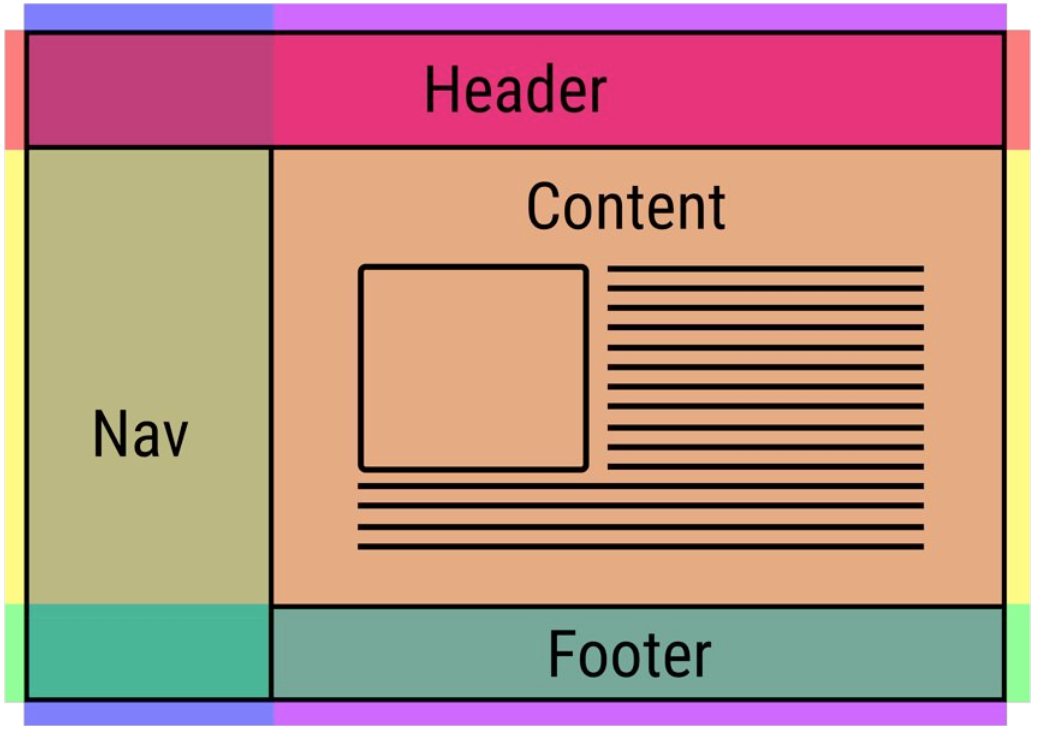
Layout

Layouts

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



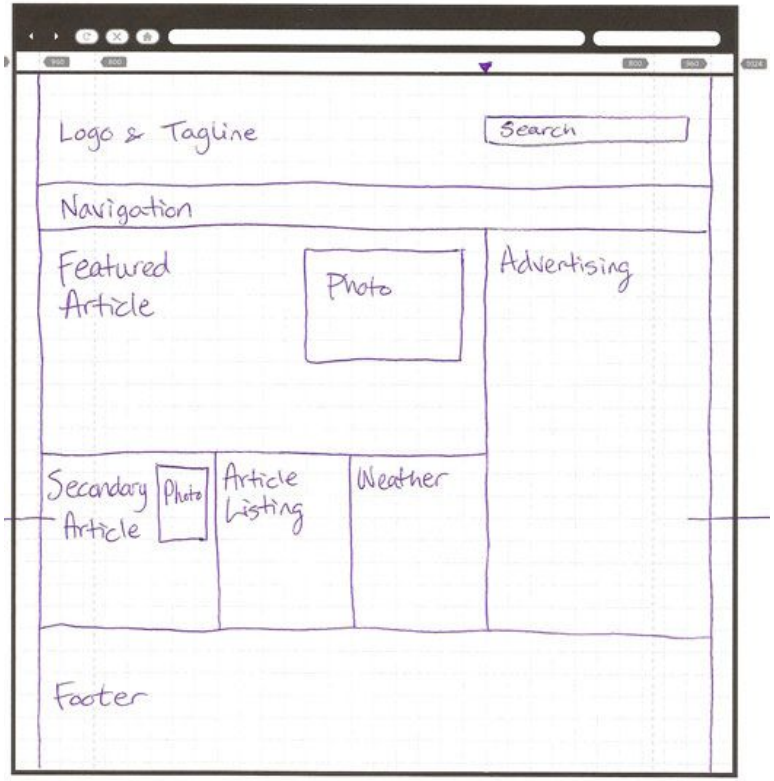
"Es el esqueleto general de la página"

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.



Box Model

Box Model - Introducción

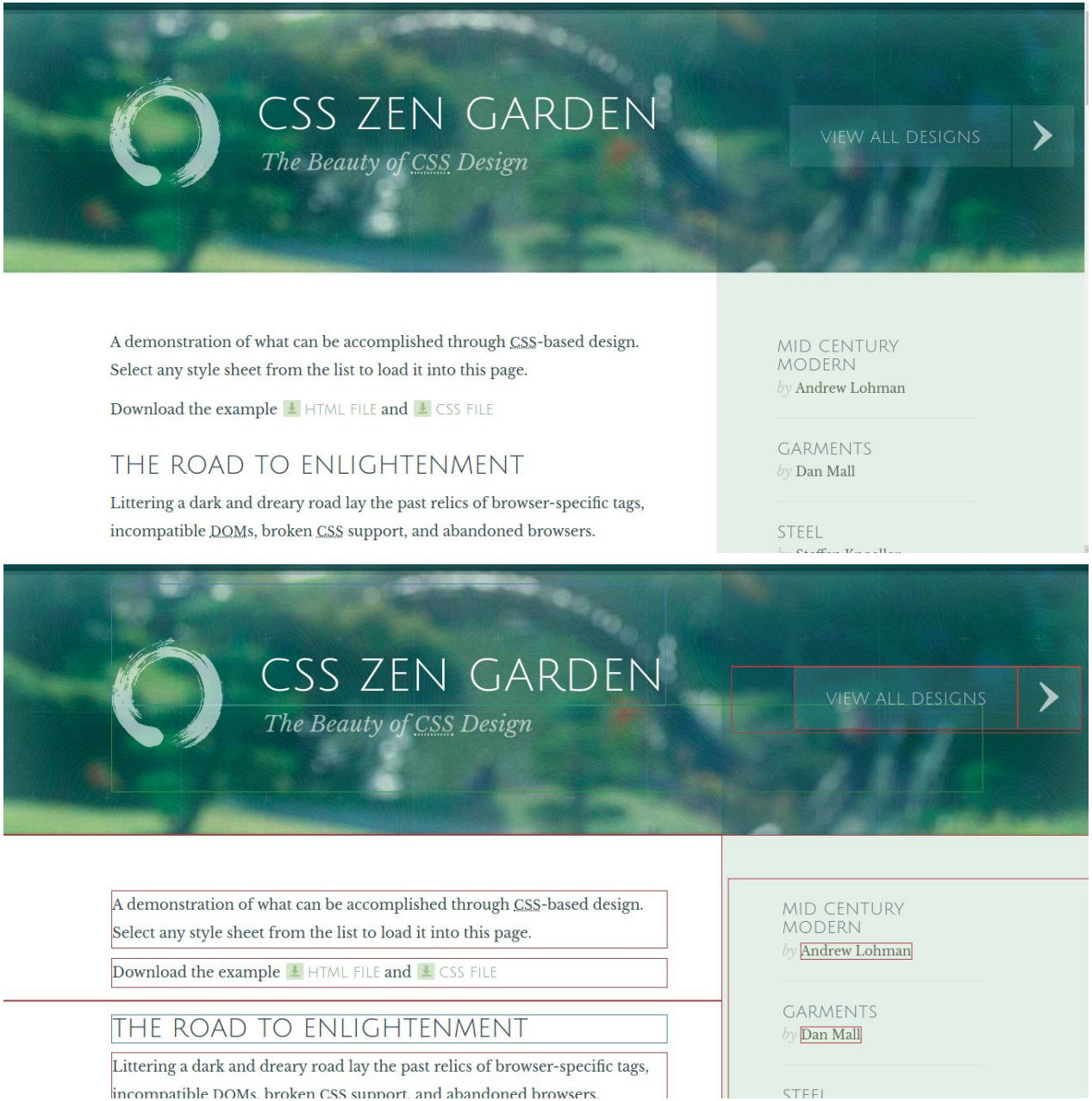
El concepto de **“Box Model”** dice que cada elemento en una página se representa mediante una caja rectangular (contenedor).

