

Layout

CFP
Programador
full-stack

Contenedores

CFP
Programador
full-stack

Contenedores

**<div> & **

- Son simples contenedores de HTML.
- Los **div** y **span** como todos los elementos, tienen las propiedades `class` o `id`.
- Es importante elegir un nombre de clase o `id` con sentido en el contexto.

Div & Span

DIV

- Es un elemento que define un bloque
- Generalmente para secciones largas del sitio
- Puede incluir varios elementos
- Nos ayuda a construir el layout y el diseño

SPAN

- Es un elemento “inline”
- Usado para agrupar texto, palabras o frases. Por ejemplo dentro de un párrafo

Bloque <div> ... </div>

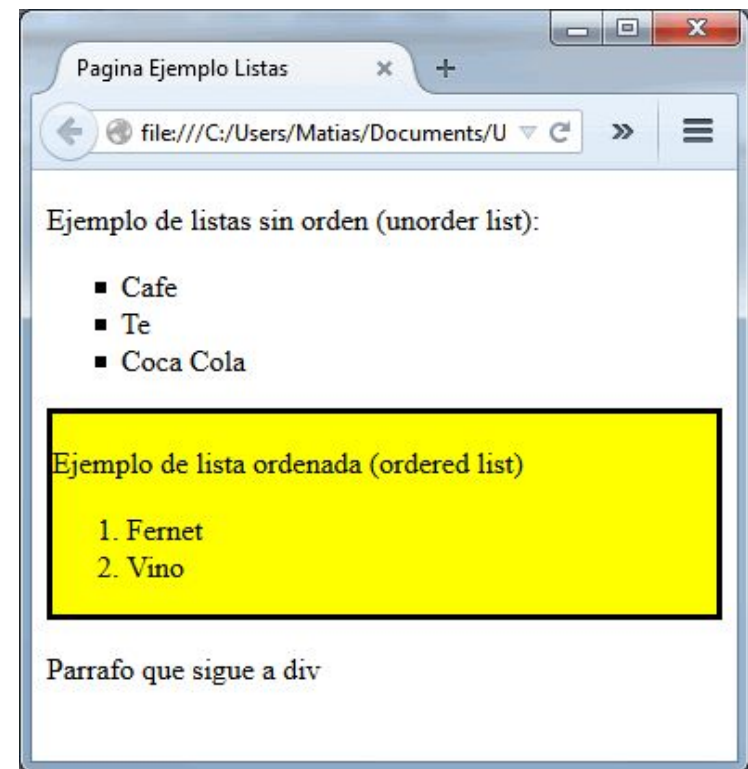
- Por defecto el elemento empieza en una nueva línea de la página y ocupa todo el ancho disponible
- Se pueden anidar uno dentro de otro

```
<!DOCTYPE html>
<html>
<head>
<title>Pagina Ejemplo Listas</title>
<link rel="stylesheet" href="estilo.css">
</head>
<body>

<p>Ejemplo de listas sin orden (unordered list):</p>
<ul>
  <li>Cafe</li>
  <li>Te</li>
  <li>Coca Cola</li>
</ul>

<div class="bloque1">
<p>Ejemplo de lista ordenada (ordered list)</p>
<ol>
  <li>Fernet</li>
  <li>Vino</li>
</ol>
</div>
<p>Parrafo que sigue a div</p>

</body>
</html>
```



Bloque ...

