will-change`.

Tanto los fotogramas clave de CSS como la propiedad de `transition` utilizan aceleración de GPU. El rendimiento aumenta al descargar los cálculos a la GPU del dispositivo. Esto se hace creando capas de pintura (partes de la página que se representan individualmente) que se descargan en la GPU para calcular. La propiedad `will-change` le dice al navegador lo que animará, lo que le permite crear áreas de pintura más pequeñas, lo que aumenta el rendimiento.

La propiedad `will-change` acepta una lista de propiedades separadas por comas para animarlas. Por ejemplo, si planea transformar un objeto y cambiar su opacidad, debe especificar:

```
.Example{  
    ...  
    will-change: transform, opacity;  
}
```

Nota: Usa la `will-change` moderación. La configuración `will-change` de `will-change` para cada elemento en una página puede causar problemas 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.

Animaciones con fotogramas clave

Para animaciones CSS de varias etapas, puede crear CSS `@keyframes`. Los fotogramas clave le permiten definir múltiples puntos de animación, llamados fotogramas clave, para definir animaciones más complejas.

Ejemplo básico

En este ejemplo, haremos una animación de fondo básica que alterna entre todos los colores.

```
@keyframes rainbow-background {  
    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; }

}

.RainbowBackground {
    animation: rainbow-background 5s infinite;
}

```

[Ver resultado](#)

Hay algunas cosas diferentes a tener en cuenta aquí. Primero, la sintaxis real de `@keyframes` .

```
@keyframes rainbow-background{
```

Esto establece el nombre de la animación a `rainbow-background` .

```
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 está el fotograma clave durante la animación. El `0%` implica que es el 0% del tiempo total de animación desde el principio.

La animación hará una transición automática entre los fotogramas clave. Por lo tanto, al establecer el siguiente color de fondo en `8.333%` , la animación tomará suavemente `8.333%` del tiempo para la transición entre esos fotogramas clave.

```
.RainbowBackground {
    animation: rainbow-background 5s infinite;
}
```

Este código adjunta nuestra animación a todos los elementos que tienen la clase

```
.RainbowBackground .
```

La propiedad de animación real toma los siguientes argumentos.

- **animación-name:** El nombre de nuestra animación. En este caso, `rainbow-background`
- **Animación-duración :** el tiempo que durará la animación, en este caso 5 segundos.
- **animación-iteración-cuenta (Opcional) :** el número de veces que la animación se repetirá. En este caso, la animación continuará indefinidamente. Por defecto, la animación se reproducirá una vez.
- **animación-retraso (Opcional)** : especifica cuánto tiempo se debe esperar antes de que comience la animación. El valor predeterminado es 0 segundos y puede tomar valores negativos. Por ejemplo, `-2s` iniciaría la animación 2 segundos en su bucle.
- **animación-temporización-función (opcional)** : especifica la curva de velocidad de la animación. Por defecto es `ease` , donde la animación comienza lento, se vuelve más rápido y termina lento.

En este ejemplo particular, tanto el 0% como el 100% fotogramas clave especifican { background-color: #ff0000; }. Cuando dos o más fotogramas clave comparten un estado, uno puede especificarlos en una sola declaración. En este caso, las dos líneas de 0% y 100% podrían reemplazarse con esta línea única:

```
0%, 100% { background-color: #ff0000; }
```

Compatibilidad entre navegadores

Para los navegadores basados en WebKit más antiguos, deberá usar el prefijo de proveedor tanto en la declaración `@keyframes` como en la propiedad de `animation`, así:

```
@-webkit-keyframes {}  
-webkit-animation: ...
```

Ejemplos de sintaxis

Nuestro primer ejemplo de sintaxis muestra la propiedad abreviada de la 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 |  
play-state | name */
```

Nuestro segundo ejemplo es un poco más simple y muestra que algunas propiedades se pueden omitir:

```
animation: 3s linear 1s slidein;  
/* duration | timing-function | delay | name */
```

Nuestro tercer ejemplo muestra la declaración más mínima. Tenga en cuenta que el nombre de la animación y la duración de la animación deben declararse:

```
animation: 3s slidein;  
/* duration | name */
```

También vale la pena mencionar que cuando se usa la abreviatura de animación, el orden de las propiedades hace la 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;
```

Lea Animaciones en línea: <https://riptutorial.com/es/css/topic/590/animaciones>

Capítulo 5: Antecedentes

Introducción

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

Es posible especificar varias combinaciones de imágenes, colores y degradados, y ajustar el tamaño, la posición y la repetición (entre otras) de estos.

Sintaxis

- fondo-color: color | transparente | inicial | heredar;
- imagen de fondo: url | ninguno | inicial | heredar;
- posición de fondo: valor;
- tamaño de fondo: <tamaño-tamaño> [tamaño-tamaño]>
- <bg-size>: auto | longitud | cubierta | contener | inicial | heredar;
- repetición de fondo: repetir | repetir-x | repetir y no repetir inicial | heredar;
- origen-fondo: caja de relleno | caja de frontera | caja de contenido | inicial | heredar;
- clip de fondo: cuadro de borde | caja de relleno | caja de contenido | inicial | heredar;
- fondo-archivo adjunto: desplazamiento | fijo | local | inicial | heredar;
- de fondo: bg-color bg-image position / bg-size bg-repeat bg-origin bg-clip bg-attachment inicial | heredar;

Observaciones

- Los gradientes CSS3 no funcionarán en versiones de Internet Explorer de menos de 10.

Examples

Color de fondo

El `background-color` propiedad establece el color de fondo de un elemento utilizando un valor de color o por medio de palabras clave, tales como la `transparent`, `inherit` o `initial`.

- **transparente**, especifica que el color de fondo debe ser transparente. Esto es por defecto
- **heredar**, hereda esta propiedad de su elemento padre.
- **inicial**, establece esta propiedad en su valor predeterminado.

Esto se puede aplicar a todos los elementos, y `::first-letter` [pseudo-elementos de `::first-letter` / `::first-line`](#).

Los colores en CSS se pueden especificar por [diferentes métodos](#).

Nombres de colores

CSS

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

HTML

```
<div>This will have a red background</div>
```

- El ejemplo utilizado anteriormente es una de las varias formas en que CSS tiene que representar un solo color.

Códigos de color hexadecimales

El código hexadecimal se utiliza para indicar 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 de 256 bits (ff) y las partes verde y azul correspondientes del color son 0 (00).

Si ambos valores en 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 puede reducirse a #f00 , y #ffffff puede reducirse a #fff .

La notación hexadecimal no distingue entre mayúsculas y minúsculas.

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

RGB / RGBa

Otra forma de declarar un color es usar RGB o RGBa.

RGB significa rojo, verde y azul, y requiere de tres valores separados entre 0 y 255, colocados entre paréntesis, que se correspondan con los valores de color decimales de rojo, verde y azul respectivamente.

RGBa le permite agregar un parámetro alfa adicional entre 0.0 y 1.0 para definir la opacidad.

```
header {  
    background-color: rgb(0, 0, 0); /* black */
```

```
}

footer {
  background-color: rgba(0, 0, 0, 0.5); /* black with 50% opacity */
}
```

HSL / HSLa

Otra forma de declarar un color es usar HSL o HSLa y es similar a RGB y RGBa.

HSL significa tono, saturación y luminosidad, y también a menudo se llama HLS:

- El tono es un grado en la rueda de color (de 0 a 360).
- La saturación es un porcentaje entre 0% y 100%.
- La ligereza es también un porcentaje entre 0% y 100%.

HSLa le permite agregar un parámetro alfa adicional entre 0.0 y 1.0 para definir la opacidad.

```
li a {
  background-color: hsl(120, 100%, 50%); /* green */
}

#p1 {
  background-color: hsla(120, 100%, 50%, .3); /* green with 30% opacity */
}
```

Interacción con imagen de fondo.

Las siguientes afirmaciones son todas equivalentes:

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

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

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

body {
  background: red url(partiallytransparentimage.png);
}
```

Todos llevarán al color rojo que se muestra debajo de la imagen, donde las partes de la imagen son transparentes o la imagen no se muestra (tal vez como resultado de `background-repeat` del

```
background-repeat ).
```

Tenga en cuenta que lo siguiente no es equivalente:

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

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

Para obtener más información sobre la propiedad de `background`, consulte la [taquigrafía de fondo](#)

Imagen de fondo

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

```
.myClass {  
  background-image: url('/path/to/image.jpg');  
}
```

Para usar varias imágenes como `background-image`, defina `url()` separado por comas `url()`

```
.myClass {  
  background-image: url('/path/to/image.jpg'),  
                 url('/path/to/image2.jpg');  
}
```

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

Valor	Resultado
<code>url('/path/to/image.jpg')</code>	Especifique la (s) ruta (s) de la imagen de fondo o un recurso de imagen especificado con el esquema de URI de datos (se pueden omitir los apóstrofes), múltiples separados por comas
<code>none</code>	Sin imagen de fondo
<code>initial</code>	Valor por defecto
<code>inherit</code>	Heredar el valor de los padres

Más CSS para la imagen de fondo

Estos siguientes atributos son muy útiles y casi esenciales también.

```
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;
```

Gradientes de fondo

Los gradientes son nuevos tipos de imágenes, agregados en CSS3. Como imagen, los gradientes se configuran con la propiedad de `background-image` `background` o la taquigrafía de `background`.

Existen dos tipos de funciones de degradado, lineal y radial. Cada tipo tiene una variante de no repetición y una variante de repetición:

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

gradiente lineal()

Un `linear-gradient` tiene la siguiente sintaxis

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

Valor	Sentido
<code><direction></code>	Podría ser un argumento como <code>to top</code> , <code>to bottom</code> , <code>to right</code> o <code>to left</code> ; o un ángulo como <code>0deg</code> , <code>90deg</code> El ángulo comienza desde arriba hacia arriba y gira hacia la derecha. Puede especificarse en <code>grados</code> , <code>graduado</code> , <code>rad</code> , o <code>giro</code> . Si se omite, el gradiente fluye de arriba a abajo
<code><color-stop-list></code>	Lista de colores, opcionalmente seguidos cada uno por un porcentaje o longitud para mostrarlo en. Por ejemplo, <code>yellow 10%, rgba(0,0,0,.5) 40px, #fff 100% ...</code>

Por ejemplo, esto crea un degradado lineal que comienza desde la derecha y las transiciones de rojo a azul

```
.linear-gradient {  
  background: linear-gradient(to left, red, blue); /* you can also use 270deg */  
}
```

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

```
.diagonal-linear-gradient {  
  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.

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

gradiente radial ()

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

Valor	Sentido
circle	Forma del gradiente. Los valores son <code>circle</code> o <code>ellipse</code> , el valor predeterminado es <code>ellipse</code> .
farthest-corner	Palabras clave que describen qué tan grande debe ser la forma final. Los valores son <code>closest-side</code> <code>farthest-side</code> <code>closest-corner</code> , el <code>farthest-side</code> <code>closest-corner</code> <code>farthest-corner</code> , la <code>closest-corner</code> <code>farthest-corner</code> <code>closest-corner</code> <code>farthest-corner</code>
top left	Establece la posición del centro de degradado, de la misma manera que <code>background-position</code> .

Gradientes de repetición

Las funciones de degradado de repetición toman los mismos argumentos que los ejemplos anteriores, pero colocan el degradado en mosaico en el fondo del elemento.

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

Valor	Sentido
-45deg	Unidad de ángulo . El ángulo comienza desde arriba hacia arriba y gira hacia la derecha. Puede especificarse en grados , graduado , rad , o giro .

Valor	Sentido
to left	Dirección de gradiente, por defecto es <code>to bottom</code> . Sintaxis: <code>to [y-axis (top OR bottom)] [x-axis (left OR right)]</code> es decir, <code>to top right</code>
yellow 10%	Color, opcionalmente seguido de un porcentaje o longitud para mostrarlo en. Se repite dos o más veces.

Tenga en cuenta que los códigos de color HEX, RGB, RGBa, HSL y HSLa pueden usarse en lugar de los nombres de color. Los nombres de colores se utilizaron para ilustrar. También tenga en cuenta que la sintaxis de gradiente radial es mucho más compleja que la de gradiente lineal, y aquí se muestra una versión simplificada. Para una explicación completa y especificaciones, vea los [documentos de MDN](#)

Taquigrafía de fondo

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

Valor	Descripción	Ver CSS
<code>background-image</code>	Imagen de fondo para usar	1+
<code>background-color</code>	Color de fondo para 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>	Cómo repetir la imagen de fondo.	1+
<code>background-origin</code>	¿Cómo se coloca el fondo (se ignora cuando <code>background-attachment</code> se <code>fixed</code>)	3+
<code>background-clip</code>	Cómo se pinta el fondo en relación con el <code>content-box border-box</code> o el <code>padding-box</code>	3+
<code>background-attachment</code>	Cómo se comporta la imagen de fondo, ya sea que se desplace junto con su bloque que contiene o tenga una posición fija dentro de la ventana gráfica	1+
<code>initial</code>	Establece la propiedad en valor por defecto	3+
<code>inherit</code>	Hereda el valor de la propiedad del parente	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 configurando un `background-color` con el valor `red`.

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

Configuración de un `background-clip` en el cuadro de borde y un `background-color` en rojo.

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

Establece una `background-repeat` en no repetición, `background-origin` en el centro y una `background-image` en una imagen.

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

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

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

En este ejemplo tenemos un fondo negro con una imagen 'picture.png' en la parte superior, la imagen no se repite en ninguno de los ejes y se coloca en la esquina superior izquierda. La `/` después de la posición debe poder incluir el tamaño de la imagen de fondo, que en este caso se configura como ancho de `600px` y automático para la altura. Este ejemplo podría funcionar bien con una imagen de entidad que puede fundirse en un color sólido.

NOTA: El uso de la propiedad de fondo abreviada restablece todos los valores de propiedad de fondo establecidos previamente, incluso si no se proporciona un valor. Si solo desea modificar un valor de propiedad de fondo previamente establecido, use una propiedad de mano larga en su lugar.

Posición de fondo

La propiedad de `background-position` se usa para especificar la posición inicial para una imagen de

fondo o degradado

```
.myClass {  
    background-image: url('path/to/image.jpg');  
    background-position: 50% 50%;  
}
```

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

Unidad	Descripción
<i>valor %</i> <i>valor %</i>	Un porcentaje para el desplazamiento horizontal es relativo a (<i>ancho del área de posicionamiento del fondo - ancho 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 el tamaño de <code>background-size</code> .
<i>valor px</i> <i>valor px</i>	Desplaza la imagen de fondo en una longitud dada en píxeles relativa a la parte superior izquierda del área de posicionamiento de fondo

Las unidades en CSS se pueden especificar por diferentes métodos (ver [aquí](#)).

Propiedades de posición de fondo a largo plazo

Además de la propiedad abreviada anterior, también se pueden usar las propiedades de `background-position-x` largo plazo `background-position-x` y `background-position-y` . Estos le 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, será compatible con estas propiedades. Hasta entonces, [hay un hack de Firefox dentro de esta respuesta de desbordamiento de pila](#).

Adjunto de fondo

La propiedad de adjunto de fondo 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
voluta	El fondo se desplaza junto con el elemento. Esto es por defecto
fijo	El fondo se fija con respecto a la ventana gráfica.
local	El fondo se desplaza junto con el contenido del elemento.
inicial	Establece esta propiedad a su valor predeterminado.
heredar	Hereda esta propiedad de su elemento padre.

Ejemplos

fondo adjunto: desplazamiento

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

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

fondo adjunto: fijo

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

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

Fondo adjunto: local

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

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

Repetición de fondo

La propiedad de repetición de fondo 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 es como una `background-repeat: repeat-y` parece:



Color de fondo con opacidad

Si establece la `opacity` en un elemento, afectará a todos sus elementos secundarios. Para establecer una opacidad solo en el fondo de un elemento, tendrá que usar colores RGBA. El siguiente ejemplo tendrá un fondo negro con 0.6 opacidad.

```
/* Fallback for web browsers that don't support RGBa */  
background-color: rgb(0, 0, 0);  
  
/* RGBa with 0.6 opacity */  
background-color: rgba(0, 0, 0, 0.6);  
  
/* For IE 5.5 - 7*/  
filter: progid:DXImageTransform.Microsoft.gradient(startColorstr=#99000000,  
endColorstr=#99000000);  
  
/* For IE 8*/  
-ms-filter: "progid:DXImageTransform.Microsoft.gradient(startColorstr=#99000000,  
endColorstr=#99000000)";
```

Imagen de fondo múltiple

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

```
#mydiv {  
    background-image: url(img_1.png), /* top image */  
                     url(img_2.png), /* middle image */  
                     url(img_3.png); /* bottom image */  
    background-position: right bottom,  
                        left top,  
                        right top;  
    background-repeat: no-repeat,  
                      repeat,  
                      no-repeat;  
}
```

Las imágenes se apilarán una encima de la otra con el primer fondo en la parte superior y el último fondo en la parte posterior. `img_1` estará en la parte superior, `img_2` e `img_3` está en la parte inferior.

También podemos usar la propiedad taquigrafía de fondo para esto:

```
#mydiv {  
    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:

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

- [Manifestación](#)

La propiedad de origen-fondo.

La propiedad de origen de fondo especifica dónde se coloca la imagen de fondo.

Nota: Si la propiedad de `background-attachment` se establece en `fixed`, esta propiedad no tiene efecto.

Valor por defecto: `padding-box`

Valores posibles:

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

- inherit

CSS

```
.example {
  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;
}

.example1 {}

.example2 { background-origin: border-box; }

.example3 { background-origin: content-box; }
```

HTML

```
<p>No background-origin (padding-box is default):</p>

<div class="example example1">
  <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="example example2">
  <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="example example3">
  <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:

No background-origin (padding-box is default):



background-origin: border-box:



background-origin: content-box:



es el valor predeterminado. Esto permite que el fondo se extienda hasta el borde exterior del borde del elemento.

- `padding-box` sujet a el fondo en el borde exterior del relleno del elemento y no permite que se extienda hasta el borde;
- `content-box` recorta el fondo en el borde del cuadro de contenido.
- `inherit` aplica la configuració n del padre al elemento seleccionado.

CSS

```
.example {  
    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;  
}  
  
.example1 {}  
  
.example2 { background-origin: border-box; }  
  
.example3 { background-origin: content-box; }
```

HTML

```
<p>No background-origin (padding-box is default):</p>  
  
<div class="example example1">  
    <h2>Lorem Ipsum Dolor</h2>  
    <p>Lorem ipsum dolor sit amet, consectetuer 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="example example2">  
    <h2>Lorem Ipsum Dolor</h2>  
    <p>Lorem ipsum dolor sit amet, consectetuer 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="example example3">  
    <h2>Lorem Ipsum Dolor</h2>  
    <p>Lorem ipsum dolor sit amet, consectetuer 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>
```

Tamañ o de fondo

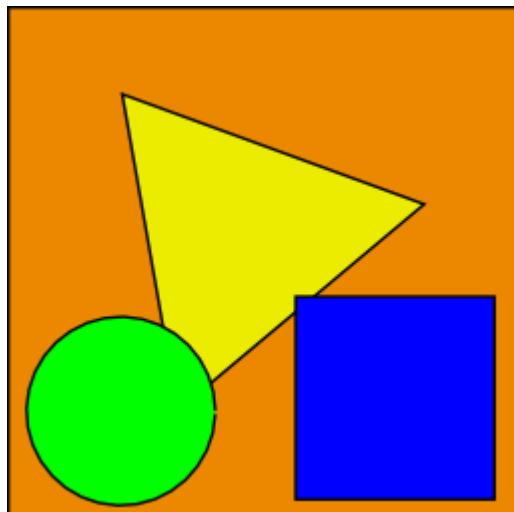
Visión general

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

`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 256 px × 256 px, todas las siguientes configuraciones de `background-size` producirían una imagen con una altura y un ancho de 50 px:

```
background-size: 50px;
background-size: 50px auto; /* same as above */
background-size: auto 50px;
background-size: 50px 50px;
```

Entonces, si comenzamos con la siguiente imagen (que tiene el tamaño mencionado de 256 px × 256 px),



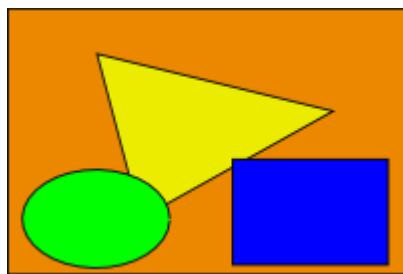
terminaremos con una imagen de 50 px × 50 px en la pantalla del usuario, contenida en el fondo de nuestro elemento:



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

```
#withbackground {
  background-image: url(to/some/background.png);
  background-size: 100% 66%;
  width: 200px;
  height: 200px;
  padding: 0;
```

```
margin: 0;  
}
```



El comportamiento depende del `background-origin`.

Manteniendo la relación de aspecto.

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

El enfoque de longitud o porcentaje no es lo suficientemente flexible para mantener la relación de aspecto en todo momento. `auto` no ayuda, ya que es posible que no sepa qué dimensión de su elemento será más grande. 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.

Plantas de huevo para `contain` y `cover`

Lo siento por el mal juego de palabras, pero vamos a usar una [foto del día de Biswarup Ganguly](#) para la demostración. Digamos que esta es su pantalla, y el área gris está fuera de su pantalla visible. Para demostración, vamos a asumir una proporción de 16×9 .



Queremos utilizar la imagen del día antes mencionada como fondo. Sin embargo, recortamos la imagen a 4×3 por alguna razón. Podríamos establecer la propiedad de `background-size` en una

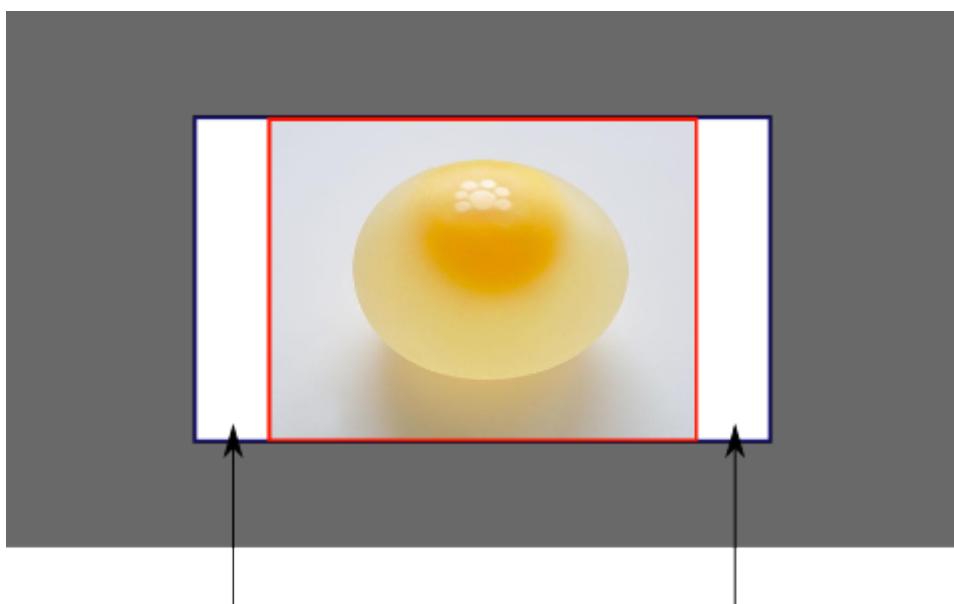
longitud fija, pero nos centraremos en `contain` y `cover`. Tenga en cuenta que también asumo que no estropeamos el ancho y / o la altura del `body`.

`contain`

contain

Escale la imagen, al tiempo que conserva su relación de aspecto intrínseca (si la hay), hasta el tamaño más grande, de manera que tanto su ancho como su altura puedan caber dentro del área de posicionamiento del fondo.

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

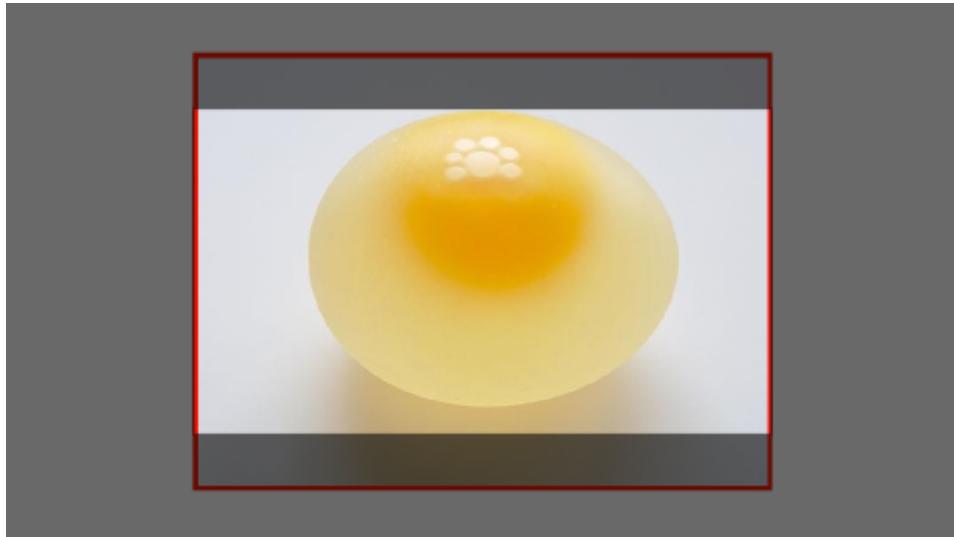


`cover`

cover

Escale la imagen, conservando su relación de aspecto intrínseco (si existe), hasta el tamaño más pequeño, de modo que tanto su ancho como su altura puedan cubrir completamente el área de posicionamiento del fondo.

Esto asegura que la imagen de fondo cubra todo. No habrá ningún `background-color` visible, sin embargo, dependiendo de la proporción de la pantalla, se puede cortar una gran parte de su imagen:



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;  
}  
*****  
Additional styles for the explanation boxes  
*****  
  
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>Note the grey background. The image does not cover the whole region, but it's fully contained.</em>.
  </p>
</div>
<div>
  <div class="cover"></div>
  <p>Note the ducks/geese at the bottom of the image. Most of the water is cut, as well as a part of the sky. You don't see the complete image anymore, but neither do you see any background color; the image <em>covers</em> all of the <code>&lt;div&gt;</code>. </p>
</div>

```



Note the grey background. The image does not cover the whole region, but it's fully *contained*.



Note the ducks/geese at the bottom of the image. Most of the water is cut, as well as a part of the sky. You don't see the complete image anymore, but neither do you see any background color; the image *covers* all of the <div>.

Propiedad de modo de mezcla de fondo

```

.my-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="my-div">Lorem ipsum</div>
```

Vea el resultado aquí: <https://jsfiddle.net/MadalinaTn/y69d28Lb/>

Sintaxis CSS: background-blend-mode: normal | multiplicar pantalla | superposición | oscurecer
aclarar color-dodge | saturacion | color | luminosidad;

Lea Antecedentes en línea: <https://riptutorial.com/es/css/topic/296/antecedentes>

Capítulo 6: Cascada y especificidad

Observaciones

La especificidad de CSS pretende promover la concisión del código al permitir que un autor defina algunas reglas de formato generales para un conjunto amplio de elementos y luego las anule para un determinado subconjunto.

Examples

En cascada

La cascada y la especificidad se utilizan juntas para determinar el valor final de una propiedad de estilo CSS. También definen los mecanismos para resolver conflictos en conjuntos de reglas CSS.

CSS orden de carga

Los estilos se leen de las siguientes fuentes, en 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 aquí significa el creador de la página web / sitio web)
 - Tal vez uno o más archivos .css
 - En el elemento `<style>` del documento HTML
4. Estilos en línea (en el atributo de `style` en un elemento HTML)

El navegador buscará los estilos correspondientes al renderizar un elemento.

¿Cómo se resuelven los conflictos?

Cuando solo un conjunto de reglas CSS intenta establecer un estilo para un elemento, entonces no hay conflicto y se usa ese conjunto de reglas.

Cuando se encuentran varios conjuntos de reglas con configuraciones en conflicto, primero las reglas de Especificidad y luego las reglas en cascada se usan para determinar qué estilo usar.

Ejemplo 1 - Reglas de especificidad

```
.mystyle { color: blue; } /* specificity: 0, 0, 1, 0 */
```

```
div { color: red; }          /* specificity: 0, 0, 0, 1 */
```

```
<div class="mystyle">Hello World</div>
```

¿De qué color será el texto? (pasa el cursor para ver la respuesta)

azul

Primero se aplican las reglas de especificidad, y la que tiene la mayor especificidad "gana".

Ejemplo 2 - Reglas de cascada con selectores idénticos

Archivo css externo

```
.class {  
  background: #FFF;  
}
```

Css interno (en archivo HTML)

```
<style>  
.class {  
  background: #000;  
}  

```

En este caso, donde tiene selectores idénticos, la cascada se activa y determina que el último cargado "gana".

Ejemplo 3 - Reglas en cascada después de las reglas de especificidad

```
body > .mystyle { background-color: blue; }  /* specificity: 0, 0, 1, 1 */  
.otherstyle > div { background-color: red; }  /* specificity: 0, 0, 1, 1 */
```

```
<body class="otherstyle">  
  <div class="mystyle">Hello World</div>  
</body>
```

¿De qué color será el fondo?

rojo

Después de aplicar las reglas de especificidad, todavía hay un conflicto entre azul y rojo, por lo que las reglas en cascada se aplican sobre las reglas de especificidad. En cascada se analiza el orden de carga de las reglas, ya sea dentro del mismo archivo .css o en la colección de fuentes de estilo. El último cargado anula los anteriores. En este caso, la regla .otherstyle > div "gana".

Una nota final

- La especificidad del selector siempre tiene prioridad.
- Hoja de estilo para romper lazos.
- Los estilos en línea triunfan sobre todo.

La! Declaración importante

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

```
#mydiv {  
    font-weight: bold !important;      /* This property won't be overridden  
                                       by the rule below */  
}  
  
#outerdiv #mydiv {  
    font-weight: normal;              /* #mydiv font-weight won't be set to normal  
                                       even if it has a higher specificity because  
                                       of the !important declaration above */  
}
```

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

Aquí hay algunos ejemplos donde se puede justificar el uso de `!important` declaración `!important`:

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

Ver también:

- [W3C - 6 Asignación de valores de propiedad, en cascada y herencia - ¡6.4.2! Reglas importantes](#)

Cálculo de la especificidad del selector

Cada selector de CSS individual tiene su propio valor de especificidad. Cada selector en una secuencia aumenta la especificidad general de la secuencia. Los selectores se clasifican en uno de tres grupos de especificidad diferentes: *A*, *B* y *c*. Cuando múltiples secuencias de selección seleccionan un elemento dado, el navegador utiliza los estilos aplicados por la secuencia con la mayor especificidad general.

Grupo	Compuesto de	Ejemplos
UNA	selectores de identificación	#foo
segundo	selectores de clase selectores de atributos pseudo-clases	.bar [title] , [colspan="2"] :hover :nth-child(2)
do	selectores de tipo pseudo-elementos	div , li ::before , ::first-letter

El grupo A es el más específico, seguido del grupo B y, finalmente, el grupo c .

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

Ejemplo 1: Especificidad de varias secuencias de selección

```
#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 se usa la especificidad por 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 de Grupo B. Un selector de ID supera cualquier número de selectores de clase. Por eso, `color:blue;` desde el selector `#foo` y el `background:black;` Desde el selector `.bar` se aplicará al elemento. La mayor especificidad del selector de ID hará que el navegador ignore la `.bar` de `color` del selector `.bar`.

Ahora imagine una implementación diferente de CSS:

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

Ejemplo 3: Cómo manipular la especificidad.

El último fragmento del Ejemplo 2 anterior se puede manipular para garantizar que se `.bar` la declaración de `color` del selector de clase `.bar` 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 común de lograr esto sería averiguar qué otros selectores se pueden aplicar a la secuencia del selector `.bar`. Por ejemplo, si el `.bar` clase sólo se aplicó nunca a `span` elementos, podríamos modificar el `.bar` selector para `span.bar`. Esto le daría una nueva especificidad para el Grupo C, que anularía la falta del mismo `.baz` selector:

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

Sin embargo, es posible que no siempre sea posible encontrar otro selector común que se comparta entre cualquier elemento que use la clase `.bar`. Debido a esto, CSS nos permite duplicar los selectores para aumentar la especificidad. En lugar de solo `.bar`, podemos usar `.bar.bar` en `.bar.bar` lugar (Ver [La gramática de los selectores, Recomendación W3C](#)). Esto aún selecciona cualquier elemento con una clase de `.bar`, pero ahora tiene el doble de especificidad de Grupo *B*:

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

Declaraciones de estilo `!important` y en línea.

Se considera que la `!important` en una declaración de estilo y los estilos declarados por el atributo de `style` HTML tienen una mayor especificidad que cualquier selector. Si existen, la declaración de estilo que afectan anularán 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 aplica al mismo elemento. Luego, las reglas de especificidad normales se aplicarán a esas propiedades en referencia entre sí.

Debido a que anulan completamente la especificidad, el uso de `!important` está mal visto en la mayoría de los casos de uso. Uno debería usarlo 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 utilizarlo `!important`

Una de esas raras excepciones en las que lo `!important` no está mal visto, es cuando se implementan clases auxiliares genéricas como una `.hidden` o `.background-yellow` que se supone que siempre anulan una o más propiedades donde sea que se encuentren. E incluso entonces, necesitas saber lo que estás haciendo. Lo último que desea, al escribir CSS mantenable, es tener `!important` todo su CSS.

Una nota final

Un error común acerca de la especificidad de CSS es que los valores de los grupos *A*, *B* y *c* deben combinarse entre sí (`a=1, b=5, c=1 => 151`). Este **no** es el caso. Si este fuera el caso, tener 20 de un selector de grupo *B* o *c* sería suficiente para anular un solo selector de grupo *A* o *B*, respectivamente. Los tres grupos deben considerarse como niveles individuales de especificidad. La especificidad no puede ser representada por un solo valor.

Al crear su hoja de estilo CSS, debe mantener la menor especificidad posible. Si necesita hacer que la especificidad sea un poco más alta para sobrescribir otro método, hágalo más alto pero lo más bajo posible para hacerlo más alto. No deberías tener un selector como este:

```
body.page header.container nav div#main-nav li a {}
```

Esto hace que los cambios futuros sean más difíciles y contaminan esa página css.

Puedes calcular la especificidad de tu selector [aquí](#)

Ejemplo de especificidad más compleja.

```
div {  
    font-size: 7px;  
    border: 3px dotted pink;  
    background-color: yellow;  
    color: purple;  
}  
  
body.mystyle > div.myotherstyle {  
    font-size: 11px;  
    background-color: green;  
}  
  
#elmnt1 {  
    font-size: 24px;  
    border-color: red;  
}  
  
.mystyle .myotherstyle {  
    font-size: 16px;  
    background-color: black;  
    color: red;  
}
```

```
<body class="mystyle">  
    <div id="elmnt1" class="myotherstyle">  
        Hello, world!  
    </div>  
</body>
```

¿Qué bordes, colores y tamaño de fuente será el texto?

tamaño de fuente:

`font-size: 24;`, dado que `#elmnt1` reglas `#elmnt1` tiene la mayor especificidad para el `<div>` en cuestión, todas las propiedades aquí se establecen.

frontera:

`border: 3px dotted red;`. El `red` color del borde se toma del conjunto de reglas `#elmnt1`, ya que tiene la mayor especificidad. Las otras propiedades del borde, grosor de borde y estilo de borde son del conjunto de reglas `div`.

color de fondo:

`background-color: green;`. El `background-color` se establece en los conjuntos de `body.mystyle > div.myotherstyle div`, `body.mystyle > div.myotherstyle` y `.mystyle .myotherstyle`. Las especificidades son $(0, 0, 1)$ vs. $(0, 2, 2)$ vs. $(0, 2, 0)$, por lo que el medio "gana".

color:

`color: red;`. El color se establece en los conjuntos de reglas `div` y `.mystyle`

`.myotherstyle` . Este último tiene la especificidad más alta de (0, 2, 0) y "gana".

Lea Cascada y especificidad en línea: <https://riptutorial.com/es/css/topic/450/cascada-y-especificidad>

Capítulo 7: Centrado

Examples

Usando la transformación CSS

Las transformaciones de CSS se basan en el tamaño de los elementos, por lo que si no sabe cuán alto o ancho es su elemento, puede colocarlo absolutamente en un 50% desde la parte superior e izquierda de un contenedor relativo y traducirlo en un 50% hacia la izquierda y hacia arriba para centrarlo verticalmente y horizontalmente.

Tenga en cuenta que con esta técnica, el elemento podría terminar siendo procesado en un límite de píxel no entero, lo que hace que se vea borroso. Ver [esta respuesta en SO](#) para una solución.

HTML

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

CSS

```
.container {
  position: relative;
}

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

[Ver ejemplo en JSFiddle](#)

COMPATIBILIDAD DE NAVEGADOR CRUZADO

La propiedad de transformación necesita que los prefijos sean compatibles con los navegadores más 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.

Aquí hay 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 [canluse](#).

MÁS INFORMACIÓN

- El elemento se posiciona de acuerdo con el primer elemento primario no estático (`position: relative`, `absolute` o `fixed`). Explora más en este [violín](#) y este [tema de documentación](#).
- Para el centrado solo horizontal, use `left: 50%` y `transform: translateX(-50%)`. Lo mismo ocurre con el centrado solo vertical: centro con la `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 principalmente con elementos que contienen texto, y se puede arreglar agregando: `margin-right: -50%`; y `margin-bottom: -50%;`. Ver este [violín](#) para más información.

