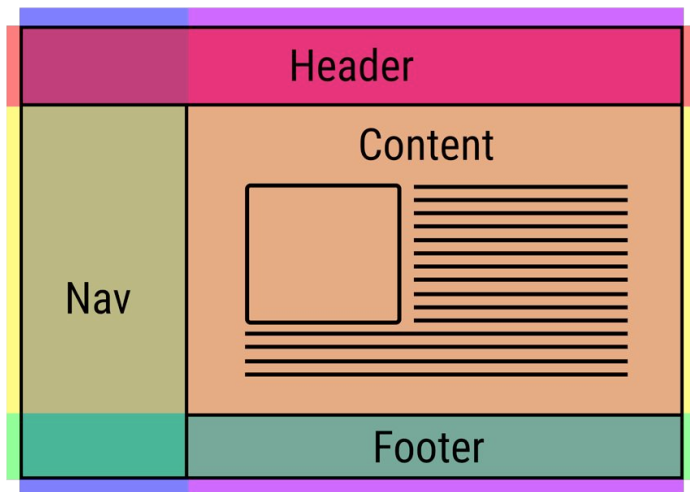


LAYOUTS

WEB 1 TUDAI 2024

Layouts

Un **layout** define la estructura básica de la *interfaz de usuario* en una aplicación.



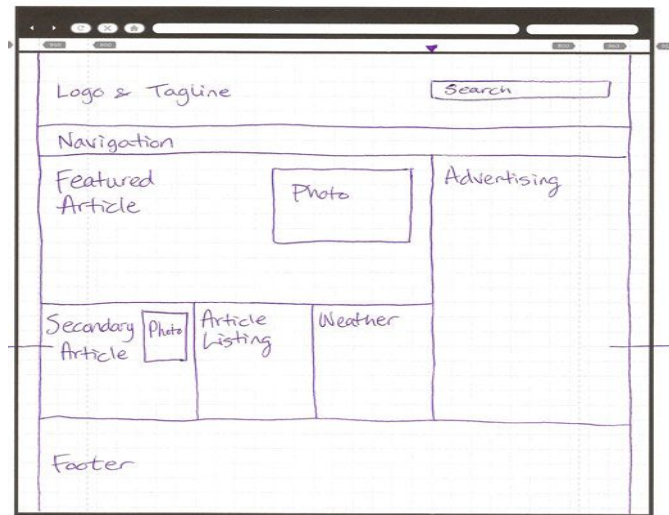
“Es el esqueleto general de la página”

Layouts

¿Cómo se empieza?

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.



Layouts en formato Digital

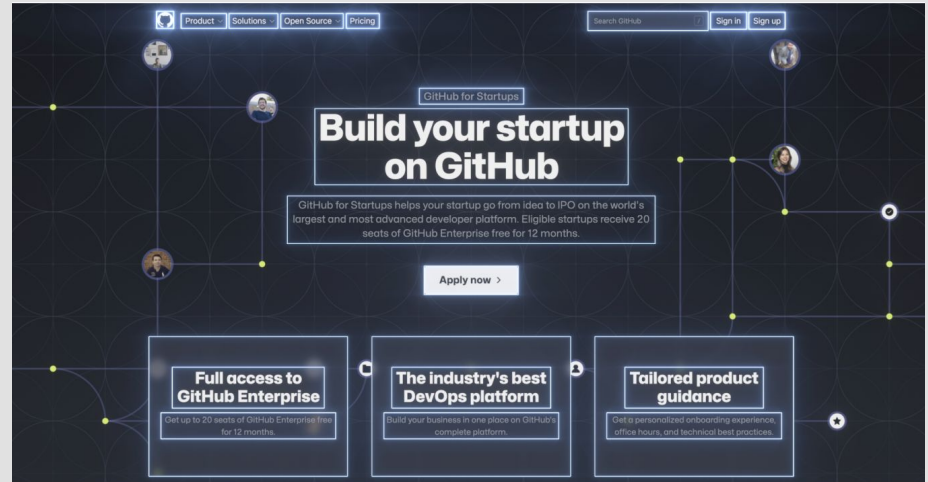
Figma, Photoshop



Box Model

BOX MODEL

El concepto de **Box Model** dice que cada elemento en una página se construye mediante “cajas rectangulares”, llamadas **contenedores**



Box Model: Intro

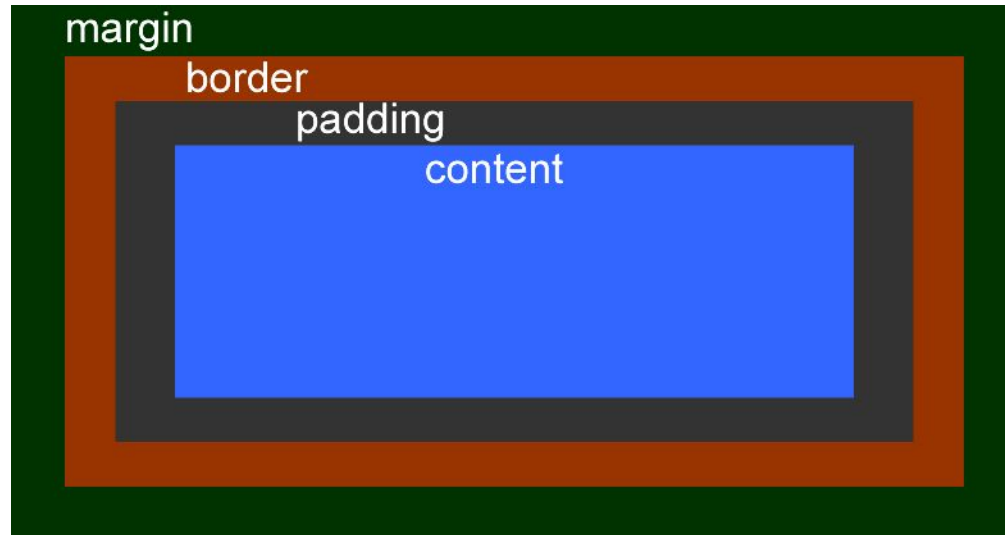
Box Model es un **concepto fundamental** para construir y diagramar sitios.