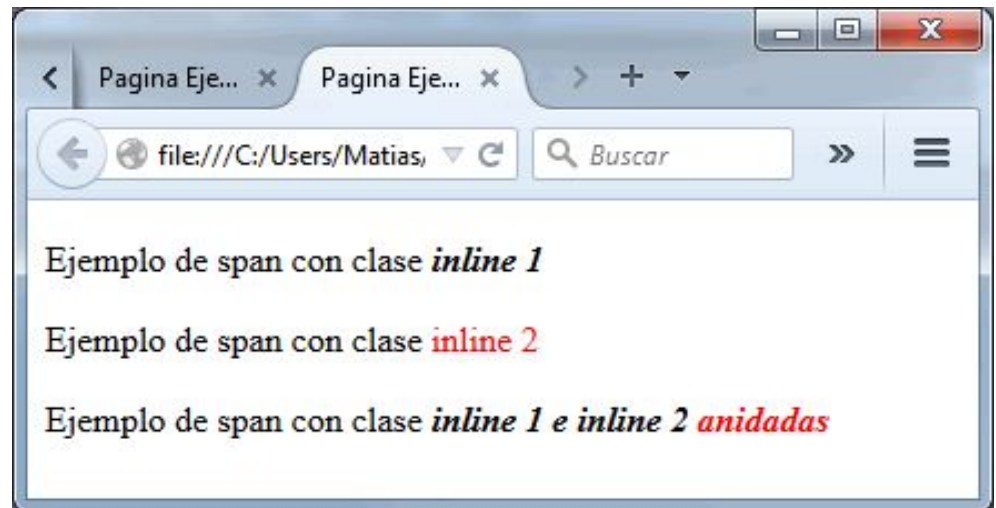
- Están **dentro del texto**, no en una línea nueva
- Su ancho depende del contenido que tengan
- No pueden anidarse con otro elemento de bloque
- Pero si se pueden anidar con otro elemento inline

<p>Ejemplo de span con clase inline 1</p>

<p>Ejemplo de span con clase inline 2</p>

<p>Ejemplo de span con clase inline 1 e inline 2 anidadas </p>

```
.inline1{  
font-style: italic;  
font-weight: bold;  
}  
.inline2{  
color: red;  
}
```



Ejercicio

Agregar a la página que venían haciendo:

- Un div que incluya varios elementos (título, párrafo, listas, etc), darle un estilo
- Más de un span en párrafos, darle un estilo

Box Model

CFP

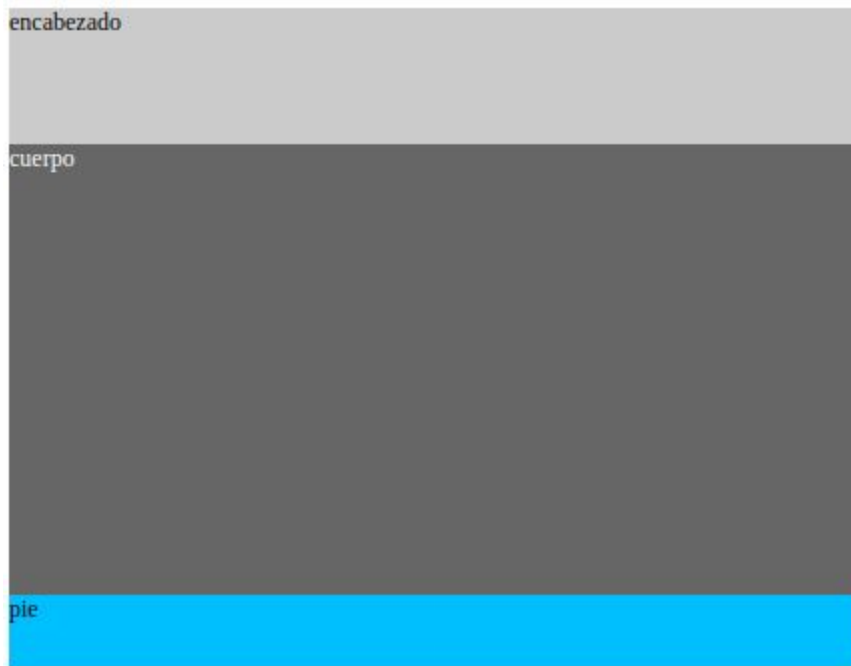
**Programador
full-stack**

Box Model - Introducción

El concepto de “**Box Model**” dice que cada elemento en una página se representa mediante una caja rectangular (contenedor).

- CSS permite controlar el aspecto y ubicación de las cajas.
- Todos los elementos HTML están representados como cajas. Es común que fondo y borde no sean visibles a simple vista, pero el concepto de **Box Model siempre es utilizado**
- Este concepto es difícil de entender, pero fundamental para construir y diagramar sitios

Usemos contenedores



HTML

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

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

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

CSS

```
.encabezado {  
  height: 90px;  
  text-align: left;  
  background-color: #CCCCCC;  
}
```

```
.cuerpo {  
  background-color: #666666;  
  color: white;  
  height: 300px  
}
```

```
.pie {  
  background-color: #00CCFF;  
  height: 50px  
}
```

Box Model

CSS utiliza el modelo de cajas / bloques que consta de 4 partes:

- CONTENT
- PADDING
- BORDER
- MARGIN



Box Model - CONTENT

- **CONTENT**
- PADDING
- BORDER
- MARGIN



ALTO (height) y
ANCHO (width) de
un elemento.



