

DISEÑO DE INTERFACES WEB

U.D. 2 (parte 1/2): HTML



Página Web

- Es un **documento HTML** (documento de texto con marcas) interpretado por los navegadores en forma gráfica, permitiendo también el acceso al código.
- Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página Web, como imágenes, listas, vídeos, etc.

¿Qué es HTML?

- HTML: *HyperText Markup Language* → Lenguaje de marcado de hipertexto:
 - **Marcado:** La información está codificada mediante marcas o etiquetas.
 - **Hipertexto:** Permite presentar el texto y la información de forma completamente estructurada.
- Es un **lenguaje de marcado**, no de programación, ya que carece de instrucciones de control de flujo, variables, operadores, funciones, etc.
- Creado por la **World Wide Web Consortium**, fundación que vela por la estandarización y las buenas prácticas de la World Wide Web.

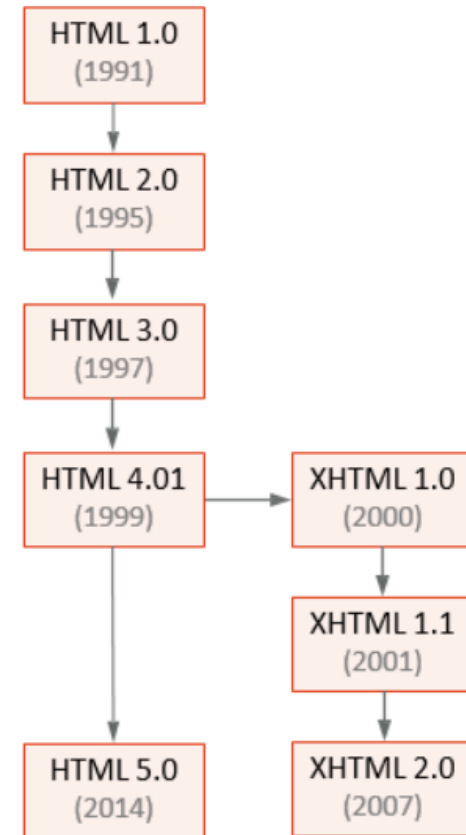
Historia HTML

- Surgió por la complejidad del lenguaje *Standard Generalized Markup Language* (SGML) creándose uno más simple y adaptado expresamente al cometido de representar contenido para la Web.
- El número de etiquetas del que se dotó a HTML era considerablemente reducido, lo que hacía que su curva de aprendizaje fuera bastante más rápida.

<html> <head> <body>	<title> <meta> <link>
<style> <script>	<p> <h1>-<h6>
<a>	 <hr>
 	<table> <tr> <td> <th>
	<iframe> <embed> <object>
<button>	<form> <label> <input>
<div>	<header> <section> <nav>

Historia HTML

- HTML 1.0 (1991): No fue considerado estándar.
- HTML 2.0 (1995): Estándar.
- HTML 3.2 (1997): Primera versión publicada bajo la recomendación de W3C.
- HTML 4.01 (1999): Versión estable y ampliamente utilizada.
- XHTML 1.0 (2000): Usa como estándar la especificación del estándar *eXtensible Markup Language* (XML). Más restrictivo.
- HTML 5.0 (2014): Se realizó por la *Web Hypertext Application Technology Working Group* (WHATWG) posteriormente apoyado por el W3C.



XML, XHTML y HTML

- XHTML es un documento HTML que cumple especificaciones XML.
- Un documento XHTML puede ser procesado por un procesador XML.
- Grosso modo, se podría considerar a HTML como un XML/XHTML "relajado".
- Hoy día se prefiere HTML5 a XHTML.



Documento HTML: Estructura básica

```
<!DOCTYPE html>
```

Instrucción al navegador sobre el tipo de documento

```
<html lang="es-ES">
```

- **<html>**: elemento raíz
- **lang** (opcional): indica el idioma (recomendable)

```
<head>
```

```
  <!-- Metadatos del documento HTML -->
```

```
  <title>Ejemplo 1</title>
```

- **<head>**: proporciona información al navegador sobre el documento (metadatos)
- **<title>**: título del documento

```
</head>
```

```
<body>
```

```
  <!-- Contenido del documento HTML -->
```

```
  Primer documento HTML
```

- **<body>**: cuerpo del fichero, donde se indica la información principal

```
</body>
```

```
</html>
```

Cierre documento

- Declaración: Versión de HTML con la que se ha creado.
- Cabecera: Información acerca del documento u otros datos que no pueden considerarse parte de su contenido.
- Cuerpo: Contenido real del documento.

Documento HTML: Estructura básica

- Para que los navegadores Web sepan qué versión de HTML es el documento que debe presentar (versión en la que está escrita la página), es necesario incluir una cabecera `DOCTYPE` que lo identifique:

```
<!DOCTYPE html>
```

- Asegura que la página sea analizada de la misma forma por diferentes navegadores.
- En HTML5 ya no se requiere una referencia a los *Definición de Tipo de Documento* (DTD) porque ya no está basado en SGML.

Documento HTML: Estructura básica

- En primer lugar, se encuentra el par de etiquetas `<html></html>` que contiene todo el resto del documento.
- Dentro de este par se pueden encontrar los dos siguientes elementos con sus pares de etiquetas:
 - `<head></head>`: Contiene **información** relativa a la página Web, como puede ser el título, metadatos, estilos, etc.
 - `<body></body>`: Contiene la información que se desea presentar, se presenta el **contenido** del propio documento.

Documento HTML: Estructura básica

- Ejemplo:

```
<html>
  <head>
    <title>Página ejemplo HTML</title>
  </head>
  <body>
    <p>Primer párrafo de ejemplo</p>
    <p>Segundo párrafo de ejemplo</p>
    
  </body>
</html>
```

Elementos HTML

- `<head>` puede contener distintos tipos de elementos, los más importantes son:
 - `<title>`: Título del documento. Imprescindible incluirlo.
 - `<base>`: Especifica la dirección URL base que se utilizará para todas las direcciones URL contenidas dentro de un documento. Si no se indica, se refiere al directorio actual.
 - `<meta>`: Aporta información sobre el documento:
 - `charset`: Codificación de caracteres. Es importante para especificar el juego de caracteres utilizados. Ejemplo: `utf-8`.
 - `description`: Describe el contenido de la página Web.
 - `keywords`: Palabras clave.
 - `author, copyright`: Autor de la página y copyright.
 - `robots`: Indexación de los robots. Tipos: `"index"`, `"noindex"`, `"follow"`, `"nofollow"`.
 - `http-equiv`: Información equivalente a las indicaciones en el encabezado HTTP. Tipos: `"cache-control"`, `"expires"`, `"refresh"`.

Elementos HTML

- `<body>` los más importantes son:
 - `<h1></h1>`: Define un **encabezado** de tipo 1. Se pueden usar desde hasta `<h6>` para representar un árbol de encabezados siendo los de tipo 1 los de mayor nivel.
 - `<p></p>`: Para representar un **párrafo** de texto.
 - `Enlace`: Permite definir un **hipervínculo** a una URL o enlazar a un email. El contenido que aparece entre `<a>` y `` será el que se represente en la página Web.
 - ``: Permite incorporar una **imagen** en el contenido que se va a representar. El atributo `src` sirve para indicar la URL donde se encuentra la imagen, `alt` permite definir un texto alternativo a la imagen que se mostrará en navegadores sin soporte gráfico, `width` y `height` para representarla con un determinado tamaño. No tiene etiqueta de cierre (sí de autocierre).
 - `<figure>` y `<figcaption>`: Pies de figura.

Elementos HTML

- `
`: Inserta un **salto de línea**. No se cierra.
- `<hr>`: Inserta una **línea horizontal**. No se cierra.
- `` y `<i></i>`: Inserta **texto en negrita** y en **cursiva** respectivamente.
 - `Texto en negrita`
 - `<i>Texto en cursiva</i>`
- ``: Define **listas no numeradas** en las que cada nuevo ítem se encuentra delimitado por ``. `` ordenadas.
 - ``
 - `Item 1`
 - `Item 2`
 - ``
- `<div>`: **División**. Crea secciones o agrupa contenidos.
- ``: **Contenedor en línea**. Aplica estilo al texto sin saltos.

Elementos HTML

- Anotación de contenido:
 - `` y ``: Negrita.
 - `<i>` y ``: Cursiva.
 - `<u>`: Subrayado (desaconsejado con confusión con enlaces).
 - `<s>`: Tachado.
 - `<small>`: Fuente de menor tamaño.
 - `<sub>` y `<sup>`: Subíndice/Superíndice.
 - `<mark>`: Resaltado. Representa un texto marcado o resaltado como referencia o anotación.
 - `` y `<ins>`: Borrado/Insertado. Marca las partes de un texto que han sido suprimidas/añadidas del/al documento.
 - `<pre>`: Texto preformateado (se respetan los espacios).

Elementos HTML

■ Abreviaturas y citas:

- `<abbr>`: Abreviatura. El atributo `title` indica expansión con ratón (`hover`).
- `<dfn>`: Concepto que se va a definir (`title` el mismo nombre).
- `<q>`: Referencia (quotation).
- `<blockquote>`: Referencia, similar a la anterior pero con tamaños mayores de texto.
- `<cite>`: Título de un trabajo (libro, artículo, película, etc.).

Elementos HTML

- Ejercicio: Crea una página Web que contenga TODOS los elementos (etiquetas) HTML vistos hasta ahora.

Elementos HTML

- `<table></table>`: Escribe una **tabla** en el contenido que se desea representar. Antes se usaban para maquetar lo que hoy se desaconseja, `<th></th>` cabecera, `<tr></tr>` para empezar nueva fila, `<td></td>` para delimitar nueva columna, `<rowspan>` filas que ocupa un dato y `<colspan>` columnas.

```
<table border="1">
  <tr>
    <td>Fila 1 Columna 1</td>
    <td>Fila 1 Columna 2</td>
  </tr>
  <tr>
    <td>Fila 2 Columna 1</td>
    <td>Fila 2 Columna 2</td>
  </tr>
</table>
```

Elementos HTML

- `<form></form>`: Crea un **formulario** usando las etiquetas `<input>`, `<option>`, `<textarea>`, `<button>`, etc. Los atributos `action` y `method` permiten definir la URL a la que se enviarán los datos del formulario y el método de envío (get o post).

```
<form action="Ejemplo-Tabla.html" method="get">
  Apellidos: <input type="text" name="apellidos"><br>
  Nombre: <input type="text" name="nombre"><br>
  Comentario: <textarea name="comentario">Escriba su
    comentario</textarea><br>
  <input type="submit" value="Enviar">
</form>
```

Elementos HTML

- `<iframe>`: Representa un contexto de navegación anidado (inline frame) que permite por ejemplo incrustar otra página HTML en la actual, un vídeo, etc.

`<p>Lo siguiente es una página Web embebida:</p>`

```
<iframe src="Ejemplo-Tabla.html" height="200"
width="300" title="Ejemplo iframe"></iframe>
```

- `<embed>/<object>`: Punto de integración para una aplicación externa, como una página Web, una imagen, un documento, etc.

```
<embed type="image/jpg" src="imagen.jpg" width="470"
height="350">
```

```
<object data="Curso HTML5.pdf" height="350"
type="application/pdf"></object>
```

Elementos HTML

- Un elemento HTML tiene en general 3 partes:

1. Etiqueta inicial

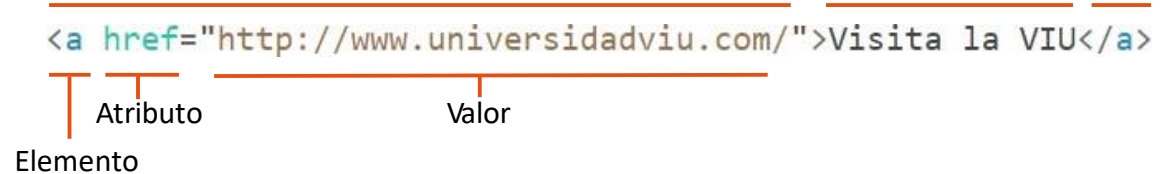
ETIQUETA INICIAL

2. Contenido

CONTENIDO

ETIQUETA
DE CIERRE

3. Etiqueta de cierre



- Algunos elementos no tienen etiqueta de cierre ni contenido (img, input, button, etc.)

```
      <hr style="border:2px;">
```

- El elemento **comentario** tiene la siguiente estructura:

```
<!-- Esto es un comentario -->
```

Elementos HTML

- La **anidación** de etiquetas es inherente al lenguaje HTML.
- Dentro una instrucción HTML se puede incluir cualquier otra instrucción, generando una anidación de etiquetas.
- La única regla a seguir consiste en incluir las etiquetas finales (si son necesarias) en orden inverso al de las etiquetas iniciales:

- Anidación correcta: `<eti1>... <eti2>... </eti2>... </eti1>` ✓
- Anidación incorrecta: `<eti1>... <eti2>... </eti1>... </eti2>` ✗

- Una anidación incorrecta genera resultados inesperados o erróneos en el navegador.

Atributos de un elemento HTML

- Los elementos pueden llevar asociadas propiedades llamadas **atributos** que permiten su configuración.

Esta es la `página web de la VIU`

- Los atributos solamente se aplican en etiquetas de apertura, no en las de cierre.
- Va el nombre del atributo, seguido del signo = y de su valor que se indica en comillas dobles (" ") o simples (' ').
- Una etiqueta puede tener más de un atributo.

``

- Existen atributos booleanos.

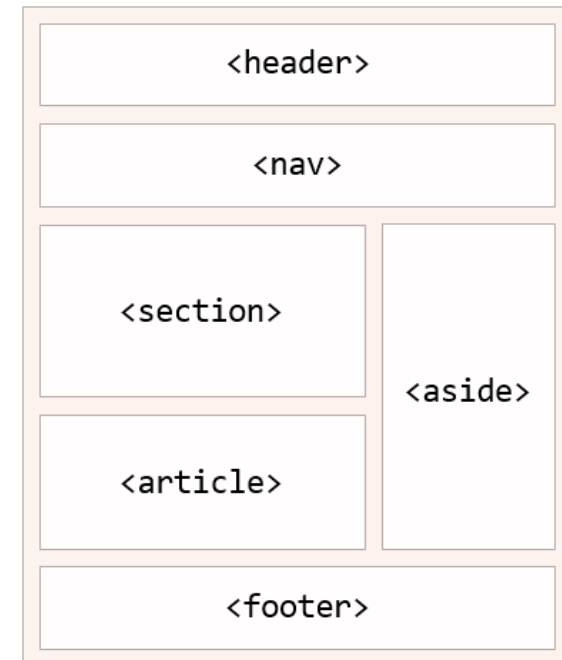
`<input disabled>`

Atributos de un elemento HTML

- `id`: Identificador **único** para cada elemento.
- `class`: Especifica la clase o clases a las que pertenece el elemento en cuestión. Puede ser utilizado en más de un elemento. Se utiliza mucho al aplicar un estilo CSS.
- `style`: Especifica el estilo CSS que se aplicará al elemento.
- `title`: Se usa para mostrar información extra acerca del elemento que lo contiene, información que aparece como un pequeño mensaje emergente al situar el ratón sobre el elemento (en desuso por temas de accesibilidad).
- Todos estos elementos son soportados por todas las etiquetas salvo por: `<head>`, `<html>`, `<meta>`, `<script>`, `<style>` y `<title>`

Maquetación HTML: Estructura

- `<header>`: Logo o cabecera de la página. Típicamente contendrá elementos `<h1>`-`<h6>`.
- `<footer>`: Sección con información resumida de la sección y puede incluir detalles del autor, derechos, enlaces, logos, etc.
- `<section>`: Sección genérica del documento.
- `<nav>`: Destinada a contener enlaces a otras páginas u otras partes del documento (menús).
- `<article>`: Pieza autocontenida de información en un documento HTML.
- `<aside>`: Sección con contenido indirectamente relacionado con el contenido principal del documento (barra lateral).
- `<details><summary>`: Puede expandirse para obtener más detalles adicionales.



Ejercicios

- Realiza todos los ejercicios del bloque 'Ejercicios 2.1'.
- Realiza todos los ejercicios del bloque 'Ejercicios 2.2'.

Modelo de objetos del documento

- **DOM:** Es un API (Application Programming Interface) estándar del W3C para documentos HTML y XML.
 - Proporciona una representación estructural del documento que permite su presentación visual o la modificación de su contenido.
 - Esencialmente, comunica las páginas Web con los scripts o los lenguajes de programación.
 - Describe el contenido del documento como un conjunto de objetos para que un programa JavaScript pueda actuar sobre ellos.

Modelo de objetos del documento

- Se eligió el nombre modelo de objetos del documento porque es un modelo de objetos en el sentido tradicional del diseño orientado a objetos: Los documentos se modelizan usando objetos, y el modelo comprende no solamente la estructura de un documento, sino también el comportamiento de un documento y de los objetos de los cuales se compone.
- Como modelo de objetos, el DOM identifica:
 - Las interfaces y objetos usados para representar y manipular un documento.
 - La semántica de estas interfaces y objetos, incluyendo comportamiento y atributos.
 - Las relaciones y colaboraciones entre estas interfaces y objetos.
- Consiste actualmente de dos partes: el núcleo del DOM y el DOM HTML. El núcleo del DOM representa la funcionalidad usada para los documentos XML, y también sirve de base para el DOM HTML.

Modelo de objetos del documento

```
<body>
  <p>Esto es un párrafo que contiene <a href="#">un
    enlace</a> en el medio.</p>
  <ul><li>Primera entrada en la lista</li>
    <li>Segunda entrada en la lista</li></ul>
</body>
```

- El elemento `<a>` se encuentra localizado dentro del elemento `<p>` del HTML, convirtiéndose en un nodo hijo (o simplemente hijo del nodo `p`), de manera similar, `p` es el nodo padre de `a`. Los dos nodos `li` son hijos del mismo padre `ul`, llamándose nodos hermanos.
- Es importante comprender la diferencia entre elementos y nodos de textos. Los elementos normalmente están asociados a las etiquetas. En HTML todas las etiquetas son elementos, tales como `<p>` por lo que tienen atributos y contienen nodos hijos. Sin embargo, los nodos de textos no poseen atributos ni hijos.

DISEÑO DE INTERFACES WEB

**U.D. 2 (parte 2/2):
CSS**

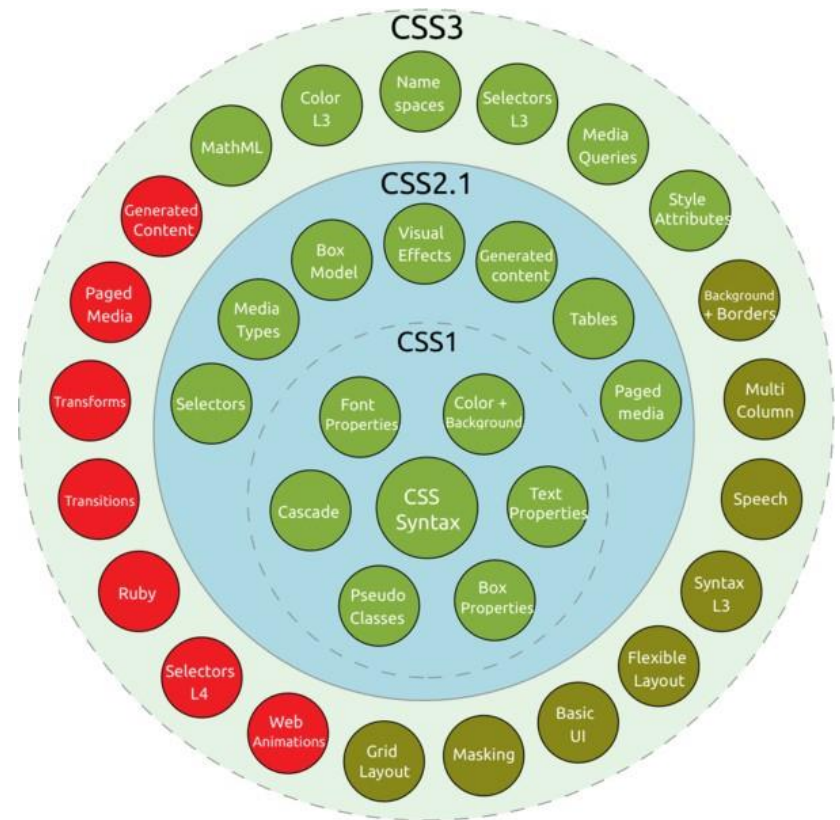


¿Qué es CSS?

- CSS: *Cascading Style Sheets* → Hojas de estilo en cascada:
 - **Estilo:** Define la apariencia del documento. Describe cómo se debe mostrar cada uno de los elementos de un documento HTML, pudiendo realizar agrupaciones por tipos.
 - **Cascada:** Son en cascada ya que se pueden incluir diversas hojas de estilo pudiendo modificar las nuevas definidas en hojas previas. El orden de las reglas importa: Cuando dos tienen la misma especificidad, se aplica a la que aparece en último lugar.
- Es un **lenguaje para dar apariencia** o estilo a un documento HTML.
- Creado por la W3C.

Historia CSS

- HTML no estaba pensado para dar formato o estilo a un documento, los elementos solo se pensaron para darle estructura.
- CSS 1.0 (1996): Primera recomendación W3C.
- CSS 2.0 (1998): Especificación única.
- CSS 3.0 (1999, actual 2011): A diferencia de la anterior, ésta está dividida en varios documentos separados, llamados “módulos”. Por ejemplo: “selectores”, “espacios de nombres” o “color”.



Sintaxis CSS

- Reglas de estilo, cada una consta de dos partes:
 - **Selector** o etiqueta para indicar qué elementos o partes de una página se van a ver afectados por el estilo.
 - **Estilo** propiamente dicho:
 - Una o más declaraciones (separadas por punto y coma).
 - Declaración compuesta por la propiedad o atributo (hace referencia a la característica que se quiere modificar) y su valor (separadas por dos puntos).
- selector1, selector2 { propiedad1: valor1; propiedad2: valor2 }



Sintaxis CSS

- Ejemplo:

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

Indica el color de fondo de toda la página

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

Indica que todos los elementos <h1> del documento tienen letra de color blanco y alineada al centro

```
p {  
  font-family: verdana;  
  font-size: 14px;  
}
```

Indica que todos los elementos <p> del documento tienen una fuente de tipo 'verdana' con tamaño 14 px

Incorporación del estilo

- En el mismo documento HTML (estilo interno o embebido):
 - Uso de unas etiquetas predefinidas para marcar el texto (`<style>` y `</style>`).
 - Dentro de la cabecera del documento HTML.
 - Se puede incluir cualquier número de elementos `<style>`.

Incorporación del estilo

- En el mismo documento HTML - Ejemplo:

```
<head>
  <style>
    a { background-color: grey; color: white }
    span { border: thin black solid; padding: 10px }
  </style>
</head>
<body>
  <p>Párrafo de <span>inicio</span></p>
  Visita la <a
    href="http://es.wikipedia.org">Wikipedia</a>
</body>
```

Incorporación del estilo

- En un archivo externo (estilo externo):
 - Las mismas hojas de estilo que se incluyen dentro del elemento `<link>`, en la cabecera del documento, pueden almacenarse en un fichero externo con extensión “.css”.
 - La forma de acceder y enlazar esos ficheros “.css” con el documento HTML/XHTML es a través del atributo `href`.

- En la misma carpeta que el fichero HTML



`href="estilo.css"`

- En la carpeta padre del fichero HTML (no habitual)



`href="../estilo.css"`

- En la carpeta hija del fichero HTML



`href="styles/estilo.css"`

Incorporación del estilo

- En un archivo externo - Ejemplo:

Archivo estilo.css:

```
a { background-color: grey; color: white }  
span { border: thin black solid; padding: 10px }
```

Archivo ejemplo.html:

```
<head>  
  <title>Ejemplo de estilo externo</title>  
  <link rel="stylesheet" type="text/css"  
    href="estilo.css" />  
</head>  
<body>  
  <p>Párrafo de <span>inicio</span></p>  
  Visita la <a  
    href="http://es.wikipedia.org">Wikipedia</a>  
</body>
```

Incorporación del estilo

- En atributos de elementos HTML (estilo en línea o inline):
 - Consiste en insertar fragmentos de CSS dentro de atributos de etiquetas HTML de la página.
 - Las reglas de estilo adoptan la forma “propiedad:valor” y van separadas por un punto y coma. Todo ello se encierra entre comillas como con cualquier otro atributo.
 - Principal desventaja: el mantenimiento y modificación del código puede resultar más complicado.

Incorporación del estilo

- En atributos de elementos HTML - Ejemplo:

```
<body>
```

```
  <p>Párrafo de <span style="border: thin black  
    solid; padding: 10px">inicio</span></p>
```

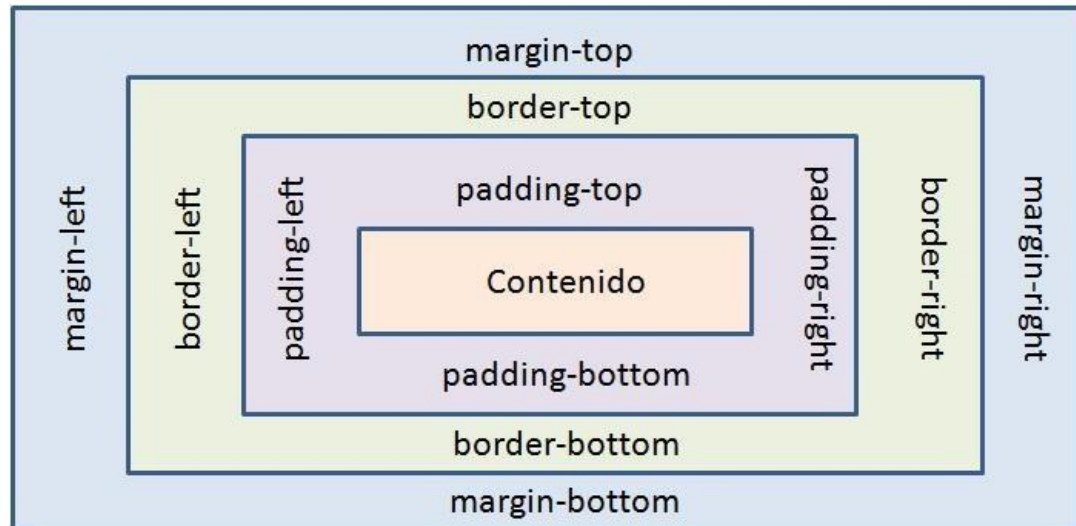
```
  Visita la <a style="background-color: grey;  
    color: white"
```

```
    href="http://es.wikipedia.org">Wikipedia</a>
```

```
</body>
```

Modelo de cajas

- La potencia de CSS radica en la gran cantidad de atributos que se pueden usar para dar estilo.
- Hay que pensar en cualquier elemento como una caja. Una página Web es una caja de cajas.



Modelo de cajas

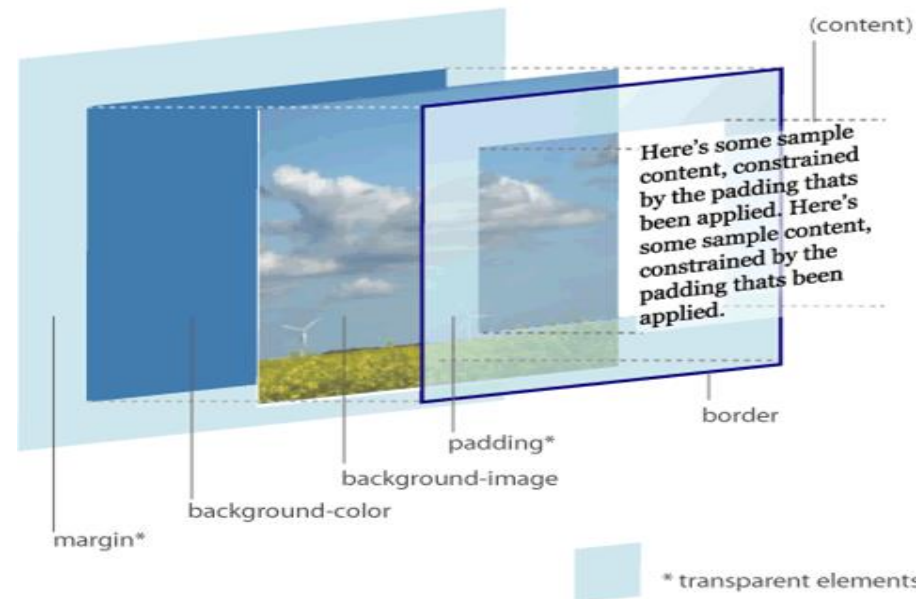
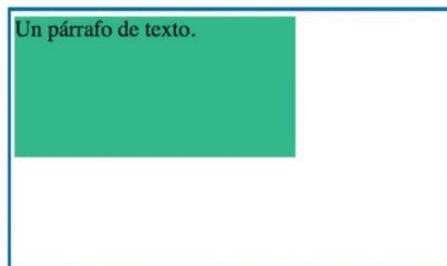
Las propiedades principales con las que se trabaja en el modelo de cajas son *width* y *height*, relativas a las dimensiones de la *caja de contenido* o *content*; se trata del ancho y el alto del área donde se muestra el contenido de la caja.

Otra de las propiedades más importantes del área de contenido es el fondo de la caja. Se diferencia entre *background image* (imagen de fondo), que corresponde a la imagen que se muestra por detrás del contenido y el espacio de relleno, y *background color* (color de fondo), que es el color que se muestra por detrás del contenido y el espacio de relleno.

- Por ejemplo, si se introduce este código:

```
article {  
  background: #5DEAAB;  
  height: 100px;  
  width: 200px;  
}
```

- En pantalla, se verá:



Modelo de cajas

- Ejercicio: En el ejemplo anterior, cambia el ancho de la caja `<article>` a 300px y el alto a 200px.
- Ejercicio: Aplica un color de fondo rojo a toda una página, un color de fondo azul claro a un encabezado `<h1>` y otro amarillo a varios párrafos `<p>`.
- Ejercicios de **fondo**:
https://www.w3schools.com/css/exercise.asp?filename=exercise_background1

Modelo de cajas

- Atributos que afectan a las **dimensiones**:
 - Unidades **absolutas**: Centímetros (cm), Milímetros (mm), Pulgadas (In, 2'54 cm), Puntos (pt, 1/72 de In) y Picas (pc, 12 pt).
 - Unidades **relativas**: Píxeles (px), em (altura font-size de la letra del elemento en el que se usa en puntos) y ex (altura de la primera letra minúscula, aprox 0'5em).
 - Para no depender de la resolución de pantalla se recomiendan las relativas, en concreto em.
 - Los porcentajes % son muy usados.

```
p { font-size: 16px } /* 1em = 16px */
```

Modelo de cajas

- Atributos que afectan a la **posición**:
 - `top`: Desde arriba. Distancia en vertical donde se colocará la capa.
 - `bottom`: Desde abajo. Idem a la anterior.
 - `left`: Desde la izquierda. Distancia en horizontal donde se colocará la capa.
 - `right`: Desde la derecha. Idem a la anterior.

Por otro lado, si el atributo `position` es *absolute* (o *fixed*), `top` indicará la distancia del borde superior de la capa con respecto al borde superior de la página. Si es *relative*, indicará la distancia desde donde se estaba escribiendo en ese momento en la página hasta el borde superior de la capa.

Modelo de cajas

- Atributos que afectan al **relleno**:

- o padding-left, padding-right, padding-top y padding-bottom.

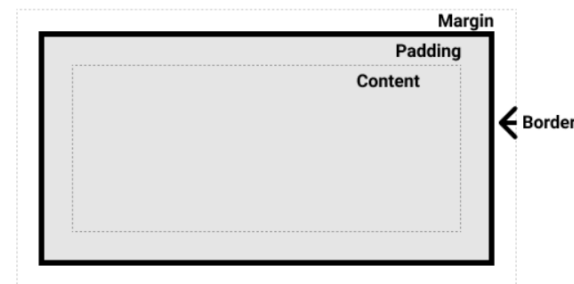
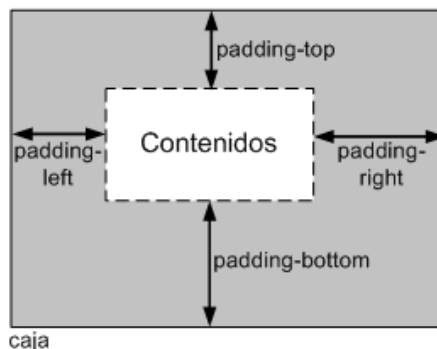
```
<head><style>
  body { margin: 100px 100px 100px 100px;
    border: 5px dotted blue }
  div { margin: 10px 10px 10px 10px;
    padding: 30px 30px 30px 30px;
    border: 20px dotted red } </style></head>
<body><div style="font-size:0.5in">Texto de 0'5
  pulgadas</div></body>
```

Modelo de cajas

La segunda área es la de relleno, llamada *padding*. Es el espacio que existe entre la caja de contenido y el borde. Las características de esta caja que pueden modificarse son su altura y anchura. Se distinguen, por tanto, cuatro zonas posibles de acción: zona inferior, zona superior, zona izquierda y zona derecha. El nombre de las propiedades que definen estas zonas son *padding-top*, *padding-right*, *padding-left* y *padding-bottom*.

- Ejemplo:

```
article {  
  padding-top: 10px;  
  padding-right: 30px;  
  padding-bottom: 10px;  
  padding-left: 20px;  
}
```



Existen más tipos de empleo de esta regla que simplifican su uso, que se basan en el empleo de la palabra *padding* seguida de varias cifras separadas sin comas. Si aparecen dos valores, el primero es asignado al límite superior y al inferior, mientras que la segunda cifra será la que se aplique al relleno horizontal. En el caso de tener tres valores, el primero y el último serán los aplicados a los límites superior e inferior, respectivamente, y la segunda cifra indicará el valor de los bordes izquierdo y derecho.

Modelo de cajas

- Ejercicio: A una caja de tipo `<div>` (incluye un texto), de color de fondo “silver” y anchura 400px:
 - Aplica el mismo formato de relleno (padding) de 50px a todos los extremos.
 - A continuación, aplica un formato de padding superior e inferior de 70px y horizontal de 30px.
 - Por último, un padding superior de 5px, horizontal de 50px e inferior de 70px.
- Ejercicios de **relleno**:

https://www.w3schools.com/css/exercise.asp?filename=exercise_padding1

Modelo de cajas

- Atributos del **contenido**:
 - `width`: Distancia entre el límite de `padding-left` y `padding-right`.
 - `height`: Altura, entre `padding-bottom` y `padding-top`.

Modelo de cajas

- Atributos que definen el estilo y color del **borde**:
 - `border-top`, `border-bottom`, `border-right` y `border-left`, `border-radius` (**redondeo**), `border-width` (**anchura**), etc.
 - **Palabras clave**: `none` (ninguno), `solid` (continuo), `dashed` (discontinuo), `dotted` (punteado), `double` (doble), `groove` (biselado), `ridge` (biselado), `inset` (hundido), `outset` (saliente), `hidden` (oculto), `inherit` (heredado), `thin` (fino), `medium` (medio), `thick` (grueso), etc.

Modelo de cajas

La siguiente área que puede verse en el diagrama de cajas es la correspondiente al borde, llamado *border*. Esta zona es la que encierra el contenido y el relleno, *content* y *padding*. Es posible modificar su grosor, estilo y color; además, al igual que ocurre con *padding*, es posible dar formato a los cuatro extremos de la caja a la vez utilizando solo la palabra *border* o hacerlo de forma individual.

- Ejemplo:

```
article {  
    border: 1px solid #000000;  
}
```

<i>border-top</i> <i>border-right</i> <i>border-bottom</i> <i>border-left</i>	Modifican a la vez el grosor, el estilo y el color de cada lado del borde.
<i>border-width</i> <i>border-style</i> <i>border-color</i>	Modifican de forma individual el grosor, el estilo y el color de los cuatros extremos a la vez.
<i>border-top-width</i>	Modifica el grosor del borde superior.
<i>border-top-style</i>	Modifica el estilo del borde superior.
<i>border-top-color</i>	Modifica el color del borde superior

Modelo de cajas

- Ejercicio: A una caja de tipo `<div>` (incluye un texto), de color de fondo “lightblue”, anchura 300px y relleno (padding) 15px, aplica un borde de estilo y color “solid navy” y tamaño 25px.
- Ejercicios de **borde**:
https://www.w3schools.com/css/exercise.asp?filename=exercise_border1

Modelo de cajas

- Atributos que afectan al **margen**:

- o margin-left, margin-right, margin-top y margin-bottom.

```
<head><style>
```

```
  body { margin-top:100px; margin-bottom:100px;  
        margin-right:100px; margin-left:100px;  
        border:5px dotted blue }
```

```
  div { margin-top:10px; margin-bottom:10px;  
        margin-right:10px; margin-left:10px; border:2px  
        dotted red } </style></head>
```

```
<body><div style="font-size:0.5in">Texto de 0'5  
  pulgadas</div></body>
```

Modelo de cajas

Finalmente, el margen que envuelve al resto de elementos CSS es el denominado *margin*, y sostiene a otras cajas del diseño. Las propiedades individuales son *margin-top*, *margin-right*, *margin-bottom* y *margin-left*. El funcionamiento es el mismo que el de la propiedad *padding*.

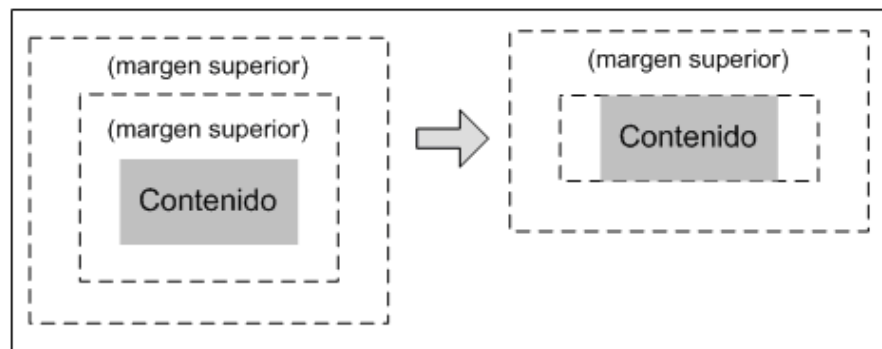
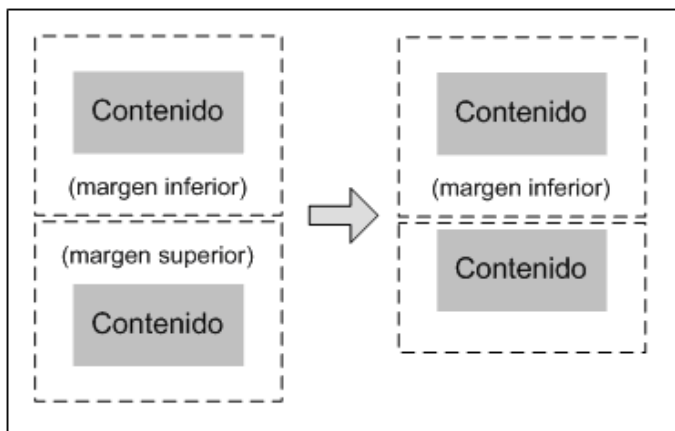
- Ejemplo:

```
article {  
    margin-top: 10px;  
    margin-right: 10px;  
    margin-bottom: 10px;  
    margin-left: 10px;  
}
```

Los márgenes tienen un comportamiento peculiar llamado *margin collapsing*. Cuando dos cajas se tocan, la distancia entre ellas es el valor del margen más grande, y no la suma de ambos.

Modelo de cajas

- Cuando se juntan dos o más márgenes verticales, se fusionan de forma automática y la altura del nuevo margen será igual a la altura del margen más alto de los que se han fusionado.
- Si un elemento está contenido dentro de otro, sus márgenes verticales se fusionan y resultan en un nuevo margen de la misma altura que el mayor margen de los que se han fusionado.



Modelo de cajas

- Ejercicio: A una caja tipo `<div>` (incluye un texto), de color de fondo “lightblue”, anchura 300px y relleno (padding) 15px, borde de estilo y color “solid navy” y tamaño 25px, aplica un margen izquierdo de 25px.
- Ejercicios de **margen**:
https://www.w3schools.com/css/exercise.asp?filename=exercise_margin1

Modelo de cajas

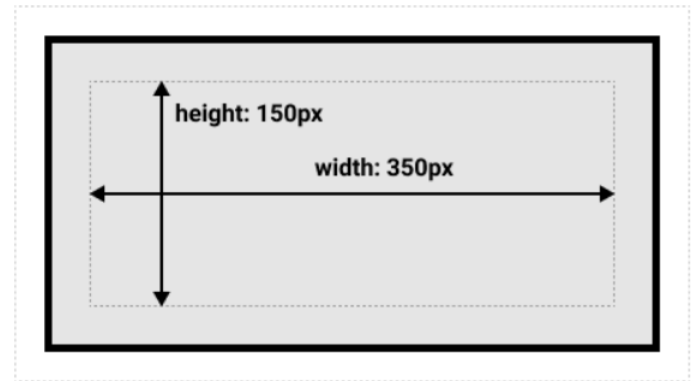
- **Modelo de cajas estándar:**
 - Por defecto, es el que usan los navegadores.
 - Cuando se establecen los atributos `width` y `height` para una caja, se define el ancho y el alto del contenido de la misma. Cualquier área de relleno y borde se añade a ese ancho y alto para obtener el tamaño total que ocupa la caja.
- **Más atributos y Modelo de cajas alternativo:**

https://developer.mozilla.org/es/docs/Learn/CSS/Building_blocks/The_box_model

Modelo de cajas

■ Modelo de cajas estándar - Ejemplo:

```
.box { width: 350px;  
      height: 150px;  
      margin: 10px;  
      padding: 25px;  
      border: 5px solid black }
```



- El ancho que ocupará en realidad la caja usando el modelo de cajas estándar será de **410px** = $350 + 25 + 25 + 5 + 5$, y su altura de **210px** = $150 + 25 + 25 + 5 + 5$, ya que el área de relleno y el borde se añaden al ancho que se utiliza para el contenido de la caja.
- El margen no se cuenta para el tamaño real de la caja. Por supuesto afecta al espacio total que la caja ocupa en la página, pero solo al espacio de fuera de la caja. El área de la caja se termina en el borde, no se extiende hasta el margen.

Ejercicios

- Realiza todos los ejercicios del bloque 'Ejercicios 2.5'.

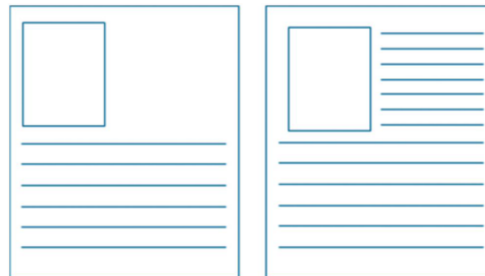
Cajas flotantes

- Debe tenerse en cuenta la posición de cada una de las cajas. Normalmente una se sitúa a continuación de otra, ya sea en horizontal o en vertical. En ocasiones, es deseable que cajas contenedoras de información (texto, imagen, etc.) se adapten entre sí.
- `float`: La caja se coloca por ejemplo a la izquierda y el resto a su derecha. Su valor por defecto es *none*. El efecto sigue hasta que lo anula otra caja con `clear`.

```
etiqueta { float: left | right | none | ... }
```

Cajas flotantes


- La existencia de cajas flotantes condiciona la situación del resto de cajas. Es debido a que aquellos elementos no flotantes (los que están en línea), adaptan su posición para evitar que se solapen las cajas en función a la colocación de las flotantes.
- El funcionamiento concreto de esta propiedad se basa en el desplazamiento total hacia la derecha o izquierda de una caja contenedora respecto de su posición inicial.



Cajas flotantes

- Ejercicio: En una página Web con una imagen y un párrafo de mínimo 10 líneas:
 - Sitúa la imagen flotando a la derecha.
 - Sitúa la imagen sin flotar.
 - Sitúa la imagen a la izquierda separando el texto de la siguiente manera:

Float



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

Cajas flotantes

- `clear`: Sirve para evitar que una capa se posicione a cualquiera de sus lados.
 - `left`: no deja que una capa flote a la izquierda.
 - `right`: a la derecha.
 - `both`: en cualquiera de sus lados.

```
etiqueta { clear: left | right | both | ... }
```

Cajas flotantes

- Ejemplo: https://www.w3schools.com/css/tryit.asp?filename=trycss_layout_clear

Without clear

div1

div2 - Notice that div2 is after div1 in the HTML code. However, since div1 floats to the left, the text in div2 flows around div1.

With clear

div3

div4 - Here, clear: left; moves div4 down below the floating div3. The value "left" clears elements floated to the left. You can also clear "right" and "both".

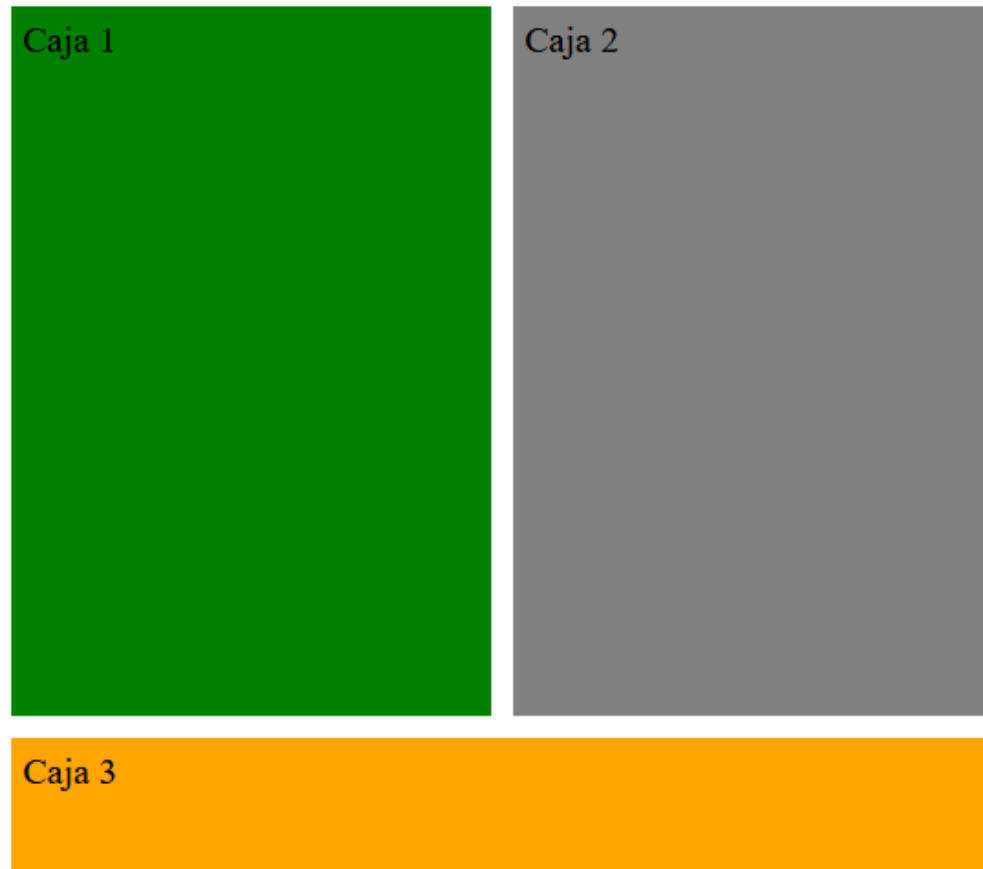
Cajas flotantes

- Ejercicio: Crea el siguiente estilo.

<pre>.cabecera { background-color: green; width: 500px; height: 50px; border-style: solid; border-width: 1px; padding: 5px; }</pre>	
<pre>.barra-lateral { background-color: orange; float: left; width: 150px; height: 200px; }</pre>	<pre>.contenido { background-color: gray; float: left; width: 338px; height: 200px; }</pre>
<pre>.pie { background-color: white; clear: both; }</pre>	

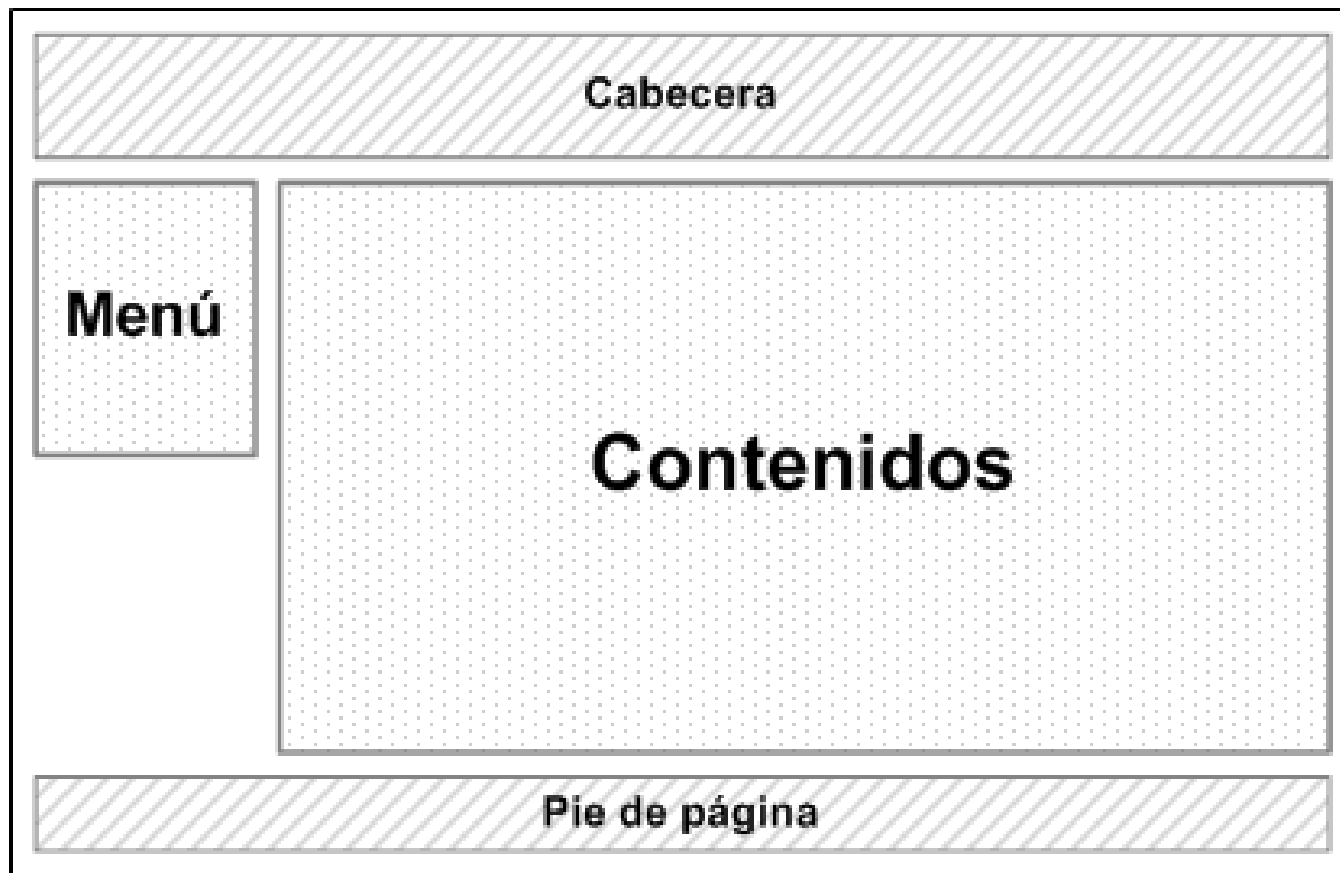
Cajas flotantes

- Ejercicio: Crea el siguiente estilo.



Cajas flotantes

- Ejercicio: Crea el siguiente estilo.

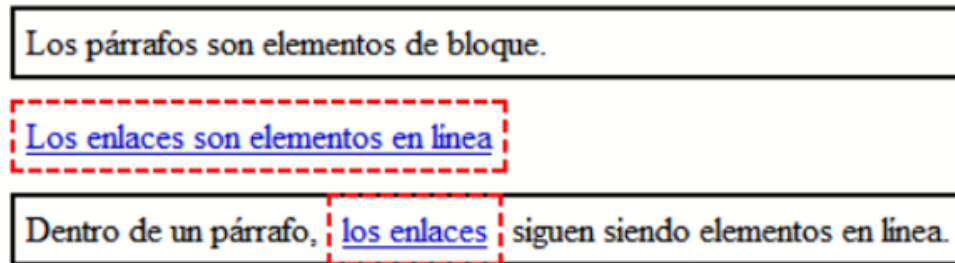


Visualización de los elementos

- El tipo de un elemento influye de forma decisiva en la caja que el navegador crea para mostrarlo o representarlo.
- Un elemento de tipo “bloque” ocupa toda la anchura disponible y tiene un salto de línea antes y después. Por ejemplo: `h1-h6`, `p`, `div`, `menu`, `header`, `footer`, `section`, `ol`, `p`, `table`, `ul`.
- Un elemento “en línea” ocupa exclusivamente la anchura necesaria y no presenta saltos de línea. Sólo el necesario para mostrar sus contenidos. Por ejemplo: `a`, `font`, `img`, `input`, `span`, `textarea`, `var`.

Visualización de los elementos

- Se muestra las cajas que crea el navegador para representar los diferentes elementos que forman una página:



- Con el atributo `display` se puede cambiar el modo en que se visualizan los elementos, mediante sus valores *block*, *inline*, *inline-block*, *flex*, *none*, etc. Por ejemplo, con *inline-block*, se respetan los márgenes o rellenos superior e inferior, con *inline* no.
- Más sobre sus valores: <https://developer.mozilla.org/es/docs/Web/CSS/display>

Selectores

- Basados en etiquetas o tipos (ya visto):
 - Se usan las propias etiquetas HTML como selectores. Se asocia a cada etiqueta una declaración del estilo que se aplica al selector (etiqueta HTML):

```
selector { atributo: valor }
```

- Universal: Mismo estilo a todos elementos de una página.

```
* { atributo: valor }
```

- Ejemplo:

```
h1 { color: blue }
```

- El atributo hace referencia a la característica que se quiere modificar de la etiqueta o elemento, por ejemplo, color.
- El valor hace referencia a la instancia del atributo o propiedad, por ejemplo, blue.

Selectores

- Basados en clases genéricas:
 - Los selectores no se aplican a ninguna etiqueta HTML en particular:

```
.nombreClase { atributo: valor }
```

- Ejemplo:

```
<head><style>
```

```
  .roja { color: red }
```

```
</style></head>
```

```
<body><h1 class="roja">Un encabezado rojo</h1>
```

```
  <p class="roja">y también el párrafo</p></body>
```

- Ahora la misma clase se puede aplicar a otras etiquetas. Se definen cuando el estilo se quiere aplicar a más de un elemento.

Selectores

- Basados en clases:

- Se usan las propias etiquetas HTML como selectores. Se asocia a cada etiqueta una declaración del estilo que se aplica al selector (etiqueta HTML):

```
nombreEtiqueta.nombreClase { atributo: valor }
```

- Ejemplo:

```
<head><style>
  h1.roja { color: red }
  h1.verde { color: green }
</style></head>
<body><h1 class="roja">Un encabezado rojo</h1>
  <h1 class="azul">y ahora verde</h1></body>
```

- Alternativa más potente ya que permite aplicar diferentes estilos a las mismas etiquetas o elementos.

Selectores

- Basados en identificadores:

- Como selector se usa el atributo `id` en la etiqueta HTML:

```
#nombreClase { atributo: valor }
```

- Ejemplo:

```
<head><style>
```

```
  #roja { color: red }
```

```
</style></head>
```

```
<body><h1 id="roja">Un encabezado rojo</h1></body>
```

- Ahora los identificadores solo se pueden aplicar a una única etiqueta.
- Se definen cuando lo que se busca es “exclusividad”, por ejemplo, para nombrar los bloques o secciones principales de un sitio Web.

Selectores

- Basados en nombre y valor de atributo:
 - Se pueden construir selectores utilizando expresiones encerradas entre corchetes que contengan el nombre del atributo y (opcionalmente) su valor o parte de él.
 - `[atributo=valor]` El selector CSS apunta a los elementos con el atributo de nombre de “atributo” y cuyo valor es exactamente la cadena de texto “valor”.
 - `elemento:not([atributo]=valor)` Todos los “elementos” que no tengan el “atributo” con texto “valor”.
 - Ejemplo:

```
[class=roja] { color: red }  
h1:not[class=verde] { color: red }  
<body><h1 class="roja">Un encabezado rojo</h1></body>
```

Selectores

- Operadores de concordancia de subcadena:
 - `[atributo^=valor]` Elementos con el atributo de nombre “atributo” y cuyo valor comienza exactamente por la cadena de texto “valor”.
 - `[atributo$=valor]` Finaliza con “valor”.
 - `[atributo*=valor]` Contiene la subcadena “valor”.
 - Ejemplo:

```
<head><style>
```

```
  [class^=ro] { color: red }
```

```
  [class$=ja] { color: red }
```

```
  [class*=oj] { color: red }
```

```
</style></head>
```

```
<body><h1 class="roja">Un encabezado rojo</h1></body>
```

Combinación de selectores

- Los selectores se pueden agrupar y anidar para conseguir estilos CSS:
 - Más concretos, definidos y optimizados.
 - Tener un archivo CSS más fácil de entender.
- Agrupamientos o encadenamientos:

```
selector1, selector2,... { atributo1: valor1;  
    atributo2: valor2;... }
```

- De esta manera se puede aplicar el mismo estilo a un conjunto de selectores al mismo tiempo.

Combinación de selectores

- Anidamientos:

- Selectores contextuales: Permite aplicar un estilo a un elemento dependiendo de los que tenga alrededor.
- Selector anidado común o descendentes:

```
selector1 selector2 ... { atributo1: valor1;  
    atributo2: valor2;... }
```

- Para crear reglas sobre elementos que están rodeados de otros. Relación no estricta entre padres e hijos.
- No hay límite de anidamientos pero no poner más de 4.

Combinación de selectores

- Anidamiento de selectores hijos:

```
selector1 > selector2 > ... { atributo1: valor1;  
                             atributo2: valor2;... }
```

- Si se desea restringir que las etiquetas estén seguidas unas de otras. Relación estricta entre padres e hijos.

- Anidamiento de selectores hermanos adyacentes:

```
selector1 + selector2 + ... { atributo1: valor1;  
                             atributo2: valor2;... }
```

- Cuando se requiere aplicar un estilo a un elemento que tiene adyacente (al lado) a otro en el mismo nivel de anidamiento.

Combinación de selectores

- Anidamiento de selectores general de hermanos:

```
selector1 ~ selector2 ~ ... { atributo1: valor1;  
    atributo2: valor2;... }
```

- Cuando se requiere aplicar un estilo a un elemento que tiene a otro en el mismo nivel de anidamiento. El segundo elemento sigue al primero (no necesariamente de forma inmediata como sucede en el caso anterior) y ambos comparten el mismo elemento padre.

Buenas prácticas en CSS

- Los selectores se nombran en minúsculas, nunca empezando por caracteres especiales o numéricos.
- El nombre de los selectores debe ser específico y claro, para que tenga una mayor capacidad expresiva.
- El nombre de las clases e identificadores no debe describir una característica visual, como color, tamaño o posición.
- Los nombres deben seguir más una visión semántica que estructural.
- Separa las palabras mediante guiones o mayúsculas.
- No hacer uso excesivo de clases.
- Agrupar las reglas según su selector siempre que sea posible.
- Definir los selectores de etiquetas al principio del CSS.
- Estructurar visualmente los atributos.

Interacción de los objetos con el navegador

- Ejercicio: Interpreta qué hace el siguiente código. ¿De qué color aparecerán los elementos de la lista? ¿De qué tamaño?

```
<head><style>
  i + b { color: red }
  .nivel1 { color: green }
  .nivel2 { color: blue }
  ul ul .nivel2 { font-size: 5px }
  ul ul li { font-size: 10px; color: yellow }
</style></head>
<body><h1><p><i>Título</i>: <b>Anidamiento y agrupamiento</b>
  de selectores</p></h1>
<ul><li class="nivel1">Agrupamiento</li>
  <li class="nivel1">Anidamiento</li><ul>
    <li>Anidamiento común</li>
    <li class="nivel2">Anidamiento de hijos</li>
    <li class="nivel2">Anidamiento de adyacentes</li></ul>
</ul></body>
```


Ejercicios

- Realiza todos los ejercicios del bloque 'Ejercicios 2.6'.
- Realiza todos los ejercicios del bloque 'Ejercicios 2.7'.

Pseudo-clase

- Palabra clave que se añade al selector para especificar un estado especial del elemento.

```
selector:pseudoclase { atributo: valor }
```

- Por ejemplo, se puede utilizar para aplicar un estilo a:
 - Un elemento cuando se pasa el ratón sobre él.
 - Enlaces visitados y no visitados.
 - Un elemento cuando se enfoca.

❑ Enlaces:

- ❑ **Normales:** `a:link {atributos}`
- ❑ **Visitados:** `a:visited {atributos}`
- ❑ **Activos:** `a:active {atributos}` (los enlaces están activos en el preciso momento en que se pincha sobre ellos)
- ❑ **Al pasar:** `a:hover {atributos}` (cuando el ratón está encima de ellos)

Pseudo-elemento

- Permite diseñar partes específicas de un elemento.

```
selector::pseudoelemento { atributo: valor }
```

- Por ejemplo, se puede utilizar para:
 - Aplicar estilo a la primera letra o línea de un elemento.
 - Insertar contenido antes o después del contenido de un elemento.

❑ Efectos especiales:

- ❑ **Primer caracter:** `p::first-letter {atributos}`
- ❑ **Primera línea:** `p::first-line {atributos}`
- ❑ **Insertar antes:** `a::before {atributos}`
- ❑ **Insertar después:** `a::after {atributos}`

Pseudo-selector

- Más sobre selectores:

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

- Listado de pseudo-clases:

<https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-classes>

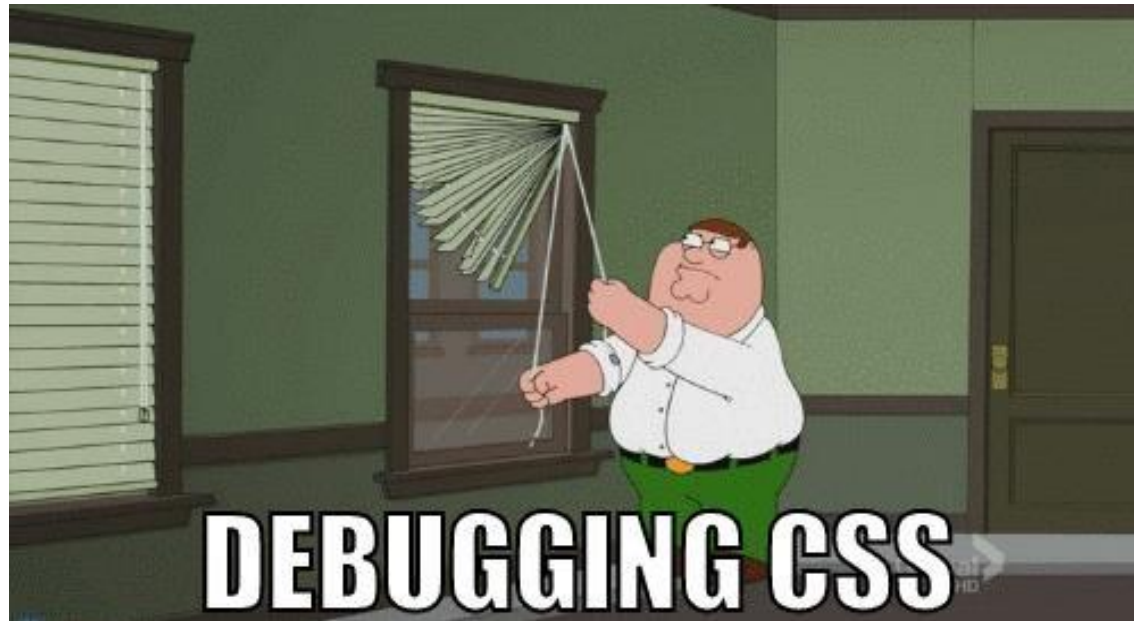
- Listado de pseudo-elementos:

<https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-elements>

Elementos generales

- Atributos relacionados con la **apariciencia**:
 - ❑ **Fondo**: `background-color`, `background-image`.
 - ❑ **Fuentes**: `color`, `font-size`, `font-family`, `font-weight`, `font-style`.
 - ❑ **Párrafos**: `line-height`, `text-decoration`, `text-align`, `text-indent`, `text-transform`.
 - ❑ **Listas**: `list-style-type`, `list-style-image`, `list-style-position`.
 - ❑ **Tablas**: `caption-side`, `table-layout`, `border-collapse`, `border-spacing`, `empty-cells`.
 - ❑ **Visibilidad**: `overflow`, `clip`, `visibility`, `display`.

Ejercicios



Ejercicios

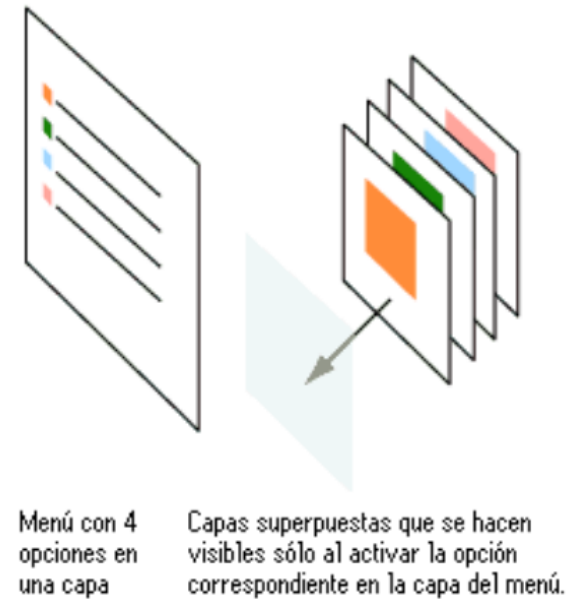
- Realiza todos los ejercicios del bloque 'Ejercicios 2.8'.
- Realiza todos los ejercicios del bloque 'Ejercicios 2.9' (entrega).

Superposición de cajas

- Anteriormente se mostró todo lo relacionado con el **posicionamiento** de los elementos, tanto vertical (atributos *top*, *bottom* y *height*) como horizontal (atributos *left*, *right* y *width*).
- El atributo *z-index* permite definir el nivel de **profundidad** de un elemento. Su valor es un número entero. El valor 0 es el nivel más bajo. Cuanto más alto sea, más cerca se mostrará la capa al usuario.
- Para determinar la **posición** de un elemento respecto al resto se mencionó el atributo *position*. Puede tener como valores: *static* (por defecto), *absolute*, *relative*, *fixed* o *inherit*.

Capas

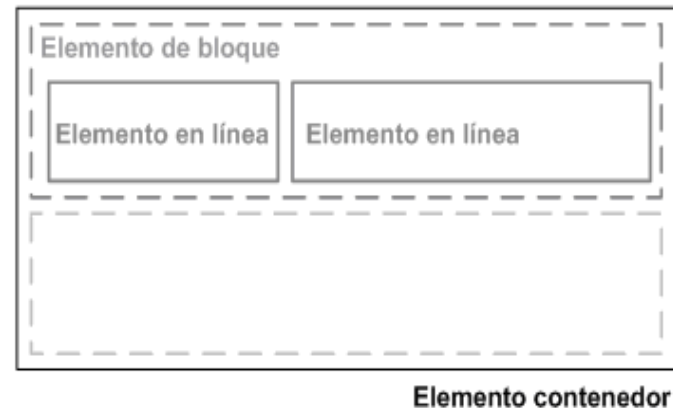
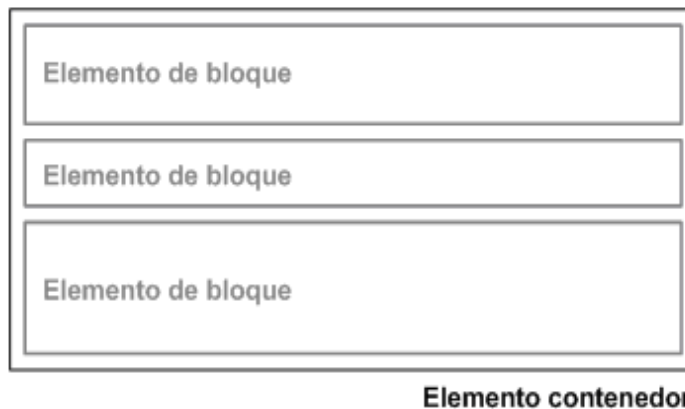
- Capa: División o recuadro, parte de una página Web.
- Pueden ocultarse (atributo *visibility*) y solaparse entre sí (atributo *z-index*), lo que proporciona grandes posibilidades de diseño.
- Se utilizan para no tener que hacer uso de varias páginas Web al momento de presentar la información, sino que dentro de una misma página se encuentre toda ella.



Posicionamiento

■ Normal o estático (por defecto):

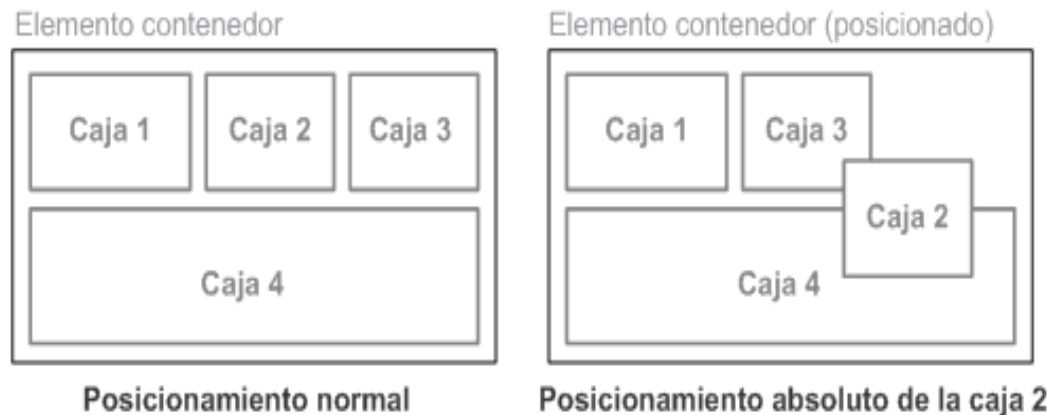
- No afectan las propiedades top, bottom, left y right. Los elementos se colocan según se van definiendo en la página.
- Ninguna caja se desplaza respecto de su posición original, solo se tiene en cuenta si el elemento es de bloque o en línea.



Posicionamiento

■ Absoluto:

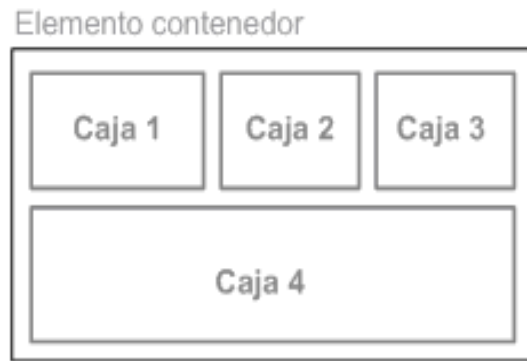
- El elemento se coloca de manera definida por los atributos top, bottom, left y right. El resto lo ignoran y ocupan el lugar original ocupado por este elemento posicionado.
- Cálculo de la posición: Se toma como origen de coordenadas la esquina superior izquierda del primer elemento **contenedor** que esté posicionado de cualquier forma diferente a *static*, recorriendo desde el más cercano a este elemento hasta llegar al `<body>`.



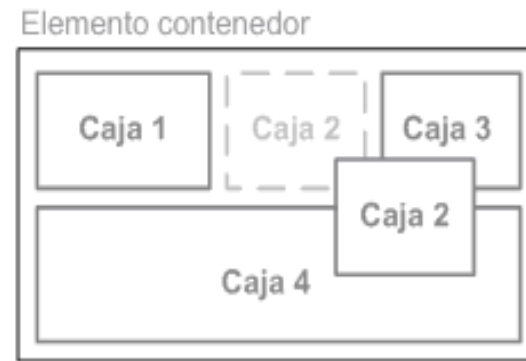
Posicionamiento

■ Relativo:

- Modifica la posición del elemento respecto a su posición original (la que tendría si fuera un posicionamiento estático).
- El desplazamiento de un elemento no afecta al resto de elementos adyacentes → se pueden producir solapamientos.



Posicionamiento normal



Posicionamiento relativo de la caja 2

Posicionamiento

- **Fijo:**

- El elemento permanecerá en la posición indicada, incluso si se hace “scroll” en la pantalla.

- **Inherit:**

- Indica que el valor de *position* tiene que heredarse del elemento padre. No se suele usar.

- **Otros ejercicios de posicionamiento:**

https://www.w3schools.com/css/exercise.asp?filename=exercise_positioning1

Ejercicios

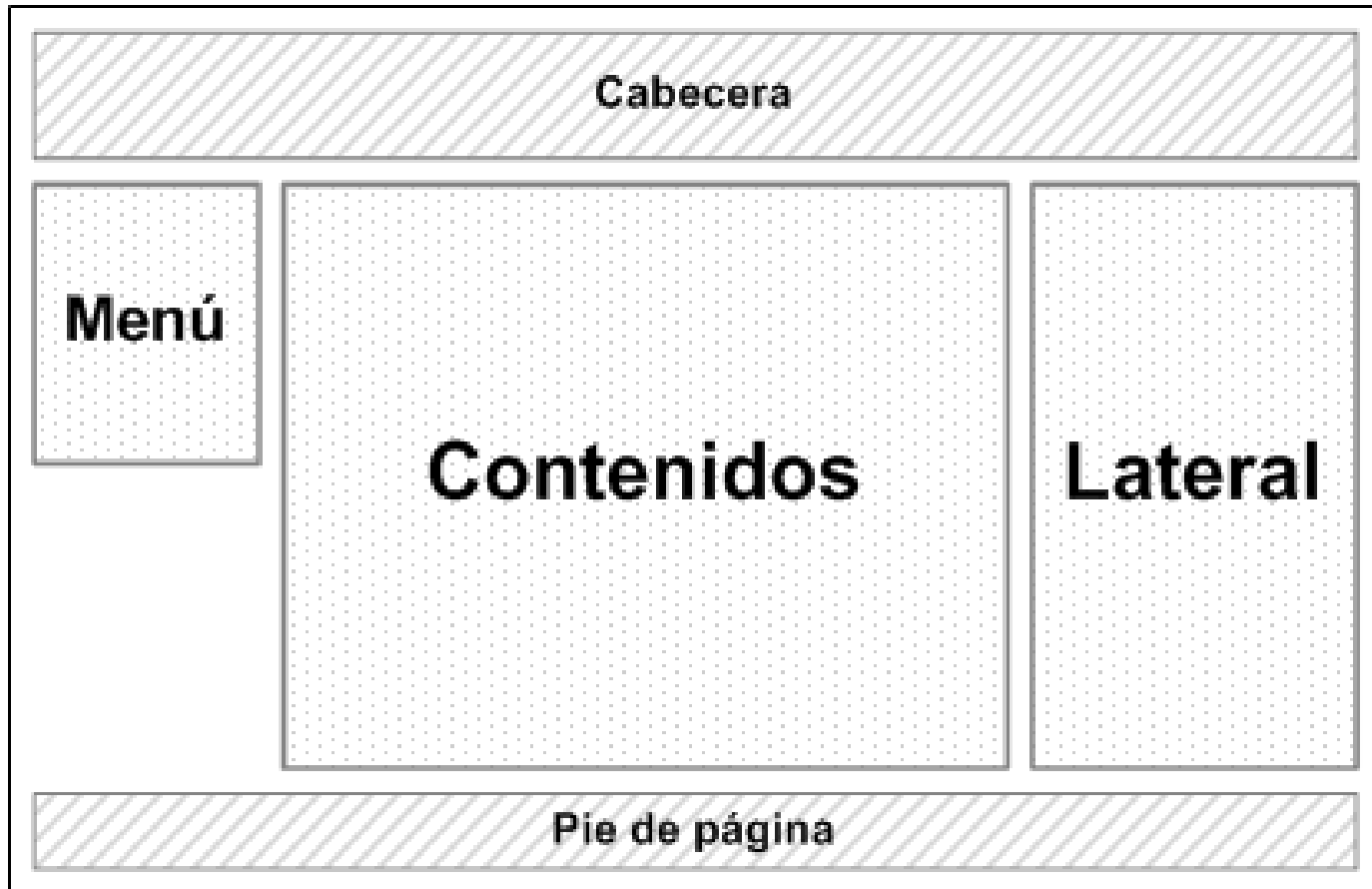
- Realiza todos los ejercicios del bloque 'Ejercicios 2.10'.
- Realiza todos los ejercicios del bloque 'Ejercicios 2.11'.
- Realiza todos los ejercicios del bloque 'Ejercicios 2.12'.

Contenedor

- A medida que aumenta el tamaño y la resolución de las pantallas, se hace más difícil diseñar páginas que se **adapten al tamaño de la ventana** del navegador.
- El principal reto que se presenta con **resoluciones muy altas**, es que las líneas de texto son **demasiado largas** para leerlas con comodidad. Por ese motivo, normalmente se opta por diseños con una **anchura fija** limitada a un valor aceptable para mantener la legibilidad del texto.
- Por otra parte, los navegadores **alinean por defecto las páginas a la izquierda de la ventana**. Cuando la resolución de la pantalla es muy grande, la mayoría de páginas de anchura fija alineadas a la izquierda **parecen muy estrechas** y provocan una **sensación de vacío**.
- La solución más sencilla para evitar los grandes espacios en blanco consiste en **crear páginas con una anchura fija adecuada y centrar la página horizontalmente** respecto de la ventana del navegador.

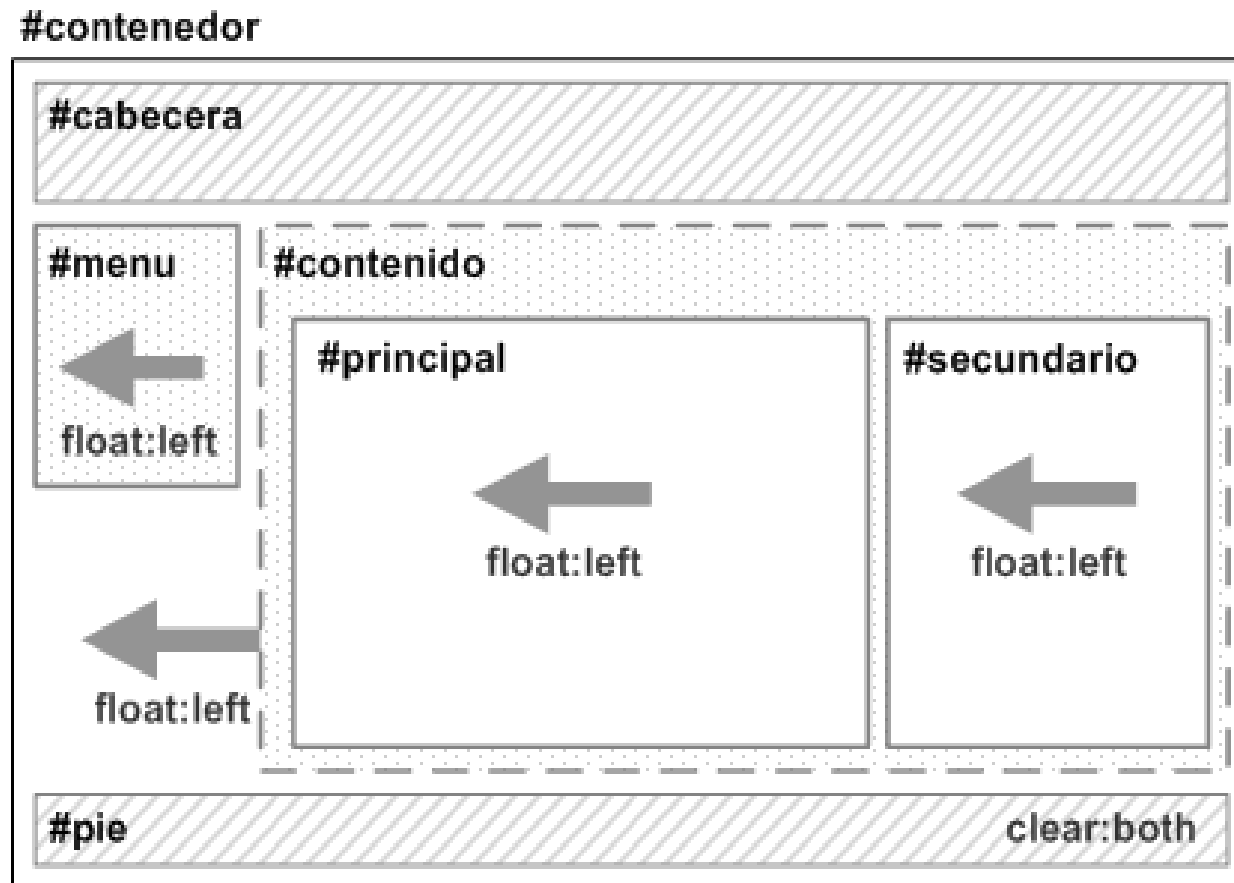
Contenedor

- Ejemplo: Estilo a 3 columnas.



Contenedor

- Ejemplo: Estilo a 3 columnas.



Contenedor

- Dentro de un `<div>` o capa se puede insertar el elemento del tipo que se quiera, incluso varios de ellos, imágenes, etc. Pero, ¿qué sucede si dicho elemento que tiene dentro es más ancho? Para ello existe la propiedad `overflow`:
 - **Visible:** Indica que si el elemento de dentro del `<div>` es de mayor tamaño en altura o anchura que el `<div>` que lo contiene, entonces se le da permiso a este elemento para sobresalga del `<div>`.
 - **Hidden:** El elemento del interior aparece recortado de modo que solo se muestra lo que cabe dentro del `<div>` que lo contiene.
 - **Scroll:** El elemento aparece también recortado como con *hidden*, pero en este caso aparecen alrededor del `<div>` las barras de desplazamiento para que el usuario pueda ver el resto del elemento.
 - **Auto:** Se mostrarán las barras de navegación necesarias (vertical, horizontal o ambas según el caso) solamente si estas son necesarias para mostrar el resto del elemento.
- Más en: <http://www.desarrolloweb.com/articulos/problema-float-maquetacion-css.html>

Precedencia de estilos

- Es común que el estilo que se definió para que se aplique a un elemento no funcione. Por lo general, el **problema** es que se han creado dos reglas que podrían aplicarse al mismo elemento.
- Acciones del navegador (agente de usuario):
 1. Selecciona los estilos a aplicar en función del **tipo de medio**.
 2. Aplica los correspondientes estilos por **importancia y origen**.
 3. En caso de empate anterior, se impone la **especificidad** del selector.
 4. En caso de empate anterior, se impone la última declaración (orden o **cascada**).

Precedencia de estilos

■ Tipos de medios:

- Las hojas de estilo permiten especificar cómo se representa un documento en función de tipo de medio de visualización para que exista uniformidad.
- **Medios:** Canales a través de los cuales es posible visualizar el sitio Web. Pueden ser todos los medios (*all*), pantallas de ordenador (*screen*), impresoras (*print*), móviles (*handheld*), proyectores (*projection*), baja resolución (*tv*), sintetizadores para navegadores con voz (*speech*), etc.

```
<link rel="stylesheet" href="estilo.css" media="screen and (max-device-width: 400px)" ...>
```

```
<style>
  @media speech and (max-device-height: 200px) {
    p.clase {
      color: green;
      ...
    }
  }
</style>
```

Precedencia de estilos

- Por importancia y origen:
 - **Autor:** Estilos CSS aplicados por el desarrollador-diseñador de la página Web.
 - **Usuario:** Cada usuario puede personalizar algunos aspectos de la navegación en el navegador (ejemplo: tamaño de la fuente por defecto).
 - **Navegador:** Cada navegador tiene una serie de estilos por defecto (ejemplo: tipo de fuente).

Precedencia de estilos

- Niveles de prioridad (de mayor a menor):
 1. Declaraciones *!important* del navegador (agente de usuario).
 2. Declaraciones *!important* en los estilos de usuario.
 3. Declaraciones *!important* en los estilos de autor.
 4. Declaraciones normales en los estilos de autor.
 5. Declaraciones normales en los estilos de usuario.
 6. Declaraciones normales del navegador.

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

Precedencia de estilos

- Especificidad: Es cómo el navegador decide qué regla se aplica si varias tienen selectores diferentes, pero aún podrían aplicarse al mismo elemento.
 - Cuádrupla (a, b, c, d) :
 - $a = 1$ si la declaración de estilo es *inline*; 0 en caso contrario
 - b = número de id (#) que haya en el selector
 - c = número de clases, otros atributos y pseudo-clases en el selector
 - d = número de elementos HTML y pseudo-elementos en el selector
 - $(a, b, c, d) > (a', b', c', d')$ si:
 - $a > a'$ ó
 - $a = a'$ y $b > b'$ ó
 - $a = a'$, $b = b'$ y $c > c'$ ó
 - $a = a'$, $b = b'$, $c = c'$ y $d > d'$



Precedencia de estilos

- Ejemplo:

```
html > head+body ul#nav *.miclase a:link {  
  font-size: 12px;  
  color: red !important;  
}
```

- $a = 0$, ya que es un selector, no un estilo en línea (*inline*)
- $b = 1$: #nav
- $c = 2$: clase .miclase y pseudoclase :link
- $d = 5$: html, head, body, ul y a

Especificidad: (0, 1, 2, 5)

- Detalles:

- Combinadores >, + y espacios en blanco no afectan a especificidad
- * no afecta
- ! influye en la importancia, no en la especificidad

○ Ejercicios: https://www.w3schools.com/css/css_specificity.asp

○ Más en: <https://entrellaves.com/css/precedencia-css/>
<https://developer.mozilla.org/es/docs/Web/CSS/Specificity>

Precedencia de estilos

- Cascada: Se refiere al orden que ocupan los diferentes elementos e instrucciones en el documento.
 - La regla declarada en último lugar en el documento tendrá preferencia.
 - Si hay distintas llamadas a distintos documentos CSS, el último tiene mayor prioridad.
 - A priori no habría ventaja entre la inclusión de un fichero externo y una declaración de estilo en <head>.
 - Cuando se aplican dos reglas que tienen la misma especificidad, la que viene en último lugar en el CSS es la que se utilizará.
 - Cuando se aplican dos reglas que tienen la distinta especificidad, la más específica es la que se utilizará.

Hojas de estilo externas

A través de '@import' es posible importar nuevas reglas desde otros ficheros CSS, e indicar el medio o medios sobre los que debe aplicarse el nuevo estilo. En este caso, a diferencia de lo que ocurría con '@media', en primer lugar, aparece la URL que apunta al fichero CSS que contiene la guía de estilo, y, a continuación, se indica el medio sobre el que se va a aplicar. Es importante tener en cuenta que estas reglas deben preceder a todas las demás reglas.

Si los datos de estilo del fichero se van de aplicar a varios medios al mismo tiempo, se indicará el nombre de estos separados por comas.

- Sintaxis de @import:

```
@import url(nombreFichero.css) nombreTipoMedio;
```

Es posible encontrar las reglas anteriores combinadas entre sí en un mismo fichero escrito en HTML. Por ejemplo, anidando las siguientes líneas:

- Utilizando 'link':

```
<link rel="stylesheet" type="text/css" media="screen" href="basico.css"/>
```
- Utilizando '@import':

```
@import url("estilos_seccion.css") screen;
```
- Utilizando '@media':

```
@media print {  
    /* Estilos específicos para impresora */  
}
```

Ejercicios

me working
in backend



**I completed the
entire backend
in just one day**

me working
in frontend



**how the f*ck
I can align
this button
at center?**



developers_team

Ejercicios

- Realiza el 'Ejercicio 2.13' (entrega).