- Todos los elementos HTML están representados como cajas
- Box Model siempre es utilizado
- CSS permite controlar el aspecto y ubicación de las cajas



Box Model

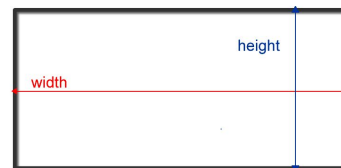
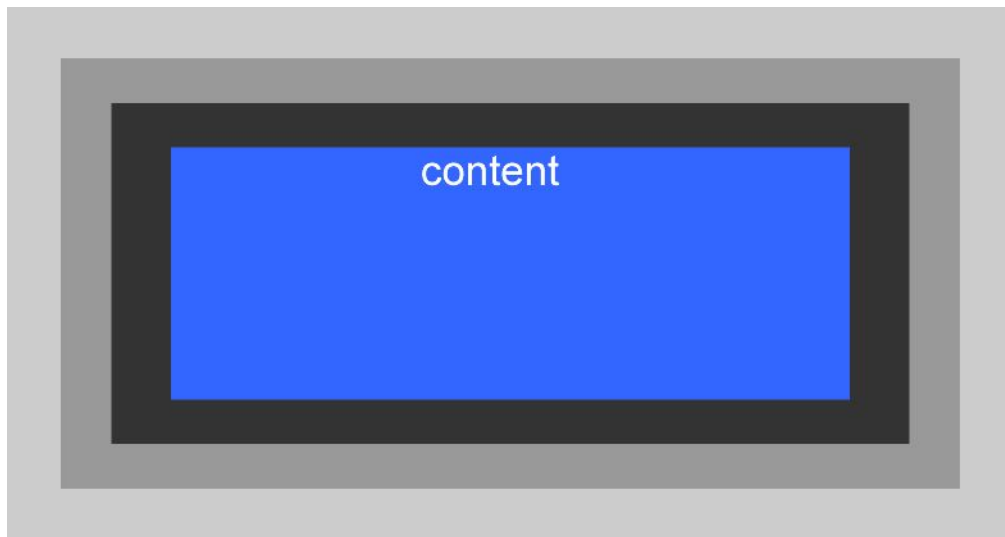
CSS utiliza el modelo de cajas/bloques y define 4 “partes” importantes:



Box Model - CONTENT

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

El contenido de la caja, donde aparecen texto, imágenes, etc.



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

Box Model - PADDING

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

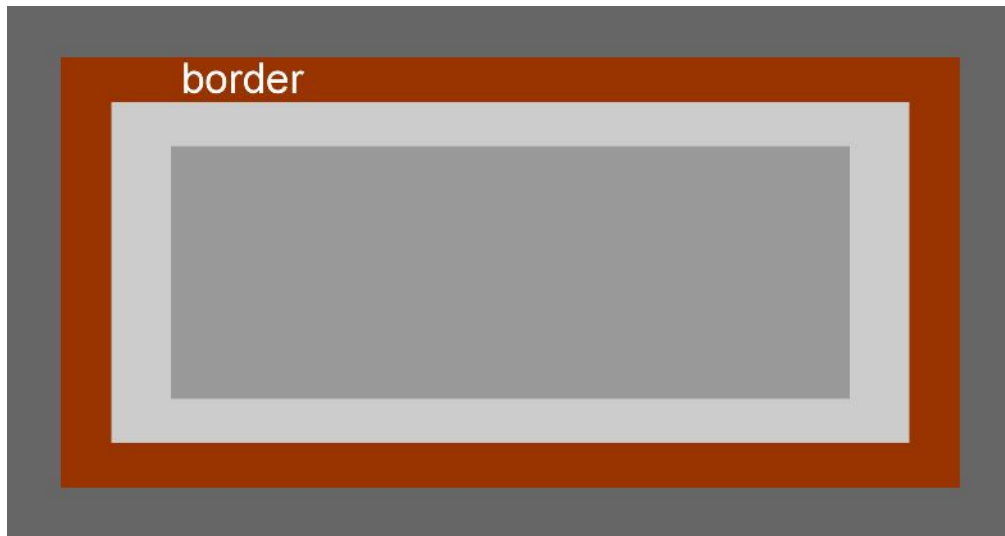
Espaciado o margen INTERIOR
(transparente dentro de un elemento)



Box Model - BORDER

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

El borde es la línea que bordea
la caja



Box Model - MARGIN

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

Margen o espaciado EXTERIOR
(transparente fuera de un elemento)

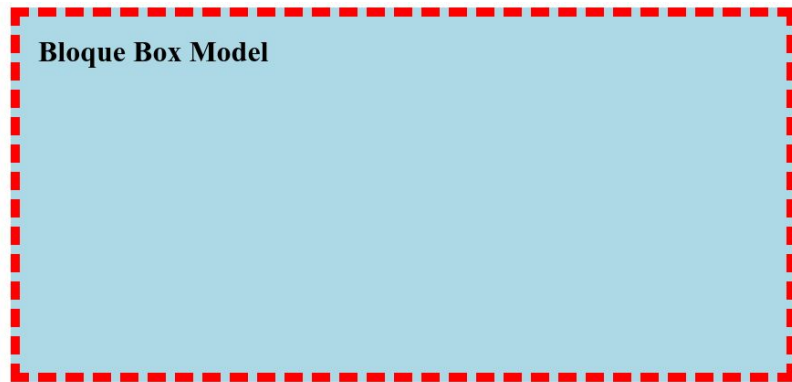


PUEDE USARSE PARA SEPARAR BLOQUES

Box Model

`<h1>Bloque Box Model</h1>`

```
h1 {  
  width: 800px;  
  height: 350px;  
  margin: 50px;  
  padding: 20px;  
  border: 10px dashed red;  
  background-color: lightblue;  
}
```



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

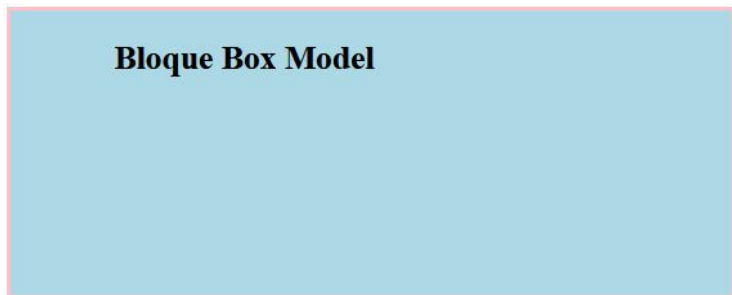
Box Model

Y si usamos tamaños de propiedades irregulares?

`<h1>Bloque Box Model</h1>`

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

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



```
h1 {  
  width: 600px;  
  height: 250px;  
  background-color: lightblue;  
  padding-top: 5px;  
  padding-bottom: 20px;  
  padding-left: 100px;  
  padding-right: 0;  
  border: 4px;  
  border-style: solid;  
  border-color: pink;  
  margin-left: 50px;  
  margin-right: 15px;  
  margin-top: 5px;  
}
```



Live: <https://codepen.io/webUnicen/pen/qoRRqY>

¿Cómo calcula el browser el tamaño de la “caja”?