Box Model - PADDING

- CONTENT
- **PADDING**
- BORDER
- MARGIN

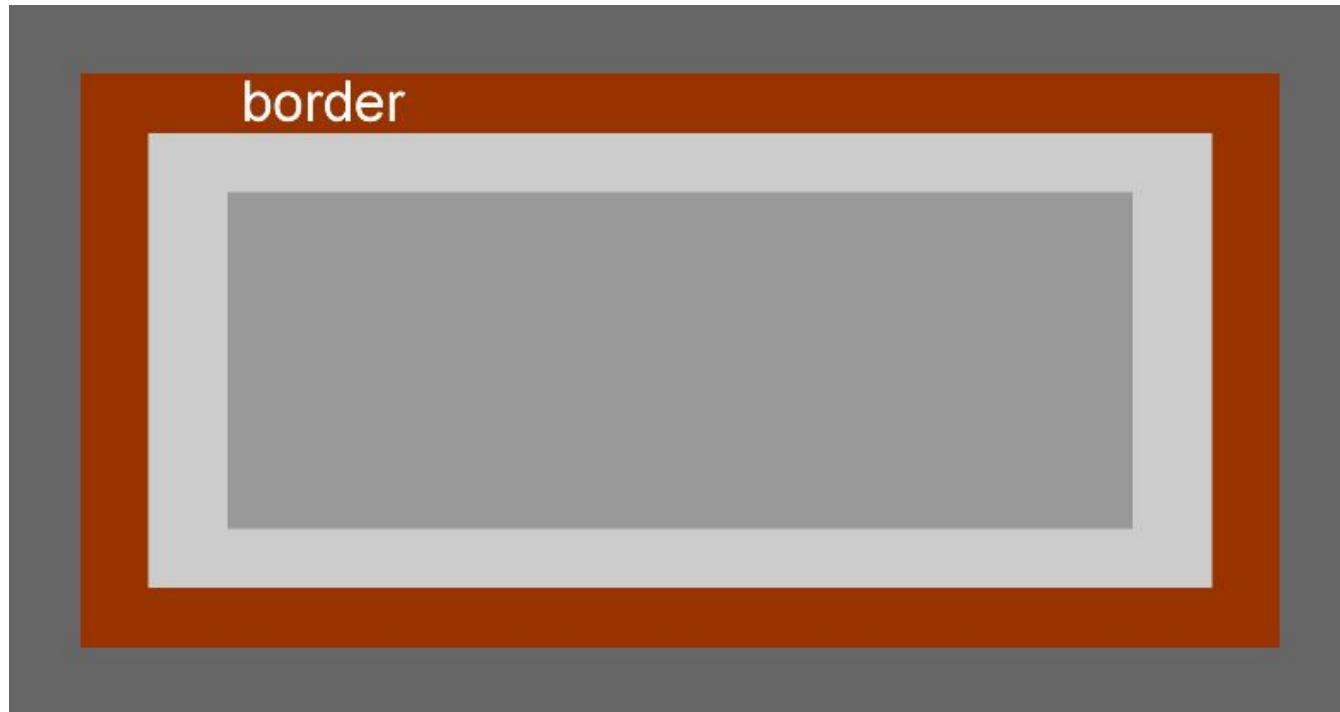
Usado para generar **espaciado o margen INTERIOR** transparente dentro de un elemento.



Box Model - BORDER

- CONTENT
- PADDING
- **BORDER**
- MARGIN

Se utiliza para bordear con una línea alrededor del elemento.



Box Model - MARGIN

- CONTENT
- PADDING
- BORDER
- **MARGIN**

Usado para generar **margen EXTERIOR** transparente fuera de un elemento.