Usando flexbox

HTML:

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

CSS:

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

[Ver resultado](#)

HTML:

```

```

CSS:

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

```
    align-items: center;      /* vertical center */
}
```

[Ver resultado](#)

Consulte [Centrado vertical y horizontal dinámico](#) en la documentación de [Flexbox](#) para obtener más detalles sobre flexbox y lo que significan los estilos.

Sopor te del navegador

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

Algunas versiones recientes del navegador, como Safari 8 y IE10, requieren [prefijos de proveedores](#).

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

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

Para una visión más detallada de la compatibilidad con el navegador flexbox, vea [esta respuesta](#).

Posición de uso: absoluta

Trabajando en navegadores antiguos (IE>=8)

Los márgenes automáticos, emparejados con valores de cero para las compensaciones `left` y `right` o `top` e `bottom`, centrarán un elemento absolutamente posicionado dentro de su parente.

[Ver resultado](#)

HTML

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

CSS

```
.parent {
  position: relative;
  height: 500px;
}

.center {
  position: absolute;
  margin: auto;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
}
```

Los elementos que no tienen su propio ancho y altura implícitos, como las imágenes, necesitarán definir esos valores.

Otros recursos: [Absolute Centering en CSS](#)

Técnica del elemento fantasma (hack de Michał Czernow)

Esta técnica funciona incluso cuando las dimensiones del contenedor son desconocidas.

Configure un elemento "fantasma" dentro del contenedor para que esté centrado a una altura del 100%, luego use `vertical-align: middle` en eso y en el elemento que se centrará.

CSS

```
/* This parent can be any width and height */
.block {
    text-align: center;

    /* May want to do this if there is risk the container may be narrower than the element
    inside */
    white-space: nowrap;
}

/* The ghost element */
.block:before {
    content: '';
    display: inline-block;
    height: 100%;
    vertical-align: middle;

    /* There is a gap between ghost element and .centered,
    caused by space character rendered. Could be eliminated by
    nudging .centered (nudge distance depends on font family),
    or by zeroing font-size in .parent and resetting it back
    (probably to 1rem) in .centered. */
    margin-right: -0.25em;
}

/* The element to be centered, can also be of any width and height */
.centered {
    display: inline-block;
    vertical-align: middle;
    width: 300px;
    white-space: normal; /* Resetting inherited nowrap behavior */
}
```

HTML

```
<div class="block">
    <div class="centered"></div>
</div>
```

Usando `text-align`

El tipo de centrado más común y más fácil es el de las líneas de texto en 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` controla solo la alineación del contenido en línea como el texto en su elemento de bloque principal.

Ver más sobre `text-align` en la sección de [tipografía](#) .

Centrado en relación a otro elemento.

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

Compatibilidad: IE8 +, todos los demás navegadores modernos.

HTML

```
<div class="content">  
  <div class="position-container">  
    <div class="thumb">  
        
    </div>  
    <div class="details">  
      <p class="banner-title">text 1</p>  
      <p class="banner-text">content content content content content content content  
      content content content content content content content content</p>  
      <button class="btn">button</button>  
    </div>  
  </div>  
</div>
```

CSS

```
.content * {  
  box-sizing: border-box;  
}  
.content .position-container {  
  display: table;  
}  
.content .details {  
  display: table-cell;  
  vertical-align: middle;  
  width: 33.333333%;  
  padding: 30px;  
  font-size: 17px;  
  text-align: center;
```

```
}

.content .thumb {
  width: 100%;
}

.content .thumb img {
  width: 100%;
}
```

Enlace a JSFiddle

Los puntos principales son los `.thumb`, `3 .thumb`, `.details` y `.position-container`:

- El `.position-container` debe tener `display: table`.
- Los `.details` deben tener el ancho real establecido `width:` y `display: table-cell`, `vertical-align: middle`.
- El `.thumb` debe tener `width: 100%` si quieras que tomará todo el espacio restante y se verá influido por el `.details` ancho.
- La imagen (si tiene una imagen) dentro de `.thumb` debe tener un `width: 100%`, pero no es necesario si tiene las proporciones correctas.

Alinear verticalmente cualquier cosa con 3 líneas de código.

Compatible con IE11 +

[Ver resultado](#)

Usa estas 3 líneas para alinear verticalmente prácticamente todo. Solo asegúrese de que la div / imagen a la que aplica el código tenga un parente con una altura.

CSS

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

HTML

```
<div class="vertical">Vertical aligned text!</div>
```

Alinear verticalmente una imagen dentro de div

HTML

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

CSS

```
.wrap {  
    height: 50px; /* max image height */  
    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;  
}
```

Centrado horizontal y vertical utilizando el diseño de la mesa.

Uno podría centrar fácilmente un elemento secundario usando la propiedad de visualización de `table`.

HTML

```
<div class="wrapper">  
    <div class="parent">  
        <div class="child"></div>  
    </div>  
</div>
```

CSS

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

Utilizando calc ()

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

CSS

```
.center {  
    position: absolute;  
    height: 50px;  
    width: 50px;  
    background: red;  
    top: calc(50% - 50px / 2); /* height divided by 2 */  
    left: calc(50% - 50px / 2); /* width divided by 2 */  
}
```

HTML

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

Alinear verticalmente elementos dinámicos de altura

Aplicar css de forma intuitiva no produce los resultados deseados porque

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

Para el soporte de navegador más amplio, una solución alternativa con elementos de ayuda:

HTML

```
<div class="vcenter--container">  
    <div class="vcenter--helper">  
        <div class="vcenter--content">  
            <!--stuff-->  
        </div>  
    </div>  
</div>
```

CSS

```
.vcenter--container {  
    display: table;  
    height: 100%;  
    position: absolute;  
    overflow: hidden;
```

```
width: 100%;  
}  
.vcenter--helper {  
display: table-cell;  
vertical-align: middle;  
}  
.vcenter--content {  
margin: 0 auto;  
width: 200px;  
}
```

[jsfiddle](#) de la [pregunta original](#). Este enfoque

- Trabaja con elementos dinámicos de altura.
- respeta el flujo de contenido
- es compatible con los navegadores heredados

Usando la línea de altura

También puede usar `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 de `line-height` solo funciona cuando el texto que se va a centrar abarca una sola línea. Si el texto se ajusta en varias líneas, la salida resultante no estará centrada.

Centrado vertical y horizontalmente sin preocuparse por la altura o el ancho

La siguiente técnica le permite agregar su contenido a un elemento HTML y centrarlo tanto horizontal como verticalmente **sin preocuparse por su altura o ancho**.

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 quieras que el texto esté centrado

Manifestación

HTML

```
<div class="outer-container">
  <div class="inner-container">
    <div class="centered-content">
      You can put anything here!
    </div>
  </div>
</div>
```

CSS

```
body {
  margin : 0;
}

.outer-container {
  position : absolute;
  display: table;
  width: 100%; /* This could be ANY width */
  height: 100%; /* This could be ANY height */
  background: #ccc;
}

.inner-container {
  display: table-cell;
  vertical-align: middle;
  text-align: center;
}

.centered-content {
  display: inline-block;
  text-align: left;
  background: #fff;
  padding: 20px;
  border: 1px solid #000;
}
```

Véase también [este violín](#) !

Centrado con tamaño fijo.

Si el tamaño de su contenido es fijo, puede usar el posicionamiento absoluto al 50% con un `margin` que reduce la mitad del ancho y la altura de su contenido:

HTML

```
<div class="center">
```

```
Center vertically and horizontally  
</div>
```

CSS

```
.center {  
    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 con un solo ancho fijo.

Puede centrar el elemento horizontalmente incluso si no conoce la altura del contenido:

HTML

```
<div class="center">  
    Center only horizontally  
</div>
```

CSS

```
.center {  
    position: absolute;  
    background: #ccc;  
  
    left: 50%;  
    width: 150px;  
    margin-left: -75px; /* width * -0.5 */  
}
```

Centrado vertical con altura fija.

Puede centrar el elemento verticalmente si conoce la altura del elemento:

HTML

```
<div class="center">  
    Center only vertically  
</div>
```

CSS

```

.center {
    position: absolute;
    background: #ccc;

    top: 50%;
    height: 200px;
    margin-top: -100px; /* width * -0.5 */
}

```

Usando el margen: 0 auto;

Los objetos se pueden centrar usando el `margin: 0 auto;` Si son elementos de bloque y tienen un ancho definido.

HTML

```

<div class="containerDiv">
    <div id="centeredDiv"></div>
</div>

<div class="containerDiv">
    <p id="centeredParagraph">This is a centered paragraph.</p>
</div>

<div class="containerDiv">
    
</div>

```

CSS

```

.containerDiv {
    width: 100%;
    height: 100px;
    padding-bottom: 40px;
}

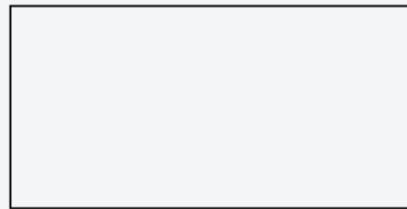
#centeredDiv {
    margin: 0 auto;
    width: 200px;
    height: 100px;
    border: 1px solid #000;
}

#centeredParagraph {
    width: 200px;
    margin: 0 auto;
}

#centeredImage {
    display: block;
    width: 200px;
    margin: 0 auto;
}

```

Resultado:



This is a centered paragraph.



Ejemplo de JSFiddle: [Centrar objetos con margen: 0 auto;](#)

Lea Centrado en línea: <https://riptutorial.com/es/css/topic/299/centrado>

Capítulo 8: Centrado vertical

Observaciones

Esto se usa cuando las dimensiones del elemento (`width` y `height`) no son conocidas o dinámicas.

Prefiere usar **Flexbox** sobre todas las demás opciones, ya que está optimizado para el diseño de la interfaz de usuario.

Examples

Centrado con `display: mesa`

HTML:

```
<div class="wrapper">
  <div class="outer">
    <div class="inner">
      centered
    </div>
  </div>
</div>
```

CSS:

```
.wrapper {
  height: 600px;
  text-align: center;
}
.outer {
  display: table;
  height: 100%;
  width: 100%;
}
.outer .inner {
  display: table-cell;
  text-align: center;
  vertical-align: middle;
}
```

Centrado con transformar

HTML:

```
<div class="wrapper">
  <div class="centered">
    centered
  </div>
</div>
```

CSS:

```
.wrapper {  
    position: relative;  
    height: 600px;  
}  
.centered {  
    position: absolute;  
    z-index: 999;  
    transform: translate(-50%, -50%);  
    top: 50%;  
    left: 50%;  
}
```

Centrado con Flexbox

HTML:

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

CSS:

```
.container {  
    height: 500px;  
    width: 500px;  
    display: flex;           // Use Flexbox  
    align-items: center;     // This centers children vertically in the parent.  
    justify-content: center; // This centers children horizontally.  
    background: white;  
}  
  
.child {  
    width: 100px;  
    height: 100px;  
    background: blue;  
}
```

Centrado de texto con altura de línea

HTML:

```
<div class="container">  
    <span>vertically centered</span>  
</div>
```

CSS:

```
.container{  
    height: 50px;           /* set height */  
    line-height: 50px;       /* set line-height equal to the height */  
    vertical-align: middle; /* works without this rule, but it is good having it explicitly  
    set */
```

```
}
```

Nota: este método solo centrará verticalmente una sola línea de texto . No centrará los elementos del bloque correctamente y si el texto se divide en una nueva línea, tendrá dos líneas muy altas de texto.

Centrado con Posición: absoluta

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 desea centrar otras imágenes, debe dar altura y ancho a ese elemento.

HTML:

```
<div class="wrapper">
  <div class="child">
    make me center
  </div>
</div>
```

CSS:

```
.wrapper{
  position: relative;
  height: 600px;
}
.wrapper .child {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  margin: auto;
  width: 200px;
```

```
height: 30px;  
border: 1px solid #f00;  
}
```

Centrado con pseudo elemento.

HTML:

```
<div class="wrapper">  
  <div class="content"></div>  
</div>
```

CSS:

```
.wrapper{  
  min-height: 600px;  
}  
  
.wrapper:before{  
  content: "";  
  display: inline-block;  
  height: 100%;  
  vertical-align: middle;  
}  
  
.content {  
  display: inline-block;  
  height: 80px;  
  vertical-align: middle;  
}
```

Este método se utiliza mejor en los casos en los que tiene un .content altura .content centrado dentro de .wrapper ; y desea .wrapper 'altura de expandir cuando .content ' s altura exceda .wrapper min-height 's.

Lea Centrado vertical en línea: <https://riptutorial.com/es/css/topic/5070/centrado-vertical>

Capítulo 9: Colores

Sintaxis

- color: #rgb
- color: #rrggbb
- color: rgb [a] (<red>, <green>, <blue> [, <alpha>])
- color: hsl [a] (<hue>, <saturation%>, <lightness%> [, <alpha>])
- color: **colorkeyword** /* verde, azul, amarillo, naranja, rojo, ..etc */

Examples

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, use la palabra clave `blue`:

```
.some-class {  
    color: blue;  
}
```

Las palabras clave de CSS no distinguen entre mayúsculas y minúsculas: `blue`, `Blue` y `BLUE` darán como resultado `#0000FF`.

Palabras clave de color

Nombre del color	Valor hexadecimal	Valores RGB	Color
Alice azul	# F0F8FF	rgb (240,248,255)	
Blanco antiguo	# FAEBD7	rgb (250,235,215)	
Agua	# 00FFFF	rgb (0,255,255)	
Aguamarina	# 7FFFAD	rgb (127,255,212)	
Azur	# F0FFFF	rgb (240,255,255)	
Beige	# F5F5DC	rgb (245,245,220)	
Sopa de mariscos	# FFE4C4	rgb (255,228,196)	

Nombre del color	Valor hexadecimal	Valores RGB	Color
Negro	# 000000	rgb (0,0,0)	
BlanchedAlmond	#FFEBCD	rgb (255,235,205)	
Azul	# 0000FF	rgb (0,0,255)	
Violeta Azul	# 8A2BE2	rgb (138,43,226)	
marrón	# A52A2A	rgb (165,42,42)	
BurlyWood	# DEB887	rgb (222,184,135)	
CadetBlue	# 5F9EA0	rgb (95,158,160)	
Monasterio	# 7FFF00	rgb (127,255,0)	
Chocolate	# D2691E	rgb (210,105,30)	
Coral	# FF7F50	rgb (255,127,80)	
Aciano Azul	# 6495ED	rgb (100,149,237)	
Seda de maiz	# FFF8DC	rgb (255,248,220)	
carmesí	# DC143C	rgb (220,20,60)	
Cian	# 00FFFF	rgb (0,255,255)	
Azul oscuro	# 00008B	rgb (0,0,139)	
DarkCyan	# 008B8B	rgb (0,139,139)	
DarkGoldenRod	# B8860B	rgb (184,134,11)	
Gris oscuro	# A9A9A9	rgb (169,169,169)	
Gris oscuro	# A9A9A9	rgb (169,169,169)	
Verde oscuro	# 006400	rgb (0,100,0)	
DarkKhaki	# BDB76B	rgb (189,183,107)	

Nombre del color	Valor hexadecimal	Valores RGB	Color
DarkMagenta	# 8B008B	rgb (139,0,139)	
DarkOliveGreen	# 556B2F	rgb (85,107,47)	
Naranja oscuro	# FF8C00	rgb (255,140,0)	
Orquídea Oscura	# 9932CC	rgb (153,50,204)	
Rojo oscuro	# 8B0000	rgb (139,0,0)	
Salmón oscuro	# 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)	
Violeta oscuro	# 9400D3	rgb (148,0,211)	
Rosa profundo	# 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)	
Ladrillo refractario	# B22222	rgb (178,34,34)	
FloralWhite	# FFFAF0	rgb (255,250,240)	
Bosque verde	# 228B22	rgb (34,139,34)	
Fucsia	# FF00FF	rgb (255,0,255)	

Nombre del color	Valor hexadecimal	Valores RGB	Color
Gainsboro	#DCDCDC	rgb (220,220,220)	
Fantasma blanco	# F8F8FF	rgb (248,248,255)	
Oro	# FFD700	rgb (255,215,0)	
Vara de oro	# DAA520	rgb (218,165,32)	
gris	# 808080	rgb (128,128,128)	
Gris	# 808080	rgb (128,128,128)	
Verde	# 008000	rgb (0,128,0)	
Verde amarillo	# ADFF2F	rgb (173,255,47)	
Gotas de miel	# F0FFF0	rgb (240,255,240)	
Rosa caliente	# FF69B4	rgb (255,105,180)	
IndianRed	# CD5C5C	rgb (205,92,92)	
Índigo	# 4B0082	rgb (75,0,130)	
Marfil	# FFFFF0	rgb (255,255,240)	
Caqui	# F0E68C	rgb (240,230,140)	
Lavanda	# E6E6FA	rgb (230,230,250)	
LavenderBlush	# FFF0F5	rgb (255,240,245)	
Césped verde	# 7CFC00	rgb (124,252,0)	
Limón chiffon	#FFFACD	rgb (255,250,205)	
Azul claro	# ADD8E6	rgb (173,216,230)	
LightCoral	# F08080	rgb (240,128,128)	
Cian claro	# E0FFFF	rgb (224,255,255)	

Nombre del color	Valor hexadecimal	Valores RGB	Color
LightGoldenRodYellow	# FAFAD2	rgb (250,250,210)	
Gris claro	# D3D3D3	rgb (211,211,211)	
Gris claro	# D3D3D3	rgb (211,211,211)	
Verde claro	# 90EE90	rgb (144,238,144)	
Rosa claro	# FFB6C1	rgb (255,182,193)	
Salmón de luz	# 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)	
Amarillo claro	# FFFF00	rgb (255,255,224)	
Lima	# 00FF00	rgb (0,255,0)	
Verde lima	# 32CD32	rgb (50,205,50)	
Lino	# FAF0E6	rgb (250,240,230)	
Magenta	# FF00FF	rgb (255,0,255)	
Granate	# 800000	rgb (128,0,0)	
MedioAquaMarine	# 66CDAA	rgb (102,205,170)	
Azul medio	# 0000CD	rgb (0,0,205)	
MediumOrchid	# BA55D3	rgb (186,85,211)	
Mediano	# 9370DB	rgb (147,112,219)	

Nombre del color	Valor hexadecimal	Valores RGB	Color
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)	
Mocasín	# FFE4B5	rgb (255,228,181)	
Navajoblanco	#FFDEAD	rgb (255,222,173)	
Armada	# 000080	rgb (0,0,128)	
Antiguo lace	# FDF5E6	rgb (253,245,230)	
Aceituna	# 808000	rgb (128,128,0)	
Verde oliva	# 6B8E23	rgb (107,142,35)	
naranja	# FFA500	rgb (255,165,0)	
Rojo naranja	# FF4500	rgb (255,69,0)	
Orquídea	# DA70D6	rgb (218,112,214)	
PaleGoldenRod	# EEE8AA	rgb (238,232,170)	
Verde pálido	# 98FB98	rgb (152,251,152)	
PaleTurquoise	#AFEEEE	rgb (175,238,238)	
PaleVioletRed	# DB7093	rgb (219,112,147)	

Nombre del color	Valor hexadecimal	Valores RGB	Color
PapayaWhip	# FFEFD5	rgb (255,239,213)	
Peachpuff	# FFDAB9	rgb (255,218,185)	
Perú	# CD853F	rgb (205,133,63)	
Rosado	# FFC0CB	rgb (255,192,203)	
ciruela	# DDA0DD	rgb (221,160,221)	
Azul pálido	# B0E0E6	rgb (176,224,230)	
Púrpura	# 800080	rgb (128,0,128)	
RebeccaPurple	# 663399	rgb (102,51,153)	
rojo	# FF0000	rgb (255,0,0)	
RosyBrown	# BC8F8F	rgb (188,143,143)	
Azul real	# 4169E1	rgb (65,105,225)	
SaddleBrown	# 8B4513	rgb (139,69,19)	
Salmón	# FA8072	rgb (250,128,114)	
SandyBrown	# F4A460	rgb (244,164,96)	
Mar verde	# 2E8B57	rgb (46,139,87)	
Concha	# FFF5EE	rgb (255,245,238)	
Tierra de siena	# A0522D	rgb (160,82,45)	
Plata	# C0C0C0	rgb (192,192,192)	
Cielo azul	# 87CEEB	rgb (135,206,235)	
SlateBlue	# 6A5ACD	rgb (106,90,205)	
SlateGray	# 708090	rgb (112,128,144)	

Nombre del color	Valor hexadecimal	Valores RGB	Color
Pizarra gris	# 708090	rgb (112,128,144)	
Nieve	#FFAFA	rgb (255,250,250)	
Primavera verde	# 00FF7F	rgb (0,255,127)	
Azul acero	# 4682B4	rgb (70,130,180)	
Bronceado	# D2B48C	rgb (210,180,140)	
Teal	# 008080	rgb (0,128,128)	
Cardo	# D8BFD8	rgb (216,191,216)	
Tomate	# FF6347	rgb (255,99,71)	
Turquesa	# 40E0D0	rgb (64,224,208)	
Violeta	# EE82EE	rgb (238,130,238)	
Trigo	# F5DEB3	rgb (245,222,179)	
Blanco	#FFFFFF	rgb (255,255,255)	
Humo blanco	# F5F5F5	rgb (245,245,245)	
Amarillo	# FFFF00	rgb (255,255,0)	
Amarillo verde	# 9ACD32	rgb (154,205,50)	

Además de los colores nombrados, también existe la palabra clave `transparent`, que representa un negro totalmente transparente: `rgba(0,0,0,0)`

Valor hexadecimal

Fondo

Los colores CSS también se pueden representar como un triplete hexagonal, donde 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 `0` a `255` en notación decimal. Se pueden usar

valores hexadecimales en mayúsculas y / o en minúsculas (es decir, #3fc = #3FC = #33ffcc). El navegador interpreta #369 como #336699 . Si eso no es lo que deseaba, sino que deseaba #306090 , debe especificarlo explícitamente.

El número total de colores que se pueden representar con notación hexadecimal es 256^3 o 16,777,216.

Sintaxis

```
color: #rrggbb;  
color: #rgb
```

Valor	Descripción
rr	00 - FF por la cantidad de rojo
gg	00 - FF por la cantidad de verde
bb	00 - FF por la cantidad de azul

```
.some-class {  
    /* This is equivalent to using the color keyword 'blue' */  
    color: #0000FF;  
}  
  
.also-blue {  
    /* If you want to specify each range value with a single number, you can!  
       This is equivalent to '#0000FF' (and 'blue') */  
    color: #00F;  
}
```

La notación hexadecimal se utiliza para especificar valores de color en el formato de color RGB, según los '[Valores numéricos de color](#)' de W3C .

Hay muchas herramientas disponibles en Internet para buscar valores de color hexadecimales (o simplemente hexadecimales).

¡Busca "paleta de colores hexadecimal" o "selector de color hexadecimal" con tu navegador web favorito para encontrar un montón de opciones!

Los valores hexadecimales siempre comienzan con un signo de número (#), tienen hasta seis "dígitos" de longitud y no distinguen entre mayúsculas y minúsculas: es decir, no les importa el uso de mayúsculas. #FFC125 y #ffc125 son del mismo color.

rgb () Notación

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-FF, que es equivalente a 0-255 en decimal y 0% -

100% en porcentajes.

```
.some-class {  
    /* Scalar RGB, equivalent to 'blue' */  
    color: rgb(0, 0, 255);  
}  
  
.also-blue {  
    /* Percentile RGB values */  
    color: rgb(0%, 0%, 100%);  
}
```

Sintaxis

```
rgb(<red>, <green>, <blue>)
```

Valor	Descripción
<red>	un número entero de 0 a 255 o un porcentaje de 0 a 100%
<green>	un número entero de 0 a 255 o un porcentaje de 0 a 100%
<blue>	un número entero de 0 a 255 o un porcentaje de 0 a 100%

hsl () Notación

HSL significa **tono** ("qué color"), **saturación** ("cuánto color") y **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%); /* Blue */  
}
```

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 el 0% está totalmente desaturado (escala de grises) y el 100% está completamente saturado (de color vivo)

Valor	Descripción
<lightness>	especificado en porcentaje donde 0% es completamente negro y 100% es completamente blanco

Notas

- Una saturación del 0% siempre produce un color en escala de grises; Cambiar el tono no tiene efecto.
- Una luminosidad del 0% siempre produce negro, y el 100% siempre produce blanco; Cambiar el tono o la saturación no tiene efecto.

color actual

`currentColor` devuelve el valor de color calculado del elemento actual.

Utilizar en el mismo elemento

Aquí `currentColor` se evalúa en rojo ya que la propiedad de `color` se establece en `red`:

```
div {
  color: red;
  border: 5px solid currentColor;
  box-shadow: 0 0 5px currentColor;
}
```

En este caso, la especificación de `currentColor` para el borde es probablemente redundante porque omitirlo debería producir resultados idénticos. Solo use `currentColor` dentro de la propiedad de borde dentro del mismo elemento si de lo contrario se sobrescribiría debido a un selector [más específico](#).

Ya que es el color computado, el borde será verde en el siguiente ejemplo debido a que la segunda regla anula la primera:

```
div {
  color: blue;
  border: 3px solid currentColor;
  color: green;
}
```

Heredado del elemento padre

El color del parent se hereda, aquí `currentColor` se evalúa como 'azul', haciendo que el color del borde del elemento hijo sea azul.

```
.parent-class {
```

```

        color: blue;
    }

.parent-class .child-class {
    border-color:currentColor;
}

```

currentColor también puede ser usado por otras reglas que normalmente no heredarían de la propiedad de color, como el color de fondo. El siguiente ejemplo muestra a los hijos utilizando el conjunto de colores en el padre como fondo:

```

.parent-class {
    color: blue;
}

.parent-class .child-class {
    background-color:currentColor;
}

```

Resultado posible:



rgba () Notación

Similar a la [notación rgb \(\)](#), pero con un valor alfa (opacidad) adicional.

```

.red {
    /* Opaque red */
    color: rgba(255, 0, 0, 1);
}

.red-50p {
    /* Half-translucent red. */
    color: rgba(255, 0, 0, .5);
}

```

Sintaxis

```
rgba(<red>, <green>, <blue>, <alpha>);
```

Valor	Descripción
<red>	un número entero de 0 a 255 o un porcentaje de 0 a 100%
<green>	un número entero de 0 a 255 o un porcentaje de 0 a 100%
<blue>	un número entero de 0 a 255 o un porcentaje de 0 a 100%
<alpha>	un número del 0 al 1, donde 0.0 es totalmente transparente y 1.0 es totalmente opaco

hsla () Notación

Similar a la [notación hsl \(\)](#), pero con un valor alfa (opacidad) agregado.

```
hsla(240, 100%, 50%, 0)      /* transparent */  
hsla(240, 100%, 50%, 0.5)    /* half-translucent blue */  
hsla(240, 100%, 50%, 1)      /* fully opaque blue */
```

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 el 0% está totalmente desaturado (escala de grises) y el 100% está completamente saturado (de color vivo)
<lightness>	porcentaje donde 0% es completamente negro y 100% es completamente blanco
<alpha>	un número del 0 al 1 donde 0 es completamente transparente y 1 es completamente opaco

Lea Colores en línea: <https://riptutorial.com/es/css/topic/644/colores>

Capítulo 10: Columnas

Sintaxis

- recuento de columnas: auto | número | heredar | inicial | no establecido;
- ancho de columna: auto | longitud;
- columna: [ancho de columna] | [recuento de columnas];
- intervalo de columnas: ninguno | todos | heredar | inicial | no establecido;
- espacio entre columnas: normal | longitud | heredar | inicial | no establecido;
- relleno de columna: auto | balance | inherit | intial | unset;
- columna-regla-color: color | inherit | initial | unset;
- columna-regla-estilo: ninguno | oculto | punteado | punteado | sólido | doble | surco | cresta | inserción | comienzo | heredado | inicial | unset;
- columna-regla-ancho: delgado | medio | grueso | largo | heredado | inicial | no establecido;
- column-rule: [column-rule-width] | [columm-rule-style] | [column-rule-color];
- break-after: auto | siempre | izquierda | derecha | recto | verso | página | columna | región | evitar | evitar-página | evitar-columna | evitar-región;
- break-before: auto | siempre | izquierda | derecha | recto | verso | página | columna | región | evitar | evitar-página | evitar-columna | evitar-región;
- break-inside: auto | Avoid | Avoid-page | Avoid-column | Avoid-region;

Examples

Ejemplo simple (conteo de columnas)

El diseño de varias columnas de CSS facilita la creación de varias columnas de texto.

Código

```
<div id="multi-columns">Lorem ipsum dolor sit amet, consectetur adipisicing 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-columns {  
    -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 and answers wiki or Di Overflow "badges" awarded receiving given to a badges for [14] which gamification or forum. licensed Attribute-

Ancho de columna

La propiedad de `column-width` establece el ancho de columna mínimo. Si no se define el `column-count` el navegador creará tantas columnas como se ajusten al ancho disponible.

Código:

```
<div id="multi-columns">
    Lorem ipsum dolor sit amet, consectetur adipisicing 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-columns {
    -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.	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]	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 Attribution-ShareAlike license.
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------

Lea Columnas en línea: <https://riptutorial.com/es/css/topic/3042/columnas>

Capítulo 11: Columnas multiples

Introducción

CSS permite definir el contenido de ese elemento envuelto en varias columnas con espacios y reglas entre ellos.

Observaciones

El Nivel 1 del Módulo de diseño de varias columnas de CSS es, a partir del 12 de abril de 2011, una Recomendación del candidato del W3C. Desde entonces, se hicieron algunos cambios más pequeños . Se considera que está en la etapa Estable .

A partir del 3 de julio de 2017, los navegadores Internet Explorer 10 y 11 y Edge de Microsoft solo admiten una versión anterior de la especificación utilizando un prefijo de proveedor.

Examples

Ejemplo basico

Considere el siguiente marcado 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;
}
```

Vea una muestra en vivo de esto en [JSFiddle](#) .

Crear múltiples columnas

```
<div class="content">
Lorem ipsum dolor sit amet, consectetuer 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 duis 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

```
.content {
-webkit-column-count: 3; /* Chrome, Safari, Opera */
-moz-column-count: 3; /* Firefox */
column-count: 3;
}
```

Lea Columnas multiples en línea: <https://riptutorial.com/es/css/topic/10688/columnas-multiples>

Capítulo 12: Comentarios

Sintaxis

- /* Comentario */

Observaciones

- Los comentarios en CSS siempre comienzan con /* y terminan con */
- Los comentarios no pueden ser anidados

Examples

Línea sola

```
/* This is a CSS comment */
div {
    color: red; /* This is a CSS comment */
}
```

Línea múltiple

```
/*
    This
    is
    a
    CSS
    comment
*/
div {
    color: red;
}
```

Lea Comentarios en línea: <https://riptutorial.com/es/css/topic/1625/comentarios>

Capítulo 13: Consultas de características

Sintaxis

- @supports [condición] {/* reglas CSS para aplicar */}

Parámetros

Parámetro	Detalles
(property: value)	Evaluá verdadero si el navegador puede manejar la regla CSS. Se requieren los paréntesis alrededor de la regla.
and	Devuelve verdadero solo si las condiciones anteriores y siguientes son verdaderas.
not	Niega la siguiente condición.
or	Devuelve verdadero si la condición anterior o siguiente es verdadera.
(...)	Condiciones de los grupos

Observaciones

La detección de `@supports` mediante `@supports` se admite en Edge, Chrome, Firefox, Opera y Safari 9 y superiores.

Examples

Uso básico de los soportes

```
@supports (display: flex) {  
  /* Flexbox is available, so use it */  
  .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)`.

Encadenamiento de detecciones de características.

Para detectar múltiples funciones a la vez, use el operador `and`.

```
@supports (transform: translateZ(1px)) and (transform-style: preserve-3d) and (perspective: 1px) {  
  /* Probably do some fancy 3d stuff here */  
}
```

También hay un operador `or` y un operador:

```
@supports (display: flex) or (display: table-cell) {  
  /* Will be used if the browser supports flexbox or display: table-cell */  
}  
@supports not (-webkit-transform: translate(0, 0, 0)) {  
  /* Will *not* be used if the browser supports -webkit-transform: translate(...) */  
}
```

Para la mejor experiencia de `@supports`, intente 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. `display: block` **sopportes** `display: block Y zoom: 1 , 0`
2. `display: flex` **Y NO** `display: table-cell , 0`
3. **Soporta la** `transform: translateX(1px)` .

Lea Consultas de características en línea: <https://riptutorial.com/es/css/topic/5024/consultas-de-caracteristicas>

Capítulo 14: Contadores

Sintaxis

- conjunto de contadores: [<nombre_contador> <integer>?] + | ninguna
- counter-reset: [<counter-name> <integer>?] + | ninguna
- contra-incremento: [<counter-name> <integer>?] + | ninguna
- contador (<counter-name> [, <counter-style>]?)
- contadores (<counter-name>, <connector-string> [, <counter-style>]?)

Parámetros

Parámetro	Detalles
nombre de contador	Este es el nombre del contador que debe crearse, incrementarse o imprimirse. Puede ser cualquier nombre personalizado que el desarrollador desee.
entero	Este número entero es un valor opcional que, cuando se proporciona junto al nombre del contador, representará el valor inicial del contador (en propiedades de <code>counter-set counter-reset</code>) o el valor por el cual el contador debe incrementarse (en incremento de <code>counter-increment</code>).
ninguna	Este es el valor inicial para las 3 propiedades de <code>counter-*</code> . Cuando este valor se utiliza para <code>counter-increment</code> de <code>counter-increment</code> , el valor de ninguno de los contadores se ve afectado. Cuando esto se usa para los otros dos, no se crea ningún contador.
contra-estilo	Esto especifica el estilo en el que se debe mostrar el valor del contador. Admite todos los valores admitidos por la propiedad de <code>list-style-type</code> . Si <code>none</code> se utiliza <code>none</code> , el valor del contador no se imprime en absoluto.
cuerda conectora	Esto representa la cadena que debe colocarse entre los valores de dos niveles de contador diferentes (como el "." En "2.1.1").

Observaciones

Los contadores no son un tema nuevo en CSS. Formaba parte de las Especificaciones de Nivel 2 de CSS (para ser más precisas, la Revisión 1) y por lo tanto tiene un soporte de navegador muy alto.

Todos los navegadores, excepto IE6 e IE7, son compatibles con los contadores de CSS.

Examples

Aplicando un estilo de números romanos a la salida del contador.

CSS

```
body {  
    counter-reset: item-counter;  
}  
  
.item {  
    counter-increment: item-counter;  
}  
  
.item:before {  
    content: counter(item-counter, upper-roman) ". "; /* by specifying the upper-roman as style  
the output would be in roman numbers */  
}
```

HTML

```
<div class='item'>Item No: 1</div>  
<div class='item'>Item No: 2</div>  
<div class='item'>Item No: 3</div>
```

En el ejemplo anterior, la salida del contador se mostraría como I, II, III (números romanos) en lugar del habitual 1, 2, 3, ya que el desarrollador ha especificado explícitamente el estilo del contador.

Numera cada ítem usando el Contador CSS

CSS

```
body {  
    counter-reset: item-counter; /* create the counter */  
}  
  
.item {  
    counter-increment: item-counter; /* increment the counter every time an element with class  
"item" is encountered */  
}  
  
.item-header:before {  
    content: counter(item-counter) ". "; /* print the value of the counter before the header and  
append a "." to it */  
}  
  
/* just for demo */
```

```
.item {  
    border: 1px solid;  
    height: 100px;  
    margin-bottom: 10px;  
}  
.item-header {  
    border-bottom: 1px solid;  
    height: 40px;  
    line-height: 40px;  
    padding: 5px;  
}  
.item-content {  
    padding: 8px;  
}
```

HTML

```
<div class='item'>  
    <div class='item-header'>Item 1 Header</div>  
    <div class='item-content'>Lorem Ipsum Dolor Sit Amet....</div>  
</div>  
<div class='item'>  
    <div class='item-header'>Item 2 Header</div>  
    <div class='item-content'>Lorem Ipsum Dolor Sit Amet....</div>  
</div>  
<div class='item'>  
    <div class='item-header'>Item 3 Header</div>  
    <div class='item-content'>Lorem Ipsum Dolor Sit Amet....</div>  
</div>
```

El ejemplo anterior numera cada "elemento" en la página y agrega el número del elemento antes de su encabezado (usando la propiedad de `content` de `.item-header` element's `:before` pseudo). Una demostración en vivo de este código está disponible [aquí](#).

Implementación de numeración multinivel utilizando contadores CSS.

CSS

```
ul {  
    list-style: none;  
    counter-reset: list-item-number; /* self nesting counter as name is same for all levels */  
}  
li {  
    counter-increment: list-item-number;  
}  
li:before {  
    content: counters(list-item-number, ".") " "; /* usage of counters() function means value of  
    counters at all higher levels are combined before printing */  
}
```

HTML

