

Aplicaciones Web y Lenguajes de Marcas
**Tema 1b3: Hojas de Estilo en Documentos Web =>
CSS**

2º Parcial - LMSGI
1º de Ciclo Superior de DAM

les de Teis (Vigo) – Curso 2020/21

Tema 1b3: Hojas de Estilo en Documentos Web => CSS

Objetivos: *identificar las ventajas de utilizar hojas de estilo en las páginas web para definir la apariencia de formato de las mismas. Crear y aplicar hojas de estilo.*

- **Evolución** de uso de las **páginas web**.
- **Necesidad** de utilizar CSS.
- **Versiones** de CSS.
- Añadir hojas de estilo a las páginas web:
 - ✓ **Selectores** CSS.
 - ✓ **Pseudoselectores**.
 - ✓ **Propiedades:** *modelo cajas, posicionamiento, visualización, texto.*
- Prefijos específicos para navegadores.
- **Optimización** de CSS.
- Estilos por defecto y **reseteo** de propiedades.
- **Mejorando apariencia** de: *enlaces*, tablas, formularios, *listas, menús, diseño de páginas.*



Evolución de uso de las páginas web

- **Evolución de uso de las páginas web:**

- 1) **Inicio de la web:** documentación técnica estructurada de comprensión fácil:

- `<h2>` representa una cabecera menos importante que `<h1>`
 - `` para especificar instrucciones ordenadas

- 2) **Uso extendido:** necesidad de **mejoras gráficas de diseño** => los diseñadores web comenzaron a utilizar más las **etiquetas para controlar la apariencia** que para estructurar la información

- **Ej1:** utilizar `<h1>`, `<h2>`, ... para que el texto sea más grande y en negrita en vez de utilizarlo con la finalidad original de presentar un encabezado.
 - **Ej2:** utilizar las **tablas** para situar imágenes y texto en determinadas partes de la página, insertar una tabla dentro de otra para que la página se vea mejor estéticamente, en vez de utilizarlas para mostrar datos tabulados o como en una hoja de cálculo.
 - **Ej3:** uso masivo de recursos gráficos.
 - Creación de nuevas etiquetas cuya **finalidad** es cambiar la **apariencia**: ``

Evolución de uso de las páginas web

- **Ejemplo de código Html sin utilizar CSS:**

```
<body>
  <h1>
    <font color="red" face="Arial" size="25">Titular de la página</font>
  </h1>
  <p>
    <font color="gray" face="Verdana" size="15">El párrafo de la página con
    atributos que afectan al color, tipo de letra y tamaño</font>
  </p>
</body>
</html>
```

- **Busca** en Internet algún ejemplo más complejo de código Html en el que se entremezcle el contenido de la página web con las características de formato.
- **Averigua:** ¿cuáles crees que son las limitaciones de Html?

Evolución de uso de las páginas web

- **Limitaciones de HTML:**
- ✓ **Lenguaje no apropiado** para definir el **formato** de la página.
- ✓ Las **posibilidades de formato** se consiguen mediante **etiquetas**: nueva necesidad implica la creación de nuevas etiquetas => **infinidad de etiquetas y navegadores** con **etiquetas propias**.
- ✓ **No** permite la **reutilización** del formato para mantener un aspecto coherente y armonioso en todo el sitio web.
- ✓ **Aumento** exponencial de **líneas de código**.
- ✓ **Penalización** en las **búsquedas**:
 - Se **pierde** el **contenido semántico** que debe de tener Html.
 - Se **mezcla contenido y formato**: los motores de búsqueda dejan de buscar
 - **Mantenimiento complejo** de páginas web:
 - Mayor dificultad
 - Mayor tiempo
 - Mayor coste

Ejemplo de página Html formateada con

- Ej: aplicando características de formato en Html con

```
<h1>Titular sin formato</h1>
<p>El párrafo de la página sin ningún tipo de atributos -ni color, tipo de letra ni tamaño-</p>
<br />

<h1><font color="red" face="Arial" size="40">Titular de la página formateado con font</font></h1>
<p><font color="gray" face="Verdana" size="15">El párrafo de la página con atributos que afectan al color, tipo de letra y tamaño</font></p>
<p><font color="gray" face="Verdana">Otro párrafo sin modificar el tamaño con size</font></p>
```

Titular sin formato

El párrafo de la página sin ningún tipo de atributos -ni color, tipo de letra ni tamaño-

Titular de la página formateado con font

El párrafo de la página con atributos que afectan al color, tipo de letra y tamaño

Otro párrafo sin modificar el tamaño con size

Necesidad de utilizar CSS

- **CSS: *Cascade Style Sheets***

*Lenguaje de texto que se **incrusta** en las páginas web para modificar el formato de la página: actúa sobre HTML haciendo que las etiquetas se muestren en el navegador con el formato que se indique.*

- ✓ Es un **lenguaje de diseño gráfico** creado para **controlar el aspecto** y presentación de los documentos web escritos en HTML.
- ✓ Permite **separar el contenido** de la **presentación**.
- ✓ Cuando un usuario solicita una página web a un servidor web, éste devuelve dicha página y los ficheros css que tenga vinculados.
- **Significado** de su nombre:
 - **Cascada:** al aplicar un estilo a un elemento, dicho estilo se propagará a todos los elementos que éste contenga.
 - **Ej:** para aplicar el mismo tipo de letra a toda la página, se definirá en el elemento <body>.
 - **Estilo:** referido a la apariencia visual que queremos asignar a los elementos Html.
 - **Hojas:** utilizamos **ficheros externos** para almacenar los diferentes estilos.

Necesidad de utilizar CSS

- **Ventajas de utilizar CSS:**
 - ✓ **Controla** la “**estética**”.
 - ✓ **Separa el contenido** de la **representación gráfica**:
 - Consigue que **HTML** vuelva a hacer lo que sabe hacer mejor: **estructurar el contenido**.
 - ✓ Permite la **reutilización** del **formato** para mantener un aspecto homogéneo, coherente y armonioso en todo el sitio web.
 - ✓ **Búsquedas optimizadas**: página html sólo con contenido semántico.
 - ✓ Facilita el **mantenimiento** de las páginas web.
 - ✓ **Disminuye** el código a escribir.
 - ✓ Mejora la **legibilidad** de los documentos.
 - ✓ Se puede aplicar también a código XML.

***“Usa HTML para organizar el contenido y
CSS para mejorar su apariencia”***

Necesidad de utilizar CSS

- **Funcionamiento de CSS:**
 - ✓ Actúa sobre **todas las etiquetas** del mismo tipo o sobre **unas** concretas.
 - ✓ Permite asignar **multitud** de **propiedades** con diferentes valores para modificar la apariencia.
 - ✓ Se puede almacenar en un archivo aparte y **reutilizarlo** en varias **páginas** a la vez.
 - ✓ Cualquier **cambio** en el estilo, se refleja al **instante** en todas las páginas.
 - ✓ Algunas **reglas de estilo** se “**heredan**” y otras “**no**”.
- Permite visualizar el **mismo documento** en infinidad de **dispositivos diferentes** => asignar **diferentes estilos** según medio de visualización:
 - Un **único Html**.
 - **Varios css** según **medio** de visualización.

Necesidad de utilizar CSS

- **Unión de Html y CSS:** funcionamiento en conjunto

a) **HTML => Semántica:** creación de **contenido**

- Al crear una página web, se utiliza HTML para **marcar** los **contenidos** => designar una función a cada elemento de la página: “esto es un párrafo, “esto es un titular”, “esto es una tabla”, etc.

b) **CSS => Estética:** definir el **aspecto** de cada elemento con CSS

- Para cada elemento dentro de la página, definir su color, tamaño, tipo de letra, posición, etc.



- **Desventajas de CSS:**

- ✓ Programación **anárquica** => hay que intentar seguir un **orden**.
- ✓ Existen cientos de términos y **propiedades**, miles de **reglas** y **selectores**, y cada declaración puede estar formada por un número **infinito** de **propiedad-valor**.

Versiones de CSS

- **Versiones de CSS:**



- La primera versión se ideó a mediados de los años 90.
- Al igual que Html, se ha ido estandarizando.

a) CSS1:

- ✓ Versión **original** de CSS. Estandarizada en **1996** por la **W3C**.
- ✓ Desarrollada *por Hakon Wium Lie y Bert Bos*, posteriores fundadores de *Mozilla*.
- ✓ Permite poner **orden** en el **caos** que suponía la **interpretación** diferenciada de los **estilos** en cada **navegador**.
- Define **características de estilo sencillas**:
 - ✓ Incluye **formatos de texto** -tipo letra, color de texto- y color de fondo.
 - ✓ **Alineaciones** -de texto, imágenes y tablas-
 - ✓ **Márgenes**, bordes, padding, ...
 - ✓ **Posicionamiento** de elementos.
 - ✓ **Tamaño** de imágenes.

Versiones de CSS

b) CSS2:

- ✓ Estandarizada en **1998** tras varios borradores y revisiones.
- ✓ **Amplía** el CSS anterior creando un estándar más **estable**.
- Incluye sobre todo **posicionamiento** *-manejo de capas-*, además de **tipos de medios**:
 - ✓ Mejoras en el **posicionamiento** de elementos.
 - ✓ Introduce el concepto de **capas** con la propiedad **z-index**.
 - ✓ Permite el **sombreado** de los elementos.
 - ✓ Permite establecer la **dirección** del texto.

b-1) CSS2.1:

- ✓ Última revisión en **2011** tras varios borradores y revisiones desde 2004.
- Modificaciones principales:
 - ✓ **Corrige errores** de la versión anterior.
 - ✓ Suprime los elementos que no utilizaron los navegadores.
 - ✓ Añade *extensiones* de los navegadores.

Versiones de CSS

c) CSS3:

- ✓ Vigente desde **2011**. Evolucionando desde **1999**.
- No es un único estándar => es **modular**.
 - ✓ Debido a la complejidad alcanzada son **módulos independientes** –m. **Selectores**, m. **Uds y Valores**, m. **Alineación de cajas**, ...-.
 - ✓ Cada módulo tiene una evolución y estandarización propia.
 - ✓ Cuando llegue CSS4 no será un estándar único.
- Posibilidades muy **avanzadas de formato**: manejo del contenido, **sombreados y rellenos** avanzados, **transparencias, transiciones, nuevos selectores**.
 - ✓ **Esquinas redondeadas**.
 - ✓ **Gradientes** de colores (para que no haya sólo colores planos).
 - ✓ Transiciones y animaciones.
 - ✓ Nuevos **layouts** para **maquetación** -*Flex, Grid*-
 - ✓ Media-Querye -para **diseño responsive**-

➤ **Averigua:** ¿qué es un **layout**? ¿y un **diseño responsive**?

Métodos de Inserción de código CSS

- **Métodos de inserción de código CSS:**
 - a) En una **etiqueta** concreta de HTML: *estilos en línea* o *inline* con el atributo **style**
 - b) En el elemento de **cabecera** **<style>**: *hoja de estilos interna*
 - c) En el elemento de **cabecera** **<style>** a través de un **archivo externo**: *directiva @import*
 - d) En un **archivo externo**: *hoja de estilos externa* vinculada con **<link>**



Métodos de Inserción de código CSS

a) En una etiqueta concreta de HTML: *estilos en línea* o *inline* con el atributo **style**

- Todos los elementos de HTML (p, h1,...) pueden utilizar un atributo llamado **style** al cual se le asignan las diferentes **propiedades** que queramos modificar con los **valores** que nos interese.

- **Sintaxis:** `<etiqueta style="propiedad1:valor1; propiedad2:valor2">Contenido etiqueta</etiqueta>`

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

- Sólo se recomienda su uso cuando hay que cambiar el estilo de alguno de los elementos en una página web.

- **Ventajas:**

- ✓ El código CSS **no** lleva **selector** alguno => el estilo se aplica directamente sobre la etiqueta con código CSS, es decir, la que utiliza **style**.
- ✓ El resto del documento no queda afectado por el código CSS.

➤ **Deduce:** ¿qué inconvenientes crees que presenta el especificar las características estéticas de los elementos de HTML con el atributo **style**?

Métodos de Inserción de código CSS

- a) En una etiqueta concreta de HTML: *estilos en línea* o *inline* con el atributo **style**
 - Inconvenientes:
 - ✓ Cualquier **modificación** de estilo que quiera hacer de un determinado elemento, se tendrá que hacer en **todas** las apariciones de dicho elemento.
 - ✓ Introducimos **reglas** CSS en la página HTML.

- Ej:

```
<h1 style="color: red; font-family: Arial; font-size:
        large;">Titular de la página</h1>

<p style="color: gray; font-family: Verdana; font-size: medium;">El
párrafo de la página con atributo "style" al que se le asignan diferentes
valores para cambiar el color, tipo de letra y tamaño</p>
```

- **Deduce:** ¿encuentras alguna diferencia entre el uso de la etiqueta de HTML **** para aplicar estilos al uso del atributo **style**?

Métodos de Inserción de código CSS

a) En una etiqueta concreta de HTML: *estilos en línea* o *inline* con el atributo **style**

- Ej: : aplicando características de formato con el atributo **style**

```
<h1 style="color: red; font-family: Arial; font-size: large;">Titular de la página</h1>  
<p style="color: gray; font-family: Verdana; font-size: medium;">El párrafo de la página el atributo "style"  
al que se le asignan diferentes valores para cambiar el color, tipo de letra y tamaño</p>
```

Titular de la página

El párrafo de la página el atributo "style" al que se le asignan diferentes valores para cambiar el color, tipo de letra y tamaño

Métodos de Inserción de código CSS

b) En el elemento de **cabecera** `<style>`: *hoja de estilos interna*

- El código CSS se inserta **dentro** del elemento HTML `<style>` situado en la zona de cabecera de la página web `<head>`.
- Este código afectará a **toda la página web**: varias **reglas** CSS que formatean varios elementos de la página actual.

▪ **Sintaxis:**

```
<head>
....
<style type="text/css">
    selector{ propiedad:valor; }
</style>
</head>
```

Ej:

```
<style type="text/css">
    h2{ color:green; }
</style>
```

▪ **Inconvenientes:**

- ✓ Para indicar a qué elementos va a afectar cada una de las **reglas**, hay que utilizar el **selector** adecuado.
- ✓ Cualquier **modificación** que quiera hacer en esa apariencia genérica de una determinada etiqueta, se tendrá que hacer en **todas** las apariciones del resto de las páginas web del sitio web.
- ✓ **Introducimos reglas CSS en la página HTML.**

Métodos de Inserción de código CSS

b) En el elemento de **cabecera** `<style>`: *hoja de estilos interna*

- **Ej:** ¿qué ocurrirá con la apariencia de todos los `<h1>` y los `<p>` de la página web?

```
<head>
  <title>Ejemplo de estilos con CSS</title>
  <style type="text/css">
    h1 { color: red; font-family: Arial; font-size: large; }
    p { color: gray; font-family: Verdana; font-size: medium; }
  </style>
</head>
<body>
  <h1>Titular de la página</h1>
  <p>El párrafo de la página con atributos que afectan al color, tipo de
  letra y tamaño</p>
</body>
```

Titular de la página

El párrafo de la página con atributos que afectan al color, tipo de letra y tamaño- aplicados gracias a los estilos, a través d la etiqueta "style" dentro de la cabecera d la página

- **Deduce:** ¿encuentras alguna diferencia en el uso de la etiqueta de HTML `<style>` en vez de utilizar el atributo *style*?

Métodos de Inserción de código CSS

c) En el elemento de **cabecera** `<style>` a través de un **archivo externo**: *directiva `@import`*

- Llamar a dicha directiva desde el elemento de cabecera `<style>`.
- Crear un **archivo de texto** con **reglas CSS** y guardarlo con extensión **css**.
- Por cada archivo CSS que se quiera utilizar, añadir un `@import`.
- Se pueden **añadir más reglas** de estilo (después de `@import`).
- Cualquier **modificación** que se haga en esa hoja de estilos, se **aplicará automáticamente** en las **páginas** que lo hayan importado.

▪ **Sintaxis:**

```
<head>
....
<style>
    @import "ruta/fichero.css";
</style>
</head>
```

Ej:

```
<style>
    @import "css/estilos.css";
    @import "css/estilos2.css";
    p { color:red; }
</style>
```

▪ **Inconvenientes:**

- ✓ Puede causar problemas de **rendimiento**, ya que hay navegadores que mientras que no se descargue completamente la hoja de estilos, no carga el resto de la página web.

Métodos de Inserción de código CSS

- d) En el elemento de **cabecera** `<style>` a través de un **archivo externo**: *hoja de estilos externa* vinculada con `<link>`
- Crear un **archivo de texto** con **reglas CSS** y guardarlo con extensión **css**.
 - Relacionar el **.css** con el **.html** mediante `<link>` en la cabecera de la página web.
 - Por cada archivo CSS que se quiera utilizar, añadir un `<link>`.
 - Cualquier **modificación** que se haga en los CSS vinculados, se **aplicará automáticamente** en las **páginas** que los hayan enlazado.
 - **Atributos de `<link>`:**
 - ✓ **rel**: para que el navegador sepa el **tipo de relación** que tiene el recurso enlazado con respecto a la página html: tipo de enlace **rel="stylesheet"**
 - ✓ **href**: indica la ruta de la hoja de estilos que se está incluyendo.
 - ✓ **type**: **tipo de recurso** enlazado: **type="text/css"**. **Opcional** en HTML5.
 - ✓ **media**: tipo de dispositivo. **Opcional** en HTML5.
 - **Ventajas:**
 - ✓ Modificaciones inmediatas.
 - ✓ **Reutilización** código.
 - ✓ Resto de ventajas que aporta CSS.

Métodos de Inserción de código CSS

d) En un **archivo externo**: *hoja de estilos externa* vinculada con **<link>**

▪ Ej:

Página Html:

```
<head>
  <title>Ejemplo de estilos con CSS</title>
  <link rel="stylesheet" type="text/css" href="css/estilos.css"
        media="screen" />
  <link rel="stylesheet" type="text/css" href="css/especial.css"
        media="print, handheld" />
</head>
<body>
  <h1>Titular de la página</h1>
  <p>El párrafo de la página con atributos que afectan al color, tipo
  de letra y tamaño –mediante un css externo-</p>
</body>
```

estilos.css

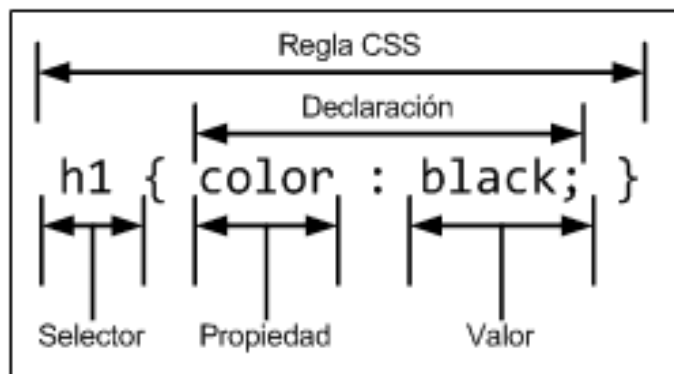
```
h1 { color: red; font-family: Arial; font-size: large; }
p { color: gray; font-family: Verdana; font-size: medium; }
```

¿Cómo se crean los ficheros CSS?

- Un **fichero .css** está formado por **reglas** y **comentarios** (entre `/*` y `*/`).
- **Reglas:** cada uno de los **estilos** q va a contener nuestra hoja css.
 - **Sintaxis básica de las reglas:**
 - a) **Selectores** o clases (**a quién**): indica el elemento o elementos de HTML a los que vamos a **aplicar las reglas** -características de formato-.

reglas o patrones que nos van a permitir seleccionar los distintos elementos de mi página web para poder modificar sus propiedades o estilos visuales

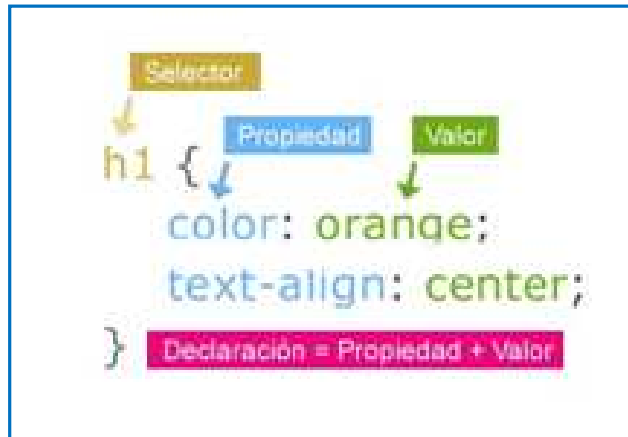
- b) **Bloque de Declaraciones** (**qué hacer**): entre **llaves** de apertura y cierre
 - ✓ Para indicar los estilos que se van a aplicar a los elementos indicados por los selectores.
 - ✓ Compuesta por **una o varias propiedades** –características visuales del elemento o selector que se quieren modificar- a las que se les aplica un **valor**.



```
/*Sintaxis genérica de las reglas CSS*/  
selector {  
    propiedad1:valor1;  
    propiedad2:valor2;  
    ...  
}
```

¿Cómo se crean los ficheros CSS?