PUEDE USARSE PARA SEPARAR BLOQUES



Box Model

```
HTML
1 <div>
2 <h1>Bloque Box Model</h1>
3 <ul>
4 <li>Ancho 600px</li>
5 <li>Alto 250px</li>
6 <li>Padding 20px</li>
7 <li>Border 4px Solid Rosa</li>
8 <li>Margin 50px</li>
9 </ul>
10 </div>
11

CSS
1 div {
2   width: 600px;
3   height: 250px;
4   margin: 50px;
5   padding: 20px;
6   border: 4px;
7   border-style: solid;
8   border-color: pink;
9
10  background-color: lightblue;
11 }
12
13
```

Bloque Box Model

- Ancho 600px
- Alto 250px
- Padding 20px
- Border 4px Solid Rosa
- Margin 50px



Live: <http://codepen.io/webUnicen/pen/yMRawg>

Box Model

Y si usamos tamaños de **propiedades irregulares?**

```
HTML
1 <div>
2 <h1>Bloque Box Model</h1>
3
4 </div>
5

CSS
1 div {
2   width: 600px;
3   height: 250px;
4   background-color: lightblue;
5   padding-top: 5px;
6   padding-bottom: 20px;
7   padding-left: 100px;
8   padding-right: 0;
9   border: 4px;
10  border-style: solid;
11  border-color: pink;
12  margin-left: 50px;
13  margin-right: 15px;
14  margin-top: 5px;
```

Bloque Box Model

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

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

Unidades de Medida

CFP
Programador
full-stack

Unidades de medida

CSS divide las unidades de medida en dos grupos

ABSOLUTAS: pixeles (px) (pt - mm - cm)

Están completamente definidas, ya que su valor **no depende de otro valor de referencia**.

- Ajustan tamaños fijos en los navegadores y pantallas.
- Poca flexibilidad.
- Sirve cuando conocemos tamaños de las salidas.

RELATIVAS: porcentaje (%) (em - rem - vw - vh)

No están completamente definidas, ya que su valor **siempre es dependiente respecto a otro valor de referencia padre**.

- Permiten ajustes con cambios de tamaños de pantalla.
- Mayor flexibilidad.

Medida “Auto”

La opción **auto** significa “**expandir**” o ajustar automáticamente un tamaño.

Para centrar un bloque:

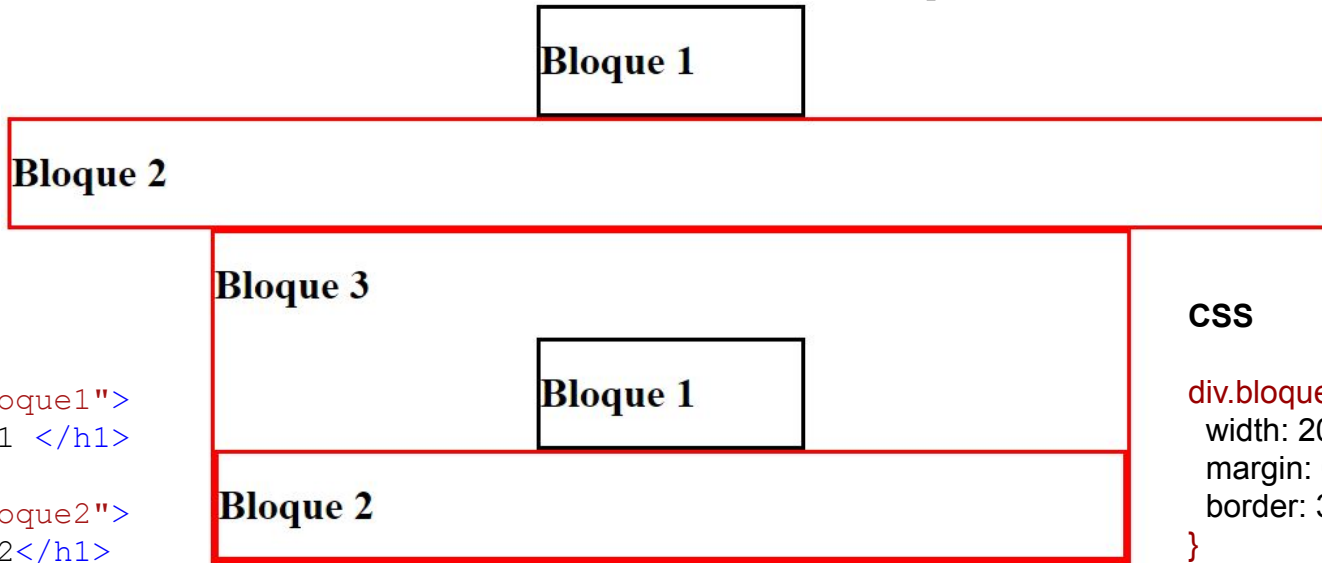
```
.bloque {  
    width: 100px;  
    margin: 0 auto;  
}
```