Box Model - Calculando el Tamaño



```
h1 {  
  width: 600px;  
  height: 250px;  
  background-color: lightblue;  
  padding: 20px;  
  border: 4px;  
  border-style: solid;  
  border-color: pink;  
  margin: 50px;  
}
```

Live:

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

Ancho:

$\text{width} + \text{padding-left} + \text{padding-right} + \text{border-left} + \text{border-right} + \text{margin-left} + \text{margin-right}$
 $600\text{px} + 20\text{px} + 20\text{px} + 4\text{px} + 4\text{px} + 50\text{px} + 50\text{px} = 748$

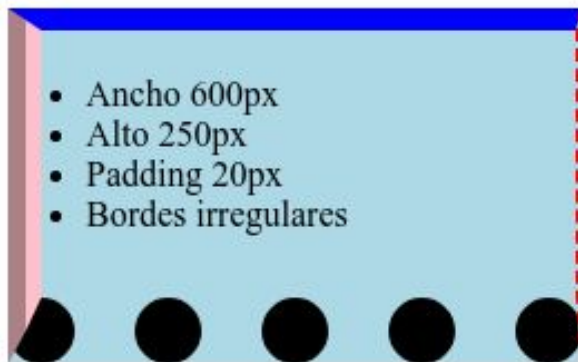
Height:

$\text{height} + \text{padding-top} + \text{padding-bottom} + \text{border-top} + \text{border-bottom} + \text{margin-top} + \text{margin-bottom}$
 $250\text{px} + 20\text{px} + 20\text{px} + 4\text{px} + 4\text{px} + 50\text{px} + 50\text{px} = 398$



También se pueden definir **bordes irregulares**

```
<ul>
  <li>Ancho 600px</li>
  <li>Alto 250px</li>
  <li>Padding 20px</li>
  <li>Bordes irregulares</li>
</ul>
```



```
ul {
  width: 200px;
  height: 250px;
  background-color: lightblue;
  padding: 20px;
  border-top: 10px solid blue;
  border-right: 3px dashed red;
  border-bottom: 30px dotted black;
  border-left: 15px groove pink;
}
```

Tipos de bordes	
none	none
inset	outset
dashed	ridge
double	solid
groove	inherit
hidden	
dotted	



Contenedores

<div>

- Es un elemento que define un bloque
- Puede incluir uno o varios elementos dentro
- **Nos ayuda a construir el layout y el diseño**

CodePen.io: Probémoslo en vivo

Cada vez que veas este ícono o un link a **codepen.io** tenés el código que hacemos en clase para probarlo y modificarlo vos.

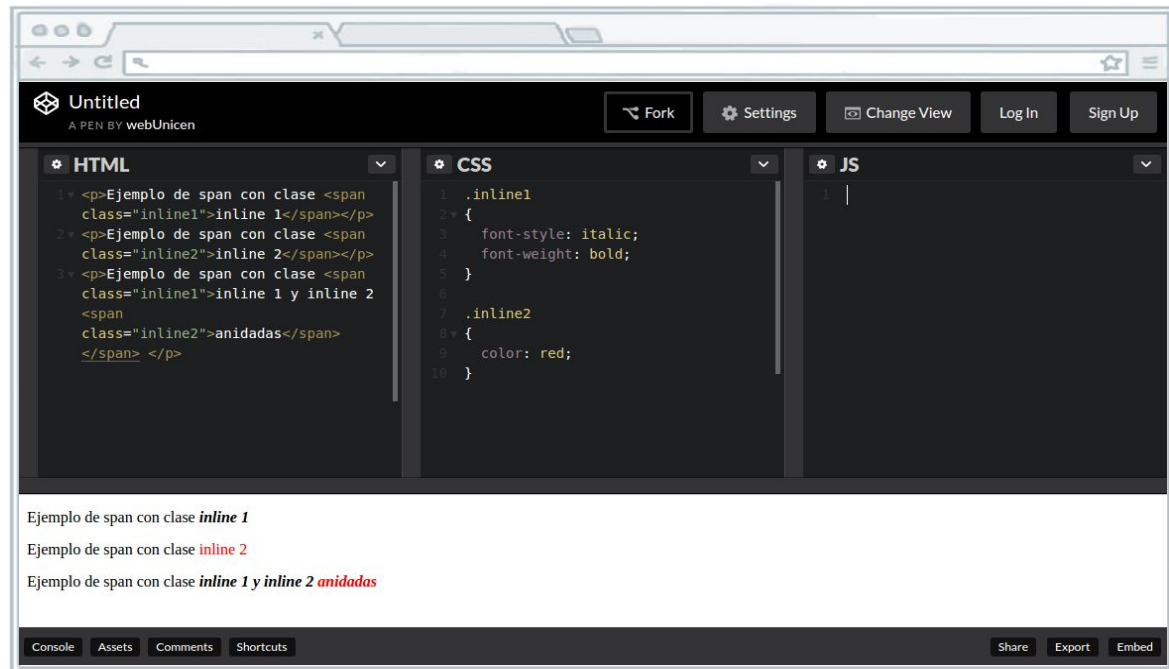


Trabajar en clase con el código básico. Agregar y probar variantes del ejemplo, cambiar valores numéricos para ver qué efecto tienen

CodePen.io: Probémoslo en vivo



CodePen es una comunidad en línea para **probar y mostrar** fragmentos de código HTML, CSS y JavaScript.

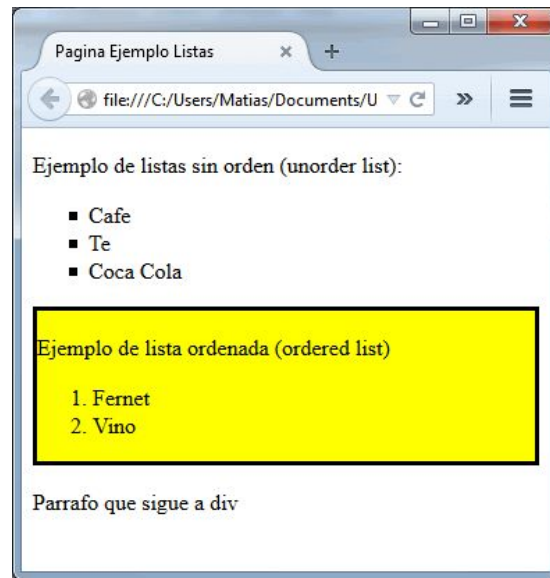


Bloque <div> ... </div>

- 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

```
<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 al div</p>
```



- Es un elemento “inline”
- Usado para agrupar texto, palabras o frases. Por ejemplo dentro de un párrafo
- Solo puede anidar elementos “prashing content” (,
, ...)