- Ej de Regla CSS:



- Ej de fichero .css:

```
h1 { color: white;
      background: orange;
      border: 1px solid black;
      padding: 0 0 0 0;
      font-weight: bold;
    }
/* begin: seaside-theme */

body {
  background-color: white;
  color: black;
  font-family: Arial, sans-serif;
  margin: 0 4px 0 0;
  border: 12px solid;
}
```



Aplicación de Reglas

- ✓ Un **elemento** puede tener **varias reglas** que modifican sus **atributos visuales**.
- ✓ El **aspecto** visual **final** se consigue tras **aplicar todas las reglas** pertinentes que afectan a dicho elemento.
- ¿**Cómo se aplican** las reglas?
- **Herencia:** una de las **características principales de CSS**
 - ✓ La mayor parte de las propiedades son **heredables**: *al definir una propiedad a un elemento, los elementos hijos también adquieren dicha propiedad.*
 - ✓ Los elementos hijos o *descendientes* incluyen varios **niveles de anidación**.
 - **etiquetas padre:** etiquetas que contienen otras.
 - **etiquetas hijas:** además de heredar el estilo de sus etiquetas padre, añaden su **propio estilo** mediante **reglas específicas**.
 - **Ej:** ¿cómo se mostrará **<h1>** y **<p>** si son hijas de **<body>**?

```
<style type="text/css">
  body {
    font-family: Arial, Helvetica, sans-serif;
    color: blue;
  }
</style>
</head>
<body>
  <h1>Titular de la página</h1>
  <p>Un párrafo de texto, otra etiqueta "hija" de <body>.</p>
</body>
```

Aplicación de Reglas: Herencia

- **Herencia:**

- **Averigua:** en la siguiente página web se utilizan los elementos `<h1>`, `<h2>`, y `<p>`. ¿Cómo se respeta proporcionalmente el tamaño, a pesar de la propiedad que se aplica a nivel de `<body>`?

```
body{  
  font-family: Arial, Helvetica, sans-serif;  
  font-size:25px;  
  color:blue;  
}
```

Ejemplos de cómo funciona la herencia

El párrafo hereda los estilos del css externo

Siguiente apartado

Este otro párrafo contiene a su vez una serie de *términos* que permiten distinguir la utilización de diferentes estilos, incluyendo las etiquetas descendientes que se añaden en este párrafo

Una buena forma de eliminar los estilos predeterminados de los navegadores, es aplicar una hoja de estilos de reset.

Los navegadores aplican el tamaño a los encabezados como porcentaje (de ahí que siempre tengan un tamaño proporcionalmente más grande q los párrafos)

Aplicación de Reglas: Herencia

■ Herencia:

- ✓ Permite **ahorrar tiempo**.
- ✓ Funciona con **cualquier** tipo de **selector**: etiquetas, clases, id's, ...
- ✓ Si no hay **ningún estilo** definido para un determinado elemento, éste se visualiza con las **propiedades predeterminadas**.
- ✓ Cada **texto** de las **etiquetas hereda** las **propiedades más cercanas** a su definición: **reglas más restrictivas**.

- Ej: ¿de qué color se escribirá “*historiador*”? ¿y con qué tamaño?

```
<p>Manuel Fernández: <em>historiador</em></p>
```

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

- Ej: ¿de qué color se escribirá “*primer párrafo*”? ¿y “*tercer párrafo*”?

```
<body>  
  <p> primer párrafo </p>  
  <div> segundo párrafo </div>  
  <em> tercer párrafo </em>  
</body>
```

- **Estilo de CSS:**
`body{color:red;}`
`p{color:blue;}`

Aplicación de Reglas: Herencia

- **Herencia:**

- ✓ Para saber si una **propiedad** se **hereda o no**, hay que verificar si en la definición de dicha propiedad aparece el término **i**, referido a **inherit**, es decir, **hereda**.
- **Averigua** qué es una **hoja rápida de estilos**, busca una de ellas e indica una propiedad que se hereda y otra que no.
- **Si no existiese la herencia:** todas las etiquetas interiores a una etiqueta padre mantendrían el tipo de letra, tamaño y color definido de **forma predeterminada** por el **navegador**.
- **Gracias a la herencia:** las etiquetas interiores a una etiqueta padre **heredan** sus características.
- **Ej:** crea una página web en la que aparezca dentro de un párrafo las etiquetas ****, **** y **<a>**. Define una regla CSS para **<p>**. Comprueba la visualización completa del párrafo. ¿Qué le pasa a la apariencia de las etiquetas hijas?
- **Averigua** qué relación hay entre **herencia** y **cascada**.

Aplicación de Reglas: Herencia

- **Coordinación entre *Herencia y Hojas de Estilo***
 - ✓ Permite dar un **aspecto uniforme** a toda la página de forma rápida y fácil.
 - **A nivel de <body>**: crear un estilo para <body> en vez de para cada etiqueta.
 - **A nivel de sección**: Crear un estilo para una sección de la página (<div> o <article>, <footer>, <aside>, <nav>, ...)
 - ✓ La **herencia CSS** asegura que **todos los elementos que están relacionados comparten un formato similar**.
 - ✓ La **herencia de estilos** permite la **combinación de múltiples estilos** que afecten a un único elemento -**estilos híbridos**-:
 - **Ej:** crea una página web en la que aparezca dentro de un párrafo la etiqueta . Define una regla CSS para <body> en la que defines el **tipo de fuente**, otra regla para <p> en la que defines un **color**, y una última regla para en la que determinas el **tamaño**. ¿Con qué características de estilos aparecerá el texto contenido en la etiqueta ?. ¿A qué regla equivaldría?

Aplicación de Reglas: Límites

- **Límites en la Herencia: propiedades que NO se transmiten por herencia**
 - a) **Regla general:** aquellas que afectan a las siguientes características
 - ✓ **Posicionamiento**
 - ✓ **Márgenes**
 - ✓ **Relleno**
 - ✓ **Color de fondo**
 - ✓ **Bordes**
 - **Ej:** si a `<p>` le aplicamos un **borde** ¿interesa que las etiquetas interiores - ``, ``, `<a>`, ...- también lo tengan?
 - b) **Estilos inherentes de los navegadores: tamaño `<h1>`, `<h2>`, ... -más grandes y en negrita- y color de `<a>` -en azul-**
 - **Ej:** si a `<body>` se le asigna cierto tamaño de letra y un color, los encabezados siguen siendo más grandes y los enlaces siguen resaltando su color azul.
 - c) Si los **estilos** entran **en conflicto**: gana el estilo **más específico**, es decir, entre aplicar el formato que se hereda y el formato que se indica desde un **estilo directo**, se aplica éste.

Orden de Aplicación de Estilos: ¡¡en cascada!!

- **Aplicación de estilos en cascada** => tiene **prioridad** la **definición más restrictiva** o menos general, es decir, la **regla** CSS más **específica**.
- En **orden de preferencia** se ejecuta:
 1. **Formato predefinido del navegador** (*estilos del navegador*): todos los navegadores poseen un estilo predefinido, que indica con que tamaño por defecto se muestra el texto, los colores, el tipo de letra,...
 2. Estilos **externos** (los que se incorporan con la etiqueta **<link>**).
 3. Los que proceden de la etiqueta **<style>**.
 4. Los que se definen internamente en el propio elemento mediante el atributo **style**.
- En el caso de **dos estilos** referidos al **mismo elemento** y definidos en el **mismo ámbito** tiene preferencia el **último** que se utilice en el código.
 - **Ej:** ante dos reglas definidas en dos .css incluidos con **<link>**, se aplicará en último lugar el estilo definido en el último css vinculado.
- 5. Alterar la preferencia de estilos utilizando la palabra clave: **!important**
 - **Ej:** asignar el color verde a todos los párrafos, obviando otros estilos de **<p>**.

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

Selectores

1. Selectores básicos:

- ✓ **Universal: Sintaxis:** * { propiedad:valor; }
- ✓ **Etiquetas o de Tipo:** elemento { propiedad:valor; }
- ✓ **Múltiple:** selector1, selector2, ... { propiedad:valor; }
- ✓ **Descendiente o de limitación:** selector1 selector2 ... { propiedad:valor; }
- ✓ **de Clase:** .selector { propiedad:valor; }
- ✓ **de Identificador o Id:** #nombreId { propiedad:valor; }
- ✓ **Combinación de selectores básicos**

2. Selectores avanzados o jerárquicos:

- ✓ **Selector de hijos:** elemento1 > elemento2 { propiedad:valor; }
- ✓ **Selector adyacente:** elemento1 + elemento2 { propiedad:valor; }
 - ✓ **Selector de hermanos:** elemento1 ~ elemento2 { propiedad:valor; }
- ✓ **Selector de atributos:** elemento[atributo] { propiedad:valor; }

Selectores

1. Selectores básicos:

- ✓ **Universal:** se utiliza para seleccionar **todos** los elementos de la página, ya que se aplica a todo el contenido de la misma.
 - Se representa con un *****
 - **Sintaxis:** `* { propiedad:valor; }`
 - **Ej:** `* { color: black; }`
- ✓ **Etiquetas o de Tipo:** se utiliza para seleccionar **todos** los elementos de la página cuya **etiqueta** coincide con ese selector.
 - Se representa con la etiqueta indicada.
 - **Sintaxis:** `etiqueta { propiedad:valor; }`
 - **Ej:** `h1 { color: orange; }`
- ✓ **Múltiple:** se utiliza para seleccionar **varias** elementos a la vez.
 - Se representa con los elementos deseados, separados con **comas**.
 - **Sintaxis:** `selector1, selector2, selector3 { propiedad:valor; }`
 - **Ej:** `h1, h2 { color: orange; }`

Selectores

1. Selectores básicos:

- ✓ **Descendiente** o de **limitación**: se utiliza para seleccionar un elemento *descendiente* o *anidado* de otro.
 - **No** tiene por qué ser **descendiente directo**.
 - Aumenta la precisión del selector de etiqueta, ya que permite aplicar diferentes estilos a elementos de un mismo tipo que sean descendientes de otros.
 - Se representa nombrando los diferentes selectores, separados por un espacio en blanco.
 - Se requiere utilizar al menos dos selectores.
 - Permite cambiar las propiedades de los elementos seleccionados por el **último selector** indicado que se encuentra **dentro** de aquellos seleccionados por selector1 => el **último selector** es el que indica sobre qué elemento se aplican los estilos definidos en la regla Css.
 - **Sintaxis:** selector1 selector2 selector3 { propiedad:valor; }
 - Ej: **td b** { color: red; }
- /* aplica ese color a todos los elementos **b** que están dentro de un **td** */*

Selectores básicos

1. Selectores básicos:

✓ Descendiente o de limitación:

- **Restricción de alcance** se puede crear un selector más **específico** indicando toda la secuencia de **etiquetas anidadas**.

*Permite aplicar un **estilo** a un elemento si está dentro de los elementos anteriores indicados en el **selector***

- **Sintaxis:** selector1 selector2 selector3 { propiedad:valor; }
- Ej: **p b a** {font-size:20px; }

/ aplica ese tamaño a todos los enlaces que estén dentro de un **b** que a su vez está dentro de un **p** */*

➤ Ej: crea una página web para comprobar el funcionamiento de estos selectores.

- Encuentra las diferencias entre el selector **múltiple** y el selector **descendiente**:
 - **p, a, span, em** { text-decoration:underline; }
 - **p a span em** { text-decoration:underline; }
- ¿Hay alguna **diferencia** entre crear un selector como **p a {...}** y **p * a {...}** ?

Selectores

1. Selectores básicos:

- ✓ de **Clase**: se utiliza para seleccionar a todos los elementos cuyo atributo **class** se corresponde con el valor indicado.
 - Para definir una **clase** hay que asignar un **valor** al atributo **class** a uno o varios elementos de **Html**:
 - Se puede aplicar a **cualquier etiqueta**, permitiendo que se **diferencie** del resto de etiquetas, incluso de otras etiquetas del **mismo tipo**.
 - Muy útil para dar formato a determinados bloques del código.
 - ❖ Se puede asignar la **misma clase** a **varios elementos**.
 - ❖ **Un elemento** de Html puede pertenecer a **varias clases**:
 - **Sintaxis**: `<etiqueta class="clase1 clase2 ...">`
 - Se representa con un **.** o con *****.
 - **Sintaxis1**: `.nombreClase { propiedad:valor; }`
 - **Sintaxis2**: `*.nombreClase { propiedad:valor; }`
 - **Ej**: `.noticias { color: blue; }`
`/* aplica ese color a todos los elementos Html cuya clase sea noticias */`

Selectores

1. Selectores básicos:

✓ de Clase:

▪ Ventajas:

- ✓ Permite aplicar las mismas características de estilo a un **grupo heterogéneo de elementos** –siempre y cuando tengan asignada la misma clase-.
- ✓ **Reutilización** de estilos para **diferentes elementos**.
- ✓ Mayor **precisión** para seleccionar cualquier elemento.
- ✓ **Imprescindible** para diseñar páginas web **complejas**.

- **Restricción de alcance:** se puede crear un selector más **específico combinándolo** con el selector de **etiquetas**.

*Permite aplicar un **estilo a uno** de los **tipos** de elementos que pertenecen a la **misma clase***

- **Ej:** si `<p>`, `<a>` y `` son de la **clase** “*destacado*” y sólo necesitamos aplicar un determinado estilo a los `<p>` de esa clase: **p.destacado { ... }**

Selectores básicos

1. Selectores básicos:

✓ de Clase:

➤ **Ej1:** a partir del siguiente código Html ¿qué tipo de selector utilizarías para aplicar una regla que aplique color azul al texto del primer párrafo?. Pruébalo utilizando:

- a) El selector **universal**.
- b) El selector **etiquetas**.
- c) El selector **descendente**.
- d) El selector **clase**.

```
<body>
  <p> primer párrafo </p>
  <p> segundo párrafo </p>
  <p> tercer párrafo </p>
</body>
```

➤ **Ej2:** indica a qué elementos afectan los siguientes selectores:

- a) **p.destacado** { ... }
- b) **p .destacado** { ... }
- c) **p, .destacado** { ... }

➤ **Ej3:** ¿qué estilos se aplicarán al siguiente <p>?:

```
<p class="destacado especial error">hola</p>

Regla1: .destacado { border: 1px;}
Regla2: .especial { color:#930; }
Regla3: .error { font-weight: bold; }
```

Selectores básicos

1. Selectores básicos:

- ✓ de **Identificador** o de **Id**: se utiliza para seleccionar al elemento cuyo atributo **id** se corresponde con el valor indicado.
 - Se puede aplicar a cualquier etiqueta y **permite diferenciarla** del resto.
 - Útil para dar formato a un **único elemento** de la página.
 - La regla sólo se aplica al elemento que tenga dicho **id**.
 - Para aplicar características de estilo a un **único elemento**, es **más eficiente** utilizar un **selector** de **id** que un selector de **clase**.
 - Se representa con una **#** y el nombre del **id**.
 - **Sintaxis**: `#nombreId { propiedad:valor; }`
 - **Ej**: `#inicio { color: blue; } /* aplica la regla al elemento Html cuyo id=inicio */`
 - **Restricción de alcance**: se puede crear un selector más **específico** combinándolo con el selector de **etiquetas**.
- **Deduce**: con este tipo de selector ¿tiene sentido utilizar la restricción de alcance? ¿y si el archivo css donde está incluido la regla vincula a varias páginas web?

Selectores básicos

1. Selectores básicos:

✓ de **Identificador** o de **Id**:

- **Restricción de *alcance***
- **Sintaxis:** `selector#nombreId { propiedad:valor; ... }`
- **Ej:** `p#inicio { color: blue; }`

/ aplica la regla a todos los párrafos, de las diferentes páginas web, cuyo id sea igual a inicio. Cualquier otro elemento de una página web que tenga ese identificador pero no sean párrafos, NO se verán afectados por dicha regla. */*

➤ **Ej1:** indica a qué elemento o elementos afectan los siguientes selectores:

a) `p#destacado { ... }`

b) `p #destacado { ... }`

c) `p, #destacado { ... }`

➤ **Ej2:** observa el siguiente código e indica cómo se verán afectados los diferentes elementos:

```
<h1 id="principal" class="warning">¡Peligro!</h1>
<h2>Selectores descendientes</h2>
<p class="warning">Cuidado... probando selectores de clase y de id</p>
<p>Otro párrafo, en este caso no pertenece a la clase común warning</p>
```

```
.warning {
    color:#990500;
    font-family: Verdana;
    font-weight: bold; }
h1.warning {
    color: red; }
#principal {
    font-size:50px; }
```


Combinación de Selectores básicos

1. Selectores básicos:

✓ Combinación de selectores básicos:

- CSS permite la **combinación** de uno o más **tipos de selectores** para **restringir** el **alcance** de las reglas CSS.
- Ej: indica a qué elemento o elementos afectan los siguientes selectores:
 - a) `.aviso .especial { ... }`
 - b) `article.aviso span.especial { ... }`
 - c) `ul#menuPrincipal li.destacado a#inicio { ... }`

Combinación de Selectores básicos

1. Selectores básicos:

✓ Combinación de selectores básicos:

■ Restricción de *alcance*:

➤ Solución:

a) `.aviso .especial { ... }`

- Selecciona aquellos elementos con un `class="especial"` que se encuentren dentro de cualquier elemento con un `class="aviso"`.

b) `article.aviso span.especial { ... }`

- Selecciona aquellos elementos de tipo `` con un atributo `class="especial"` que estén dentro de cualquier elemento de tipo `<article>` que tenga un atributo `class="aviso"`.

c) `ul#menuPrincipal li.destacado a#inicio { ... }`

- Selecciona a un enlace con un atributo `id="inicio"` que se encuentra dentro de un elemento de tipo `` con un atributo `class="destacado"`, que forma parte de una lista `` con un atributo `id="menuPrincipal"`.

Selectores avanzados

2. Selectores avanzados:

- Los selectores **básicos** son **suficientes** para **diseñar** cualquier página web, sencilla o compleja.
- Gracias a los selectores **avanzados**, podremos **simplificar** las hojas de estilo.
- Por cuestiones de **compatibilidad**, hay que comprobar su funcionamiento en diferentes **navegadores**.
- Ya hemos visto que **HTML** crea páginas de forma **jerárquica**: está formado por *elementos* que a su vez contienen otros *elementos* formando una **estructura de árbol**.
- Gracias a los **selectores avanzados**, veremos cómo se pueden **seleccionar** elementos que encajen con una cierta **jerarquía**.
- **Clasificación principal de Selectores avanzados o jerárquicos:**
 - ✓ **Selector de hijos**
 - ✓ **Selector adyacente**
 - ✓ **Selector de atributos**

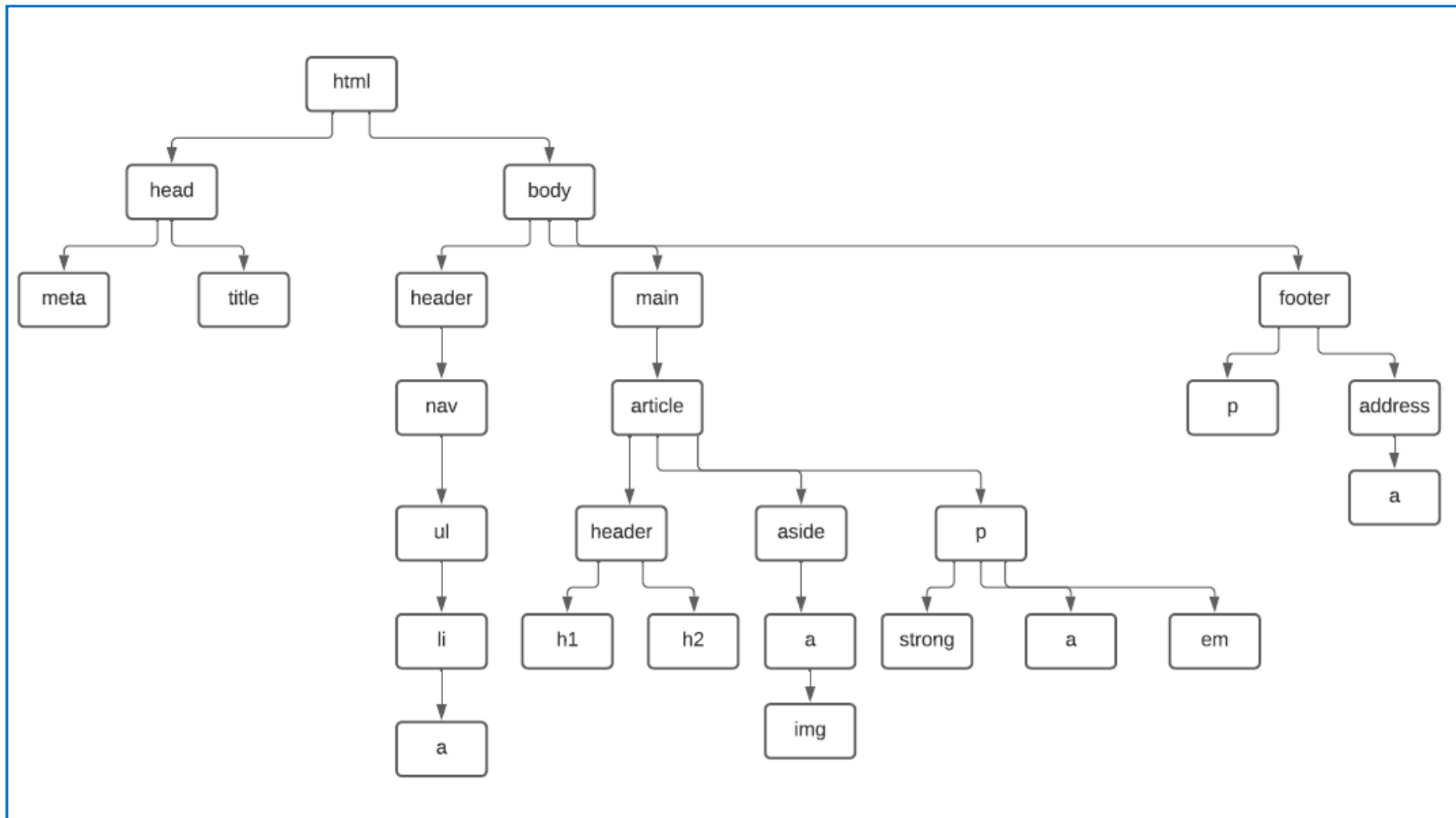
Selectores avanzados

- Observa el siguiente código HTML y crea un diagrama con la estructura jerárquica:

```
<body>
  <header>
    <nav id="menu">
      <ul>
        <li><a href="#autores">Autores</a></li>
        <li>Entrevistas:</li>
        <li><a href="5d_img_enlaces.html">Imágenes</a></li>
        <li><a href="obras.html">Obras</a></li>
      </ul>
    </nav>
  </header>
  <main>
    <article id="manuelfernandez">
      <header>
        <h1>Manuel Fernández Álvarez: historiador</h1>
        <h2>Pequeña Historia de España</h2>
        <h3>¡Con ilustraciones a todo color!</h3>
      </header>
      <aside>
        <a href="https://www.boolino.es/es/libros/autor/manuel-fernandez-alvarez/" rel="n
          
      </aside>
      <p><strong>Miembro de la Real Academia de la Historia</strong>. Profesor emérito de l
        Ha dedicado más de cincuenta años al estudio del siglo XVI y es autor de <a href="htt
          rel="noreferrer nofollow" target="_blank">38 libros</a> y de más de 100 artículos.
        En su mayoría sobre la España de los Austrias, en la que es considerado uno de los má
        Ganó en 1985 el Premio Nacional de Historia con su libro La sociedad española del sig
        También ha recibido la <em>Medalla de Oro</em> de la ciudad de <strong>Salamanca</str
      </article>
    </main>
    <footer id="final">
      <p>Diseño de Desarrollos Web S.L.</p>
      <p>Imágenes con derechos de autpr Copyright&copy</p>
      <address>
        Contacte con el autor desde su cuenta oficial <a href="mailto:ilustrator@gmail.com?su
      </address>
    </footer>
```

