



DECODED

ACADEMY

FULL STACK DEVELOPER BOOTCAMP

Desarrollamos
{ talento }

The background of the slide features a light gray polygonal grid pattern. Overlaid on this are several large, semi-transparent orange and red triangles. One triangle is positioned in the upper left quadrant, another is at the bottom left, and a smaller one is near the center. A single short orange horizontal line segment is located in the middle-right area of the slide.

CSS

Introducción

¿Qué es CSS?

CSS es acrónimo de Cascade Style Sheets (Hojas de Estilo en Cascada)

Con el uso de CSS se le indica al documento HTML los estilos de la página

Historia del CSS

[W3C](#) Mantiene el estándar.

En el año 1995 se crea la primera versión.

En el año 1998 se publica la segunda especificación 2.0

En el año 2009 se publica la especificación 2.1

Desde el 2011 estamos en la especificación 3.1

Qué podemos hacer con CSS

- Estructurar el layout
- Maquetar los estilos
- Crear transiciones e interacciones simples
- Crear objetos gráficamente sin incluir imágenes que ensucien el HTML
- Generar entornos aptos para móviles

Qué evitar

- Limitarse a formatear textos u objetos
- Utilizar siempre la etiqueta style o el atributo del mismo nombre en el HTML
- Repetir definiciones de características generales

Lo que hay que tener en cuenta

- Todos los objetos son cajas
- Funciona a través de selectores
- Los objetos son modificados con propiedades
- Hay propiedades/selectores incompatibles con algunas navegadores
- Algunas propiedades se heredan de padres e hijos y otras no

Recursos

- developer mozilla CSS
- developer mozilla guía
- Codrops css reference

Uso

Sintaxis

```
nombre-del-selector
{
    /*Comentario*/
    nombre-de-propiedad: valor;
    nombre-de-propiedad: "valor largo con espacios";
    nombre-de-la-n-propiedad: valor;
}
```

Donde podemos definir los estilos.

- Extraerlo en un archivo externo
- Escribiendo CSS dentro de la etiqueta style
- Con el atributo style

Como archivo externo al documento

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

Embebido en el HTML

```
<style>
  body {
    background: red;
  }
</style>
```

Como atributo en la etiqueta style

No recomendado.

```
<p style="color: red;">Importante</p>
```

Selectores

Selectores de etiqueta

```
div {  
    background: blue;  
}  


soy un div </div>


```

Selectores ID

```
#identificadorUnico {  
    background: blue;  
}  
  
<div id="identificadorUnico"> soy un div </div>
```

Selecciona un único elemento

Selectores de clase

```
.miClase {  
    background: blue;  
}  
  
<div class="miClase"> soy un div </div>
```

Selecciona un grupo de elementos

Selector universal

```
* {  
    background: blue;  
}
```

```
<div> soy un div </div>  
<p> soy un p </p>
```

Selecciona cualquier tipo de elemento

Selectores de atributos

```
[data-type] {  
    color: green;  
}  
[data-type = "liquid"] {  
    color: blue;  
}  
[data-type *= "spicy"] {  
    color: red;  
}  
  
<div data-type>Tomate</div>  
<div data-type="spicy solid">Pimiento rojo</div>  
<div data-type="liquid">Vino</div>  
<div data-type="liquid fruit">Zumo</div>
```

- [attr] : Selecciona elementos con el atributo attr independientemente de su valor
- [attr=val] : Selecciona elementos con el atributo attr exactamente igual a val
- [attr^=val] : Selecciona elementos cuyo atributo attr comienza por val.
- [attr\$=val] : Selecciona elementos cuyo atributo attr termina por val.
- [attr*=val] : Selecciona elementos cuyo atributo attr contiene val
- [attr~=val]: Selecciona la palabra val, tanto si empieza o acaba con val, como si está rodeada de espacios.
- [attr|=val] : Selecciona elementos con el atributo attr cuyo valor sea exactamente val o empieza por val-

```
[lang|=es] {
  font-weight: bold;
}
[data-quantity$="kg"] {
  font-weight: bold;
}
[data-vegetable~="liquid"] {
  color: green;
}

<body lang="es-es">
  <div data-quantity="1kg" data-type>Tomate</div>
  <div data-quantity="200gr" data-type="spicy solid">Pimiento
rojo</div>
  <div data-type="liquid">Vino</div>
  <div data-type="liquid fruit">Zumo</div>
  <div data-type="no-liquid">Carne</div>
</body>
```

Pseudo-clases

Una pseudo-clase CSS consta de una clave precedida de dos puntos (:) que añadiremos al final del selector para indicar que daremos estilo a los elementos seleccionados solo cuando estos se encuentren en un **estado determinado**.

```
a:hover {  
    color: darkred;  
    text-decoration: none;  
}
```