- CSS permite controlar el aspecto y ubicación de las cajas
- Todos los elementos HTML están representados como cajas
- **Box Model siempre es utilizado**
- Concepto fundamental para construir y diagramar sitios
- Fondo y borde pueden ser transparentes

Box Model - Developer Tools (add on)

Marcado de elemento de bloque

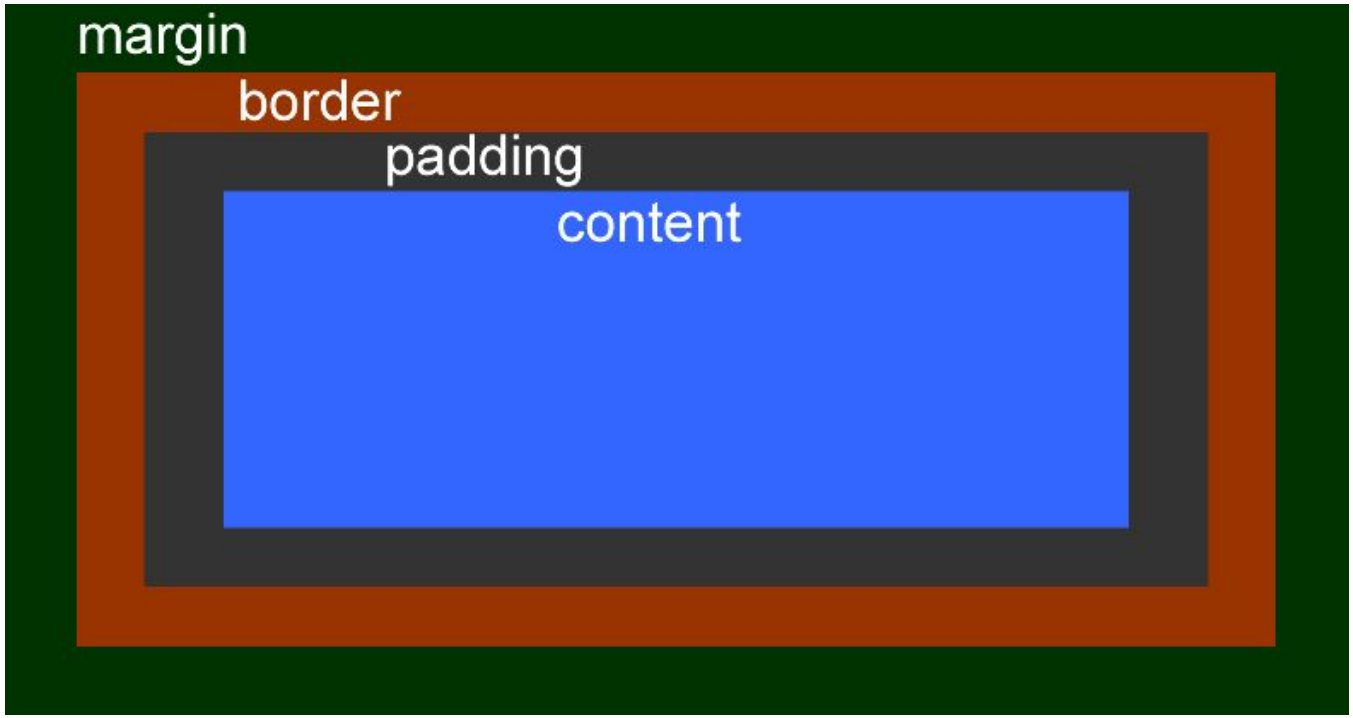
- Chrome con el Add-On Web Developer (Link)
- Firefox para desarrollo (Outline > Outline Block Elements)



Box Model

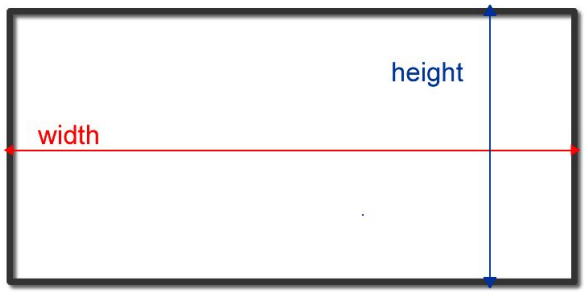
CSS utiliza el modelo de cajas / bloques que consta de 4 partes:

- CONTENT
- PADDING
- BORDER
- MARGIN

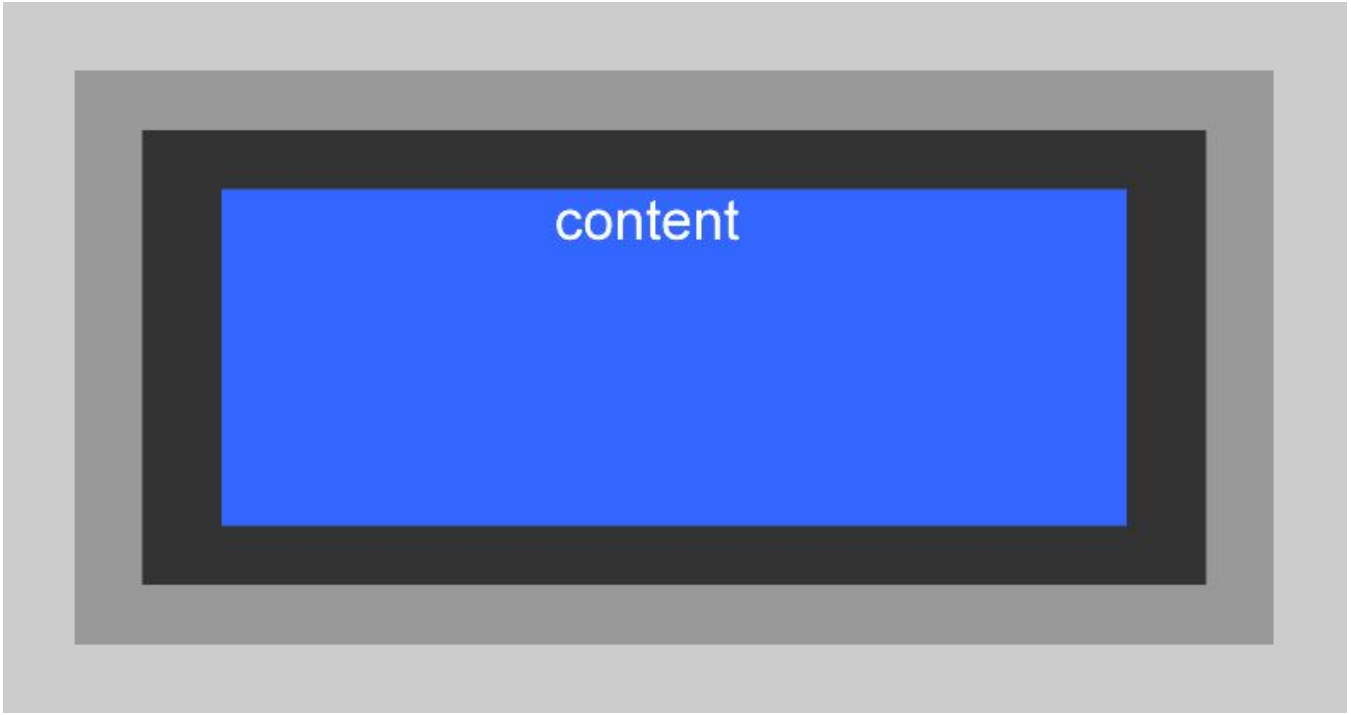


Box Model - CONTENT

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



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



Box Model - PADDING

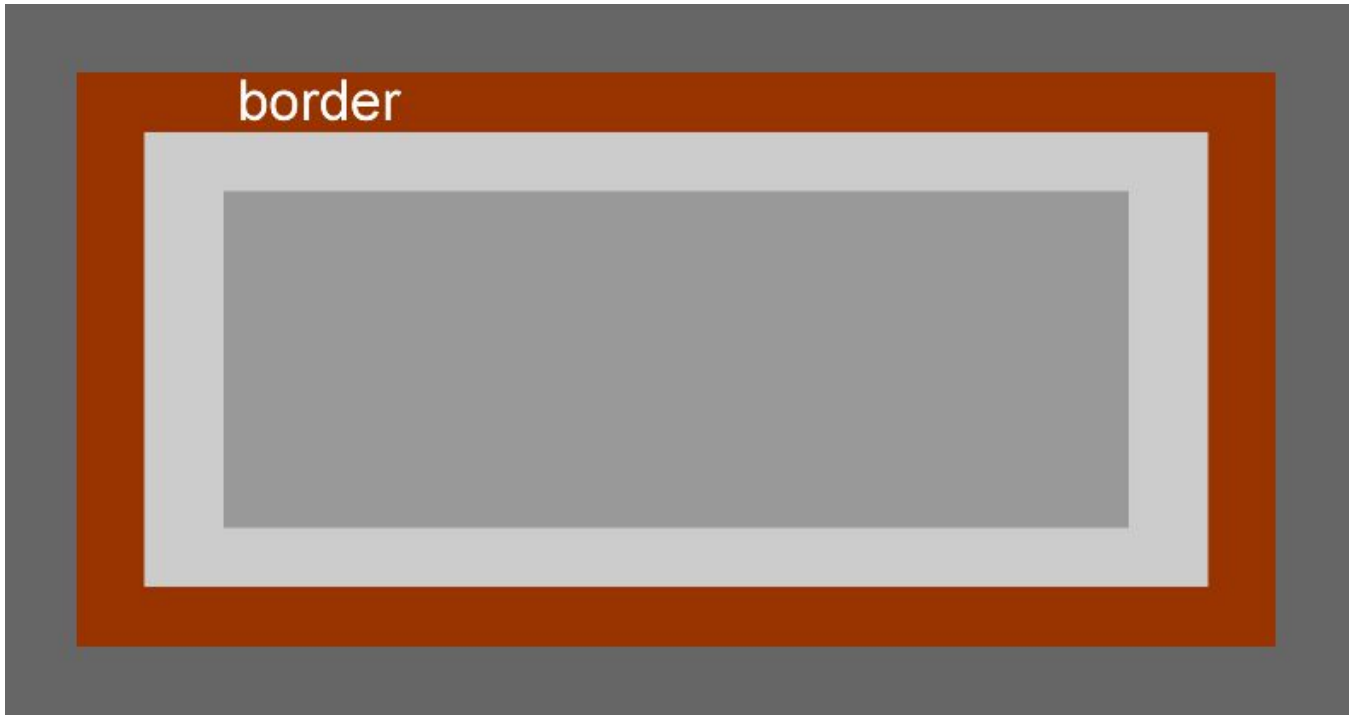
- CONTENT
- **PADDING**
- BORDER
- MARGIN