Para hacerlo ocupar el máximo espacio

```
.bloque {  
    width: auto;  
}
```

Medida “Auto”

Posicionando elementos con opción “auto”



HTML

```
<div class="bloque1">
  <h1>Bloque 1 </h1>
</div>
<div class="bloque2">
  <h1>Bloque 2</h1>
</div>
<div class="bloque3">
  <h1>Bloque 3</h1>
  <div class="bloque1">
    <h1>Bloque 1 </h1>
  </div>
  <div class="bloque2">
    <h1>Bloque 2</h1>
  </div>
</div>
```

CSS

```
div.bloque1 {
  width: 200px;
  margin: 0 auto;
  border: 3px solid;
}

div.bloque2 {
  width: auto;
  border: 3px solid red;
}

div.bloque3 {
  width: 700px;
  border: 3px solid red;
  margin: 0 auto;
}
```



Live: <http://codepen.io/webUnicen/pen/qRQzQd>

Tipos de cajas

CFP
Programador
full-stack

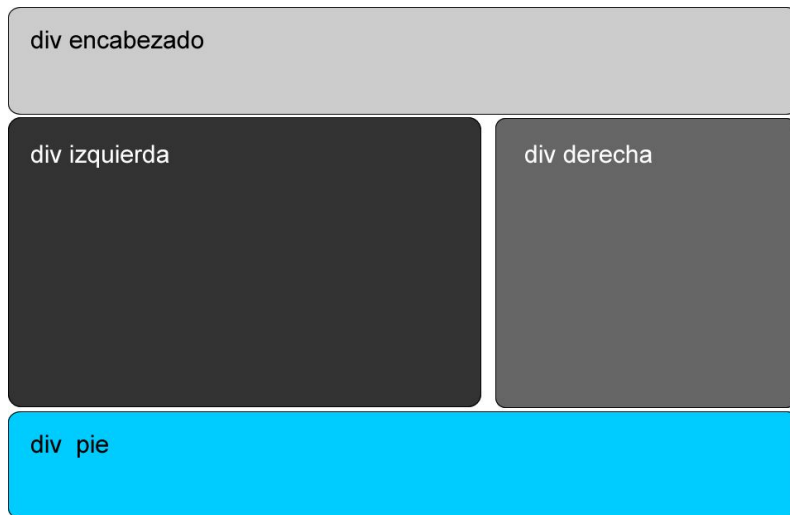
Pregunta

¿Pero cómo hacemos si queremos posicionar elementos de una manera más avanzada?

Layout - Un bosquejo

¿Cómo queremos que se vea?