```
<ul>
  <li>Level 1
    <ul>
      <li>Level 1.1
        <ul>
          <li>Level 1.1.1</li>
        </ul>
      </li>
    </ul>
  </li>
<li>Level 2
  <ul>
    <li>Level 2.1
      <ul>
        <li>Level 2.1.1</li>
        <li>Level 2.1.2</li>
      </ul>
    </li>
  </ul>
</li>
<li>Level 3</li>
</ul>
```

Lo anterior es un ejemplo de numeración multinivel utilizando contadores de CSS. Hace uso del concepto de **auto-anidamiento** de los contadores. Self nesting es un concepto en el que si un elemento ya tiene un contador con el nombre dado pero tiene que crear otro, lo crea como un elemento secundario del contador existente. Aquí, el segundo nivel `ul` ya hereda el contador de `list-item-number` de su padre, pero luego tiene que crear su propio `list-item-number` (para sus hijos `li`) y así crea el `list-item-number[1]` (contador para segundo nivel) y lo anida bajo el `list-item-number[0]` (contador para el primer nivel). De este modo se consigue la numeración multinivel.

La salida se imprime utilizando la función `counters()` lugar de la función `counter()` porque la función `counters()` está diseñada para prefijar el valor de todos los contadores de nivel superior (padre) al imprimir la salida.

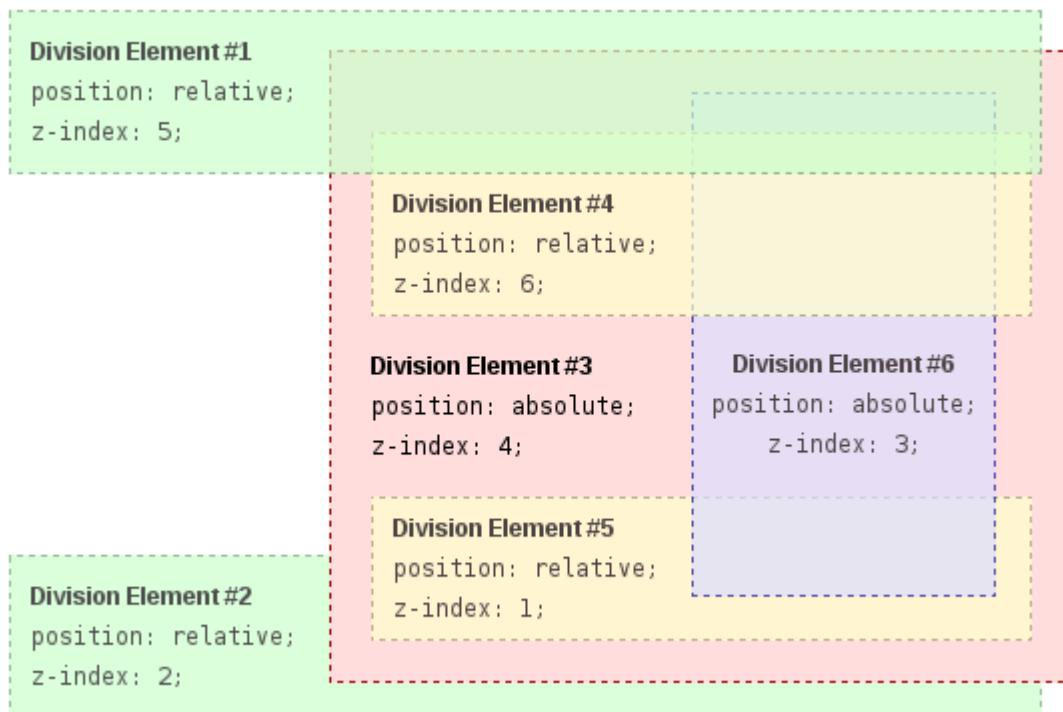
Lea Contadores en línea: <https://riptutorial.com/es/css/topic/2575/contadores>

Capítulo 15: Contexto de apilamiento

Examples

Contexto de apilamiento

En este ejemplo, cada elemento posicionado crea su propio contexto de apilamiento, debido a su posicionamiento y valores de índice z. La jerarquía de los contextos de apilamiento se organiza de la siguiente manera:



- Raíz
 - DIV # 1
 - DIV # 2
 - DIV # 3
 - DIV # 4
 - DIV # 5
 - DIV # 6

Es importante tener en cuenta que DIV # 4, DIV # 5 y DIV # 6 son hijos de DIV # 3, por lo que el apilamiento de esos elementos se resuelve completamente dentro de DIV # 3. Una vez que se completa el apilamiento y la representación dentro de DIV # 3, se pasa todo el elemento DIV # 3 para apilar en el elemento raíz con respecto al DIV de su hermano.

HTML:

```
<div id="div1">
  <h1>Division Element #1</h1>
```

```

<code>position: relative;<br/>
z-index: 5;</code>
</div>
<div id="div2">
    <h1>Division Element #2</h1>
    <code>position: relative;<br/>
z-index: 2;</code>
</div>
<div id="div3">
    <div id="div4">
        <h1>Division Element #4</h1>
        <code>position: relative;<br/>
z-index: 6;</code>
    </div>
    <h1>Division Element #3</h1>
    <code>position: absolute;<br/>
z-index: 4;</code>
    <div id="div5">
        <h1>Division Element #5</h1>
        <code>position: relative;<br/>
z-index: 1;</code>
    </div>
    <div id="div6">
        <h1>Division Element #6</h1>
        <code>position: absolute;<br/>
z-index: 3;</code>
    </div>
</div>

```

CSS:

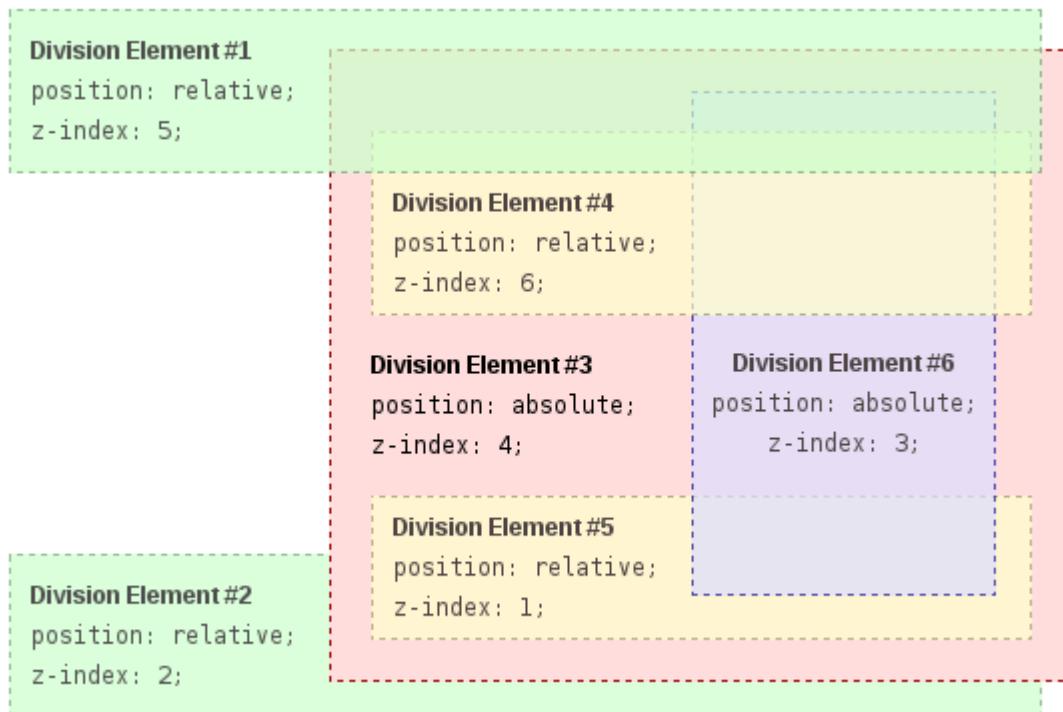
```

* {
    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/en-US/docs/Web/CSS/CSS_Positioning/Understanding_z_index/The_stacking_context.

Lea Contexto de apilamiento en línea: <https://riptutorial.com/es/css/topic/5037/contexto-de-apilamiento>

Capítulo 16: Contextos de formato de bloque

Observaciones

[Un contexto de formato de bloque es parte de una representación visual de CSS de una página web. Es la región en la que se produce el diseño de las cajas de bloques y en la que los flotadores interactúan entre sí.] [1]

[1]: https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Block_formatting_context MDN

Examples

Usando la propiedad de desbordamiento con un valor diferente a visible

```
img{  
  float:left;  
  width:100px;  
  margin:0 10px;  
}  
.div1{  
  background:#f1f1f1;  
  /* does not create block formatting context */  
}  
.div2{  
  background:#f1f1f1;  
  overflow:hidden;  
  /* creates block formatting context */  
}
```

```

1 
2 <div class=div1>
3 <p>Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia. Eam in velit graecis, sea mucius 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 instructior usu ne. At ludus suscipit disputationi vel.</p>
4
5 <p>Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitibus 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 mucius 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 instructior usu ne. At ludus suscipit disputationi vel.</p>
11
12 <p>Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitibus 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>

```

```

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 c
13 }

```



Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia. Eam in velit graecis, sea mucius 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 instructior usu ne. At ludus suscipit disputationi vel.



Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitibus sea. Cu eos munere virtute vituperata. Exerci bonorum sed id, id nec tantas praesent complectitur. Vel cu legendos mediocritatem. Enim liberavisse ei sea.

<https://jsfiddle.net/MadalinaTn/qkwwmu6m/2/>

El uso de la propiedad de desbordamiento con un valor diferente a visible (su valor predeterminado) creará un nuevo contexto de formato de bloque. Esto es técnicamente necesario: si un flotador se intersecaba con el elemento de desplazamiento, volvería a envolver el contenido a la fuerza.

Este ejemplo que muestra cómo interactúan varios párrafos con una imagen flotada es similar a [este ejemplo](#), en [css-tricks.com](#).

2 : <https://developer.mozilla.org/en-US/docs/Web/CSS/overflow> MDN

Lea Contextos de formato de bloque en línea: <https://riptutorial.com/es/css/topic/5069/contextos-de-formato-de-bloque>

Capítulo 17: Contornos

Sintaxis

- contorno: contorno de color contorno de estilo contorno de ancho | inicial | heredar;
- ancho del contorno: medio | delgado | grueso | longitud | inicial | heredar;
- esquema de estilo: ninguno | oculto | punteado | sólido doble | surco cresta inserción inicio | inicial | heredar;

Parámetros

Parámetro	Detalles
punteado	contorno punteado
precipitado	contorno discontinuo
sólido	contorno sólido
doble	doble contorno
ranura	Contorno ranurado 3D, depende del valor del contorno del color.
cresta	Esquema acanalado 3D, depende del valor del color del contorno
recuadro	Contorno 3D inserto, depende del valor del contorno-color.
comienzo	Contorno de inicio 3D, depende del valor del contorno del color.
ninguna	sin contorno
oculto	contorno oculto

Observaciones

`outline` ahora se describe en la [IU básica](#), un módulo CSS de nivel 3 (ya se describió en REC CSS2.1)

La propiedad de esquema se define de forma predeterminada en los navegadores para elementos `:focus` en el estado de `:focus`.

No debe eliminarse, consulte <http://outlinenone.com> que indica:

¿Qué hace la propiedad de esquema?

Proporciona comentarios visuales para enlaces que tienen "enfoque" cuando navega

por un documento web usando la tecla TAB (o equivalente). Esto es especialmente útil para las personas que no pueden usar un mouse o tener una discapacidad visual. Si elimina el esquema, está haciendo que su sitio sea inaccesible para estas personas.
(...)

Ejemplos relacionados interesantes sobre desbordamiento de pila:

- [Cómo eliminar el resaltado de borde en un elemento de texto de entrada](#)
- [¿Cómo eliminar el contorno de puntos de Firefox en los BOTONES y los enlaces?](#)

Examples

Visión general

El contorno es una línea que gira alrededor del elemento, fuera del borde. A diferencia de los `border`, los contornos no ocupan espacio en el modelo de cuadro. Por lo tanto, agregar un esquema a un elemento no afecta la posición del elemento u otros elementos.

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

Las partes esenciales del `outline` son `outline-color` `outline-style` `outline-width`.

La definición de un esquema 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 de la frontera.

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

esquema de estilo

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

```
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;  
}
```

HTML

```
<p class="p1">A dotted outline</p>  
<p class="p2">A dashed outline</p>  
<p class="p3">A solid outline</p>  
<p class="p4">A double outline</p>  
<p class="p5">A groove outline</p>  
<p class="p6">A ridge outline</p>  
<p class="p7">An inset outline</p>  
<p class="p8">An outset outline</p>
```

A dotted outline

A dashed outline

A solid outline

A double outline

A groove outline

A ridge outline

An inset outline

An outset outline

Lea Contornos en línea: <https://riptutorial.com/es/css/topic/4258/contornos>

Capítulo 18: Control de diseño

Sintaxis

- pantalla: ninguna | en línea | bloque lista-item | inline-list-item | bloque en linea | mesa en linea | mesa | mesa-celda | tabla-columna | tabla-columna-grupo | grupo de pies de mesa | table-header-group | tabla-fila | tabla-fila-grupo | flexionar inline-flex | rejilla inline-grid | run-in | rubí | base de rubí | texto de rubí | rubí-base-contenedor | ruby-text-container | contenido;

Parámetros

Valor	Efecto
none	Oculta el elemento y evita que ocupe espacio.
block	Elemento de bloque, ocupa el 100% del ancho disponible, rompe después del elemento.
inline	Elemento en línea, no ocupa ancho, no hay ruptura después del elemento.
inline-block	Tomando propiedades especiales de los elementos en línea y de bloque, no hay ruptura, pero puede tener ancho.
inline-flex	Muestra un elemento como un contenedor flexible en el nivel en línea.
inline-table	El elemento se muestra como una tabla de nivel en línea.
grid	Se comporta como un elemento de bloque y establece su contenido de acuerdo con el modelo de cuadrícula.
flex	Se comporta como un elemento de bloque y establece su contenido de acuerdo con el modelo flexbox.
inherit	Hereda el valor del elemento padre.
initial	Restablezca el valor al valor predeterminado tomado de los comportamientos descritos en las especificaciones HTML o de la hoja de estilo predeterminada del usuario / navegador.
table	Se comporta como el elemento de <code>table</code> HTML.
table-cell	Deja que el elemento se comporte como un elemento <code><td></code>
table-column	Deja que el elemento se comporte como un elemento <code><col></code>

Valor	Efecto
table-row	Deja que el elemento se comporte como un elemento <tr>
list-item	Deje que el elemento se comporte como un elemento .

Examples

La propiedad de visualización

La propiedad CSS de `display` es fundamental para controlar el diseño y el flujo de un documento HTML. La mayoría de los elementos tienen un valor de `display` predeterminado de `block` o en `inline` (aunque algunos elementos tienen otros valores predeterminados).

En línea

Un elemento en `inline` ocupa solo el ancho necesario. Se apila horizontalmente con otros elementos del mismo tipo y puede que no contenga otros elementos no en línea.

```
<span>This is some <b>bolded</b> text!</span>
```

This is some **bolded** text!

Como se demostró anteriormente, dos elementos en `inline`, `` y ``, están en línea (de ahí el nombre) y no interrumpen el flujo del texto.

Bloquear

Un elemento de `block` ocupa el ancho máximo disponible de su elemento principal. Comienza con una nueva línea y, a diferencia de los elementos en `inline`, no restringe el tipo de elementos que puede contener.

```
<div>Hello world!</div><div>This is an example!</div>
```

Hello world!

This is an example!

El elemento `div` está a nivel de bloque por defecto, y como se muestra arriba, los dos elementos de `block` están apilados verticalmente y, a diferencia de los elementos en `inline`, el flujo del texto se interrumpe.

Bloque en linea

El valor del `inline-block` nos da lo mejor de ambos mundos: combina el elemento con el flujo del texto y nos permite usar `padding`, `margin`, `height` y propiedades similares que no tienen efecto visible en los elementos en `inline`.

Los elementos con este valor de visualización actúan como si fueran texto regular y, como resultado, 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: unordered list-->
<style>
li {
    display : inline;
    background : lightblue;
    padding:10px;

    border-width:2px;
    border-color:black;
    border-style:solid;
}
</style>

<ul>
<li>First Element </li>
<li>Second Element </li>
<li>Third Element </li>
</ul>
```

First Element	Second Element	Third Element
---------------	----------------	---------------

```
<!--block: unordered list-->
<style>
li {
    display : block;
    background : lightblue;
    padding:10px;

    border-width:2px;
    border-color:black;
    border-style:solid;
}
</style>

<ul>
<li>First Element </li>
<li>Second Element </li>
<li>Third Element </li>
</ul>
```

First Element

Second Element

Third Element

```
<!--Inline-block: unordered list-->
<style>
li {
    display : inline-block;
    background : lightblue;
    padding:10px;

    border-width:2px;
    border-color:black;
    border-style:solid;
}

</style>

<ul>
<li>First Element </li>
<li>Second Element </li>
<li>Third Element </li>
</ul>
```

First Element

Second Element

Third Element

ninguna

Un elemento al que se le asigna el valor `none` a su propiedad de visualización no se mostrará en absoluto.

Por ejemplo, vamos a crear un elemento div que tenga un ID de `myDiv`:

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

Esto ahora se puede marcar como no mostrado por la siguiente regla CSS:

```
#myDiv {
    display: none;
}
```

Cuando un elemento se ha configurado para `display:none`, el navegador ignora todas las demás propiedades de diseño para ese elemento específico (`position` y `float`). No se representará ningún cuadro para ese elemento y su existencia en html no afecta la posición de los siguientes elementos.

Tenga en cuenta que esto es diferente de establecer la propiedad de `visibility` en `hidden`. Configuración de `visibility: hidden;` para un elemento no se mostraría el elemento en la página, pero el elemento todavía ocuparía 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` para la propiedad de visualización se usa comúnmente junto con JavaScript para mostrar u ocultar elementos a voluntad, eliminando la necesidad de eliminarlos y volver a crearlos.

Para obtener la estructura de la tabla de edad utilizando div

Esta es la estructura de tabla HTML normal

```
<style>
  table {
    width: 100%;
  }
</style>

<table>
  <tr>
    <td>
      I'm a table
    </td>
  </tr>
</table>
```

Puedes hacer la misma implementación como esta

```
<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">
      I behave like a table now
    </div>
  </div>
</div>
```

Lea Control de diseño en línea: <https://riptutorial.com/es/css/topic/1473/control-de-diseno>

Capítulo 19: Cuadrícula

Introducción

Grid layout es un nuevo y poderoso sistema de diseño CSS que permite dividir el contenido de una página web en filas y columnas de una manera fácil.

Observaciones

El nivel 1 del módulo de diseño de cuadrícula de CSS es, a partir del 9 de septiembre de 2016, una recomendación de candidato del W3C. Se considera que está en la etapa de prueba (<https://www.w3.org/Style/CSS/current-work>) .

A partir del 3 de julio de 2017, los navegadores Internet Explorer 10 y 11 y Edge de Microsoft solo admiten una versión anterior de la especificación utilizando un prefijo de proveedor.

Examples

Ejemplo básico

Propiedad	Valores posibles
monitor	rejilla / rejilla en línea

La cuadrícula de CSS se define como una propiedad de visualización. Se aplica a un elemento padre y sus hijos inmediatos solamente.

Considere el siguiente marcado:

```
<section class="container">
  <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 de `display` en `grid`:

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

Sin embargo, hacer esto invariablemente hará que todos los elementos secundarios se colapsen uno encima del otro. Esto se debe a que los niños actualmente no saben cómo posicionarse dentro de la cuadrícula. Pero podemos decírselo explícitamente.

Primero debemos indicar al elemento de la cuadrícula `.container` cuántas filas y columnas conformarán su estructura, y podemos hacerlo utilizando las propiedades de `grid-columns` y `grid-rows` (observe la pluralización):

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

Sin embargo, eso todavía no nos ayuda mucho porque necesitamos dar una orden a cada elemento secundario. Podemos hacer esto especificando los valores de la `grid-row` `grid-column` que le indicarán dónde se encuentra en la cuadrícula:

```
.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;  
}
```

Al dar a cada artículo un valor de columna y fila, identifica el orden de los artículos dentro del contenedor.

Ver un ejemplo de trabajo en [JSFiddle](#) . Necesitará ver esto en IE10, IE11 o Edge para que funcione, ya que estos son actualmente los únicos navegadores que admiten Grid Layout (con el prefijo del proveedor `-ms-`) o habilitar una bandera en Chrome, Opera y Firefox de acuerdo con el [uso](#) en orden para probar con ellos.

Lea Cuadrícula en línea: <https://riptutorial.com/es/css/topic/2152/cuadricula>

Capítulo 20: Diseño de caja flexible (Flexbox)

Introducción

El módulo de la caja flexible, o simplemente 'flexbox' para abreviar, es un modelo de caja diseñado para las interfaces de usuario, y permite a los usuarios alinear y distribuir el espacio entre los elementos en un contenedor de manera tal que los elementos se comporten de manera predecible cuando el diseño de la página debe adaptarse a diferentes elementos desconocidos. Tamaños de pantalla. Un contenedor flexible expande los elementos para llenar el espacio disponible y los reduce para evitar el desbordamiento.

Sintaxis

- pantalla: flexión;
- dirección flexible: fila | fila inversa | columna | columna inversa
- flex-wrap: nowrap | envolver | envolver-revertir
- flex-flow: <'flex-direction'> || <'flex-wrap'>
- justify-content: flex-start | extremo flexible | centro | espacio intermedio | espacio alrededor
- elementos de alineación: inicio flexible | extremo flexible | centro | línea de base | tramo;
- align-content: flex-start | extremo flexible | centro | espacio intermedio | espacio-alrededor | tramo;
- orden: <integer>;
- flex-grow: <número>; /* por defecto 0 */
- flexión retráctil: <número>; /* por defecto 1 */
- base flexible: <length> | auto; /* predeterminado automático */
- flex: ninguno | [<'flex-grow'> <'flex-shrink'>? || <'base flexible'>]
- align-self: auto | flex-start | extremo flexible | centro | línea de base | tramo;

Observaciones

Prefijos de Vender

- pantalla: -webkit-box; /* Chrome <20 */
- pantalla: -webkit-flex; /* Chrome 20+ */
- pantalla: -moczona; /* Firefox */
- pantalla: -ms-flexbox; /* IE */
- pantalla: flexión; /* Los navegadores modernos */

Recursos

- [Una guía completa para Flexbox](#)
- [Resuelto por flexbox](#)

- [¿Qué es el Flexbox?](#)
- [Flexbox en 5 minutos](#)
- [Flexbugs](#)

Examples

Pie de página variable pegajoso

Este código crea un pie de página pegajoso. Cuando el contenido no llega al final de la ventana gráfica, el pie de página se pega a la parte inferior de la ventana gráfica. Cuando el contenido se extiende más allá de la parte inferior de la ventana gráfica, el pie de página también se elimina de la ventana gráfica. [Ver resultado](#)

HTML:

```
<div class="header">
  <h2>Header</h2>
</div>

<div class="content">
  <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. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero.</p>
</div>

<div class="footer">
  <h4>Footer</h4>
</div>
```

CSS:

```
html, body {
  height: 100%;
}

body {
  display: flex;
  flex-direction: column;
}

.content {
  /* Include `0 auto` for best browser compatibility. */
  flex: 1 0 auto;
}

.header, .footer {
  background-color: grey;
  color: white;
  flex: none;
}
```

Disposición del Santo Grial usando Flexbox

El diseño del Santo Grial es un diseño con un encabezado y pie de página de altura fija, y un centro con 3 columnas. Las 3 columnas incluyen un sidenav de ancho fijo, un centro fluido y una columna para otro contenido 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="container">
  <header class="header">Header</header>
  <div class="content-body">
    <main class="content">Content</main>
    <nav class="sidenav">Nav</nav>
    <aside class="ads">Ads</aside>
  </div>
  <footer class="footer">Footer</footer>
</div>
```

CSS:

```
body {
  margin: 0;
  padding: 0;
}

.container {
  display: flex;
  flex-direction: column;
  height: 100vh;
}

.header {
  flex: 0 0 50px;
}

.content-body {
  flex: 1 1 auto;

  display: flex;
  flex-direction: row;
}

.content-body .content {
  flex: 1 1 auto;
  overflow: auto;
}

.content-body .sidenav {
  order: -1;
  flex: 0 0 100px;
  overflow: auto;
}

.content-body .ads {
  flex: 0 0 100px;
  overflow: auto;
```

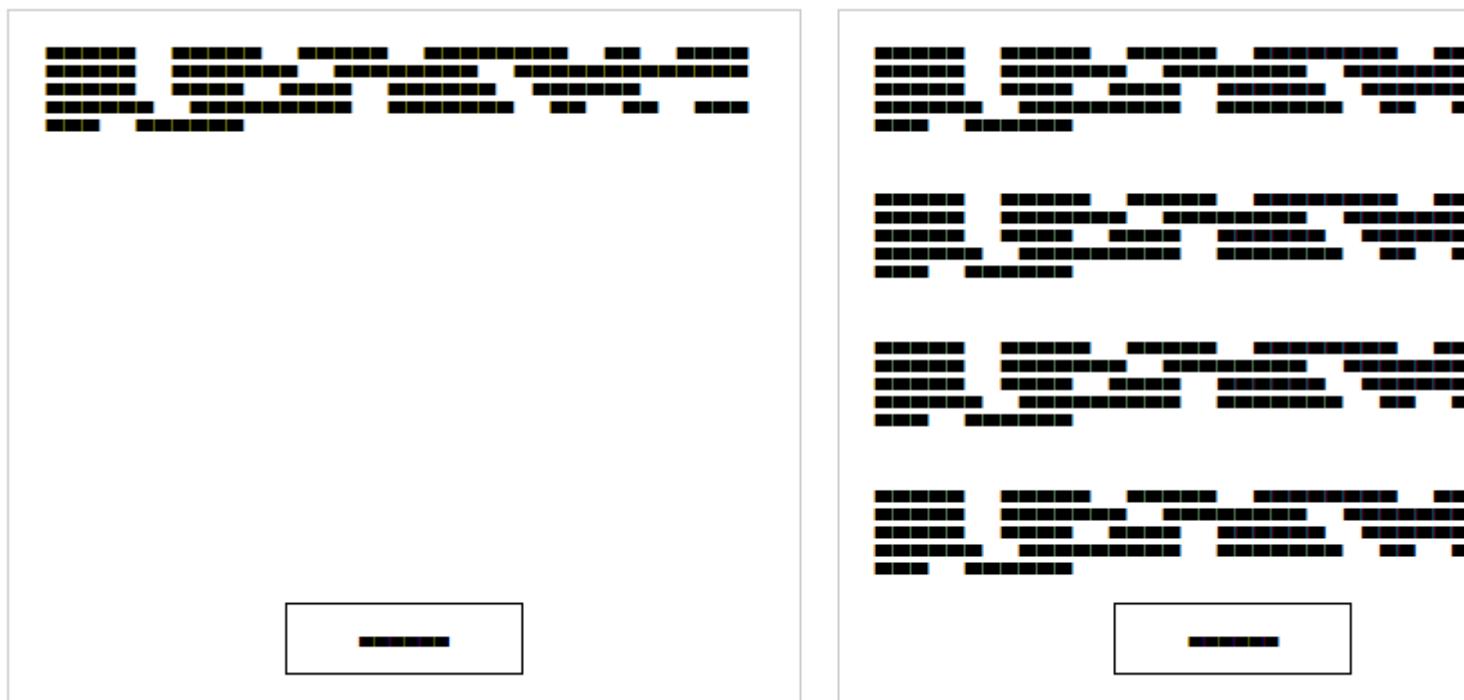
```
}

.footer {
  flex: 0 0 50px;
}
```

Manifestación

Botones perfectamente alineados dentro de tarjetas con flexbox

Es un patrón regular en el diseño en estos días para alinear verticalmente las **llamadas a las acciones** dentro de sus tarjetas que contienen como esto:



Esto se puede lograr utilizando un truco especial con `flexbox`

HTML

```
<div class="cards">
  <div class="card">
    <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>Action</button></p>
  </div>
  <div class="card">
    <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>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>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>Action</button></p>
```

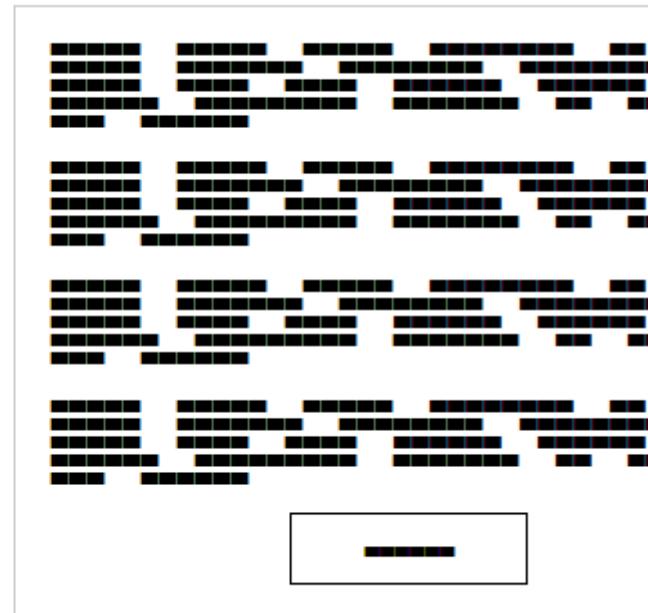
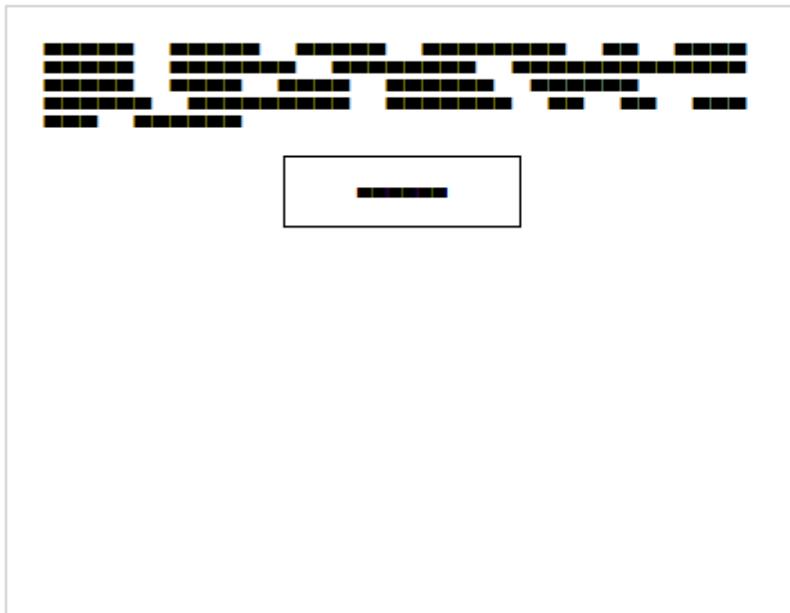
```
</div>  
</div>
```

En primer lugar, usamos CSS para aplicar `display: flex;` al contenedor. Esto creará 2 columnas iguales en altura con el contenido que fluye naturalmente dentro de él.

CSS

```
.cards {  
    display: flex;  
}  
.card {  
    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 se hará así:



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

CSS final:

```
.cards {
    display: flex;
}
.card {
    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;
}
```

Centrado dinámico vertical y horizontal (alinear elementos, justificar contenido)

Ejemplo simple (centrando un solo elemento)

HTML

```
<div class="aligner">
  <div class="aligner-item">...</div>
</div>
```

CSS

```
.aligner {
    display: flex;
    align-items: center;
    justify-content: center;
}

.aligner-item {
    max-width: 50%; /*for demo. Use actual width instead.*/
}
```

Aquí hay una [demo](#).

Razonamiento

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

Ejemplos de propiedad individual

Todos los estilos a continuación se aplican a este diseño simple:

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

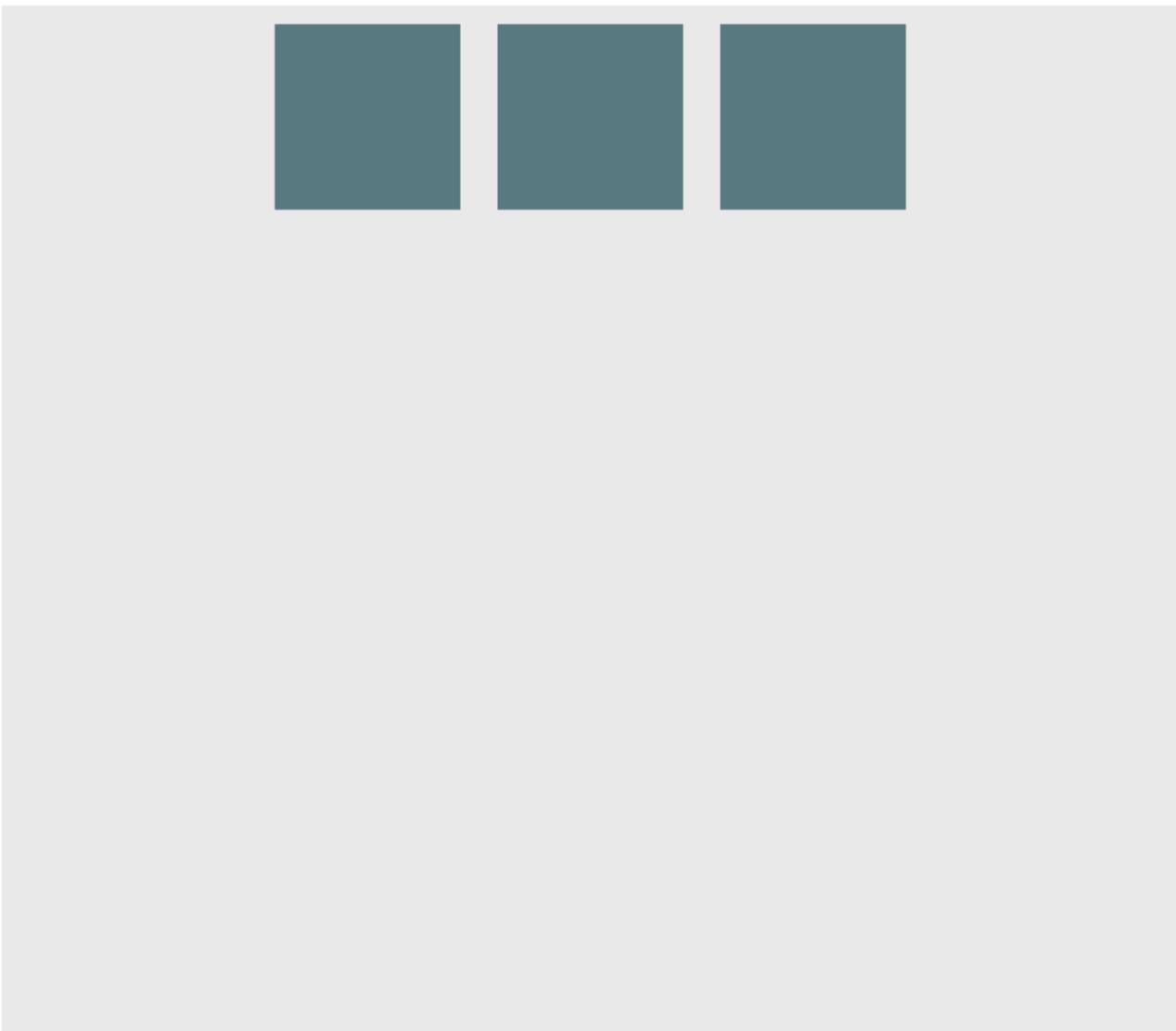
donde `#container` es la flex-box .

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

CSS:

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

Salir:



Aquí hay una [demo](#).

Ejemplo: justify-content: center en un flexbox vertical

CSS:

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

Salir:



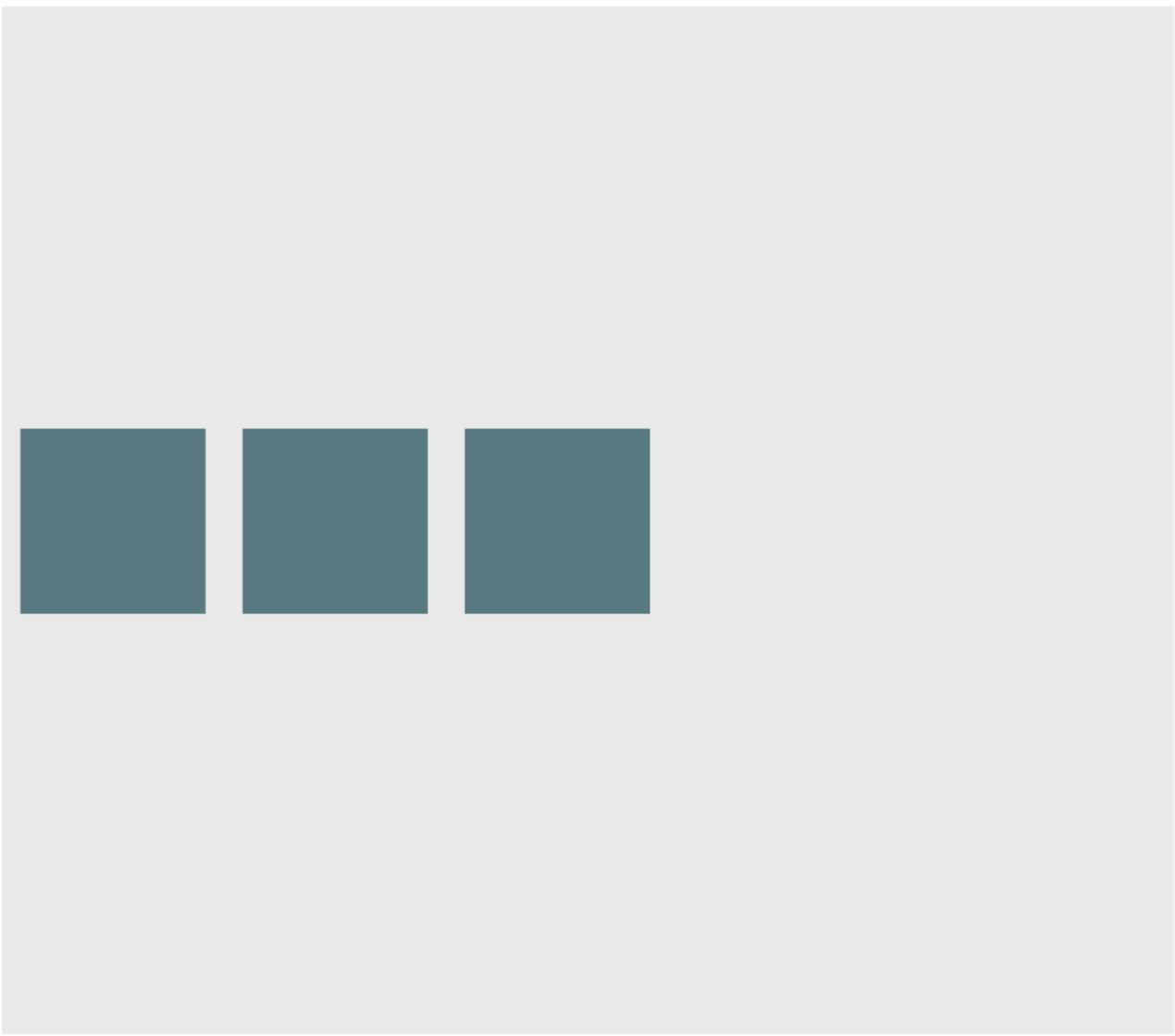
Aquí hay una [demo](#).