:active	:any	:checked
:default	:dir()	:disabled
:empty	:enabled	:first
:first-child	:first-of-type	:fullscreen
:focus	:hover	:indeterminate
:in-range	:invalid	:lang()
:last-child	:last-of-type	:left
:link	:not()	:nth-child()
:nth-last-child()	:nth-last-of-type()	:nth-of-type()
:only-child	:only-of-type	:optional
:out-of-range	:read-only	:read-write
:required	:right	:root
:scope	:target	:valid
:visited		

```
div:first-child {  
    color: blue;  
}  
p:nth-child(2) {  
    color: red;  
}  
p:nth-of-type(2) {  
    color: green;  
}  
p:not(.last) {  
    background: grey;  
}  
  
<section>  
    <div> primer div </div>  
    <p> primer p </p>  
    <div> segundo div </div>  
    <p class="last"> segundo p </p>  
</section>
```

Pseudo-elementos

Los pseudo-elementos hace referencia a elementos que no existen el marcado html. Con ellos podremos seleccionar partes de elementos o crear elementos nuevos.

- ::after
- ::before
- ::first-letter
- ::first-line
- ::selection
- ::placeholder

Seleccionar partes de elementos

```
<a href="http://developer.mozilla.org/en-US/docs/Glossary/CSS">CSS</a>

<a href="http://GeekshubsAcademy.com">GeekshubsAcademy</a>

<input placeholder="email"/>

::first-letter {
  font-size: 25px;
}
::selection {
  /* SOLO PUEDEN MODIFICARSE ESTAS PROPIEDADES */
  background: yellow;
  color: black;
  text-shadow: 2px 2px 2px rgba(0, 0, 0, 0.5);
}
::placeholder {
  color: blue;
}
```

Crear elementos nuevos

```
<a target="_blank" href="http://developer.mozilla.org/en-US/docs/Glossary/CSS">CSS</a>
```

```
<a href="http://GeekshubsAcademy.com">GeekshubsAcademy</a>
```

```
[href*=GeekshubsAcademy]::after {  
    /* CONTENT ES DE USO OBLIGATORIO */  
    content: '⭐';  
}  
[target="_blank"]::before {  
    content: url(external-link.png);  
}
```

Diferentes notaciones :y ::

Lo más probable es que te encuentres la notación :after en lugar de ::after

En CSS1 y CSS2, los pseudo-elementos se definieron con dos puntos (:), al igual que las pseudo-clases (por ejemplo :hover). En CSS3, se introdujo la notación de dobles dos puntos (::) para los pseudo-elementos con el fin de diferenciarlos de las pseudo-clases.

Todos los navegadores que admiten la notación de dobles dos puntos (::) también admiten la notación de dos puntos (:). Internet Explorer 8, sin embargo, no es compatible con la notación de dos puntos (:).

Combinaciones

Combinación	Selecciona
A B	Un elemento seleccionado por B descendiente de un elemento seleccionado por A
A, B	Un elemento seleccionado por A y/o B
A > B	Un elemento seleccionado por B y es hijo directo de un elemento seleccionado por A.
AB	Selecciona A y también B.
A + B	Un elemento seleccionado por B y es el siguiente hermano de un elemento seleccionado por A (o sea, el siguiente hijo del mismo padre).
A ~ B	Un elemento seleccionado por B y es uno de los siguientes hermanos del elemento seleccionado por A (uno de los siguientes hermanos del mismo padre).

Descendiente

```
section a {  
  text-transform: uppercase;  
}  
section .container a {  
  text-transform: unset;  
}
```

Cada selector en la cadena está separado por un espacio en blanco.

Un combinador descendiente se dirige a cualquier elemento que esté anidado dentro de otro elemento.

La anidación puede tener la mayor cantidad de niveles posible, es decir, hijo, nieto y más allá.

Se pueden encadenar para aumentar la especificidad.

```
<style>
  article h2 {
    color: grey;
  }
  aside * {
    color: whitesmoke;
  }
  aside p {
    color: red;
  }
</style>

<article>
  <h2>Lollipop muffin sweet cake</h2>
  <aside>
    <h2>Tiramisu sweet roll ice cream</h2>
    <p>Cake jelly-o jelly-o sweet roll powder</p>
  </aside>
  <p>Cake chocolate cake brownie brownie</p>
</article>
```

Selector O (,)

Selecciona varios elementos con el mismo estilo, separe cada nombre de elemento con una coma.

```
h1, p {  
    color: black;  
}  
.container ,  
.main,  
#footer {  
    text-align: center;  
}
```

Selector Y (dos selectores juntos)