Bloque ` ... `

- Están **dentro del texto**, no en una línea nueva
- Su ancho depende del contenido que tengan

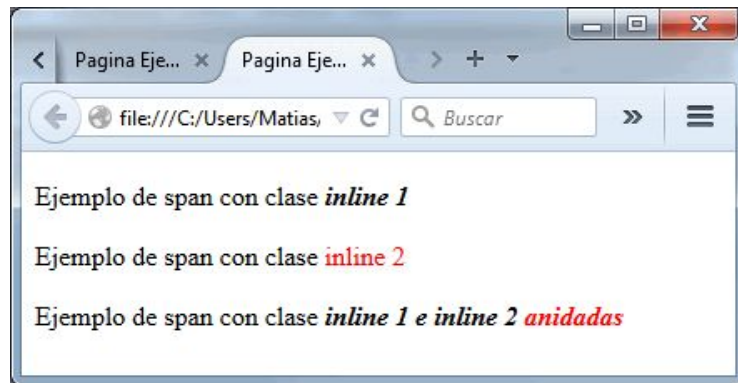
`<p>Ejemplo de span con clase inline 1</p>`

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

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

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

```
.inline2
{
  color: red;
}
```



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

Flujo de renderizado

Pregunta

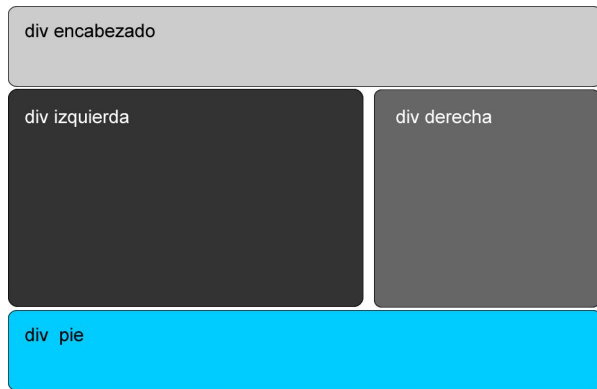
¿Pero cómo hacemos si queremos posicionar elementos de una manera más avanzada?



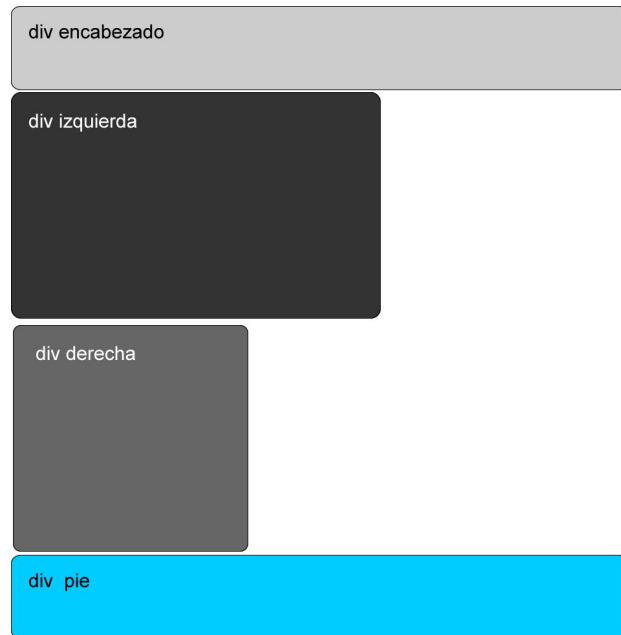
Layout - Un bosquejo

¿Cómo queremos que se vea?

EXPECTATIVA



REALIDAD

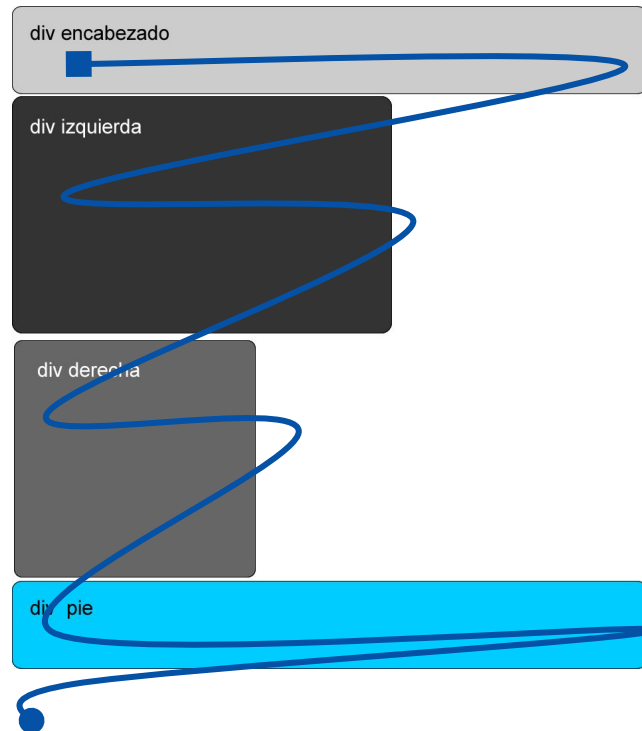


Posicionamiento

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 “salto de línea”



Block vs Inline

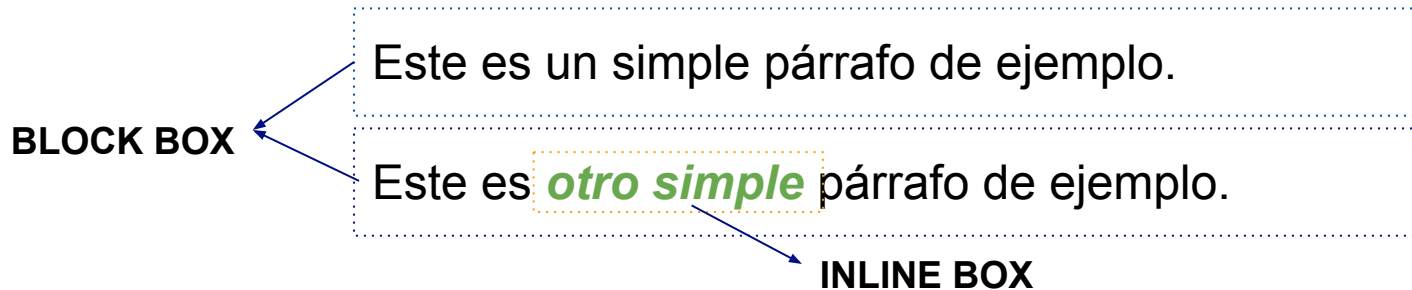
CSS puede definir la manera en la que los elementos de una página “**encajan**” uno con otros.