Ejemplo: align-content: center en un flexbox horizontal

CSS:

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

Salir:



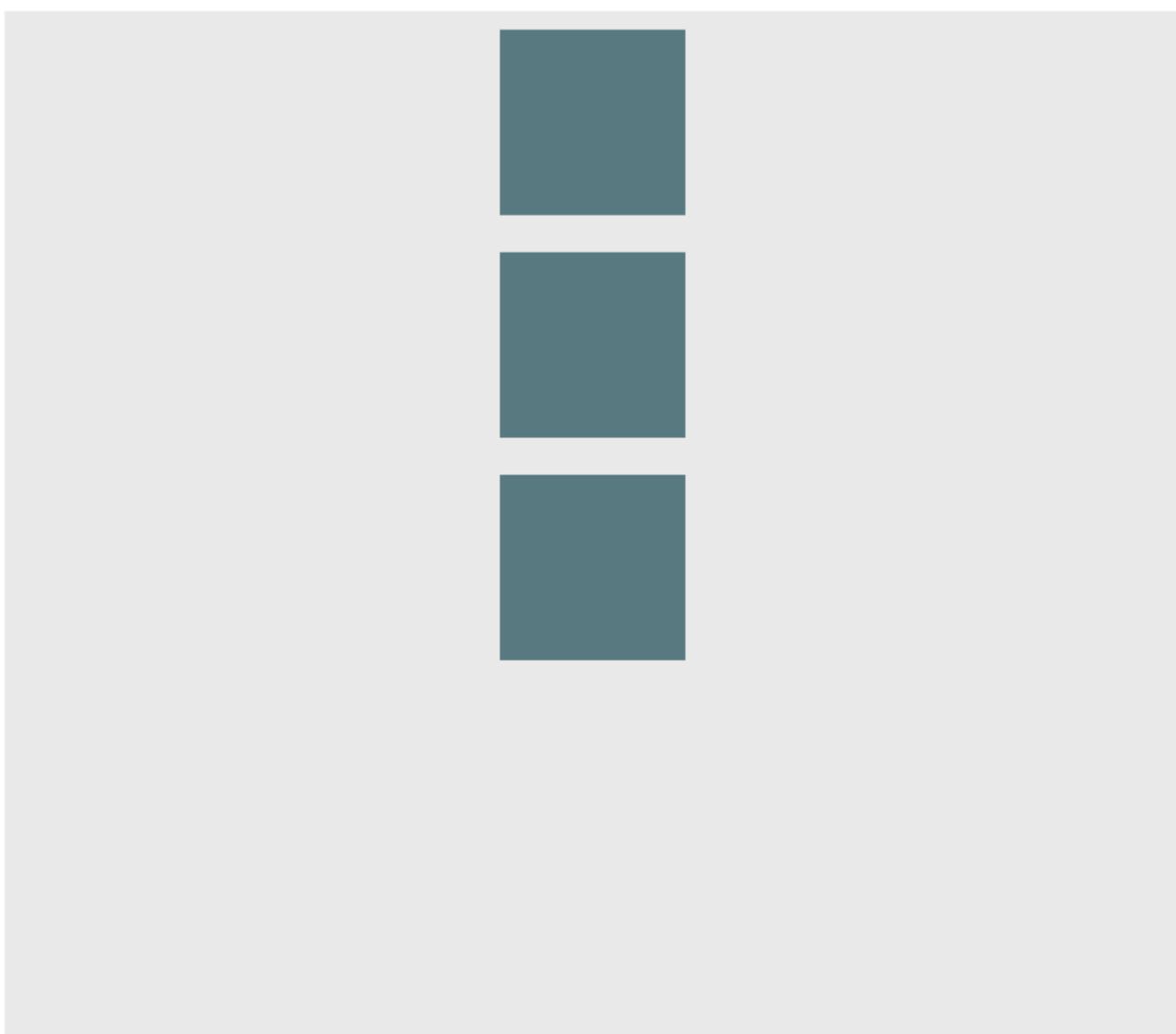
Aquí hay una [demo](#).

Ejemplo: align-content: center en un flexbox vertical

CSS:

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

Salir:

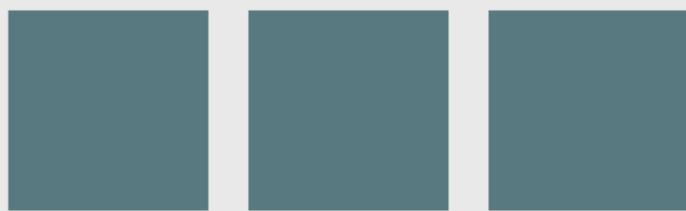


Aquí hay una [demo](#).

Ejemplo: combinación para centrar ambos en flexbox horizontal

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

Salir:



Aquí hay una [demo](#).

Ejemplo: combinación para centrar ambos en flexbox vertical

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

Salir:



Aquí hay una [demo](#).

Misma altura en contenedores anidados.

Este código garantiza que todos los contenedores anidados siempre tengan la misma altura. Esto se hace asegurándose de que todos los elementos anidados tengan la misma altura que la división parrent contenedora. [Ver ejemplo de trabajo : https://jsfiddle.net/3wwh7ewp/](https://jsfiddle.net/3wwh7ewp/)

Este efecto se logra debido a que la propiedad `align-items` está configurada para `stretch` de manera predeterminada.

HTML

```
<div class="container">
  <div style="background-color: red">
    Some <br />
    data <br />
    to make<br />
    a height <br />
  </div>
```

```
<div style="background-color: blue">
  Fewer <br />
  lines <br />
</div>
</div>
```

CSS

```
.container {
  display: flex;
  align-items: stretch; // Default value
}
```

Nota: [no funciona en versiones de IE bajo 10](#)

Ajustar los elementos de forma óptima a su contenedor.

Una de las características más agradables de flexbox es permitir que los contenedores se ajusten de manera óptima a su elemento principal.

[Demo en vivo](#) .

HTML:

```
<div class="flex-container">
  <div class="flex-item">1</div>
  <div class="flex-item">2</div>
  <div class="flex-item">3</div>
  <div class="flex-item">4</div>
  <div class="flex-item">5</div>
</div>
```

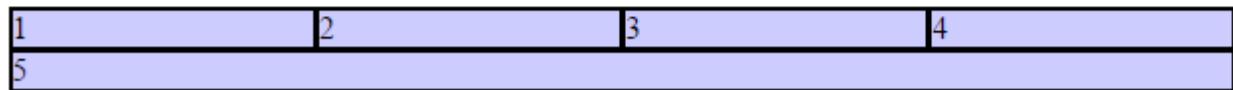
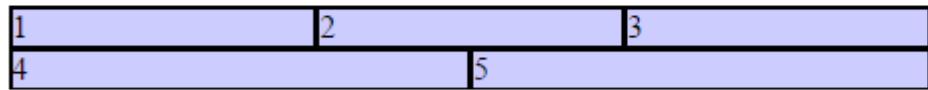
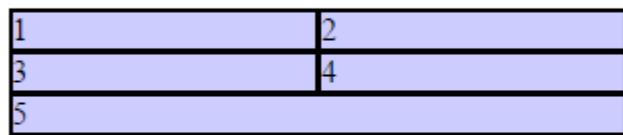
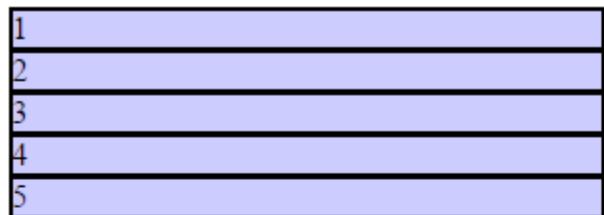
CSS:

```
.flex-container {
  background-color: #000;
  height: 100%;
  display:flex;
  flex-direction: row;
  flex-wrap: wrap;
  justify-content: flex-start;
  align-content: stretch;
  align-items: stretch;
}

.flex-item {
  background-color: #ccf;
  margin: 0.1em;
  flex-grow: 1;
  flex-shrink: 0;
  flex-basis: 200px; /* or % could be used to ensure a specific layout */
}
```

Salir:

Las columnas se adaptan a medida que la pantalla cambia de tamaño.



Lea Diseño de caja flexible (Flexbox) en línea: <https://riptutorial.com/es/css/topic/445/diseno-de-caja-flexible--flexbox->

Capítulo 21: Disposición de bloques en línea

Examples

Barra de navegación justificada

La barra de navegación (menú) justificada horizontalmente tiene una cantidad de elementos que deben estar justificados. El primer elemento (izquierdo) no tiene margen izquierdo dentro del contenedor, el último elemento (derecho) no tiene margen derecho dentro del contenedor. La distancia entre los elementos es igual, independiente del ancho del elemento individual.

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` fueron elegidas por su significado semántico de 'una lista de elementos de navegación (menú)'. Por supuesto, también se pueden usar otras etiquetas.
- El `:after` pseudo-elemento causa una 'línea' extra en la `ul` y, por lo tanto, una altura extra y vacía de este bloque, empujando el contenido hacia abajo. Esto se resuelve con el `margin-bottom` negativo, que debe tener la misma magnitud que la `line-height` (pero negativo).
- Si la página se vuelve demasiado estrecha para que quepan todos los elementos, los elementos se separarán en una nueva línea (comenzando desde la derecha) y se justificarán en esta línea. La altura total del menú crecerá según sea necesario.

Lea Disposición de bloques en línea en línea: <https://riptutorial.com/es/css/topic/3308/disposicion-de-bloques-en-linea>

Capítulo 22: El modelo de caja

Sintaxis

- tamaño de caja: *parámetro* ;

Parámetros

Parámetro	Detalle
content-box	El ancho y alto del elemento solo incluye el área de contenido.
padding-box	El ancho y alto del elemento incluye contenido y relleno.
border-box	El ancho y alto del elemento incluye contenido, relleno y borde.
initial	Establece el modelo de caja a su estado predeterminado.
inherit	Hereda el cuadro de modelo del elemento padre.

Observaciones

Acerca de la caja de relleno

Este valor solo lo implementó **Firefox** y, por lo tanto, no debe utilizarse. Fue eliminado en la versión 50.0 de Firefox.

Examples

¿Qué es el modelo de caja?

Los bordes

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

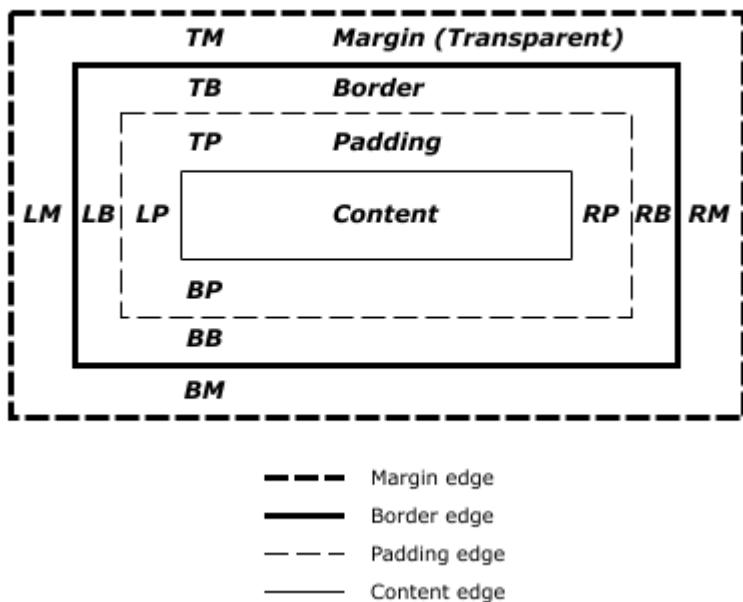


Diagrama del *borrador de trabajo* CSS2.2

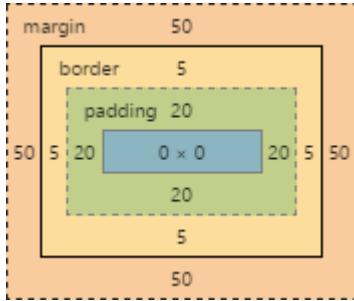
El perímetro de cada una de las cuatro áreas se llama *borde*. Cada borde define una caja.

- El rectángulo más interno es el **cuadro de contenido**. El ancho y el alto de esto dependen del contenido renderizado del elemento (texto, imágenes y cualquier elemento secundario que pueda tener).
- El siguiente es el **cuadro de relleno**, como lo define la propiedad de `padding`. Si no hay un ancho de `padding` definido, el borde de relleno es igual al borde de contenido.
- Luego tenemos el **cuadro de borde**, como lo define la propiedad de `border`. Si no hay un ancho de `border` definido, el borde del borde es igual al borde de relleno.
- El rectángulo más externo es el **cuadro de margen**, como lo define la propiedad de `margin`. Si no hay un ancho de `margin` definido, el borde del margen es igual al borde del borde.

Ejemplo

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

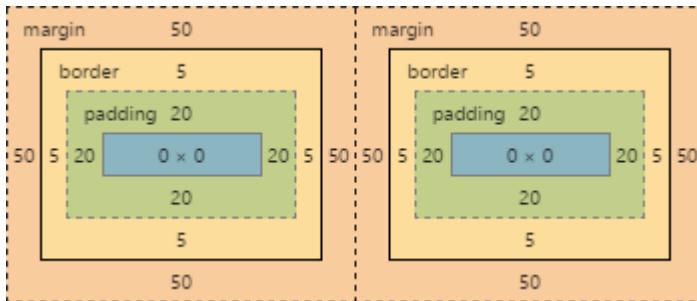
Este CSS da estilo a todos los elementos `div` para que tengan un borde superior, derecho, inferior e izquierdo de 5 `5px` de ancho; un margen superior, derecho, inferior e izquierdo de `50px`; y un relleno superior, derecho, inferior e izquierdo de `20px`. Ignorando el contenido, nuestro cuadro generado se verá así:



Captura de pantalla del panel de estilos de elementos de Google Chrome

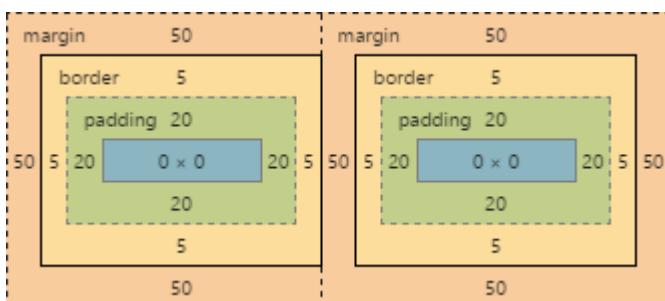
- Como no hay contenido, la región de contenido (el cuadro azul en el centro) no tiene altura ni ancho (0px por 0px).
- El cuadro de relleno por defecto es del mismo tamaño que el cuadro de contenido, más el ancho de 20 px en los cuatro bordes que definimos anteriormente con la propiedad de `padding` (40 px por 40 px).
- El cuadro de borde es del mismo tamaño que el cuadro de relleno, más el ancho de 5 px que definimos anteriormente con la propiedad de `border` (50 px por 50 px).
- Finalmente, el cuadro de margen es del mismo tamaño que el cuadro de borde, más el ancho de 50 px que definimos anteriormente con la propiedad de `margin` (lo que otorga a nuestro elemento un tamaño total de 150 px por 150 px).

Ahora vamos a darle a nuestro elemento un hermano con el mismo estilo. El navegador mira el modelo de caja de ambos elementos para averiguar dónde, en relación con el contenido del elemento anterior, el nuevo elemento debe colocarse:



El contenido de cada elemento está separado por un espacio de 150px, pero los cuadros de los dos elementos se tocan entre sí.

Si luego 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 se verían así:



tamaño de caja

El modelo de cuadro predeterminado (cuadro de `content-box`) puede ser contraintuitivo, ya que el `width / height` de un elemento no representará su ancho o alto real en la pantalla tan pronto como comience a agregar estilos de `border` y `padding` al elemento.

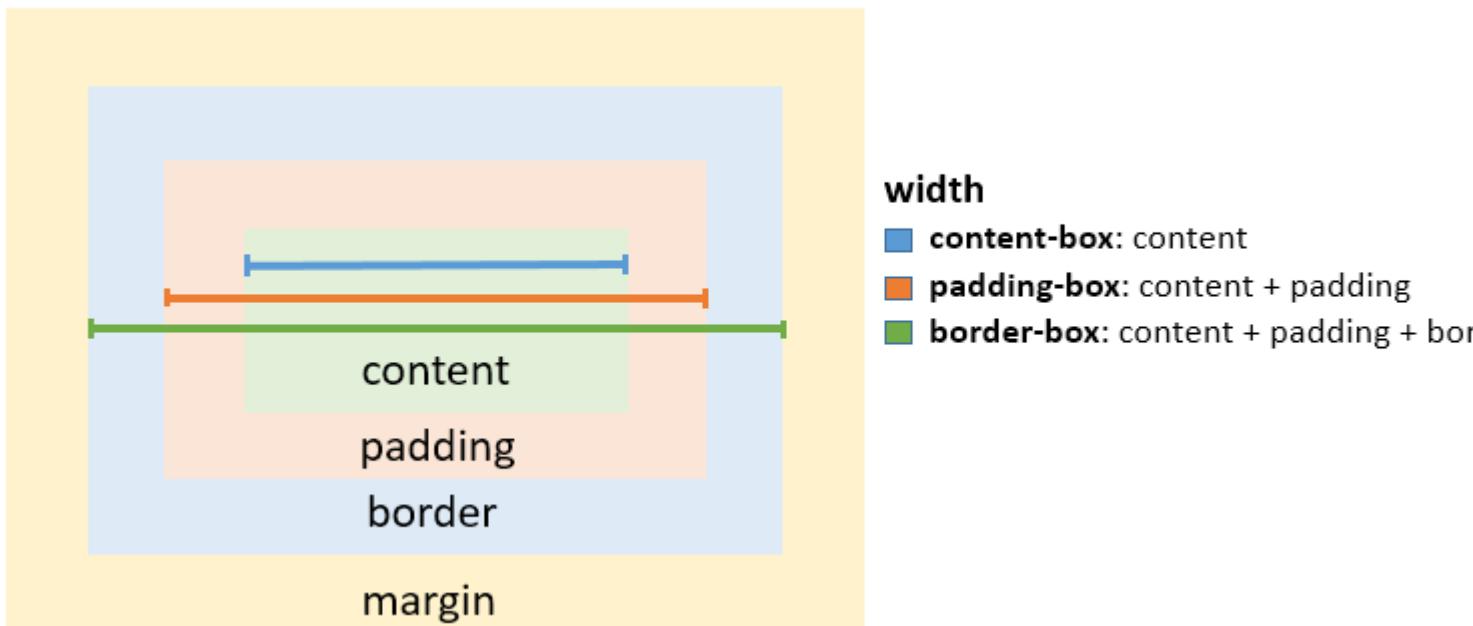
El siguiente ejemplo demuestra este problema potencial con `content-box`:

```
textarea {  
    width: 100%;  
    padding: 3px;  
    box-sizing: content-box; /* default value */  
}
```

Dado que el relleno se agregará al ancho del área de texto, el elemento resultante es un área de texto que es más ancho que el 100%.

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

- `content-box`: el modelo de cuadro común: ancho y alto solo incluye el contenido, no el relleno ni el borde
- `padding-box`: ancho y alto incluye el contenido y el relleno, pero no el borde
- `border-box`: el ancho y la altura incluyen el contenido, el relleno y el borde



Para resolver el problema de área de `textarea` anterior, simplemente puede cambiar la propiedad de `box-sizing` a `padding-box` o `border-box`. `border-box` es el más utilizado.

```
textarea {  
    width: 100%;
```

```
padding: 3px;  
box-sizing: border-box;  
}
```

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

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

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

Lea El modelo de caja en línea: <https://riptutorial.com/es/css/topic/646/el-modelo-de-caja>

Capítulo 23: Estilo del cursor

Sintaxis

- cursor: auto | por defecto | ninguno | menú contextual | ayuda | puntero progreso | espera | celular punto de mira texto | texto vertical | alias | copia | mover no-drop | no permitido | e-redimensionar | n-redimensionar | ne-redimensionar | nw-redimensionar | s-redimensionar | se-redimensionar | cambiar tamaño w-redimensionar | ew-redimensionar | ns-redimensionar | nesw-redimensionar | nwse-redimensionar | col-redimensionar | redimensionar | todo desplazamiento | acercar | alejar | agarrar agarrar

Examples

Cambiando el tipo de cursor

```
cursor: value;
```

	default
	crosshair
	hand
	pointer
	Cross browser
	move
	text
	wait
	help

	n-resize
	ne-resize
	e-resize
	se-resize
	s-resize
	sw-resize
	w-resize
	nw-resize
	progress

	not-allowed
	no-drop
	vertical-text
	all-scroll
	col-resize
	row-resize

Ejemplos:

Valor	Descripción
ninguna	No se renderiza ningún cursor para el elemento.
auto	Defecto. El navegador pone un cursor.
ayuda	El cursor indica que hay ayuda disponible.
Espere	El cursor indica que el programa está ocupado.
movimiento	El cursor indica que hay que mover algo.
puntero	El cursor es un puntero e indica un enlace.

eventos punteros

La propiedad puntero-eventos permite controlar cómo los elementos HTML responden a los eventos del mouse / táctil.

```
.disabled {  
  pointer-events: none;  
}
```

En este ejemplo,

'ninguno' impide todas las opciones de clic, estado y cursor en el elemento HTML especificado [[1]]

Otros valores válidos para los elementos HTML son:

- auto;
- heredar.

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>

color caret

La propiedad CSS de color caret especifica el color de caret, el indicador visible del punto de inserción en un elemento donde el texto y otro contenido se insertan al escribir o editar el usuario.

HTML

```
<input id="example" />
```

CSS

```
#example {  
  caret-color: red;  
}
```

Recursos:

- <https://developer.mozilla.org/en-US/docs/Web/CSS/caret-color>

Lea Estilo del cursor en línea: <https://riptutorial.com/es/css/topic/1742/estilo-del-cursor>

Capítulo 24: Estructura y formato de una regla CSS

Observaciones

Para facilitar la lectura, mantenga todas las declaraciones sangradas en un nivel de su selector y la llave de cierre en su propia línea. Agregue un solo espacio después de los selectores y los dos puntos, y siempre coloque un punto y coma después de la declaración final.

Bueno

```
p {  
    color: maroon;  
    font-size: 16px;  
}
```

Malo

```
p{  
    color: maroon;  
    font-size:16px }
```

Un trazador de líneas

Si solo hay una o dos declaraciones, *puede* salirse con la suya. No recomendado para la mayoría de los casos. Siempre sea consistente cuando sea posible.

```
p { color: maroon; font-size: 16px; }
```

Examples

Reglas, selectores y bloques de declaración

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

```
h1 {}
```

Listas de propiedades

Algunas propiedades pueden tomar varios valores, conocidos colectivamente como una **lista de**

propiedades .

```
/* Two values in this property list */
span {
    text-shadow: yellow 0 0 3px, green 4px 4px 10px;
}

/* Alternate Formatting */
span {
    text-shadow:
        yellow 0 0 3px,
        green 4px 4px 10px;
}
```

Selectores múltiples

Cuando agrupa los selectores de CSS, aplica los mismos estilos a varios elementos diferentes sin repetir los estilos en su hoja de estilos. Use una coma para separar los selectores agrupados múltiples.

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

Así que el color azul se aplica a todos los elementos `<div>` y todos los elementos `<p>`. Sin la coma, solo los elementos `<p>` que son un elemento secundario de un `<div>` serían rojos.

Esto también se aplica a todos los tipos de selectores.

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

Esta regla se aplica a:

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

Lea Estructura y formato de una regla CSS en línea:

<https://riptutorial.com/es/css/topic/4313/estructura-y-formato-de-una-regla-css>

Capítulo 25: Flotadores

Sintaxis

- claro: ninguno | izquierda | derecha | ambos | inline-start | inline-end;
- flotar: izquierda | derecha | ninguno | inline-start | inline-end;

Observaciones

Como float implica el uso del diseño de bloque, modifica el valor calculado de los valores de visualización en algunos casos [1]

[1]: <https://developer.mozilla.org/en-US/docs/Web/CSS/float> MDN

Examples

Flotar una imagen dentro del texto

El uso más básico de un flotador es envolver el texto alrededor de una imagen. El siguiente código producirá dos párrafos y una imagen, y el segundo párrafo fluirá alrededor de la imagen. Observe que siempre está contenido *después* del elemento flotado que fluye alrededor del elemento flotado.

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;  
}
```

Esta será 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 ne 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. Maecena 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.

[Codepen Link](#)

Diseño simple de dos columnas de ancho fijo

Un diseño simple de dos columnas consta de 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 difíciles con diseños de varias columnas que usan flotadores y requiere soluciones para que varias columnas parezcan de la misma altura.

HTML:

```

<div class="wrapper">

  <div class="sidebar">
    <h2>Sidebar</h2>

    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio.</p>
  </div>

  <div class="content">
    <h1>Content</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:

```

.wrapper {
  width:600px;
  padding:20px;
  background-color:pink;

  /* Floated elements don't use any height. Adding "overflow:hidden;" forces the
   parent element to expand to contain its floated children. */
  overflow:hidden;
}

.sidebar {
  width:150px;
  float:left;
  background-color:blue;
}

.content {
  width:450px;
  float:right;
  background-color:yellow;
}

```

Diseño simple de tres columnas de ancho fijo

HTML:

```

<div class="wrapper">
  <div class="left-sidebar">
    <h1>Left Sidebar</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
  </div>
  <div class="content">
    <h1>Content</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="right-sidebar">
  <h1>Right Sidebar</h1>
  <p>Fusce ac turpis quis ligula lacinia aliquet.</p>
</div>
</div>

```

CSS:

```

.wrapper {
  width:600px;
  background-color:pink;
  padding:20px;

  /* Floated elements don't use any height. Adding "overflow:hidden;" forces the
   parent element to expand to contain its floated children. */
  overflow:hidden;
}

.left-sidebar {
  width:150px;
  background-color:blue;
  float:left;
}

.content {
  width:300px;
  background-color:yellow;
  float:left;
}

.right-sidebar {
  width:150px;
  background-color:green;
  float:right;
}

```

Disposición perezosa / codiciosa de dos columnas

Este diseño utiliza una columna flotada para crear un diseño de dos columnas sin anchos definidos. En este ejemplo, la barra lateral izquierda es "perezosa", ya que solo ocupa todo el espacio que necesita. Otra forma de decir esto es que la barra lateral izquierda está "encogida". La columna de contenido correcta es "codiciosa", ya que ocupa todo el espacio restante.

HTML:

```

<div class="sidebar">
  <h1>Sidebar</h1>
  
</div>

<div class="content">
  <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:

```

.sidebar {
    /* `display:table;` shrink-wraps the column */
    display:table;
    float:left;
    background-color:blue;
}

.content {
    /* `overflow:hidden;` prevents `.content` from flowing under `.sidebar` */
    overflow:hidden;
    background-color:yellow;
}

```

[Violín](#)

claro propriedad

La propiedad clara está directamente relacionada con flotadores. Valores de propiedad:

- ninguno - Predeterminado. Permite elementos flotantes en ambos lados.
- izquierda - No se permiten elementos flotantes en el lado izquierdo
- derecha - No se permiten elementos flotantes en el lado derecho
- ambos - No se permiten elementos flotantes en el lado izquierdo o derecho
- inicial: establece esta propiedad en su valor predeterminado. Lea acerca de la inicial
- heredar: hereda esta propiedad de su elemento principal. Lea acerca de heredar

```

<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 </p>
<body><p class="clear">Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum
    Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum </p>
</body>
</html>
```

Clearfix

El hack de 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 flotantes, por lo tanto, con la posible confusión). Para *contener flotantes*, debe agregar la clase `.cf` o `.clearfix` en el contenedor (**el padre**) y `.clearfix` esta clase con algunas reglas que se describen a continuación.

3 versiones con efectos ligeramente diferentes (fuentes: [un nuevo hack de micro clearfix](#) de N. Gallagher y [clearfix recargado](#) por TJ Koblentz):

Clearfix (con el colapso del margen superior de los flotadores contenidos todavía ocurriendo)

```
.cf:after {
  content: "";
  display: table;
}

.cf:after {
  clear: both;
}
```

Clearfix también evita el colapso del margen superior de los flotadores contenidos

```
/**
 * For modern browsers
 * 1. The space content is one way to avoid an Opera bug when the
 *    contenteditable attribute is included anywhere else in the document.
 *    Otherwise it causes space to appear at the top and bottom of elements
 *    that are clearfixed.
 * 2. The use of `table` rather than `block` is only necessary if using
 *    `:before` to contain the top-margins of child elements.
 */
.cf:before,
```

```
.cf:after {  
    content: " "; /* 1 */  
    display: table; /* 2 */  
}  
  
.cf:after {  
    clear: both;  
}
```

Clearfix con soporte de navegadores obsoletos IE6 y IE7

```
.cf:before,  
.cf:after {  
    content: " ";  
    display: table;  
}  
  
.cf:after {  
    clear: both;  
}  
  
/**  
 * For IE 6/7 only  
 * Include this rule to trigger hasLayout and contain floats.  
 */  
.cf {  
    *zoom: 1;  
}
```

Codepen mostrando efecto clearfix

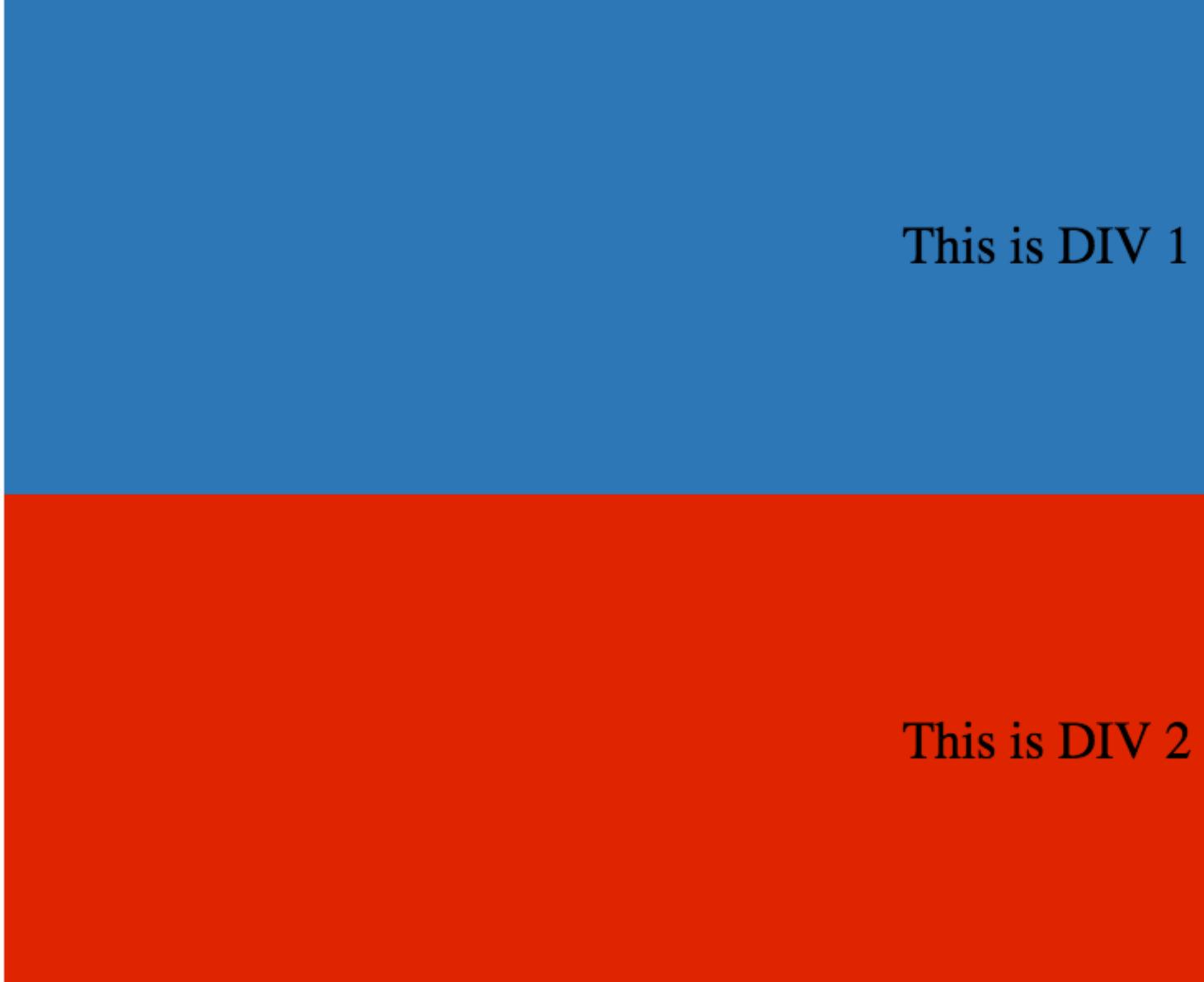
Otro recurso: [todo lo que sabe sobre clearfix es incorrecto](#) (clearfix y BFC - Contexto de formato de bloque, mientras que hasLayout se relaciona con los navegadores obsoletos IE6 tal vez 7)

DIV en línea usando flotador

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

```
<div>  
    <p>This is DIV 1</p>  
</div>  
<div>  
    <p>This is DIV 2</p>  
</div>
```

La salida del siguiente código será



This is DIV 1

This is DIV 2

Podemos hacerlos en línea agregando una propiedad css `float` al `div`.

HTML:

```
<div class="outer-div">
  <div class="inner-div1">
    <p>This is DIV 1</p>
  </div>
  <div class="inner-div2">
    <p>This is DIV 2</p>
  </div>
</div>
```

CSS

```
.inner-div1 {
  width: 50%;
  margin-right:0px;
  float:left;
```

```
background : #337ab7;
padding:50px 0px;
}

.inner-div2 {
    width: 50%;
    margin-right:0px;
    float:left;
    background : #dd2c00;
    padding:50px 0px;
}

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



This is DIV 1

[Codepen Link](#)

Uso de la propiedad de desbordamiento para eliminar flotadores.

Al establecer el valor de `overflow` en `hidden`, `auto` o `scroll` a un elemento, se borrarán todos los flotantes dentro de ese elemento.

Nota: usando `overflow:scroll` siempre mostrará el cuadro de scroll

Lea Flotadores en línea: <https://riptutorial.com/es/css/topic/405/flotadores>

Capítulo 26: Formas de un solo elemento

Examples

Cuadrado

Para crear un cuadrado, define un elemento con ancho y alto. En el siguiente ejemplo, tenemos un elemento con un `width` y `height` de 100 píxeles cada uno.



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

```
.square {  
    width: 100px;  
    height: 100px;  
    background: rgb(246, 156, 85);  
}
```

triangulos

Para crear un triángulo CSS, defina un elemento con un ancho y alto de 0 píxeles. La forma del triángulo se formará usando propiedades de borde. Para un elemento con altura 0 y ancho, los 4 bordes (arriba, derecha, abajo, izquierda) forman cada uno un triángulo. Aquí hay un elemento con 0 altura / ancho y 4 bordes de diferentes colores.



Al establecer algunos bordes en transparente y otros en un color, podemos crear varios triángulos. Por ejemplo, en el triángulo hacia arriba, establecemos el borde inferior en el color deseado, luego establecemos los bordes izquierdo y derecho en transparente. Aquí hay una imagen con los bordes izquierdo y derecho sombreados ligeramente para mostrar cómo se está formando el triángulo.



Las dimensiones del triángulo se pueden modificar cambiando los diferentes anchos de borde: más alto, más corto, ladeado, etc. Los siguientes ejemplos muestran un triángulo de 50x50 píxeles.

