


# HTML y CSS básico





# Introducción

# Introducción

Una página web está formada por un conjunto de ficheros

- Ficheros de **código**: HTML, CSS y Javascript
- Ficheros **multimedia**: Imágenes, audio, video



Name	Date modified	Type
 images	10/26/2019 9:26 AM	File folder
 index.html	10/26/2019 9:25 AM	Chrome HTML Do...
 main.js	10/26/2019 9:26 AM	JavaScript File
 styles.css	10/26/2019 9:26 AM	Cascading Style S...

# Introducción

Estos ficheros son interpretados por unos programas llamados **navegadores**

- Chrome
- Firefox
- Internet Explorer
- Safari

# Introducción

Los ficheros de código incluyen instrucciones en diferentes **lenguajes** para que los **navegadores** sepan cómo dibujar la página web en cuestión

- **HTML**
  - Qué elementos hay que pintar
- **CSS**
  - Qué apariencia deben tener esos elementos
- **Javascript:**
  - Cómo deben cambiar y moverse esos elementos
  - Cómo deben reaccionar a las acciones del usuario (click, teclado, etc. )
  - Conexiones con el servidor

# Introducción

Cuando accedemos a una URL en el navegador, este se conecta a un servidor de donde descarga los archivos de **código** y los archivos **multimedia**

El navegador interpreta los ficheros y **renderiza** la página web

# HTML

# HTML

HTML es un **lenguaje** que describe los elementos de una página web y su estructura


Como todos los lenguajes tiene una **sintaxis**

- Su sintaxis está basada en [XML](#)
- Si la sintaxis es correcta, el navegador podrá **renderizar** los elementos



# HTML

El fichero **index.html** es el único indispensable para crear una web

 index.html

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>Page Title</title>
```

```
  </head>
```

```
  <body>
```

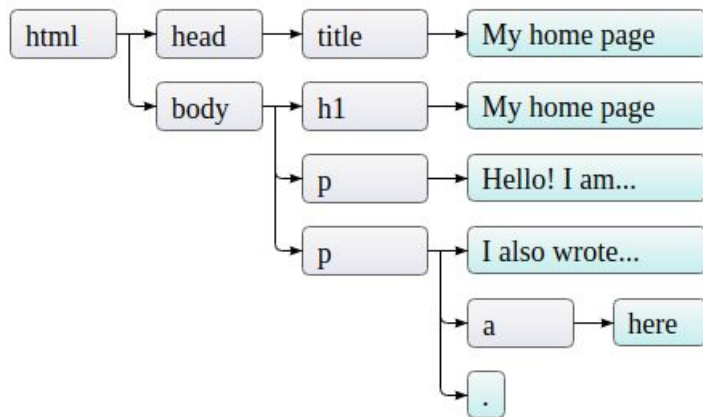
```
    <h1>My First Heading</h1>
```

```
    <p>My first paragraph.</p>
```

```
  </body>
```

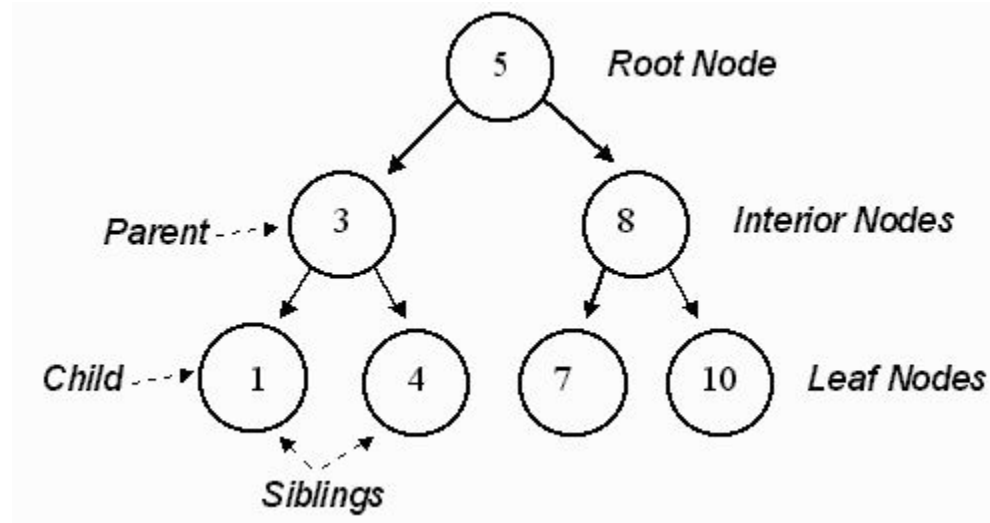
```
</html>
```

# Document Object Model (DOM)



- El DOM tiene una estructura de **árbol**
- Cada pieza blanca es lo que llamamos un **nodo** o **elemento**

# Estructura de árbol



- Estructura jerárquica de padres e hijos

# HTML

Los **nodos** se representan utilizando **tags** (etiquetas)

```
<tagName> content </tagName>
```

- <tagName> abre un nodo
- </tagName> lo cierra
- Entre la apertura y el cierre de un nodo se encuentran su **contenido**

Ejemplo:

```
<body>  
  <h1>My First Heading</h1>  
  <p>My first paragraph.</p>  
</body>
```

# HTML


Hay cuatro **nodos** fundamentales

```
<!DOCTYPE html>
<html>
  <head>
  </head>

  <body>
  </body>
</html>
```

- `<!DOCTYPE html>` Marca que esto es un fichero **HTML5**
- `<html></html>` Es el **nodo raíz**
- `<head></head>` Contiene **propiedades** de la página
- `<body></body>` Contiene los **elementos** visibles de la página

# HTML

 index.html

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>Page Title</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1>My First Heading</h1>
```

```
    <p>My first paragraph.</p>
```

```
  </body>
```

```
</html>
```

# Content tags

# Content tags

- `<p> texto </p>`

Contiene **párrafos** de texto

- `<h1> título </h1>`

Contiene **títulos**

- `<h2> título </h2>`

Contiene **títulos** más pequeños (eg: **subtítulos**)

- `...`

- `<h6> título </h6>`

- `<div> nodos </div>`

Se utiliza como **contenedor** de otros nodos

- `<br>`

Salto de línea (no cierra)



# Ejercicio Introducción

- Crea una carpeta **workspace** donde pondrás todos los ejercicios
- Crea una subcarpeta **workspace/html** donde pondrás todos los ejercicios de HTML

# Ejercicio HTML

- Crea una nueva carpeta dentro de **workspace/html** que contenga este ejercicio
- Crea un fichero **index.html** que contenga
  - La estructura fundamental de una página HTML
  - Un **título de página** personalizado
- Añade un nodo **h1** con el mismo título al contenido de la página

# Ejercicio content tags

- Partiendo del ejercicio anterior
  - Añade un div que contenga **tres** párrafos
  - Utiliza los tres primeros párrafos de [lorem ipsum](#)