Según su propiedad “display”:

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

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

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



Tipos de Cajas - Block vs Inline

BLOCK

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

element-1 {display: block}

element-2 {display: block}

element-3 {display: block}

element-4 {display: block}

INLINE

`<a>, , , ...`

element-1
{ display:
inline}

element-2
{ display:
inline}

element-3
{ display:
inline}

Tipos de Cajas - Block vs Inline

HTML tiene elementos que por defecto son tipo bloque (**block**) o tipo en línea (**inline**)

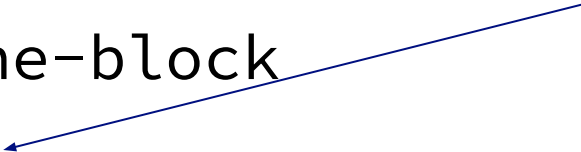
BLOCK	INLINE
<code><div> <p> <h1> - <h6> <table> <form> <article> <nav></code>	<code><a> <input> <button> <abbr></code>
Ubicación: uno abajo de otro	Ubicación: uno al lado de otro
El tamaño por defecto es el 100%	En su mayoría ignoran las propiedades width / height

Controlar el flujo de renderizado

Podemos controlar el **flujo de renderizado** con diferentes propiedades.

display:

- block
- inline
- inline-block
- **flex**
- grid



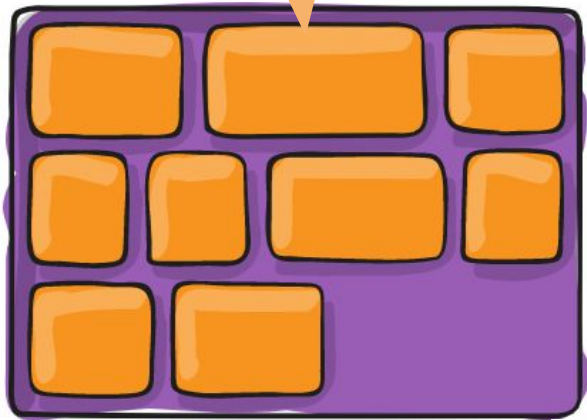
Uno de los mecanismos más eficientes para construir layouts.

Flexbox

Flex

Le da al **contenedor** la capacidad de alterar las dimensiones y orden de sus **items** para manejar mejor el espacio disponible.

items



otorga más control de
cómo se distribuyen
los hijos

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

container

Propiedades del eje principal

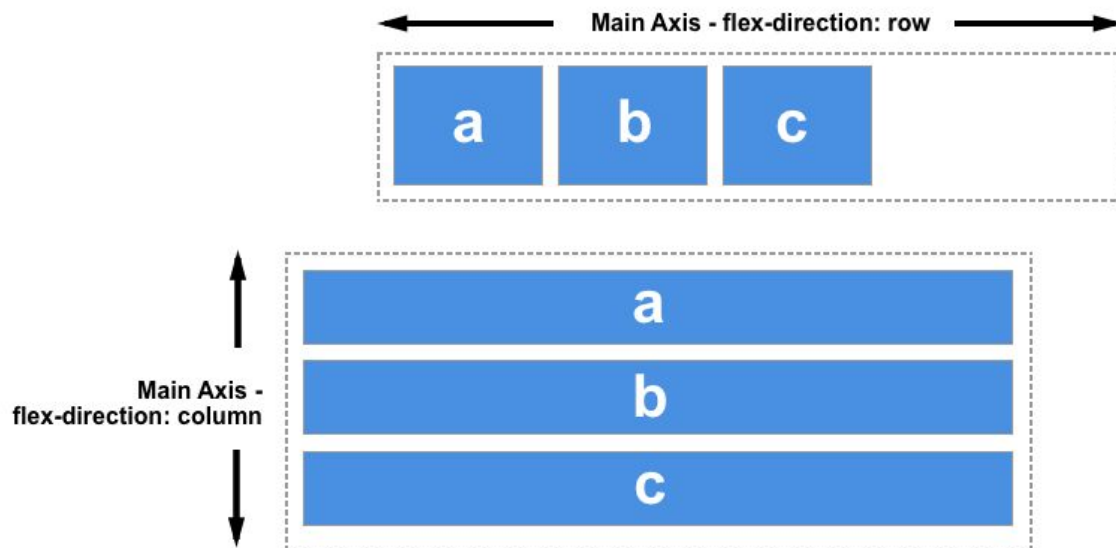
Flex diferencia dos ejes:

- **eje principal** definido por la propiedad *flex-direction*
- **eje transversal** es perpendicular al principal (el otro)

Todo lo que hacemos con flexbox está referido a estos dos ejes.

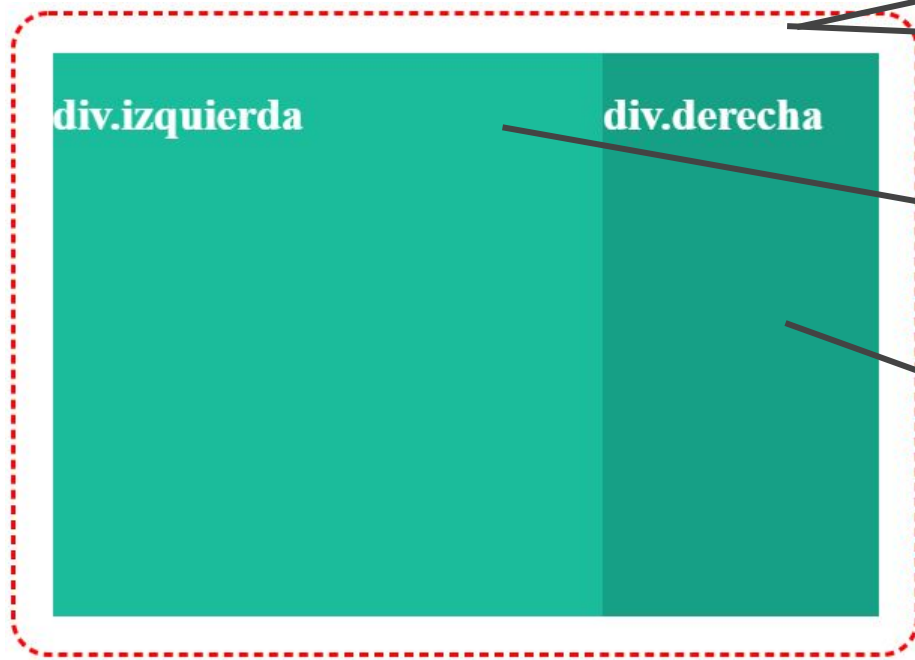
flex-direction:

- row
- column
- row-reverse
- column-reverse



Layouts usando flex

Creemos el layout de ejemplo usando **flexbox**.



es un div invisible

```
.container {  
  display: flex;  
}  
  
.izquierda {  
  width: 400px;  
  background-color: #333333;  
  color: white;  
}  
  
.derecha {  
  width: 200px;  
  background-color: #16a085;  
  color: white;  
}
```

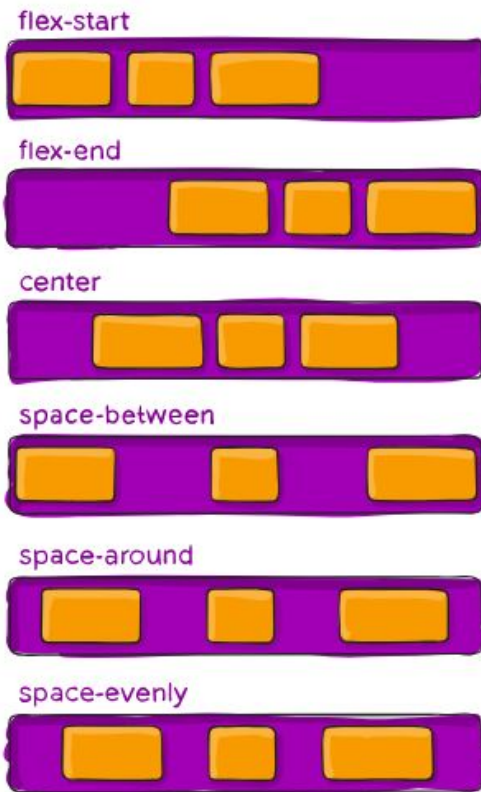


<https://codepen.io/webUnicen/pen/wmXqQg>

Flexbox - Alineación

La propiedad **justify-content** define la alineación de los componentes **a lo largo del eje principal**.

Ayuda a distribuir el espacio libre entre los items.



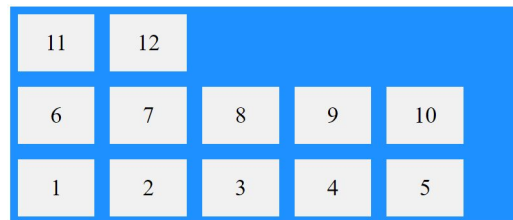
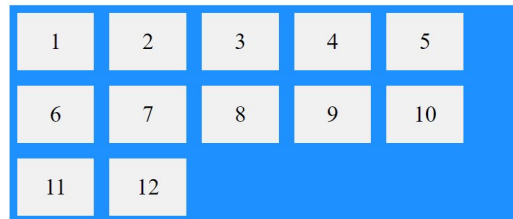
Flex

La propiedad **flex-wrap** especifica si los elementos flexibles deben ajustarse o no al contenedor.

- nowrap
- wrap
- wrap-reverse

Tanto flex-direction como flex-wrap se pueden concatenar en una sola propiedad llamada: flex-flow

`flex-flow: row wrap;`

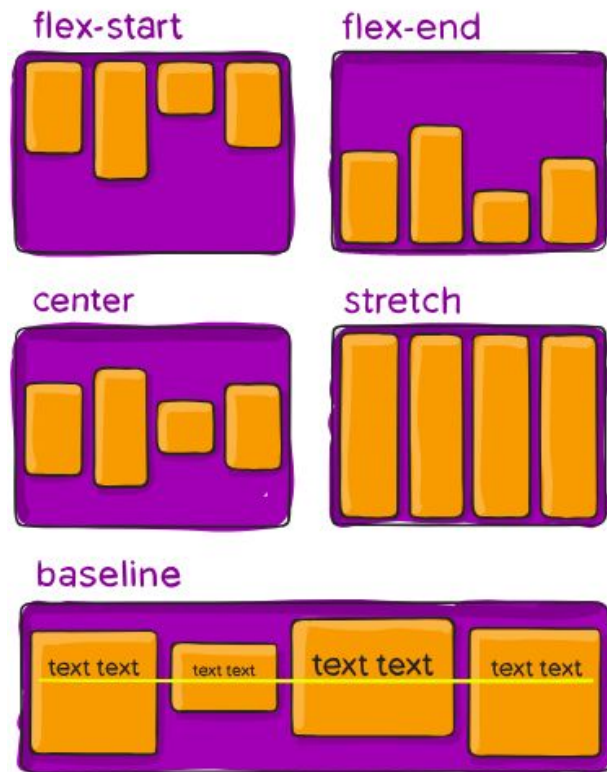


Default Value (por defecto) : nowrap

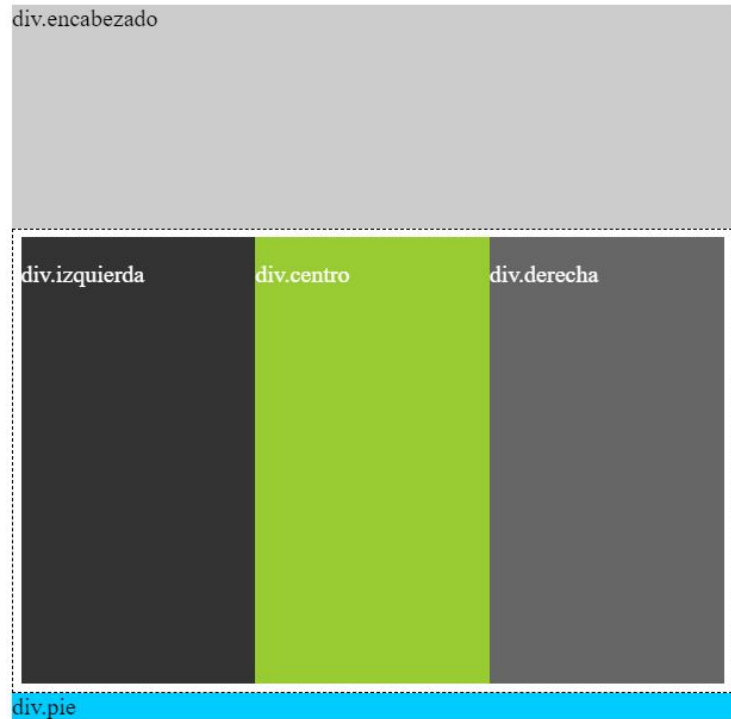
Flexbox - Alineación

La propiedad **align-items** define la alineación de los componentes **a lo largo del eje perpendicular**.

Es como la versión justify-content para el eje perpendicular.



Layouts usando flex



o de la ventana

```
.encabezado {  
  height: 150px;  
  background-color: #CCCCCC;  
}  
.contenedor {  
  display: flex;  
  justify-content: space-between;  
}  
.izquierda {  
  width: 200px;  
  background-color: #333333;  
  color: white;  
}  
.centro {  
  width: 200px;  
  background-color: #99cc33;  
  color: white;  
}  
.derecha {  
  width: 200px;  
  background-color: #666666;  
  color: white;  
}  
.pie {  
  background-color: #00CCFF;  
}
```



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

Layouts usando flex

Ejemplo con muchos “centro”:

Cambiar la propiedad **flex-wrap: wrap | nowrap**

Probar achicar la ventana y ver como se “bajan” las cosas

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

Default Value (por defecto) : nowrap

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>



Propiedades para cada elemento interior

`order`: Posición de cada uno.

`flex-grow`: cuánto crecerá en relación con el resto.

`flex-shrink`: cuánto se encogerá en relación con el resto.

`flex-basis`: especifica la longitud inicial de un elemento.

`align-self`: Similar a `align-item` pero para un elemento particular

- `stretch`
- `flex-start`
- `flex-end`
- `center`

FLEXBOX FROGGY



Bienvenido a Flexbox Froggy, un juego donde ayudarás a Froggy y a sus amigos escribiendo código CSS. Guía a esta rana hacia la hoja de lirio en la derecha, usando la propiedad

`justify-content`, la cual alinea elementos horizontalmente y acepta los siguientes valores:

- **`flex-start`**: Alinea elementos al lado izquierdo del contenedor.
- **`flex-end`**: Alinea elementos al lado derecho del contenedor.
- **`center`**: Alinea elementos en el centro.
- **`space-between`**: Muestra elementos con espacio entre ellos.
- **`space-around`**: Muestra elementos con espacio alrededor de ellos.

Por ejemplo, **`justify-content: flex-end`**.

```
1 #pond {  
2   display: flex;  
3  
4 }
```

¿Jugamos?

<https://flexboxfroggy.com/>

Siguiente

POR FAVOR PROFE



**LOS 5 MINUTOS
DE DESCANSO**

Unidades de Medida

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.

Pixeles [px] vs Porcentuales [%]

```
<div>
  <h1>div A en 40%</h1>
  <div class="chico">
    <h1>div B en 50%, hijo de A</h1>
  </div>
  <div class="fijo1">
    <h1>div C en 300px, hijo de A </h1>
  </div>
</div>
<div class="chico">
  <h1>div C en 50% </h1>
</div>
<div class="fijo2">
  <h1>div D en 500px </h1>
</div>
```

```
div {
  width: 40%;
  background-color: orange;
}
.chico {
  width: 50%;
  background-color: red;
}
.fijo1 {
  width: 300px;
  background-color: green;
}
.fijo2 {
  width: 500px;
  background-color: green;
}
```

div A en 40%

div B en 50%, hijo de A

div C en 300px, hijo de A

div C en 50%

div D en 500px



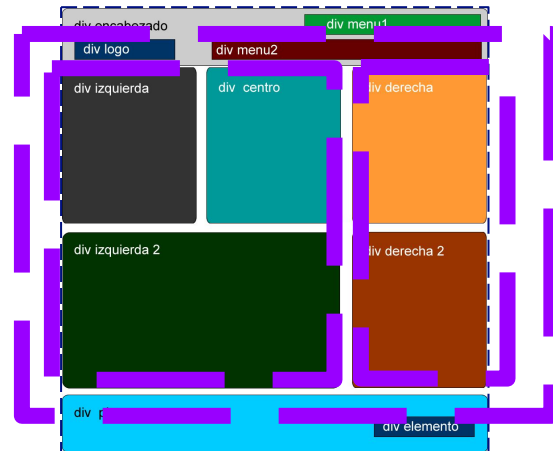
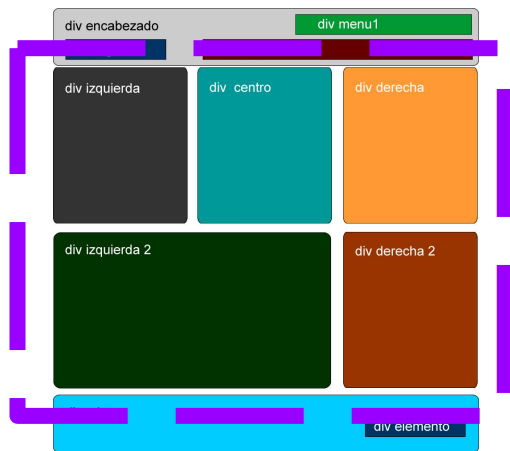
Live: <https://codepen.io/webUnicen/pen/xWXOOK>

Ejercicio

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



Ejercicio - Cuál opción es posible? Y la correcta?



Position

Posicionamiento

La propiedad **position** sirve para posicionar un elemento dentro de la página.

- Muy útil cuando queremos posicionar elementos fuera del flujo normal de la página
- Es fundamental interpretar el funcionamiento del posicionamiento para poder dar la ubicación exacta a cada elemento dentro del Box Model. Dependiendo de cual sea la propiedad que usemos, el elemento tomará una referencia u otra para posicionarse.

static | absolute | relative | fixed | sticky

Posicionando Elementos

La propiedad “**position**” que posee diferentes valores.

- **static**: valor por default. Mantiene el elemento en el flujo normal.
- **relative**: permite usar propiedades como top, right, bottom y left para mover el elemento en la página.
- **absolute**: funciona con las mismas propiedades, pero rompen el flujo normal. Se corresponden con la posición de un ancestro (*el primero que tiene position no static*).
- **fixed**: funciona con las mismas propiedades, pero rompe el flujo normal. Al punto que establece una posición fija en la pantalla.
- **sticky**: el elemento es posicionado en base al scroll del usuario.

Posicionando Elementos - Relative + Absolute

```
HTML
1 <div>
2   <ul>
3     <li>Elemento 1</li>
4     <li>Elemento 2</li>
5   </ul>
6 </div>

CSS
1 div {
2   position: relative;
3   border: 6px dashed #ccc;
4   height: 200px;
5   top: 50px;
6   width: 80%;
7 }
8 ul {
9   position: absolute;
10  top: 25px;
11  right: 40px;
12  border: 1px solid red;
13 }
```

El posicionamiento de **** es respecto de su padre **<div>**



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

Posicionando Elementos - Absolute

Comentamos “position: relative” del div.

The image shows two side-by-side code editors and their corresponding browser renderings. The left editor shows a CSS rule for a `div` with `position: relative;`. The right editor shows the same rule but with `position: relative;` commented out. Below each editor is a browser preview showing a dashed box representing the element's bounding box. In the left preview, the dashed box is positioned 50px from the top. In the right preview, the dashed box is positioned at the top of the container, indicating that the `top: 50px;` rule has no effect when the parent is not relative.

HTML	CSS
<pre>1 <div> 2 3 Elemento 1 4 Elemento 2 5 6 </div></pre>	<pre>1 div { 2 position: relative; 3 border: 6px dashed #ccc; 4 height: 200px; 5 top: 50px; 6 width: 80%; 7 } 8 ul { 9 position: absolute; 10 top: 25px; 11 right: 40px; 12 border: 1px solid red; 13 }</pre>
<pre>1 <div> 2 3 Elemento 1 4 Elemento 2 5 6 </div></pre>	<pre>1 div { 2 /*position: relative;*/ 3 border: 6px dashed #ccc; 4 height: 200px; 5 top: 50px; 6 width: 80%; 7 } 8 ul { 9 position: absolute; 10 top: 25px; 11 right: 40px; 12 border: 1px solid red; 13 }</pre>

• Elemento 1
• Elemento 2

• Elemento 1
• Elemento 2

No tiene efecto **top: 50px.**

El posicionamiento de **** ahora es desde el root element, no de **<div>**

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

Posicionando Elementos

sticky, ahora **** no está dentro de **<div>** (ya no son anidados), sino que siempre se va a ver en la misma posición de la pantalla



```
HTML
1+ <div>
2+   <ul>
3+     <li>Elemento 1</li>
4+     <li>Elemento 2</li>
5+   </ul>
6+ </div>

CSS
2+ /*position: relative;*/
3+ border: 6px dashed #ccc;
4+ height: 200px;
5+ top: 50px;
6+ width: 80%;
7+ }
8+
9+ ul {
10+   position: sticky;
11+   top: 25px;
12+   right: 40px;
13+   border: 1px solid red;
14+ }
```

- Elemento 1
- Elemento 2

Posicionando Elementos

absolute , ahora **** no está dentro de **<div>** (ya no son anidados)

```
HTML
1 <div>
2   <p>Esto es un div</p>
3 </div>
4
5 <ul>
6   <li>Elemento 1</li>
7   <li>Elemento 2</li>
8 </ul>

CSS
1 div {
2   position: relative;
3   border: 6px dashed #ccc;
4   height: 200px;
5   top: 50px;
6   text-align: left;
7   padding-left: 10px;
8 }
9
10 ul {
11   position: absolute;
12   top: 25px;
13   right: 40px;
14   border: 1px solid red;
15 }
```

<div> es **relative**, tiene efecto **top: 50px**.
**** también es respecto del root element (**<body>**)



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.

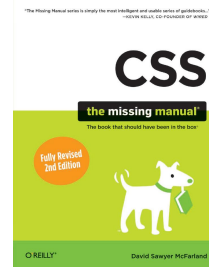


Referencias



HTML & CSS - Design and Build Websites.
JON DUCKETT

CSS - the missing manual.
DAVID SAWYER MCFARLAND



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

AHORA LES TOCA PRACTICAR :D



Extras



overflow: controla lo que sucede cuando el contenido excede a las dimensiones del bloque.

Las opciones son: **auto**, **hidden**, **scroll**, **visible**, **inherit**

A screenshot of a code editor with two panels. The left panel is titled 'HTML' and the right panel is titled 'CSS'. Both panels have a 'Tidy' button and a close 'x' button. The HTML panel contains code for two paragraphs, one with class 'scroll' and one with class 'hidden'. The CSS panel contains styles for these two classes, setting background color, width, height, and overflow property.

```
HTML
1 dimensiones del bloque.
2 .</p>
3
4 <p>overflow:scroll</p>
5 <div class="scroll"> Podes usar la propiedad
  overflow para tener un mejor control del
  layout. El valor por default es visble.
  </div>
6
7 <p>overflow:hidden</p>
8 <div class="hidden"> Podes usar la propiedad
  overflow para tener un mejor control del
  layout. El valor por default es visble.
  </div>

CSS
1 div.scroll {
2   background-color: #00FFFF;
3   width: 100px;
4   height: 100px;
5   overflow: scroll;
6 }
7
8 div.hidden {
9   background-color: #00FF00;
10  width: 100px;
11  height: 100px;
12  overflow: hidden;
13 }
```



1. Hacer un bloque.
2. Agregarle contenido que desborde las dimensiones, por ejemplo un párrafo.
3. Aplicar la propiedad overflow con las distintas variantes para ver cómo funciona cada una.

Encimando elementos



z-index : Cuando se superponen dos o más elementos se puede decidir cual queda por encima o por debajo. Sirve para establecer el orden de los fondos con fotos, transparencias, textos, etc. Se pueden entender como capas



Encimando elementos



Las opciones son **auto**, **number**, **inherit**

Ejemplo:

```
.box1 {  
  z-index: -1;  
}
```

```
.box2 {  
  z-index: 1;  
}
```

```
.box3 {  
  z-index: 2;  
}
```

en este caso box3 tiene prioridad al frente, luego box2 y por último box1 al fondo.

<http://codepen.io/webUnicen/pen/ZeZdBO>

Herramientas

Box Model - Herramientas Chrome



The screenshot shows the CSS Zen Garden website in a Chrome browser. The page features a large header with the site's logo and a 'VIEW ALL DESIGNS' button. Below the header, there's a section titled 'THE ROAD TO ENLIGHTENMENT' and another titled 'MID CENTURY MODERN'. The Chrome DevTools developer tools are open at the bottom. The 'Elements' panel on the left shows the HTML structure, with the 'View All Designs' link selected. The 'Styles' panel on the right shows the CSS rules applied to this link, including a background color and a background image. An arrow labeled 'CSS' points from the 'Elements' panel to the 'Styles' panel.

HTML

CSS

Marcado con la herramienta de desarrollo de Chrome.
Menú > Más Herramientas > Herramientas de desarrollador
(Ctrl + Mayusc. + I)

Box Model - Herramientas, Firebug



The screenshot shows the CSS Zen Garden website in a browser. The main banner features a circular logo and the text "CSS ZEN GARDEN The Beauty of CSS Design". A button labeled "VIEW ALL DESIGNS" is visible. Below the banner, a section titled "MID CENTURY MODERN by Andrew Lohman" is partially visible. A text box on the page reads: "A demonstration of what can be accomplished through CSS-based design. Select any style sheet from the list to load it into this page." Below this, there are links to "Download the example" for "HTML FILE" and "CSS FILE".

Inspect

Console HTML CSS Script DOM

```
<div class="page-wrapper">
  <section class="intro" id="zen-intro">
    <header role="banner">
      <div class="summary" id="zen-summary" role="article">
        <p>
          Download the example
          <a href="/examples/index" title="This page's source HTML code, not to be modified.">html file</a>
          and
```

Style Computed DOM

```
p {
  line-height: 2;
  margin: 0.75em 0;
}
* {
  box-sizing: border-box;
}
Inherited from body#css-zen-garden
```