Triángulo - apuntando hacia arriba



```
<div class="triangle-up"></div>
```

```
.triangle-up {  
    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="triangle-down"></div>
```

```
.triangle-down {  
    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="triangle-right"></div>
```

```
.triangle-right {  
    width: 0;  
    height: 0;  
    border-top: 25px solid transparent;  
    border-bottom: 25px solid transparent;  
    border-left: 50px solid rgb(246, 156, 85);  
}
```

Triángulo - Señalando a la izquierda



```
<div class="triangle-left"></div>
```

```
.triangle-left {  
    width: 0;  
    height: 0;  
    border-top: 25px solid transparent;  
    border-bottom: 25px solid transparent;  
    border-right: 50px solid rgb(246, 156, 85);  
}
```

Triángulo - Apuntando hacia arriba / derecha



```
<div class="triangle-up-right"></div>
```

```
.triangle-up-right {  
    width: 0;  
    height: 0;  
    border-top: 50px solid rgb(246, 156, 85);  
    border-left: 50px solid transparent;  
}
```

Triángulo - hacia arriba / izquierda



```
<div class="triangle-up-left"></div>
```

```
.triangle-up-left {  
    width: 0;  
    height: 0;  
    border-top: 50px solid rgb(246, 156, 85);  
    border-right: 50px solid transparent;  
}
```

Triángulo - apuntando hacia abajo / derecha



```
<div class="triangle-down-right"></div>
```

```
.triangle-down-right {  
    width: 0;  
    height: 0;  
    border-bottom: 50px solid rgb(246, 156, 85);  
    border-left: 50px solid transparent;  
}
```

Triángulo - apuntando hacia abajo / izquierda



```
<div class="triangle-down-left"></div>
```

```
.triangle-down-left {  
    width: 0;  
    height: 0;  
    border-bottom: 50px solid rgb(246, 156, 85);  
    border-right: 50px solid transparent;
```

}

Ráfagas

Una ráfaga es similar a una estrella, pero con los puntos que se extienden menos distancia del cuerpo. Piense en una forma de ráfaga como un cuadrado con cuadrados adicionales, ligeramente girados, en capas en la parte superior.

Los cuadrados adicionales se crean utilizando los elementos `::before` y `::after` psuedo.

Ráfaga de 8 puntos

Una ráfaga de 8 puntos son cuadrados de 2 capas. El cuadrado inferior es el elemento en sí mismo, el cuadrado adicional se crea utilizando el `::before` pseudo-elemento. La parte inferior se gira 20 °, el cuadrado superior se gira 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 cuadrados de 3 capas. El cuadrado inferior es el elemento en sí mismo, los cuadrados adicionales se crean utilizando `::before` y `::after` pseudo-elementos. La parte inferior se gira 0 °, la siguiente casilla se gira 30 ° y la parte superior se 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);
}
```

Círculos y elipses

Círculo

Para crear un **círculo**, defina un elemento con un `width` y una `height` iguales (un *cuadrado*) y luego establezca la propiedad de `border-radius` de este elemento en `50%`.



HTML

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

CSS

```
.circle {  
    width: 50px;  
    height: 50px;  
    background: rgb(246, 156, 85);  
    border-radius: 50%;  
}
```

Elipse

Una **elipse** es similar a un círculo, pero con diferentes valores para el `width` y la `height`.



HTML

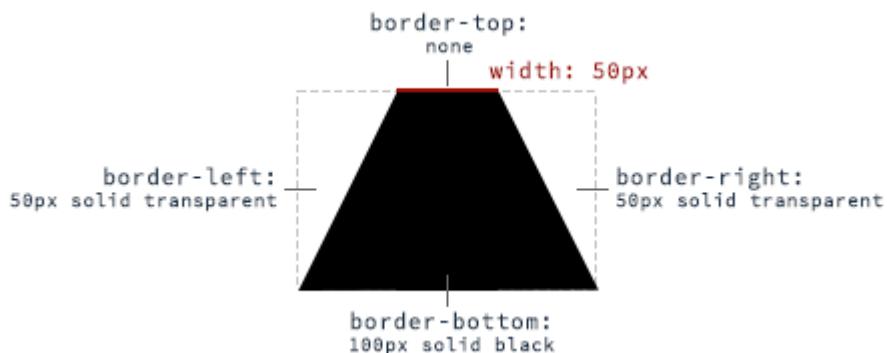
```
<div class="oval"></div>
```

CSS

```
.oval {  
    width: 50px;  
    height: 80px;  
    background: rgb(246, 156, 85);  
    border-radius: 50%;  
}
```

Trapecio

Un trapezoide se puede hacer con un elemento de bloque con altura cero (altura de `0px`), un ancho mayor que cero y un borde, que es transparente, excepto por un lado:



HTML:

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

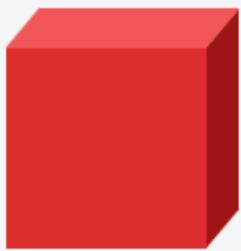
CSS:

```
.trapezoid {  
    width: 50px;  
    height: 0;  
    border-left: 50px solid transparent;  
    border-right: 50px solid transparent;  
    border-bottom: 100px solid black;  
}
```

Al cambiar los lados del borde, se puede ajustar la orientación del trapecio.

Cubo

Este ejemplo muestra cómo crear un cubo usando los métodos de transformación 2D `skewX()` y `skewY()` en pseudo elementos.



HTML:

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

CSS:

```
.cube {  
    background: #dc2e2e;  
    width: 100px;  
    height: 100px;  
    position: relative;  
    margin: 50px;  
}  
  
.cube::before {  
    content: '';  
    display: inline-block;  
    background: #f15757;  
    width: 100px;  
    height: 20px;  
    transform: skewX(-40deg);  
    position: absolute;  
    top: -20px;  
    left: 8px;
```

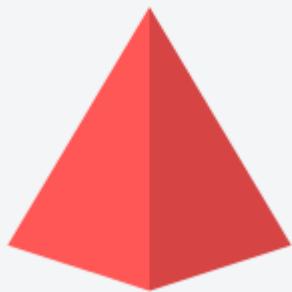
```
}

.cube::after {
  content: '';
  display: inline-block;
  background: #9e1515;
  width: 16px;
  height: 100px;
  transform: skewY(-50deg);
  position: absolute;
  top: -10px;
  left: 100%;
}
```

[Ver demo](#)

Pirámide

Este ejemplo muestra cómo crear una **pirámide** usando bordes y métodos de transformación 2D `skewY()` y `skewY() rotate()` en pseudo elementos.



HTML:

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

CSS:

```
.pyramid {
  width: 100px;
  height: 200px;
  position: relative;
  margin: 50px;
}

.pyramid::before, .pyramid::after {
  content: '';
  display: inline-block;
  width: 0;
  height: 0;
  border: 50px solid;
  position: absolute;
}

.pyramid::before {
  border-color: transparent transparent #ff5656 transparent;
```

```
    transform: scaleY(2) skewY(-40deg) rotate(45deg);  
}  
  
.pyramid::after {  
    border-color: transparent transparent #d64444 transparent;  
    transform: scaleY(2) skewY(40deg) rotate(-45deg);  
}
```

Lea Formas de un solo elemento en línea: <https://riptutorial.com/es/css/topic/2862/formas-de-un-solo-elemento>

Capítulo 27: Formas para flotadores

Sintaxis

- forma fuera: ninguna | [<basic-shape> || <shape-box>] | <imagen>
- forma-margen: <length> | <percentage>
- forma-imagen-umbral: <número>

Parámetros

Parámetro	Detalles
ninguna	Un valor de <code>none</code> significa que el área flotante (el área que se usa para envolver el contenido alrededor de un elemento flotante) no se ve afectada. Este es el valor predeterminado / inicial.
forma básica	Se refiere a uno entre <code>inset()</code> , <code>circle()</code> , <code>ellipse()</code> o <code>polygon()</code> . Usando una de estas funciones y sus valores, se define la forma.
caja de formas	Se refiere a uno entre <code>margin-box</code> <code>border-box</code> <code>padding-box</code> <code>content-box</code> . Cuando solo se proporciona <shape-box> (sin <basic-shape>) esta caja es la forma. Cuando se usa junto con <basic-shape>, esto actúa como el cuadro de referencia.
imagen	Cuando se proporciona una imagen como valor, la forma se calcula en función del canal alfa de la imagen especificada.

Observaciones

El soporte del navegador para el módulo CSS Shapes es muy limitado en este momento.

Es compatible con Chrome v37 + y Opera 24+ sin prefijos de navegador / proveedor. Safari lo admite desde v7.1 + pero con el prefijo `-webkit-` .

Todavía no es compatible con IE, Edge y Firefox.

Examples

Forma exterior con forma básica - círculo ()

Con la propiedad CSS `shape-outside` la `shape-outside` se pueden definir valores de forma para el área flotante, de modo que el contenido en línea se ajuste a la forma en lugar de a la caja flotante.

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; /* purely for demo */  
}
```

HTML

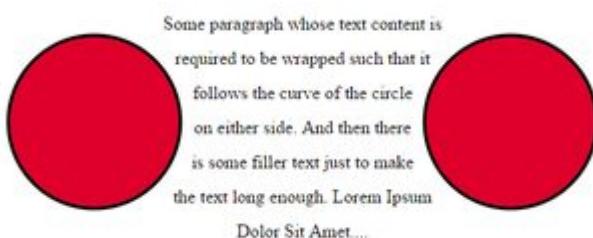
```
  
  
<p>Some paragraph whose text content is required to be wrapped such that it follows the curve  
of the circle on either side. And then there is some filler text just to make the text long  
enough. Lorem Ipsum Dolor Sit Amet....</p>
```

En el ejemplo anterior, ambas imágenes son en realidad imágenes cuadradas y, cuando el texto se coloca sin la propiedad de `shape-outside`, no fluirá alrededor del círculo en ninguno de los lados. Fluirá alrededor de la caja contenedora de la imagen solamente. Con `shape-outside` el área flotante se redefine como un *círculo* y el contenido fluye alrededor de este *círculo imaginario* que se crea utilizando `shape-outside`.

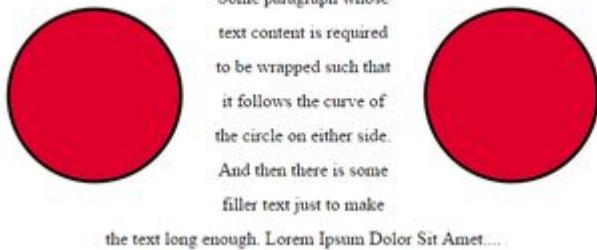
El *círculo imaginario* que se usa para redefinir el área de flotación es un círculo con un radio de 80 píxeles dibujado desde el centro-medio del cuadro de referencia de la imagen.

Abajo hay un par de capturas de pantalla para ilustrar cómo se envolvería el contenido cuando se usa `shape-outside` y cuando no se usa.

Salida con `shape-outside`



Salida sin `shape-outside`



Margen de forma

La `shape-margin` propiedad CSS añade un *margen* para `shape-outside`.

CSS

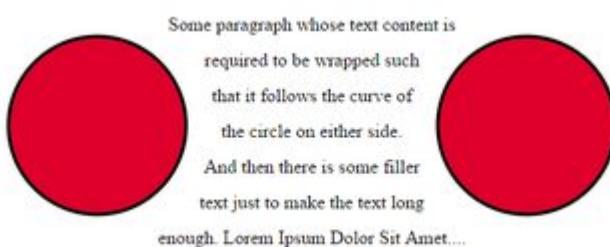
```
img:nth-of-type(1) {  
  shape-outside: circle(80px at 50% 50%);  
  shape-margin: 10px;  
  float: left;  
  width: 200px;  
}  
img:nth-of-type(2) {  
  shape-outside: circle(80px at 50% 50%);  
  shape-margin: 10px;  
  float: right;  
  width: 200px;  
}  
p {  
  text-align: center;  
  line-height: 30px; /* purely for demo */  
}
```

HTML

```
  
  
<p>Some paragraph whose text content is required to be wrapped such that it follows the curve of the circle on either side. And then there is some filler text just to make the text long enough. Lorem Ipsum Dolor Sit Amet....</p>
```

En este ejemplo, se agrega un margen de 10px alrededor de la **forma** usando `shape-margin`. Esto crea un poco más de espacio entre el *círculo imaginario* que define el área flotante y el contenido real que está fluyendo alrededor.

Salida:



Lea Formas para flotadores en línea: <https://riptutorial.com/es/css/topic/2034/formas-para-flotadores>

Capítulo 28: Fragmentación

Sintaxis

- Salto de página después: auto | siempre | evitar izquierda | derecha | inicial | heredar;
- Page-break-before: auto | siempre | evitar izquierda | derecha | inicial | heredar;
- Page-break-inside: auto | evitar inicial | heredar;

Parámetros

Valor	Descripción
auto	Defecto. Saltos de página automáticos
siempre	Siempre inserta un salto de página
evitar	Evite el salto de página (si es posible)
izquierda	Insertar saltos de página para que la página siguiente se formatee como una página izquierda
Correcto	Insertar saltos de página para que la página siguiente se formatee como una página de la derecha
inicial	Establece esta propiedad a su valor predeterminado.
heredar	Hereda esta propiedad de su elemento padre.

Observaciones

No hay propiedad de salto de página en CSS. Solo las 3 propiedades (**salto de página anterior**, **salto de página posterior**, **salto de página interno**).

Relacionados: `orphans` , `widows` .

Examples

Impresión de medios de página

```
@media print {  
  p {  
    page-break-inside: avoid;  
  }  
  h1 {  
    page-break-before: always;  
  }  
}
```

```
}

h2 {
    page-break-after: avoid;
}

}
```

Este código hace 3 cosas:

- evita un salto de página dentro de cualquier etiqueta p, lo que significa que un párrafo nunca se dividirá en dos páginas, si es posible.
- obliga a un salto de página anterior en todos los encabezados h1, lo que significa que antes de cada aparición de h1, habrá un salto de página.
- Evita saltos de página justo después de cualquier h2.

Lea Fragmentación en línea: <https://riptutorial.com/es/css/topic/4316/fragmentacion>

Capítulo 29: Frontera

Sintaxis

- **frontera**
 - borde: borde-ancho borde-estilo borde-color | inicial | heredar;
 - borde superior: borde ancho ancho borde estilo color | inicial | heredar;
 - borde inferior: borde ancho ancho borde estilo color | inicial | heredar;
 - borde izquierdo: borde ancho ancho borde estilo color | inicial | heredar;
 - borde derecho: borde ancho ancho borde estilo color | inicial | heredar;
- **estilo de borde**
 - estilo de borde: 1-4 ninguno | oculto | punteado | punteado | sólido doble | surco cresta inserción inicio | inicial | heredar;
- **radio del borde**
 - radio de borde: 1-4 longitud | % / 1-4 de longitud | % | inicial | heredar;
 - borde superior-izquierdo-radio: longitud | % [longitud | %] | inicial | heredar;
 - borde superior-derecho-radio: longitud | % [longitud | %] | inicial | heredar;
 - borde inferior-izquierdo-radio: longitud | % [longitud | %] | inicial | heredar;
 - borde inferior-derecho-radio: longitud | % [longitud | %] | inicial | heredar;
- **imagen de borde**
 - border-image: border-image-source border-image-slice [border-image-width [border-image-outset]] border-image-repeat
 - borde-imagen-fuente: ninguna | imagen;
 - borde-imagen-rebanada: 1-4 número | porcentaje [relleno]
 - borde-imagen-repetición: 1-2 estiramiento | repetir redondo | espacio
- **colapso de la frontera**
 - colapso de la frontera: separado | colapso | inicial | heredar

Observaciones

Propiedades relacionadas:

- frontera
- borde inferior
- color de borde inferior
- borde inferior-izquierdo-radio
- borde inferior-derecha-radio
- estilo de borde inferior
- borde inferior ancho
- color del borde
- imagen de borde
- borde-imagen-inicio
- borde-imagen-repetición
- borde-imagen-rebanada
- borde-imagen-fuente
- borde-imagen-ancho
- borde izquierdo
- color de borde izquierdo
- borde izquierdo
- ancho de borde izquierdo
- radio del borde
- borde derecho
- color de borde derecho
- estilo de borde derecho
- borde derecho ancho
- estilo de borde
- borde superior
- borde superior de color

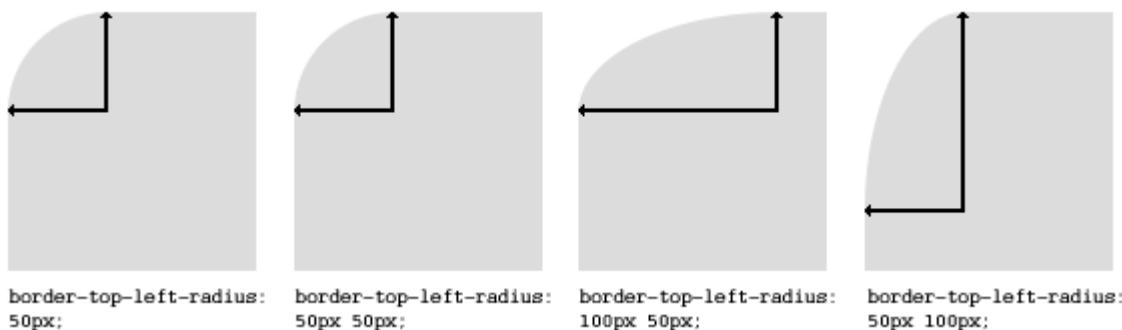
- borde superior-izquierdo-radio
- borde superior-derecho-radio
- estilo de borde superior
- ancho del borde superior
- ancho del borde

Examples

radio del borde

La propiedad border-radius le 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 de valores opcional, precedido por una '/', define el radio vertical. Si solo 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;
```

El 10px es el radio horizontal de la parte superior izquierda y de la parte 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 la parte superior izquierda, superior derecha, inferior derecha e inferior izquierda.

Al igual que con muchas propiedades de CSS, las abreviaturas se pueden utilizar para cualquiera o todos los valores posibles. Por lo tanto, puede especificar cualquier cosa de uno a ocho valores. La siguiente taquigrafía le permite establecer el radio horizontal y vertical de cada esquina al mismo valor:

HTML:

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

CSS:

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

El radio de borde es más comúnmente usado 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:

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

Debido a que el radio del borde acepta porcentajes, es común usar el 50% para evitar el cálculo manual del valor del radio del borde:

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

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

Ejemplo de radio de borde 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;
```

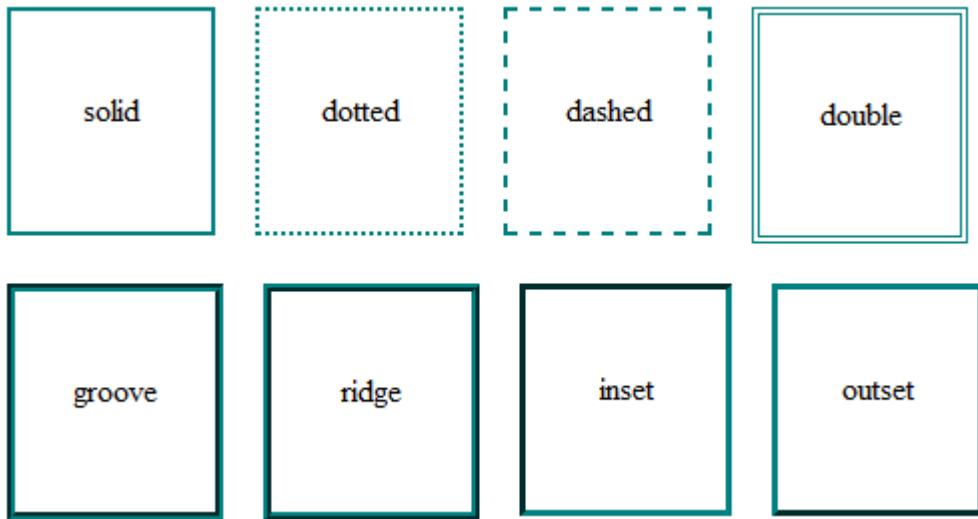
estilo de borde

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

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 trabajos `hidden` para la resolución de conflictos de borde para elementos `<table>`. En una `<table>` con múltiples fronteras, `none` tiene la prioridad más baja (es decir, en un conflicto, la frontera mostraría), y `hidden` tiene la prioridad más alta (es decir, en un conflicto, la frontera no se mostraría).

frontera (taquigrafía)

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

En lugar de escribir:

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

Usted puede simplemente escribir:

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

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

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

imagen de borde

Con la propiedad de `border-image` tiene la posibilidad de configurar una imagen para usarla en

lugar de los estilos de borde normales.

Una `border-image` consiste esencialmente en una

- `border-image-source` : la ruta a la imagen que se usará
- `border-image-slice` : especifica el desplazamiento que se usa 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 la mitad de la imagen del borde

Considere el siguiente ejemplo wheras border.png es una imagen de 90x90 píxeles:

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

La imagen se dividirá en nueve regiones con 30x30 píxeles. Los bordes se utilizarán como las esquinas del borde, mientras que los lados se utilizarán en el medio. Si el elemento es mayor / se **estirará** más ancha que 30px esta parte de la imagen. La parte media de la imagen por defecto es transparente.

borde- [izquierda | derecha | arriba | abajo]

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

Por ejemplo, si desea agregar un borde al lado izquierdo de un elemento, puede hacer:

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

colapso de la frontera

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

```
table {  
    border-collapse: separate; /* default */  
    border-spacing: 2px; /* Only works if border-collapse is separate */  
}
```

Ver también [Tablas](#) - entrada de documentación de [colapso de borde](#)

Múltiples fronteras

Utilizando esquema:

```
.div1{  
    border: 3px solid black;  
    outline: 6px solid blue;
```

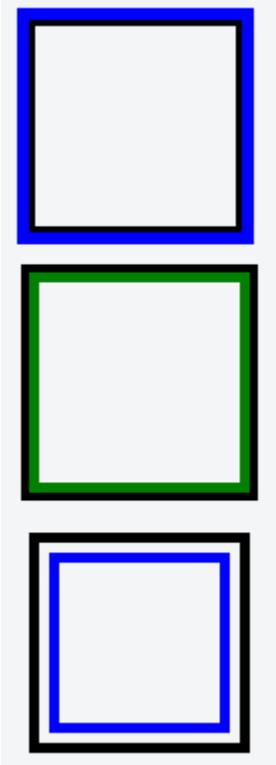
```
width: 100px;  
height: 100px;  
margin: 20px;  
}
```

Usando box-shadow:

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

Usando un pseudo elemento:

```
.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;  
}
```



<http://jsfiddle.net/MadalinaTn/bvqpcohm/2/>

Creando un borde multicolor usando una imagen de borde

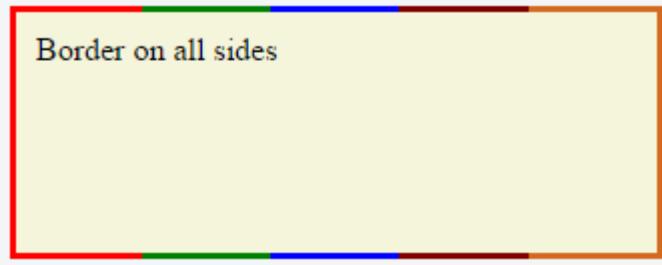
CSS

```
.bordered {  
    border-image: linear-gradient(to right, red 20%, green 20%, green 40%, blue 40%, blue 60%,  
    maroon 60%, maroon 80%, chocolate 80%); /* gradient with required colors */  
    border-image-slice: 1;  
}
```

HTML

```
<div class='bordered'>Border on all sides</div>
```

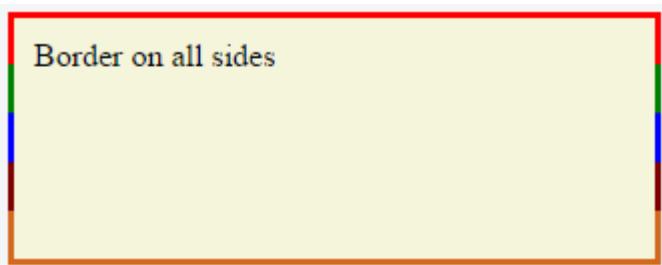
El ejemplo anterior produciría un borde que se compone de 5 colores diferentes. Los colores se definen a través de un `linear-gradient` (puede encontrar más información sobre los degradados en los [documentos](#)). Puede encontrar más información sobre la propiedad `border-image-slice` en el [ejemplo de `border-image`](#) en la misma página.



(*Nota: Se agregaron propiedades adicionales al elemento para fines de presentación*) .

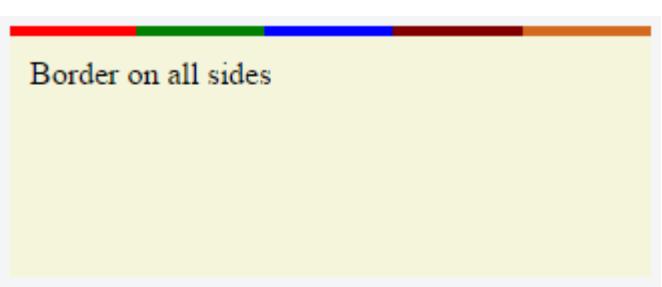
Habría notado que el borde izquierdo tiene un solo color (el color de inicio del degradado), mientras que el borde derecho también tiene un solo color (el color final del degradado). Esto se debe a la forma en que funciona la propiedad de imagen de borde. Es como si el degradado se aplicara a todo el cuadro y luego se enmascararan los colores de las áreas de relleno y contenido, por lo que parece que solo el borde tiene el degradado.

Los bordes que tienen un solo color dependen de la definición del degradado. Si el degradado es un degradado `to right`, el borde izquierdo sería el color de inicio del degradado y el borde derecho sería el color final. Si se tratara de un gradiente `to bottom` el borde superior sería el color de inicio del degradado y el borde inferior sería el color final. A continuación se muestra la salida de un gradiente de color de `to bottom 5`.



Si el borde solo se requiere en lados específicos del elemento, la propiedad de `border-width` se puede usar como con cualquier otro borde normal. Por ejemplo, agregar el siguiente código produciría un borde solo en la parte superior del elemento.

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



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

Los fondos de una caja, pero no su imagen de borde, se recortan a la curva apropiada (según lo determinado por 'clip de fondo').

Lea Frontera en línea: <https://riptutorial.com/es/css/topic/2160/frontera>

Capítulo 30: Funciones

Sintaxis

- <calc()> = calc(<calc-sum>)
- <calc-sum> = <calc-product> [['+' | '-'] <calc-product>]*
- <calc-product> = <calc-value> ['*' <calc-value> | '/' <number>]*
- <calc-value> = <number> | <dimension> | <percentage> | (<calc-sum>)

Observaciones

Para `calc()`, se requiere espacio en blanco alrededor de los operadores " - " y " + ", pero no los operadores " * " o " / ".

Todas las unidades deben ser del mismo tipo; tratar de multiplicar una altura por una duración de tiempo, por ejemplo, no es válido.

Examples

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 de px de un porcentaje) para calcular el valor de un atributo.

Se pueden usar todos los operadores + , - , / , y * , y se pueden agregar paréntesis para especificar el orden de las operaciones si es necesario.

Usa `calc()` para calcular el ancho 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;  
}
```

Use `calc()` para determinar la posición de una imagen de fondo:

```
background-position: calc(50% + 17px) calc(50% + 10px), 50% 50%;
```

Usa `calc()` para determinar la altura de un elemento:

```
height: calc(100% - 20px);
```

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 usar (por ejemplo, dentro de `::before` y `::after` **pseudo-element**) utilizando 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);  
}
```

función de gradiente lineal ()

Crea una imagen que representa un degradado 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 en amarillo al 50% y terminan en azul.

función de gradiente radial ()

Crea una imagen que representa un degradado de colores que se irradia desde el centro del degradado.

```
radial-gradient(red, orange, yellow) /*A gradient coming out from the middle of the  
gradient, red at the center, then orange, until it is finally yellow at the edges*/
```

función var ()

La función var () permite acceder a las variables CSS.

```
/* set a variable */  
:root {  
    --primary-color: blue;  
}  
  
/* access variable */  
selector {  
    color: var(--primary-color);  
}
```

Esta característica está actualmente en desarrollo. Compruebe caniuse.com para la última compatibilidad del navegador.

Lea Funciones en línea: <https://riptutorial.com/es/css/topic/2214/funciones>

Capítulo 31: Hacks de Internet Explorer

Observaciones

Estos "hacks" pueden usarse para apuntar a un navegador / cliente específico. Esto se puede usar para solucionar las diferencias de representación del navegador mediante la aplicación de estilos en uno de los envoltorios mencionados anteriormente.

Examples

Modo de alto contraste en Internet Explorer 10 y superior

En Internet Explorer 10+ y Edge, Microsoft proporciona el selector de medios de `-ms-high-contrast` para exponer la configuración de "Alto contraste" desde el navegador, lo que permite al programador ajustar los estilos de su sitio en consecuencia.

El `-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 es compatible con Edge en el futuro.

Ejemplos

```
@media screen and (-ms-high-contrast: active), (-ms-high-contrast: black-on-white) {  
    .header{  
        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 esté en `black-on-white` modo `black-on-white`.

```
@media screen and (-ms-high-contrast: white-on-black) {  
    .header{  
        background: #000;  
        color: #fff;  
    }  
}
```

Similar al primer ejemplo, pero esto 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](#) en `-ms-high-contrast`

Sólo Internet Explorer 6 e Internet Explorer 7

Para apuntar a Internet Explorer 6 e Internet Explorer 7, inicie sus propiedades con * :

```
.hide-on-ie6-and-ie7 {  
    *display : none; // This line is processed only on IE6 and IE7  
}
```

Los prefijos no alfanuméricicos (que no sean guiones y guiones bajos) se ignoran en IE6 e IE7, por lo que este truco funciona para cualquier par de `property: value` prefijo.

Sólo Internet Explorer 8

Para apuntar a Internet Explorer 8, envuelva sus selectores dentro de la `@media \0 screen { }` :

```
@media \0 screen {  
    .hide-on-ie8 {  
        display : none;  
    }  
}
```

Todo entre la `@media \0 screen { }` es procesado solo por I

Adición de soporte de bloque en línea a IE6 y IE7

```
display: inline-block;
```

Internet Explorer 6 y 7 no admiten la propiedad de `display` con el valor de `inline-block`. Una solución alternativa para esto es:

```
zoom: 1;  
*display: inline;
```

La propiedad de `zoom` activa la característica `hasLayout` de los elementos, y está disponible solo en Internet Explorer. La `*display` se asegura de que la propiedad no válida se ejecute solo en los navegadores afectados. Otros navegadores simplemente ignorarán la regla.

Lea Hacks de Internet Explorer en línea: <https://riptutorial.com/es/css/topic/5056/hacks-de-internet-explorer>

Capítulo 32: Herencia

Sintaxis

- *propiedad*: heredar;

Examples

Herencia automática

La herencia es un mecanismo fundamental de CSS mediante el cual los valores computados de algunas propiedades de un elemento se aplican a sus 'hijos'. Esto es particularmente útil cuando desea establecer un estilo global para sus elementos en lugar de tener que establecer dichas propiedades para cada elemento de su marca.

Las propiedades comunes que se heredan automáticamente son: `font`, `color`, `text-align`, `line-height`.

Supongamos la siguiente hoja de estilo:

```
#myContainer {  
    color: red;  
    padding: 5px;  
}
```

Esto aplicará `color: red` no solo al elemento `<div>` sino también a los elementos `<h3>` y `<p>`. Sin embargo, debido a la naturaleza del `padding` su valor **no** se heredará de esos elementos.

```
<div id="myContainer">  
    <h3>Some header</h3>  
    <p>Some paragraph</p>  
</div>
```

Herencia forzada

Algunas propiedades no se heredan automáticamente de un elemento a sus hijos. Esto se debe a que normalmente se desea que esas propiedades sean únicas para el elemento (o selección de elementos) al que se aplica la propiedad. Estas propiedades comunes son el `margin`, el `padding`, el `background`, la `display`, etc.

Sin embargo, a veces la herencia se desea de todos modos. Para lograr esto, podemos aplicar el valor `inherit` a la propiedad que se debe heredar. El valor `inherit` se puede aplicar a *cualquier* propiedad CSS y *cualquier* elemento HTML.

Supongamos la siguiente hoja de estilo:

```
#myContainer {  
    color: red;  
    padding: 5px;  
}  
#myContainer p {  
    padding: inherit;  
}
```

Esto aplicará `color: red` a los elementos `<h3>` y `<p>` debido a la naturaleza hereditaria de la propiedad de `color`. Sin embargo, el elemento `<p>` también heredará el valor de `padding` de su padre porque se especificó.

```
<div id="myContainer">  
    <h3>Some header</h3>  
    <p>Some paragraph</p>  
</div>
```

Lea Herencia en línea: <https://riptutorial.com/es/css/topic/3586/herencia>

Capítulo 33: Imagen de CSS Sprites

Sintaxis

- // Usando la posición de fondo
fondo: url ("sprite-image.png");
posición de fondo: -20px 50px;
- // taquigrafía de propiedad de fondo
fondo: url ("sprite-image.png") -20px 50px;

Observaciones

Para algunos casos de uso, los sprites están perdiendo popularidad, siendo reemplazados por iconos webfonts o [imágenes SVG](#).

Examples

Una implementación básica

¿Qué es un sprite de imagen?

Un sprite de imagen es un activo único ubicado dentro de una hoja de sprite de imagen. Una hoja de sprite de imagen es un archivo de imagen que contiene más de un recurso que se puede extraer de ella.

Por ejemplo:



La imagen de arriba es una hoja de sprite de imagen, y cada una de esas estrellas es un sprite dentro de la hoja de sprite. Estas hojas de sprites son útiles porque mejoran el rendimiento al reducir la cantidad de solicitudes HTTP que un navegador podría tener que hacer.

Entonces, ¿cómo implementar uno? Aquí hay un código de ejemplo.

HTML

```
<div class="icon icon1"></div>
```

```
<div class="icon icon2"></div>
<div class="icon icon3"></div>
```

CSS

```
.icon {
    background: url("icons-sprite.png");
    display: inline-block;
    height: 20px;
    width: 20px;
}
.icon1 {
    background-position: 0px 0px;
}
.icon2 {
    background-position: -20px 0px;
}
.icon3 {
    background-position: -40px 0px;
}
```

Al usar la configuración del ancho y la altura del sprite y al usar la propiedad de posición de fondo en CSS (con un valor de x e y), puede extraer fácilmente sprites de una hoja de sprite usando CSS.

Lea Imagen de CSS Sprites en línea: <https://riptutorial.com/es/css/topic/3690/imagen-de-css-sprites>

Capítulo 34: Listar estilos

Sintaxis

- estilo de lista: tipo de estilo de lista | list-style-position | list-style-image | inicial | heredar;

Parámetros

Valor	Descripción
tipo de estilo de lista	el tipo de marcador de lista de elementos.
list-style-position	Especifica dónde colocar el marcador.
list-style-image	Especifica el tipo de marcador de lista de elementos.
inicial	establece esta propiedad a su valor predeterminado
heredar	hereda esta propiedad de su elemento padre

Observaciones

Aunque el `list-style-type` es en realidad una propiedad que se aplica solo a los elementos de lista (normalmente ``), a menudo se especifica para la etiqueta de lista (`` o ``). En este caso, los elementos de la lista heredan la propiedad.

Examples

Tipo de viñeta o numeración

Específico para etiquetas `` dentro de una lista no ordenada (``):

```
list-style: disc;          /* A filled circle (default) */
list-style: circle;        /* A hollow circle */
list-style: square;        /* A filled square */
list-style: '-';           /* any string */
```

Específico para etiquetas `` dentro de una lista ordenada (``):

```
list-style: decimal;       /* Decimal numbers beginning with 1 (default) */
list-style: decimal-leading-zero; /* Decimal numbers padded by initial zeros (01, 02, 03, ... 10)
*/
list-style: lower-roman;    /* Lowercase roman numerals (i., ii., iii., iv., ...) */
list-style: upper-roman;    /* Uppercase roman numerals (I., II., III., IV., ...) */
list-style-type: lower-greek; /* Lowercase roman letters (α., β., γ., δ., ...) */
list-style-type: lower-alpha; /* Lowercase letters (a., b., c., d., ...) */
```

```
list-style-type: lower-latin;      /* Lowercase letters (a., b., c., d., ...) */
list-style-type: upper-alpha;       /* Uppercase letters (A., B., C., D., ...) */
list-style-type: upper-latin;       /* Uppercase letters (A., B., C., D., ...) */
```

No específico:

```
list-style: none;                  /* No visible list marker */
list-style: inherit;               /* Inherits from parent */
```

Posición de la bala

Una lista consta de 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 predeterminados para el margen y el relleno pueden ser diferentes para cada navegador. Para obtener el mismo navegador cruzado de diseño, estos deben configurarse específicamente.

Cada elemento de la lista obtiene un 'cuadro de marcador', que contiene el marcador de viñeta. Este cuadro puede colocarse dentro o fuera del cuadro de elemento de 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 bala a la izquierda del elemento ``. Si no hay suficiente espacio en el relleno del elemento que lo contiene, el cuadro de marcador se extenderá hacia la izquierda, incluso si se cae de la página.

Mostrando el resultado de posicionamiento `inside` y `outside`: [jsfiddle](#)

Eliminando Balas / Números

A veces, una lista no debe mostrar ningún punto o número. En ese caso, recuerde especificar el margen y el relleno.