Selectores avanzados

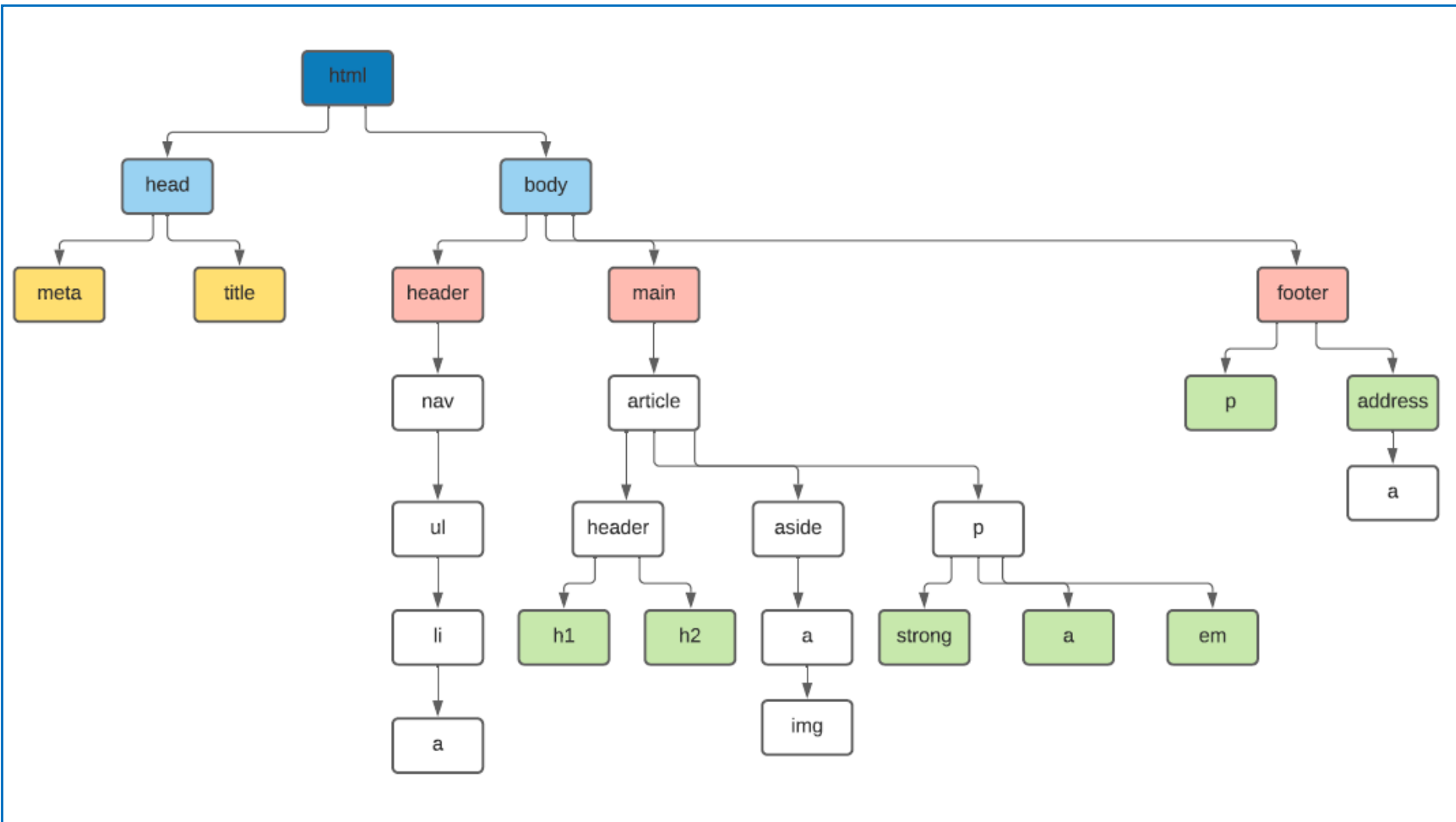
- **Sol.** (añadiendo la parte habitual del `<head>`)



- ¿Cuáles son los elementos *hijos*, *hermanos* y *padres* de cada etiqueta?

Selectores avanzados

- **Sol.:** a partir de la etiqueta inicial de <html>, todos los elementos tienen un *padre*, la mayoría tienen a su vez elementos *hijos*; además, hay varios elementos que son *hermanos* entre sí.



Selectores avanzados

2. Selectores avanzados:

- ✓ **Selector de hijos:** se utiliza para seleccionar aquellos elementos de la página que son **hijos** o **descendientes directos** de otro elemento.
 - Se representa con el símbolo > indicando la **relación jerárquica** existente entre dichos elementos.
 - **Sintaxis:** elementoPadre > elementoHijo { propiedad:valor; }
 - **Ej:** `td > p { color: blue; }`
- **Ej:** ¿qué **diferencia** hay entre **selector descendiente** y **selector hijo**?
Comprueba el funcionamiento con el siguiente código:

```
<p><a href="#">Enlace1</a></p>           /* código HTML */  
<p><span><a href="#">Enlace2</a></span></p>  
  
p a { color: red; }                       /* código CSS */  
p > a { color: yellow; }
```

- ¿y si cambiamos el **orden** de las **reglas**?

Selectores avanzados

2. Selectores avanzados:

✓ Selector de hijos

- **Sol.:** con **selector descendiente** los hijos no tienen por qué ser directos.

Etiqueta "a" hija directa de "p", debería de pintarse en amarillo, regla avanzada de hijos

Etiqueta "a" hija directa de "span", y a su vez hija de "p". Le debería de afectar la regla descendiente, de color rojo.

- Al cambiar el **orden** de las **reglas**, en **último lugar** se aplica la regla de selector **descendiente** que se cumple en ambos párrafos:

```
p > a { color: yellow; }
```

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

Etiqueta "a" hija directa de "p", debería de pintarse en amarillo, regla avanzada de hijos

Etiqueta "a" hija directa de "span", y a su vez hija de "p". Le debería de afectar la regla descendiente, de color rojo.

- ¡¡Más adelante veremos por qué el **orden** de las reglas afecta al resultado!!

Selectores avanzados

2. Selectores avanzados:

- ✓ **Selector adyacente:** se utiliza para seleccionar elementos **adyacentes** y aplicar un estilo al elemento que está situado más a la **derecha**:
 - Selecciona un determinado elemento si va **precedido** de otro
 - Los elementos que forman el selector **adyacente** deben cumplir dos **condiciones**:
 - *elemento1* y *elemento2* deben ser **elementos hermanos** (mismo padre).
 - *elemento2* debe aparecer **inmediatamente** después de *elemento1*.
 - Se representa utilizando un **+** entre dichos elementos.
 - **Sintaxis:** `elemento1 + elemento2 { propiedad:valor; }`
 - **Ej:** `h1 + h2 { color: orange; }`
/ aplica esa característica sobre todos los elementos <h2> cuando el elemento anterior es <h1>, es decir, aparecen de forma consecutiva en la página Html */*

Selectores avanzados

2. Selectores avanzados:

✓ Selector adyacente:

- Ej: Comprueba el funcionamiento del selector **adyacente** con el siguiente código:

```
<body>                                /* código HTML */  
  <header>  
    <h1>Titulo1</h1>  
    <h2>Subtítulo</h2>  
  </header>  
  <h2>Otro subtítulo</h2>  
  ...
```

```
h2 { color: green; }                    /* código CSS */  
h1 + h2 { color: red; }
```

- ¿y si cambiamos el **orden** de las **reglas**?

Selectores avanzados

✓ Selector adyacente:

- **Sol:** el selector **adyacente** implica que los elementos son hermanos consecutivos:

Selectores adyacentes

Título h2 hermano consecutivo de h1

Un h2 que no tiene ningún hermano antes

Vamos a comprobar qué ocurre con los selectores adyacentes. Para ello, a

Nuevo párrafo, a ver si se aplica la regla de selector descendiente al enlace dentro un enlace [Enlace a ninguna parte](#). Todas las etiquetas están de

Otro título h2 cuyo hermano adyacente es un p

- Aunque se cambie el **orden** de las **reglas**, el resultado es el **mismo**, ya que la regla con selector adyacente es más **restrictiva** que la regla con selector de etiqueta.

```
/* selector adyacente, regla más específica */
h1 + h2 {
  font-family: Verdana;
  font-size: 20px;
  color: #339999;
}

/* selector de etiqueta, regla más general */
h2 {
  font-family: Arial;
  font-size: 20px;
  color: orange;
}
```

Selectores avanzados

2. Selectores avanzados:

- ✓ **Selector de atributos:** se utiliza para seleccionar elementos de Html en función de si tiene o no un determinado **atributo**.
 - También se puede comprobar el **valor** del atributo.
 - Se puede utilizar **cualquiera** de los atributos de las etiquetas Html.
 - **Tipos:**
 - a) Selecciona todos los elementos que **tienen** asignado un determinado **atributo**.
 - b) Selecciona todos los elementos que **tienen** asignado un determinado **valor** en el **atributo** indicado.
 - c) Selecciona todos los elementos que **tienen** asignado un determinado **atributo** y al menos **contiene uno** de los **valores** indicados.
 - d) Idem, pero el valor del atributo **finaliza** con el valor indicado.
 - e) Idem, pero el valor del atributo **empieza** con el valor indicado.
 - f) Idem, pero el valor del atributo **contiene** el valor indicado.

Selectores avanzados

✓ Selector de atributos:

- **Tipos:**

- a) Selecciona todos los elementos que **tienen** asignado un determinado **atributo**.

- Se representa utilizando un **[]** para indicar el **atributo**.

- **Sintaxis:** `elemento[nombre_atributo] { propiedad:valor; }`

- **Ej:** `a[class] { color: red; }`

/ colorear de rojo todos los enlaces que utilicen el atributo **class**, independientemente del valor que tenga asignado*/*

- b) Selecciona todos los elementos que **tienen** asignado un determinado **valor** en el **atributo** indicado.

- Se representa utilizando un **[]** para indicar el **atributo** y un **=** para asignar el valor correspondiente.

- **Sintaxis:** `elemento[nombre_atributo="valor"] { propiedad:valor; }`

- **Ej:** `a[class="colorado"] { color: red; }`

/ colorear de rojo todos los enlaces que utilicen el atributo **clase** y el valor que tienen asignado es **colorado***/*

Selectores avanzados

✓ Selector de atributos:

- **Tipos:**

- c) Selecciona todos los elementos que **tienen** asignado **al menos** un determinado **valor** en el **atributo** indicado.

- Se representa utilizando un **[]** para indicar el **atributo** y un **~=** para asignar el valor que debe contener.

- **Sintaxis:** `elemento[nombre_atributo~=“valor”] { propiedad:valor; }`

- **Ej:** `a[class~=“colorado”] { color: red; }`

/ colorear de rojo todos los enlaces en cuya **clase** aparezca, entre otros, el valor **colorado***/*

➤ **Ej:** ¿se aplicaría la regla anterior sobre los siguientes códigos de Html?:

a) `Buscador`

b) `Buscador`

Selectores avanzados

2. Selectores avanzados:

- ✓ **Selector elementos hermanos:** es una **generalización** del selector **adyacente**.
 - Selecciona un determinado elemento si va **precedido** de otro, aunque **no** sea de forma **inmediata**.
 - Se representa utilizando un ~ entre dichos elementos.
 - **Sintaxis:** **elemento1 ~ elemento2** { propiedad:valor; }
 - Ej: **h1 ~ p** { color: orange; }

/* aplica esa característica sobre todos los elementos <p> si es hermano de <h1>, aunque **no** sean hermanos **consecutivos** */

Pseudo-Clases

- **Pseudo-Clase:** *estado especial del elemento o selector seleccionado*
- Permiten manipular aspectos de determinados elementos dependiendo de cómo el usuario **interactúe** con ellos => crear diferentes **reglas** de estilo **en función del estado en el que se encuentra dicho elemento**.
- Se utilizan para añadir **efectos dinámicos** a nuestra página web, ofrecer ayuda, mostrar diferente apariencia, etc.
- Algunos estilos de las pseudo-clases se **aplican** en **intervalos** muy **cortos**.
- **Sintaxis:** **selector:palabraclave** { propiedad:valor; }
- **Algunos pseudo-clases según el estado:**
 - ✓ **:link:** referido a aquellos **enlaces** aún **no visitados** y en reposo.
 - ✓ **:visited:** referido a los **enlaces** ya **visitados** (según el historial del navegador).
 - ✓ **:hover:** referido a los **enlaces** y otros **elementos** –p, div, ...-, cuando el puntero del **ratón** pasa por **encima**.
 - ✓ **:active:** cuando el puntero del ratón hace **clic** sobre cualquier **elemento**.
- **Ej:** **a:hover** {
 background-color: transparent;
 text-decoration: underline;
}

Pseudo-Clases

- Un mismo **elemento** puede verse **afectado** por **varias pseudo-clases** diferentes de forma **simultánea**.
- Las pseudo-clases se pueden **combinar**.
- **Ej:** cuando se pulsa un enlace que fue visitado previamente ¿qué pseudo-clases le afectan a dicho enlace?
- Debido a esta **característica** y al comportamiento en **cascada** de los estilos CSS, es importante cuidar el **orden** en el que se establecen las diferentes **reglas** utilizando **pseudo-clases**.
- **Orden correcto** para establecer **reglas** de estilo utilizando las cuatro pseudo-clases principales en un **enlace**:
 - 1) **a:link** { ... }
 - 2) **a:visited** { ... }
 - 3) **a:hover** { ... }
 - 4) **a:active** { ... }

Otras Pseudo-Clases

- **Otras pseudo-clases:**
 - **:required:** para seleccionar los elementos indicados por el selector que son obligatorios.
 - **:optional:** para seleccionar los elementos indicados por el selector que son opcionales (no obligatorios).
 - **:empty:** para seleccionar los elementos indicados por el selector que están vacíos.

pseudoclase	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 (<i>es</i> para español, <i>en</i> 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.

- **Ej: `input:focus:hover { background-color: yellow; }`**
/ poner amarillo el color de fondo cuando un control de tipo **input** tenga el foco y el ratón pase por encima de él */*

Otras Pseudo-Clases

■ De posición o jerárquicos:

elemento:nth-child(n)	<p>Se aplica al elemento indicado cuando sea el hijo número <i>n</i> de su elemento padre. <i>n</i> puede ser un número (por ejemplo 3, para el tercer hijo), o una expresión más compleja como:</p> <ul style="list-style-type: none"> • 2n+1. Se aplica a los hijos con número impar de orden • odd. Igual que la anterior • even. A los pares • 3n+1. Cada tres elementos hijos <p>Ejemplo:</p> <pre>tr:nth-child(2n+1) td{ background-color: green; }</pre> <p>Colorea de verde a los elementos <i>td</i> (celdas) que se encuentren en filas (elemento <i>tr</i>) cuyo número de hijo sea múltiplo de tres. Es decir pinta de verde una fila de cada tres.</p>
elemento:nth-last-child(n)	Igual que el anterior pero cuenta el orden de atrás hacia delante.
elemento:nth-of-type(n)	<p>Funciona como <i>nth-child</i> pero ahora se refiere al elemento número <i>n</i> (con <i>n</i> teniendo todas las posibilidades comentadas anteriormente) siendo <i>n</i> el número de hijo de ese tipo.</p> <p>Es decir, si tenemos dentro de <i>body</i> un párrafo de tipo <i>h1</i> y otro de tipo <i>p</i>; ambos son hijos de <i>body</i>, <i>p</i> sería el segundo hijo (<i>nth-child(2)</i>) pero es el primero de su tipo (<i>nth-of-type(1)</i>).</p>
elemento:nth-last-of-type(n)	Como el anterior pero cuenta <i>n</i> 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 tipo
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

Pseudo-Elementos

- **Pseudo-Elemento:** *partes de la página que no tienen etiqueta*
- Los pseudo-elementos son aquellas partes de una página web que no tienen ninguna etiqueta asociada.
 - **Ej:** primera línea de un párrafo, elementos que están antes, que están después, elementos que están seleccionados, etc.
- Se utilizan con elementos de **bloque** y las **celdas** de datos de las tablas.
- **Algunos Pseudo-Elementos:**
 - ✓ **:first-letter:** funciona como la **letra capital** en los procesadores de texto (diferenciar el aspecto de la primera letra del párrafo sobre el resto –mayor tamaño, negrita, ...-).
 - **Ej:** `p:first-letter {font-size: 30px; }`
 - ✓ **:first-line:** referido a la primera línea de un párrafo.
 - **Ej:** `p:first-line { text-transform: uppercase; }`
- Desde **CSS3** se cambia la **sintaxis** para **diferenciarlos** mejor de las **pseudo-clases**, añadiendo un doble :
 - **Sintaxis:** `::first-letter` y `::first-line`

Reglas básicas para aplicar estilos en cascada

- **Colisión de Estilos:**
- ¿Que ocurre cuando **dos reglas** se **aplican** al **mismo elemento** sobre una **misma propiedad** asignando **valores diferentes**? Sólo se puede aplicar **una** de ellas.
- Hay unas **normas básicas** que se deben tener en cuenta para aplicar la regla adecuada en cada caso, basándose en la **prioridad** de cada regla:
- **Reglas básicas para aplicar estilos en cascada:**
 - 1) **Importancia**
 - 2) **Especificidad**
 - 3) **Orden de aparición**
- Ya hemos visto el **orden** en el que se deben **aplicar las diferentes reglas** de estilo que afectan a un mismo elemento basándonos en la **importancia**:
 1. Formato **predefinido** del **navegador** (*estilos del navegador*): por sus hojas de estilo, y **después** según la **configuración del navegador** (usuario).
 2. **Diseñador**:
 - i. Estilos **externos** enlazados con **<link>**
 - ii. Estilos creados en la etiqueta **<style>**
 - iii. Estilos definidos en el atributo **style**.
 - iv. Alterar la preferencia de estilos utilizando **!important**

Reglas básicas para aplicar estilos en cascada

2) Especificidad: conjunto de *reglas* que indican el *peso* de un *selector* CSS

- El navegador sabe qué regla de estilo debe aplicar tras rastrear el documento en búsqueda de todas las **instrucciones CSS** que **afectan** una por una a **todas y cada una de las etiquetas Html** del documento.
- A cada **tipo de selector** se le aplica una **puntuación** determinada en base a unas determinadas **reglas**.
- El **selector** que sume **más puntos** es el que **gana**, y la declaración de estilos de ese selector es el que se visualizará en la página web.
- **Versión formal:** para asignar la **puntuación** se tiene en cuenta si aparecen determinados **elementos** en el **selector**:
 - A: Un 1 si existe el atributo **style**
 - B: Nº de id's.
 - C: Nº de selectores de atributo, de clases y pseudo-clases
 - D: Nº de elementos y pseudo-elementos
- Con los valores anteriores, se obtiene el número **ABCD**
- Orden de peso de columnas: **A > B > C > D**
- Ej1: `<p style="color:#000000;">` a=1, b=0, c=0, d=0 → 1000

Reglas básicas para aplicar estilos en cascada

2. Especificidad:

- **Ej:** Comprueba cómo se aplica la **especificidad** en el siguiente ejemplo en el que aparecen **reglas de estilo redundantes** que pueden generar **conflicto**:
 - a) Observa los distintos selectores con los que se puede llegar a la misma etiqueta Html.
 - b) ¿qué **colores** se le pueden asignar a cada una de las etiquetas ?
 - c) **Calcula la especificidad** de cada regla.
 - d) Según el cálculo anterior ¿qué color tendrá cada uno de esos elementos ?

```
<ul>  
  <li>Lista con selectores redundantes</li>  
  <li class="important">Con clase</li>  
  <li id="id1">Con id id1</li>  
  <li id="id2" class="important">Con id id2 y clase important</li>  
  <li id="id3" class="important" style="color:pink">Con id id3, clase important y atributo style</li>  
</ul>
```

```
#id1, #id2, #id3 {color:red;}  
ul li.important{color:green;}  
ul li {color:blue;}  
li.important{color:yellow;}  
li {color:purple;}
```

Reglas básicas para aplicar estilos en cascada

2. Especificidad:

➤ Sol.:

c) Calcula la **especificidad** de cada regla.

	A	B	C	D
#id1, #id2, #id3	0	3	0	0
ul li.important	0	0	1	2
ul li	0	0	0	2
li.important	0	0	1	1
li	0	0	0	1
style	1	0	0	0

d) Según el cálculo anterior ¿qué color tendrá cada uno de esos elementos ?

- Lista con selectores redundantes
- Con clase
- Con id id1
- Con id id2 y clase important
- Con id id3, clase important y atributo style

Reglas básicas para aplicar estilos en cascada

3) Orden de aparición

- A **igual especificidad** entre selectores, manda el **último**.
- El motivo es que **sobrescribe** al anterior.