Selecciona un elemento que cumple dos o más condiciones

```
p.container {  
    color: black;  
}  
div.container.main {  
    text-align: center;  
}  
div.container[class][data-foo] {  
    text-align: center;  
}
```

Selector de hijo

Selecciona hijos directos

```
<style>
  article > h2 {
    color: grey;
  }
</style>

<article>
  <h2>Lollipop muffin sweet cake</h2>
  <aside>
    <h2>Tiramisu sweet roll ice cream</h2>
    <p>Cake jelly-o jelly-o sweet roll powder</p>
  </aside>
  <p>Cake chocolate cake brownie brownie</p>
</article>
```

Selectores de hermanos generales

El combinador ~ selecciona hermanos. Esto quiere decir que el segundo elemento sigue después del primero (no necesariamente de forma inmediata), y ambos comparten el mismo elemento padre.

```
<style>
  p ~ h2 {
    color: grey;
  }
</style>

<article>
  <h2>NO SELECCIONADO</h2>
  <p>Cake jelly-o jelly-o sweet roll powder</p>
  <div>chocolate</div>
  <h2>SELECCIONADO</h2>
  <h2>SELECCIONADO</h2>
</article>
```

Selectores de hermanos adyacentes

El combinador + selecciona hermanos adyacentes. Esto quiere decir que el segundo elemento sigue directamente después del primero, y ambos estos comparten el mismo elemento padre.

```
<style>
p + h2 {
  color: grey;
}
</style>

<article>
  <h2>Tiramisu sweet roll ice cream</h2>
  <p>Cake jelly-o jelly-o sweet roll powder</p>
  <div>chocolate</div>
  <h2>Lollipop muffin sweet cake</h2>
  <p>Cake chocolate cake brownie brownie</p>
  <h2>SELECCIONADO</h2>

</article>
```

Valores y unidades

Longitud y tamaño

Usamos unidades de longitud/tamaño muy a menudo en diseños CSS, tipografía y otros.

```
<p>This is a paragraph.</p>
```

```
p {  
    margin: 5px;  
    padding: 10px;  
    border: 2px solid black;  
    font-size: 16px;  
}
```

Unidades absolutas

Nos referimos a píxeles (px) como **unidades absolutas** pues siempre tienen el mismo tamaño independientemente de cualquier otra medida. Otras unidades absolutas:

- mm, cm, in: Milímetros, centímetros, o pulgadas.
- pt, pc: Puntos (1/72 de una pulgada) or picas (12 puntos.)

Unidades relativas

- **em:** 1em es el tamaño de fuente del elemento actual (es el ancho de la letra M mayúscula). El tamaño de fuente por defecto que los navegadores usan antes de aplicar CSS es de 16 píxeles, lo que significa que este es el valor asignado por defecto a un elemento (1em). Ojo – los tamaños de fuente de los elementos se heredan de los padres, por lo que si a los padres se les aplica otros tamaños de fuente.
- **ems son las unidades relativas más usadas en el desarrollo web.**

Unidades relativas

- rem: (em raíz) funciona igual que em, excepto que esta siempre igualará el tamaño del tamaño de fuente por defecto; los tamaños de fuente heredados no afectan, por lo que parece mejor solución que ems, rem no funciona en versiones antiguas de Internet Explorer

Unidades relativas

- vw, vh: Estas son respectivamente 1/100 del ancho de la ventana, y 1/100 de la altura de la ventana. Tampoco son tan soportadas como los rems.

Unidades relativas

- **vmax, vmin:** vmax es la mayor de vh o vw y vmin es la menor de las dos medidas.

Unidades relativas

- ex, ch: Son respectivamente la altura de la x minúscula, y el ancho del número 0. Aunque no son tan soportadas por los navegadores como los ems.

Valores sin unidades

En algunas ocasiones encontramos en CSS valores sin unidades – no siempre es un error, de hecho, es algo permitido bajo determinadas circunstancias.

```
p {  
  margin: 0;  
  line-height: 1.5;  
}
```

El (tamaño de fuente) font-size son 16px; la altura de linea 1.5 veces esta, o sea 24px.