```
<ul>
  <li>first item</li>
  <li>second item</li>
</ul>
```

CSS

```
ul {
  list-style-type: none;
}
li {
```

```
margin: 0;  
padding: 0;  
}
```

Lea Listar estilos en línea: <https://riptutorial.com/es/css/topic/4215/listar-estilos>

Capítulo 35: Márgenes

Sintaxis

- margen: <arriba & derecha & abajo & izquierda> ;
- margen: <arriba> , <izquierda y derecha> , <fondo> ;
- margen: <arriba y abajo> , <izquierda y derecha> ;
- margen: <top> , <right> , <bottom> , <left> ;
- margin-top: <top> ;
- margen derecho: <right> ;
- margen inferior: <bottom> ;
- margen izquierdo: <left> ;

Parámetros

Parámetro	Detalles
0	establecer el margen a ninguno
auto	Se usa para centrar, estableciendo uniformemente los valores en cada lado
unidades (por ejemplo, px)	Vea la sección de parámetros en Unidades para una lista de unidades válidas
heredar	heredar el valor del margen del elemento padre
inicial	restaurar al valor inicial

Observaciones

Más sobre "Márgenes colapsados": [aquí](#).

Examples

Aplicar margen en un lado dado

Propiedades específicas de la dirección

CSS le permite especificar un lado dado para aplicar margen a. Las cuatro propiedades proporcionadas para este fin son:

- margin-left

- margin-right
- margin-top
- margin-bottom

El siguiente código aplicaría un margen de 30 píxeles al lado izquierdo de la división seleccionada. [Ver resultado](#)

HTML

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

CSS

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

Parámetro	Detalles
margin izquierdo	La dirección en la que se debe aplicar el margen.
30px	El ancho del margen.

Especificando la dirección usando la propiedad abreviada

La propiedad de `margin` estándar se puede expandir para especificar diferentes anchos a cada lado de los elementos seleccionados. La sintaxis para hacer esto es la siguiente:

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

El siguiente ejemplo aplica un margen de ancho cero en la parte superior del div, un margen de 10 píxeles en el lado derecho, un margen de 50 píxeles en el lado izquierdo y un margen de 100 píxeles en el lado izquierdo. [Ver resultado](#)

HTML

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

CSS

```
#myDiv {
  margin: 0 10px 50px 100px;
  height: 40px;
```

```
width: 40px;  
background-color: red;  
}
```

Colapso de margen

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

Ejemplo de márgenes verticales adyacentes:

Considere los siguientes estilos y marcas:

```
div{  
    margin: 10px;  
}
```

```
<div>  
    some content  
</div>  
<div>  
    some more content  
</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>some</span><span>content</span>
```

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

Superposición con diferentes tamaños.

```
.top{  
    margin: 10px;  
}  
.bottom{  
    margin: 15px;  
}
```

```
<div class="top">  
    some content
```

```
</div>
<div class="bottom">
    some more content
</div>
```

Estos elementos estarán separados 15px verticalmente. Los márgenes se superponen tanto como pueden, pero el margen más grande determinará el espacio entre los elementos.

Margen de superposición gotcha

```
.outer-top{
    margin: 10px;
}
.inner-top{
    margin: 15px;
}
.outer-bottom{
    margin: 20px;
}
.inner-bottom{
    margin: 25px;
}
```

```
<div class="outer-top">
    <div class="inner-top">
        some content
    </div>
</div>
<div class="outer-bottom">
    <div class="inner-bottom">
        some more content
    </div>
</div>
```

¿Cuál será el espacio entre los dos textos? (para ver la respuesta)

El espaciado será de 25px. Dado que los cuatro márgenes se tocan entre sí, se colapsarán, por lo que utilizarán el margen más grande de los cuatro.

Ahora, ¿qué pasa si agregamos algunos bordes a la marca de arriba.

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

¿Cuál será el espacio entre los dos textos? (para ver la respuesta)

El espacio será de 59px! Ahora solo los márgenes de .outer-top y .outer-bottom se tocan entre sí, y son los únicos márgenes colapsados. Los márgenes restantes están separados por los bordes. Así que tenemos 1px + 10px + 1px + 15px + 20px + 1px + 25px + 1px. (Los 1px son las fronteras ...)

Disminución de márgenes entre elementos padre e hijo:

HTML:

```
<h1>Title</h1>
<div>
  <p>Paragraph</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, solo se aplica el margen más grande. Es posible que haya esperado que el párrafo esté ubicado a 60 px del h1 (ya que el elemento div tiene un margen superior de 40 px y la p tiene un margen superior de 20 px). Esto no sucede porque los márgenes se colapsan juntos para formar un margen.

Centrar horizontalmente los elementos en una página utilizando el margen.

Mientras el elemento sea un **bloque** y tenga un **valor de ancho establecido explícitamente**, los márgenes se pueden usar para centrar los elementos de bloque en una página horizontalmente.

Agregamos un valor de ancho que es menor que el ancho de la ventana y la propiedad automática de margen luego distribuye el espacio restante a la izquierda y la derecha:

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

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

En el ejemplo anterior, el elemento #myDiv se establece en un 80% de ancho, lo que deja un 20% restante de uso. El navegador distribuye este valor a los lados restantes de modo que:

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

Simplificación de propiedades de margen

```

p {
    margin:1px;           /* 1px margin in all directions */

    /*equals to:*/
    margin:1px 1px;
    /*equals to:*/
    margin:1px 1px 1px;
    /*equals to:*/
    margin:1px 1px 1px 1px;
}

```

Otro ejemplo:

```

p{
    margin:10px 15px;      /* 10px margin-top & bottom And 15px margin-right & left*/

    /*equals to:*/
    margin:10px 15px 10px 15px;
    /*equals to:*/
    margin:10px 15px 10px;
    /* margin left will be calculated from the margin right value (=15px) */
}

```

Márgenes negativos

El margen es una de las pocas propiedades CSS que se pueden establecer en valores negativos. Esta propiedad se puede utilizar para **superponer elementos sin posicionamiento absoluto**.

```

div{
    display: inline;
}

#over{
    margin-left: -20px;
}

<div>Base div</div>
<div id="over">Overlapping div</div>

```

Ejemplo 1:

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

```

.parent {
    width : 500px;
    height: 300px;
}

```

```
}

.child {
    width : 100px;
    height: 100px;
    margin-left: 10%; /* (parentWidth * 10/100) => 50px */
}
```

Pero ese 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 principal, sino al **ancho** del contenedor principal.

Así que,

```
.parent {
    width : 500px;
    height: 300px;
}

.child {
    width : 100px;
    height: 100px;
    margin-left: 10%; /* (parentWidth * 10/100) => 50px */
    margin-top: 20%; /* (parentWidth * 20/100) => 100px */
}
```

Lea Márgenes en línea: <https://riptutorial.com/es/css/topic/305/margenes>

Capítulo 36: Mesas

Sintaxis

- diseño de la mesa: *auto* | fijo;
- colapso de la frontera: *separado* | colapso;
- espacio entre bordes: <length> | <length> <length>;
- células vacías: *mostrar* | esconder;
- lado del título: *arriba* | fondo;

Observaciones

Estas propiedades se aplican tanto a los elementos `<table>` (*) como a los elementos HTML mostrados como `display: table` O `display: inline-table`

(*) Los elementos `<table>` están obviamente diseñados de forma nativa por UA / browsers como `display: table`

Las tablas HTML son semánticamente válidas para datos tabulares. No se recomienda utilizar tablas para el diseño. En su lugar, utiliza CSS.

Examples

diseño de la mesa

La propiedad de `table-layout` cambia el algoritmo que se usa para el diseño de una tabla.

Debajo de un ejemplo de dos tablas, ambas configuradas al `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`. El primero es más ancho que el ancho especificado (210px en lugar de 150px) pero el contenido se ajusta. Este último toma el ancho definido de 150 px, independientemente de si el contenido se desborda o no.

Valor	Descripción
<code>auto</code>	Este es el valor predeterminado. Define el diseño de la tabla que se determinará por el contenido de sus celdas.
<code>fijo</code>	Este valor establece el diseño de la tabla que se determina por la propiedad de

Valor	Descripción
	ancho proporcionada a la tabla. Si el contenido de una celda excede este ancho, la celda no cambiará de tamaño, sino que dejará que el contenido se desborde.

colapso de la frontera

La propiedad `border-collapse` determina si los bordes de las tablas deben separarse o fusionarse.

Debajo de un ejemplo de dos tablas con valores diferentes a 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
separar	Este es el valor predeterminado. Hace que los bordes de la mesa se separen entre sí.
colapso	Este valor establece que los bordes de la tabla se fusionen, en lugar de ser distintos.

espaciado de la frontera

La propiedad de espacio entre `border-spacing` determina el espacio entre celdas. Esto no tiene ningún efecto a menos que `border-collapse` se establezca en `separate`.

Debajo de un ejemplo de dos tablas con valores diferentes a la propiedad de `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` (predeterminado), mientras que la de la derecha tiene `border-spacing: 8px`.

Valor	Descripción
<code><length></code>	Este es el comportamiento predeterminado, aunque el valor exacto puede variar entre los navegadores.

Valor	Descripción
<length> <length>	Esta sintaxis permite especificar valores horizontales y verticales separados respectivamente.

celdas vacías

La propiedad de las `empty-cells` determina si se deben mostrar o no las celdas sin contenido. Esto no tiene ningún efecto a menos que `border-collapse` se establezca en `separate`.

Debajo de un ejemplo con dos tablas con diferentes valores establecidos en la propiedad de `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 lo hace.

Valor	Descripción
espectáculo	Este es el valor predeterminado. Muestra celdas aunque estén vacías.
esconder	Este valor oculta una celda por completo si no hay contenidos en la celda.

Más información:

- <https://www.w3.org/TR/CSS21/tables.html#empty-cells>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/empty-cells>
- <http://codepen.io/SitePoint/pen/yfhtq>
- <https://css-tricks.com/almanac/properties/e/empty-cells/>

lado del título

La propiedad del `caption-side` determina la posición vertical del elemento `<caption>` dentro de una tabla. Esto no tiene efecto si tal elemento no existe.

Debajo de un ejemplo con dos tablas con diferentes valores establecidos en la propiedad del `caption-side` del `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
<i>parte superior</i>	Este es el valor predeterminado. Coloca la leyenda sobre la mesa.
fondo	Este valor coloca el título debajo de la tabla.

Lea Mesas en línea: <https://riptutorial.com/es/css/topic/1074/mesas>

Capítulo 37: Modelo de objetos CSS (CSSOM)

Observaciones

El Modelo de objetos CSS (CSSOM) es una especificación por sí misma.

El borrador actual se puede encontrar aquí: <https://www.w3.org/TR/cssom-1/>

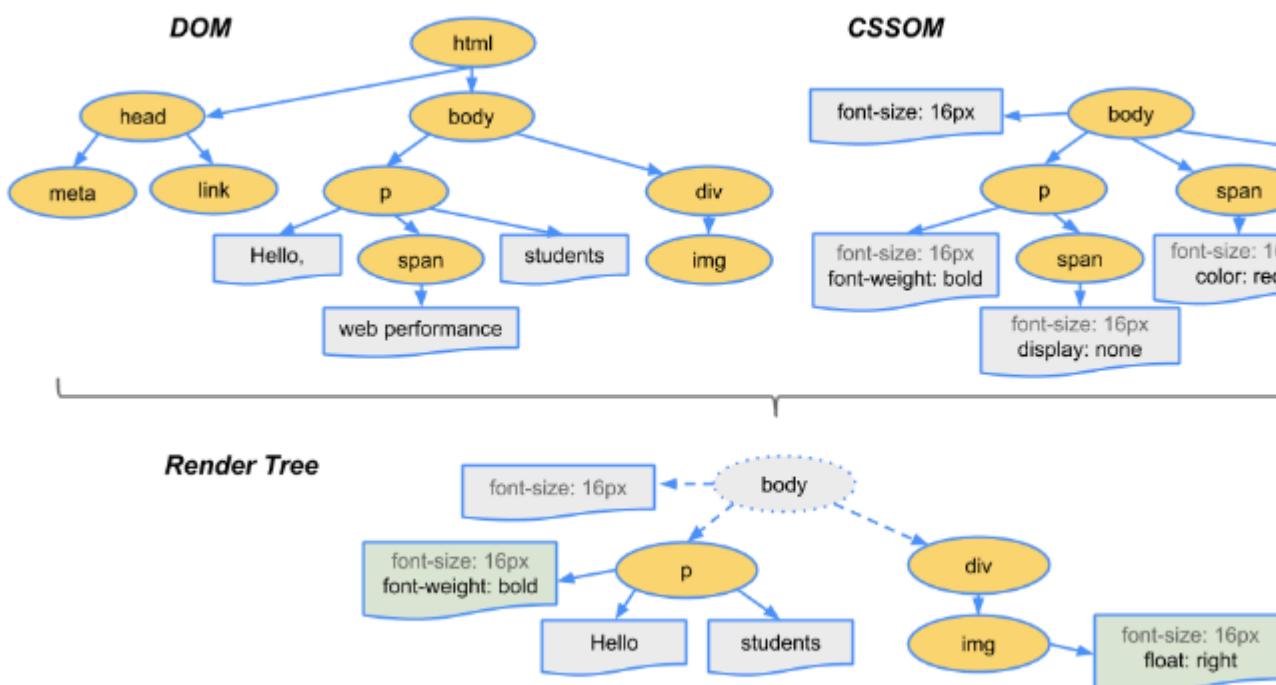
Examples

Introducción

El navegador identifica tokens de la hoja de estilo y los cubre en nodos que están vinculados 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 mostrar la página web, un navegador web realiza los siguientes pasos.

1. El navegador web examina su HTML y crea el DOM (Modelo de objetos de documento).
2. El navegador web examina su CSS y crea el CSSOM (Modelo de objetos CSS).
3. El navegador web combina el DOM y el CSSOM para crear un árbol de procesamiento. El navegador web muestra su página web.



Agregando una regla de imagen de fondo a través del CSSOM

Para agregar una regla de imagen de fondo a través del CSSOM, primero obtenga una referencia

a las reglas de la primera hoja de estilo:

```
var stylesheet = document.styleSheets[0].cssRules;
```

Luego, obtenga una referencia al final de la hoja de estilo:

```
var end = stylesheet.length - 1;
```

Finalmente, inserte una regla de imagen de fondo para el elemento del cuerpo al final de la hoja de estilo:

```
stylesheet.insertRule("body { background-image:  
url('http://cdn.sstatic.net/Sites/stackoverflow/img/favicon.ico'); }", end);
```

Lea **Modelo de objetos CSS (CSSOM)** en línea: <https://riptutorial.com/es/css/topic/4961/modelo-de-objetos-css--cssom->

Capítulo 38: Normalizar los estilos del navegador

Introducción

Cada navegador tiene un conjunto predeterminado de estilos CSS que utiliza para representar elementos. Es posible que estos estilos predeterminados no sean consistentes en todos los navegadores porque: las especificaciones de idioma no son claras, por lo que los estilos de base son fáciles de interpretar, los navegadores pueden no seguir las especificaciones que se proporcionan o los navegadores pueden no tener estilos predeterminados para los elementos HTML más nuevos. Como resultado, las personas pueden desear normalizar los estilos predeterminados en tantos navegadores como sea posible.

Observaciones

El Restablecimiento de Meyer, aunque es efectivo, realiza los mismos cambios en casi todos los elementos de uso común. Esto tiene el resultado de la contaminación de las ventanas del inspector del navegador web con los mismos estilos aplicados una y otra vez, y crea más trabajo para el navegador (más reglas para aplicar a más elementos). La técnica Normalizar, por otro lado, es mucho más enfocada y menos técnica de pincel ancho. Esto simplifica el trabajo en la parte del navegador y da como resultado menos desorden en las herramientas de inspección del navegador.

Examples

normalize.css

Los navegadores tienen un conjunto predeterminado de estilos CSS que utilizan para representar elementos. Algunos de estos estilos pueden incluso personalizarse utilizando la configuración del navegador para cambiar las definiciones de tamaño y cara de fuente predeterminadas, por ejemplo. Los estilos contienen la definición de qué elementos se supone que son a nivel de bloque o en línea, entre otras cosas.

Debido a que estos estilos predeterminados tienen un margen de maniobra por las especificaciones de idioma y debido a 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 inconsistencias más comunes y corrige errores conocidos.

Qué hace

- Conserva valores predeterminados útiles, a diferencia de muchos

restablecimientos de CSS.

- Normaliza los estilos para una amplia gama de elementos.
- Corrige errores e inconsistencias comunes del navegador.
- Mejora la usabilidad con modificaciones sutiles.
- Explica qué código hace usando comentarios detallados.

Por lo tanto, al incluir normalize.css en su proyecto, su diseño se verá más parecido y consistente en diferentes navegadores.

Diferencia a reset.css

Es posible que haya oido hablar de `reset.css`. ¿Cuál es la diferencia entre los dos?

Mientras que normalize.css proporciona consistencia al establecer diferentes propiedades a valores predeterminados unificados, reset.css logra consistencia al **eliminar** todo el estilo básico que puede aplicar un navegador. Si bien esto puede parecer una buena idea al principio, en realidad esto significa que debe escribir **todas las** reglas, lo que va en contra de tener un estándar sólido.

Enfoques y ejemplos

Los restablecimientos de CSS toman enfoques separados para los valores predeterminados del navegador. El restablecimiento de CSS de Eric Meyer ha existido por un tiempo. Su enfoque anula muchos de los elementos del navegador que se sabe que causan problemas desde el principio. Lo 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;
}
```

Restablecer CSS de Eric Meyer

Normaliza CSS en el otro y trata muchos de estos por separado. La siguiente es una muestra 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;
}

```

Normalizar CSS

Lea Normalizar los estilos del navegador en línea:

<https://riptutorial.com/es/css/topic/1211/normalizar-los-estilos-del-navegador>

Capítulo 39: Opacidad

Sintaxis

- opacidad: número (* estrictamente entre 0 y 1) | heredar | inicial | desarmado

Observaciones

Si no desea aplicar opacidad, puede usar esto en su lugar:

fondo: rgba (255, 255, 255, 0.6);

Recursos:

- MDN: <https://developer.mozilla.org/en/docs/Web/CSS/opacity> ;
- Transparencia W3C: la propiedad 'opacidad': <https://www.w3.org/TR/css3-color/#transparency>
- Soporte del navegador: <http://caniuse.com/#feat=css-opacity>

Examples

Propiedad de opacidad

La opacidad de un elemento se puede establecer utilizando la propiedad `opacity`. Los valores pueden ser desde `0.0` (transparente) hasta `1.0` (opaco).

Ejemplo de uso

```
<div style="opacity:0.8;">  
    This is a partially transparent element  
</div>
```

Valor de 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

Compatibilidad de IE para la 'opacidad'

Para usar la `opacity` en todas las versiones de IE, el orden es:

```
.transparent-element {  
    /* for IE 8 & 9 */  
    -ms-filter:"progid:DXImageTransform.Microsoft.Alpha(Opacity=60)"; // IE8  
    /* works in IE 8 & 9 too, but also 5, 6, 7 */  
    filter: alpha(opacity=60); // IE 5-7  
    /* Modern Browsers */  
    opacity: 0.6;  
}
```

Lea Opacidad en línea: <https://riptutorial.com/es/css/topic/2864/opacidad>

Capítulo 40: Patrones de diseño CSS

Introducción

Estos ejemplos son para documentar patrones de diseño específicos de CSS como [BEM](#) , [OOCSS](#) y [SMACSS](#) .

Estos ejemplos NO son para documentar marcos CSS como [Bootstrap](#) o [Foundation](#) .

Observaciones

Estos ejemplos son para documentar metodologías / patrones de diseño específicos de CSS.

Estas metodologías incluyen pero no son exclusivas de las siguientes:

- [BEM](#)
- [OOCSS](#)
- [SMACSS](#)

Estos ejemplos NO son para documentar marcos CSS como [Bootstrap](#) o [Foundation](#) . Si bien puede incluir ejemplos de cómo aplicar uno o más patrones de metodología / diseño de CSS con un marco de trabajo de CSS, estos ejemplos se centran en las metodologías / patrones de diseño con ese marco en particular y en el uso del propio marco.

Examples

BEM

[BEM](#) significa `Blocks, Elements and Modifiers` . Es una metodología inicialmente concebida por la compañía de tecnología rusa [Yandex](#) , pero que también ganó bastante tracción entre los desarrolladores web estadounidenses y de Europa occidental.

Como lo mismo implica, la metodología BEM se basa en la creación de componentes de su código HTML y CSS en tres tipos de componentes:

- **Bloques:** entidades independientes que son significativas por sí mismas

Los ejemplos son `header` , `container` , `menu` , `checkbox` y `textbox`

- **Elementos:** parte de los bloques que no tienen un significado independiente y están ligados semánticamente a sus bloques.

Los ejemplos son: `menu item` , `menu item list item` , `checkbox caption` y `header title`

- **Modificadores:** indicadores en un bloque o elemento, utilizados para cambiar la apariencia o el comportamiento

El objetivo de BEM es mantener la optimización de la legibilidad, la capacidad de mantenimiento y la flexibilidad de su código CSS. La forma de lograrlo, es aplicar las siguientes reglas.

- Los estilos de bloque nunca dependen de otros elementos en una página
- Los bloques deben tener un nombre corto y simple y evitar _ o - caracteres
- Cuando estilice elementos, use selectores de formato `blockname__elementname`
- Cuando `blockname--modifiername` estilo, use selectores de formato nombre de `blockname--modifiername` y nombre de `blockname__elementname--modifiername`
- Los elementos o bloques que tienen modificadores deben heredar todo del bloque o elemento que está modificando, excepto las propiedades que el modificador debe modificar.

Ejemplo de código

Si aplica BEM a los elementos de su formulario, los selectores de CSS deberían tener este aspecto:

```
.form { }                                // Block
.form--theme-xmas { }                      // Block + modifier
.form--simple { }                          // Block + modifier
.form__input { }                           // Block > element
.form__submit { }                          // Block > element
.form__submit--disabled { }                // Block > element + modifier
```

El HTML correspondiente debe verse algo como esto:

```
<form class="form form--theme-xmas form--simple">
  <input class="form__input" type="text" />
  <input class="form__submit form__submit--disabled" type="submit" />
</form>
```

Lea Patrones de diseño CSS en línea: <https://riptutorial.com/es/css/topic/10823/patrones-de-diseno-css>

Capítulo 41: Posicionamiento

Sintaxis

- posición: estática | absoluta | fija | relativa | pegajosa | inicial | heredar | unset;
- índice z: auto | *número* | inicial | heredar;

Parámetros

Parámetro	Detalles
estático	Valor por defecto. Los elementos se procesan en orden, tal como aparecen en el flujo de documentos. Las propiedades superior, derecha, inferior, izquierda e índice z no se aplican.
relativo	El elemento se posiciona en relación con su posición normal, por lo que a la <code>left: 20px</code> agrega 20 píxeles a la posición IZQUIERDA del elemento
fijo	El elemento se posiciona en relación a la ventana del navegador.
absoluto	El elemento se posiciona en relación con su primer elemento ancestral posicionado (no estático)
inicial	Establece esta propiedad a su valor predeterminado.
heredar	Hereda esta propiedad de su elemento padre.
pegajoso	Característica experimental. Se comporta como <code>position: static</code> dentro de su padre hasta que se alcanza un umbral de desplazamiento dado, entonces actúa como <code>position: fixed</code> .
desarmado	Combinación de inicial y heredar. Más información aquí .

Observaciones

Flujo normal es el flujo de elementos si la posición del elemento es *estática* .

1. definir el *ancho* es beneficioso porque en algunos casos evita la superposición del contenido del elemento.

Examples

Posición fija

Definiendo la posición como fija, podemos eliminar un elemento del flujo de documentos y establecer su posición en relación con 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;  
}
```

Elementos superpuestos con índice z

Para cambiar el [orden](#) predeterminado de la [pila de](#) elementos posicionados (propiedad de `position` establecida en `relative`, `absolute` o `fixed`), use la propiedad `z-index`.

Cuanto más alto sea el índice z, más arriba estará en el contexto de apilamiento (en el eje z).

Ejemplo

En el siguiente ejemplo, un valor de índice z de 3 pone verde en la parte superior, un índice z de 2 pone rojo justo debajo de él, y un índice z de 1 pone azul debajo de eso.

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:



Vea un ejemplo de trabajo en [JSFiddle](#).

Sintaxis

```
z-index: [ number ] | auto;
```

Parámetro Detalles

number	Un valor entero. Un número más alto es más alto en la pila del <code>z-index</code> . 0 es el valor predeterminado. Se permiten valores negativos.
--------	----------------------------------------------------------------------------------------------------------------------------------------------------

auto	Da al elemento el mismo contexto de apilamiento que su parente. (Predeterminado)
------	-------------------------------------------------------------------------------------------

Observaciones

Todos los elementos se presentan en un eje 3D en CSS, incluido un eje de profundidad, medido por la propiedad `z-index`. `z-index` solo funciona con elementos posicionados: (ver: [¿Por qué z-](#)

[index necesita una posición definida para trabajar?](#)). El único valor donde se ignora es el valor predeterminado, `static`.

Lea sobre la propiedad de índice z y los contextos de apilamiento en la [Especificación de CSS](#) en presentaciones por capas y en la [Red de desarrolladores de Mozilla](#).

Posición relativa

El posicionamiento relativo mueve el elemento en relación a donde habría estado en *el flujo normal*. Propiedades compensadas:

1. parte superior
2. izquierda
3. Correcto
4. fondo

se utilizan para indicar a qué distancia se debe mover el elemento desde donde habría estado en flujo normal.

```
.relpos{  
    position:relative;  
    top:20px;  
    left:30px;  
}
```

Este código moverá el cuadro que contiene el elemento con el atributo class = "relpos" 20px hacia abajo y 30px hacia la derecha desde donde habría estado en el flujo normal.

Posición absoluta

Cuando se utiliza el posicionamiento absoluto, el cuadro del elemento deseado se saca del *flujo normal* y ya no afecta a la posición de los otros elementos en la página. Propiedades de compensación:

1. parte superior
2. izquierda
3. Correcto
4. fondo

Especifique que el elemento debe aparecer en relación con su siguiente elemento que no es estático.

```
.abspos{  
    position:absolute;  
    top:0px;  
    left:500px;  
}
```

Este código moverá el cuadro que contiene el elemento con el atributo `class="abspos"` hacia abajo 0px y 500px a la derecha en relación con su elemento que contiene.

Posicionamiento estático

La posición predeterminada de un elemento es `static`. Para citar [MDN](#):

Esta palabra clave permite que el elemento utilice el comportamiento normal, es decir, se presenta en su posición actual en el flujo. Las propiedades superior, derecha, inferior, izquierda e índice z no se aplican.

```
.element{  
  position:static;  
}
```

Lea Posicionamiento en línea: <https://riptutorial.com/es/css/topic/935/posicionamiento>

Capítulo 42: Preguntas de los medios

Sintaxis

- @media [no | solo] tipo de medio y (función de medios) {/* reglas de CSS para aplicar */}

Parámetros

Parámetro	Detalles
mediatype	(Opcional) Este es el tipo de medio. Podría ser cualquier cosa en el rango de <code>all</code> a la <code>screen</code> .
not	(Opcional) No aplica el CSS para este tipo de medio en particular y se aplica para todo lo demás.
media feature	Lógica para identificar caso de uso para CSS. Opciones que se describen a continuación.
Característica de los medios	Detalles
aspect-ratio	Describe la relación de aspecto del área de visualización seleccionada del dispositivo de salida.
color	Indica el número de bits por componente de color del dispositivo de salida. Si el dispositivo no es un dispositivo de color, este valor es cero.
color-index	Indica el número de entradas en la tabla de búsqueda de colores para el dispositivo de salida.
grid	Determina si el dispositivo de salida es un dispositivo de cuadrícula o un dispositivo de mapa de bits.
height	La función de medios de altura describe la altura de la superficie de representación del dispositivo de salida.
max-width	CSS no se aplicará en un ancho de pantalla más ancho que el especificado.
min-width	CSS no se aplicará en un ancho de pantalla más estrecho que el especificado.
max-height	CSS no se aplicará en una altura de pantalla más alta que la especificada.

Parámetro	Detalles
min-height	CSS no se aplicará en una altura de pantalla más corta que la especificada.
monochrome	Indica el número de bits por píxel en un dispositivo monocromo (escala de grises).
orientation	CSS solo se mostrará si el dispositivo está utilizando la orientación especificada. Ver comentarios para más detalles.
resolution	Indica la resolución (densidad de píxeles) del dispositivo de salida.
scan	Describe el proceso de escaneo de los dispositivos de salida de televisión.
width	La función de ancho de medios describe el ancho de la superficie de representación del dispositivo de salida (como el ancho de la ventana del documento o el ancho del cuadro de página en una impresora).
Características en desuso	Detalles
device-aspect-ratio	CSS en Deprecated solo se mostrará en dispositivos cuya relación altura / anchura coincide con la proporción especificada. Esta es una característica deprecated y no se garantiza que funcione.
max-device-width	Deprecated Igual que <code>max-width</code> pero mide el ancho físico de la pantalla, en lugar del ancho de visualización del navegador.
min-device-width	Deprecated Igual que <code>min-width</code> pero mide el ancho físico de la pantalla, en lugar del ancho de visualización del navegador.
max-device-height	Deprecated Igual que <code>max-height</code> pero mide el ancho físico de la pantalla, en lugar del ancho de visualización del navegador.
min-device-height	Deprecated Igual que <code>min-height</code> pero mide el ancho físico de la pantalla, en lugar del ancho de visualización del navegador.

Observaciones

Las consultas de medios son compatibles con todos los navegadores modernos, incluidos Chrome, Firefox, Opera e Internet Explorer 9 y superiores.

Es importante tener en cuenta que la función de medios de `orientation` no se limita a los dispositivos móviles. Se basa en el ancho y el alto de la ventana gráfica (no de la ventana o los dispositivos).

El modo horizontal es cuando el ancho de la ventana gráfica es mayor que la altura de la ventana gráfica.

El modo vertical es cuando la altura de la ventana gráfica es mayor que el ancho de la ventana gráfica.

Esto generalmente se traduce en un monitor de escritorio en modo horizontal, pero a veces puede ser vertical.

En la mayoría de los casos, los dispositivos móviles informarán su resolución y no su tamaño real de píxeles, que puede diferir debido a la densidad de píxeles. Para obligarlos a informar su tamaño real de píxeles, agregue lo siguiente dentro de la etiqueta de su `head`:

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

Examples

Ejemplo básico

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

La consulta de medios 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 consulta de medios estarán activos y el color de fondo de la página será azul cielo.

Las consultas de medios se aplican dinámicamente. Si en la carga de la página se cumplen las condiciones especificadas en la consulta de medios, se aplicará el CSS, pero se desactivará de inmediato si las condiciones dejan de cumplirse. A la inversa, si las condiciones no se cumplen inicialmente, 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 mayor a 720 píxeles, pero el usuario reduce el ancho del navegador, el color de fondo dejará de ser azul cielo tan pronto como el usuario haya cambiado el tamaño del puerto de vista a menos de 720 píxeles. anchura.

Usar en la etiqueta de enlace

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

Esta hoja de estilo aún se descarga, pero se aplica solo en dispositivos con un ancho de pantalla superior a 600 px.

tipo de medio

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

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

El código CSS anterior le dará al elemento `HTML` DOM un color de fondo blanco cuando se imprima.

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

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

Y de la misma forma, para mostrarlo solo en la pantalla, se puede utilizar:

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

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

Tipo de medio	Descripción
all	Aplicar a todos los dispositivos
screen	Computadoras predeterminadas
print	Impresoras en general. Se utiliza para diseñar versiones impresas de sitios web.
handheld	PDA's, celulares y dispositivos de mano con una pequeña pantalla.
projection	Para presentación proyectada, por ejemplo proyectores.

Tipo de medio	Descripción
aural	Sistemas del habla
braille	Aparatos táctiles braille
embossed	Impresoras braille paginadas
tv	Aparatos tipo televisión
tty	Dispositivos con una rejilla de caracteres de paso fijo. Terminales, portátiles.

Uso de consultas de medios para identificar diferentes tamaños de pantalla

Muchas veces, el diseño web sensible implica consultas de medios, que son bloques de CSS que solo se ejecutan si se cumple una condición. Esto es útil para el diseño web sensible porque puede usar consultas de medios para especificar diferentes estilos CSS para la versión móvil de su sitio web en lugar de la versión de escritorio.