EXPECTATIVA



REALIDAD

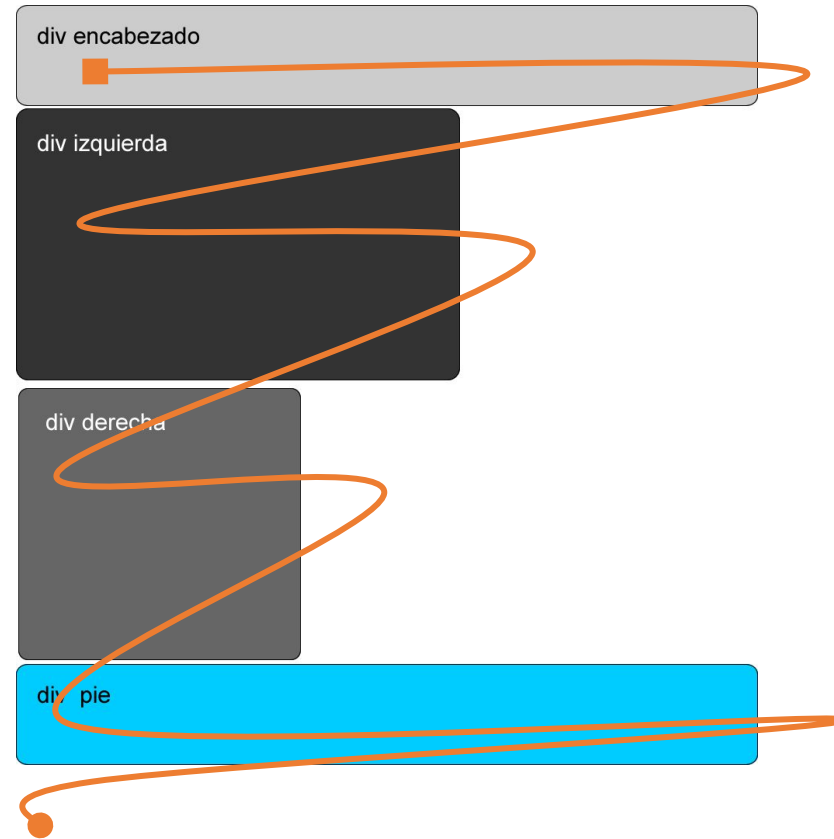


Posicionamiento

Bloques flujo normal sin posicionamiento

¿Qué pasó?

- Definimos medidas de las columnas, pero el flujo de la página las apilo una abajo de otra.
- Cada caja pone un “enter”



Tipos de Cajas (Type of Boxes)

CSS puede definir la manera en la que los elementos de una página “**encajan**” uno con otros. Estas maneras se pueden categorizar en:

- **BLOCK**
- **INLINE**

Se controlan con la propiedad **display**.

```
elemento {  
    display: block | inline ;  
}
```

Tipos de Cajas

Block vs Inline

`<h1>...<h5>, <p>, <div>`

element-1 {display: block}

element-2 {display: block}

element-3 {display: block}

element-4 {display: block}

`<a>, , , ...`

element-1
{ display:
inline}

element-2
{ display:
inline}

element-3
{ display:
inline}

Tipos de Cajas - Block vs Inline

Las cajas **block** por defecto se apilan una encima de otra.

Las cajas **inline** no mueven los elementos alrededor de ellas.

BLOCK BOX

Este es un simple párrafo de ejemplo.

Este es *otro simple* párrafo de ejemplo.

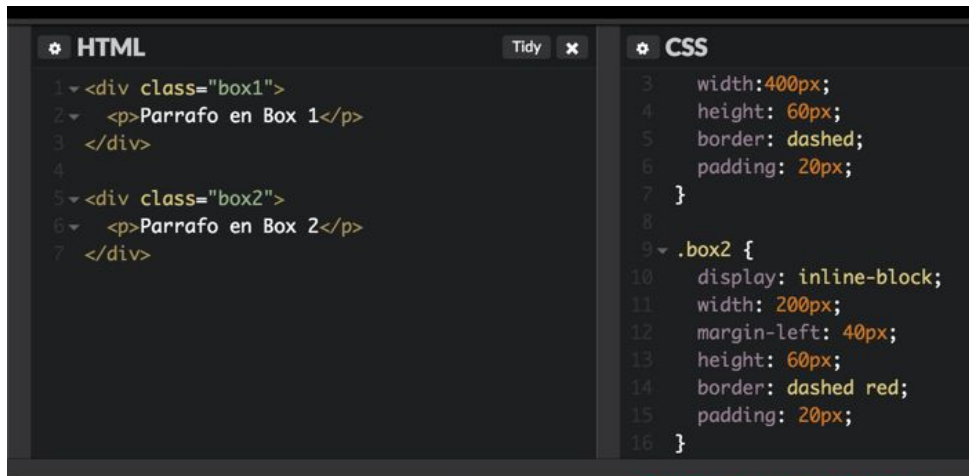
INLINE BOX

Otras Propiedades de Posicionamiento



display:
inline-block; es una combinación de los dos tipos anteriores.

display: none;
oculta el elemento de la página.