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

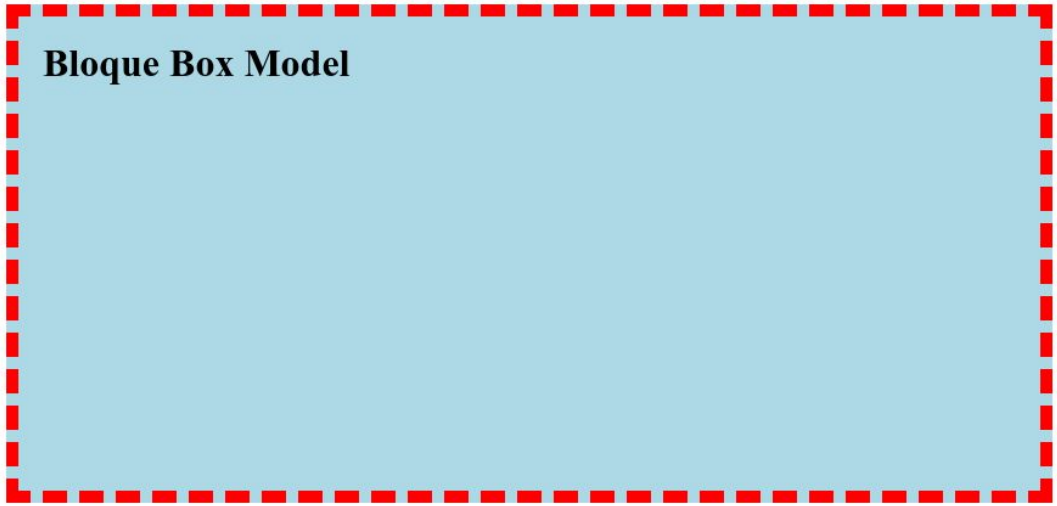
- CONTENT
- PADDING
- BORDER
- **MARGIN**

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

PUEDE USARSE PARA SEPARAR BLOQUES

Box Model

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




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

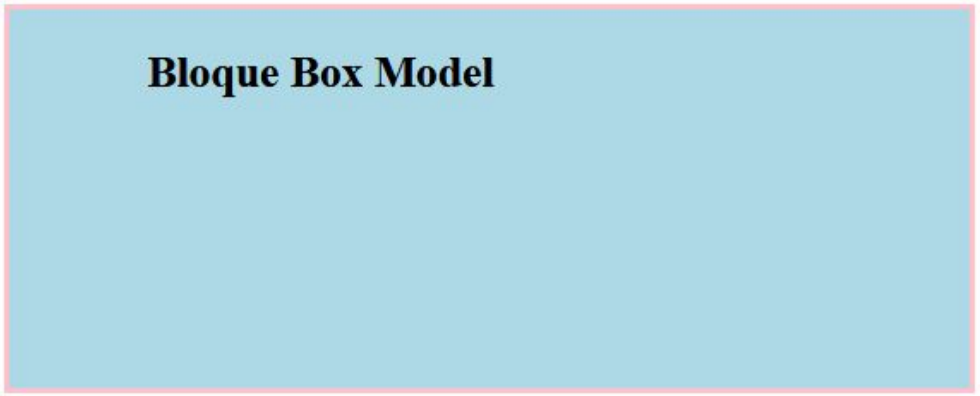
DEMO Live: http://codepen.io/webUnicen/pen/yMRawg

Box Model

Y si usamos tamaños de propiedades irregulares?

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

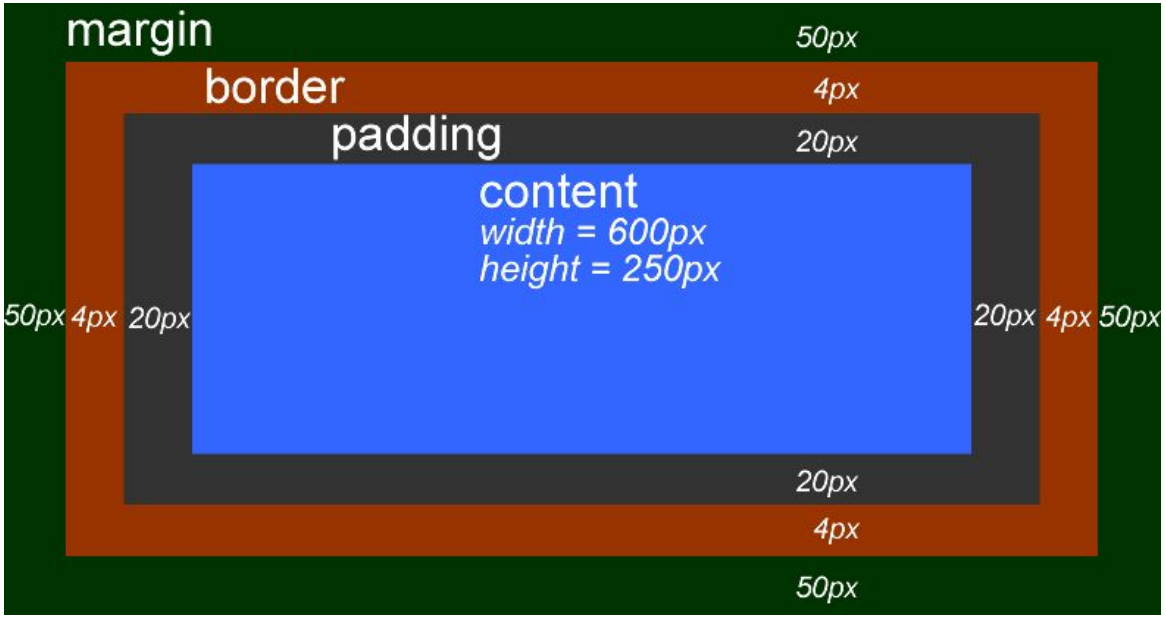
margin-top	padding-top
margin-bottom	padding-bottom
margin-left	padding-left
margin-right	padding-right




```
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;
}
```

DEMO Live: https://codepen.io/webUnicen/pen/qoRRqY

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

Para calcular el alto total y ancho total del element hacemos:

Ancho:

width + padding-left + padding-right + border-left + border-right + margin-left + margin-right

600px + 20px + 20px + 4px + 4px + 50px + 50px = 748

Height:

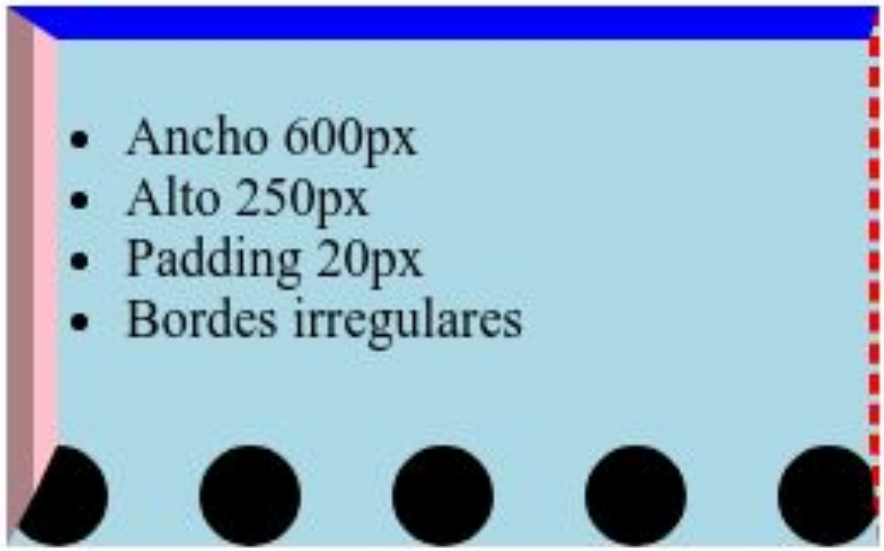
height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom

250px + 20px + 20px + 4px + 4px + 50px + 50px = 398

Box Model - Border

También se pueden definir **bordes irregulares**

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




```
ul {
   width: 200px;
   height: 80px;
   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

inset	none
dashed	outset
double	ridge
groove	solid
hidden	inherit
dotted	

http://codepen.io/webUnicen/pen/aJRmqL

https://developer.mozilla.org/en-US/docs/Web/CSS/border-style#Values

Height & Width: Datos extras

- Todos los elementos tienen un alto y ancho heredado.
- Si algún elemento es clave para el layout o diseño de la página seguramente tenga un ancho y alto específico.
- El ancho default de un elemento depende de su tipo.
  - Los elementos de bloque tienen un ancho default de 100%.
  - Los elementos inline, ocupan solo el tamaño de lo que contienen.
  - Como los elementos inline no pueden tener un tamaño fijo las propiedades width y height se reservan para los elementos de bloque.


BACK IN
5 MINS

Contenedores

&

- Son simples contenedores de HTML
- Son cajas sin ningún significado semántico
- Como cualquier elemento podemos usar class o id
- Elegir un nombre de clase representativo

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

CodePen.io: Probémoslo en vivo

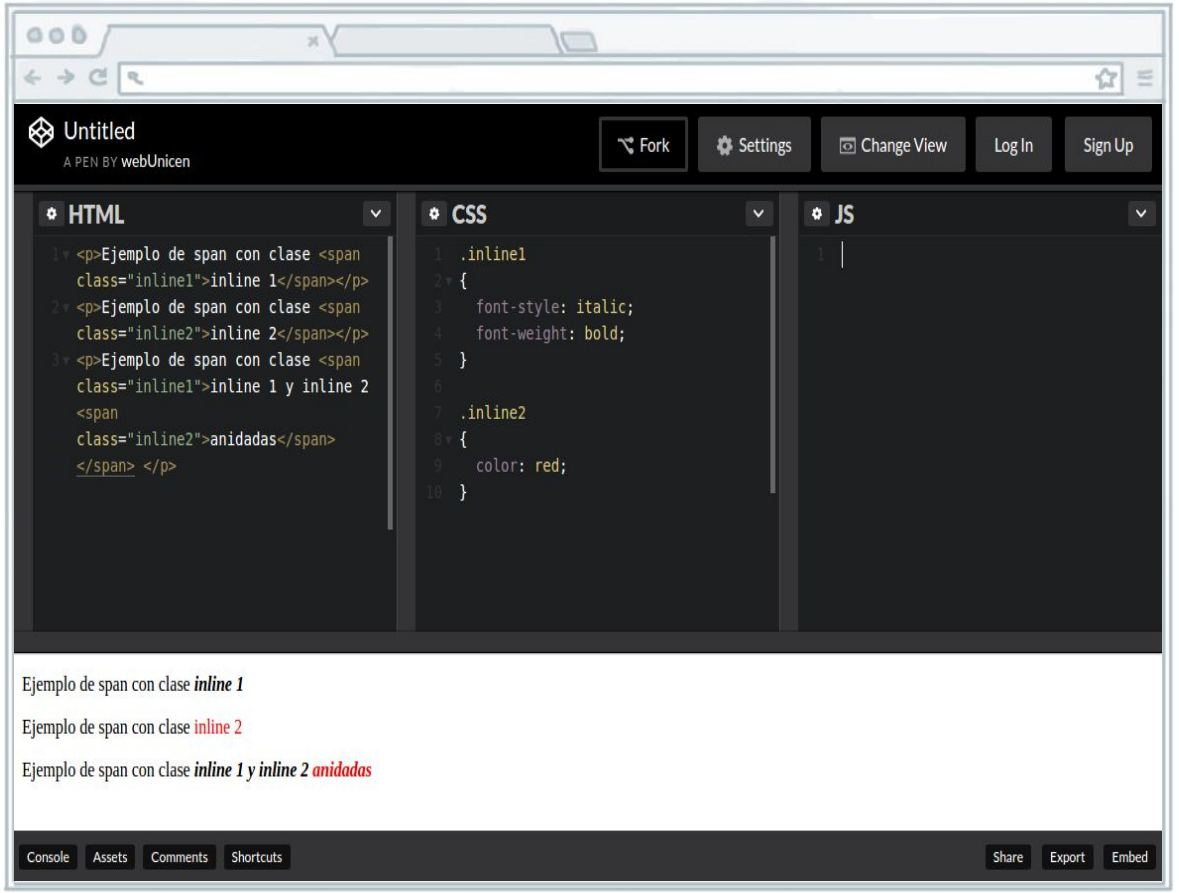
Cada vez que veas este ícono o un link a **codepen.io** tenés el código que hacemos en clase para probarlo y modificarlo vos.

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

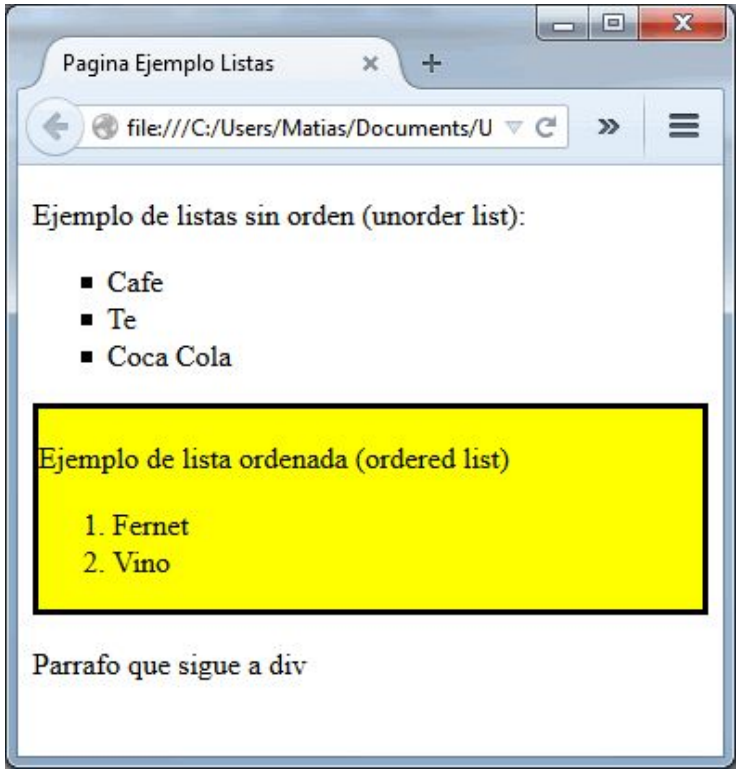
CodePen es una comunidad en línea para probar y mostrar fragmentos de código HTML, CSS y JavaScript.



Bloque ...

- 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>
```



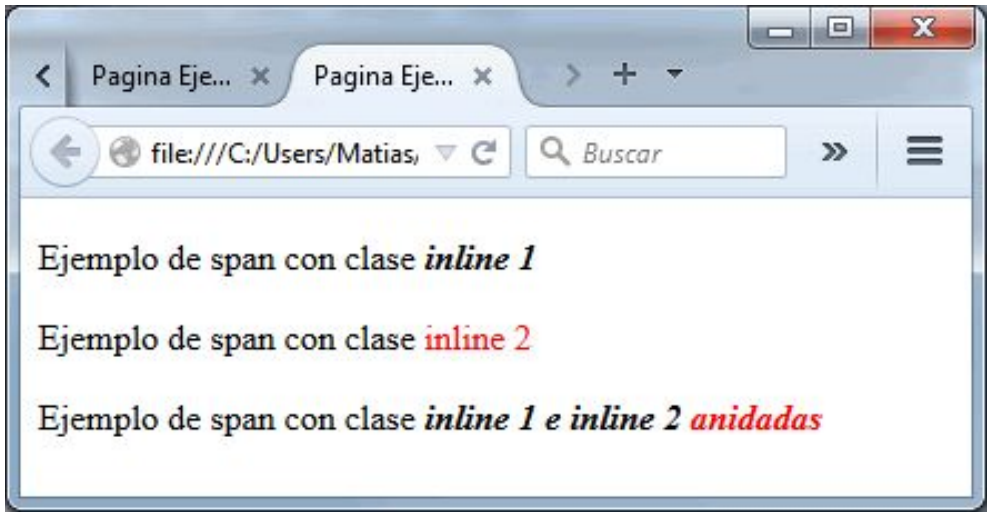
DEMO http://codepen.io/webUnicen/pen/mWxpYM

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 <span class="inline1">inline 1</span></p>
<p>Ejemplo de span con clase <span class="inline2">inline 2</span></p>
<p>Ejemplo de span con clase <span class="inline1">inline 1 y inline 2 <span
class="inline2">anidadas</span> </span> </p>
```

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

.inline2
{
color: red;
}
```



DEMO

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

Flujo de renderizado

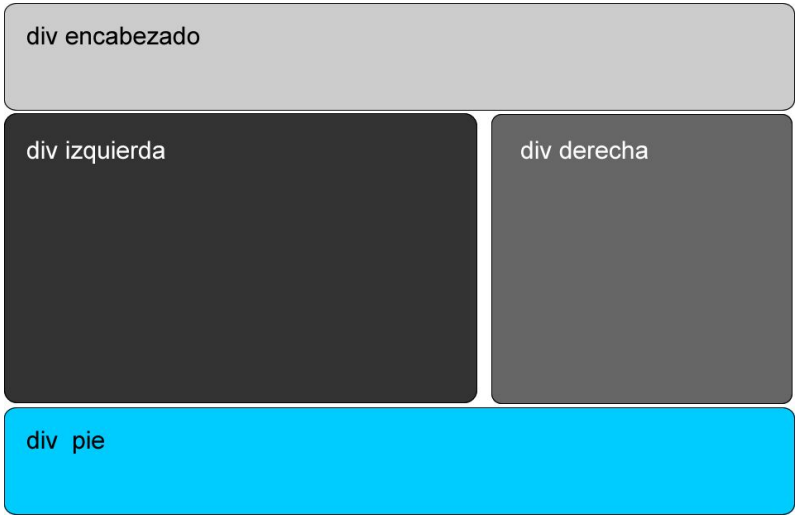
Pregunta

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

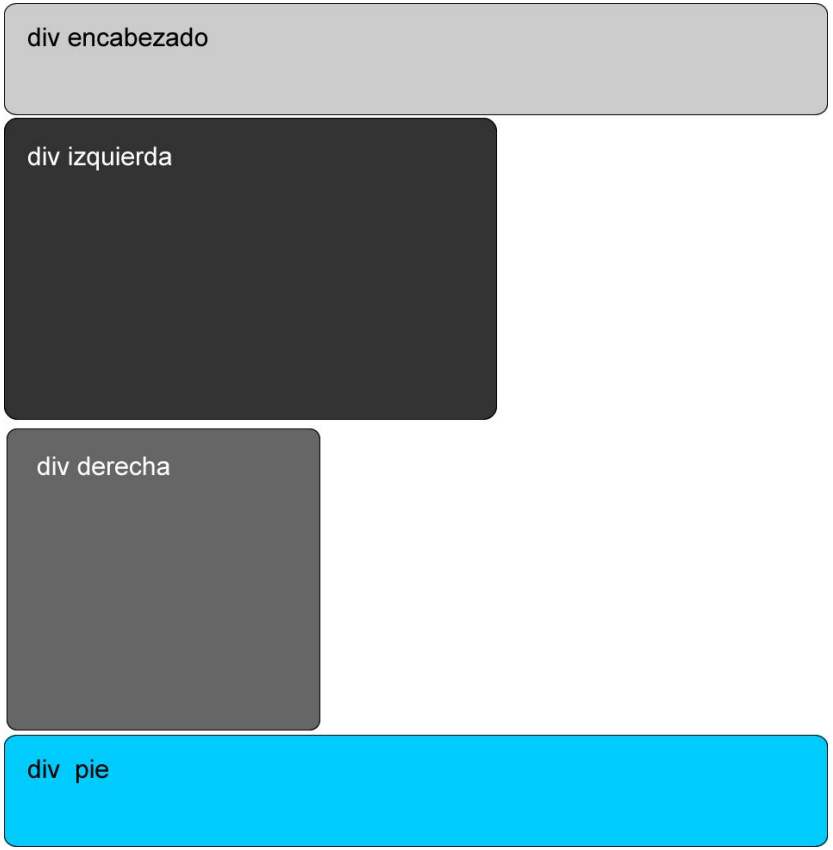
Layout - Un bosquejo

¿Cómo queremos que se vea?