Porcentajes

También podemos usar valores porcentuales para expresar la mayoría de cosas que requieran de valores numéricos, lo que nos permite crear, por ejemplo, cajas cuya anchura siempre cambie según el ancho del contenedor padre. En contraposición a las cajas cuya anchura este definida por un cierto valor (en px o en ems), que siempre serán de la misma longitud, incluso aunque cambie el ancho de los contenedores padres.

```
<div>Fixed width layout with pixels</div>
<div>Liquid layout with percentages</div>

div {
    margin: 10px;
    font-size: 200%;
    color: white;
    height: 150px;
    border: 2px solid black;
}

div:nth-child(2) {
    background-color: blue;
    width: 75%;
}
```

Cascada y herencia

La cascada

CSS (Cascading Style Sheets) indica que el orden de las reglas CSS importa. Que prevalezcan unos selectores sobre otros en la cascada depende de tres factores (en orden de importancia – los primeros prevalecen sobre los últimos):

1. Importancia
2. Especificidad
3. Orden del código

Orden del código

Las últimas reglas prevalecen sobre las primeras

```
p {  
    color: blue;  
}  
  
/* This rule will win over the first one */  
p {  
    color: red;  
}
```

Pero en el siguiente ejemplo, la primera regla gana porque el orden del código es sobrescrito por la especificidad:

```
/* This rule will win */
.footnote {
  color: blue;
}

p {
  color: red;
}
```

Especificidad

La especificidad que tiene un selector se mide mediante 4 valores (o componentes) diferentes, podemos pensar en ellos como en 4 columnas de unidades de millar, centenas, decenas y unidades:

1. Unidades de millar: Puntúa 1 en esta columna si la declaración está dentro de un atributo 2. style (como las declaraciones que no tienen selectores, que su especificidad es siempre 1000). Sino puntúa 0.
2. Centenas: Puntúa 1 en esta columna por cada selector ID contenido en el selector.
3. Decenas: Puntúa 1 en esta columna para cada selector de clase, selector de atributo o de pseudo-clase contenidos en el selector.
4. Unidades: Puntúa 1 en esta columna por cada selector de elemento o pseudo-elemento contenidos en el selector.

Ejemplo Especificidad

Selector	1000	100	10	1	Total
h1	0	0	0	1	0001
.navbar	0	0	1	0	0010
#mildentificador	0	1	0	0	0100
h1 + p::first-letter	0	0	0	3	0003
li > a[href*="en-US"] > .inline-warning	0	0	2	2	0022
#mildentificador div > div > a:hover	0	1	1	3	0113

```
/* specificity: 0101 */
#outer a {
    background-color: red;
}

/* specificity: 0201 */
#outer #inner a {
    background-color: blue;
}

/* specificity: 0104 */
#outer div ul li a {
    color: yellow;
}

/* specificity: 0113 */
#outer div ul .nav a {
    color: white;
}

/* specificity: 0024 */
div div li:nth-child(2) a:hover {
    border: 10px solid black;
}

/* specificity: 0023 */
div li:nth-child(2) a:hover {
    border: 10px dashed black;
}

/* specificity: 0033 */
div div .nav:nth-child(2) a:hover {
    border: 10px double black;
}
```

!important

```
<p class="better worse">This is a paragraph.</p>
<p class="better" id="winning">One selector to rule them all!</p>
```

```
.better {
    background-color: gray;
    border: none !important;
}

.worse {
    background-color: blue;
    border: 1px solid black;
    color: white;
}

#winning {
    background-color: red;
    border: 1px solid black;
}
p {
    border: 1px solid red;
}
```

Herencia

- Unos elementos se heredarán por los elementos hijos, pero otros no.
- Las propiedades que se heredan por defecto y las que no, viene marcado en gran medida por el sentido común.

- Por ejemplo, tiene sentido que `font-family` y `color` sean heredadas, pues nos facilita establecer un ancho de fuente básico aplicando una familia de fuentes al elemento ; después podemos reemplazar las fuentes de elementos individuales si es necesario. Sería realmente molesto tener que establecer la fuente base para cada elemento por separado.
- Otro ejemplo: tiene sentido que `margin`, `padding`, `border`, y `background-image` NO se hereden. Imaginemos el lio de formato/estilo que ocurriría si aplicamos estas propiedades en un elemento y fuera heredado por todos y cada uno de sus hijos, y después tener que "desaplicarlas" a todos los elementos también.

Control de la herencia

CSS dispone de tres valores especiales para manejar las herencias:

- **inherit** : Este valor establece el valor de la propiedad de un elemento seleccionado en el mismo que su elemento padre.
- **initial** : Este valor establece el valor de la propiedad de un elemento seleccionado en el valor por defecto que establece la hoja de estilos del navegador, si este no existe, la propiedad se hereda naturalmente, adoptando el valor de **inherit**.
- **unset** : Este valor reestablece la propiedad a su valor natural, esto es: si la propiedad se hereda de forma natural entonces actuará como **inherit**, sino, actuará como **initial**.