A screenshot of a code editor with two panels. The left panel is titled "HTML" and shows the following code:

```
1 <div class="box1">
2   <p>Parrafo en Box 1</p>
3 </div>
4
5 <div class="box2">
6   <p>Parrafo en Box 2</p>
7 </div>
```

The right panel is titled "CSS" and shows the following code:

```
3 width:400px;
4 height: 60px;
5 border: dashed;
6 padding: 20px;
7 }
8
9 .box2 {
10  display: inline-block;
11  width: 200px;
12  margin-left: 40px;
13  height: 60px;
14  border: dashed red;
15  padding: 20px;
16 }
```

Se usa comúnmente con JavaScript para ocultar y mostrar elementos sin eliminarlos ni volver a crearlos.

Parrafo en Box 1

Parrafo en Box 2

Botonera

Hagamos una botonera para nuestro sitio



Podemos usar una lista:

```
<ul class="navigation">  
  <li>Home</li>  
  <li>Productos</li>  
  <li>Nosotros</li>  
  <li>Contacto</li>  
</ul>
```



- Home
- Productos
- Nosotros
- Contacto

¿Cómo hacemos para que se vea como una botonera?



<https://codepen.io/webUnicen/pen/oqybjQ>

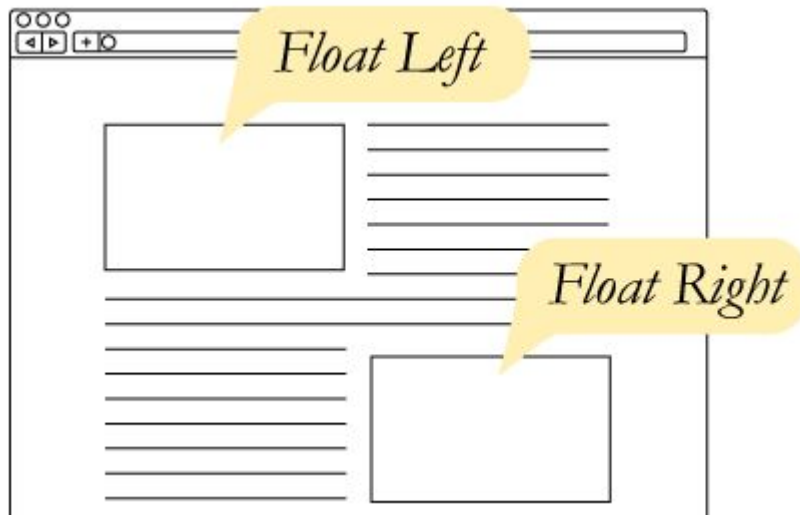
Flotar

CFP

Programador full-stack

Flotar Elementos

La propiedad CSS **float** permite a los elementos colocarse del lado **izquierdo** o **derecho** de su contenedor, permitiendo que el texto y los elementos en línea se envuelvan alrededor de él.



- A diferencia de **inline-block** podemos indicar cómo se posicionan los elementos dentro del flujo de la página (izquierda/derecha y como se comportan algunos elementos por dentro).
- Muy útil para diseñar el layout del sitio.

Layouts usando float

Creemos el layout de ejemplo flotando los elementos



```
.izquierda {  
  float: left;  
  width: 400px;  
  background-color: #333333;  
  color: white;  
}
```

```
.derecha {  
  float: left;  
  width: 300px;  
  background-color: #666666;  
  color: white;  
}
```

Flotar Elementos - Cosas Importantes

- Cuando se flotan elementos, van a ir hasta el borde de su contenedor padre.
- Si no tiene padre, va a ir hasta el borde de la página.
- Si tengo muchos elementos flotados a la derecha, se van a acomodar uno al lado de otro según el flujo de la misma página.

Limpiando elementos flotados

- Cuando un elemento es flotado, rompe con el flujo normal de la página.
- A veces esto es bueno y esperable.
- Para poder flotar un elemento y luego volver al flujo normal de la página siempre usar la propiedad “**clear**”.
- Cuando un elemento tiene esa propiedad, a partir de ese punto en adelante la página vuelve al flujo normal.