EXPECTATIVA



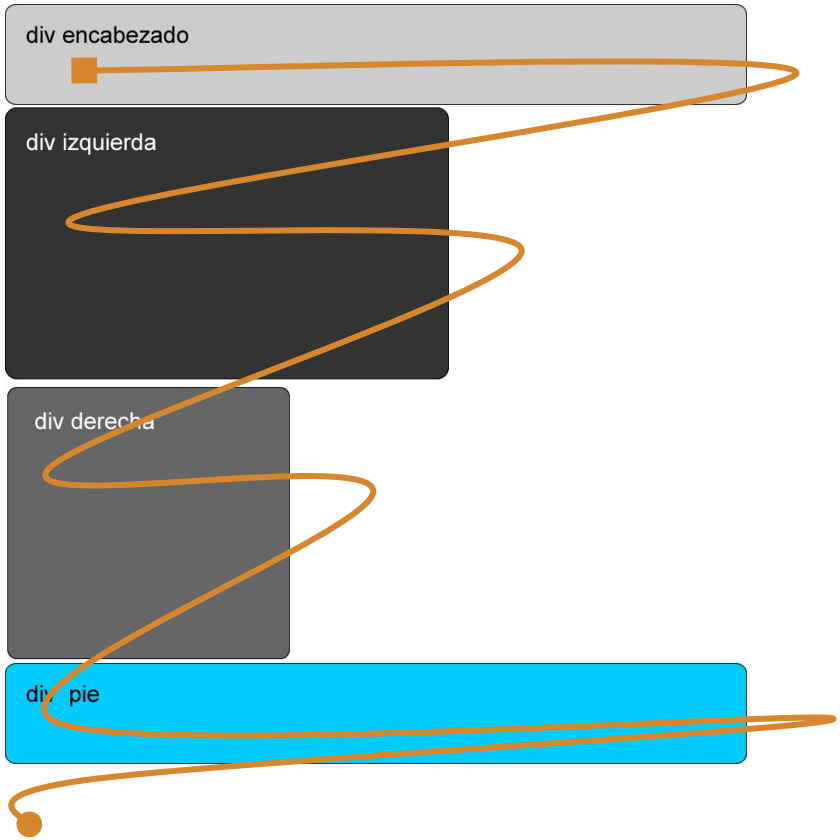
REALIDAD



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 linea”



Block vs Inline

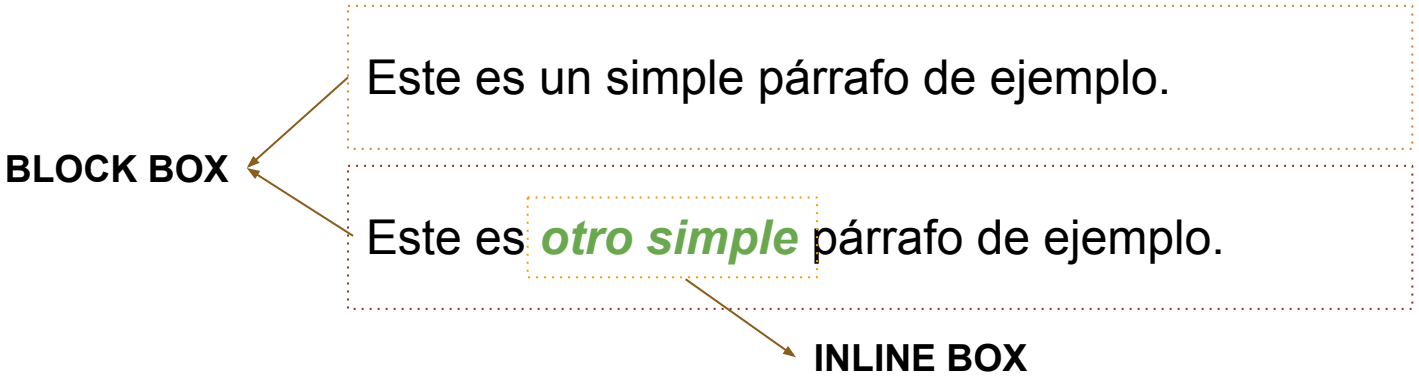
CSS puede definir la manera en la que los elementos de una página **“encajan”** uno con otros.

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>, <span>, <strong>, <img> ...`

element-1 { display: inline}

element-2 { display: inline}

element-3 { display: inline}

Controlar el flujo de renderizado

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

`display:`

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

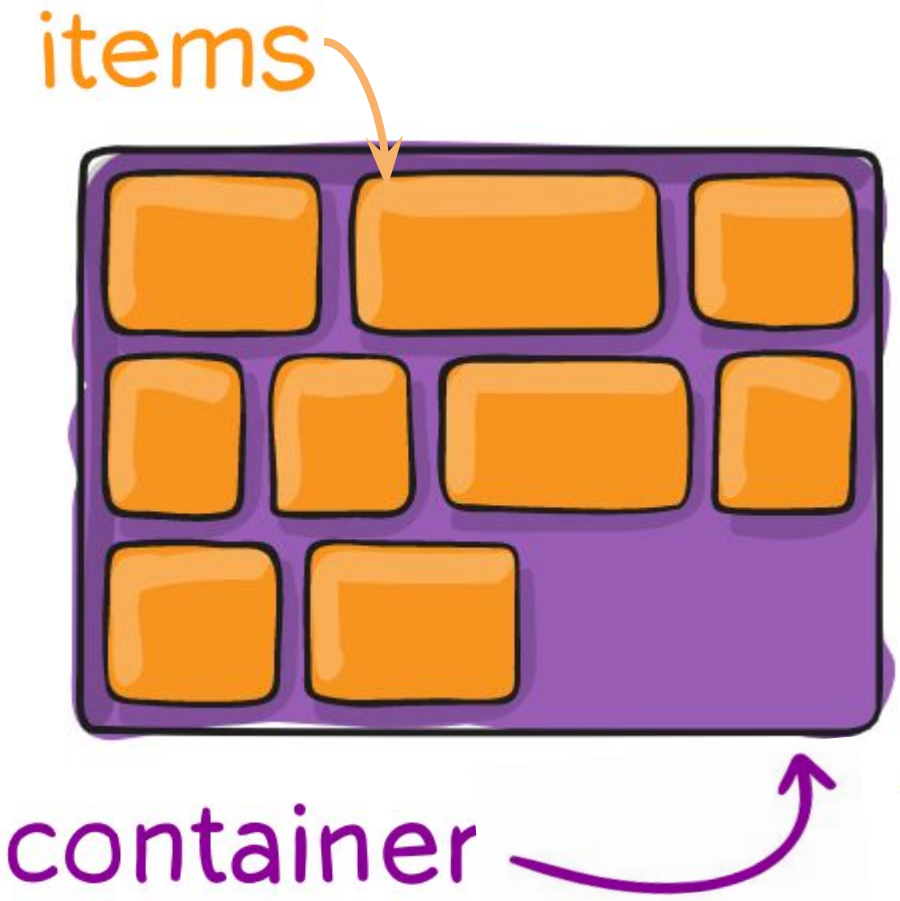
Uno de los mecanismos más eficientes para construir layouts.

`float/clear`

Flexbox

Flex

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



otorga más control de cómo se distribuyen los hijos

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

Propiedades del eje principal

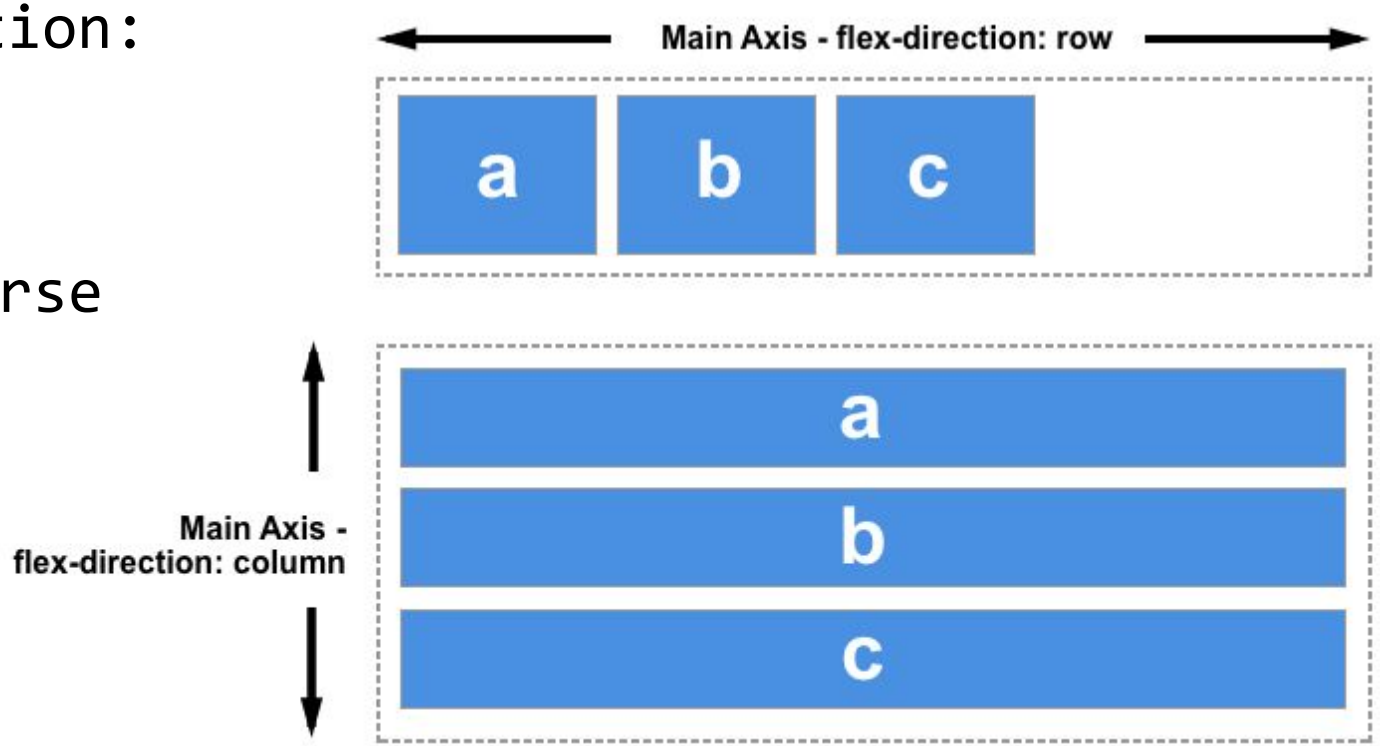
Flex diferencia dos ejes:

- **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.

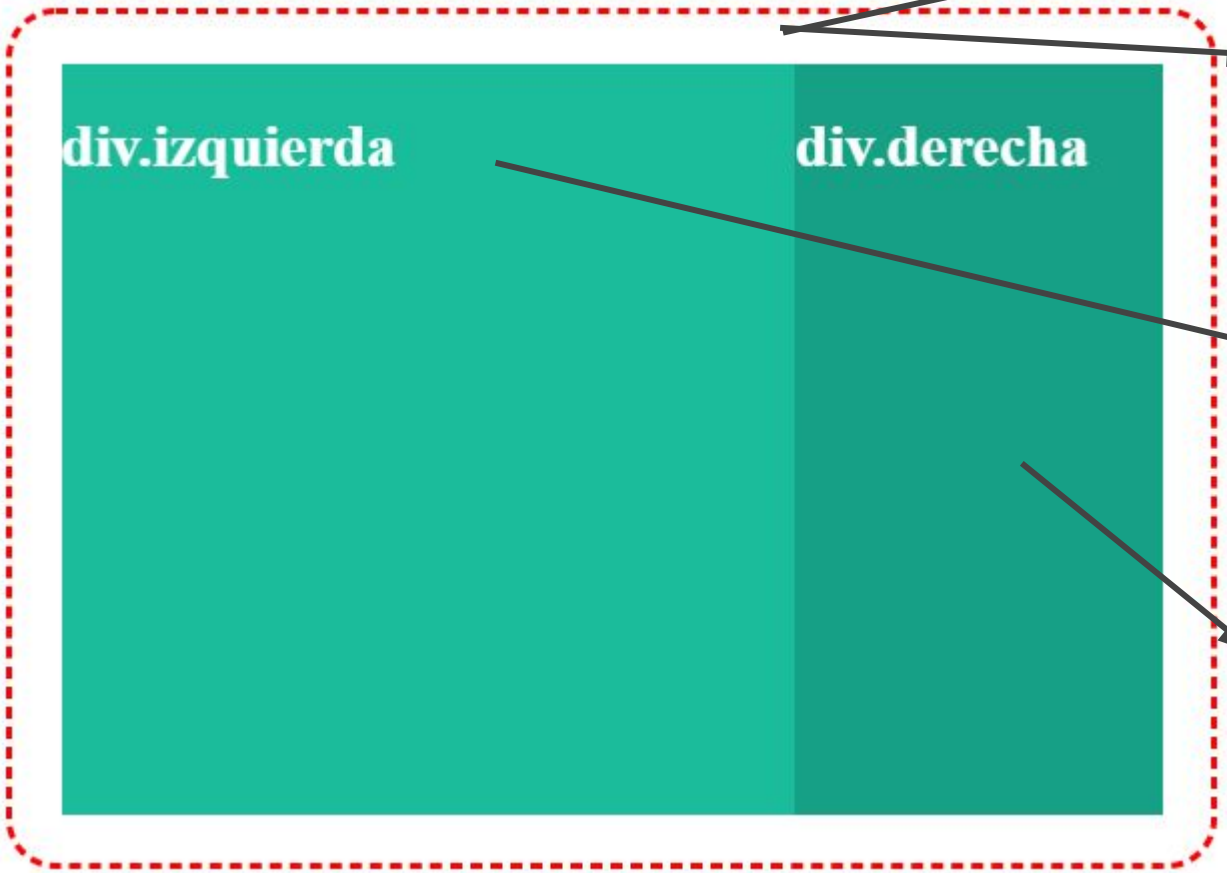
`flex-direction:`

- `row`
- `column`
- `row-reverse`
- `column-r`



Layouts usando flex

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



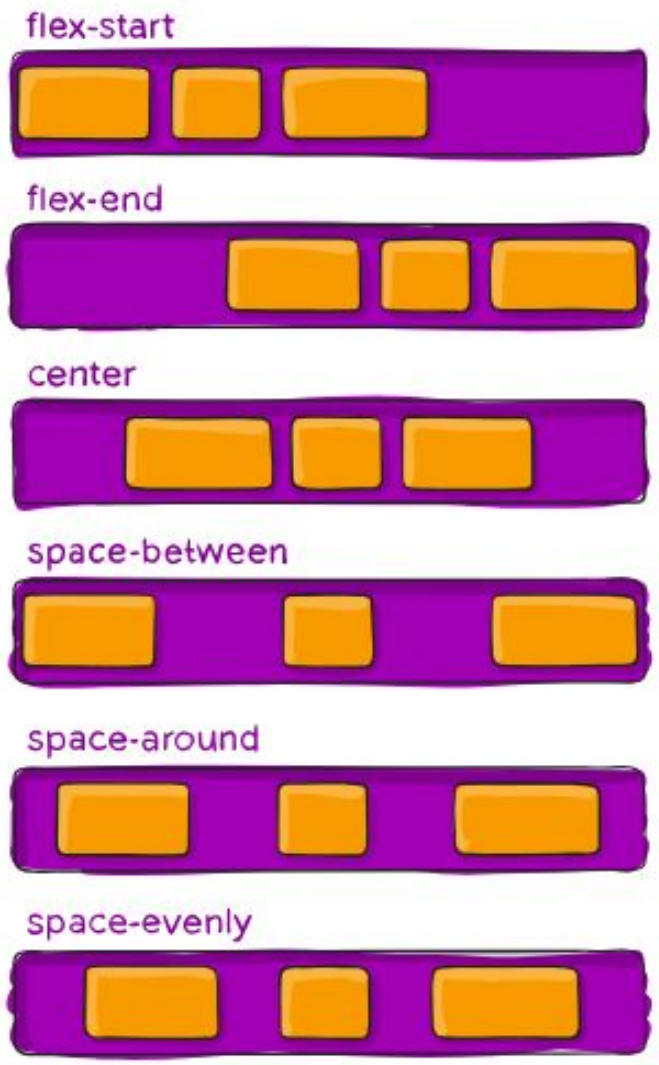

```
.container {
    display: flex;
}

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

.derecha {
    width: 200px;
    background-color: #16a085;
    color: white;
}
```

DEMO 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.

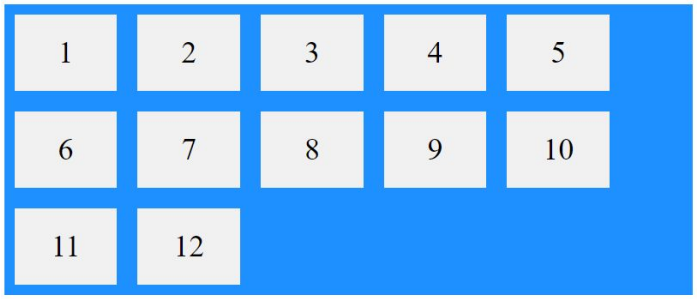


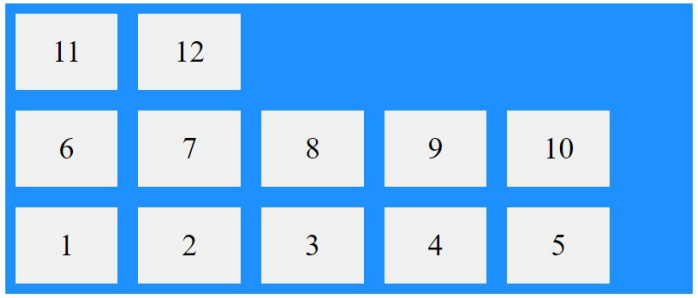
Fuente: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Flex

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

- nowrap
- wrap
- wrap-reverse





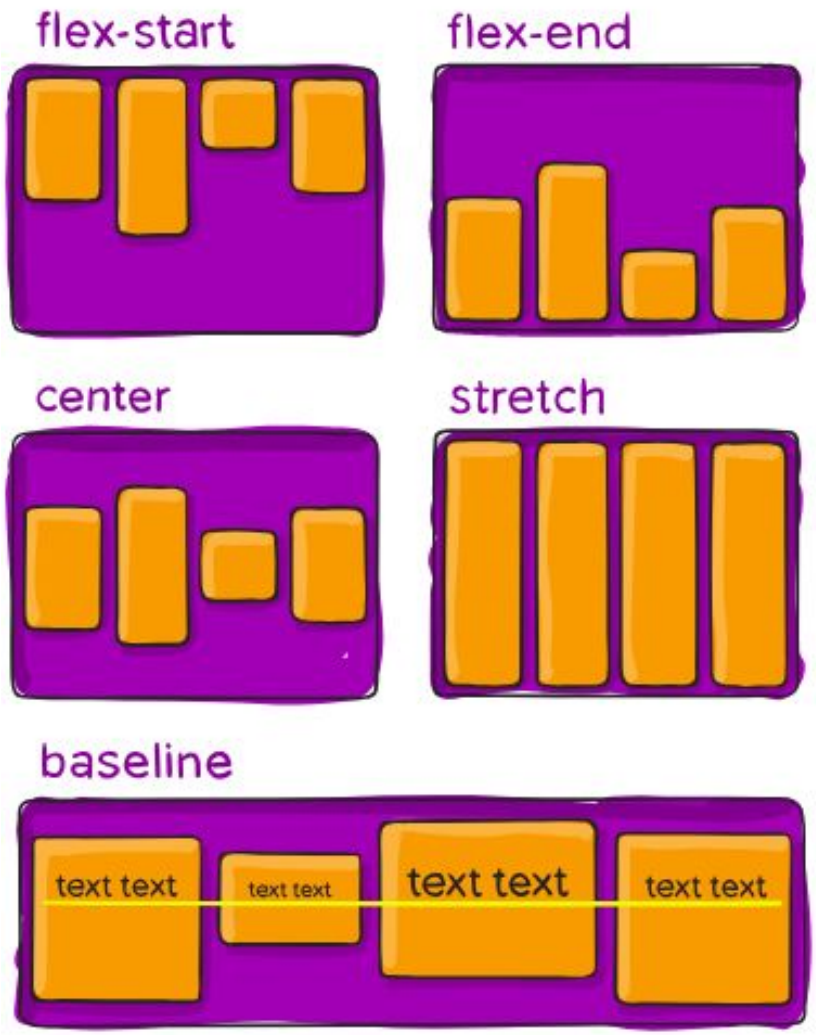
Tanto `flex-direction` como `flex-wrap` se pueden concatenar en una sola propiedad llamada: `flex-flow`

```
flex-flow: row wrap;
```

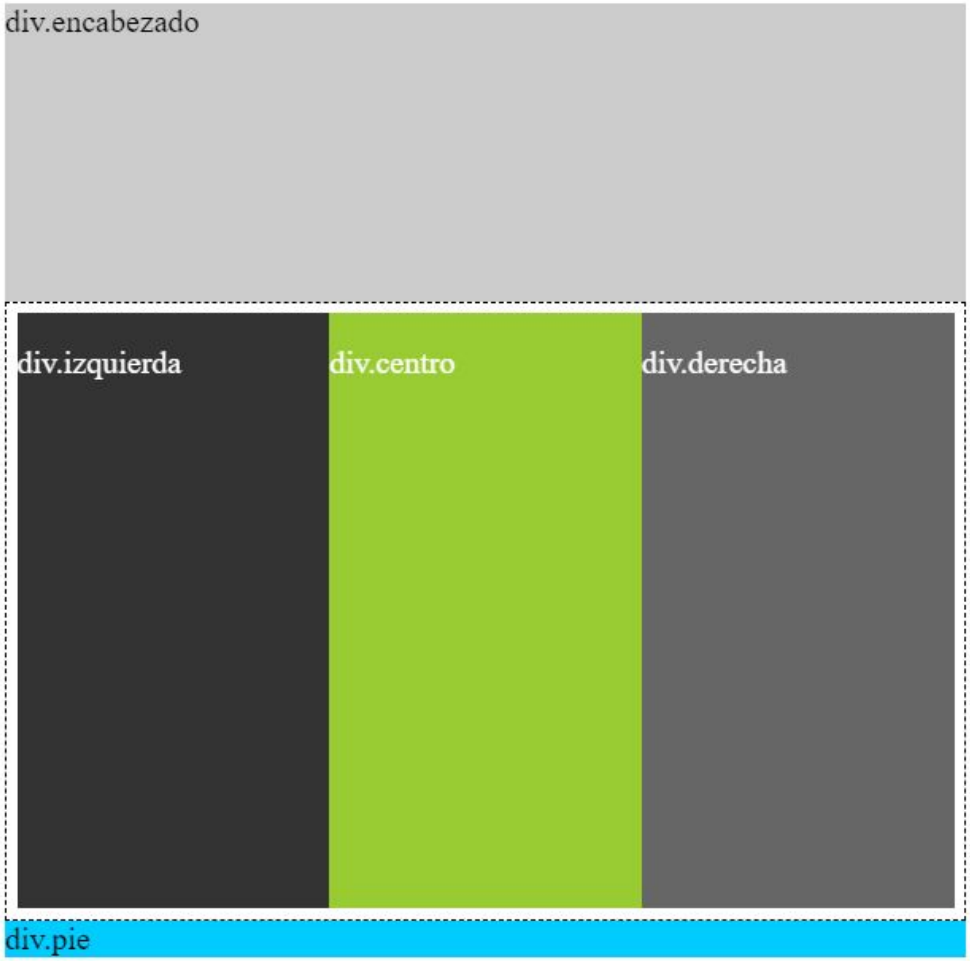
Flexbox - Alineación

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




```
.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;
}
```

DEMO https://codepen.io/webUnicen/pen/qrwLZg

Layouts usando flex

Ejemplo con muchos “centro”:

Cambiar la propiedad **`flex-wrap: wrap | no-wrap`**

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

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

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?

DEMO https://codepen.io/webUnicen/pen/oqybjQ

Flexbox - Items

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`

Fuente: https://developer.mozilla.org/es/docs/Web/CSS/flex

FLEXBOX FROGGY

Nivel 1 de 24

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
- `space-between`: Muestra
- `space-around`: Muestra

Por ejemplo, `justify-content`

```
1  #pond {
2    display: flex;
3
4  }
5
6
7
8
9
10
```

Siguiente

¿Jugamos?

https://flexboxfroggy.com/


BACK IN
5 MINS

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>
```

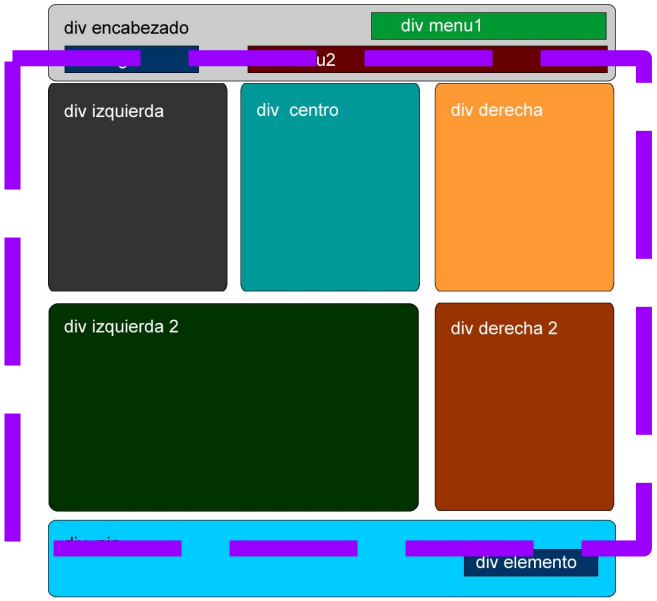
```
div {
   width: 40%;
   background-color: orange;
 }
 .chico {
   width: 50%;
   background-color: red;
 }
 .fijo1 {
   width: 300px;
    background-color: green;
 }
 .fijo2 {
   width: 500px;
    background-color: green;
 }
```

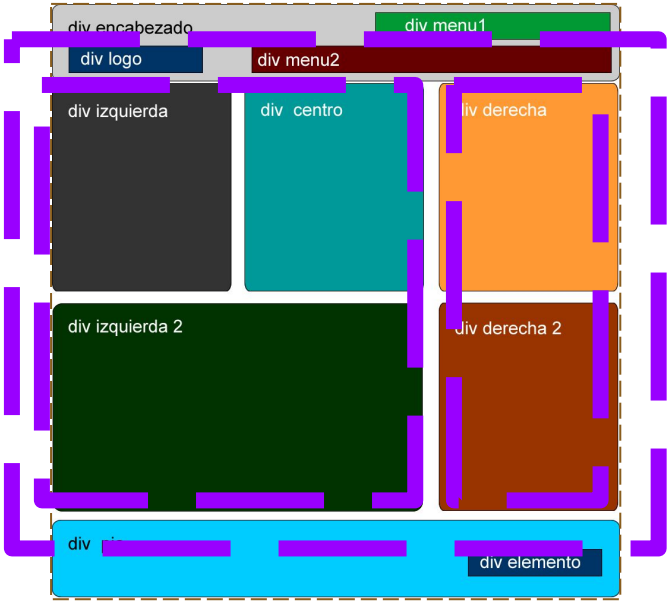
DEMO Live:https://codepen.io/webUnicen/pen/xWXOOK

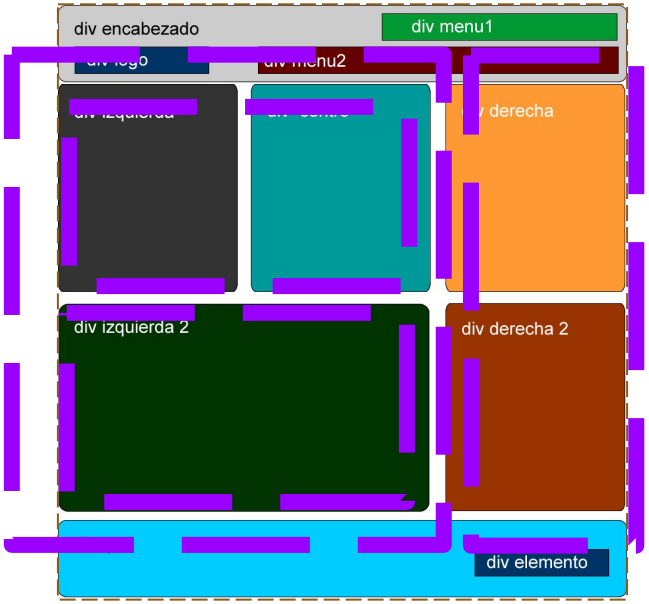
Ejercicio

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.

https://developer.mozilla.org/es/docs/Web/CSS/position

Posicionando Elementos - Relative + Absolute

HTML

```
<div>
  <ul>
    <li>Elemento 1</li>
    <li>Elemento 2</li>
  </ul>
</div>
```

CSS

```
div {
  position: relative;
  border: 6px dashed #ccc;
  height: 200px;
  top: 50px;
  width: 80%;
}
ul {
  position: absolute;
  top: 25px;
  right: 40px;
  border: 1px solid red;
}
```

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

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

Posicionando Elementos - Absolute

Comentamos “position: relative” del div.

HTML

```
<div>
  <ul>
    <li>Elemento 1</li>
    <li>Elemento 2</li>
  </ul>
</div>
```

CSS

```
div {
  position: relative;
  border: 6px dashed #ccc;
  height: 200px;
  top: 50px;
  width: 80%;
}
ul {
  position: absolute;
  top: 25px;
  right: 40px;
  border: 1px solid red;
}
```

- Elemento 1
- Elemento 2

HTML

```
<div>
  <ul>
    <li>Elemento 1</li>
    <li>Elemento 2</li>
  </ul>
</div>
```

CSS

```
div {
  /*position: relative;*/
  border: 6px dashed #ccc;
  height: 200px;
  top: 50px;
  width: 80%;
}

ul {
  position: absolute;
  top: 25px;
  right: 40px;
  border: 1px solid red;
}
```

- Elemento 1
- Elemento 2

No tiene efecto top: 50px.

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

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

Posicionando Elementos

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

HTML

```
<div>
  <ul>
    <li>Elemento 1</li>
    <li>Elemento 2</li>
  </ul>
</div>
```

CSS

```
  /*position: relative;*/
  border: 6px dashed #ccc;
  height: 200px;
  top: 50px;
  width: 80%;
}

ul {
  position: sticky;
  top: 25px;
  right: 40px;
  border: 1px solid red;
}
```

Live: https://codepen.io/webUnicen/pen/jJQBWW

Posicionando Elementos

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

HTML

```
<div>
  <p>Esto es un div</p>
</div>

<ul>
    <li>Elemento 1</li>
    <li>Elemento 2</li>
</ul>
```

CSS

```
div {
  position: relative;
  border: 6px dashed #ccc;
  height: 200px;
  top: 50px;
  text-align: left;
  padding-left: 10px;
}

ul {
  position: absolute;
  top: 25px;
  right: 40px;
  border: 1px solid red;
}
```

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

Live: http://codepen.io/webUnicen/pen/EWJMqL

Ejercicio

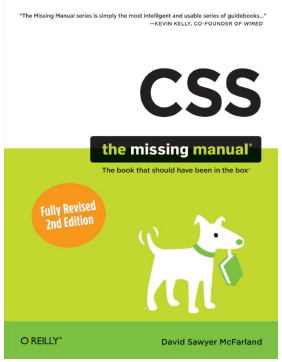
Realizar una página que contenga:

- 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

Exceso de Contenido

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

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

HTML

```
dimensiones del bloque.
.</p>

<p>overflow:scroll</p>
<div class="scroll"> Podes usar la propiedad overflow para tener un mejor control del layout. El valor por default es visble.
</div>

<p>overflow:hidden</p>
<div class="hidden"> Podes usar la propiedad overflow para tener un mejor control del layout. El valor por default es visble.
</div>
```

CSS

```
div.scroll {
    background-color: #00FFFF;
    width: 100px;
    height: 100px;
    overflow: scroll;
}

div.hidden {
    background-color: #00FF00;
    width: 100px;
    height: 100px;
    overflow: hidden;
}
```

Live: http://codepen.io/webUnicen/pen/GWLebK

Ejercicio

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, texto, etc. Se pueden entender como capas

Encimando elementos

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

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

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

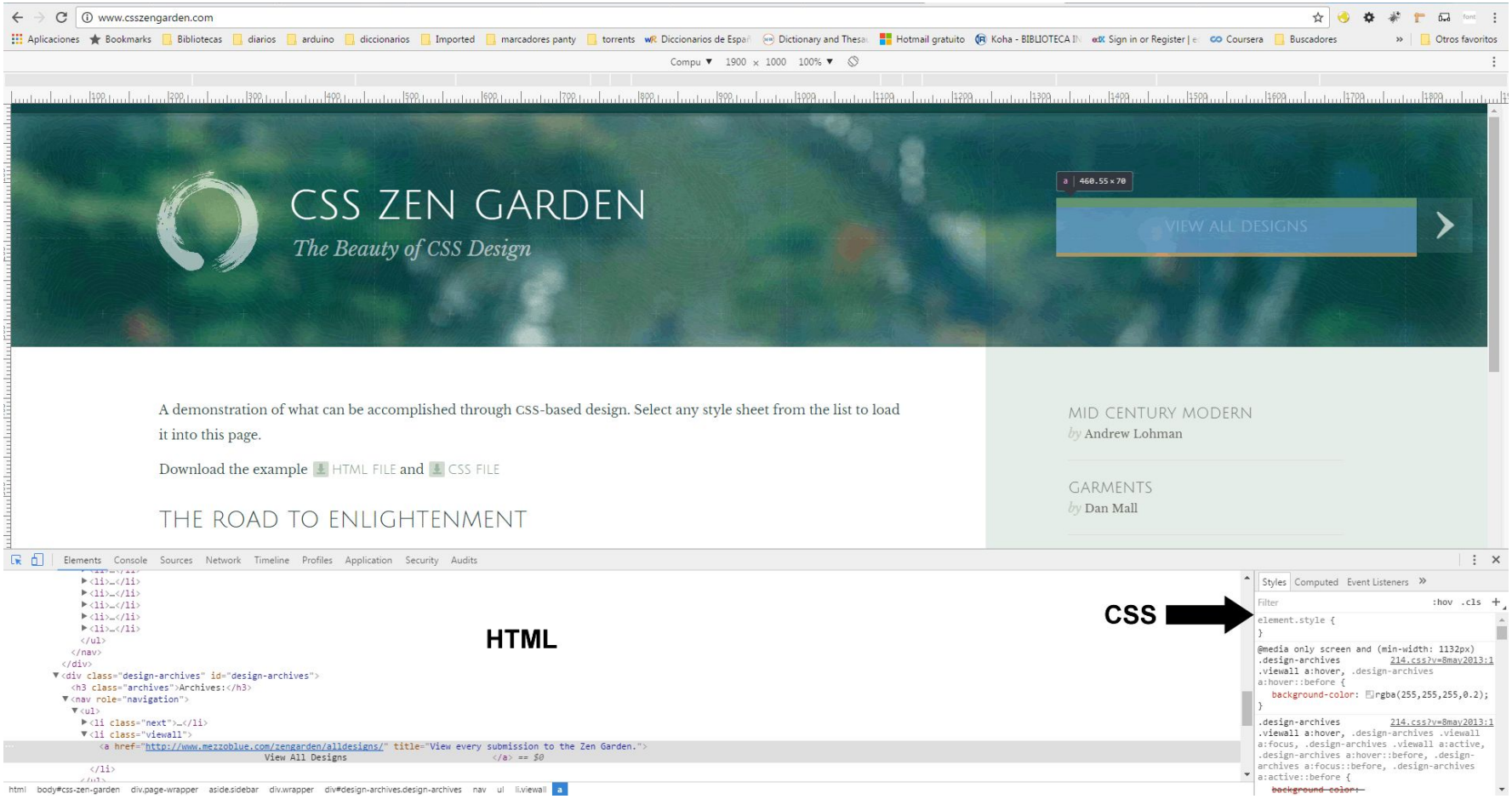
```
.box3 {
z-index: 2;
}
```

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



Herramientas >Herramientas de desarrollador
(Ctrl + Mayusc. + I)

Box Model - Herramientas, Firebug