```
@media only screen and (min-width: 300px) and (max-width: 767px) {
    .site-title {
        font-size: 80%;
    }

    /* Styles in this block are only applied if the screen size is atleast 300px wide, but no
       more than 767px */
}

@media only screen and (min-width: 768px) and (max-width: 1023px) {
    .site-title {
        font-size: 90%;
    }

    /* Styles in this block are only applied if the screen size is atleast 768px wide, but no
       more than 1023px */
}

@media only screen and (min-width: 1024px) {
    .site-title {
        font-size: 120%;
    }

    /* Styles in this block are only applied if the screen size is over 1024px wide. */
}
```

Ancho vs Viewport

Cuando usamos "ancho" con consultas de medios, es importante configurar la metaetiqueta correctamente. La metaetiqueta básica se ve así y debe colocarse dentro de la etiqueta `<head>`.

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

¿Por qué esto es importante?

Basado en la definición de "ancho" de MDN es

La función de ancho de medios describe el ancho de la superficie de representación del dispositivo de salida (como el ancho de la ventana del documento o el ancho del cuadro de página en una impresora).

Qué significa eso?

View-port es el ancho del dispositivo en sí. Si la resolución de su pantalla dice que la resolución es 1280 x 720, el ancho del puerto de visualización es "1280px".

Más a menudo muchos dispositivos asignan diferentes cantidades de píxeles para mostrar un píxel. Por ejemplo, el iPhone 6 Plus tiene una resolución de 1242 x 2208. Pero el ancho de la ventana gráfica y la altura de la ventana gráfica son 414 x 736. Eso significa que se utilizan 3 píxeles para crear 1 píxel.

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

Consultas de medios para pantallas de retina y no retina

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

```
/* ----- Non-Retina Screens ----- */
@media screen
  and (min-width: 1200px)
  and (max-width: 1600px)
  and (-webkit-min-device-pixel-ratio: 1) {

}

/* ----- Retina Screens ----- */
@media screen
  and (min-width: 1200px)
  and (max-width: 1600px)
  and (-webkit-min-device-pixel-ratio: 2)
  and (min-resolution: 192dpi) {
```

Información de fondo

Hay dos tipos de píxeles en la pantalla. Uno es los píxeles lógicos y el otro son los píxeles físicos. En su mayoría, los píxeles físicos siempre permanecen iguales, porque es el mismo para todos los dispositivos de visualización. Los píxeles lógicos cambian según la resolución de los dispositivos para mostrar píxeles de mayor calidad. La proporción de píxeles del dispositivo es la proporción entre los píxeles físicos y los píxeles lógicos. Por ejemplo, el MacBook Pro Retina, iPhone 4 y superior informa una proporción de píxeles del dispositivo de 2, porque la resolución lineal física es el doble de la resolución lógica.

La razón por la que esto funciona solo con los 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.

Terminología y estructura

Las **consultas de medios** le permiten a uno aplicar reglas de CSS según el tipo de dispositivo / medios (por ejemplo, pantalla, impresión o computadora de mano) llamado **tipo de medios**, los aspectos adicionales del dispositivo se describen con **características de medios** tales como la disponibilidad de color o las dimensiones de la ventana gráfica.

Estructura general de una consulta de medios

```
@media [...] {  
    /* One or more CSS rules to apply when the query is satisfied */  
}
```

Una consulta de medios que contiene un tipo de medio

```
@media print {  
    /* One or more CSS rules to apply when the query is satisfied */  
}
```

Una consulta de medios que contiene un tipo de medio y una característica de medios

```
@media screen and (max-width: 600px) {  
    /* One or more CSS rules to apply when the query is satisfied */  
}
```

Una consulta de medios que contiene una característica de medios (y un tipo de medios implícito de "todos")

```
@media (orientation: portrait) {  
    /* One or more CSS rules to apply when the query is satisfied */  
}
```

}

Consultas de medios e IE8

Las consultas de medios no son compatibles en absoluto en IE8 y más abajo.

Una solución basada en Javascript

Para agregar soporte para IE8, puede usar una de varias soluciones JS. Por ejemplo, [Responder](#) se puede agregar para agregar soporte de consulta de medios para IE8 solo con el siguiente código:

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

[CSS Mediaqueries](#) es otra biblioteca que hace lo mismo. El código para agregar esa biblioteca a su HTML sería idéntico:

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

La alternativa

Si no le gusta una solución basada en JS, también debe considerar agregar una hoja de estilo IE <9 solo donde ajuste su estilo específico a IE <9. Para eso, debes agregar el siguiente HTML a tu 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 CSS](#) para apuntar a IE <9. Tiene el mismo impacto que una hoja de estilo IE <9, pero no necesita una hoja de estilo separada para eso. Sin embargo, no recomiendo esta opción, ya que producen un código CSS no válido (que es solo una de las varias razones por las que el uso de hacks CSS generalmente está mal visto hoy).

Lea Preguntas de los medios en línea: <https://riptutorial.com/es/css/topic/317/preguntas-de-los-medios>

Capítulo 43: Propiedad de filtro

Sintaxis

- filtro: ninguno (valor predeterminado)
- filtro: inicial (predeterminado en ninguno);
- filtro: heredar (por defecto al valor principal);
- filtro: desenfoque (px)
- filtro: brillo (numero |%)
- filtro: contraste (número |%)
- filtro: sombra-sombra (sombra horizontal-sombra-vertical-sombra-px sombra-desenfoque-sombra-sombra - color de propagación)
- filtro: escala de grises (número |%)
- filtro: hue-rotate (grados)
- filtro: invertido (número |%)
- filtro: opacidad (número |%)
- filtro: saturar (número |%)
- filtro: sepia (numero |%)

Parámetros

Valor	Descripción
difuminar (x)	Difumina la imagen por x píxeles.
brillo (x)	Ilumina la imagen en cualquier valor por encima de 1.0 o 100%. Debajo de eso, la imagen se oscurecerá.
contraste (x)	Proporciona más contraste con la imagen en cualquier valor por encima de 1.0 o 100%. Debajo de eso, la imagen se saturará menos.
sombra paralela (h, v, x, y, z)	Da a la imagen una sombra paralela. h y v pueden tener valores negativos. x, y, yz son opcionales.
escala de grises (x)	Muestra la imagen en escala de grises, con un valor máximo de 1.0 o 100%.
hue-rotate (x)	Aplica una rotación de tono a la imagen.
invertir (x)	Invierte el color de la imagen con un valor máximo de 1.0 o 100%.
opacidad (x)	Establece la opacidad / transparencia de la imagen con un valor máximo de 1.0 o 100%.
saturar (x)	Satura la imagen en cualquier valor por encima de 1.0 o 100%. Debajo de eso, la imagen comenzará a desaturarse.

Valor	Descripción
sepia (x)	Convierte la imagen en sepia con un valor máximo de 1.0 o 100%.

Observaciones

1. Dado que el filtro es una función experimental, debe usar el prefijo -webkit. Puede cambiar la sintaxis y el comportamiento, pero los cambios probablemente serán pequeños.
2. Puede que no sea compatible con versiones anteriores de los principales navegadores. Puede que no sea totalmente compatible con los navegadores móviles.
3. Debido a su soporte relativamente limitado, intente usar `box-shadow` lugar de `filter: drop-shadow()`. Usa la `opacity` lugar del `filter: opacity()`.
4. Se puede animar a través de Javascript / jQuery. Para Javascript, use `object.style.WebkitFilter`.
5. Compruebe [W3Schools](#) o [MDN](#) para más información.
6. W3Schools también tiene una [página de demostración](#) para todos los diferentes tipos de valores de filtro.

Examples

Sombra paralela (usa box-shadow si es posible)

HTML

```
<p>My shadow always follows me.</p>
```

CSS

```
p {
  -webkit-filter: drop-shadow(10px 10px 1px green);
  filter: drop-shadow(10px 10px 1px green);
}
```

Resultado

My shadow always follows me.
My shadow always follows me.

Valores de filtro múltiples

Para usar múltiples filtros, separe cada valor con un espacio.

HTML

```
<img src='donald-duck.png' alt='Donald Duck' title='Donald Duck' />
```

CSS

```
img {  
  -webkit-filter: brightness(200%) grayscale(100%) sepia(100%) invert(100%);  
  filter: brightness(200%) grayscale(100%) sepia(100%) invert(100%);  
}
```

Resultado



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



Invertir color

HTML

```
<div></div>
```

CSS

```
div {  
    width: 100px;  
    height: 100px;  
    background-color: white;  
    -webkit-filter: invert(100%);  
    filter: invert(100%);  
}
```

Resultado



Vuelve de blanco a negro.

Difuminar

HTML

```
<img src='donald-duck.png' alt='Donald Duck' title='Donald Duck' />
```

CSS

```
img {  
    -webkit-filter: blur(1px);  
    filter: blur(1px);  
}
```

Resultado



Te hace querer frotar tus gafas.

Lea Propiedad de filtro en línea: <https://riptutorial.com/es/css/topic/1567/propiedad-de-filtro>

Capítulo 44: Propiedades personalizadas (variables)

Introducción

Las Variables CSS permiten a los autores crear valores reutilizables que se pueden usar en un documento CSS.

Por ejemplo, es común en CSS reutilizar un solo color a lo largo de un documento. Antes de las Variables CSS, esto significaría reutilizar el mismo valor de color muchas veces en un documento. Con las Variables CSS, el valor del color se puede asignar a una variable y se puede hacer referencia en varios lugares. Esto facilita el cambio de valores y es más semántico que el uso de valores CSS tradicionales.

Sintaxis

- : pseudo-clase raíz {} /* que permite una definición más global de las variables */
- --nombre-variable: *valor*; /* Definir variable */
- var (- nombre-variable, *valor predeterminado*) /* usa la variable definida con un valor predeterminado de reserva */

Observaciones

Las variables CSS son actualmente consideradas una tecnología experimental.

APOYO / COMPATIBILIDAD DEL NAVEGADOR

Firefox: Versión 31+ (*Habilitado de forma predeterminada*)

[Más información de Mozilla](#)

Chrome: Versión 49+ (*habilitada por defecto*).

"*Esta función se puede habilitar en Chrome Versión 48 para realizar pruebas al habilitar la función de experimental Web Platform. Ingrese chrome://flags/ en su barra de direcciones de Chrome para acceder a esta configuración*".

IE: No es compatible.

Edge: [En desarrollo](#)

Safari: Versión 9.1+

Examples

Color variable

```
:root {  
  --red: #b00;  
  --blue: #4679bd;  
  --grey: #ddd;  
}  
.Bx1 {  
  color: var(--red);  
  background: var(--grey);  
  border: 1px solid var(--red);  
}
```

Dimensiones variables

```
:root {  
  --W200: 200px;  
  --W10: 10px;  
}  
.Bx2 {  
  width: var(--W200);  
  height: var(--W200);  
  margin: var(--W10);  
}
```

Cascada variable

Las variables CSS caen en cascada de la misma manera que otras propiedades, y pueden actualizarse de forma segura.

Puede definir variables varias veces y solo la definición con la mayor especificidad se aplicará al elemento seleccionado.

Asumiendo este HTML:

```
<a class="button">Button Green</a>  
<a class="button button_red">Button Red</a>  
<a class="button">Button Hovered On</a>
```

Podemos escribir este CSS:

```
.button {  
  --color: green;  
  padding: .5rem;  
  border: 1px solid var(--color);  
  color: var(--color);  
}  
.button:hover {  
  --color: blue;
```

```
}
```



```
.button_red {
```

```
  --color: red;
```

```
}
```

Y consigue este resultado:



Válido / Inválido

Nombrar Al nombrar las variables CSS, contiene solo letras y guiones al igual que otras propiedades de CSS (por ejemplo: altura de línea, tamaño de caja de mazmorras), pero debe comenzar con guiones dobles (-)

```
//These are Invalids variable names
```

```
--123color: blue;
```

```
--#color: red;
```

```
--bg_color: yellow
```

```
--$width: 100px;
```



```
//Valid variable names
```

```
--color: red;
```

```
--bg-color: yellow
```

```
--width: 100px;
```

Las variables CSS distinguen entre mayúsculas y minúsculas.

```
/* The variable names below are all different variables */
```

```
--pcolor: ;
```

```
--Pcolor: ;
```

```
--pColor: ;
```

Espacio Vs vacío

```
/* Invalid */
--color:;
```



```
/* Valid */
--color: ; /* space is assigned */
```

Concatenaciones

```
/* Invalid - CSS doesn't support concatenation*/
.logo{
  --logo-url: 'logo';
  background: url('assets/img/' var(--logo-url) '.png');
}

/* Invalid - CSS bug */
```

```

.logo{
    --logo-url: 'assets/img/logo.png';
    background: url(var(--logo-url));
}

/* Valid */
.logo{
    --logo-url: url('assets/img/logo.png');
    background: var(--logo-url);
}

```

Cuidado al usar Unidades

```

/* Invalid */
--width: 10;
width: var(--width)px;

/* Valid */
--width: 10px;
width: var(--width);

/* Valid */
--width: 10;
width: calc(1px * var(--width)); /* multiply by 1 unit to convert */
width: calc(1em * var(--width));

```

Con consultas de medios.

Puede restablecer las variables dentro de las consultas de los medios y hacer que esos nuevos valores aparezcan en cascada donde sea que se usen, algo que no es posible con las variables del preprocesador.

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: 'This is desktop';
}

@media only screen and (max-width: 767px) {
    :root{
        --width:50%;
        --content: 'This is mobile';
    }
}

@media only screen and (max-width: 480px) {
}

```

```
:root{  
    --width:100%;  
}  
}  
  
div{  
    width: calc(var(--width) - 20px);  
    height: 100px;  
}  
div:before{  
    content: var(--content);  
}  
  
/* Other Styles */  
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 intentar cambiar el tamaño de la ventana en esta [demostración de CodePen](#)

Aquí hay una captura de pantalla animada del cambio de tamaño en acción:

This is desktop

This is desktop

This is desktop

This is desktop

Lea Propiedades personalizadas (variables) en línea:

<https://riptutorial.com/es/css/topic/1755/propiedades-personalizadas--variables>

Capítulo 45: Pseudoelementos

Introducción

Los pseudo-elementos, al igual que las pseudo-clases, se agregan a los selectores de CSS, pero en lugar de describir un estado especial, le permiten ubicar y diseñar ciertas partes de un elemento html.

Por ejemplo, el :: pseudo-elemento de primera letra se dirige solo a la primera letra de un elemento de bloque especificado por el selector.

Sintaxis

- selector :: pseudo-elemento {propiedad: valor}

Parámetros

pseudo-elemento	Descripción
::after	Insertar contenido después del contenido de un elemento.
::before	Insertar contenido antes del contenido de un elemento.
::first-letter	Selecciona la primera letra de cada elemento.
::first-line	Selecciona la primera línea de cada elemento.
::selection	Coincide con la parte de un elemento que es seleccionado por un usuario
::backdrop	Se utiliza para crear un fondo que oculta el documento subyacente para un elemento en la pila de la capa superior
::placeholder	Le permite aplicar estilo al texto del marcador de posición de un elemento de formulario (Experimental)
::marker	Para aplicar atributos de estilo de lista en un elemento dado (Experimental)
::spelling-error	Representa un segmento de texto que el navegador ha marcado como incorrectamente escrito (Experimental)
::grammar-error	Representa un segmento de texto que el navegador ha marcado como gramaticalmente incorrecto (Experimental)

Observaciones

- Usted verá a veces dos puntos dobles (::) en lugar de sólo uno (:). Esta es una manera de separar pseudo-clases de pseudo-elementos, pero algunos navegadores antiguos como Internet Explorer 8 **sólo** es compatible con el colon solo (:) para los pseudo-elementos.
- Uno puede usar solo un pseudo-elemento en un selector. Debe aparecer después de los selectores simples en la declaración.
- Los pseudo-elementos no son parte del DOM, por lo tanto, no son direccionables por :hover u otros eventos de usuario.

Examples

Pseudoelementos

Los pseudo-elementos se agregan a los selectores, pero en lugar de describir un estado especial, le permiten diseñar ciertas partes de un documento.

El atributo de `content` es necesario para que los pseudo-elementos se representen; 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;  
}
```

before div element after

Pseudo-elementos en las listas

Los pseudo-elementos se usan a menudo para cambiar el aspecto de las listas (principalmente para listas desordenadas, `ul`).

El primer paso es eliminar las viñetas de lista predeterminadas:

```
ul {
```

```
list-style-type: none;  
}
```

A continuación, agregue el estilo personalizado. En este ejemplo, crearemos cuadros de degradado para 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



Test I
Test II

Lea Pseudoelementos en línea: <https://riptutorial.com/es/css/topic/911/pseudoelementos>

Capítulo 46: Rebosar

Sintaxis

- desbordamiento: visible | oculto | desplazarse auto | inicial | heredar;

Parámetros

Valor de Overflow	Detalles
visible	Muestra todo el contenido desbordado fuera del elemento.
scroll	Oculta el contenido desbordado y agrega una barra de desplazamiento.
hidden	Oculta el contenido desbordado, ambas barras de desplazamiento desaparecen y la página se arregla
auto	Igual que scroll si el contenido se desborda, pero no agrega la barra de desplazamiento si el contenido se ajusta
inherit	Hereda el valor del elemento padre para esta propiedad.

Observaciones

La propiedad de `overflow` especifica si se debe recortar el contenido, representar barras de desplazamiento o estirar un contenedor para mostrar el contenido cuando se desborda su contenedor de nivel de bloque.

Cuando un elemento es demasiado pequeño para mostrar su contenido, ¿qué sucede? De forma predeterminada, el contenido solo se desbordará y se mostrará fuera del elemento. Eso hace que tu trabajo se vea mal. Desea que su trabajo se vea bien, así que configura la propiedad de desbordamiento para manejar el contenido desbordado de una manera deseable.

Los valores para la propiedad de `overflow` son idénticos a los de las propiedades `overflow-x` y `overflow-y`, excepto que se aplican a lo largo de cada eje

La propiedad `overflow-wrap` también se conoce como la propiedad `word-wrap`.

Nota importante: el uso de la propiedad de desbordamiento con un valor diferente al `visible` creará un nuevo contexto de formato de bloque.

Examples

desbordamiento: desplazamiento

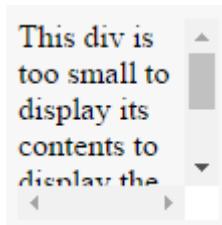
HTML

```
<div>
  This div is too small to display its contents to display the effects of the overflow
  property.
</div>
```

CSS

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

Resultado



El contenido anterior se recorta en un cuadro de 100px por 100px, con desplazamiento disponible para ver el contenido desbordado.

La mayoría de los navegadores de escritorio mostrarán barras de desplazamiento tanto horizontales como verticales, ya sea que el contenido esté recortado o no. Esto puede evitar problemas con la aparición y desaparición de barras de desplazamiento en un entorno dinámico. Las impresoras pueden imprimir contenido desbordado.

envoltura de desbordamiento

`overflow-wrap` le dice a un navegador que puede dividir una línea de texto dentro de un elemento específico en varias líneas en un lugar de otro modo irrompible. Útil para evitar que una larga cadena de texto cause problemas de diseño debido al desbordamiento de 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>: Small words are displayed normally, but a long word like <span  
    style="red;">supercalifragilisticexpialidocious</span> is too long so it will overflow past  
    the edge of the line-break  
</div>  
  
<div id="div2">  
    <strong>#div2</strong>: Small words are displayed normally, but a long word like <span  
    style="red;">supercalifragilisticexpialidocious</span> will be split at the line break and  
    continue on the next line.  
</div>
```

#div1: Small words are displayed normally, but a long word like **supercalifragilisticexpialidocious** is too long so it will overflow past the edge of the line-break

#div2: Small words are displayed normally, but a long word like **supercalifragilisticexpialidocious** will be split at the line break and continue on the next line.

overflow-wrap - Valor	Detalles
normal	Permite que una palabra se desborde si es más larga que la línea.
break-word	Dividirá una palabra en múltiples líneas, si es necesario
inherit	Hereda el valor del elemento padre para esta propiedad.

desbordamiento: visible

HTML

```
<div>
    Even if this div is too small to display its contents, the content is not clipped.
</div>
```

CSS

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

Resultado



Even if
this div
is too
small
to
display
its
contents,
the
content
is not
clipped.

El contenido no se recorta y se procesará fuera del cuadro de contenido si supera su tamaño de contenedor.

Contexto de formato de bloque creado con desbordamiento

El uso de la propiedad de `overflow` con un valor diferente al `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; /* creates block formatting context */
}
```

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 debit is sea.</p>
</div>
```

Resultado

The containing div of this text does not have overflow set

100x100

 Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia. Eam in velit graecis, sea mucius 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 instructior usu ne. At ludus suscipit disputationi vel.

Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debit is sea. Cu eos munere virtute vituperata. Exerci bonorum sed id, id nec tantas praesent complectitur. Vel cu legendos mediocritatem. Enim liberavisse ei sea.

The containing div of this text has overflow:hidden

100x100

 Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia. Eam in velit graecis, sea mucius 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 instructior usu ne. At ludus suscipit disputationi vel.

Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debit is sea. Cu eos munere virtute vituperata. Exerci bonorum sed id, id nec tantas praesent complectitur. Vel cu legendos mediocritatem. Enim liberavisse ei sea.

Este ejemplo muestra cómo los párrafos dentro de un div con el conjunto de propiedades de `overflow` interactuarán con una imagen flotada.

overflow-x y overflow-y

Estas dos propiedades funcionan de manera similar a la propiedad de `overflow` y aceptan los mismos valores. El parámetro `overflow-x` solo 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">
  If this div is too small to display its contents,
  the content to the left and right will be clipped.
</div>

<div id="div-y">
  If this div is too small to display its contents,
  the content to the top and bottom will be clipped.
</div>
```

CSS

```
div {
  width: 200px;
  height: 200px;
}
```

```
#div-x {  
    overflow-x: hidden;  
}  
  
#div-y {  
    overflow-y: hidden;  
}
```

Lea Rebosar en línea: <https://riptutorial.com/es/css/topic/4947/rebosar>

Capítulo 47: Recorte y enmascaramiento

Sintaxis

- **Recorte**
- ruta de clip: <clip-source> | [<basic-shape> || <clip-geometry-box>] | ninguna
- **Enmascaramiento**
- máscara-imagen: [ninguno | <referencia de máscara>] #
- modo de máscara: [<modo de máscara>] #
- repetición de máscara: [<estilo de repetición>] #
- posición de la máscara: [<posición>] #
- máscara-clip: [<geometry-box> | no hay video]#
- origen de la máscara: [<geometry-box>] #
- tamaño de máscara: [<tamaño_gg>] #
- compuesto de máscara: [<compositing-operator>] #
- máscara: [<mask-reference> <masking-mode>? || <position> [/ <bg-size>]? || <repeat-style> || <geometry-box> || [<geometry-box> | sin clip] || <compositing-operator>] #

Parámetros

Parámetro	Detalles
fuente de clip	Una URL que puede apuntar a un elemento SVG en línea (o) un elemento SVG en un archivo externo que contiene la definición de la ruta del clip.
forma básica	Se refiere a uno entre <code>inset()</code> , <code>circle()</code> , <code>ellipse()</code> o <code>polygon()</code> . Usando una de estas funciones se define el trazado de recorte. Estas funciones de forma funcionan exactamente de la misma forma que en Formas para flotadores
clip-geometría-caja	Esto puede tener uno entre <code>content-box</code> , <code>content-box padding-box border-box margin-box fill-box stroke-box view-box</code> como valores. Cuando esto se proporciona sin ningún valor para <basic-shape>, los bordes del cuadro correspondiente se utilizan como la ruta para recortar. Cuando se usa con una <forma básica>, esto actúa como el cuadro de referencia para la forma.
máscara-referencia	Esto puede ser <code>none</code> o una imagen o una URL de referencia a una fuente de imagen de máscara.
estilo de repetición	Esto especifica cómo se debe repetir o colocar la máscara en los ejes X e Y. Los valores admitidos son <code>repeat-x</code> , <code>repeat-y</code> , <code>repeat</code> , <code>space</code> , <code>round</code> , <code>no-repeat</code> .
modo de máscara	Puede ser <code>alpha</code> o <code>luminance</code> o <code>auto</code> e indica si la máscara debe tratarse como una máscara alfa o una máscara de luminancia. Si no se proporciona ningún valor y la referencia de la máscara es una imagen directa, se consideraría

Parámetro	Detalles
	una máscara alfa (o) si la referencia de la máscara es una URL, se consideraría una máscara de luminancia.
posición	Esto especifica la posición de cada capa de máscara y es similar en comportamiento a la propiedad de <code>background-position</code> . El valor se puede proporcionar en la sintaxis de 1 valor (como <code>top</code> , <code>10%</code>) o en la sintaxis de 2 valores (como <code>top right</code> , <code>50% 50%</code>).
caja de geometría	Esto especifica el cuadro en el que se debe recortar la <i>máscara</i> (<i>área de pintura de máscara</i>) o el cuadro que se debe utilizar como referencia para el origen de la <i>máscara</i> (<i>área de posicionamiento de máscara</i>) según la propiedad. La lista de valores posibles es <code>content-box padding-box border-box margin-box fill-box stroke-box view-box</code> . La explicación detallada de cómo funciona cada uno de esos valores está disponible en la especificación W3C .
tamaño bg	Esto representa el tamaño de cada capa de imagen de máscara y tiene la misma sintaxis que <code>background-size</code> . El valor puede ser longitud o porcentaje o auto o cubrir o contener. La longitud, el porcentaje y el auto pueden proporcionarse como un solo valor o como uno para cada eje.
operador de composición	Puede ser 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 se debe usar para esta capa con las que están debajo. La explicación detallada sobre cada valor está disponible en las especificaciones del W3C .

Observaciones

CSS Clipping y Masking son conceptos muy nuevos, por lo que el soporte del navegador para estas propiedades es bastante bajo.

Mascarillas

Al momento de escribir (Jul '16), Chrome, Safari y Opera admiten estas propiedades con el prefijo `-webkit-`.

Firefox no requiere prefijos, pero admite máscaras solo cuando se usa con elementos de `mask` SVG. Para los elementos de `mask` SVG en línea, la sintaxis es `mask: url(#msk)` mientras que para el uso de elementos de `mask` en un archivo SVG externo, `mask: url('yourfilepath/yourfilename.svg#msk')`. `#msk` en ambos casos se refiere al `id` del elemento de `mask` que se hace referencia. Como se indica en [esta respuesta](#), en la actualidad, Firefox no admite ningún parámetro que no sea `mask-reference` a la `mask` en la propiedad de la `mask`.

Internet Explorer (y Edge) aún no ofrece soporte para esta propiedad.

La propiedad del `mask-mode` actualmente no es compatible con ningún navegador **con o sin** prefijos.

Recorrido del clip:

Como en el momento de escribir (Jul '16) Chrome, Safari y Opera son compatibles `clip-path` cuando la ruta se crea utilizando formas básicas (como `circle`, `polygon`) o la sintaxis `url(#clipper)` con SVG en línea. No admiten el recorte basado en formas que forman parte de archivos SVG externos. Además, requieren que el prefijo `-webkit` esté presente.

Firefox solo admite la sintaxis `url()` para `clip-path` mientras que Internet Explorer (y Edge) no ofrecen soporte.

Examples

Recorte (polígono)

CSS:

```
div{  
    width:200px;  
    height:200px;  
    background:teal;  
    clip-path: polygon(0 0, 0 100%, 100% 50%); /* refer remarks before usage */  
}
```

HTML:

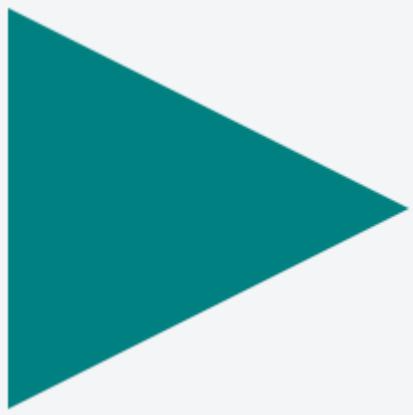
```
<div></div>
```

En el ejemplo anterior, se utiliza un trazado de recorte **poligonal** para recortar el elemento cuadrado (200 x 200) en forma de triángulo. La forma de salida es un triángulo porque la ruta comienza en (es decir, las primeras coordenadas están en) `0 0`, que es la esquina superior izquierda del cuadro, luego va a `0 100%`, que es la esquina inferior izquierda del cuadro y finalmente, al `100% 50%` que no es más que el punto medio derecho de la caja. Estos caminos se cierran automáticamente (es decir, el punto de inicio será el punto final) y, por lo tanto, la forma final es la de un triángulo.

Esto también se puede utilizar en un elemento con una imagen o un degradado como fondo.

[Ver ejemplo](#)

Salida:



Recorte (círculo)

CSS:

```
div{  
    width: 200px;  
    height: 200px;  
    background: teal;  
    clip-path: circle(30% at 50% 50%); /* refer remarks before usage */  
}
```

HTML

```
<div></div>
```

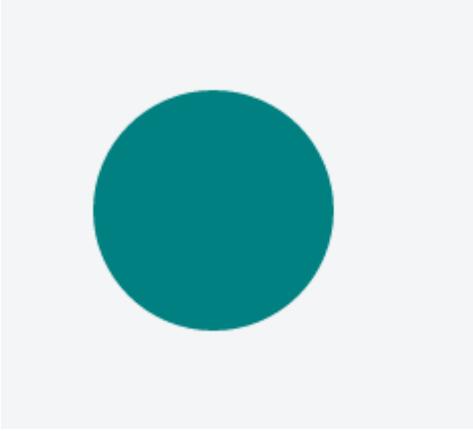
Este ejemplo muestra cómo recortar un div en un círculo. El elemento se recorta en un círculo cuyo radio es del 30% en función de las dimensiones del cuadro de referencia con su punto central en el centro del cuadro de referencia. Aquí, ya que no se proporciona `<clip-geometry-box>` (en otras palabras, cuadro de referencia), el `border-box` de `border-box` del elemento se usará como cuadro de referencia.

La forma del círculo debe tener un radio y un centro con las coordenadas (x, y) :

```
circle(radius at x y)
```

[Ver ejemplo](#)

Salida:

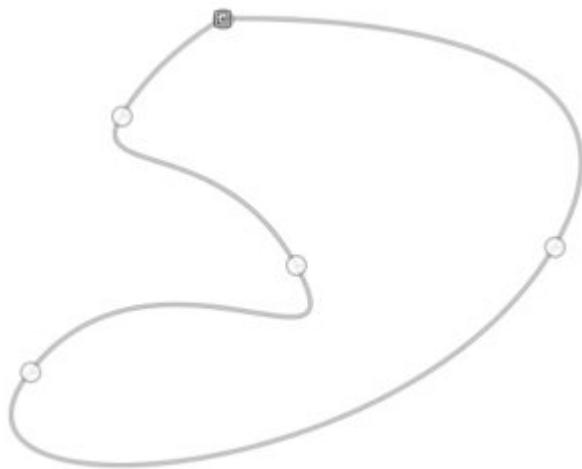


Recorte y enmascaramiento: Resumen y diferencia

Con **Recorte** y **Máscara** puede hacer que algunas partes específicas de los elementos sean transparentes u opacas. Ambos pueden aplicarse a cualquier elemento HTML.

Recorte

Los clips son caminos vectoriales. Fuera de este camino el elemento será transparente, dentro de él será opaco. Por lo tanto, puede definir una propiedad de `clip-path` en los elementos. Cada elemento gráfico que también existe en SVG se puede usar aquí como una función para definir la ruta. Los ejemplos son `circle()`, `polygon()` o `ellipse()`.



Ejemplo

```
clip-path: circle(100px at center);
```

El elemento solo será visible dentro de este círculo, que se coloca en el centro del elemento y tiene un radio de 100px.

Enmascaramiento

Las máscaras son similares a los Clips, pero en lugar de definir una ruta, se define una máscara

en qué capas sobre el elemento. Puede imaginar esta máscara como una imagen que consiste principalmente en dos colores: blanco y negro.

Máscara de luminancia : Negro significa que la región es opaca y blanca que es transparente, pero también hay un área gris que es semitransparente, por lo que puede hacer transiciones suaves.

Máscara alfa : solo en las áreas transparentes de la máscara el elemento será opaco.



Esta imagen, por ejemplo, puede usarse como una máscara de luminancia para hacer que un elemento tenga una transición muy suave de derecha a izquierda y de opaco a transparente.

La propiedad de `mask` permite especificar el tipo de máscara y una imagen que se utilizará como capa.

Ejemplo

```
mask: url(masks.svg#rectangle) luminance;
```

Un elemento llamado `rectangle` definido en `masks.svg` se usará como una **máscara de luminancia** en el elemento.

Máscara simple que desvanece una imagen de sólido a transparente.

CSS

```
div {  
    height: 200px;  
    width: 200px;  
    background: url(http://lorempixel.com/200/200/nature/1);  
    mask-image: linear-gradient(to right, white, transparent);  
}
```

HTML

```
<div></div>
```

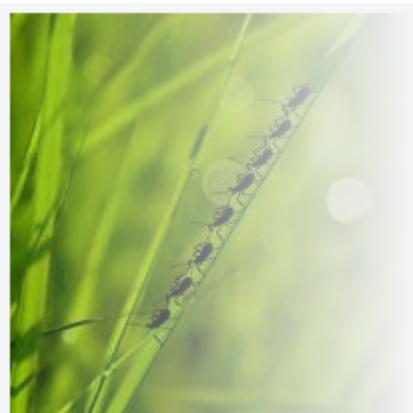
En el ejemplo anterior hay un elemento con una imagen como fondo. La máscara que se aplica en la imagen (mediante CSS) hace que parezca que se está desvaneciendo de izquierda a derecha.

El enmascaramiento se logra utilizando un `linear-gradient` que va de blanco (a la izquierda) a transparente (a la derecha) como máscara. Como es una máscara alfa, la imagen se vuelve transparente donde la máscara es transparente.

Salida sin la máscara:



Salida con la máscara:



Nota: Como se mencionó en las observaciones, el ejemplo anterior funcionaría en Chrome, Safari y Opera solo cuando se usa con el prefijo `-webkit`. Este ejemplo (con un `linear-gradient` como imagen de máscara) aún no es compatible con Firefox.

Usando máscaras para cortar un agujero en el medio de una imagen

CSS

```
div {  
    width: 200px;  
    height: 200px;  
    background: url(http://lorempixel.com/200/200/abstract/6);  
    mask-image: radial-gradient(circle farthest-side at center, transparent 49%, white 50%); /*  
check remarks before using */
```

```
}
```

HTML

En el ejemplo anterior, se crea un círculo transparente en el centro usando `radial-gradient` y luego se usa como máscara para producir el efecto de un círculo que se corta desde el centro de una imagen.

Imagen sin máscara:



Imagen con máscara:



Usando máscaras para crear imágenes con formas irregulares.

CSS

```
div { /* check remarks before usage */
  height: 200px;
  width: 400px;
  background-image: url(http://lorempixel.com/400/200/nature/4);
  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;
```

}

HTML

```
<div></div>
```

En el ejemplo anterior, tres imágenes de `linear-gradient` (que cuando se colocan en sus posiciones apropiadas cubrirían el 100% x 100% del tamaño del contenedor) se utilizan como máscaras para producir un corte transparente de forma triangular en la parte inferior de la imagen.

Imagen sin la máscara:



Imagen con la máscara:



Lea Recorte y enmascaramiento en línea: <https://riptutorial.com/es/css/topic/3721/recorte-y-enmascaramiento>

Capítulo 48: Relleno

Sintaxis

- relleno: *longitud* | inicial | heredar | unset;
- padding-top: *length* | initial | inherit | unset;
- padding-right: *length* | initial | inherit | unset;
- padding-bottom: *length* | initial | inherit | unset;
- padding-left: *length* | initial | inherit | unset;

Observaciones

La propiedad de relleno 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 permiten valores negativos .**

1 : <https://developer.mozilla.org/en/docs/Web/CSS/padding> MDN

También vea esta pregunta , "¿Por qué CSS no admite el relleno negativo?" y sus respuestas.

También considere el modelo de caja cuando use relleno. Dependiendo del valor del tamaño de la caja, el relleno en un elemento puede agregarse a la altura / anchura previamente definida de un elemento o no.

Propiedades relacionadas:

[margen](#)

El relleno en los elementos en línea solo se aplicará a la izquierda y derecha del elemento, y no a la parte superior e inferior, debido a las propiedades de visualización inherentes de los elementos en línea.

Examples

Relleno en un lado dado

La propiedad de relleno 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 permiten valores negativos.

Puedes especificar un lado individualmente:

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

El siguiente código agregaría un relleno de 5px a la parte superior de la división:

```
<style>
.myClass {
    padding-top: 5px;
}
</style>

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

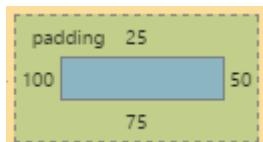
Relleno de taquigrafía

La propiedad de relleno 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 permiten valores negativos.

Para guardar la adición de relleno a cada lado individualmente (usando `padding-top`, `padding-left` etc.) puede escribirlo como una taquigrafía, como se muestra a continuación:

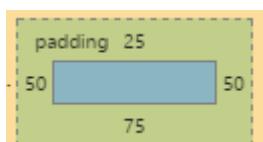
Cuatro valores :

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



Tres valores :

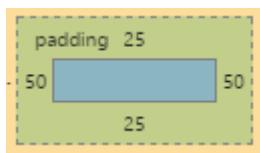
```
<style>
.myDiv {
    padding: 25px 50px 75px; /* top left/right bottom */
}
</style>
<div class="myDiv"></div>
```



Dos valores :

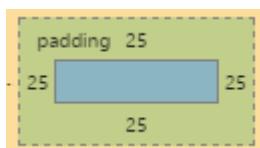
```
<style>
.myDiv {
    padding: 25px 50px; /* top/bottom left/right */
}
</style>
```

```
</style>
<div class="myDiv"></div>
```



Un valor:

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



Lea Relleno en línea: <https://riptutorial.com/es/css/topic/1255/relleno>

Capítulo 49: Selectores

Introducción

Los selectores CSS identifican elementos HTML específicos como objetivos para los estilos CSS. Este tema trata sobre cómo los selectores de CSS tienen como objetivo los elementos HTML. Los selectores utilizan una amplia gama de más de 50 métodos de selección ofrecidos por el lenguaje CSS, incluidos elementos, clases, ID, pseudo-elementos y pseudo-clases, y patrones.

Sintaxis

- `# id`
- `. nombre de la clase`
- `: nombre de pseudo-clase`
- `:: nombre de pseudo-elemento`
- `[attr] / * tiene el atributo attr . * /`
- `[attr = " valor "] / * tiene el atributo attr , y su valor es exactamente " valor ". * /`
- `[attr ~ = " valor "] / * tiene el atributo attr , y su valor, cuando se divide en espacios en blanco , contiene " valor ". * /`
- `[attr | = " valor "] / * tiene el atributo attr , y su valor es exactamente " valor ", o su valor comienza con " valor - ". * /`
- `[attr ^ = " valor "] / * tiene el atributo attr , y su valor comienza con " valor ". * /`
- `[attr $ = " valor "] / * tiene el atributo attr , y su valor termina con " valor ". * /`
- `[attr * = " value "] / * tiene el atributo attr , y su valor contiene " value ". * /`
- `nombre-elemento`
- `*`

Observaciones

- Usted verá a veces dos puntos dobles (`::`) en lugar de sólo uno (`:`). Esta es una manera de separar las **pseudo-clases** de los **pseudo-elementos** .
- Navegadores antiguos, como Internet Explorer 8, **sólo** admiten un único dos puntos (`:`) para la definición de pseudo-elementos.
- A diferencia de las pseudo-clases, solo se puede usar un pseudo-elemento por selector, si está presente, debe aparecer después de la secuencia de selectores simples que representan a los sujetos del selector (una versión futura de la **especificación W3C** puede permitir múltiples pseudo-elementos por selector)).

Examples

Selectores de atributos

Visión general

Los selectores de atributos se pueden utilizar con varios tipos de operadores que cambian los criterios de selección en consecuencia. Seleccionan un elemento utilizando la presencia de un atributo o valor de atributo dado.

Selector (1)	Elemento emparejado	Selecciona elementos ...	Versión CSS
[attr]	<div attr>	Con atributo attr	2
[attr='val']	<div attr="val">	Donde atributo attr tiene valor val	2
[attr~='val']	<div attr="val val2 val3">	Donde aparece val en el lista de espacios en blanco separados de attr	2
[attr^='val']	<div attr="val1 val2">	Donde el valor de attr comienza con val	3
[attr\$='val']	<div attr="sth aval">	Donde el valor del attr termina con val	3
[attr*='val']	<div attr="somevalhere">	Donde attr contiene val cualquier lugar	3
[attr = 'val']	<div attr="val-sth etc">	Donde el valor de attr es exactamente val , o comienza con val e inmediatamente seguido de - (U + 002D)	2
[attr='val' i]	<div attr="val">	Donde attr tiene valor val , haciendo caso omiso de la caja de letras de val .	4 (2)

Notas:

1. El valor del atributo puede estar entre comillas simples o dobles. Ninguna cita puede funcionar, pero no es válida de acuerdo con el estándar CSS, y no se recomienda.
2. No existe una especificación CSS4 única e integrada, ya que se divide en módulos separados. Sin embargo, hay módulos de "nivel 4". [Ver soporte del navegador](#) .

Detalles

```
[attribute]
```

Selecciona elementos con el atributo dado.

```
div[data-color] {  
  color: red;  
}
```

```
<div data-color="red">This will be red</div>  
<div data-color="green">This will be red</div>  
<div data-background="red">This will NOT be red</div>
```

[Demo en vivo en JSBin](#)

```
[attribute="value"]
```

Selecciona elementos con el atributo y valor dados.

```
div[data-color="red"] {  
  color: red;  
}
```

```
<div data-color="red">This will be red</div>  
<div data-color="green">This will NOT be red</div>  
<div data-color="blue">This will NOT be red</div>
```

[Demo en vivo en JSBin](#)

```
[attribute*="value"]
```

Selecciona elementos con el atributo y valor dados donde el atributo dado contiene el valor dado en cualquier lugar (como subcadena).