# Content tags

## Bullet lists

```
<ul>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>
```

- Coffee
- Tea
- Milk

## Ordered lists

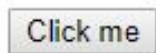
```
<ol>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>
```

1. Coffee
2. Tea
3. Milk

# Content tags

## Button

```
<button>Click me</button>
```



# Tag attributes

# Tag attributes

Los **nodos** pueden tener **atributos**

- Los atributos definen información adicional sobre el nodo

```
<tagName attributeName="value"> content </tagName>
```

Ejemplo:

```
<a href="https://www.google.com"> This is a link </a>
```

# Content tags

Algunos tags dependen de sus **atributos** para ser útiles:

## Enlaces (href: url destino)

```
<a href="https://www.google.com"> This is a link </a>
```

[This is a link](https://www.google.com)

## Imagenes (src: url imagen)

```

```

```

```





# Ejercicio content tags II

- Partiendo del ejercicio anterior
  - Añade una imagen a través de una URL de internet
  - Añade una imagen a través de un fichero en tu ordenador (URL local)

# Ejercicio content tags III

- Partiendo del ejercicio anterior
  - Añade una **bullet list** con 4 enlaces a 4 páginas web diferentes

# Ejercicio content tags IV

- Partiendo del ejercicio anterior
  - Haz que las imágenes enlacen a <http://www.google.com>

# HTML forms

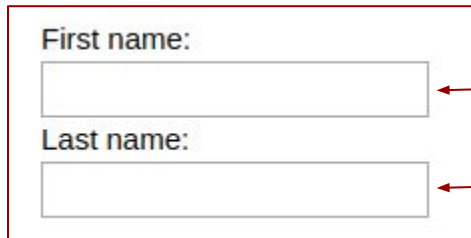
# <form> node

El tag **form** define el inicio y el fin de un **formulario** para **recoger datos**

- Un formulario HTML incluye **form elements**

```
<form>  
  form elements  
</form>
```

Form



A diagram of a form with a red border. Inside, there are two input fields. The first field is preceded by the text "First name:" and the second by "Last name:". Red arrows point from the text "Form elements" to each of the input fields.

Form elements

# <Input> element

El elemento input se utiliza para recoger información del usuario

```
<input type="typeName" name="elemName">
```

El **aspecto** y **funcionalidad** de los elementos input cambia mucho en función del atributo **type**

# <Input> element

Input para recoger **texto**

```
<input type="text" name="user">
```

```
<input type="password" name="pass">
```



Ejemplo

# <Input> element

Input para recoger **fechas**

```
<input type="date" name="bday">
```

[Ejemplo](#)



# <Input> element

Input para recoger **opciones**

```
<input type="radio" name="gender" value="male"> Male<br>  
<input type="radio" name="gender" value="female"> Female<br>  
<input type="radio" name="gender" value="other"> Other
```

- ☐ Male
- ☐ Female
- ☐ Other

Ejemplo

# <Input> element

Input para recoger **múltiples opciones**

```
<input type="checkbox" name="vehicle1" value="Bike"> I have a bike<br>
<input type="checkbox" name="vehicle2" value="Car"> I have a car<br>
<input type="checkbox" name="vehicle3" value="Boat" checked> I have a boat<br>
```

- ☐ I have a bike
- ☐ I have a car
- ☒ I have a boat

[Ejemplo](#)

# <Input> element

El usuario confirma su elección mediante un input de tipo **submit**

```
<form action="/action_page.php">
  <input type="checkbox" name="vehicle1" value="Bike"> I have a bike<br>
  <input type="checkbox" name="vehicle2" value="Car"> I have a car<br>
  <input type="checkbox" name="vehicle3" value="Boat" checked> I have a boat<br><br>
  <input type="submit" value="Submit">
</form>
```

- ☐ I have a bike
- ☐ I have a car
- ☒ I have a boat

Submit

Ejemplo


# <Input> element

Lista completa de input types: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input>

# <select> element

Los elementos **<select>** nos permiten escoger opciones de una lista

```
<select name="car-select">  
  <option value="volvo">Volvo</option>  
  <option value="saab">Saab</option>  
  <option value="mercedes">Mercedes</option>  
  <option value="audi">Audi</option>  
</select>
```



[Ejemplo](#)

# <select> element

Podemos cambiar el valor por defecto mediante el atributo **selected**

```
<select name="car-select">  
  <option value="volvo">Volvo</option>  
  <option value="saab">Saab</option>  
  <option value="mercedes">Mercedes</option>  
  <option value="audi" selected="selected">Audi</option>  
</select>
```



[Ejemplo](#)

# Ejercicio HTML forms

- Partiendo del ejercicio anterior
  - Añade un formulario que recoja usuario, contraseña y correo
  - El usuario también debe poder seleccionar si desea recibir publicidad (activado por defecto)

# Ejercicio HTML forms II

- Partiendo del ejercicio anterior
  - Investiga qué es un placeholder
  - Añade un placeholder a los campos **usuario** y **correo**



# Ejercicio HTML forms III

- Partiendo del ejercicio anterior
  - Añade la opción de seleccionar un rango de edad de una lista
    - -18
    - 18-25
    - 25+
  - **18-25** debe ser el valor por defecto

# Media tags

# Audio

El tag **HTML5** `<audio></audio>` nos permite reproducir sonidos en el navegador

# Ejercicio

- Busca y lee la documentación del elemento HTML5 **audio**
- Crea un tag audio que reproduzca un sonido. El audio debe poder pausarse y rebobinarse utilizando los controles HTML5

# Video

El tag **HTML5** `<video></video>` nos permite reproducir sonidos en el navegador

# Ejercicio

- Busca y lee la documentación del elemento HTML5 **video**
- Incluye dos videos en un documento HTML. El video debe ser el mismo pero reproducirse en dos tamaños diferentes
- Los videos deben poder pausarse y rebobinarse utilizando los controles HTML5

# Documentación

Hay muchos **elementos** HTML

El 99% del tiempo se usan los básicos, para otros casos leer [documentación](#)

CSS



# CSS


CSS es un **lenguaje** diferente que describe **cómo** se deberían renderizar los elementos HTML

Como todos los lenguajes tiene una **sintaxis**

- Si la sintaxis es correcta, el navegador Aplicará las instrucciones a la hora de **renderizar** los elementos

# CSS

Lo ficheros **nombre.css** contienen las instrucciones **CSS**


 styles.css

```
body {  
  background-color: lightblue;  
}
```

```
h1 {  
  color: white;  
  text-align: center;  
}
```

# CSS

Para que los ficheros CSS funcionen deben enlazarse desde el fichero **index.html**

 index.html

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>Page Title</title>
```

```
    <link rel="stylesheet" href="styles.css">
```

```
  </head>
```

```
  <body>
```

```
</body>
```

```
</html>
```



# Sintaxis CSS

```
selector {  
  property: value;  
  property: value;  
}
```

```
p {  
  /* Como cambiar color de texto: */  
  color: red;  
}
```

```
h1 {  
  color: white;  
  /* Como cambiar color de fondo: */  
  background-color: blue;  
}
```

Lorem ipsum dolor

Title

# Comentarios CSS

```
p {  
  color: red;  
}
```

```
h1 {  
  color: white;  
  background-color: blue;  
}
```

Lorem ipsum dolor

Title

Demo

# Ejercicio CSS

- Crea una nueva carpeta en workspace/HTML
- Crea un `index.html` y un fichero `estilo.css`
- Crea un párrafo rojo con lorem ipsum

# Sintaxis CSS

¿Qué pasa si quiero pintar dos párrafos de diferente color?

```
p {  
  color: red;  
}
```



# Selectores


# Selectores

Hemos visto los selectores de tipo tag, pero hay más

```
p {  
  color: red;  
}
```


# Selectores Id

A cualquier **nodo** se le puede asignar un **atributo id**

 index.html

```
<p id="parrafo-1"> rojo </p>
<p id="parrafo-2"> verde </p>
<p id="parrafo-3"> azul </p>
```

Para cambiar sus estilos **individualmente**

 styles.css

```
#parrafo-1 {
  color: red;
}
#parrafo-2 {
  color: green;
}
#parrafo-3 {
  color: blue;
}
```


rojo

verde


azul

# Selectores Id

A cualquier **nodo** se le puede asignar un **atributo id**

 index.html


```
<p id="parrafo-1"> rojo </p>
<p id="parrafo-2"> verde </p>
<p id="parrafo-3"> azul </p>
```

 styles.css

```
#parrafo-1 {
  color: red;
}
#parrafo-2 {
  color: green;
}
#parrafo-3 {
  color: blue;
}
```

# Selectores Id

¿Y si queremos un párrafo rojo y **dos** verdes?

 index.html


```
<p id="parrafo-1"> rojo </p>
```

```
<p id="parrafo-2"> verde </p>
```


```
<p id="parrafo-3"> verde</p>
```

# Selectores Id

¿Y si queremos un párrafo rojo y **dos** verdes?

 index.html


```
<p id="parrafo-1"> rojo </p>
<p id="parrafo-2"> verde </p>
<p id="parrafo-3"> verde</p>
```

 styles.css


```
#parrafo-1 {
  color: red;
}
#parrafo-2 {
  color: green;
}
#parrafo-3 {
  color: green;
}
```

# Selectores Id

¿Y si queremos un párrafo rojo y **dos** verdes?

 index.html


```
<p id="parrafo-1"> rojo </p>
<p id="parrafo-2"> verde </p>
<p id="parrafo-3"> verde</p>
```

 styles.css


```
#parrafo-1 {
  color: red;
}
#parrafo-2, #parrafo-3 {
  color: green;
}
```

# Selectores **class**

A cualquier **nodo** se le puede asignar un **atributo class**

 index.html

```
<p class="parrafo-rojo"> rojo </p>
<p class="parrafo-verde"> verde </p>
<p class="parrafo-verde"> verde </p>
<p class="parrafo-verde"> verde </p>
<p class="parrafo-verde"> verde </p>
```

 styles.css


```
.parrafo-rojo {
  color: red;
}
.parrafo-verde {
  color: green;
}
```

rojo  
verde  
verde  
verde  
verde



# Selectores class

A cualquier **nodo** se le puede asignar un **atributo class**

 index.html

```
<p class="parrafo-rojo"> rojo </p>
<p class="parrafo-verde"> verde </p>
<p class="parrafo-verde"> verde </p>
<p class="parrafo-verde"> verde </p>
<p class="parrafo-verde"> verde </p>
```

 styles.css

```
.parrafo-rojo {
  color: red;
}
.parrafo-verde {
  color: green;
}
```

rojo  
verde  
verde  
verde  
verde

# Ejercicio Selectores

- Partiendo del ejercicio anterior
  - Añade 3 párrafos más de diferentes colores


# Ejercicio Selectores

- Partiendo del ejercicio anterior
  - Cambia el color de fondo de dos de esos párrafos utilizando **un solo selector de clase**

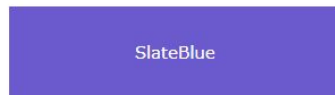
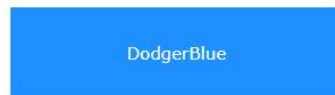
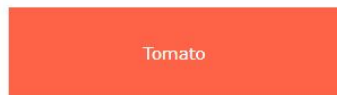
# Color

# Colores

Podemos asignar colores utilizando sus nombres:

 styles.css

```
.parrafo-rojo {  
  color: red;  
}  
.parrafo-verde {  
  color: green;  
}
```



[Lista completa de nombres](#)

# Colores

Pero es más habitual utilizar los valores **hexadecimales**


 styles.css

```
.parrafo-rojo {  
  color: #AA5555;  
}  
.parrafo-verde {  
  color: #7FFF00;  
}
```

[Referencia](#)

# Colores

También podemos usar códigos **RGB** y **RGBA**


 styles.css

```
/* RGB */  
.parrafo-rojo {  
  color: rgb(255, 0, 0);  
}
```

```
/* RGBA (70% de opacidad) */  
.parrafo-verde {  
  color: rgba(0, 255, 0, 0.7);  
}
```

# Colores

Podemos cambiar el color del **texto**, el color de **fondo** y el color de los **bordes**

 styles.css

```
p {  
  color: #AA5555;  
  background-color: #AA5555;  
  border-color: #AA5555;  
}
```

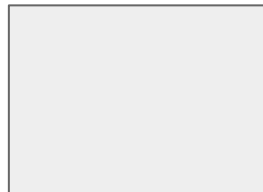


# Tamaño

# Tamaño

El tamaño de los elementos se modifica utilizando las propiedades **width** y **height**

```
#node {  
  background-color: #888888;  
  width: 400px;  
  height: 300px;  
}
```

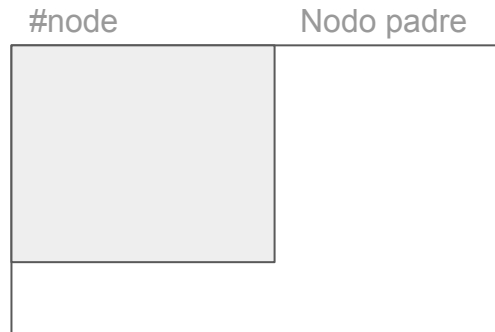


# Tamaño

Los **valores** que son **medidas** aceptan diferentes tipos de **unidades**

- **Porcentaje** respecto al **nodo padre**

```
#node {  
  background-color: #888888;  
  width: 50%;  
  height: 75%;  
}
```

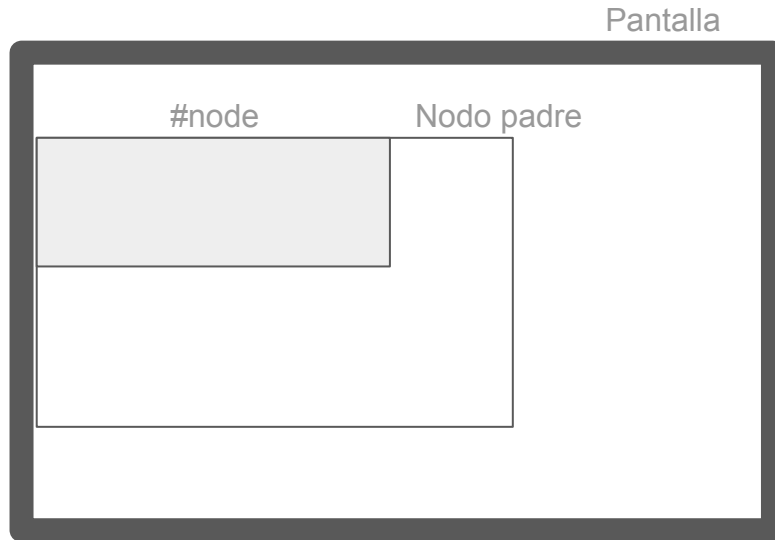


# Tamaño

Los **valores** que son **medidas** aceptan diferentes tipos de **unidades**

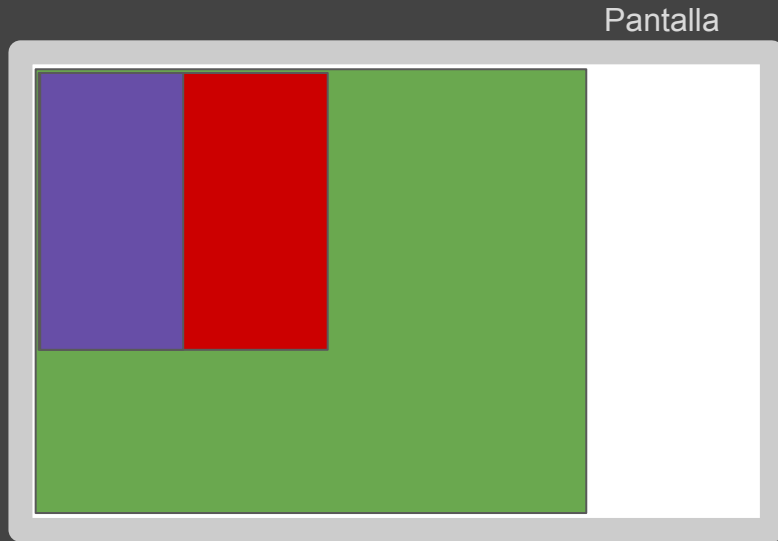
- **Porcentaje** respecto al **tamaño de la pantalla**

```
#node {  
  background-color: #888888;  
  width: 50vw; /* vw: visor width */  
  height: 25vh; /* vw: visor height */  
}
```



# Ejercicio Tamaños (revisar height)

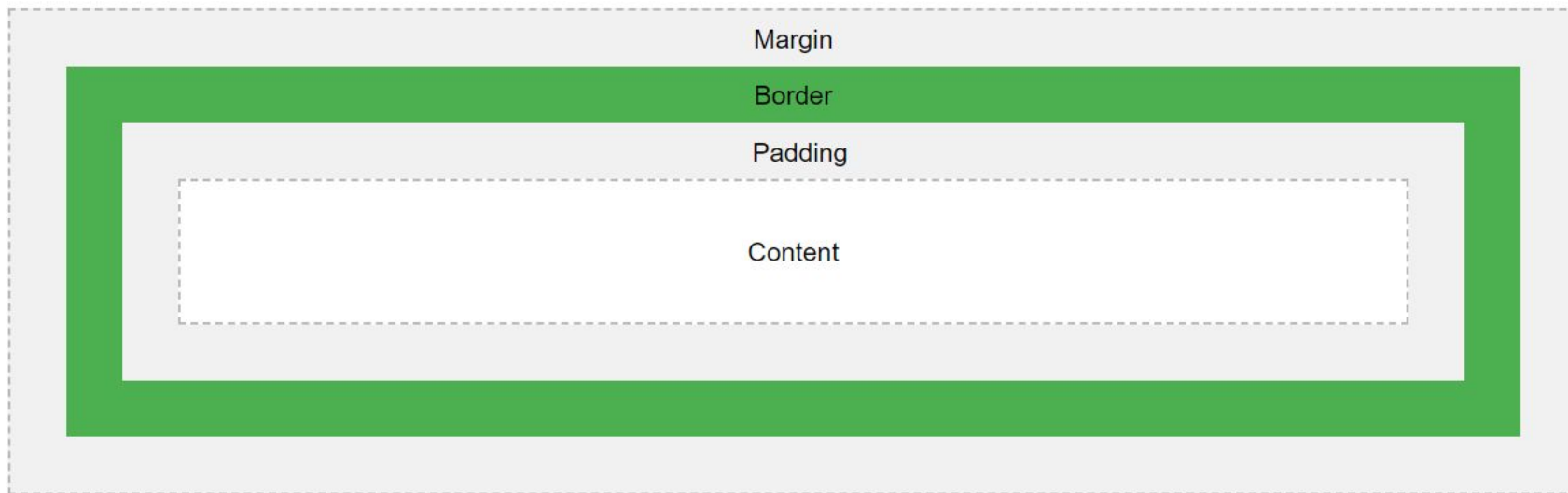
- Replica esta figura con HTML y CSS



# The box model

# Box model

Todos los elementos HTML se pueden entender como cajas que tienen cuatro elementos



# Border



# Border

Los bordes se configuran a partir de 4 propiedades diferentes:

```
#node {  
  border-width: 2px;           // Grosor  
  border-color: #222222;      // Color  
  border-style: solid;        // Estilo (referencia)  
  border-radius: 5px;         // Redondeado  
}
```

# Border

Los bordes se configuran a partir de 4 propiedades diferentes:

```
#node {  
  border-width: 2px;           // Grosor  
  border-color: #222222;       // Color  
  border-style: solid;         // Estilo (referencia)  
  border-radius: 5px;          // Redondeado  
}
```

O a partir de una única propiedad **border**:

```
#node {  
  border: 2px solid #222222;    // El redondeado iría aparte  
}
```

# Border

Se puede modificar la propiedad de un único lado utilizando las palabras **left**, **right**, **top**, **bottom**

```
#node {  
  border-left-width: 2px;           // Grosor  
  border-right-color: #222222;     // Color  
  border-top-style: solid;         // Estilo (referencia)  
  border-bottom-radius: 5px;       // Redondeado  
}
```

# Box-sizing

Por defecto, el alto y el ancho que deben medir los componentes **no tiene en cuenta el borde**

Podemos modificar este comportamiento mediante box-sizing

```
#node {  
  box-sizing: border-box;           // Tiene en cuenta el borde  
}
```

```
#node {  
  box-sizing: content-box;         // No tiene en cuenta el borde  
}
```

# Ejercicio Bordes

- Añade bordes de diferentes grosores al ejercicio anterior: cuanto más grande el div, más grande el grosor del borde

# Margen y padding

# Box model

**Margin y padding** se utilizan para crear espacio alrededor de los elementos



# Margin y Padding

Se configuran a partir de 4 propiedades:

```
#node {  
  margin-left: 1px;  
  margin-right: 2px;  
  margin-top: 3px;  
  margin-bottom: 4px;  
}
```

```
#node {  
  padding-left: 1px;  
  padding-right: 2px;  
  padding-top: 3px;  
  padding-bottom: 4px;  
}
```

Ejemplos: [margin](#), [padding](#)



# Margin y Padding

Se puede condensar en una sola línea:

```
#node {  
  margin: 1px 2px 3px 4px; /* top right bottom left */  
}
```

```
#node {  
  padding: 1px 2px 3px 4px; /* top right bottom left */  
}
```

# Margin y Padding

Se puede condensar más:

```
#node {  
  margin: 10px 20px;      /* top and bottom: 10px    left and right: 20px */  
}
```

```
#node {  
  padding: 10px 20px;     /* top and bottom: 10px    left and right: 20px */  
}
```

# Margin y Padding

Se puede condensar todavía más:

```
#node {  
  margin: 20px; /* 20px en todas direcciones */  
}
```

```
#node {  
  padding: 20px; /* 20px en todas direcciones */  
}
```

# Problemas asignando altura

# Box-sizing

Para que asignar la altura de un elemento en porcentaje funcione

```
#child {  
  box-sizing: border-box;  
}
```

Su padre debe tener una altura asignada

```
#parent {  
  height: 800px;  
}
```

# Box-sizing

Por defecto, los elementos **no tienen en cuenta el padding del elemento contenedor** a la hora de calcular su tamaño mediante porcentajes

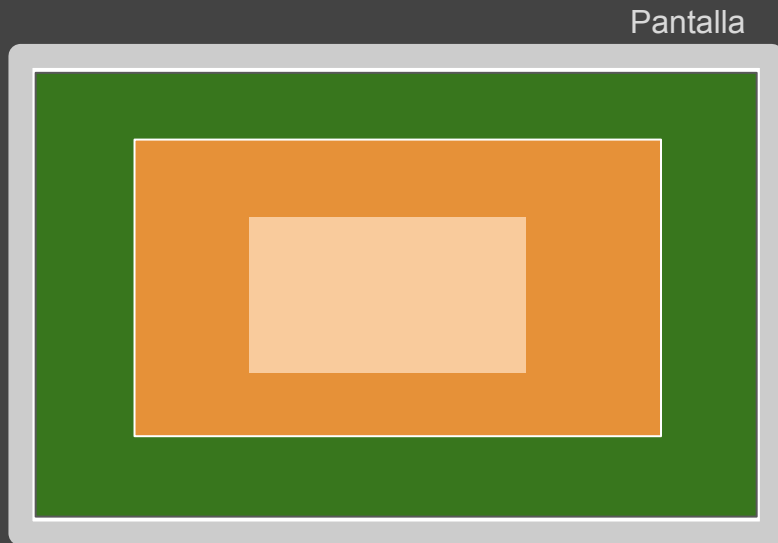
Podemos modificar este comportamiento mediante box-sizing

```
#node {  
  box-sizing: border-box;           // Los hijos tendrán en cuenta el padding  
}
```

```
#node {  
  box-sizing: content-box;         // Los hijos tendrán en cuenta el padding  
}
```

# Ejercicio Tamaños

- Replica esta figura con HTML y CSS
  - Utiliza **padding** para separar el rectángulo naranja del verde y **margin** en el resto de casos
  - El rectángulo pequeño es de color blanco transparente



Texto



# Texto

CSS dispone de muchas propiedades para dar formato al texto

## Color

```
#node {  
  color: black;  
  background-color: #f1f1f1;  
}
```

Lorem ipsum dolor

# Texto

## Alineado

```
#node {  
  text-align: left;  
}
```

Lorem ipsum dolor

```
#node {  
  text-align: center;  
}
```

Lorem ipsum dolor

```
#node {  
  text-align: right;  
}
```

Lorem ipsum dolor

# Texto

## Alineado

```
#node {  
  text-align: left;  
}
```

Lorem ipsum dolor sit amet,  
consectetur adipiscing elit.  
Nulla mauris nibh, consequat  
sit amet sem sit amet

```
#node {  
  text-align: justify;  
}
```

Lorem ipsum dolor sit amet,  
consectetur adipiscing elit.  
Nulla mauris nibh, consequat  
sit amet sem sit amet

# Texto

## Decoración

```
#node {  
  text-decoration: underline;  
}
```

Lorem ipsum dolor

```
#node {  
  text-decoration: line-through;  
}
```

~~Lorem ipsum dolor~~

# Texto

## Espaciado

```
#node {  
  letter-spacing: 10px;  
}
```

Lorem ipsum dolor

# Texto

## Alineado vertical

```
#node {  
  line-height: 0.8;  
}
```

```
#node {  
  line-height: 1.5;  
}
```

Lorem ipsum dolor sit amet,  
consectetur adipiscing elit.  
Nulla mauris nibh, consequat  
sit amet sem sit amet

Lorem ipsum dolor sit amet,  
consectetur adipiscing elit.  
Nulla mauris nibh, consequat  
sit amet sem sit amet

Fuente

# Fuente

## Tamaño

```
#node {  
  font-size: 18px;  
}
```

Lorem ipsum dolor

```
#node {  
  font-weight: 800;  
}
```

**Lorem ipsum dolor**

```
#node {  
  font-weight: 500;  
}
```

**Lorem ipsum dolor**



# Fuente

## Familia y estilo

```
#node {  
  font-family: "Times New Roman";  
}
```

Lorem ipsum dolor

```
#node {  
  font-style: italic;  
}
```

*Lorem ipsum dolor*

# Content tags

- `<b> texto </b>`

Aplica **negrita** al texto

- `<span> texto </span>`

Contenedor sin salto de línea ([ejemplo](#))

# Ejercicio texto

- Crea un título con las siguientes características
  - Tamaño de la fuente 40px
  - Fuente: Helvetica
  - El texto debe estar centrado respecto al párrafo siguiente
- Crea un párrafo de 3 líneas con las siguientes características
  - Tamaño de la fuente: 20 px
  - El texto debe estar en cursiva
  - El texto debe cuadrarse horizontalmente (debe estar justificado)

# Display

# Display

**Display** determina de qué forma se muestran los elementos

Display tiene **tres** modos

```
#node {  
  display: block;  
}
```

```
#node {  
  display: inline-block;  
}
```

```
#node {  
  display: inline;  
}
```

# Display

```
#node {  
  display: block;  
}
```

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

*hi*

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

- Por defecto ocupan el máximo de ancho y se colocan uno debajo de otro
- Valor **por defecto** en elementos que actúan como bloques enteros
  - **div, ul, p, h1**, etc.

# Display

```
#node {  
  display: inline;  
}
```

Pellentesque *inline element* morbi tristique senectus  
Donec eu libero sit amet quam egestas semper. Aenean

- Valor **por defecto** en elementos que pueden estar contenidos en un texto
  - **a, b, span**, etc.
- Se mantiene **en línea** con el texto. No acepta modificaciones en las propiedades **width** y **height**

# Display

```
#node {  
  display: inline-block;  
}
```



- Permite posicionar elementos horizontalmente y modificar su **width** y **height**



# Display

```
#node {  
  display: inline-block;  
}
```

Pellentesque

*inline  
block*

*inline  
block*

*inline  
block*

morbi tristique

senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

- Permite posicionar elementos horizontalmente y modificar su **width** y **height** incluso si están en línea con un texto

Ejemplo

# Display

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

- Oculta el elemento y **deja de ocupar espacio**

# Display

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

- Oculta el elemento y **deja de ocupar espacio**

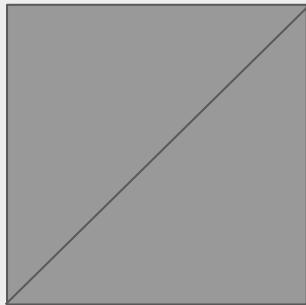
```
#node {  
  display: none;  
}
```

- Oculta el elemento y **sigue** ocupando espacio

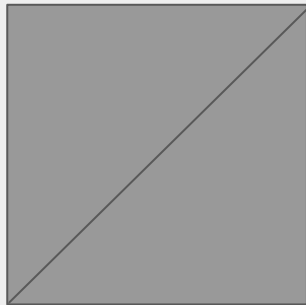
**Ejemplo:** [https://www.w3schools.com/css/css\\_display\\_visibility.asp](https://www.w3schools.com/css/css_display_visibility.asp)

# Ejercicio Tamaños

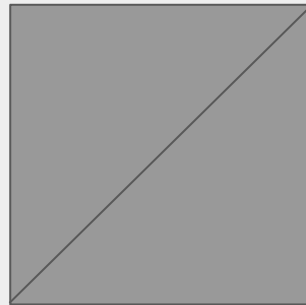
- Crea tres divs posicionados horizontalmente. Dentro de cada div tiene que haber una imagen cualquiera y un texto en la parte inferior a modo de descripción



Targaryen



Lannister



Stark

# Position

# Position

Hemos visto que las propiedades **margin** y **padding** se pueden usar para **posicionar** elementos HTML

Disponemos de otra herramienta: **position**

# Position

La posición se modifica a partir de cuatro propiedades

```
#node {  
  left: 20px;  
  top: 30px;  
}
```

```
#node {  
  right: 20px;  
  bottom: 30px;  
}
```



# Position

**Left, top, right y bottom** se comportan de forma diferente en función del valor que asignemos a la propiedad **position**

```
#node {  
  position: static;  
}
```

```
#node {  
  position: relative;  
}
```

```
#node {  
  position: absolute;  
}
```

```
#node {  
  position: fixed;  
}
```

# Position static

En la posición **static**, las propiedades LRTB no afectan al elemento

- Este es el valor **por defecto** en todos los elementos HTML

```
#node {  
  position: static;  
  left: 50px;  
  top: 50px;  
}
```

[Ejemplo](#)

# Position relative

En la posición **relative** los elementos se mueven partiendo de su posición inicial

```
#node {  
  position: relative;  
  left: 20px;  
}
```



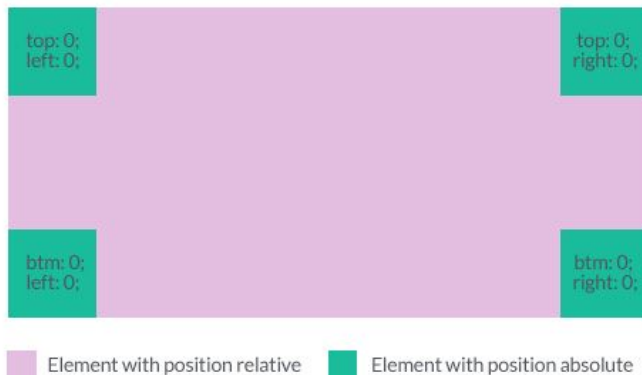
Ejemplo

# Position absolute

En la posición **absolute** los elementos se posicionan respecto al **pariente más cercano con position relative**

- Si no lo hay, se posiciona respecto a la pantalla
- El elemento deja de ocupar espacio, sale del **document flow**

```
#node {  
  position: absolute;  
  left: 0px;  
  top: 0px;  
}
```



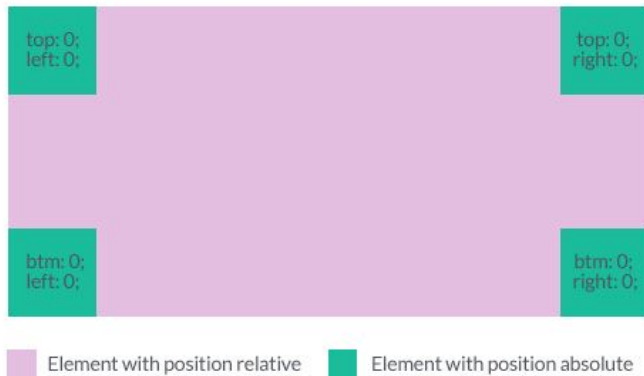
[Ejemplo](#)

# Position fixed

Similar a la posición absolute, pero mantienen su posición **aunque se haga scroll**

- El elemento deja de ocupar espacio, sale del **document flow**

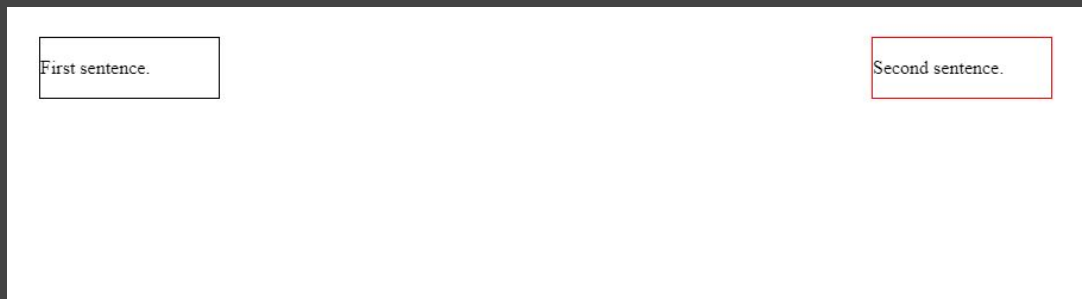
```
#node {  
  position: absolute;  
  left: 0px;  
  top: 0px;  
}
```



[Ejemplo](#)

# Ejercicio posicionamiento

Replica la siguiente figura:



- Reglas:
  - Las cajas no se pueden posicionar usando **margin**
  - Puedes posicionar el texto usando **margin**
  - 10px de margen entre las cajas y los bordes de la pantalla

# Ejercicio extra posicionamiento

Replica la siguiente figura: <https://cdpn.io/jorgecardoso/debug/dVaKZP>

# Alineamiento



# Alineamiento horizontal

Para alinear texto hemos visto **text-align**

```
#node {  
  text-align: center;  
}
```

# Alineamiento horizontal

Podemos alinear elementos **automáticamente** utilizando

```
#node {  
  margin: 0 auto;  
}
```

En el caso de centrar **imágenes** hay que recordar **display:block**

```
#image-node {  
  display: block;  
  margin: 0 auto;  
}
```

[Ejemplo](#)

# Alineamiento vertical

Podemos alinear elementos **verticalmente** utilizando **padding**

```
#node {  
  padding: 70px 0;  
}
```

[Ejemplo](#)

# Alineamiento vertical

Podemos alinear elementos **verticalmente** utilizando **line-height**

```
#node {  
  height: 200px;  
  line-height: 200px;  
}
```

Excepto en el caso de querer centrar un **texto con múltiples líneas**

[Ejemplo](#)

# Alineamiento vertical

Podemos alinear elementos **horizontal y verticalmente** utilizando **position y transform**

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

Es necesario que el contenedor **padre** tenga **position:relative**

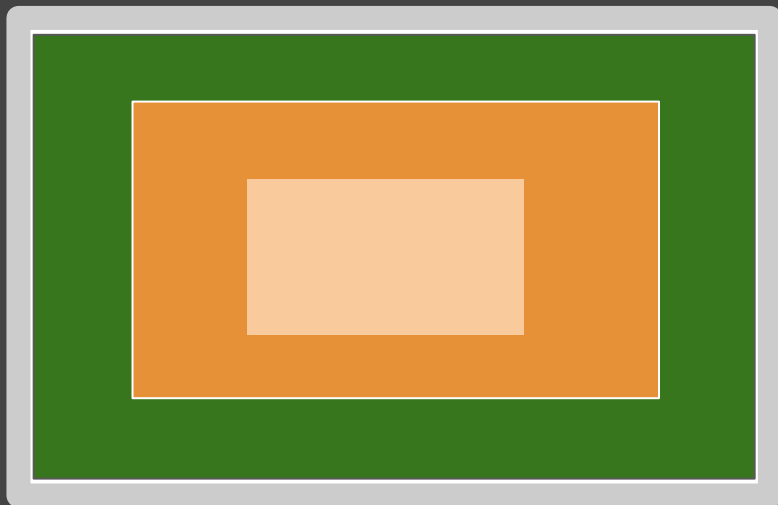
```
#parent-node {  
  position: absolute;  
}
```

[Ejemplo](#)

# Ejercicio Alineamiento

- Replica esta figura con HTML y CSS
  - margin auto para el alineado horizontal
  - Utiliza transform para el alineado vertical

Pantalla



# Tablas

# Tablas

El tag HTML `<table>` nos permite crear tablas

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

Ejemplo



# Tablas

Las tablas contienen tres tipos de tags diferentes:

- **<tr>**: row
- **<th>**: header cell
- **<td>**: normal cell

```
<table>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$80</td>
  </tr>
</table>
```

Month	Savings
January	\$100
February	\$80

# Tablas

Podemos aumentar el número de columnas que ocupa una celda

- Ejemplo border

```
table, th, td {  
  border: 1px solid black;  
  border-collapse: collapse;  
}
```

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

- Ejemplo espaciado

```
th, td {  
  padding: 15px;  
}
```

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

# Tablas

Podemos aumentar el número de columnas que ocupa una celda

```
<tr>  
  <th>Name</th>  
  <th colspan="2">Telephone</th>  
</tr>
```

Name	Telephone	
Bill Gates	55577854	55577855

Podemos aumentar el número de columnas que ocupa una celda

```
<tr>  
  <th rowspan="2">Telephone:</th>  
  <td>55577854</td>  
</tr>
```

Name:	Bill Gates
Telephone:	55577854
	55577855

# Ejercicio tablas

- Reproduce esta tabla utilizando HTML y CSS
- El color azul es: #00cccc

HEADER	HEADER	HEADER	HEADER
Item	Item	Item	Item
Item	Item	Item	Item
Item	Item	Item	Item
Item	Item	Item	Item

# Ejercicio tablas

- Reproduce esta tabla utilizando HTML y CSS

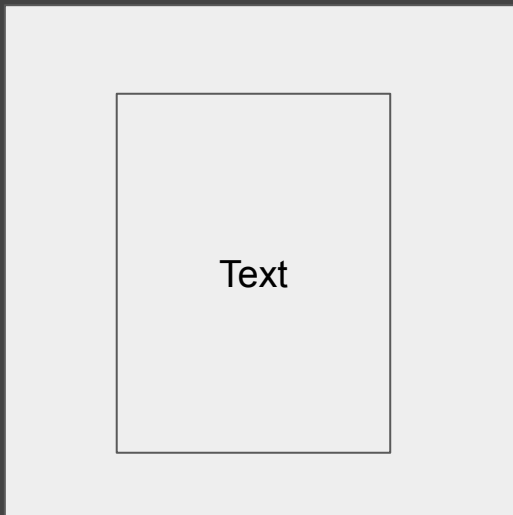
Name / Description	Qty.	@	Cost
Paperclips	1000	0.01	10.00
Staples (box)	100	1.00	100.00
Subtotal			110.00
Tax		8%	8.80
Grand Total			\$ 118.80

# Transform

[Referencia](#)

# Ejercicio transform

- Reproduce la siguiente figura utilizando **transform**
- No utilices margin, padding ni posicionamiento



# Overflow

[Referencia](#)



# Ejercicio overflow

- Crea un contenedor que contenga un párrafo en su interior
- El párrafo debe superar las dimensiones del contenedor
- El contenedor debe permitir utilizar una barra de scroll para leer el contenido oculto

# Semantic tags

[referencia](#)

# Semantic tags

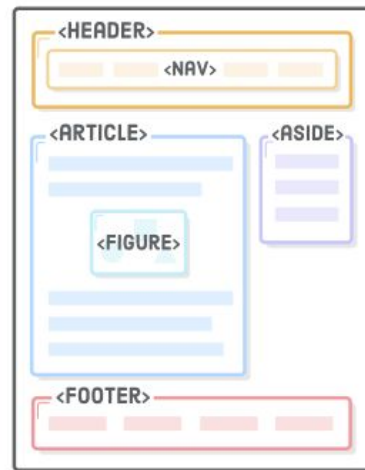
Los tags semánticos se introducen en HTML5 para ayudar a los **motores de búsqueda** a entender la estructura de una página

Mejor comprensión por parte de los motores de búsqueda implica mejor posicionamiento en los buscadores (**SEO**)

Mejor posicionamiento implica más tráfico de entrada

# Semantic tags

La mayoría de páginas tienen una estructura similar

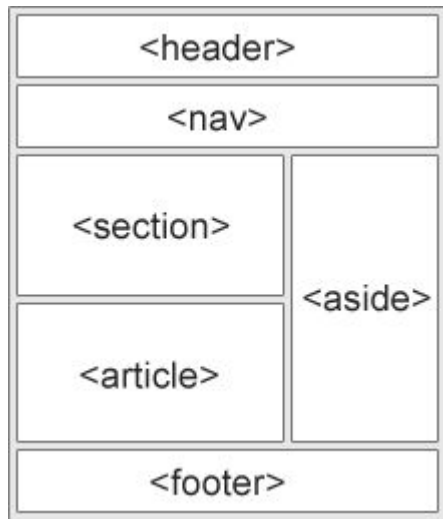


En **HTML4** los desarrolladores maquetaban estos elementos mediante content tags como **<div>**

- Asignaban sus propios ids/clases para identificar estos elementos
- La variedad de nombres posibles confundía a los motores de búsqueda

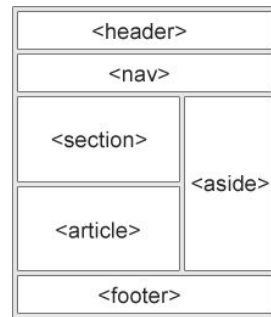
# Semantic tags

Los tags semánticos **se comportan igual que un `<div>`** pero **identifican inequívocamente** las partes de una página



# Semantic tags

- `<header>` Contiene la cabecera de la página (título, intro...)
- `<nav>` Contiene el menú de navegación
- `<main>` Engloba el contenido principal de la página
- `<section>` Separa secciones de contenido
- `<article>` Define posts/artículos. Se le da peso en el SEO
- `<aside>` Define contenido secundario (eg: sidebar)
- `<footer>` Parte inferior donde se define copyright, datos empresa, etc.
- `<figure>` Imagen o gráfico
- `<figcaption>` Explicación de la figura



# Ejercicio semantic tags

- Refactoriza el código de esta página mediante el uso de tags semánticos
  - <https://codepen.io/antonioRedradix/pen/ExxQQRB>

# Selectores avanzados



# Selectores avanzados

Todos los **div** que tengan la clase **lorem**

```
div.lorem{  
  property: value;  
}
```

# Selectores avanzados

Todos los **hijos y nietos** de **#node** que sean **p**

```
#node p{  
  property: value;  
}
```

Todos los **hijos** de **#node** que sean **p**

```
#node > p{  
  property: value;  
}
```

# Selectores avanzados

Todos los **hermanos** de **#node** que sean **p**

```
#node ~ p{  
  property: value;  
}
```

Todos los **hermanos inmediatos** de **#node** que sean **p**

```
#node + p{  
  property: value;  
}
```

# Selectores avanzados

## El primer hijo de #node

```
#node:first-child{  
  property: value;  
}
```

## El último hijo de #node

```
#node:last-child{  
  property: value;  
}
```

# Selectores avanzados

El **hijo** número **n** de **#node**

```
#node:nth-child(n) {  
  property: value;  
}
```

El **hijo** número **n** de **#node** contando desde el final

```
#node:nth-last-child(n) {  
  property: value;  
}
```

# Pseudoclasses

# Pseudoclasses

Las **pseudoclasses** definen **estados especiales** de un elemento

```
#selector:pseudo-class{  
  property: value;  
  property: value;  
}
```

# Pseudoclases

Los nuevos estilos son asignados cuando el nodo **cambia de estado**

```
/* visited link */  
a:visited {  
  color: #00FF00;  
}
```

```
/* mouse over link */  
a:hover {  
  color: #FF00FF;  
}
```

```
/* selected link */  
a:active {  
  color: #0000FF;  
}
```

Ejemplo: [https://www.w3schools.com/css/tryit.asp?filename=trycss\\_link](https://www.w3schools.com/css/tryit.asp?filename=trycss_link)



# Pseudoclases

Los nuevos estilos son asignados cuando el nodo **cambia de estado**

```
/* focused input */  
input: focused {  
  color: #00FF00;  
}
```

Ejemplo: [https://www.w3schools.com/css/tryit.asp?filename=trycss\\_link](https://www.w3schools.com/css/tryit.asp?filename=trycss_link)

# Ejercicio Pseudoclases

- En una misma página añade
  - un div que cambie de color al hacer hover
  - un div que pierda opacidad al hacer hover (utiliza internet)

# Transiciones CSS

# Transiciones

Las **transiciones CSS** nos permiten cambiar valores de propiedades **a través del tiempo**

```
#node{  
  width: 100px  
  transition: width 2s; /* Cualquier cambio futuro en width durará 2 segundos */  
}
```

Ejemplo: [https://www.w3schools.com/css/tryit.asp?filename=trycss3\\_transition1](https://www.w3schools.com/css/tryit.asp?filename=trycss3_transition1)

# Transiciones

Los valores de las propiedades pueden **cambiar en el tiempo** de varias formas:

```
#node{  
  width: 100px  
  transition: width 2s; /* Cualquier cambio futuro en width durará 2 segundos */  
}
```

- Mediante una **pseudoclase**

```
#node:hover{  
  width: 400px  
}
```

# Transiciones

Los valores de las propiedades pueden **cambiar en el tiempo** de varias formas:

```
#node{  
  width: 100px  
  transition: width 2s; /* Cualquier cambio futuro en width durará 2 segundos */  
}
```

- Asignando una **nueva clase** al nodo mediante Javascript

```
.wide{  
  width: 400px  
}
```

```
document.querySelector("#node").classList.add("wide")
```

# Transiciones

Los valores de las propiedades pueden **cambiar en el tiempo** de varias formas:

```
#node{  
  width: 100px  
  transition: width 2s; /* Cualquier cambio futuro en width durará 2 segundos */  
}
```

- Asignando **nuevos estilos** al nodo mediante Javascript

```
node.style.width = "400px"
```

# Transiciones

Las transiciones se construyen a partir de 4 propiedades

```
#node{  
    transition-property: width;  
    transition-duration: 2s;  
    transition-timing-function: linear;    /* Curva animación: ejemplos */  
    transition-delay: 1s;                /* Cuanto tarda en empezar la transición */  
}
```

```
#node{  
    transition: width 2s linear 1s;  
}
```



# Transiciones

Es posible asignar varias transition-properties usando comas

```
#node{  
  transition: width 2s, height 2s, transform 2s;  
}
```

# Ejercicio transiciones

- Crea un div que cambie de color gradualmente asignando una nueva clase con Javascript

# Ejercicio transiciones II

- Crea un div que
  - pierda opacidad gradualmente hasta desaparecer on mouseover
  - Recupere opacidad gradualmente on mouseout

# Ejercicio Transiciones III

- Crea un div que parpadee (fadeIn, fadeOut) constantemente utilizando **Javascript**

# Background

# Background

Hemos visto **background-color** para cambiar el color de fondo

```
#node {  
  background-color: red;  
}
```

También podemos utilizar imágenes como fondo

```
#node {  
  background-image: url("img_file.jpg");  
}
```

# Background

Disponemos de varias propiedades para configurar el posicionamiento de la imagen de fondo

```
body {  
  background-image: url("img_tree.png");  
  background-repeat: no-repeat;  
  background-position: right top;  
  background-attachment: scroll;  
}
```

[Referencia](#)