Flotar Elementos



- Analizar que sucede al modificar el tamaño de la ventana del navegador.
- Hacer que el div de la derecha flote a la izquierda, y notar que cambios se producen.
- Quitar el clear del pie

```

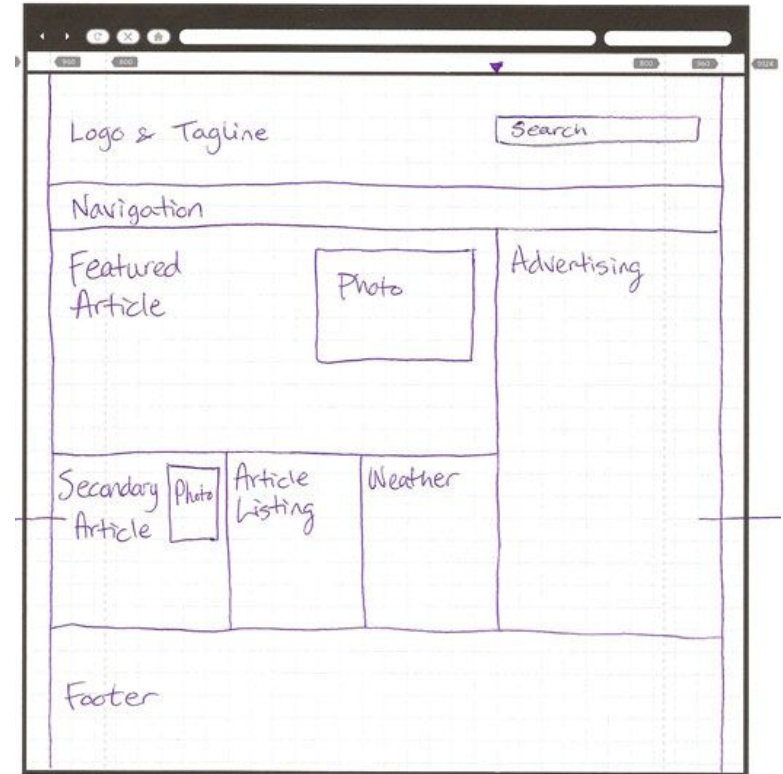
CSS
28
29 div.izquierda {
30   float: left;
31   width: 200px;
32   background-color: #333333;
33   color: white;
34 }
35
36 div.centro {
37   float: left;
38   width: 200px;
39   background-color: #99cc33;
40   color: white;
41 }
42
43 div.derecha {
44   float: right;
45   width: 200px;
46   background-color: #666666;
47   color: white;
48 }
49 div.pie {
50   clear: both;
51   background-color: #00CCFF;
52 }
53
```

Hacer un layout completo

CFP
Programador
full-stack

Sugerencia

Hacer un **diagrama del layout en papel**, lo más completo posible y con sus medidas. (wireframe)



Un vez que se tiene una idea clara del diseño que se desea lograr, comenzar a escribir código para ajustarlo al diseño.

Ejercicio

Realizar una página que contenga:

- Dos o más divs, uno dentro del otro.
- A cada uno darle las propiedades vistas (borde, padding, margen, tamaño) y contenido (títulos, párrafo, imagen).
- Probar cómo se modifica la apariencia cambiando el tamaño, el padding, márgenes y bordes.
- Agregar div con tamaño en porcentaje, ver qué sucede cuando achicamos la ventana del navegador.

Ejercicio

Modificar el ejemplo para que la página quede con un modelo como el siguiente



Experimenten distintos valores de las propiedades de las cajas, de las propiedades float y clear. Es **importante** que comprendan cómo funcionan.

Referencias



HTML & CSS - Design and Build Websites.

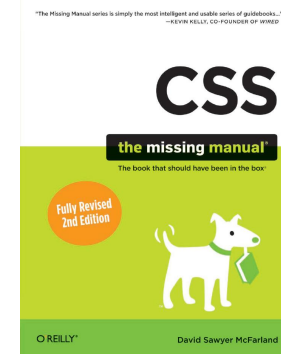
JON DUCKETT

<https://drive.google.com/open?id=0B4N5SXjhTVLtMnltSkpNTXktaEE>

CSS - the missing manual.

DAVID SAWYER MCFARLAND

<https://drive.google.com/open?id=0B4N5SXjhTVLtaWJDWDIaR3BJTkk>



Unidades en CSS <https://www.w3.org/Style/Examples/007/units.en.html>