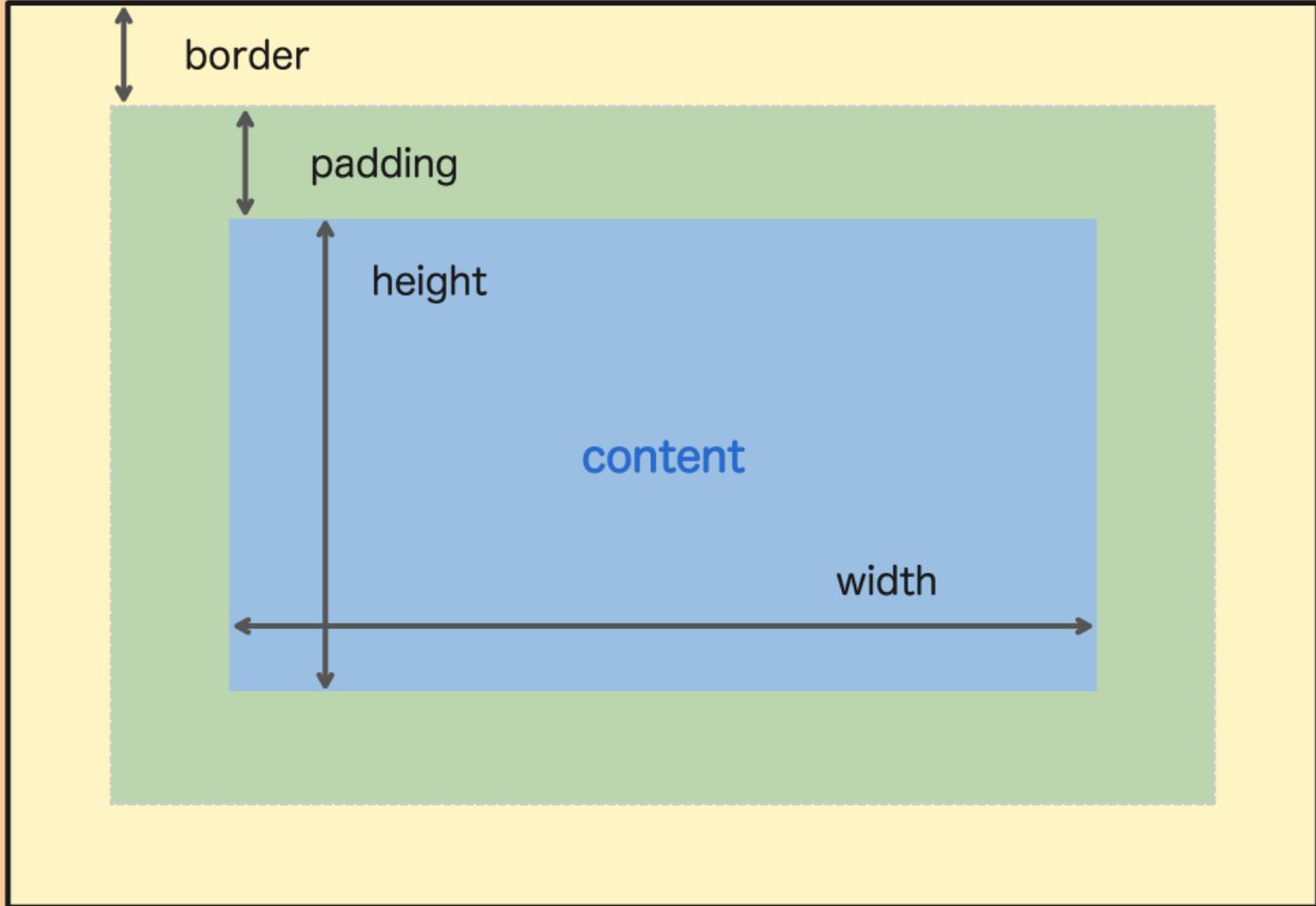
BOXMODEL

Documentación

- [MDN](#)
[mozilla](#)

content, padding, border, margin





box-sizing: content-box;

Es el valor por defecto.

- La altura y la anchura son referentes al contenido.
- padding, border y margin se suman a la altura y/o anchura.

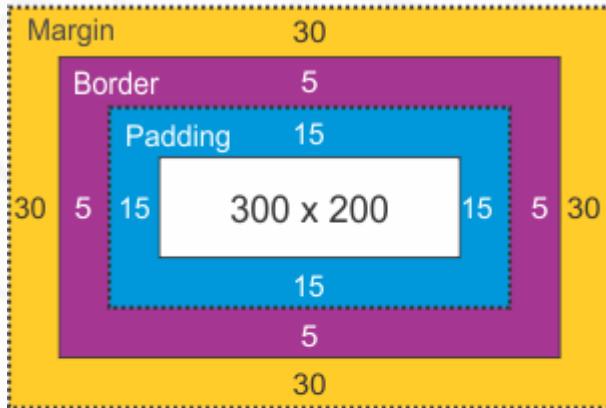
box-sizing: border-box;

PRO TIP Es el valor preferido.

- La altura y la anchura son referentes al contenido + padding + border.
- El margin se suma a la altura y/o anchura.

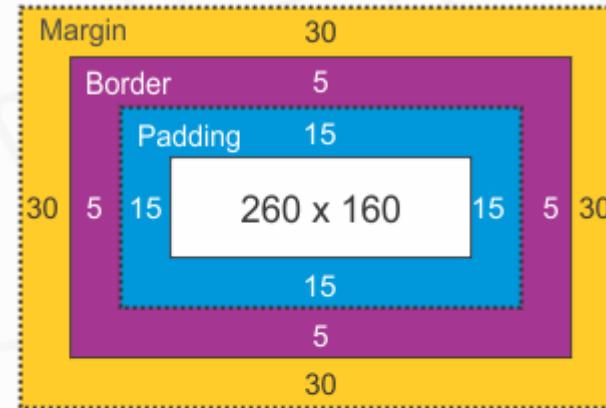
content-box vs border-box

Box Model is content-box



```
div{
    width: 300px;
    height: 200px;
    padding: 15px;
    border: 5px solid grey;
    margin: 30px;
    -moz-box-sizing: content-box;
    -webkit-box-sizing: content-box;
    box-sizing: content-box;
}
```

Box Model is border-box



```
div{
    width: 300px;
    height: 200px;
    padding: 15px;
    border: 5px solid grey;
    margin: 30px;
    -moz-box-sizing: border-box;
    -webkit-box-sizing: border-box;
    box-sizing: border-box;
}
```

Propiedad display

- [Más info](#)
- [Docs](#)

- `display: block;`
 - div, section, header, p, h1, h2, etc..
- `display: inline;`
 - span, strong, em, b, etc...
- `display: none;`
 - Desaparece del DOM
- `display: inline-block;`
 - Inline pero se puede ajustar width y height como en un block
- `display: flex;`
 - Crea un contenedor flexbox (lo veremos más a fondo)
- `display: grid;`
 - Crea un contenedor grid (lo veremos más a fondo)

inline

- El valor predeterminado para elementos como ``, ``, ``, etc...
- Estos elementos dentro de una cadena de texto no interrumpe el flujo del texto.
- Las propiedades `width` y `height` no hacen efecto.

Etiquetas inline

- b, big, i, small, tt
- abbr, acronym, cite, code, dfn, em, kbd, strong, samp, time, var
- a, bdo, br, img, map, object, q, script, span, sub, sup
- button, input, label, select, textarea

block

- Elementos contenedor, como <div>, <section> y . O de texto como <p> y <h1>.
- Por defecto (sin establecer un ancho) ocupan tanto espacio horizontal como pueden.
- Estos elementos rompen el flujo del texto. Se puede decir que actúan como un retorno de carro.
- Las propiedades width y height hacen efecto.

Etiquetas block

address	article	aside	audio
blockquote	canvas	dd	div
dl	fieldset	figcaption	figure
footer	form	h1, h6	header
hgroup	hr	li	main
nav	noscript	ol	output
p	pre	section	table
tfoot	ul	video	

inline block

- Un elemento configurado Inline Block es muy similar al Inline porque se ajustará en línea con el flujo natural de texto (en la "línea base").
- La diferencia es que puedes establecer un ancho y una altura.

Todos los tipos de display:

block	inline	run-in
flow	flow-root	table
flex	grid	ruby
subgrid	block flow	inline table
flex run-in	list-item	list-item block
list-item inline	list-item flow	list-item flow-root
list-item block flow	list-item block flow-root	flow list-item block
table-row-group	table-header-group	table-footer-group
table-row	table-cell	table-column-group
table-column	table-caption	ruby-base
ruby-text	ruby-base-container	ruby-text-container
contents	none	inline-block
inline-table	inline-flex	inline-grid
inherit	initial	unset

Propiedad position

- Docs

- **position: static;**
 - Valor por defecto. Las propiedades top, right, bottom, left, and z-index no tienen efecto
- **position: relative;**
 - top, right, bottom, left lo mueven desde donde está.
- **position: absolute;**
 - El elemento no ocupa espacio donde estaba. top, right, bottom, left lo posicionan respecto a su parent más cercano q está en position: relative. z-index tienen efecto.
- **position: fixed;**
 - El elemento no ocupa espacio donde estaba. top, right, bottom, left lo posicionan en relación al viewport
- **position: sticky;**
 - Se comporta como relative si está dentro del viewport, pero fixed si está fuera del viewport.

Flexbox

- [Reference](#)
- [Docs](#)

Usos

- Centrar verticalmente un bloque dentro de su padre
- Hacer que todos los elementos de un contenedor tengan la misma altura/anchura, independientemente de cuanta altura y anchura esta disponible
- Hacer que todas las columnas tengan el mismo tamaño, incluso cuando tiene diferente cantidad de contenido.

flex-direction

flex-direction especifica cómo colocar los objetos en el contenedor definiendo el eje principal y la dirección (normal o al revés).

flex-direction

Puede tomar 4 valores

row | row-reverse | column | column-reverse

```
.container {  
    display: flex;  
    flex-direction: row;  
}  
  
<div class="container">  
    <div>1</div>  
    <div>2</div>  
</div>
```

justify-content

define cómo el navegador distribuye el espacio entre y alrededor de los items flex, a lo largo del eje principal de su contenedor.

`flex-start | flex-end | center | space-between | space-around | space-evenly`

justify-content

```
.container {  
    display: flex;  
    justify-content: space-around;  
}
```

align-items

Alinea los elementos de la línea flexible actual de la misma forma que justify-content, pero en dirección perpendicular. flex-start | flex-end | center | stretch | baseline

align-items

```
.container {  
    /* Alinea los elementos al borde de inicio */  
    align-items: flex-start;  
    /* Alinea los elementos al borde de fin */  
    align-items: flex-end;  
    /* Centra los elementos en el eje transversal */  
    align-items: center;  
    /* Alinea la base de los elementos */  
    align-items: baseline;  
    /* Estira los elementos para ajustarlos */  
    align-items: stretch;  
    /* Valores globales */  
    align-items: inherit;  
    align-items: initial;  
    align-items: unset;  
}
```

align-self

Alinea los elementos hijos reemplazando el valor de align-items

auto | flex-start | flex-end | center | baseline | stretch

```
.item {  
    align-self: auto;  
    align-self: flex-start;  
    align-self: flex-end;  
    align-self: center;  
    align-self: baseline;  
    align-self: stretch;  
}
```

flex

La propiedad CSS flex es una propiedad resumida que indica la capacidad de un elemento flexible para alterar sus dimensiones y llenar el espacio disponible. Los elementos flexibles pueden ser estirados para utilizar el espacio disponible proporcional a su factor de crecimiento flexible o su factor de contracción flexible para evitar desbordamiento.

[docs](#)

```
.item {  
  /* 0 0 auto */  
flex: none;  
/* Un valor, número sin unidades: flex-grow */  
flex: 2;  
/* Un valor, width/height: flex-basis */  
flex: 10em;  
flex: 30px;  
flex: auto;  
flex: content;  
/* Dos valores: flex-grow | flex-basis */  
flex: 1 30px;  
/* Dos valores: flex-grow | flex-shrink */  
flex: 2 2;  
/* Tres valores: flex-grow | flex-shrink | flex-basis */  
flex: 2 2 10%;  
}
```

order

La propiedad CSS `order` especifica el orden utilizado para disponer los elementos en su contenedor flexible. Los elementos estarán dispuestos en orden ascendente según el valor de `order`. Los elementos con el mismo valor de `order` se dispondrán en el orden en el cual aparecen en el código fuente.

flex-wrap

La propiedad **flex-wrap** de CSS especifica si los elementos "hijos" son obligados a permanecer en una misma línea o pueden fluir en varias líneas.

nowrap | wrap | wrap-reverse

Todos las propiedades flexbox

align-content
flex
flex-flow
flex-wrap

align-items
flex-basis
flex-grow
justify-content

align-self
flex-direction
flex-shrink
order

Soporte en navegadores

- caniuse

CSS grid

- [Docs](#)

Que es un grid

Una rejilla es un conjunto de líneas horizontales y verticales que se intersectan - un conjunto que define columnas y el otro filas. Los elementos se pueden colocar en la rejilla respetando estas líneas de columnas y de filas.

El contenedor Grid

Creamos un contenedor de rejilla al declarar `display: grid` o `display: inline-grid` en un elemento.

Todos los hijos directos de ese elemento se convertirán en elementos de la rejilla.

Definiendo columnas y filas

Definimos filas y columnas en el contenedor grid con las propiedades `grid-template-columns` y `grid-template-rows`. Estos espacios son llamados vías.

```
/* Rejilla 3 x 3*/
.containerGrid {
  display: grid;
  grid-template-columns: 100px 200px 150px;
  grid-template-rows: 20vh 30vh 50vh ;
}
```

La unidad fr

Las vías se pueden definir usando cualquier unidad de longitud. Grid también presenta una unidad de longitud adicional para ayudarnos a crear vías de rejilla flexibles. La nueva unidad `fr` representa una fracción del espacio disponible en el contenedor de la rejilla.

```
.containerGrid {  
  display: grid;  
  grid-template-columns: 1fr 3fr 1fr;  
}
```

La función repeat()

Rejillas grandes con muchas vías pueden utilizar la notación repeat() con le fin de repetir todas o una sección de la lista de vías.

```
.containerGrid {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    /* Es lo mismo que: */  
    /* grid-template-columns: 1fr 1fr 1fr; */  
}
```

Uso avanzado de repeat

```
.containerGrid {  
    display: grid;  
    grid-template-columns: 10vw repeat(3, 1fr) 3em;  
    /* Es lo mismo que: */  
    /* grid-template-columns: 10vw 1fr 1fr 1fr 3em; */  
}  
  
.containerGrid {  
    display: grid;  
    grid-template-columns: repeat(2, 1fr 2fr 3fr);  
    /* Es lo mismo que: */  
    /* grid-template-columns: 1fr 2fr 3fr 1fr 2fr 3fr; */  
}
```

Tamaño de vía y minmax()

Al configurar una rejilla es posible que se desee dar a las vías un tamaño mínimo, pero asegúrese de que se expandan para adaptarse a cualquier contenido que se pueda agregar.

Con `minmax(100px, 300px)` se puede definir un tamaño mínimo y otro máximo para poder adaptarse a diseños responsive

```
.containerGrid {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    grid-template-rows: minmax(100px, auto);  
}
```

Espacio entre filas y columnas

La propiedad CSS grid-gap es una propiedad abreviada para grid-row-gap y grid-column-gap que especifica el espacio entre las filas y las columnas de la cuadrícula.

```
.containerGrid {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    grid-gap: 20px;  
    /* igual que: */  
    /* grid-row-gap: 20px; */  
    /* grid-column-gap: 20px; */  
    grid-gap: 3em 6em;  
    /* igual que: */  
    /* grid-row-gap: 3em; */  
    /* grid-column-gap: 6em; */  
  
}
```

Ejemplo sencillo

```
<div class="containerGrid">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>

.containerGrid {
  display: grid;
  grid-template-columns: 1fr 3fr 1fr;
  /* Las filas se crean implícitamente si hubiera mas de 3 elementos */
}
```

CSS grid en diseños complejos

Existen muchas más propiedades que no vamos a cubrir, pero debéis saber que css grid es muy potente y quien quiera profundizar más puede revisar los siguiente puntos:

- [grid-template-areas y grid-area](#)
- [filling multiples rows or columns](#)
- [justify-items y align-items](#)
- [explicit grid vs implicit grid](#)
- [grid auto flow](#)
- [grid shorthands](#)

Soporte en navegadores

- caniuse

Formato de Texto

Propiedad color

Se puede establecer en formato hexadecimal o rgba

```
color:#ff5bc3;
```

Propiedad font-family

Define la familia tipográfica

```
font-family: "Avant Garde CE", Helvetica, sans-serif;
```

Propiedad font-size

Define el tamaño del texto (px, em)

```
font-size: 50px;
```

Propiedad font-weight

Establece el peso/grosor del texto

Puede ser bold, normal o el valor en puntos de 100 a 900)

```
font-weight:bold;
```

Propiedad font-style

Cambia el estilo de texto

Puede ser: normal, italic, oblique

```
font-style:italic;
```

Propiedad text-transform

Fuerza la utilización de un estilo de escritura

Valores posibles:uppercase, lowercase, capitalize

`text-transform:uppercase;`

Propiedad text-decoration

Define la decoración del texto

Valores posibles: none, underline, overline y line-through

```
text-decoration:underline;
```

Propiedad text-indent

Genera un sangrado para el texto

Sus valores pueden ser positivos o negativos

```
text-indent:20px;
```

Propiedad text-align

Alinea el texto de la caja

Sus valores posibles: left, right, center, justify

```
text-align:right;
```

Propiedad letter-spacing

Genera una separación entre caracteres.

Pueden ser también positivos o negativos

```
letter-spacing: 3px;
```

Propiedad line-height

Aumenta o reduce el interlineado de un texto

```
line-height:30px;
```

Background

Propiedad background-color

Define el color de fondo para el objeto

Puede ser hexadecimal o rgba

```
background-color: #3f9ff5
```

Propiedad background-image

Establece una imagen de fondo para el objeto

```
background-image:url(../images/fondo.jpg);
```

Propiedad background-repeat

Define la forma de repetición del fondo

Sus valores pueden ser: repeat, repeat-y, repeat-x y no repeat

```
background-repeat: repeat-y;
```

Propiedad background-position

Indica la posición del fondo

Sus valores son: center, top, left, right, bottom y coordenadas x,y

```
background-position: top center;
```

Propiedad background-size

Define el tamaño del fondo del objeto

```
background-size: cover;
```

Otras tecnologías que se han desarrollado a partir de CSS

- Sass
- Lesscss
- PostCSS

Funciones

Media queries