```
[class*="foo"] {  
  color: red;  
}
```

```
<div class="foo-123">This will be red</div>  
<div class="fool23">This will be red</div>  
<div class="bar123foo">This will be red</div>  
<div class="barfoo0123">This will be red</div>  
<div class="barfo0">This will NOT be red</div>
```

[Demo en vivo en JSBin](#)

```
[attribute~= "value"]
```

Selecciona elementos con el atributo y valor dados donde 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">This will be red</div>  
<div class="color-blue foo-bar the-div">This will NOT be red</div>
```

Demo en vivo en JSBin

[attribute^="value"]

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

```
[class^="foo-"] {  
    color: red;  
}
```

```
<div class="foo-123">This will be red</div>  
<div class="foo-234">This will be red</div>  
<div class="bar-123">This will NOT be red</div>
```

Demo en vivo en JSBin

[attribute\$="value"]

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

```
[class$="file"] {  
    color: red;  
}
```

```
<div class="foobar-file">This will be red</div>  
<div class="foobar-file">This will be red</div>  
<div class="foobar-input">This will NOT be red</div>
```

Demo en vivo en JSBin

[attribute|= "value"]

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

```
[lang|="EN"] {  
    color: red;  
}
```

```
<div lang="EN-us">This will be red</div>  
<div lang="EN-gb">This will be red</div>  
<div lang="PT-pt">This will NOT be red</div>
```

Demo en vivo en JSBin

```
[attribute="value" i]
```

Selecciona elementos con un atributo y valor dados donde el valor del atributo puede representarse como `Value`, `VALUE`, `vAlUe` o cualquier otra posibilidad que no `vAlUe` entre mayúsculas y minúsculas.

```
[lang="EN" i] {  
    color: red;  
}
```

```
<div lang="EN">This will be red</div>  
<div lang="en">This will be red</div>  
<div lang="PT">This will NOT be red</div>
```

Demo en vivo 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 usar un selector de atributos para seleccionar un elemento por su ID en un nivel de especificidad más bajo que si se seleccionara con un [selector de ID](#) : `[id="my-ID"]` apunta al mismo elemento que `#my-ID` pero con menor especificidad.

Vea la [sección de sintaxis](#) para más detalles.

Combinadores

Visión general

Selector	Descripción
<code>div span</code>	Selector descendiente (todos los <code></code> que son descendientes de un <code><div></code>)
<code>div > span</code>	Selector secundario (todos los <code></code> que son un hijo directo de un <code><div></code>)
<code>a ~ span</code>	Selector general de hermanos (todos los <code></code> que son hermanos después de un <code><a></code>)
<code>a + span</code>	Selector de hermanos adyacente (todos los <code></code> que están inmediatamente

Selector	Descripción
	después de un <a>)

Nota: Los selectores de hermanos se dirigen a los elementos que vienen después de ellos en el documento de origen. CSS, por su naturaleza (se conecta en cascada), no puede apuntar a elementos *anteriores* o *principales*. Sin embargo, al usar la propiedad de `order` flexible, [se puede simular un selector de hermanos anterior en medios visuales](#).

Combinador Descendiente: selector selector

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

```
div p {
  color:red;
}

<div>
  <p>My text is red</p>
  <section>
    <p>My text is red</p>
  </section>
</div>

<p>My text is not red</p>
```

[Demo en vivo en JSBin](#)

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

Combinador infantil: 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>My text is red</p>
  <section>
    <p>My text is not red</p>
```

```
</section>  
</div>
```

Demo en vivo en JSBin

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

El segundo elemento `<p>` no está seleccionado porque no es un elemento secundario directo de `<div>`.

Combinador de hermanos adyacente: selector + selector

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

```
p + p {  
  color:red;  
}
```

```
<p>My text is not red</p>  
<p>My text is red</p>  
<p>My text is red</p>  
<hr>  
<p>My text is not red</p>
```

Demo en vivo en JSBin

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

Combinador general de hermanos: selector ~ selector

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

```
p ~ p {  
  color:red;  
}
```

```
<p>My text is not red</p>  
<p>My text is red</p>  
<hr>  
<h1>And now a title</h1>  
<p>My text is red</p>
```

Demo en vivo en JSBin

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

Selectores de nombre de clase

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

```
<div class="warning">
  <p>This would be some warning copy.</p>
</div>
```

También puede combinar nombres de clase para orientar elementos más específicamente. Vamos a aprovechar el ejemplo anterior para mostrar una selección de clase más complicada.

CSS

```
.important {
  color: orange;
}
.warning {
  color: blue;
}
.warning.important {
  color: red;
}
```

HTML

```
<div class="warning">
  <p>This would be some warning copy.</p>
</div>

<div class="important warning">
  <p class="important">This is some really important warning copy.</p>
</div>
```

En este ejemplo, todos los elementos con la `.warning` clase tendrán un color azul, y elementos con el `.important` clase con tener un color anaranjado del texto, y todos los elementos que tienen tanto el `.important` y `.warning` nombre de la clase tendrán un texto de color rojo color.

Tenga en cuenta que dentro de la CSS, la declaración `.warning.important` no tenía ningún espacio entre los dos nombres de clase. Esto significa que solo encontrará elementos que contengan la `warning` ambos nombres de clase y que sean `important` en su atributo de `class`. Esos nombres de clase podrían estar en cualquier orden en el elemento.

Si se incluyera un espacio entre las dos clases en la declaración de CSS, solo seleccionaría elementos que tengan elementos primarios con nombres de clase `.warning` y elementos secundarios con nombres de clase `.important`.

Selectores de ID

Los selectores de ID seleccionan elementos DOM con el ID de destino. Para seleccionar un elemento por una ID específica en CSS, se usa # prefijo # .

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

```
<div id="exampleID">
  <p>Example</p>
</div>
```

... se puede seleccionar mediante `#exampleID` en CSS como se muestra a continuación:

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

Nota : las especificaciones HTML no permiten múltiples elementos con la misma ID

Pseudo-clases

Las **pseudo-clases** son **palabras clave** que permiten la selección en función de la información que se encuentra fuera del árbol del documento o que no pueden ser expresadas por otros selectores o combinadores. Esta información puede asociarse a cierto estado (**estado** y pseudo-clases **dinámicas**), a ubicaciones (**estructural** y **objetivo** pseudo-clases), a negaciones de la primera (pseudo-clase de **negación**) o a idiomas (pseudo-clase **lang**). Los ejemplos incluyen si se ha seguido o no un enlace (`:visited`), el mouse se encuentra sobre un elemento (`:hover`), se marca una casilla de verificación (`:checked`), etc.

Sintaxis

```
selector:pseudo-class {
  property: value;
}
```

Lista de pseudo-clases:

Nombre	Descripción
<code>:active</code>	Se aplica a cualquier elemento que el usuario active (es decir, haga clic).
<code>:any</code>	Le permite crear conjuntos de selectores relacionados mediante la creación de grupos que los artículos incluidos coincidirán. Esta es una alternativa a repetir un selector completo.
<code>:target</code>	Selecciona el elemento activo actual #news (se hace clic en una URL)

Nombre	Descripción
	que contiene ese nombre de ancla)
:checked	Se aplica a elementos de radio, casilla de verificación o opción que están marcados o cambiar a un estado "activado".
: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 esté deshabilitado.
:empty	Se aplica a cualquier elemento que no tenga hijos.
:enabled	Se aplica a cualquier elemento de UI que esté en un estado habilitado.
:first	Utilizado junto con la regla @page , esto selecciona la primera página en un documento impreso.
:first-child	Representa cualquier elemento que sea el primer elemento secundario de su padre.
:first-of-type	Se aplica cuando un elemento es el primero del tipo de elemento seleccionado dentro de su parent. Esto puede o no ser el primer hijo.
:focus	Se aplica a cualquier elemento que tenga el enfoque del usuario. Esto puede ser dado por el teclado del usuario, eventos del mouse u otras formas de entrada.
:focus-within	Puede usarse para resaltar una sección completa cuando un elemento dentro de ella está enfocado. Coincide con cualquier elemento que coincida con la pseudo-clase: focus o que tiene un foco descendente.
:full-screen	Se aplica a cualquier elemento mostrado 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 que se encuentre sobre el dispositivo señalador del usuario, pero no esta activado.
:indeterminate	Aplica elementos de UI de radio o casilla de verificación que no están marcados ni sin marcar, pero están en un estado indeterminado. Esto puede ser debido a una Atributo del elemento o manipulación de DOM.

Nombre	Descripción
:in-range	<p>La pseudo-clase CSS <code>:in-range</code> coincide cuando un elemento tiene su atributo de valor dentro de las limitaciones de rango especificadas para este elemento.</p> <p>Permite a la página dar una retroalimentación de que el valor actualmente definido</p> <p>El uso del elemento está dentro de los límites del rango.</p>
:invalid	<p>Se aplica a <code><input></code> elementos <code><input></code> cuyos valores no son válidos de acuerdo con</p> <p>El tipo especificado en el atributo <code>type=</code>.</p>
:lang	<p>Se aplica a cualquier elemento que esté envolviendo el elemento <code><body></code> tiene un</p> <p><code>lang=</code> designado <code>lang=</code> atributo. Para que la pseudo-clase sea válida, debe Contiene un código de idioma de dos o tres letras válido.</p>
:last-child	Representa cualquier elemento que sea el último elemento secundario de su padre.
:last-of-type	Se aplica cuando un elemento es el último del tipo de elemento seleccionado dentro su padre Esto puede o no ser el último hijo.
:left	<p>Utilizado junto con la regla <code>@page</code>, esto selecciona todos los elementos de la izquierda.</p> <p>Páginas en un documento impreso.</p>
: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) O :not(.class-name))</code> por ejemplo. Debe tener un valor para ser Válido y solo puede contener un selector. Sin embargo, puede encadenar múltiples <code>:not</code> selectores juntos.
:nth-child	<p>Se aplica cuando un elemento es el <code>n</code>-ésimo elemento de su padre, donde <code>n</code></p> <p>puede ser un número entero, una expresión matemática (por ejemplo, <code>n+3</code>) o las palabras clave</p> <p><code>odd</code> O <code>even</code></p>
:nth-of-type	Se aplica cuando un elemento es el <code>n</code> -ésimo elemento de su padre de la mismo tipo de elemento, donde <code>n</code> puede ser un entero, un matemático la expresión (por ejemplo, <code>n+3</code>) o las palabras clave <code>odd</code> O <code>even</code> .
:only-child	<p>La pseudo-clase CSS <code>:only-child</code> representa cualquier elemento que es el único hijo de su padre. Esto es lo mismo que</p> <p><code>:first-child:last-child O :nth-child(1):nth-last-child(1)</code>,</p>

Nombre	Descripción
	Pero con una menor especificidad.
:optional	La pseudo-clase CSS <code>:optional</code> representa cualquier elemento que no tiene el atributo requerido establecido en él. Esto permite formularios para indicar fácilmente los campos opcionales y darles un estilo acorde
:out-of-range	La pseudo-clase CSS <code>:out-of-range</code> coincide cuando un elemento tiene su atributo de valor fuera de las limitaciones de rango especificadas para este elemento. Permite a la página dar una retroalimentación de que el valor actualmente definido utilizando el El elemento está fuera de los límites del rango. Un valor puede estar fuera de un rango si es ya sea más pequeño o más grande que los valores máximos y mínimos establecidos.
:placeholder-shown	Experimental. Se aplica a cualquier elemento de formulario que muestre actualmente el 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 <code><input></code> elementos <code><input></code> .
:right	Utilizado junto con la regla <code>@page</code> , esto selecciona todas las páginas correctas en un documento impreso.
:root	coincide con el elemento raíz de un árbol que representa el documento.
:scope	CSS pseudo-clase coincide con los elementos que son una referencia punto para que los selectores coincidan contra.
:target	Selecciona el elemento activo actual #news (se hace clic en una URL) que contiene ese nombre de ancla)
:visited	Se aplica a cualquier enlace que haya sido visitado por el usuario.

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

Selectores basicos

Selector	Descripción
*	Selector universal (todos los elementos)
div	Selector de etiquetas (todos los elementos <div>)
.blue	Selector de clase (todos los elementos con clase blue)
.blue.red	Todos los elementos con clase blue y red (un tipo de selector compuesto)
#headline	Selector de ID (el elemento con el atributo "id" establecido en el headline)
:pseudo-class	Todos los elementos con pseudo-clase.
::pseudo-element	Elemento que coincide con el pseudo-elemento.
:lang(en)	Elemento que coincide con: declaración lang, por ejemplo
div > p	selector de niños

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

Se puede encontrar una lista completa de los selectores en la [especificación del nivel 3 de los selectores de CSS](#).

Cómo diseñar una entrada de rango

HTML

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

CSS

Efecto	Selector Pseudo
Pulgar	input[type=range]::-webkit-slider-thumb, input[type=range]::-moz-range-thumb, input[type=range]::-ms-thumb
Pista	input[type=range]::-webkit-slider-runnable-track, input[type=range]::-moz-range-track, input[type=range]::-ms-track
Enfocado	input[type=range]:focus
Parte inferior de la pista	input[type=range]::-moz-range-progress, input[type=range]::-ms-fill-lower (no es posible en los navegadores WebKit actualmente - se necesita JS)

Global booleano con casilla de verificación: marcado y ~ (combinador general de hermanos)

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

Añadir booleano como casilla de verificación

Al principio de su documento, agregue todos los valores booleanos que desee con una `id` única y el conjunto de atributos `hidden`:

```
<input type="checkbox" id="sidebarShown" hidden />
<input type="checkbox" id="darkThemeUsed" hidden />

<!-- here begins actual content, for example: -->
<div id="container">
    <div id="sidebar">
        <!-- Menu, Search, ... -->
    </div>

    <!-- Some more content ... -->
</div>

<div id="footer">
    <!-- ... -->
</div>
```

Cambiar el valor del booleano.

Puede alternar el valor booleano agregando una `label` con el conjunto de atributos `for`:

```
<label for="sidebarShown">Show/Hide the sidebar!</label>
```

Accediendo al valor booleano con CSS

El selector normal (como `.color-red`) especifica las propiedades predeterminadas. Se pueden anular siguiendo los selectores de `true` / `false`:

```
/* true: */
<checkbox>:checked ~ [sibling of checkbox & parent of target] <target>

/* false: */
<checkbox>:not(:checked) ~ [sibling of checkbox & parent of target] <target>
```

Tenga en cuenta que `<checkbox>`, `[sibling ...]` y `<target>` deben ser reemplazados por los selectores adecuados. `[sibling ...]` 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;
}

```

En acción

Vea [este violín](#) para una implementación de estos booleanos globales.

CSS3: ejemplo de selector de rango

```

<style>
input:in-range {
    border: 1px solid blue;
}
</style>

<input type="number" min="10" max="20" value="15">
<p>The border for this value will be blue</p>

```

La pseudo-clase CSS `:in-range` coincide cuando un elemento tiene su atributo de valor dentro de las limitaciones del rango especificado para este elemento. Permite a la página dar una respuesta de que el valor actualmente definido utilizando el elemento está dentro de los límites de rango. [\[1\]](#)

Pseudo clase infantil

"La pseudo-clase CSS: `nth-child (an + b)` coincide con un elemento que tiene $a + b - 1$ hermanos delante de él en el árbol de documentos, para un **valor positivo o cero** dado para 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>	✓	✓	✓							

pseudo-selector	1	2	3	4	5	6	7	8	9	10
:nth-child(odd)	✓	✓	✓	✓	✓	✓	✓	✓		
:nth-child(even)		✓	✓	✓	✓	✓	✓	✓	✓	
:last-child									✓	
:nth-last-child(3)								✓		

Seleccione el elemento utilizando su ID sin la alta especificidad del selector de ID

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

HTML:

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

CSS

```
#element { ... } /* High specificity will override many selectors */
[id="element"] { ... } /* Low specificity, can be overridden easily */
```

A. El ejemplo: no pseudo-clase & B.: focus-within CSS pseudo-class

A. La sintaxis se presenta arriba.

El siguiente selector coincide con todos `<input>` elementos `<input>` en un documento HTML que no están deshabilitados y no tienen la clase `.example`:

HTML:

```
<form>
  Phone: <input type="tel" class="example">
  E-mail: <input type="email" disabled="disabled">
  Password: <input type="password">
</form>
```

CSS:

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

La pseudo-clase `:not()` también admitirá selectores separados por comas en el Nivel 4 de selectores:

CSS:

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

[Demo en vivo en JSBin](#)

Ver la sintaxis de fondo [aquí](#).

B. El: enfoque dentro de la pseudo-clase CSS

HTML:

```
<h3>Background is blue if the input is focused .</p>  
<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.

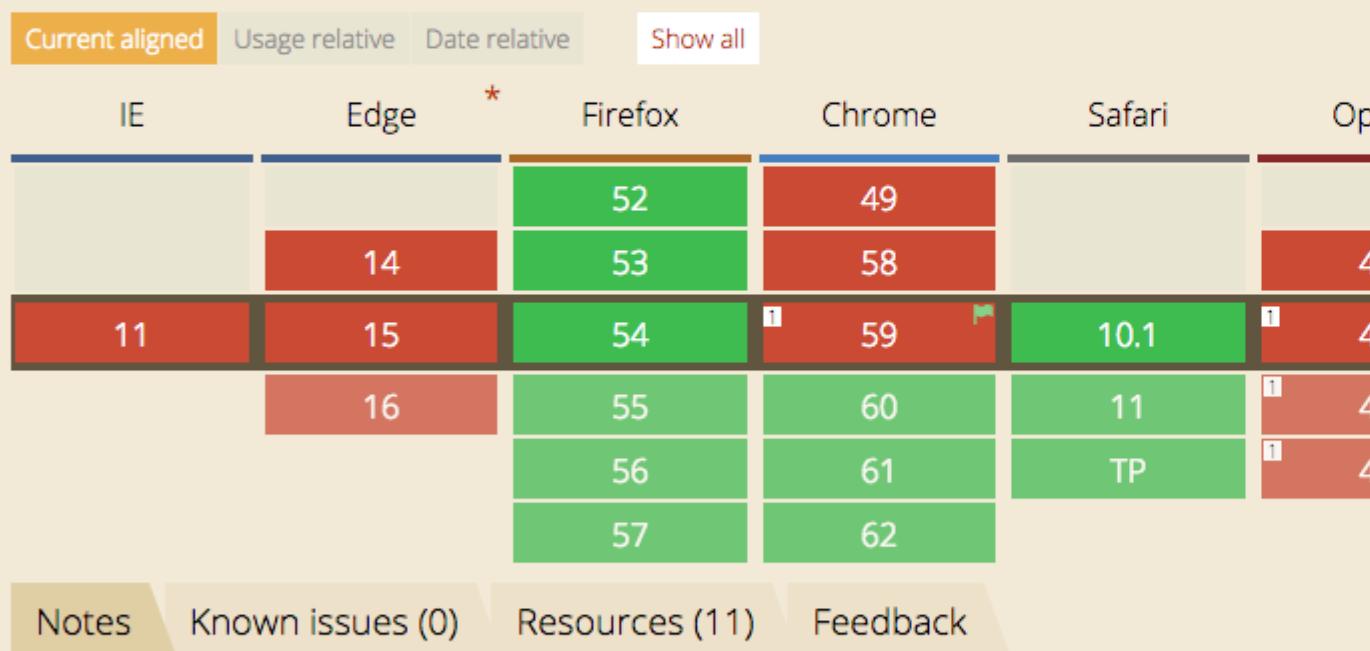


#

:focus-within CSS pseudo-class

- UNOFF

The `:focus-within` pseudo-class matches elements that either themselves match `:focus` or that have descendants which match `:focus`.



¹ Can be enabled via the "Experimental Web Platform Features" flag

El ejemplo de selector de pseudo-clase de hijo único

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

HTML:

```
<div>
  <p>This paragraph is the only child of the div, it will have the color blue</p>
</div>

<div>
  <p>This paragraph is one of the two children of the div</p>
  <p>This paragraph is one of the two children of its parent</p>
</div>
```

CSS:

```
p:only-child {
```

```
    color: blue;  
}
```

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

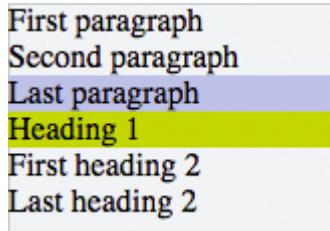
[Demo en vivo en JSBin](#)

El: selector de último tipo

El `:last-of-type` selecciona el elemento que es el último hijo, de un tipo particular, de su padre. En el siguiente ejemplo, el css selecciona el último párrafo y el último encabezado `h1`.

```
p:last-of-type {  
  background: #C5CAE9;  
}  
h1:last-of-type {  
  background: #CDDC39;  
}
```

```
<div class="container">  
  <p>First paragraph</p>  
  <p>Second paragraph</p>  
  <p>Last paragraph</p>  
  <h1>Heading 1</h1>  
  <h2>First heading 2</h2>  
  <h2>Last heading 2</h2>  
</div>
```



[jsFiddle](#)

Lea Selectores en línea: <https://riptutorial.com/es/css/topic/611/selectores>

Capítulo 50: sombra de la caja

Sintaxis

- cuadro-sombra: ninguno | h-shadow v-shadow desenfoque difusión color | inserción | inicial | heredado;

Parámetros

Parámetros	Detalles
recuadro	de forma predeterminada, la sombra se trata como una sombra paralela. la palabra clave insertada dibuja la sombra dentro del marco / borde.
offset-x	la distancia horizontal
offset-y	la distancia vertical
radio de desenfoque	0 por defecto. El valor no puede ser negativo. cuanto mayor sea el valor, más grande y clara será la sombra.
radio de propagación	0 por defecto. Los valores positivos harán que la sombra se expanda. los valores negativos causarán que la sombra se contraiga.
color	puede ser de varias notaciones: una palabra clave de color, hexadecimal, rgb(), rgba(), hsl(), hsla()

Observaciones

Soporte del navegador:

- Cromo 10.0
- IE 9.0
- Firefox 4.0 3.5 -moz
- Safari 5.1 3.1 -webkit-
- Opera 10.5

Examples

sombra paralela

JSFiddle: <https://jsfiddle.net/UnsungHero97/80qod7aL/>

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;  
}
```

sombra interior

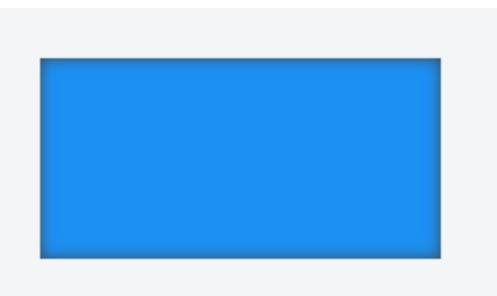
HTML

```
<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:



JSFiddle: <https://jsfiddle.net/UnsungHero97/80qod7aL/1/>

Sombra de la parte inferior sólo con un pseudo-elemento

JSFiddle: <https://jsfiddle.net/UnsungHero97/80qod7aL/2/>

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;  
}
```



múltiples sombras

JSFiddle: <https://jsfiddle.net/UnsungHero97/80qod7aL/5/>

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;  
}
```



Lea sombra de la caja en línea: <https://riptutorial.com/es/css/topic/1746/sombra-de-la-caja>

Capítulo 51: Soporte de navegador y prefijos

Parámetros

Prefijo	Navegador (s)
-webkit-	Google Chrome, Safari, versiones más recientes de Opera 12 y versiones posteriores, navegadores Android, Blackberry y UC
-moz-	Mozilla Firefox
-ms-	Internet Explorer, Edge
-o-, -xv-	Opera hasta la versión 12
-khtml-	Konquerer

Observaciones

Los prefijos de proveedores se utilizan para permitir el soporte de vista previa para la nueva funcionalidad CSS donde la especificación aún no recomienda la funcionalidad.

Se recomienda que no utilice prefijos de proveedores en entornos de producción. Estos prefijos existen para probar nuevas funcionalidades que aún no están finalizadas, y el comportamiento es inherentemente inesperado. El simple uso de prefijos **no** otorga compatibilidad con el navegador para los navegadores antiguos, ya que no puede garantizar que la función no haya cambiado con el tiempo para tener un rendimiento diferente, y aún *podría* estar descompuesto en los navegadores antiguos que dice ser compatibles.

Si es importante admitir navegadores más antiguos, debería considerar utilizar JavaScript u otras soluciones para imitar los efectos y garantizar realmente la compatibilidad con navegadores antiguos.

Los navegadores usarán sus prefijos e ignorarán las propiedades que no entienden.

NOTA : Los prefijos siempre deben aparecer antes de la sintaxis oficial, sin prefijo. De lo contrario, se sobrescribirían con las propiedades prefijadas, lo que puede ser otra implementación al final.

Si un navegador admite una versión sin prefijo y prefijada de una propiedad, la propiedad más reciente que se declarará tendrá prioridad.

Examples

Transiciones

```
div {  
    -webkit-transition: all 4s ease;  
    -moz-transition: all 4s ease;  
    -o-transition: all 4s ease;  
    transition: all 4s ease;  
}
```

Transformar

```
div {  
    -webkit-transform: rotate(45deg);  
    -moz-transform: rotate(45deg);  
    -ms-transform: rotate(45deg);  
    -o-transform: rotate(45deg);  
    transform: rotate(45deg);  
}
```

Lea Soporte de navegador y prefijos en línea: <https://riptutorial.com/es/css/topic/1138/soporte-de-navegador-y-prefijos>

Capítulo 52: Tipografía

Sintaxis

- font: [*font-style*] [*font-variant*] [*font-weight*] *font-size* [/ *line-height*] *font-family* ;
- estilo de *fuente* : estilo de *fuente*
- variante de *fuente* : variante de *fuente*
- *font-weight*: *font-weight* ;
- tamaño de *letra* : tamaño de *letra* ;
- línea-altura: *línea-altura* ;
- Familia de *fuentes* : Familia de *fuentes* ;
- color: *color* ;
- comillas: *ninguna* | *cadena* | *inicial* | *heredar* ;
- estiramiento de la *fuente* : estiramiento de la *fuente* ;
- text-align: *text-align* ;
- texto-sangría: *longitud* | *inicial* | *heredar* ;
- texto desbordado: *clip* | *puntos suspensivos* | *cadena* | *inicial* | *heredado* ;
- text-transform: *none* | *capitalize* | *mayúsculas* | *minúsculas* | *initial* | *inherit* ;
- text-shadow: *h-shadow v-shadow blur-radius color* | *none* | *initial* | *inherit* ;
- font-size-adjust: *number* | *none* | *initial* | *inherit* ;
- fuente-estiramiento: *ultra-condensado* | *extra-condensado* | *condensado* | *semi-condensado* | *normal* | *semi-expandido* | *expandido* | *extra-expandido* | *ultra-expandido* | *inicial* | *heredado* ;
- guiones: *ninguno* | *manual* | *auto*
- tab-size: *número* | *longitud* | *inicial* | *heredar* ;
- espacio entre letras: *normal* | *longitud* | *inicial* | *heredar* ;
- espacio entre palabras: *normal* | *longitud* | *inicial* | *heredar* ;

Parámetros

Parámetro	Detalles
<i>Estilo de fuente</i>	<code>italics</code> <code>u</code> oblique
<i>variante de fuente</i>	<code>normal</code> <code>O</code> small-caps
<i>peso de fuente</i>	normal , bold o numérica de 100 a 900.
<i>tamaño de fuente</i>	El tamaño de fuente dado en % , px , em , o cualquier otra medida de CSS válida
<i>altura de la línea</i>	La altura de línea dada en % , px , em , o cualquier otra medida de CSS válida

Parámetro	Detalles
<i>Familia tipográfica</i>	Esto es para definir el nombre de la familia.
<i>color</i>	Cualquier representación de color CSS válida, como red , #00FF00 , hsl(240, 100%, 50%) etc.
<i>estiramiento de la fuente</i>	Si se usa o no una cara confiada o expandida de la fuente. Los valores válidos son normal , ultra-condensed , extra-condensed , condensed , semi-condensed , semi-expanded , expanded , extra-expanded o ultra-expanded
<i>texto alineado</i>	start , end , left , right , center , justify , match-parent
<i>decoracion de texto</i>	none , underline , overline , line-through , initial , inherit ;

Observaciones

- La propiedad text-shadow no es compatible con versiones de Internet Explorer de menos de 10.

Examples

Tamaño de fuente

HTML:

```
<div id="element-one">Hello I am some text.</div>
<div id="element-two">Hello I am some smaller text.</div>
```

CSS:

```
#element-one {
    font-size: 30px;
}

#element-two {
    font-size: 10px;
}
```

El texto dentro de #element-one tendrá un tamaño de 30px , mientras que el texto en #element-two 10px tendrá un tamaño de 10 10px .

La taquigrafía de la fuente

Con la sintaxis:

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

Puede tener todos los estilos relacionados con la `font` en una declaración con la abreviatura de la `font`. Simplemente use la propiedad de `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 20 px y usar Arial como la familia de fuentes, normalmente lo codificaría de la siguiente manera:

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

Sin embargo, con la abreviatura de la fuente se puede condensar de la siguiente manera:

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

Nota : como `font-style` `font-variant` `font-weight`, `font-weight` y `line-height` son opcionales, los tres se omiten en este ejemplo. Es importante tener en cuenta que el uso del método abreviado **restablece** los otros atributos no dados. Otro punto importante es que los dos atributos necesarios para que funcione el método abreviado de `font-size` son `font-size` `font-family`. Si ambos no están incluidos, 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`

Pilas de fuentes

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

El navegador intentará aplicar la fuente tipográfica "Segoe UI" a los caracteres dentro de los elementos seleccionados por la propiedad anterior. Si esta fuente no está disponible, o si la fuente no contiene un glifo para el carácter requerido, el navegador retrocederá a Tahoma y, si es necesario, cualquier fuente sans-serif en la computadora del usuario. Tenga en cuenta que cualquier nombre de fuente con más de una palabra como "Segoe UI" debe tener comillas simples o dobles a su alrededor.

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

El navegador intentará aplicar la fuente tipográfica "Consolas" a los caracteres dentro de los elementos seleccionados por la propiedad anterior. Si esta fuente no está disponible, o si la fuente no contiene un glifo para el carácter requerido, el navegador retrocederá a "Courier New" y, si es necesario, cualquier fuente monoespaciada en la computadora del usuario.

Espaciado de letras

```
h2 {  
    /* adds a 1px space horizontally between each letter;  
       also known as tracking */  
    letter-spacing: 1px;  
}
```

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

! el espaciado entre letras también admite valores negativos:

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

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

Transformación de texto

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

CSS:

```
.example1 {  
    text-transform: uppercase;  
}  
.example2 {  
    text-transform: capitalize;  
}  
.example3 {  
    text-transform: lowercase;  
}
```

HTML

```
<p class="example1">  
    all letters in uppercase <!-- "ALL LETTERS IN UPPERCASE" -->  
</p>  
<p class="example2">  
    all letters in capitalize <!-- "All Letters In Capitalize (Sentence Case)" -->  
</p>  
<p class="example3">  
    all letters in lowercase <!-- "all letters in lowercase" -->  
</p>
```

Guion de texto

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

La propiedad `text-indent` especifica la cantidad de texto de espacio horizontal que se debe mover antes del comienzo de la primera línea del contenido de texto de un elemento.

Recursos:

- [¿Sangrar solo la primera línea de texto en un párrafo?](#)
- <https://www.w3.org/TR/CSS21/text.html#propdef-text-indent>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/text-indent>

Decoracion de texto

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

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

`text-decoración` se puede utilizar en combinación con `text-decoración-estilo` y `text-decoración-color` como una propiedad taquigráfica:

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

Esta es una versión abreviada de

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

Cabe señalar que las siguientes propiedades solo son compatibles con Firefox

- `texto-decoración-color`
- `texto-decoración-línea`
- `texto-decoración-estilo`
- `texto-decoracion-saltar`

Desbordamiento de texto

La propiedad de `text-overflow` trata sobre cómo se debe señalar a los usuarios el contenido desbordado. En este ejemplo, los `ellipsis` representan texto recortado.

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

Desafortunadamente, el `text-overflow: ellipsis` solo funcionan en una sola línea de texto. No hay forma de admitir puntos suspensivos en la última línea de CSS estándar, pero se puede lograr con la implementación no estándar de flexboxes solo para webkit.

```
.giveMeEllipsis {  
    overflow: hidden;  
    text-overflow: ellipsis;  
    display: -webkit-box;  
    -webkit-box-orient: vertical;  
    -webkit-line-clamp: N; /* number of lines to show */  
    line-height: X;          /* fallback */  
    max-height: X*N;        /* fallback */  
}
```

Ejemplo (abierto en Chrome o Safari):

<http://jsfiddle.net/csYjC/1131/>

Recursos:

<https://www.w3.org/TR/2012/WD-css3-ui-20120117/#text-overflow0>

Espaciado de palabras

La propiedad de espaciado de palabras especifica el comportamiento de espaciado entre etiquetas y palabras.

Valores posibles

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

CSS

```
.normal     { word-spacing: normal; }  
.narrow      { word-spacing: -3px; }  
.extensive  { word-spacing: 10px; }
```

HTML

```
<p>  
    <span class="normal">This is an example, showing the effect of "word-spacing".</span><br>  
    <span class="narrow">This is an example, showing the effect of "word-spacing".</span><br>  
    <span class="extensive">This is an example, showing the effect of "word-spacing".</span><br>  
</p>
```

Demo en línea

[Inténtalo tú mismo](#)

Otras lecturas:

- [espaciado de palabras - MDN](#)
- [espacio entre palabras - w3.org](#)

Dirección del texto

```
div {  
    direction: ltr; /* Default, text read from left-to-right */  
}  
.ex {  
    direction: rtl; /* text read from right-to-left */  
}  
.horizontal-tb {  
    writing-mode: horizontal-tb; /* Default, text read from left-to-right and top-to-bottom.  
*/  
}  
.vertical-rtl {  
    writing-mode: vertical-rl; /* text read from right-to-left and top-to-bottom */  
}  
.vertical-ltr {  
    writing-mode: vertical-rl; /* text read from left-to-right and top to bottom */  
}
```

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

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

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

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

Variante de fuente

Atributos:

normal

Atributo por defecto de las fuentes.

letras minúsculas

Establece cada letra en mayúsculas, **pero** hace que las letras en minúsculas (del texto original) sean más pequeñas en tamaño que las letras que originalmente estaban en mayúsculas.

CSS:

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

HTML:

```
<p class="smallcaps">  
    Documentation about CSS Fonts  
    <br>  
    aNd ExAmpLe  
</p>
```

SALIDA:

DOCUMENTATION ABOUT CSS Fonts
AND EXAMPLE

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`.

Citas

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

```
q {  
    quotes: "«" "»";  
}
```

Sombra de texto

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

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

Sombra sin radio borroso

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

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

Sombra con radio borroso

Para agregar un efecto de desenfoque, agregue una opción `blur radius` argumento de `blur radius`

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

Sombras múltiples

Para dar un elemento a múltiples sombras, sepáralos con comas.

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

Lea Tipografía en línea: <https://riptutorial.com/es/css/topic/427/tipografia>

Capítulo 53: Transformaciones 2D

Sintaxis

- **Transformar rotar**
- transformar: rotar (<angle>)
- **Traducir Transformar**
- transformar: traducir (<length-or-percentage> [, <length-or-percentage>]?)
- transform: translateX (<length-or-percentage>)
- transform: translateY (<length-or-percentage>)
- **Transformación sesgada**
- transformar: sesgar (<angle> [, <angle>]?)
- transformar: skewX (<angle>)
- transformar: skewY (<angle>)
- **Transformación de escala**
- transform: scale (<scale-factor> [, <scale-factor>]?)
- transform: scaleX (<scale-factor>)
- transform: scaleY (<scale-factor>)
- **Transformada de matriz**
- transformar: matriz (<número> [, <número>] {5,5})

Parámetros

Función / Parámetro	Detalles
rotate(x)	Define una transformación que mueve el elemento alrededor de un punto fijo en el eje Z
translate(x, y)	Mueve la posición del elemento en los ejes X e Y
translateX(x)	Mueve la posición del elemento en el eje X
translateY(y)	Mueve la posición del elemento en el eje Y
scale(x, y)	Modifica el tamaño del elemento en los ejes X e Y
scaleX(x)	Modifica el tamaño del elemento en el eje X
scaleY(y)	Modifica el tamaño del elemento en el eje Y
skew(x, y)	Mapeo de corte, o transvección, distorsionando cada punto de un elemento por un cierto ángulo en cada dirección
skewX(x)	Mapeo de corte horizontal distorsionando cada punto de un elemento por un cierto ángulo en la dirección horizontal

Función / Parámetro	Detalles
<code>skewY(y)</code>	Mapeo de corte vertical distorsionando cada punto de un elemento por un cierto ángulo en la dirección vertical
<code>matrix()</code>	Define una transformación 2D en forma de matriz de transformación.
ángulo	El ángulo por el cual el elemento debe rotarse o sesgarse (dependiendo de la función con la que se use). Ángulo puede ser proporcionado en grados (<code>deg</code>), gradianes (<code>grad</code>), radianes (<code>rad</code>) o vueltas (<code>turn</code>). En la función <code>skew()</code> , el segundo ángulo es opcional. Si no se proporciona, no habrá (0) sesgo en el eje Y.
longitud o porcentaje	La distancia expresada como una longitud o un porcentaje por el cual el elemento debe ser traducido. En la función <code>translate()</code> , la segunda longitud o porcentaje es opcional. Si no se proporciona, entonces no habría (0) traducción en el eje Y.
factor de escala	Un número que define cuántas veces se debe escalar el elemento en el eje especificado. En la función <code>scale()</code> , el segundo factor de escala es opcional. Si no se proporciona, el primer factor de escala también se aplicará para el eje Y.

Observaciones

Sistema coordinador 2D

Las transformaciones se realizan de acuerdo con un sistema de coordenadas X / Y 2D. El eje X va de derecha a izquierda y el eje Y va hacia abajo, como se muestra en la siguiente imagen:

