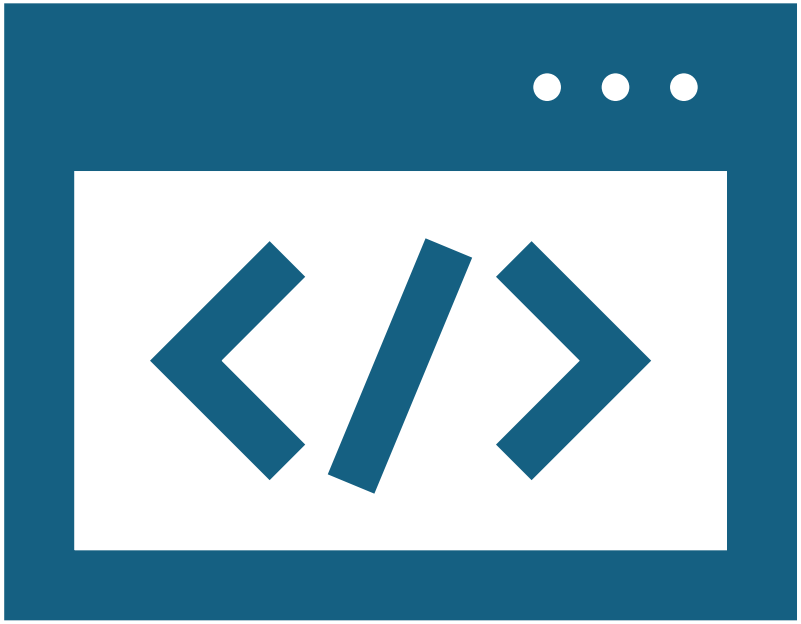
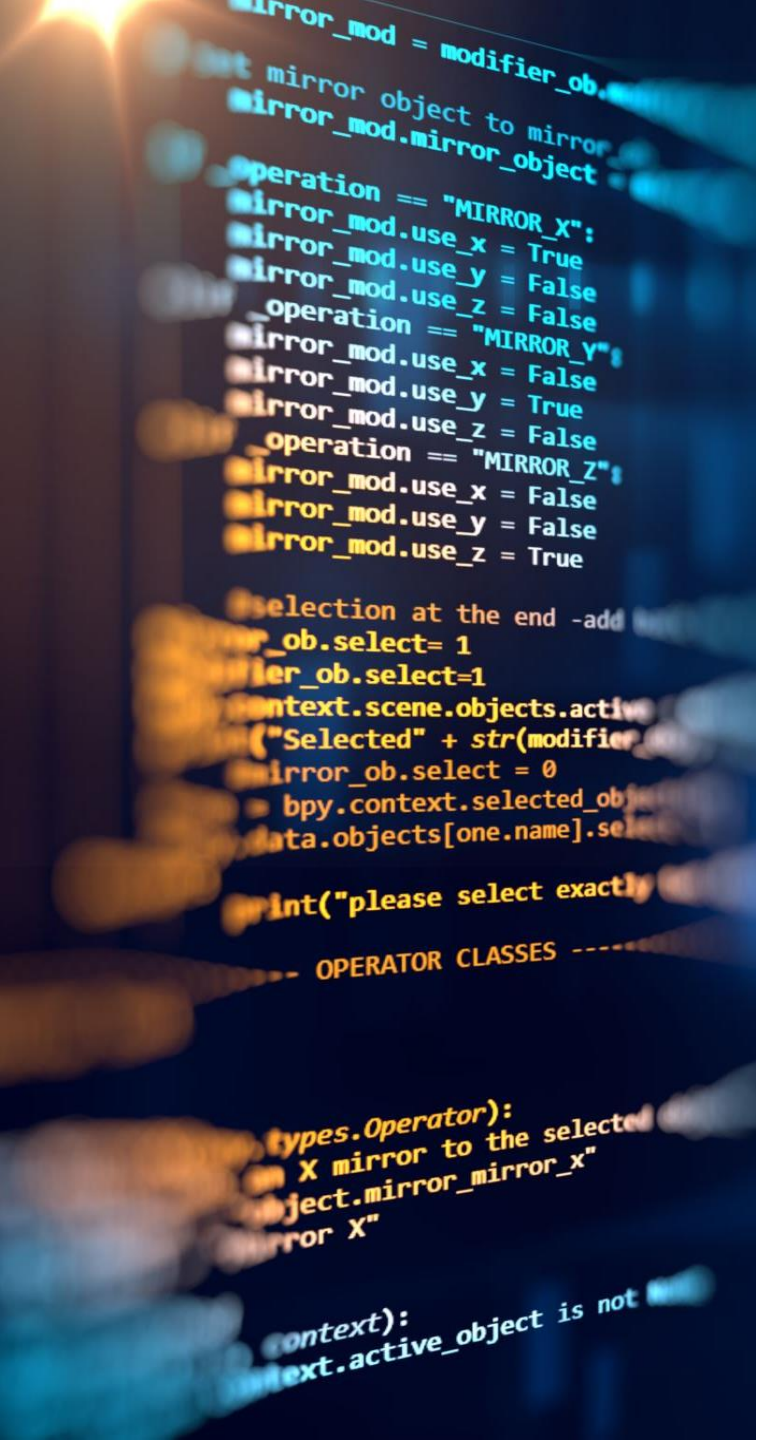


LMSGI

UT 2 - CSS



Sintaxis básica de CSS

- CSS es un lenguaje distinto de HTML es muy sencillo, pero su dificultad radica en que es muy extenso. CSS sigue esta sintaxis
  - **selector {**
    - propiedad1**:**valor1**;**
    - propiedad2**:**valor2**;**
    - …
  - **}**
- **El selector** es la parte de CSS que nos permite indicar a qué elemento (o elementos) de la página web se va a aplicar el formato CSS.

Sintaxis básica de CSS

- Por ejemplo:

```
p{
    font-size:14pt;
    color:red;
}
```

- Ese código CSS provocaría que los elementos de tipo p salgan con tamaño de letra de 14 puntos y color rojo.

Comentarios

- Dentro del código CSS se pueden colocar comentarios. Para ello el texto del comentario se encierra entre los símbolos /* y */. Ejemplo:

```
p {
   line-height:10pt;
   /*El siguiente código marcará el texto
     subrayado*/
   text-decoration:underline;
   text-align:center;
}
```

Inserción de código CSS

1. **En una etiqueta concreta de HTML**

   Todos los elementos de HTML (***p***, ***strong***, ***abbr***, ***h1***,...) pueden utilizar un atributo llamado **style**, dentro de este atributo podemos añadir código CSS.

   <p **style**="color:red;">Este texto sale de color rojo </p>

   En este caso el código CSS no lleva selector alguno, porque el estilo se aplica al elemento en el que se incrustó el código CSS.

Inserción de código CSS

2. **En el elemento de cabecera style**

   En este caso el código CSS se inserta debajo del elemento HTML **style** que se colocará en la zona de cabecera (**head**).

```
<head>
    <style type="text/css">
            p{
                color:red;
             }
    </style>
</head>
```

Inserción de código CSS

- El elemento **style** usa fundamentalmente dos atributos:
- **type**. Que siempre contiene el valor **text/css**. En el caso de que apareciera otro lenguaje de estilos su contenido sería el tipo MIME correspondiente a ese lenguaje.
- **media**. Identifica a qué tipo de dispositivo se aplican los estilos. Podemos diseñar diferentes estilos en función de los dispositivos. Sus posibilidades son:
  - **all**. Opción por defecto que significa que los estilos se aplicarán a cualquier tipo de dispositivo en el que se esté viendo la página.
  - **screen**- Para pantallas a color
  - **print**. Para ser impreso.
  - **tty**. Pantallas de texto (como los terminales **ssh** por ejemplo o los de teletexto)
  - **tv**. Pantallas de televisión
  - **projection**. Proyectores de vídeo
  - **Etc ...**

Inserción de código CSS

```
<style>
    h1 {color:#FFoooo;}
    p {color:#ooooFF;}
    body {background-color:#FFEFD6;}
</style>
<style media="print">
    h1 {color:#oooooo;}
    p {color:#oooooo;}
    body {background-color:#FFFFFF;}
</style>
```

- En cualquier medio aparece el texto en azul, el título en rojo y el fondo en amarillo. Si lo imprimes sale el texto en negro y el fondo blanco.

Inserción de código CSS

```
<style>
body {
    background-color: pink;
}
@media screen and (min-width: 480px) {
    body {
        background-color: lightgreen;
    }
}
</style>
```

- Si el dispositivo es una pantalla a color y con una anchura mínima de 480px, el fondo aparecerá en verde y sino en rosa

Inserción de código CSS

Otra forma (más utilizada) de indicar diferentes medios es utilizar dentro del código CSS la directiva **@media**, ejemplo:

```
<style type="text/css" >
    @media screen{     <!—pantalla de color-->
        p{
            color:green;
        }
    }
    @media print{
        p{
            color:#333333; }
        }
</style>
```

Inserción de código CSS

3. **En un archivo externo**
   - NO es recomendable porque, la web descarga el css antes de actualizarse
     <style> @import 'css/import.css'; </style>
4. **En un archivo externo**
   - Es el caso más habitual . La ventaja es que el mismo archivo nos sirve para aplicar sus estilos a diferentes documentos y si variamos simplemente el archivo CSS, todos los documentos que usen esa hoja de estilo cambiarán automáticamente.

```
<head>
        <meta charset="UTF-8">
        <title></title>
        <link rel="stylesheet" href="estilo1.css" type="text/css">
</head>
```

- La etiqueta **link** tiene los atributos:
  - **href**. Con el que se indica la ruta de la hoja de estilos que se está incluyendo.
  - **rel**. Que siempre contiene el texto **stylesheet** (para indicar que estamos incluyendo una hoja de estilos).
  - **Type.** De tipo text/css

Orden de aplicación de estilos

- Es posible que algunas páginas web utilicen varios estilos referidos al mismo elemento. Ejemplo:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
      <title>Document</title>
      <style type="text/css" media="screen">
          p{
                color:blue;
          }
   </style>
  </head>
  <body>
    <p style="color:red;">Hola</p>
  </body>
</html>
```

La cuestión es de color saldrá el texto ***Hola*** (que dentro del elemento p al que hacen referencia dos definiciones).

Orden de aplicación de estilos

- La respuesta es de color rojo, porque tiene prioridad la definición más restrictiva. Es decir, en orden de preferencia se ejecuta:

  1. Primero se aplican los estilos del navegador. Es decir el formato predefinido del navegador. Todos los navegadores poseen un estilo predefinido, que dicta con que tamaño por defecto se muestra el texto, los colores, el tipo de letra,... Estos estilos son los que se ejecutan en primer lugar. Pero cualquier estilo definido fuera del navegador, tiene preferencia.
  2. Después se aplican los estilos externos (los que se incorporan con la etiqueta **link**)
  3. Después los que proceden de la etiqueta **style**.
  4. Después los que se definan internamente en el elemento (mediante el atributo **style**).

- En caso de dos estilos referidos al mismo elemento y definidos en el mismo ámbito (por ejemplo ambos procedentes de archivos externos e incluidos con el elemento **link**) tiene preferencia el último que se utilice en el código (es decir ganan los estilos del segundo **link**).

Orden de aplicación de estilos

```
<style>
   strong{ color:red; }
   p{ color:green; }
</style>
<body>
   <p>Soy verde<strong>Soy rojo</strong></p>
</body>
```

- El elemento **strong** aparece de color rojo ya que es más interno que el elemento **p**.

Orden de aplicación de estilos

- No obstante se puede alterar la preferencia utilizando una palabra clave: **!important**
- Los estilos marcados con ella tienen preferencia sobre cualquier otro. Ejemplo:

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

- El color verde para los párrafos tendrá preferencia sobre cualquier redefinición de estilos sobre el elemento **p**.

Herencias

- Hay que tener en cuenta que hay etiquetas que son *padre* de otras. Es decir etiquetas que contienen a otras. En el ejemplo:

  <**p**>Arturo Herrero: <**em**>Los años veinte</**em**></**p**>

- La etiqueta **p** es padre de la etiqueta **em** (**em** está dentro de **p**). Esto hace que **em** herede todo el estilo que posea **p** y además añada el suyo propio. Por ejemplo, si hemos definido:

  p{ color:blue; font-size:12pt }
  em{ font-size:14pt; }

- En el ejemplo anterior, ***los años veinte*** tendrán color azul y tamaño 14

Selectores

- Los estilos CSS se aplican hacia el elemento HTML que indiquemos. Este elemento puede ser una etiqueta HTML, pero también podemos hacer selecciones más elaboradas

```
p{
}
#identificador{
}
.clase{
}
```

Selectores – Elementos HTML

- Podemos aplicar un estilo a un elemento concreto de HTML. Como en los ejemplos anteriores:

  p{ color:blue; font-size:12pt }

- En principio con ese código todos los elementos de tipo p de la página saldrán de color azul.

  Podemos incluso aplicar el estilo a varias etiquetas a la vez:

  h1,h2,h3{ color:blue; }

- Los títulos de tipo **h1**, **h2** o **h3** saldrán de color azul.

Selectores – Clases

- Una clase es un identificador que asignamos a una serie de propiedades y valores CSS. Se configuran de esta forma:

```
selector.nombreclase{
    propiedad1:valor1;
    propiedad2:valor2;
    …
}
```

- Por ejemplo:

```
p.clase1{
   color: #339999;
   background-color: #D6D6D6;
}
```

Selectores – Clases

- Para que un párrafo (necesariamente marcado con la etiqueta **p**) adopte este estilo hay que indicarlo gracias a un atributo presente en todos los elemento HTML; se trata del atributo **class**. Ejemplo:

  <p **class**="**clase1**">
  Este texto sale con el formato indicado por la clase
  clase1
  </p>

- Definir la clase sin indicar a qué selector pertenece:

  .clase1{ color: #339999; background-color: #D6D6D6; }

- En ese caso cualquier elemento HTML podrá utilizar esa clase. Ejemplo:

  <h1 **class**="**clase1**"> Título con estilo clase1 </h1>

Selectores – Identificadores

- La idea es parecida a la de las clases, pero ahora el nombre que indiquemos se referirá al atributo identificador de un elemento, el cual se marca con el conocido atributo **id** de HTML .
- **Los identificadores no se pueden repetir en el mismo documento, por lo que el formato indicado sólo se aplicará a un elemento de la página.**
- Ejemplo:

  #parrafo1{ color: #339999; background-color: #D6D6D6; }
- Y la forma de aplicar:

  <p **id**="**parrafo1**"> texto del párrafo coloreado </p>

Selectores – De Limitación

- Permite indicar que el estilo definido se aplica a una determinada etiqueta pero cuando sea hija de otra que especificada. Ejemplo:

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

- Se aplica a los elementos de tipo ***p*** cuando están dentro de elementos de celda (***td***).

Selectores – Selector universal

- Existe un selector que permite aplicar un estilo a todas las etiquetas. Es el asterisco. Ejemplo:

  *{ color:black; }

- No se suele utilizar de esa forma ya que es demasiado indiscriminada. Pero sí se utiliza para elementos de este tipo:

  table * **p**{ color:red; }

- Permite colorear en rojo el texto de los párrafos cuando esta etiqueta se use en tablas (es decir, dentro de elementos ***table***); no importará si entre ***table*** y ***p*** hay otras etiquetas (sean del tipo que sean).

Selectores – por atributos

- Permite aplicar estilos a un elemento cuando este usa un atributo sobre el que toma un determinado valor. Para ello se indica el atributo entre corchetes, seguido del signo de igualdad y el valor entre comillas. Ejemplo:

```
<head> <meta charset="UTF-8">
        <title>Document</title>
        <style type="text/css" >
                p[lang="en"]{
                        font-style: italic;
                }
        </style>
</head>
<body>
        <p lang="es">texto que sale de forma normal</p>
        <p lang="en">texto que sale en cursiva</p>
</body>
```

Selectores – por atributos

- Se puede mezclar este tipo de definiciones con clases o definiciones por identificador:

```
p.clase1[lang="en"]{
    font-style: italic;
}
```

En este caso el estilo definido se aplica a párrafos de clase 1 que estén marcados con el valor ***en*** en el atributo ***lang*** (es decir que estén en inglés).

- Se puede incluso utilizar más de un atributo:

```
p [lang="en"][spellcheck="true"]{ font-style: italic; }
```

En este caso se aplica el estilo para los elementos de tipo p que usen los atributos **lang** y **spellcheck** con los valores indicados

Selectores – por atributos

- Por otro lado gracias a CSS3 disponemos de estas posibilidades:

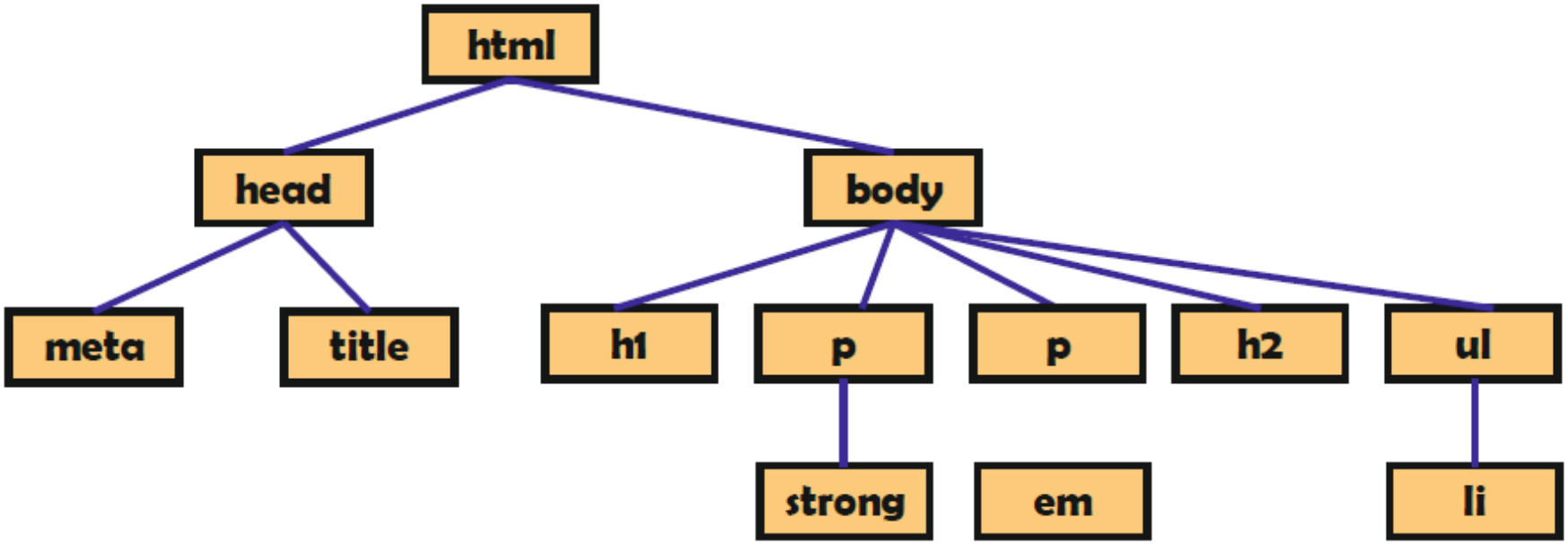
sintaxis	significado
elemento[atributo~="valor"]	Elementos que usen el atributo indicado que contengan el valor aunque separado de otros valores por espacios
elemento[atributo$="valor"]	Elementos que utilicen el atributo y cuyo contenido finalice con el valor indicado
elemento[atributo^="valor"]	Elementos que utilicen el atributo y cuyo contenido empiece con el valor indicado
elemento[atributo*="valor"]	Elementos que utilicen el atributo indicado y contengan (en cualquier parte) el atributo indicado

Selectores – Jerárquicos

- Podemos entender HTML como un documento formado de manera jerárquica, donde hay elementos que contienen otros elementos formando una estructura de árbol.

- ¿Cuál es la jerarquía del siguiente código?

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="UTF-8">
      <title></title>
  </head>
  <body>
    <h1>Monumentos de Palencia</h1>
    <p>Palencia dispone de numerosos monumentos interesantes. Debido a su
       importante actividad medieval posee una gran cantidad de edificios
       religiosos entre los que destaca la <strong>Catedral de Palencia</strong>
       así como la iglesia de San Miguel con su peculiar torre de defensa y el
       monumento más conocido de la ciudad: El Cristo del Otero</p>
    <p>A finales del siglo <em>XIX</em> y principios del <em>XX</em>
       aparecieron edificios suntuosos y civiles que han embellecido una buena
       parte de la ciudad, en especial la transitada Calle Mayor.</p>
    <h2>Edificios religiosos</h2>
    <ul>
            <li>Cristo del Otero</li>
            <li>Catedral Mayor</li>
            <li>Iglesia de San Miguel</li>
            <li>Iglesia de San Lázaro</li>
            <li>Convento de San Pablo</li>
            <li>Iglesia de la Compañía</li>
    </ul>
  </body>
</html>
```



Selectores – Jerárquicos

Selectores – Jerárquicos

- En él se observa como los elementos ***body*** y ***head*** son hijos de ***html***. Mientras que meta y ***title*** son hijos de ***head*** (luego nietos de ***html***). Los nodos al mismo nivel forman hermanos (***h1***, ***h2, p*** y ***ul*** en este esquema son hermanos).
- Para especificar una relación padre/hijo, vimos antes como indicarla de esta forma:

  td p{ color:red; }
- Y esto significa lo mismo que:

  td > p{ color:red; }

  El signo > indica una relación jerárquica de padre a hijo

Selectores – Jerárquicos

Sintaxis	Significado
elemento1 + elemento2	El estilo se aplica al ***elemento2*** cuando es hermano del ***elemento1*** y además el ***elemento1*** precede inmediata-mente al ***elemento2***.
elemento1 ~ elemento2	Se aplica al ***elemento2*** cuando es hermano del ***elemento1*** y éste le precede, aunque no sea inmediatamente.

Selectores – Jerárquicos

Sintaxis	Significado
elemento**:nth-child**(n)	Se aplica al elemento indicado cuando sea el hijo con el número indicado (por ejemplo número sería 3, para el tercer hijo). Se pueden utilizar expresiones más complejas mediante el uso de la variable **n** para conseguir fórmulas más complejas Se permite también usa las palabras clave **odd** (*impar*) y **even** (*par*) Ejemplos de uso: **tr:nth-child(3)**. Selecciona la tercera fila de una tabla **tr:nth-child(odd)**. Selecciona las filas impares **td:nth-child(even)**. A los pares **tr:nth-child(2n+1)**. Selecciona las filas 1,3,5,… (las impares) **tr:nth-child(3n+1)**. Selecciona las filas 1, 4, 7, 10, 13… (de tres en tres)

Selectores – Jerárquicos

Sintaxis	Significado
elemento**:nth-last-child**(n)	Igual que el anterior pero cuenta el orden de atrás hacia delante.
elemento**:nth-of-type**(n)	Funciona como **nth-child** pero ahora se refiere al elemento número **n** (con n teniendo todas las posibilidades comentadas anteriormente) siendo n el número de hijo de ese tipo. **<style>** article p:**nth-of-type(2) { color:red; }** </style> </head> <body> <article> Primer hijo, no se colorea Segundo hijo, primero de tipo p. No se colorea Tercer hijo, segundo de tipo p. SE COLOREA Cuarto hijo, tercero de tipo p. No se colorea </article> </body>

Selectores – Jerárquicos

Sintaxis	Significado
elemento**:nth-last-of-type**(n)	Como el anterior pero cuenta ***n*** desde el final.
elemento**:first-child**	Se aplica al elemento cuando es el primer hijo
elemento**:last-child**	Se aplica al elemento cuando es el último hijo
elemento**:first-of-type**	Primer descendiente de su tipo
elemento**:last-of-type**	Último descendiente de su hijo

Selectores – Jerárquicos

Sintaxis	Significado
elemento:**empty**	Se aplica cuando el elemento está vacío
elemento:**only-child**	Se aplica cuando el elemento es el único hijo
elemento:**only-of-type**	Se aplica cuando el elemento es el único hijo de ese tipo

Podemos incluso hacer combinaciones avanzadas. Ejemplo:

ul > li + li { color:green; }

Se aplica a los elementos ***li*** que estén dentro de elementos ***ul*** y además estén inmediatamente precedidos por otro elemento ***li***.

Pseudoclases

- Las pseudoclases permiten asociar estilos a un selector cuando le ocurre una determinada circunstancia. Las pseudoclases más famosas son las que se aplican a los enlaces (elemento ***a***). Las cuales son:

  a:link. Se aplica para los enlaces no visitados

  a:visited. Enlaces visitados

  a:active. Enlaces activos (aquellos sobre los que hacemos clic)

  a:hover. Se aplica cuando el ratón pasa por encima del enlace

- Estas pseudoclases permiten un cierto dinamismo en la página HTML. Actualmente además es posible utilizarlas en otros elementos distintos de la etiqueta ***a*** (como **p**, **div**, etc.) lo que da enormes posibilidades.

Pseudoclases

Sintaxis	Significado
:focus	Cuando el elemento obtiene el foco. Muy útil en formularios.
:lang(código)	Se aplica cuando el elemento esté marcado con el lenguaje indicado por su código (***es*** para español, ***en*** para inglés,...)
:enabled	Cuando está habilitado (útil en formularios)
:disabled	Cuando está deshabilitado (útil en formularios)
:checked	Para controles de formulario de tipo **radio** o **checkbox** cuando estén activados
:before	Para indicar contenido anterior al párrafo. Siempre se suele usar con la propiedad **content** para añadir contenido al elemento.
:after	Para indicar contenido después del elemento.
:first-line	Aplica estilo a la primera línea del elemento.
:first-letter	Aplica el estilo a la primera letra del elemento.

PROPIEDADES PARA COLUMNAS

- **column-count**: nº de columnas
- **column-width**: para fijar el ancho de las columnas.
- **column-gap:** separación entre columnas
- **column-rule:** estilo para la línea que separa las columnas (como border)
- **column-span:** si el elemento sigue el número de columnas o no (valores ***all y none***)
- **column-fill:** para establecer cómo se rellenan las columnas. El contenedor debe tener altura. Valores ***auto*** o ***balance*** (todas las columnas la misma altura)
- **break-inside: avoid** si queremos que el elemento no quede roto de una columna a otra.

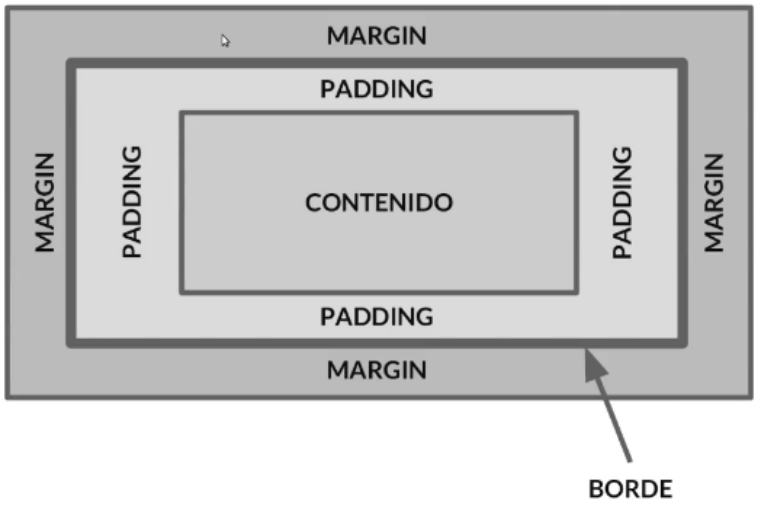
Clase sin columnar

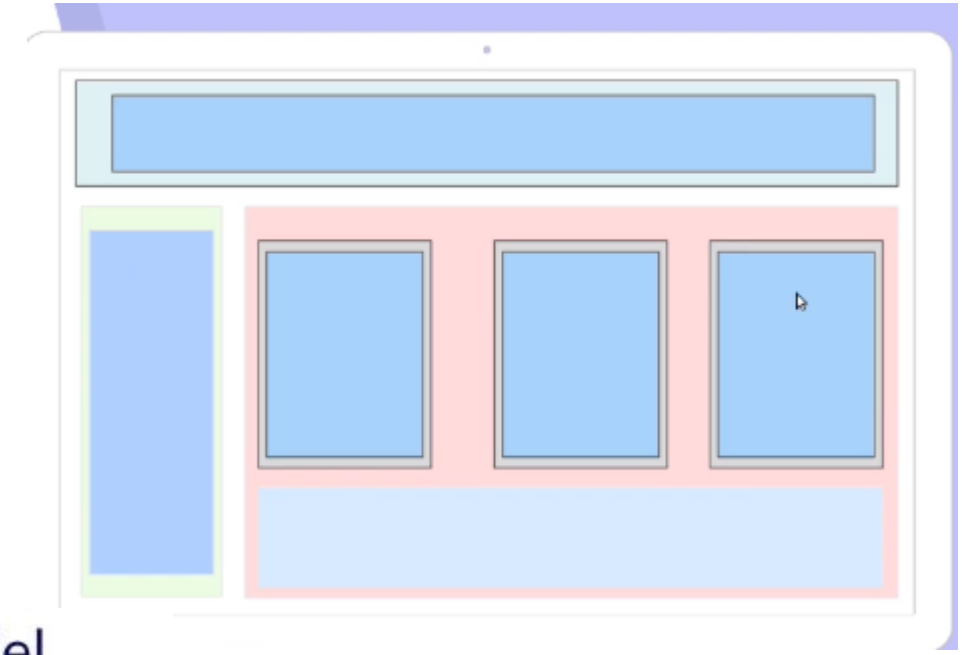
```
.span {
  -webkit-column-span: all;
  column-span: all;
}
```

Clase con tres columnas

```
.threecolumn {
  border: 1px solid black;
  -webkit-column-count: 3;
  column-count: 3;
  margin: 20px auto;
  width: 60%;
}
```

Maquetación Web, caja box-sizing





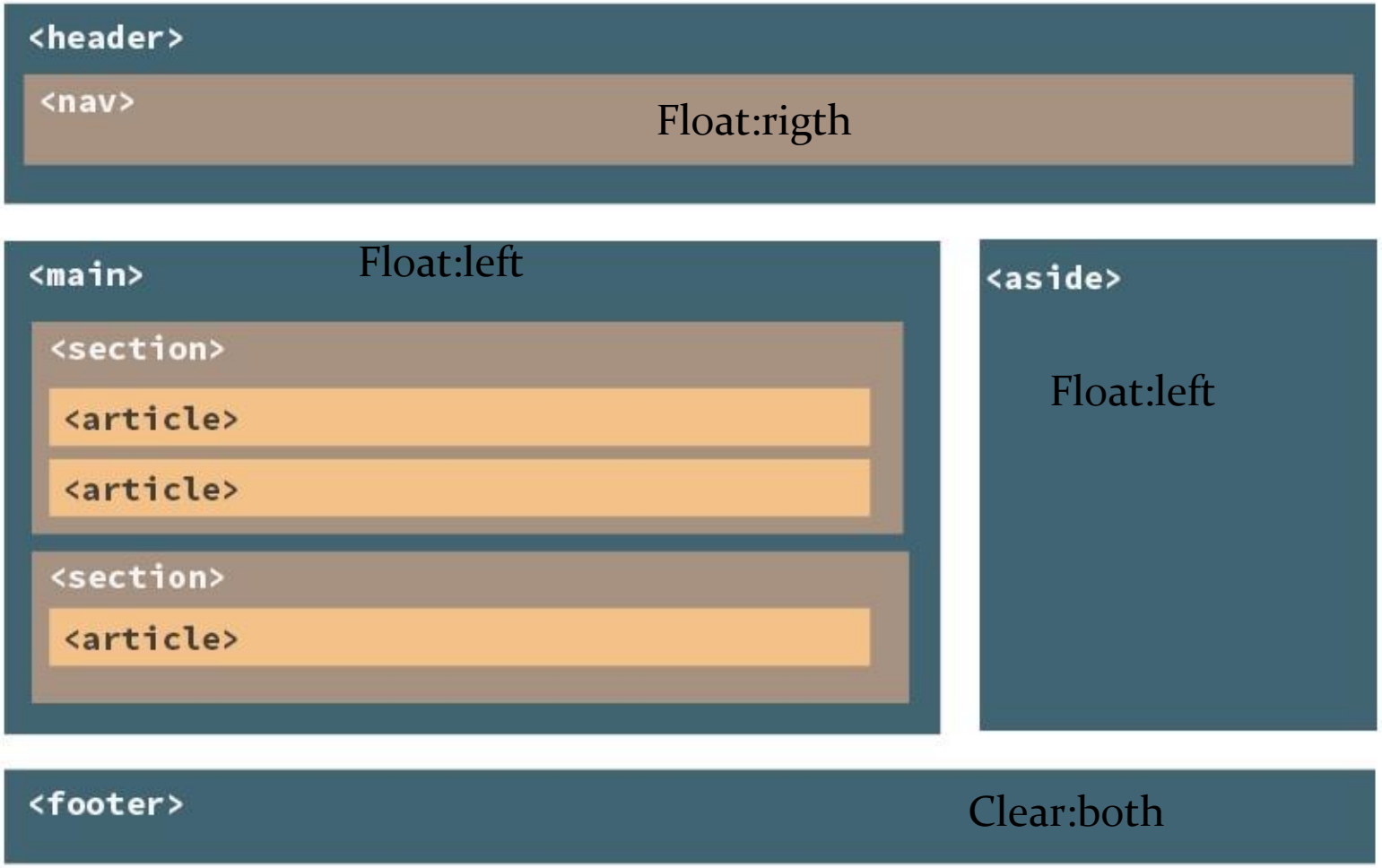
Altura del elemento = altura del contenido+ padding + borde

Anchura del elemento = anchura del contenido+ padding + borde

Siempre para la maquetación:

```
*{
    box-sizing: border:box;
}
```

Estructura web html5



Cuidado con elementos flotantes y la altura del bloque

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Similique omnis obcaecati, veritatis consequatur laboriosam sunt cum assumenda accusamus eius deleniti mollitia, at tenetur nulla quis quidem explicabo non accusantium? Placeat?

La soluciones son dos (ambas son propiedades CSS al elemento contenedor)

```
selector {
 overflow-y: auto;
 height: Altura_suficiente; /*Una de las dos*/
}
```

Dado un mismo elemento HTML que recibe estilos desde 4 ubicaciones distintas, indica cuál es el orden de aplicación de esos estilos sobre el elemento

1 - !important
2 - hoja de estilos .css
3 - estilos de la cabecera HTML
4 - estilos de la etiqueta del elemento

2, 3, 4, 1