❖ Mecanismo de resolución complejo de colisión de estilos:

1. Determinar **todas las declaraciones** que se aplican al mismo elemento
2. **Ordenarlas** según origen –estilos del navegador, usuario o del diseñador- y **prioridad** (!important).
3. **Ordenarlas** según lo **específico** que sea el **selector** => selector **más genérico** con **menor importancia**.
4. Si siguen existiendo reglas con la **misma prioridad** => aplicar la **última**.

¡¡Cuanto **más específico** sea un selector, **más importancia** tiene su regla asociada!!

❖ Recomendaciones para crear estilos CSS:

- 1) Normalmente, al definir los estilos de una página web se **empieza** por los **estilos más genéricos**.
 - 2) A medida que vamos avanzando en la hoja de estilos, se van añadiendo los **más específicos**.
- Gracias a estas recomendaciones, el código CSS es más **sencillo** de **mantener**.

Reglas básicas para aplicar estilos en cascada

- **Colisión de estilos => Reglas múltiples que se aplican al mismo elemento**
- **Ej:** Dado los siguientes códigos Html y Css respectivamente ¿de qué color se mostrarán los diferentes párrafos? ¿y el resto de elementos?

a) **Html:**

```
<p>...</p>
```

Css:

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

b) **Html:**

```
<article>  
<header>  
  <h2>Comprobando cómo se aplican reglas múltiples al mismo elemento</h2>  
  <h3>Colisión de estilos de párrafos:</h3>  
</header>  
<p id="especial">Párrafo especial</p>  
<p>Párrafo normal</p>  
....
```

Css:

```
p { color: red; }  
p#especial { color: green; }  
* { color: blue; }
```

Reglas básicas para aplicar estilos en cascada

- **Colisión de estilos => Reglas múltiples que se aplican al mismo elemento**
- **Ej:** Dado el siguiente código Html y Css ¿de qué color se mostrará el enlace al abrir la página? ¿y cuando se produce el estado hover? ¿y tras visitarlo?

Html:

```
<article>
  <header>
    <h2>Comprobando pseudoclases sobre enlaces</h2>
    <h3>Colisión de estilos de pseudoclases:</h3>
  </header>
  <p>Párrafo normal con un enlace <a href="">Primer enlace</a>.</p>
  <p id="especial">Párrafo especial</p>
  <p>Otro párrafo con un nuevo enlace <a href="">Otro enlace</a>.</p>
```

Css:

```
<style type="text/css">
  * { color: maroon; }
  a { text-decoration:none;}
  a:link { color:red; }
  a:hover { color:green; }
  a:visited { color:blue;}
</style>
```

Unidades de Medida

- **Unidades de Medida:**
- **Imprescindibles** para **definir** correctamente ciertas **propiedades** CSS: *altura, anchura, márgenes de los elementos, tamaño de letra del texto*, etc.
- Se puede indicar un valor con **diferentes medidas**: *píxeles, pulgadas, centímetros*, etc.
- **Dependiendo** del **dispositivo de salida** utilizado para mostrar el contenido de una página web, habrá un tipo de **unidades** de medida **más adecuado** que otro.
- La **unidad de medida** elegida se especifica inmediatamente **detrás del nº**.
- El nº puede ser **entero, decimal o negativo** (sólo para determinadas propiedades).
- Si el valor que se quiere asignar es **0** => la unidad de medida es **opcional** (no hay que añadirla).
- Si el valor es **diferente a 0** y **no** se especifica ninguna **unidad** => la medida **se ignora**
- **Sintaxis: númerounidad**
 - Ej: **p { font-size: 12pt; }** */* 12pt de tamaño de letra para los párrafos */*

➤ **Averigua:** ¿qué **tipos** de unidades de medida hay?

Unidades de Medida

- **Tipos de unidades de medida:**

- a) **Absolutas:** establecen de forma completa el valor de la medida

- Ej: *in, cm, mm, pt, pc*
- **Palabras clave para medidas absolutas:** se aplican **sólo** para definir el **tamaño del texto** con la propiedad ***font-size***:
 - Tamaño respecto del texto base –**medium**–
 - Ej: *xx-small, x-small, small, medium, large, x-large, xx-large*

- b) **Relativas:** definen su valor en relación a otra medida

- Ej: *em, ex, px*
- **Palabras clave para medidas relativas:** se aplican **sólo** para definir el **tamaño del texto** con la propiedad ***font-size***:
 - Ej: *smaller, larger*

- c) **Porcentuales:** tipo de medida **relativa**

- Ej: *valor%*

Unidades de Medida

a) **Absolutas:** establecen de forma completa el valor de la medida

- ✓ **in:** se corresponde con las **pulgadas** (*inches*)
 - **1in** se corresponde con **2,54cm** aproximadamente o **1 pulgada**
- ✓ **cm:** centímetros del sistema métrico decimal
- ✓ **mm:** milímetros del sistema métrico decimal
- ✓ **pt:** son puntos, equivale a **1pulgada/72 -0,35mm** aproximadamente-
 - La más utilizada como medida absoluta para **imprimir** (*media="print"*)
- ✓ **pc:** son las **picas**, equivalen a **12pt** o **4,23mm**
- **Ventajas:**
 - Al ser valor real, **no** se necesita realizar ningún **cálculo** intermedio.
- **Desventajas:**
 - Medida **poco flexible**.
 - NO se adaptan a los diferentes medios.
- Ej: **h1 { line-height: 2cm; }**
/ aplica 2cm al interlineado -espacio entre líneas- */*

Unidades de Medida

b) Relativas: para obtener su valor real, hay que hacer **operaciones** entre el valor indicado y el valor que se toma de **referencia**.

✓ **em:** ud. tipográfica, hace referencia al *tamaño en puntos* de la letra que se está utilizando.

- **1em** equivale a la **anchura** de la letra **M** del tipo y tamaño de letra del elemento (tipografía aplicada).

- Los navegadores, por defecto, muestran el texto de los **<p>** con un tamaño de **16px** como tamaño base.

- **Ej:** con una tipografía de **14pt**, **1em** equivale a 14pt.

✓ **ex:** ud. tipográfica, relativa respecto a la **altura** de la letra **x** del tipo y tamaño de letra del elemento.

- **Ej:** el valor de **1ex** se aproxima a **0,5em**.

✓ **px:** relativa respecto a la **resolución de la pantalla** del **dispositivo** en el que se **visualiza** la página web.

- **Ventajas:**

- Medida muy **flexible**, se **adapta** a los diferentes medios.

- Siempre **mantienen las proporciones** del diseño de la página.

- Mejoran la **accesibilidad** de la página.

Unidades de Medida

b) Relativas:

- Ej: `p { font-size: 32px; margin: 0.5em; }`

`/* aplica la mitad del tamaño de la letra para el margen de los párrafos. Si cambio el tamaño de la letra, el margen cambiará proporcionalmente */`

- ❖ Las unidades de medida se pueden **mezclar**:

- Ej: `body { font-size: 10px; }`

`h1 { font-size: 2.5em; }`

`/* aplica 10px como tamaño de letra base para toda la página, y para los h1 de 2.5em que equivalen a 25px */`

- ❖ **Herencia**: el valor de las **medidas relativas heredan** su valor ya calculado:

- Ej: ¿qué tabulación se aplica para toda la página? ¿y para los h1?

`body { font-size: 12px; text-indent: 3em; }`

`h1 { font-size: 15px; }`

`/* aplica una tabulación de 12*3=36px para toda la página, y para los h1 también -no se recalcula su valor relativo sobre su tamaño de letra base */`

Unidades de Medida

c) Porcentuales: tipo de medida relativa

- Asigna valores **respecto** del tamaño del **elemento padre**.
- Utiliza el símbolo %.
- Útiles para definir tamaños de *cajas*, *anchura* de una sección, tamaño de *texto*, etc.
- Ej: redimensionar el tamaño de una tabla con respecto al tamaño de la página.

```
table { width: 50%; }
```

```
/* el ancho de la tabla se corresponde con la mitad de la pantalla */
```

- Ej: redimensionar el tamaño de los h1 respecto del tamaño del texto de la página.

```
body { font-size: 1em; }
```

```
h1 { font-size: 200%; }
```

```
/* aplica 2em como tamaño de letra para los h1 */
```

❖ Recomendaciones:

- Utilizar unidades relativas siempre que sea posible por las ventajas.
- Ej: **px** y **%** para definir el **layout** del documento (**anchura** de columnas y de los elementos de las páginas), y **em** y **%** para el tamaño de **letra del texto**.

Color

- El color es una de las características que ayudan más a que las páginas web tengan un diseño atractivo.
- Se aplica a diferentes elementos Html: color de texto, color de fondo, de bordes, etc.
- **Formas de especificar el color:**
 - a) **Palabras clave**
 - b) **RGB decimal**
 - c) **RGB porcentual**
 - d) **RGB hexadecimal**
 - e) **Mediante función RGB y transparencia**
 - f) **Mediante función HSL**
 - g) **Mediante función HSLA**
 - h) **Colores web safe**
 - i) **Herramienta online y programas de diseño**

Color

a) Palabras clave

- Son **17** palabras clave con su **nombre estándar** para referirse a los **colores básicos**:
 - ✓ aqua - black – blue – fuchsia – gray – Green – lime – maroon – navy – olive
 - ✓ orange – purple – red – silver – teal – white - yellow

maroon #800000	red #ff0000	orange #ffa500	yellow #ffff00	olive #808000
purple #800080	fuchsia #ff00ff	white #ffffff	lime #00ff00	green #008000
navy #000080	blue #0000ff	aqua #00ffff	teal #008080	
black #000000	silver #c0c0c0	gray #808080		

- Ej: `p { color: maroon; }`
- **Ventajas:**
 - Método muy **sencillo** de elegir un color.
- **Desventajas:**
 - Gama de colores muy **limitada**.
 - No se utilizan colores puros.

Color

b) RGB decimal

- En el campo del diseño gráfico, se han definido varios **modelos** para hacer referencia a los colores:
- ❖ **Modelo CMYK**: ideal para **imprimir** en **cuatricromía**. Utiliza valores entre **0** y **100** de los colores *cyan-magenta-yellow-black* para crear cualquier color.
- ❖ **Modelo RGB**: ideal para definir colores en **pantalla**. Consiste en definir un color indicando la **cantidad de color rojo, verde y azul** que se debe **mezclar** para obtener dicho color.
- Es un modelo **aditivo** => los **colores** se obtienen **sumando** sus componentes.
- Cada **nivel de color** se define con un **nº** comprendido entre **0** y **255**.
 - **Sintaxis**: **rgb(r,g,b)**
 - **Ej1**: **p {color: rgb(122,22,255); }**
 - **Ej2**: para obtener el color **azul puro** se asigna el máximo valor al componente azul y 0 al resto de componentes **rgb(0,0,255)**
 - **Ej3**: para obtener el color **negro** todos los componentes valen **0** **rgb(0,0,0)**
 - **Ej4**: para obtener el color **blanco** todos los componentes valen **255** **rgb(255,255,255)**

c) RGB porcentual










- Funcionamiento **similar** al modelo **RGB decimal**.
- Define un color determinando la cantidad de color rojo, verde y azul empleando valores porcentuales: entre **0%** y **100%**.
- Se puede **convertir** un valor RGB decimal en un valor RGB porcentual con una regla de tres, sabiendo que el **valor 0** equivale a **0%** y el valor **255** equivale a **100%**.
 - **Sintaxis:** `rgb(valorR%, valorG%, valorB%)`
 - **Ej1:** `p {color: rgb(50%,22%,12%); }`
 - **Ej2:** `p {color: rgb(27%,38%,69%); }`
`/* equivale a rgb(71,98,176) */`

d) RGB hexadecimal

- Método **más utilizado**.
- Forma más **compacta** de indicar un color.
- Utiliza el **sistema hexadecimal** que utiliza hasta **16 símbolos** para representar los nº: de **0** a **9**, y de **A** a **F** (la **A** se corresponde con el nº **10**, la **B** con el **11**, etc.).
- Define un color determinando la cantidad de color rojo, verde y azul asignando un valor para cada componente.
- **Formas posibles:**
 - ❖ A través de un valor comprendido entre **00** y **FF** para **cada componente**.
 - **Sintaxis:** **#valorRvalorGvalorB**
 - Ej: **p { color: #80121A; }**
 - ❖ A través de un valor comprendido entre **0** y **F** para **cada componente comprimiendo** sus valores cuando todas sus componentes son **iguales dos a dos**.
 - Ej1: **#A0F** /* equivale a #AA00FF */
- Los programas de **diseño gráfico convierten** automáticamente los valores **RGB decimales** a sus valores **RGB hexadecimales**.

Color

d) Colores básicos en RGB hexadecimal y en RGB decimal:

Color	Color HEX	Color RGB
	#000000	rgb(0,0,0)
	#FF0000	rgb(255,0,0)
	#00FF00	rgb(0,255,0)
	#0000FF	rgb(0,0,255)
	#FFFF00	rgb(255,255,0)
	#00FFFF	rgb(0,255,255)
	#FF00FF	rgb(255,0,255)
	#C0C0C0	rgb(192,192,192)
	#FFFFFF	rgb(255,255,255)

e) Colores con nombre estándar y RGB Hexadecimal y Decimal (lista X11 colors):

HTML name	R G B	
	Hex	Decimal
Pink colors		
MediumVioletRed	C7 15 85	199 21 133
DeepPink	FF 14 93	255 20 147
PaleVioletRed	DB 70 93	219 112 147
HotPink	FF 69 B4	255 105 180
LightPink	FF B6 C1	255 182 193
Pink	FF C0 CB	255 192 203

Color

e) Mediante función RGB y transparencia

- Disponible en **CSS3**.
- Comprobar compatibilidad con navegadores.
- Utiliza en la función RGB un cuarto **parámetro** que es un **valor de transparencia** - *parámetro alpha*-.
 - Este parámetro puede tener un valor entre **0** y **1**:
 - **0.0: transparencia total**
 - **1.0: opacidad total**
- **Sintaxis:** `rgba(valorR,valorG,valorB,valorAlpha)`
 - Ej: `p { color: rgba(0,0,255,0.3); }`



f) Mediante función HSL

- Disponible en **CSS3**.
- Permite seleccionar colores utilizando **tres valores**:
 - ❖ **Tono** (*hue*): un valor de **0** a **360** que indica el **giro** en la **rueda de colores**.
 - El **0** es el **rojo**, el **120** el **verde** y el **240** el **azul**.
 - ❖ **Saturación** (*saturation*): un número del **0%** al **100%**, que indica el **nivel** de **saturación** del **color**.
 - Una saturación del **0%**, el color en **escala de grises**.
 - A mayor saturación, el **color** es **más vivo**.
 - ❖ **Luminosidad** (*lightness*): un número del **0%** al **100%**, que indica el **nivel** de **luminosidad**.
 - **Menos** luminoso **-0% negro-**.
 - **Más** luminoso significa más **claridad** para el color **-100% blanco-**
- **Sintaxis**: `hsl(valorTono,valorSaturación%,valorLuminosidad%)`
 - Ej: `p { color: hsl(120,90%,70%); }`

Color

g) Mediante función HSLA

- Disponible en **CSS3**.
- Añade al modelo anterior (*tono-saturación-luminosidad*) un cuarto valor para indicar el nivel ***alpha***: **transparencia** o **nivel de opacidad**.
- Este parámetro puede tener un valor entre **0** y **1**:
 - **0.0: transparencia total**
 - **1.0: opacidad total**
- **Sintaxis:** `hsla(valorTono,valorSaturación%,valorLuminosidad%)`
 - Ej: `p { color: hsl(120,90%,70%,0.4); }`



Color

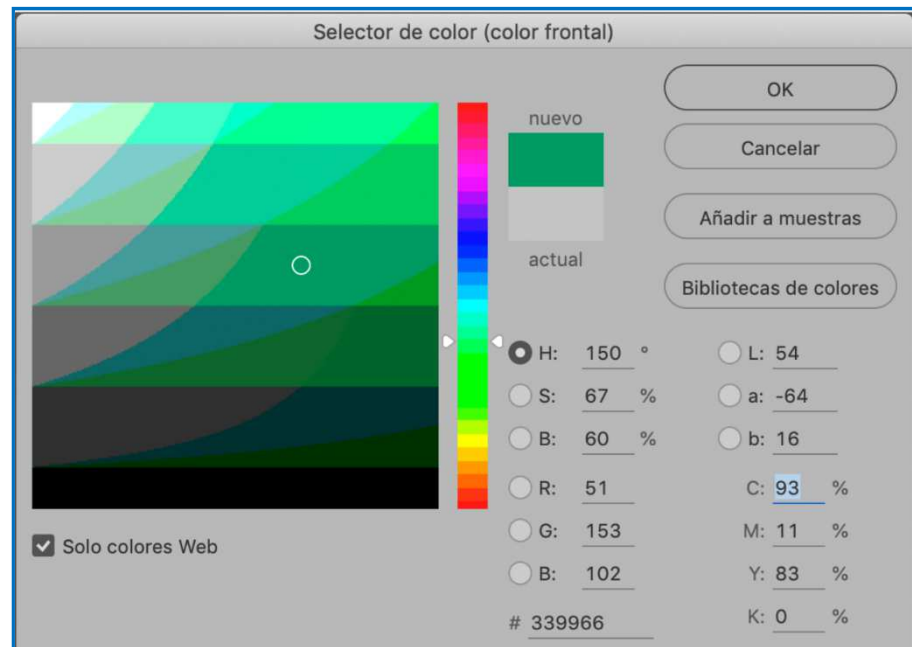
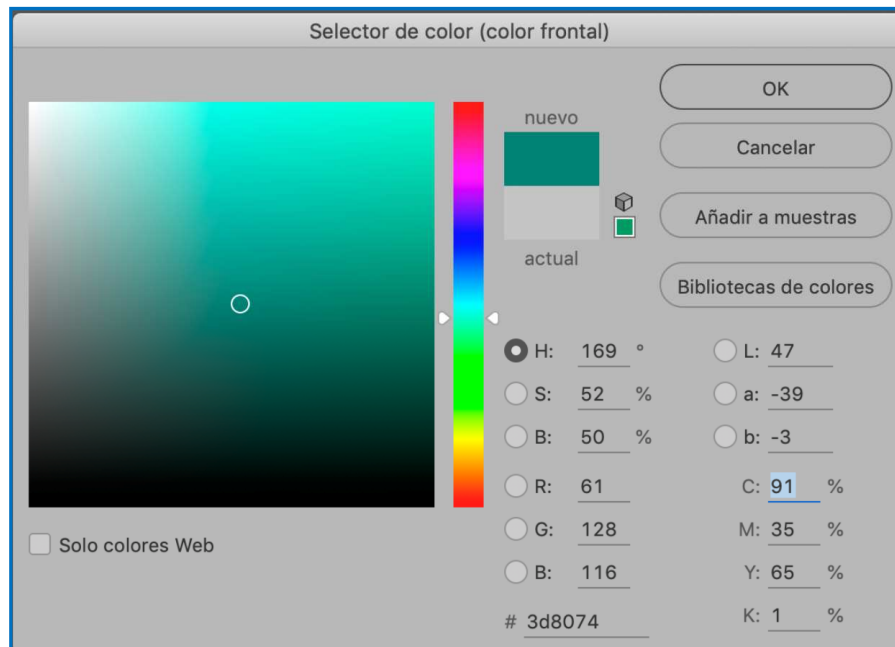
h) Colores web safe

- ¿Cuántos colores se pueden representar con el modelo RGB?
 - Cada **componente** puede tomar un valor entre **0 y 255** => **nº total de colores**
 $256 \times 256 \times 256 = \mathbf{16.777.216}$ colores.
 - Los primeros **monitores** de los usuarios no eran capaces de mostrar más de **256** colores diferentes.
 - A partir de todos los colores disponibles, **se eligieron 216 colores** que formaron la **paleta de colores "web safe"**. Esta paleta de colores podía ser utilizada por los diseñadores con la seguridad de que se **verían correctamente** en cualquier navegador de cualquier sistema operativo de cualquier usuario.
 - Hoy en día, su importancia ha descendido notablemente, ya que prácticamente todos los usuarios utilizan dispositivos con una profundidad de color de **16 y 32 bits**.
 - En muchos **dispositivos móviles** sólo se pueden representar un nº reducido de colores.
- **Averigua** cuáles son los **nombres de colores soportados actualmente** por los navegadores. Puedes utilizar esta dirección:
https://en.wikipedia.org/wiki/Web_colors#Web-safe_colors

Color

i) Herramientas online y programas de diseño

- Proporcionan los códigos hexadecimales directamente, **convirtiendo** de forma automática los valores de un **RGB decimal** a un **RGB hexadecimal**, facilitan el valor en el modelo **CMYK**, etc.
- **Programas de diseño:** *photoshop, illustrator, flash, ...*
- **Ej:** herramienta de color de Photoshop

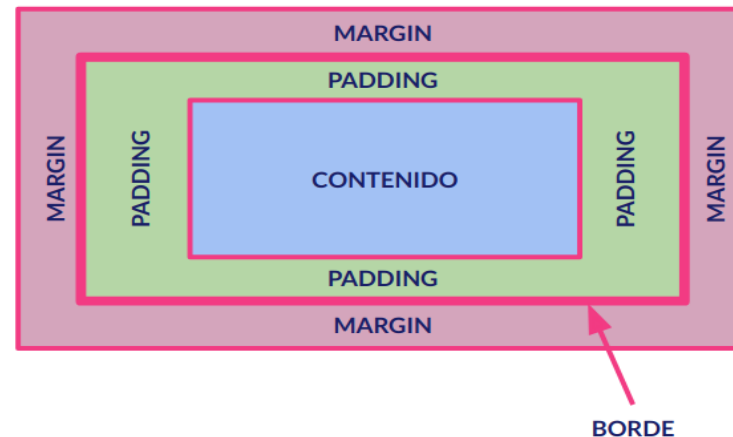


- **Busca** en Internet alguna **aplicación web** que permita seleccionar **colores** para el diseño de tus páginas web. Quédate con la que te parezca más sencilla de manejar. Comparte dicha información en el **foro de sugerencias**.

Modelo de Cajas

*todos los **elementos** incluidos en una página HTML se **representan** en el navegador mediante **cajas rectangulares***

- Es uno de los conceptos más **importantes** de CSS que **condiciona el diseño** de todas las páginas web. Gracias a CSS **controlaremos el aspecto** de dichas cajas.
- Los navegadores consideran **cada etiqueta** como una **caja, independientemente** de su **contenido** –texto, imágenes, ...-
- Todas las etiquetas HTML tienen asociadas multitud de **propiedades** que afectan a cómo se **muestran en los navegadores**:
 - ✓ Propiedades que **se ven** directamente: *bordes, color de fondo, ...*
 - ✓ Propiedades “**invisibles**”: *relleno, márgenes, etc.*, que controlan el **espacio vacío** que hay entre las diferentes etiquetas.
- **Visión sencilla del modelo de cajas:**



Modelo de Cajas

- No todas las **cajas iguales**, aunque los **navegadores** tratan a cualquier elemento como una **caja**.
- **Tipos de cajas:** *inline boxes* y *block boxes*.
- Se corresponden con los **elementos** *inline* y *block* respectivamente.

a) Elementos en *línea*:

- ✓ No crean ningún tipo de **salto**: mientras que haya **espacio suficiente**, se muestran en la **misma línea** que la etiqueta anterior.
- ✓ Si se redimensiona el tamaño de la ventana del navegador *“flotan”*.
- ✓ Pueden tener **márgenes** y **relleno** a **izquierda** y **derecha**.
- ✓ **NO** pueden tener **márgenes** y **relleno** **arriba** y **abajo**.
 - Ej: *strong, a, span, img, ...*

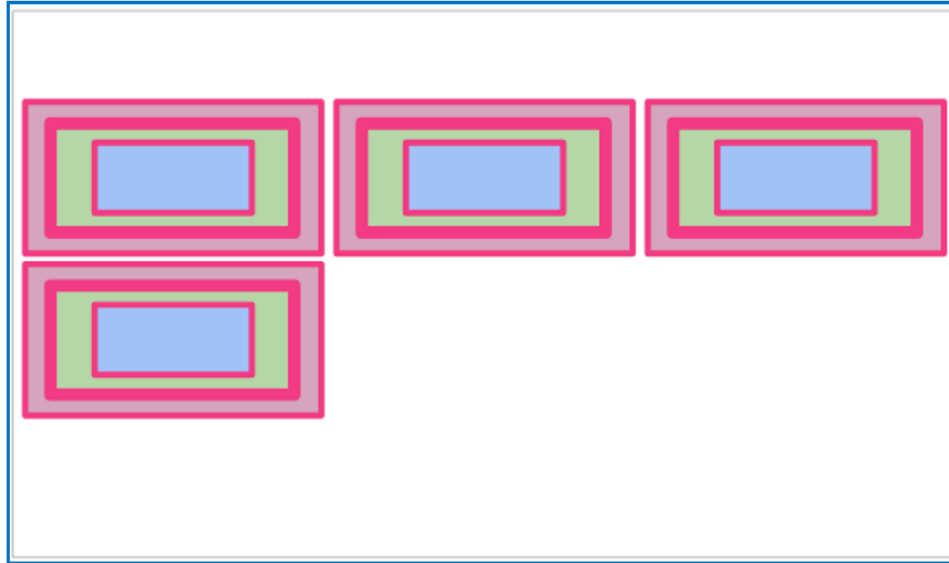
b) Etiquetas en *bloque*:

- Crean un salto de página después de insertarlos.
- Ej: *p, h1, ul, li, table, div, ...*

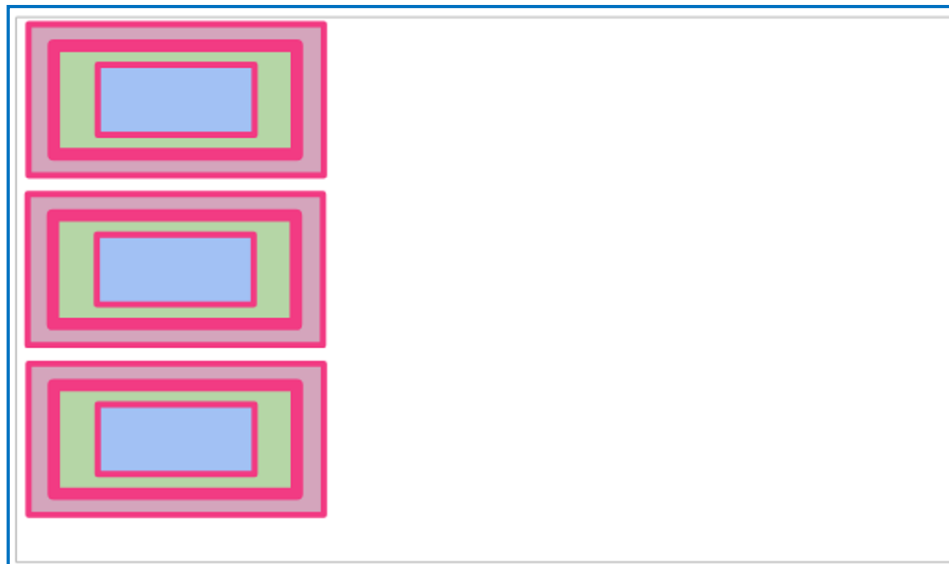
Modelo de Cajas

- Comportamiento de cajas:

a) *inline boxes*:

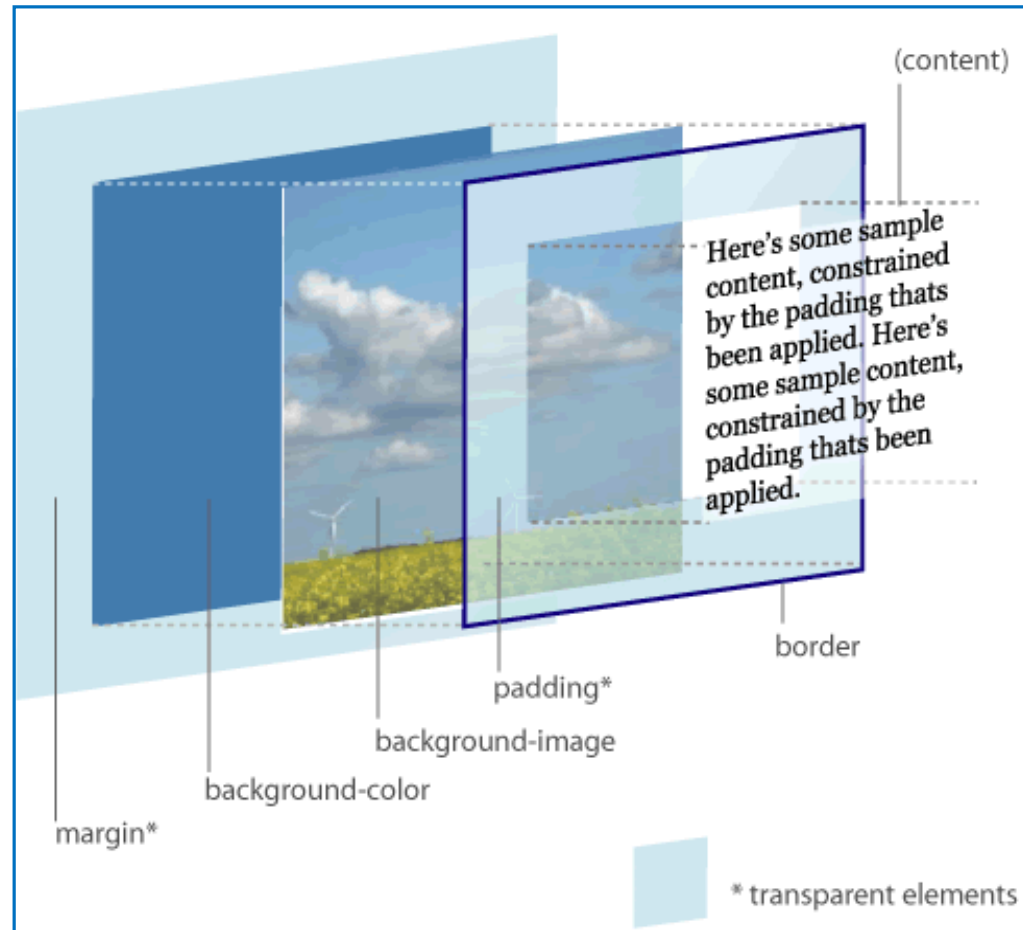


b) *block boxes*:



Modelo de Cajas

- Visión completa del modelo de cajas:



- Para las **propiedades** *padding*, *border* y *margin* hay que definir los cuatro lados: superior, derecho, inferior e izquierdo.

Modelo de Cajas

- **Boxmodel:**
- La **jerarquía** del modelo de cajas indica las **partes que componen cada caja** y su **orden de visualización**, desde el punto de vista del usuario:
 1. **Contenido (*content*)**: referido al texto o imagen que contiene la **etiqueta** HTML. Se muestra en el **primer plano**.
 2. **Relleno (*padding*)**: **espacio libre** entre contenido y **borde** de la caja. Es **opcional** y **transparente**.
 3. **Borde (*border*)**: **línea que se dibuja en cada lado de la caja**. Encierra el contenido y el relleno. Es **opcional** y tiene **diferentes formatos**, independientes por cada uno de los lados de la caja.
 4. **Imagen de fondo (*background image*)**: imagen que se muestra **por detrás del contenido y del espacio de relleno**. **Opcional**.
 5. **Color de fondo (*background color*)**: color que se muestra **por detrás del contenido y del espacio de relleno y de la imagen de fondo**. **Opcional**.
 6. **Margen (*margin*)**: **espacio libre** entre la **caja** y las posibles **cajas adyacentes**, es decir, el resto de elementos. **Opcional y transparente**.

Modelo de Cajas

- **Boxmodel:**

➤ **Observa** el siguiente ejemplo e indica qué elementos se visualizan:

Modelo de Cajas

... y tal y como podemos comprobar en nuestros [Horario de Clases](#) la característica principal de esta distribución es la de conseguir una perspectiva general de todas las materias que se imparten, con el objetivo principal de alcanzar una interrelación curricular que permita a los alumnos poseer las competencias básicas necesarias para afrontar un nivel de autonomía y de interrelaciones personales aptos para vivir en sociedad, ...

... Fijándonos en el código que se muestra desde [listing 4](#) podemos ver perfectamente la interface primaria de una declaración API que ... Recordar que la API está diseñada para utilizar la codificación UTF-8.

El genial historiador Manuel Fernández Álvarez, consiguió ... Recordamos una de sus maravillosas obras destinada a público infantil, y no tan infantil, que permite acercar la historia de nuestra querida España al público en general (ver las ilustraciones dedicadas desde [Algunas Obras](#)).

Horario de Clases

	Lunes	Martes	Miércoles	Jueves	Viernes
10:30	Matemáticas	Geografía	Física	Dibujo	Matemáticas
11:30	Inglés	Lenguaje	Geografía	Química	Física

Modelo de Cajas

- **Html:**

```
<h2 id="inicio">Modelo de Cajas</h2>
<p>... y tal y como podemos comprobar en nuestros <a href="#horarios"><b>Horario de Clases</b></a> la característic
</br>
<p id="margen">... Fijándonos en el código que se muestra desde <a href="#l4"><b>listing 4</b></a> podemos ver perf
</br>
<p>El genial historiador Manuel Fernández Alvarez, consiguió ... Recordamos una de sus maravillosas obras destinada
</br>
<figure id="horarios">
  <figcaption><b>Horario de Clases</b></figcaption>
<table border="2">
  <tr>
    <th></th>
    <th>Lunes</th>
```

- **Reglas CSS:**

```
/*aplicar bordes y relleno a las etiquetas principales */
body, h1, h2, p, a, table, figure, figcaption, img, pre {
    border-width: medium;
    border-color: #00AA55;
    border-style: dotted;
    padding: 1em;
}

/*aplicar márgenes al elemento con el id "margen" */
#margen{
    margin: 2em;
    background-color: #00AA55;
    border-style: solid;
    border-color: #FFAA55;
    border-width: .75em;
}
```

Propiedades del Modelo de Cajas

- ❖ Anchura (*width*) y altura (*height*)
- ❖ Margen (*margin*) y relleno (*padding*)
 - ✓ *margin-top, margin-right, margin-bottom, margin-left // margin (shorthand)*
 - ✓ *padding-top, padding-right, padding-bottom, padding-left // padding (shorthand)*
- ❖ Bordes (*border*)
 1. Anchura o grosor:
 - ✓ *border-top-width, border-right-width, border-bottom-width, border-left-width // border-width (shorthand)*
 2. Color:
 - ✓ *border-top-color, border-right-color, border-bottom-color, border-left-color // border-color (shorthand)*
 3. Estilo:
 - ✓ *border-top-style, border-right-style, border-bottom-style, border-left-style // border-style (shorthand)*
 - Para definir el **estilo completo** de los cuatro bordes:
 - ✓ *border-top, border-right, border-bottom, border-left*
 - ✓ *border (shorthand)*

Propiedades del Modelo de Cajas

- ❖ Bordes redondeados (***border-radius***)
 - ✓ *border-top-left-radius, border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius*
 - ✓ *border-radius* (***shorthand***)
- ❖ Bordes con imágenes (***border-image***)
 - ✓ *border-image* (***shorthand***)
- ❖ Color (***background-color***) e imagen de fondo (***background-image***)
 - ✓ *background-repeat, background-attachment, background-position*
 - ✓ *background* (***shorthand***)

Propiedades del Modelo de Cajas

- **Propiedades del modelo de cajas:**
- ❖ **Anchura** (*width*) y **altura** (*height*)
- Muy útiles para asignar una determinada **dimensión** a diferentes objetos: tablas, columnas, barra de navegación, ...
- ❖ **Anchura:**
- **Descripción:** establece la **anchura** de un elemento (*contenido*).
- **Valor inicial:** *auto*
- **Se aplica a:** todos los elementos, **excepto** los **elementos en línea** que **no** sean imágenes, filas de tabla y grupos de filas de tabla.
- **Valores:**
 - ✓ **Medida:** absoluta, relativa o porcentaje.
 - ✓ **Auto:** para que tenga una anchura automática calculada por el navegador.
 - ✓ **Inherit:** se hereda del elemento padre.
- **Ej:** `h2 { width:200px; }`

Propiedades del Modelo de Cajas

❖ **Altura** *height*:

- **Descripción:** establece la **altura** de un elemento (*contenido*).
- **Valor inicial:** *auto*
- **Se aplica a:** todos los elementos, **excepto** los **elementos en línea** que **no** sean imágenes, filas de tabla y grupos de filas de tabla.
- **Valores:**
 - ✓ **Medida:** absoluta, relativa o porcentaje.
 - ✓ **Auto:** para que tenga una altura automática calculada por el navegador.
 - ✓ **Inherit:** se hereda del elemento padre.
- **Ej:** `h2 {height:100px; }`

Propiedades del Modelo de Cajas

❖ Margen (*margin*) y relleno (*padding*)

- Se puede **gestionar** el margen/relleno de cada lado de la caja de dos **formas**:

a) **De forma independiente**: hay una propiedad específica para cada lado

✓ *margin-top, margin-right, margin-bottom, margin-left*

✓ *padding-top, padding-right, padding-bottom, padding-left*

b) **De forma conjunta**: con la propiedad *shorthand*, propiedad **especial** que permite establecer en una **única instrucción** el **valor** de **todos** los márgenes/relleno

✓ *margin*

✓ *padding*

▪ Ej:

```
p {  
    margin-right: 20px;  
    padding-top: 3em;  
    margin-left: 10%; }
```

➤ ¿**Recuerdas** qué elemento de Html permite escribir un poco más hacia la derecha y no llegar hasta el final? Averigua qué hace **<blockquote>**

Propiedades del Modelo de Cajas

❖ *margin-top, margin-right, margin-bottom, margin-left:*

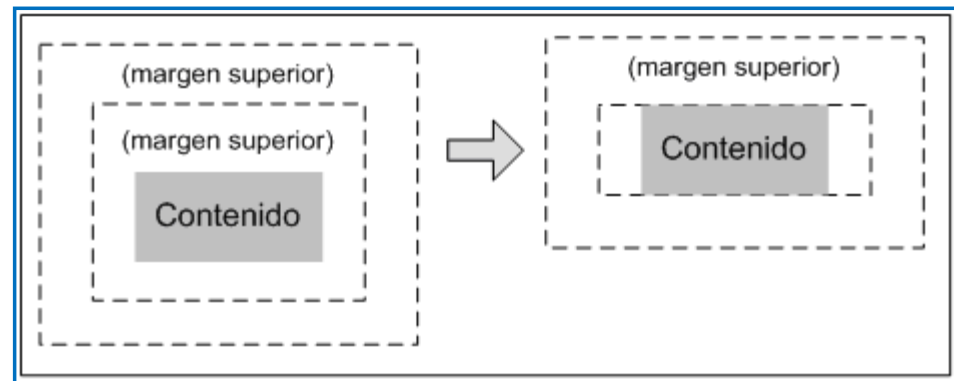
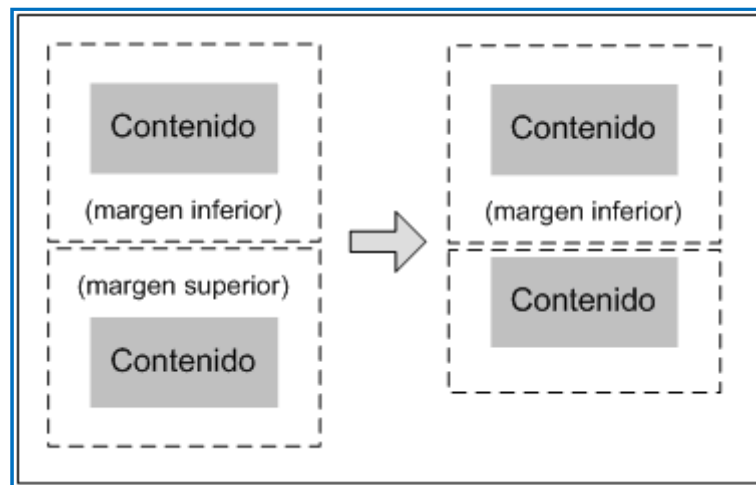
- **Descripción:** establece cada uno de los **márgenes horizontales y verticales** de un elemento (contenido).
- **Valor inicial:** 0
- **Se aplica a:** todos los elementos, excepto *margin-top* y *margin-bottom* que sólo se aplican a los **elementos en bloque** y a las **imágenes**.
- **Uds. de medida:** las más utilizadas son **px, em y %**.

❖ **Margen** (*margin*)

- **Descripción:** establece cada uno de los márgenes horizontales y verticales de un elemento (contenido). Admite entre uno y cuatro **valores**:
- Si sólo se indica **1 valor** => todos los márgenes tienen ese valor.
- Si se indican **2 valores** => el 1º se asigna al **margen superior e inferior** y el 2º a los márgenes **izquierdo y derecho**.
- Si se indican **3** => el 1º se asigna al margen **superior**, el 2º valor se asigna los márgenes **izquierdo y derecho**, y el 3º al margen **inferior**.
- Si se indican los 4 => **orden** de asignación: **sentido de las agujas del reloj**.
- **Ej:** **p {margin: 0 10px 10px 20px; }**

Propiedades del Modelo de Cajas

- **Comportamiento de márgenes verticales colindantes:** *margin-top* y *margin-bottom*
- Es complicado.
- Se aplica automáticamente una **fusión** de valores.
- Para conseguir espacio adicional extra, utilizar la propiedad de *relleno*.
- **Mecanismo de fusión:**
 - Permite dar uniformidad y **aspecto homogéneo** a las páginas web.
 - Se aplica entre los valores de los **márgenes verticales**, independientemente de que los **elementos** sean **consecutivos** o estén **anidados** =>
los márgenes se **fusionarán** y resultará como **margen el del valor superior**.
 - **Ej:** fusión automática de márgenes verticales



Propiedades del Modelo de Cajas

❖ *padding-top, padding-right, padding-bottom, padding-left:*

- **Descripción:** establece cada uno de los **rellenos horizontales y verticales** de un elemento, es decir, define la **separación** entre el **contenido** y los **bordes** de la caja del elemento.
- **Valor inicial:** 0
- **Se aplica a:** todos los elementos, excepto algunos elementos de tablas, como grupos de cabeceras y grupos de pies de tabla.
- **Valores:**
 - ✓ **Medida:** absoluta, relativa o porcentaje.
 - ✓ **Auto:** para que tenga una altura automática calculada por el navegador.
 - ✓ **Inherit:** se hereda del elemento padre.
- **Ej:** `h2 { padding-top: 3em; }`

❖ **Relleno** (*padding*)

- Funciona exactamente igual que *margin*.
- **Ej:** `h2 { padding: 1em 2em; }`
/* aplica 1em al relleno superior e inferior, y de 2em al izquierdo y derecho */

Propiedades del Modelo de Cajas

❖ Bordes (*border*)

- Hay varias **propiedades** para modificar el aspecto de los bordes de las cajas:

1. Anchura o grosor:

❖ *border-top-width, border-right-width, border-bottom-width, border-left-width*

- **Descripción:** establece la **anchura** de cada uno de los **bordes** de la caja del elemento.
- **Valor inicial:** *medium*
- **Se aplica a:** todos los elementos.
- **Valores:**
 - ✓ **Medida:** absoluta, relativa o palabra clave (*thin, medium, thick*).
 - ✓ **Inherit:** se hereda del elemento padre.
- Ej: `h2 {border-top-width: 10px; border-bottom-width: thick; }`

❖ *border-width (shorthand)*

- Funciona exactamente igual que las otras propiedades *shorthand*.
- Ej: `h2 {border-width: thin thick medium; }`

/ aplica un borde thin al borde superior, un borde thick a borde derecho e izquierdo, y un borde medium al borde inferior */*

Propiedades del Modelo de Cajas

❖ Bordes (*border*)

2. Color:

❖ *border-top-color, border-right-color, border-bottom-color, border-left-color*

- **Descripción:** establece el **color** de cada uno de los **bordes** de la caja del elemento.
- **Valor inicial:** el color del elemento en sí
- **Se aplica a:** todos los elementos.
- **Valores:**
 - ✓ **Color:** palabra reservada, RGB decimal, RGB hexadecimal, etc.
 - ✓ **Transparent:** el borde no aparece pero ocupa el sitio definido.
 - ✓ **Inherit:** se hereda del elemento padre.
- **Ej:** `h2 {border-top-color: blue; border-bottom-color: #CCC; }`
- ❖ *border-color* (*shorthand*)
 - Funciona exactamente igual que las otras propiedades *shorthand*.
 - **Ej:** `h2 {border-color: #00AA55; }`

Propiedades del Modelo de Cajas

❖ Bordes (*border*)

3. Estilo:

❖ *border-top-style, border-right-style, border-bottom-style, border-left-style*

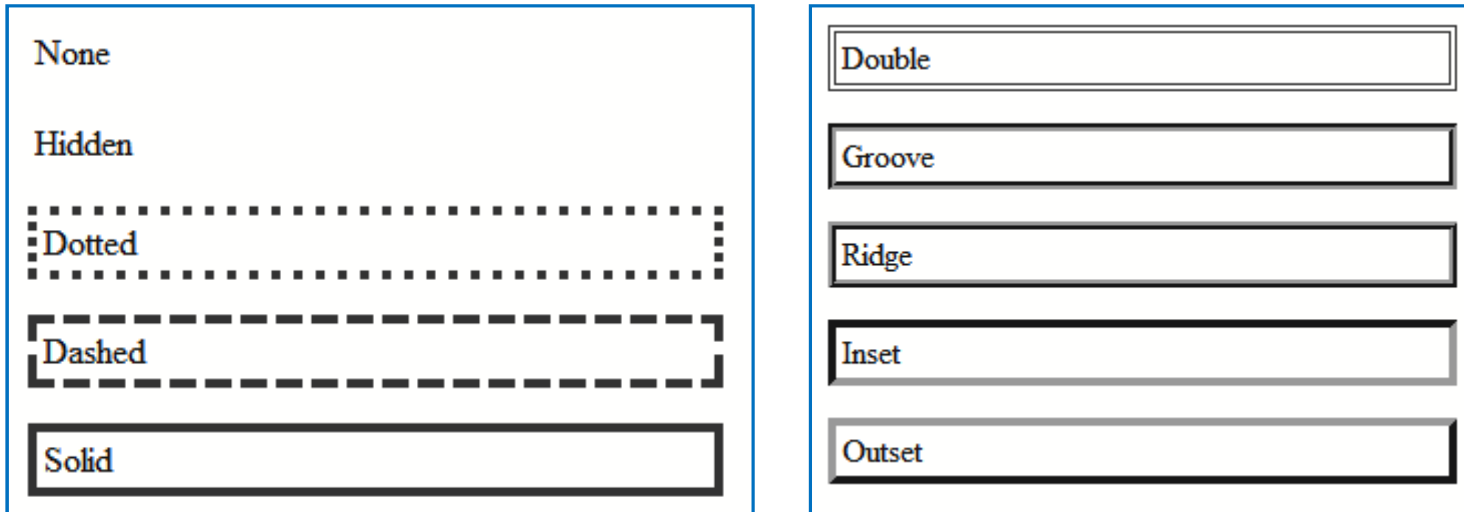
- **Descripción:** establece el **estilo** de cada uno de los **bordes** de la caja.
- **Valor inicial:** *none*
- **Se aplica a:** todos los elementos.
- **Valores:**
 - ✓ **none:** sin borde, no aplica ningún tipo de línea al borde.
 - ✓ **hidden:** visualmente como *none*, pero internamente los navegadores resuelven de diferente forma los conflictos entre celdas adyacentes.
 - ✓ **dotted:** punteado
 - ✓ **dashed:** discontinuo (punteado más largo).
 - ✓ **solid:** línea continua sólida
 - ✓ **double:** línea doble
 - ✓ **groove:** borde 3D con efecto de hundimiento -colores negro y gris-
 - ✓ **ridge:** borde 3D con efecto de elevación -al revés que *groove*-
 - ✓ **inset:** efecto 3D de esquinas inferiores
 - ✓ **outset:** intercambia los colores de *inset*
 - ✓ **inherit**

Propiedades del Modelo de Cajas

❖ Bordes (*border*)

3. Estilo:

■ Tipos de bordes:



- Ej: `h2 {border-top-style: dashed; border-bottom-style: dotted; }`

❖ *border-style* (*shorthand*)

- Funciona exactamente igual que las otras propiedades *shorthand*.
- Ej: `h2 {border-style: double; }`

Propiedades del Modelo de Cajas

❖ Bordes (*border*)

- Para definir el **estilo completo** de los bordes de forma directa -**grosor, estilo y color**- de los **cuatro bordes** con propiedades *shorthand*:

❖ *border-top, border-right, border-bottom, border-left*

- **Descripción:** establece el **estilo parcial o completo** de cada uno de los **bordes** de la caja.
- **Se aplica a:** todos los elementos.
- **Valores:** grosor_borde color_borde estilo_borde. Es **indiferente el orden**.
- **Ej1:** `h2 { border-top: 3px solid silver; }`
- **Ej2:** `h2 { border-bottom: double orange; }`

/ al no indicar la anchura del borde, toma su valor por defecto –medium- */*

❖ *border (shorthand)*

- **Descripción:** propiedad **global** que establece el **estilo parcial o completo** de **todos los bordes** de la caja.
- **Se aplica a:** todos los elementos.
- **Valores:** grosor_borde color_borde estilo_borde. Es **indiferente el orden**.
- **Ej:** `h2 { border: 3px solid silver; }`

Propiedades del Modelo de Cajas

❖ Bordes redondeados (***border-radius***)

- Nueva propiedad en **CSS3**.
- Crea bordes redondeados: las **esquinas de los bordes** forman un **círculo** con el **tamaño de radio** que especifiquemos.

❖ ***border-top-left-radius, border-top-right-radius, border-bottom-left-radius, border-bottom-right-radius***

- **Descripción:** propiedades específicas para definir cada vértice del borde.
- **Se aplica a:** todos los elementos.
- **Valores:** indicar la longitud del radio en las diferentes unidades de medida
 - Para esquinas **circulares**: asignar un valor
 - Para esquinas **elípticas**: asignar un **valor/otrovalor** o en %
- Ej: `p { border: solid #F23; border-top-left-radius: 10px; border-bottom-right-radius: 1em; border-bottom-left-radius: 15%; }`

❖ ***border-radius*** (***shorthand***)

- Indica a la vez varios **tamaños de redondeo** en el orden de izquierda a derecha y de arriba abajo.
- Ej: `p { border: solid #F23; border-radius: 10px 5px 0px 20px; }`

Propiedades del Modelo de Cajas

❖ Bordes con imagen (*border-image*)

- Nueva propiedad en **CSS3**.
- Crea una imagen en formato **.png** para que se pinten los bordes de la siguiente forma:
 - La imagen tiene coloreado sus bordes. El resto de la imagen son píxeles transparentes que permiten ver el fondo.
 - Se **estira** la imagen en la zona que no son las esquinas para cubrir todo el tamaño del elemento que se está bordeando con dicha imagen.
 - Hay que indicar la **parte** de la **imagen** dedicada a las **esquinas** usando su tamaño original. Al estar definida la imagen en píxeles, No se indica la ud.
 - Para los lados, indicar al navegador que **repita** cada elemento (**round**) o que lo **estire** (**stretch**). Puede ser diferente en horizontal que en vertical (indicar ambos valores).
- **Sintaxis:** `border-image: URL_Imagen tamañoEsquinas modoEstirar;`
- Ej:

```
p { border-width: 15px;
      border-image: url(borde2.png) 18 19 round; }
```
- Si quieres ampliar cómo funciona esta propiedad, lee el siguiente artículo:
<https://lenguajecss.com/css/modelo-de-cajas/bordes-imagenes/>

Propiedades del Modelo de Cajas

❖ Color (*background-color*) e imagen de fondo (*background-image*)

- Se pueden aplicar a cualquier elemento.
- Es habitual añadirlos al elemento mayor que contiene al resto: **body**

❖ *background-color*

- **Descripción:** propiedad para definir un **color** de fondo.
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *transparent*
- **Valores:** cualquier forma para especificar el color (palabra reservada, rgb decimal, rgb hexadecimal, ...) o indicar q sea *transparent* o *inherit*.
- Ej: `body { background-color: #00AA55; }`

❖ *background-image*

- **Descripción:** propiedad para definir una **imagen** de fondo.
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *none*
- **Valores:** indicar la dirección **URL** con **url()** de dónde está la imagen a cargar o indicar q sea *none* o *inherit*.
- Ej: `body { background-img: url(ruta/imagen.jpg); }`

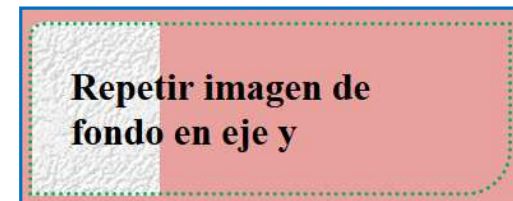
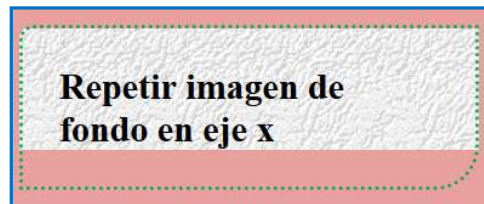
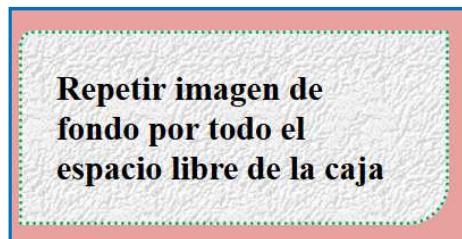


Propiedades del Modelo de Cajas

❖ Color (***background-color***) e imagen de fondo (***background-image***)

❖ ***background-repeat***

- **Descripción:** especifica cómo queremos que se **rellene** el **espacio libre** que no ocupa la **imagen** de fondo, controlando la forma en la que se **repite** la imagen de fondo.
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *repeat*
- **Valores:**
 - ✓ **repeat:** rellena el espacio repitiendo la imagen (mosaico)
 - ✓ **repeat-x:** se repite de izquierda a derecha (eje x)
 - ✓ **repeat-y:** se repite de arriba abajo (eje y)
 - ✓ **no-repeat:** no se repite
 - ✓ **inherit**
- **Ej:** `h3 { background-img: url(ruta/imagen.jpg); background-repeat: repeat-y; }`



Propiedades del Modelo de Cajas

❖ Color (**background-color**) e imagen de fondo (**background-image**)

❖ **background-attachment**

- **Descripción:** propiedad para indicar cómo se comporta la **imagen** de fondo, controlando el **desplazamiento** del **fondo**: ¿la imagen de fondo se va a mover igual que el resto de la página o no?.
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *scroll*
- **Valores:**
 - ✓ **scroll:** la imagen de fondo y el texto se mueven juntos cuando el usuario se desplaza por la página
 - ✓ **fixed:** la imagen de fondo es siempre fija aunque se desplace la ventana
- Ej:

```
p { background-attachment: fixed;
```

```
...  
}
```



Propiedades del Modelo de Cajas

❖ Color (**background-color**) e imagen de fondo (**background-image**)

❖ **background-position**

- **Descripción:** propiedad para controlar la **posición** que ocupa la **imagen** de fondo. La posición original es la **esquina superior izquierda**.
- **Valor inicial:** 0% 0% -posición horizontal y vertical-
 - **posicionHorizontal:** indicar una coordenada horizontal concreta (en px o %) o utilizar las siguientes palabras:
 - ✓ **left:** equivale a una posición del 0%
 - ✓ **right:** equivale a una posición del 100%
 - ✓ **middle:** equivale a una posición del 50%
 - **posicionVertical:** indicar una coordenada vertical concreta (en px o %) o utilizar las siguientes palabras:
 - ✓ **top:** equivale a una posición del 0%
 - ✓ **bottom:** equivale a una posición del 100%
 - ✓ **center:** equivale a una posición del 50%
- **Ej:**

```
body { background-position: right 20%;  
    ...  
}
```



Propiedades del Modelo de Cajas

❖ Color (*background-color*) e imagen de fondo (*background-image*)

❖ *background* (*shorthand*)

- **Descripción:** fija en una sola propiedad **todas las propiedades del fondo**. El **orden** habitual es **color, imagen, repetición, comportamiento y posición**
- Es **indiferente** el **orden** de las propiedades.
- Tampoco es necesario indicar todos los atributos.
- **Sintaxis:** { background: background-color background-image background-repeat background-attachment background-position; }
- **Ej:** `body { background: maroon url('fondo1.gif') no-repeat fixed left bottom ; }`

➤ Observa las siguientes reglas e indica cómo se visualiza la imagen indicada. ¿Son equivalentes?:

- `background: url(fondo1.png);`
- `background: transparent url(fondo1.png) repeat scroll left top;`

Propiedades del Modelo de Cajas

- ❖ Color (*background-color*) e imagen de fondo (*background-image*)
 - Conflictos entre background y propiedades específicas de background
 - Los conflictos que se pueden producir en el uso de estas propiedades, se resuelven de la misma forma que el resto de conflictos entre reglas CSS.
 - **Ej1:** en mi hoja de estilos defino las siguientes declaraciones en una o varias reglas que se aplican al mismo elemento y en este orden. ¿Cuál será el resultado de aplicarlas?
 - a) El resultado es que sobre ese elemento se aplica un color de fondo amarillo y la imagen logo1.
 - b) El resultado es que sobre ese elemento se aplica la imagen logo1
 - c) Si cambio el orden de las reglas el resultado es ...
 - **background-color: yellow;**
 - **background: url('logo1.jpg') no-repeat ;**

Propiedades del Modelo de Cajas

- ❖ Color (*background-color*) e imagen de fondo (*background-image*)
 - Conflictos entre **background** y propiedades específicas de **background**
 - **Ej2:** en mi hoja de estilos defino las siguientes declaraciones en varias reglas para aplicarlas un determinado elemento. ¿A qué elemento afecta la 1ª regla? ¿y la 2ª?. ¿Cuál será el resultado de aplicarlas? ¿y si se cambian de orden?
 - **p { background: url(icon.png) left top no-repeat rgb(0,30,0); }**
 - **h2 + p { background: blue; }**
- **Deduce:** ¿cuáles son las ventajas de utilizar las propiedades *shorthand*? ¿y los inconvenientes?

Gradientes de Color

- ❖ **Gradiente:** *degradado que permite colorear con una variación de colores, yendo de un color a otro (mínimo 2 colores)*
- **Antes:** se creaba el degradado desde aplicaciones de **diseño gráfico** y se asignaba esa imagen a *background-image*.
- **Ahora:** se generan los diferentes gradientes de color a través de código, e igualmente se asigna a *background-image*.
- **Nueva función** en CSS3
- **Tipos:**
 - ✓ **Lineal:** degradado de un color hacia otro partiendo de un punto
 - ✓ **Radial:** deforma el degradado en forma radial –círculo o elipse- desde un punto central hacia el exterior.
- **Compatibilidad con navegadores:** uso de **prefijos**
 - ✓ **-webkit-:** para Safari, Chrome, nuevas versiones de Ópera, Firefox para iOS
 - ✓ **-moz-:** para Firefox (resto de s.o.)
 - ✓ **-o-:** antiguas versiones de Ópera
 - ✓ **-ms-:** para Internet Explorer y Microsoft Edge

Gradientes de Color

❖ Gradiente:

▪ Tipos:

✓ **Lineal:** degradado de un color hacia otro partiendo de un punto

▪ **Sintaxis:** `linear-gradient(inicio, angulo, color1, color2, [...]);`

▪ **Inicio:** punto de inicio o comienzo de degradado. En **px** o en una posición de pantalla –*top, left, bottom, right, ...*–

▪ **Ángulo:** ángulo en el que queremos que se muestre el degradado –con una pequeña inclinación o no-. Los grados se expresan con **deg** –degree-.

▪ **Color 1:** color **origen** a degradar

▪ **Color 2:** color **intermedio**

▪ **Color n:** color **final**

▪ Ej1: `linear-gradient(bottom, 0deg, #FF1122, #33EE11, #1111FF);`

▪ Ej2: `linear-gradient(top, red 0%, orange 20%, yellow 80%, violet 100%);`

▪ Ej1: `-webkit-linear-gradient(145deg, red, blue, yellow);`

Gradientes de Color

❖ Gradiente:

▪ Tipos:

- ✓ **Radial:** deforma el degradado en forma radial –círculo o elipse- desde un punto central hacia el exterior.

- **Sintaxis:** `radial-gradient(forma, color1, color2, [...]);`

- **Forma:** circular o elíptica (*circle* | *ellipse*)

- **Color 1:** color **origen** a degradar

- **Color 2:** color **intermedio**

- **Color n:** color **final**

- Ej1: `radial-gradient(circle, #FF1122, #33EE11, #1111FF);`

- Ej2: `radial-gradient(ellipse, #FF1122, #33EE11, #1111FF);`

- Ej3: `-moz-radial-gradient(ellipse, #FF1122, #33EE11, #1111FF);`

- **Comprueba:** para saber cuándo debo poner el prefijo a una determinada propiedad CSS consultar la página ***shouldiprefix.com***. Muestra el listado de las propiedades y según el color asignado, es obligatorio o no añadirlo.

Gradientes de Color

❖ Gradiente:

▪ Ej:

... y tal y como podemos comprobar en nuestros **Horarios de Clases**, la característica principal de esta distribución es la de conseguir una perspectiva **general de todas las materias** que se imparten, con el objetivo principal de alcanzar una interrelación curricular que permita a los alumnos poseer las competencias básicas necesarias para afrontar un nivel de autonomía y de interrelaciones personales aptos para vivir en sociedad, ...

... Fijándonos en el código que se muestra desde [listing 4](#) podemos ver perfectamente la interface primaria de una declaración API que ...
Recordar que la API está diseñada para utilizar la codificación UTF-8.

Imágenes con Derechos de Autor ©

Generadores online de degradados de Color

- **Generadores online de degradados de color:**
- Herramientas muy útiles para editar de forma gráfica los gradientes.
- Se puede construir un degradado de color y **copiar el código** que genera automáticamente.
- **Ejemplos:**
 - ✓ colorzilla.com/gradient-editor
 - ✓ gradients.glrzad.com
 - ✓ css3factory.com/linear-gradients/
 - Hacen degradados de 2 colores => código resultante más sencillo.
 - ✓ css3generator.com
 - Permite elegir cualquier instrucción de CSS3 novedosa y copiar el código correspondiente.
 - Se puede **previsualizar** el efecto elegido.
 - Indica la versión necesaria de cada navegador para que muestre dicha propiedad.

Posicionamiento y Visualización

- ❖ **Posicionamiento:** *cómo los navegadores crean y posicionan las cajas en pantalla de forma automática (modelo de cajas)*
- El proceso de **posicionamiento** y **visualización** de los diferentes elementos en pantalla es muy **complejo**.
- **CSS** permite **modificar** la **posición** en la que se muestra cada caja.
- ¿Cuáles son los **factores** que tienen en cuenta los **navegadores** para **crear cada caja**?:
 - ✓ **Altura** (*height*)
 - ✓ **Anchura** (*width*)
 - ✓ **Tipo de elemento:** *en bloque, en línea*
 - ✓ **Posicionamiento:** *normal, relativo, absoluto, fijo o flotante*
 - ✓ **Relaciones entre elementos:** *descendientes, ...*
 - ✓ **Otro tipo de información:** tamaño imágenes, tamaño de la ventana del navegador, ...
- **Deduce:** ¿cómo se crearán las *cajas* de los siguientes elementos: un párrafo, seguido de un enlace, seguido de otro párrafo que contiene un enlace?

Posicionamiento

- **Tipo de elemento:**

- a) Elementos en **línea**:

- ✓ **Permanecen** en línea con los demás elementos.
- ✓ Empiezan en una **nueva línea** sólo **si es necesario** y ocupan el espacio apropiado para mostrar sus contenidos.
- ✓ Puede aparecer tanto **dentro** de **elementos en bloque** como **en línea**
 - Ej: `<a>`, ``, ``, ``, `<i>`, ``, ``, ``, `<sub>`, `<sup>`, `<cite>`, `<input>`, `<select>`, `<textarea>`, `<label>`, `<button>`

- b) Elementos en **bloque**:

- ✓ Forman un bloque separado:
 - Empiezan en **una nueva línea siempre** y ocupan todo el espacio disponible hasta el final, independientemente de su contenido.
 - Los siguientes elementos aparecerán en líneas inferiores.
- Pueden aparecer **dentro** de otros **elementos en bloque**.
- Ej: `<p>`, `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`, `<div>`, ``, ``, ``, `<dd>`, `<dt>`, `<pre>`, `<hr>`, `<blockquote>`, `<address>`, `<fieldset>`, `<table>`, `<tbody>`, `<tfoot>`, `<thead>`, `<tr>`, `<th>`, `<form>`, `<button>`

Modelos de Posicionamiento y Propiedades

❖ Modelos de *posicionamiento*:

- a) **Normal** o **estático**: posicionamiento predeterminado que utilizan los navegadores.
- b) **Relativo**: desplazamiento de un elemento respecto de su posición original (*normal*).
- c) **Absoluto**: posicionamiento preciso y absoluto de un elemento respecto del elemento contenedor. Puede haber **solapamiento** con el resto de elementos.
- d) **Fijo**: convierte la posición de un elemento en inamovible.
- e) **Flotante**: más complejo. Los elementos se desplazan a la zona más hacia la izquierda/derecha de su posición original.

❖ Propiedades relacionadas:

- ✓ ***position***: indica el **posicionamiento** de un elemento.
- ✓ ***top, right, bottom, left***: para controlar el desplazamiento de las cajas posicionadas (menos modelo flotante).
- ✓ ***float***: permite posicionar de forma flotante una caja.
- ✓ ***clear***: para modificar el comportamiento por defecto del posicionamiento flotante.

Propiedades de Posicionamiento

❖ **position:**

- **Descripción:** propiedad para indicar el **modelo de posicionamiento** aplicado a un elemento y poder **modificar** la **disposición original** de las cajas.
 - ✓ Indica cómo se posiciona una caja, pero **NO** la **desplaza**
 - ✓ Para el **desplazamiento** hay que utilizar las **propiedades *top, right, bottom* y *left***.
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *static*
- **Valores:**
 - ✓ **static:** para indicar que el posicionamiento es el **normal** o **estático**. Es el valor **predeterminado** que utilizan los navegadores para situar la caja de un elemento.
 - Sólo se tiene en cuenta el **tipo de elemento, width, height** y su **contenido**.
 - **NO** es necesario indicarlo.
 - Ej: **img.estatico { position: static; }**
/ asigna a una imagen de la clase estático un posicionamiento normal; equivale a no ponerlo */*

Propiedades de Posicionamiento

❖ Valores de *position*:

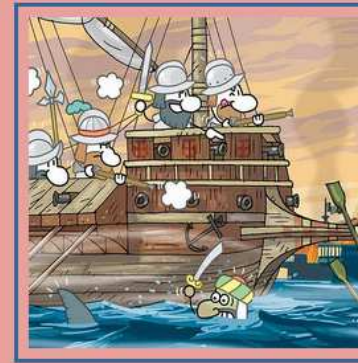
- ✓ **relative**: indica un posicionamiento *relativo* -el desplazamiento de un elemento respecto de su posición original-
 - El desplazamiento se indica con las propiedades: **top**, **right**, **bottom** y **left**.
 - **top**: desplazamiento entre el borde superior de la caja en su posición final y el borde superior de la misma caja en su posición original => **movimiento descendente**.
 - **right**: desplazamiento entre borde derecho de la caja en su posición final y borde superior en posición original => **movimiento** hacia la **izquierda**.
 - **bottom**: el movimiento es **ascendente**.
 - **left**: movimiento hacia la **derecha**.
 - Si se utilizan **valores negativos**, el efecto es **inverso**.
 - Puede producirse un **solapamiento** entre elementos, ya que este desplazamiento **no afecta al resto de cajas adyacentes**, que se muestran en la **misma posición** que si la caja desplazada no se hubiera movido de su posición original.
 - Ej: **#relativo { position: relative; top: 15px; right: 15px; }**
/* asigna una nueva posición al elemento con el id relativo, situándolo con respecto a su posición original 15 px hacia **abajo** y 15 px hacia la **izquierda** */

Propiedades de Posicionamiento

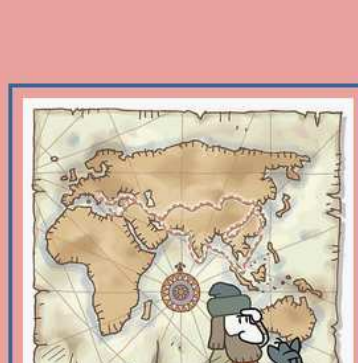
❖ *position*:

- **Observa:** ¿qué valores se están asignando al **posicionamiento**? ¿y para definir cada desplazamiento?

Posicionamiento normal o static:



Posicionamiento relativo con prop top, right, bottom y left:



Propiedades de Posicionamiento

❖ Valores de *position*:

✓ **absolute**:

- Permite indicar de una forma exacta la posición que ocupará la caja de un elemento.
- Interpretación más **compleja**:
 - 1) Desplazamiento de un elemento según origen de **coordenadas** del elemento **contenedor**.
 - 2) **Encontrar** elemento **contenedor** más cercano cuya posición sea **diferente de static**.
 - 3) Si no lo hay, es con respecto a la ventana del navegador.
 - 4) Nueva posición según valor de **top, right, left, bottom**.
- Puede haber **solapamiento**.
- El elemento con este posicionamiento se **sale del flujo normal** del documento => el resto de elementos ocupan su lugar.
 - Ej: **#absoluta { position: absolute; right: 3em; }**



Propiedades de Posicionamiento

❖ Valores de *position*:

✓ **fixed**:

- Caso particular de posicionamiento *absoluto*.
- La forma de obtener el origen de coordenadas para interpretar su desplazamiento es idéntica al posicionamiento absoluto.
- La diferencia están en que la caja se mantiene **fija** aunque utilicemos el **scroll**.
- El **elemento contenedor** es la **ventana del navegador**.
 - Ej: `#fijo { position: fixed; bottom: 3em; }`



Imágenes con Derechos de Autor ©

Página construída con HTML5

Posicionamiento Flotante

❖ Modelo de Posicionamiento flotante:

- Muy utilizado en estructuras de páginas complejas.
- **Finalidad:** que el texto *fluya* alrededor de las imágenes.
- **Funcionamiento:**
 1. Los elementos flotantes **se salen del *flujo normal*** del documento:
 - Se desplazan a la zona **más hacia la izquierda/derecha** de su posición original.
 - El **resto** de elementos **ocupan su lugar**.
 2. Se crea el ***flujo de las cajas flotantes***:
 - Todas las cajas flotantes **respetan el espacio ocupado** por otras cajas flotantes.
 - Influyen en la **disposición del resto de cajas**.

❖ Propiedades para definir el posicionamiento flotante:

- ✓ ***float***: propiedad que permite definir el modelo de posicionamiento *flotante* a una caja.
- ✓ ***clear***: propiedad que **controla el comportamiento** de los elementos que **fluyen alrededor** de las cajas flotantes.

Posicionamiento Flotante

❖ *float*:

- Se aplica a: todos los elementos.
 - Valor inicial: *none*
 - Valores:
 - ✓ **left**: la caja flota a la izquierda, el resto fluyen a su alrededor derecho
 - ✓ **right**: la caja flota a la derecha, el resto fluyen a su alrededor izquierdo
 - ✓ **none**: anula posicionamiento flotante
 - Ej: `.floatL{ float: left; margin: 1em; }`
- /* asigna un posicionamiento flotante izquierdo a la imagen, de forma que ésta se sitúa lo más a la izquierda posible y el resto del contenido fluye hacia la derecha */

Posicionamiento Flotante a la izquierda



Este párrafo va después d las img, cuando no se toca el tipo d posic, es decir, se aplica el posicionamiento normal o predeterminado. Podemos intentar hacer un párrafo más grande, añadiendo más texto, q permita posteriorm visualizar los efectos reales q produce la propiedad *float*. Este párrafo va después d las img, cuando no se toca el tipo d posic, es decir, se aplica el posicionamiento normal o predeterminado. Podemos intentar hacer un párrafo más grande, añadiendo más texto, *float*. Este párrafo va después d las img, cuando no se toca el tipo d posic, es decir, se aplica el posicionamiento normal o predeterminado. Podemos intentar hacer un párrafo más grande, añadiendo más texto, q permita posteriorm visualizar los efectos reales q produce la propiedad *float*. Este párrafo va después d las img, cuando no se toca el tipo d posic, es decir, se aplica el posicionamiento normal o predeterminado.

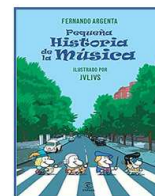
Podemos intentar hacer un párrafo más grande, añadiendo más texto, q permita posteriorm visualizar los efectos reales q produce la propiedad *float*. Este párrafo va después d las img, cuando no se toca el tipo d posic, es decir, se aplica el posicionamiento normal o predeterminado. Podemos intentar hacer un párrafo más grande, añadiendo más texto, q permita posteriorm visualizar los efectos reales q produce la propiedad *float*. Este párrafo va después d las img, cuando no se toca el tipo d posic, es decir, se aplica el posicionamiento normal o predeterminado. Podemos intentar hacer un párrafo más grande, añadiendo más texto, q permita posteriorm visualizar los efectos reales q produce la propiedad *float*. Este párrafo va después d las img, cuando no se toca el tipo d posic, es decir, se aplica el posicionamiento normal o predeterminado. Podemos intentar hacer un párrafo más grande, añadiendo más texto, q permita posteriorm visualizar los efectos reales q produce la propiedad *float*.

Posicionamiento Flotante

❖ *clear*:

- **Descripción:** modifica el comportamiento que producen los elementos flotantes indicando el lado del elemento que **NO** debe ser **adyacente** a ninguna caja **flotante**, desplazando de forma descendente hasta que pueda colocarse en una línea en la que no haya ninguna caja flotante.
- **Se aplica a:** todos los elementos de bloque.
- **Valor inicial:** *none*
- **Valores:**
 - ✓ **left:** coloca el elemento de forma que a su izquierda no haya ningún otro elemento flotante
 - ✓ **right:** coloca el elemento de forma que a su derecha no haya ningún otro elemento flotante
 - ✓ **both:** coloca el elemento de forma que ni a su izquierda ni a su derecha haya ningún elemento flotante
- **Ej:** `.img{ float: left; } .p { clear: left }`

Posicionamiento Flotante a la izquierda de la img y clear a la izq del párrafo

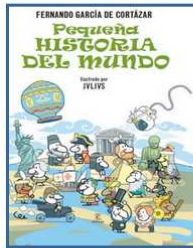


Este párrafo va después d las img, cuando no se toca el tipo d posic, es decir, se apl posteriorm visualizar los efectos reales q produce la propiedad *float*. Este párrafo v hacer un párrafo más grande, añadiendo más texto, *float*. Este párrafo va después d párrafo más grande, añadiendo más texto, q permita posteriorm visualizar los efectos reales q produce la propiedad *float*. Este párrafo va después d las img, cuando no se toca el tipo d posic, es decir, se aplica el posicionamiento normal o predeterminado. Podemos intentar hacer un párrafo más grande, añadiendo más texto, q permita posteriorm visualizar los efectos reales q produce la propiedad *float*. Este párrafo va después d las img, cuando no se toca el tipo d posic, es decir, se aplica el posicionamiento normal o predeterminado. Podemos intentar hacer un párrafo más grande, añadiendo más texto, q permita posteriorm visualizar los efectos reales q produce la propiedad *float*. Este párrafo va después d las img, cuando no se toca el tipo d posic, es decir, se aplica el posicionamiento normal o predeterminado. Podemos intentar hacer un párrafo más grande, añadiendo más texto, q permita posteriorm visualizar los efectos reales q produce la propiedad *float*.

Posicionamiento Flotante

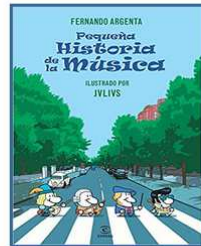
- **Deduce:** observa el siguiente pantallazo e indica qué propiedades de posicionamiento flotante se están aplicando a los distintos grupos de imágenes. ¿Cuándo es interesante aplicar la propiedad *clear*?

Dos imágenes flotantes y dps un párrafo



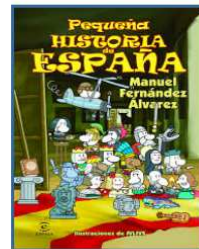
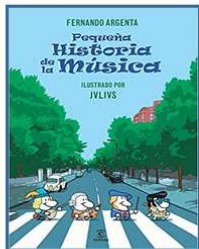
Este párrafo va después d las img flotantes a la izquierda. Al no indicar nada al párrafo, éste debería de "fluir" al lado d las imágenes. Vamos a comprobar si funciona tal y como esperamos. Si necesitamos cambiar la ubicación del párrafo, es cuando hay q añadir la propiedad *clear*.

Todas las imágenes flotantes



Este párrafo va después d las img, cuando no se toca el tipo d posic, es decir, se aplica el posicionamiento normal o predeterminado. Podemos intentar hacer un párrafo más grande, añadiendo más texto, q permita posteriorm visualizar los efectos reales q produce la propiedad *float*.

Todas las imágenes flotantes



Este párrafo va después d las img, cuando no se toca el tipo d posic, es decir, se aplica el posicionamiento normal o predeterminado. Podemos intentar hacer un párrafo más grande, añadiendo más texto, q permita posteriorm visualizar los efectos reales q produce la propiedad *float*.

Visualización

❖ Visualización:

- Además del posicionamiento, desde CSS se puede controlar la visualización de los diferentes elementos.
- Con las propiedades adecuadas, se le indica a los navegadores cómo debe mostrarse cada elemento:
 - ✓ **Ocultarlos:** eliminarlos o hacerlos invisibles
 - ✓ **Mostrarlos**
 - ✓ Cambiar su **comportamiento**
 - ✓ Indicar **posición** en elementos superpuestos

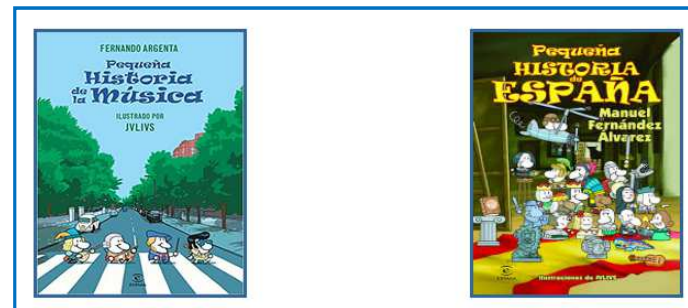
❖ Propiedades:

- ✓ **visibility:** afecta a la visibilidad de un elemento.
- ✓ **display:** cambia el comportamiento de un elemento, puede llegar a ocultarlo totalmente de la página.
- ✓ **overflow:** indica cómo se va a tratar la parte sobrante de un elemento.
- ✓ **z-index:** ordena en profundidad las cajas solapadas.

Visualización

❖ *visibility*:

- **Descripción:** propiedad para hacer **invisible** un elemento haciendo que no se vea en la página. El elemento invisible sigue **ocupando** su lugar. Se utiliza poco.
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *visible*
- **Valores:**
 - ✓ **visible:** muestra el elemento. Es la opción predeterminada.
 - ✓ **hidden:** convierte una caja en **invisible**, ocultando su contenido. Queda un **hueco** vacío en su lugar.
 - ✓ **collapse:** sólo se debe utilizar en elementos de tablas (filas, grupos de filas, columnas, grupos de columnas). Oculta un elemento y el resto de elementos ocupan su lugar –como ocurre con la propiedad *display-*. Aplicado a otros elementos equivale a *hidden*.
 - ✓ **inherit**
- **Ej:** `#invisible { visibility: hidden; }`



Visualización

❖ **display:**

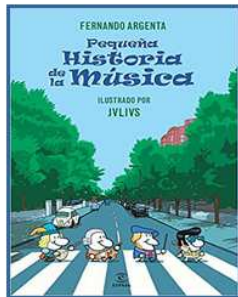
- **Descripción:** propiedad muy importante que permite controlar la forma de visualización de un elemento, cambiando su **comportamiento** -de elemento de línea a elemento de bloque y viceversa-. También permite **ocultarlo** completamente. Se utiliza muchísimo.
 - **Se aplica a:** todos los elementos.
 - **Valor inicial:** *inline*
 - **Valores:**
 - ✓ **none:** para ocultar el elemento, como si no existiese.
 - ✓ **inline:** muestra el elemento como elemento de línea, independientemente del tipo de elemento que sea.
 - ✓ **block:** muestra el elemento como elemento de bloque.
 - ✓ **inline-block:** para que los elementos en bloque se comporten como elementos en línea manteniendo sus propiedades de bloque (*width, height, márgenes verticales*).
- **Deduce:** ¿qué diferencias hay entre ocultar un elemento con **visibility:hidden** o con **display:none**?

Visualización

❖ *display*:

- **Comprueba:** ¿qué efecto se produce cuando cambiamos el comportamiento predeterminado de los elementos imágenes y párrafos?
- **Ej:** crea una **barra de navegación** que se utilice como menú y que contenga las opciones habituales de las páginas web. Utiliza las propiedades correspondientes de posicionamiento y visualización para que dicho menú se muestre horizontalmente.

▪ Ej:



Primer párrafo convertido a elemento de línea Segundo párrafo convertido a elemento de línea

Menú de Navegación Horizontal

[Inicio](#) [Historia](#) [Proyectos](#) [Fotografías](#) [Sobre Nosotros](#) [Contacto](#)

Visualización

❖ **overflow:**

- **Descripción:** propiedad que controla cómo se visualiza el contenido de un elemento que se desborda (sobresale de su espacio reservado).
- **Se aplica a:** elementos en bloque y a las celdas de las tablas.
- **Valor inicial:** *visible*
- **Valores:**
 - ✓ **visible:** para mostrar el contenido sobrante, sobresaliendo de su zona reservada. Es la opción predeterminada.
 - ✓ **hidden:** el contenido sobrante se oculta, y se muestra el contenido que entra dentro de su caja.
 - ✓ **scroll:** muestra el contenido que entra correctamente dentro de su espacio reservado, pero añade barras de scroll para acceder al resto del contenido.
 - ✓ **auto:** según cómo está configurado el navegador, normalmente, como la opción de scroll.
 - ✓ **inherit**
- **Ej: .sobresale { overflow: visible;
width:300px;
height:150px; }**

Este párrafo tiene mucho contenido, tanto, q en el espacio reservado para representar este elemento en el navegador es imposible q entre, d forma q se va a producir un "desbordamiento". La forma d solucionar este problema el diseñador, en vez d hacer más grande el espacio para el párrafo, es mostrar una barra scroll para acceder a todo el contenido. Vamos a comprobar si d verdad funciona así.

Visualización

❖ **z-index:**

- **Descripción:** propiedad que establece la posición **tridimensional** que ocupará un elemento que se **solapa** con otros.
- La posición se indica a través del **eje z**, usando una numeración para cada caja. A **mayor valor, más cerca** se coloca la caja del usuario.
- **Se aplica a:** elementos posicionados (*position*).
- **Valor inicial:** *auto*
- **Valores:**
 - ✓ **numero**
 - ✓ **auto**
 - ✓ **inherit**
- **Ej:** `#img1 { position: absolute;
z-index: 5;
top: 10em;
left: 20em;}`



Módulo de Texto

❖ Propiedades básicas que afectan a la fuente:

- ✓ **color** (color)
- ✓ **font-family** (tipo letra)
- ✓ **font-size** (tamaño)
- ✓ **font-weight** (grosor)
- ✓ **font-style** (estilo)
- ✓ **font** (**shorthand**)

❖ Propiedades básicas que afectan al texto:

- ✓ **text-align** (alineación)
- ✓ **line-height** (Interlineado)
- ✓ **text-decoration** (Subrayado)
- ✓ **text-transform** (Mayúsculas/Minúsculas)
- ✓ **vertical-align** (alineación vertical)
- ✓ **text-indent** (tabulación)
- ✓ **letter-spacing** (separación letras)
- ✓ **word-spacing** (separación palabras)
- ✓ **white-spacing** (espacios en blanco)

Módulo de Texto

❖ Otras propiedades de texto:

- ✓ ***direction*** (dirección del texto)
- ✓ ***text-overflow*** (texto sobrante)
- ✓ ***text-shadow*** (texto con sombra)
- ✓ ***opacity*** (transparencia)

❖ Elegir otros tipos de fuente:

- ✓ ***@font-face***: descarga una fuente en el servidor web
- ✓ ***Google-fonts***

Propiedades básicas que afectan a la fuente

❖ **color** (color)

- **Descripción:** propiedad que asigna el **color del texto** a través de *palabras reservadas, RGB decimal, RGB hexadecimal, HSL, ...* El color elegido debe **contrastar** con el **color de fondo** del elemento para aumentar la **legibilidad**.
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *auto* (según el navegador, suele ser el negro)
- **Valores:**
 - ✓ **color**
 - ✓ **inherit**
- **Ej:** `p { color: orange; }`

❖ **font-family** (tipo de letra)

- Se recomienda utilizar un **máximo** de 2 o 3 fuentes en una página web.
- El tipo de letra a utilizar tiene que estar **instalada en el navegador** del equipo **local** donde se está visualizando el contenido web => indicar **varias fuentes**.
- **Formas de indicar el tipo de letra:**
 - ✓ **Nombre de familia** tipográfica: indicando un nombre específico de una tipo de letra.
 - ✓ **Nombre genérico** de una familia tipográfica: se refieren al **estilo** del tipo de letra

Propiedades básicas que afectan a la fuente

▪ Familias de fuentes:

a) **serif**: ideales para texto largo, facilitan la lectura

- **“Times New Roman”**, Times, **serif**
- **Georgia**, “Times New Roman”, Times, serif
- ✓ Baskerville, “Palatino Linotype”, Times, serif
- ✓ “Hoefler Text”, Garamond, Times, serif

b) **sans-serif**: apariencia clara y limpia, ideales para titulares

- **Arial**, Helvetica, sans-serif
- **Verdana**, Arial, Helvetica, sans-serif
- ✓ Geneva, Arial, Helvetica, sans-serif
- ✓ **Tahoma**, “Lucida Grande”, Arial, sans-serif
- ✓ “Trebuchet MS”, Arial, Helvetica, sans-serif
- ✓ “Century Gothic”, “Gill Sans”, Arial, sans-serif

c) **fantasy**: Impact (común para pc’s y mac’s)

d) **cursive**

e) **monospace**: útiles para fragmentos de código; cualquier letra con mismo ancho

- **“Courier New”**, Courier, monospace
- ✓ “Lucida Console”, Monaco, monospace

f) Fuentes **divertidas**:

- ✓ “Copperplate Light”, “Copperplate Gothic Light”, serif
- ✓ “Marker Felt”, “Comic Sans MS”, fantasy

Propiedades básicas que afectan a la fuente

❖ **font-family** (tipo de letra)

- **Descripción:** propiedad que asigna el **tipo de letra** con el que se va a mostrar el texto contenido en los elementos.
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *auto* (según el navegador, suele ser el negro)
- **Valores:**
 - ✓ Nombre de familia
 - ✓ Nombre de familia genérica
 - ✓ inherit
- **Sintaxis1:** font-family: Nombre_familia_genérica;
- **Ej:** p {font-family: sans-serif; }

/* el navegador asignará a los párrafos la fuente que más se parezca al tipo Arial de todos los que tenga instalados el usuario */

- **Sintaxis2:** font-family: TipoLetra1, "Tipo Letra 2", Nombre_familia_genérica;
- **Ej:** h3 {font-family: Georgia, "Times New Roman", Times, serif; }

/* el navegador asignará a los h3 la primera fuente indicada si la tiene instalada, si no, asigna la siguiente fuente y así sucesivamente */

Propiedades básicas que afectan a la fuente

❖ **font-size** (tamaño)

- **Descripción:** propiedad que asigna el **tamaño del texto**
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *medium*
- **Valores:**
 - ✓ **Tamaño absoluto:** para tamaños predefinidos. Valores entre *xx-small*, *x-small*, *small*, *medium*, *large*, *x-large*, *xx-large*
 - ✓ **Tamaño relativo:** su referencia es el tamaño de letra del **elemento padre**, haciéndola más pequeña o más grande respectivamente. Valores *smaller*, *larger*.
 - ✓ Medida exacta: nº y unidad con medidas absolutas, relativas, porcentuales.
 - ✓ **inherit**
- Ej: **h1 { font-family: Arial, Verdana, Helvetica, sans-serif; font-size: xx-large; }**

Propiedades básicas que afectan a la fuente

❖ **font-weight** (anchura)

- **Descripción:** propiedad que asigna el **grosor del texto**. Afecta a la característica de negrita. Los navegadores muestran los elementos con negrita.
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *normal*
- **Valores:**
 - ✓ **Palabrar reservada:** *normal, bold, bolder, lighter*
 - ✓ Un nº entre 100 y 900 (*normal* se corresponde con 400, *bold* con 700)
 - ✓ **inherit**
- **Ej:** `strong { font-weight: normal;`
`background-color: #FE5; }`
/ sustituye el estilo predeterminado de la etiqueta strong por un grosor normal pero con color de fondo */*
- **Comprueba:** ¿a qué valor numérico se corresponde la opción “*bolder*”? ¿y “*lighter*”?

Propiedades básicas que afectan a la fuente

❖ **font-style** (estilo)

- **Descripción:** propiedad que asigna un **estilo de texto**. Afecta a la característica de cursiva. Los navegadores muestran los elementos `` con cursiva.
 - **Se aplica a:** todos los elementos.
 - **Valor inicial:** *normal*
 - **Valores:**
 - ✓ **Palabrar reservada:** *normal, italic, oblique*
 - ✓ **inherit**
 - **Ej:** `em { font-weight: bold;
 font-style: normal; }`
/ sustituye el estilo predeterminado de la etiqueta em por una letra en negrita y sin cursiva */*
- **Comprueba:** ¿hay alguna diferencia entre el estilo *italic* y el *oblique*?

Propiedades básicas que afectan a la fuente

❖ **font** (*shorthand*)

- **Descripción:** fija en una sola propiedad **todas las características** sobre la **fuentes de texto**. El **orden** habitual es **estilo, grosor, tamaño, interlineado y fuente**
- Es **indiferente** el **orden** de las propiedades.
- Es **obligatorio** indicar el **tamaño** y la **fuentes**.
- Los distintos tipos de fuentes se separan utilizando comas.
- Para diferenciar el tamaño del interlineado se utiliza la barra inclinada /.
- Los valores que no se indican se sustituyen por su valor predeterminado.
- **Sintaxis:**
`{ font: font-style font-variant font-weight font-size/line-height font-family; }`
- Ej: `p { font: 1.5em/150% Georgia, Times, serif; }`

Probando propiedad font con tamaño 1.5em,
la familia serif.

Propiedades básicas que afectan al Texto

- **Propiedades básicas que afectan al *texto***

- ❖ ***text-align*** (alineación)

- **Descripción:** propiedad que determina la **alineación horizontal** del contenido de un elemento respecto al tamaño de su caja.
- **Se aplica a:** elementos en bloque y celdas de tablas.
- **Valor inicial:** *left*
- **Valores:**
 - ✓ **left:** alineación a la izquierda
 - ✓ **right:** alineación a la derecha
 - ✓ **center:** alineación centrada
 - ✓ **justify:** alineación a izquierda y derecha
 - ✓ **inherit**
- **Ej: `.centrada { text-align: center; }`**
 - /* todos los elementos en bloque que tengan asignada la clase “centrada” se van a alinear en el centro de su caja */*

Propiedades básicas que afectan al Texto

❖ **line-height** (interlineado)

- **Descripción:** propiedad que determina la **altura** que ocupa cada **línea** de texto. Mejora la legibilidad del texto.
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *normal*
- **Valores:**
 - ✓ **normal:** según tamaño de letra
 - ✓ **número:** múltiplo del tamaño de letra, sin indicar la ud. de medida
 - ✓ **Ud. de medida:** **absoluta**, relativa, porcentual
 - ✓ **inherit**
- **Ej:** `p {font-size: 12px; line-height:1.2;}`

/ aplica a los párrafos un tamaño de letra de 12px y un interlineado de 1,2 veces mayor que dicho tamaño */*

Propiedades básicas que afectan al Texto

❖ ***text-decoration*** (subrayado)

- **Descripción:** propiedad que determina un estilo de **decoración** del texto.
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *none*
- **Valores:**
 - ✓ **none:** no aplica ningún efecto
 - ✓ **underline:** subrayado. Es el valor predeterminado de los enlaces
 - ✓ **overline:** línea por encima del texto
 - ✓ **line-through:** tachado
 - ✓ **blink:** parpadeo. NO recomendable. En desuso. Los navegadores suelen deshabilitarlo.
 - ✓ **inherit**
- **Ej:** `a:visited { text-decoration: none; }`
/ elimina el subrayado a los enlaces ya visitados */*

Propiedades básicas que afectan al Texto

- ❖ ***text-transform*** (convertir a mayúsculas/minúsculas)
 - **Descripción:** propiedad que **transforma** el texto escrito en Html a mayúsculas, minúsculas o la primera letra de cada palabra a mayúsculas.
 - **Se aplica a:** todos los elementos.
 - **Valor inicial:** *none*
 - **Valores:**
 - ✓ **none:** no aplica ningún efecto
 - ✓ **capitalize:** primera letra de cada palabra en mayúsculas
 - ✓ **uppercase:** todo en mayúsculas
 - ✓ **lowercase:** todo en minúsculas
 - ✓ **inherit**
 - **Ej:** `h1.inicio { text-transform: capitalize; }`
/ Aplica la opción de capitalize a los h1 de la clase inicio */*

En Un Lugar De La Mancha...

Propiedades básicas que afectan al Texto

❖ **vertical-align** (alineación vertical)

- **Descripción:** propiedad que define la **alineación vertical** del contenido de un elemento –texto e imagen- o dentro de una celda.
- **Se aplica a:** elementos en línea y celdas de tabla.
- **Valor inicial:** *baseline*
- **Valores:**
 - ✓ **baseline:** coloca todos los elementos en la línea base inferior
 - ✓ **sub:** subíndice
 - ✓ **super:** superíndice
 - ✓ **top:** arriba respecto al elemento más alto de la línea
 - ✓ **text-top:** en la línea superior del texto
 - ✓ **middle:** medio respecto a la altura del texto o contenedor. Muy utilizada.
 - ✓ **bottom:** abajo respecto al elemento más alto de la línea
 - ✓ **text-bottom:** en la línea inferior del texto
 - ✓ **unidad de medida:** relativa, porcentaje
 - ✓ **inherit**
- **Ej:** `img {vertical-align:top; }`
/ Aplica una alineación vertical superior a las imágenes */*

Propiedades básicas que afectan al Texto

❖ **text-indent** (tabulación)

- **Descripción:** propiedad que define la **tabulación** o **indentación** de la primera línea de un párrafo, dejando una distancia extra con respecto al resto de líneas del párrafo. Cuando hay mucho texto, ayuda a la legibilidad del mismo.
- **Se aplica a:** elementos en bloque y celdas de tablas.
- **Valor inicial:** 0
- **Valores:**
 - ✓ **0:** no aplica ninguna tabulación
 - ✓ **Ud. de medida**
 - ✓ **Porcentaje**
 - ✓ **inherit**
- **Ej:** `p { text-indent: 30px; }`

Párrafo muy largo en el que se habilita la opción d indentación, gracias a la propiedad text-indent. De esta forma, se facilita la lectura del contenido del mismo. Interesante aplicarla a varios párrafos, no sólo a uno. Es una manera d diferenciar cada bloque d texto.

Párrafo muy largo en el que se habilita la opción d indentación, gracias a la propiedad text-indent. De esta forma, se facilita la lectura del contenido del mismo. Interesante aplicarla a varios párrafos, no sólo a uno. Es una manera d diferenciar cada bloque d texto.

Propiedades básicas que afectan al Texto

- ❖ **letter-spacing** y **word-spacing** (separación entre letras y entre palabras)
 - **Descripción:** propiedades que establecen una **separación extra** entre las **letras** de cada palabra y entre las **palabras** respectivamente.
 - **Se aplica a:** todos los elementos.
 - **Valor inicial:** *normal*
 - **Valores:**
 - ✓ **normal:** no aplica ningún efecto
 - ✓ **Ud. de medida**
 - ✓ **inherit**
 - **Ej:** `p {letter-spacing: 3px; word-spacing: 10px; }`

Comprobando cómo funciona la separación entre letras y entre palabras. La intención de esta propiedad es también mejorar la legibilidad del texto.

Propiedades básicas que afectan al Texto

❖ **white-spacing** (espacios en blanco)

- **Descripción:** propiedad que determina cómo los navegadores deben **tratar los espacios en blanco adicionales**.
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *normal*
- **Valores:**
 - ✓ **normal:** elimina espacios y líneas en blanco sobrantes. Predeterminado.
 - ✓ **nowrap:** como el normal, pero sin ajustar las líneas demasiado largas – saliéndose de su espacio asignado-
 - ✓ **pre:** el texto se conserva tal y como esté en el documento HTML. Una línea muy larga se saldrá de su espacio asignado.
 - ✓ **pre-wrap:** el texto se conserva tal y como esté en el documento HTML, ajustando cada línea al espacio asignado para ese contenido.
 - ✓ **pre-line:** elimina los espacios en blanco y respeta las nuevas líneas, ajustando cada línea al espacio asignado para ese contenido.
 - ✓ **inherit**
- Ej: `#poesía { white-spacing: pre-wrap; }`

Otras Propiedades de Texto

❖ **direction** (dirección del texto)

- **Descripción:** propiedad que indica la **dirección** en la que se escribe el texto.
- **Se aplica a:** todos los elementos.
- **Valor inicial:** *ltr*
- **Valores:**
 - ✓ **ltr:** de izquierda a derecha. Predeterminado.
 - ✓ **rtl:** de derecha a izquierda (utilizada en otros idiomas).
 - ✓ **inherit**

- Ej: **p.arabe { direction: rtl; }**

.Escribiendo en árabe y no digo más

❖ **text-overflow** (texto sobrante)

- **Descripción:** propiedad que indica qué hacer con el texto sobrante en su contenedor. Es de **CSS3**.
- **Valores:**
 - ✓ **clip:** el texto sale recortado. Sólo se ve el texto que cabe en la caja, el resto no se muestra.
 - ✓ **ellipsis:** el texto sale también recortado, añadiendo puntos suspensivos al final.

Otras Propiedades de Texto

❖ **text-shadow** (texto con sombra)

- **Descripción:** propiedad que permite **colocar una sombra al texto** para darle efecto de volumen (efecto 3D). Es de **CSS3**.
- **Sintaxis:** **text-shadow: color distanciaX distanciaY desenfoque**
 - ✓ **color:** color de la sombra.
 - ✓ **distanciaX:** desplazamiento horizontal de la sombra. Valor positivo va hacia la derecha; negativo hacia la izquierda.
 - ✓ **distanciaY:** desplazamiento vertical de la sombra. Valor positivo va hacia abajo; negativo hacia arriba.
 - ✓ **desenfoque:** **opcional**, indica cuánto se va a desenfocar la sombra. A mayor valor, mayor será el desenfoque del fondo
- Ej: **h2{ text-shadow: 3px 3px 2px #696; color: #666; }**

Título con sombra efecto pixelart

CSS box-shadow: efectos de
sombras

Ejemplo de varias sombras en el mismo elemento
Ejemplo de varias sombras en el mismo elemento

Otras Propiedades de Texto

❖ **opacity** (transparencia)

- **Descripción:** propiedad que permite asignar **transparencias** en cualquiera de los elementos. Es de **CSS3**.
- **Valores:**
 - ✓ **número:** entre 0 y 1. Con 0 se consigue la transparencia total (**invisible**) y con 1 la **opacidad total**.
- **Ej:** `.texto { color:maroon; opacity:0.3; }`

`/* aplica un color oscuro y una transparencia de 0,3 a todos los elementos de la clase texto */`

texto casi en negrita

texto casi en negrita con transparencias

Módulo de Texto

❖ Elegir otros tipos de fuente:

- ✓ *@font-face*: descarga una fuente en el servidor web
- ✓ **Google-fonts**:
 - Gran variedad de fuentes.
 - No es necesario almacenarla en el servidor web.
 - Genera el código *font-face* adecuado para copiarlo y pegarlo en nuestro código
 - Elegir desde la url: ***www.google.com/webfonts***
 - **Proceso**:
 - 1) **Elegir**: utilizar el buscador “*search*” especificando nombre de fuente o diferentes opciones de búsqueda.
 - 2) **Revisar**
 - 3) **Usar**: **copiar código** para Html, Css o Java.
 - Ej: `<link href='http://fonst.googleapis.com/css?family='Metal+Mania' rel='stylesheet' type='text/css'>`
 - Para **usarla**: **font-family: 'Metal Mania';**

Personalizando la apariencia de los elementos

❖ Apariencia de enlaces:

- ✓ *pseudoclases, iconos, botones e imágenes*

❖ Listas:

- ✓ Viñetas personalizadas: *list-style-type*
- ✓ Posición de las viñetas: *list-style-position*
- ✓ Imágenes de viñetas: *list-style-image*
- ✓ Propiedad shorthand: *list-style*

❖ Menú Vertical y Menú Horizontal

❖ Tablas

❖ Formularios

❖ Resetear estilos predeterminados

❖ Diseño de páginas

- ✓ **Centrar contenido** horizontalmente

❖ Propiedades avanzadas de CSS3:

- ✓ *Opacidad, gradientes, transformaciones –transform-, transiciones –transition-, animaciones –@keyframes-.*

Enlaces personalizados

❖ Mejorar la apariencia de los **enlaces** mediante:

- ✓ **Propiedades de texto:** tamaño, color, subrayado, grosor, etc.
 - Para cambiar el grosor, estilo y color del subrayado, independientemente del estilo del texto: ***border-bottom***
- ✓ Aplicar diferentes **estilos según estado:** ***pseudoclases***
 - **Orden** correcto: *link*, *visited*, *hover* y *active*.
 - Los estado *link* y *visited* son **exclusivos** de los **enlaces**.
- ✓ Añadir un **icono** según el **tipo de enlace**:
 - Definir la imagen con ***background*** (sin repetir y situada a la izquierda y en el centro) y separarla con relleno a la izquierda.
- ✓ Diseñar enlace con apariencia de **botón**:
 - Añadir color de fondo, borde, relleno, etc.
- ✓ Utilizar una **imagen como enlace**
 - Eliminar el borde de la imagen-enlace que añaden los navegadores

Enlaces personalizados

- ❖ **Mejorar la apariencia** de los **enlaces** mediante:
 - Ej: crea una página web que contenga varias **enlaces** con diferente apariencia, cambiando el *color*, *subrayado*, añadiendo un *icono* junto al enlace, y crear un *botón* y una *imagen* como enlaces.

Enlaces con apariencia diferente:

[Ir a entrevista](#)



[Descarga](#)

[Pulsa el botón](#)



Listas personalizadas

❖ Tipo de viñeta: *list-style-type*

- **Descripción:** propiedad que permite especificar el **tipo de viñeta** de una lista.
- **Se aplica a:** elementos de una lista.
- **Valor inicial:** *disc*
- **Valores:**
 - ✓ **none:** elimina la viñeta/nº/letra. Imprescindible para menú de navegación.
 - **Gráficos:**
 - ✓ **disc:** círculo relleno. Predeterminado
 - ✓ **circle:** círculo vacío
 - ✓ **square:** cuadrado
 - **Númericos:**
 - ✓ **decimal** (sin el cero)
 - ✓ **decimal-leading-zero**
 - ✓ **lower-roman, upper-roman**
 - ✓ **armenian, georgian, hebrew, hiragana, katakana**
 - **Alfanuméricos:**
 - ✓ **lower-greek, lower-latin, upper-latin, lower-alpha, upper-alpha**

Listas personalizadas

❖ Posición de la viñeta: *list-style-position*

- **Descripción:** propiedad que permite establecer la **posición** de la **viñeta**.
- **Se aplica a:** elementos de una lista.
- **Valor inicial:** *outside*
- **Valores:**
 - ✓ **outside:** viñeta hacia fuera y resto de líneas del párrafo más hacia dentro. Predeterminado.
 - ✓ **inside:** viñeta alineada junto con el resto de líneas del párrafo.
 - ✓ **inherit**

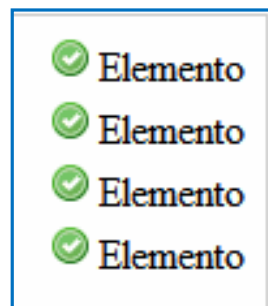
❖ Imagen de viñeta: *list-style-image*

- **Descripción:** propiedad que permite cambiar la **imagen** que se muestra como viñeta. Las imágenes personalizadas se indican mediante la URL de la imagen.
- **Se aplica a:** elementos de una lista.
- **Valor inicial:** *none*
- **Valores:**
 - ✓ **none:** se usa la viñeta predeterminada.
 - ✓ **url:** para indicar la imagen deseada (*.jpg, *.gif, *.png).
 - ✓ **inherit**

Listas personalizadas

❖ Propiedad shorthand: **list-style**

- **Descripción:** propiedad que permite establecer una o varias características personalizables de las viñetas de las listas.
- **Se aplica a:** elementos de una lista.
- **Sintaxis:** `list-style: tipo posición imagen_viñeta_alternativa;`
- **Ej1:** `ul { list-style: none; }`
/ lista de opciones que sirve como **menú de navegación** (sin viñeta) */*
- **Ej2:** `ul { list-style: url("logo.png") square; }`
/ lista de opciones que utiliza como viñeta una imagen almacenada en el fichero "logo.png". En caso de no poder cargar dicha imagen, usa la viñeta "square" */*



Barra de Menú Vertical

❖ Proceso de creación y transformación de un menú vertical:

- 1) **Crear** dentro de la etiqueta correspondiente `<nav>` una **lista** de opciones, y cada una de las opciones que sea un **enlace**.
- 2) **Modificar** el **menú** en su conjunto: `nav ul` o `ul.clase`
 - ✓ Recomendable asignar una **clase** a la **lista** creada dentro de `<nav>`
 - ✓ Definir **anchura**, **eliminar viñetas**, quitar márgenes y relleno, asignar un borde.
- 3) **Modificar** cada **elemento** del menú: `nav ul li`
 - ✓ Definir **color de fondo**, **borde** superior e inferior.
 - ✓ Eliminar borde inferior del menú (para que no sea doble).
- 4) **Modificar** cada **enlace** de cada elemento del menú: `nav ul li a`
 - ✓ Eliminar el **subrayado**, añadir espacio de **relleno** y añadir **color** de texto.
 - ✓ **Mostrar cada enlace como bloque.**
- 5) Modificar color de fondo cuando se señala el enlace (**:hover**)
- 6) Añadir otras características según el **estado** del enlace.

Barra de Menú Vertical

- ❖ **Proceso de creación y transformación de un menú vertical:**
 - Ej: crea una página web que contenga una barra de navegación estándar y otra que se muestre en vertical.

Barra de menú standard

- *Inicio*
- *Resumen*
- *Contenido*
- *Imágenes*
- *Videos*

Barra de menú Vertical

Inicio

Resumen

Contenido

Imágenes

Videos

Barra de Menú Horizontal

❖ Proceso de creación y transformación de un menú horizontal:

- El proceso es muy similar a la creación de un menú vertical, lo único que hay que terminar convirtiendo la **barra vertical en horizontal**:

- a) Convertir elementos *–nav, ul, li, a–* a **inline** o **inline-block**
- b) Utilizar **float**

3) Modificar cada elemento del menú: **nav ul li**

- ✓ Establecer la **anchura** de cada elemento y **bordes**.
- ✓ **Convertir el menú a horizontal**: asignar **posicionamiento flotante** a la izquierda a cada **li** con **float**

4) Modificar cada enlace de cada elemento del menú: **nav ul li a**

- ✓ Eliminar el **subrayado**, **mostrarlo como bloque**, añadir espacio de **relleno**, **bordes** y **color** de texto y de fondo .
- ✓ Añadir **bordes** si es necesario.

Barra de Menú Horizontal

- ❖ **Proceso de creación y transformación de un menú horizontal:**
 - Ej: crea una página web que contenga una barra de navegación estándar y otra que se muestre en horizontal.

Barra de menú standard

- *Inicio*
- *Resumen*
- *Contenido*
- *Imágenes*
- *Videos*

Barra de menú Horizontal

<i>Inicio</i>	<i>Resumen</i>	<i>Contenido</i>	<i>Imágenes</i>	<i>Videos</i>
---------------	----------------	------------------	-----------------	---------------

Resetear Estilos predeterminados

❖ Resetear estilos predeterminados:

- Cada **navegador** tiene una *hoja de estilos predeterminada*, que da formato básico a todos los elementos de las páginas web.
- **Problema:** estilos diferentes y cualquier pequeña variación –margen-, puede descolocar la página completamente.
- **Solución:**
 - **CSS Reset:** hoja de estilos que **elimina** los estilos predeterminados de los navegadores.
 - ✓ Todos los elementos con el mismo aspecto liso y plano.
 - ✓ Necesario crear un CSS que cubra todos los matices.
 - ✓ Mayor control sobre los nuevos estilos a aplicar.
 - **CSS Normalize:** hoja de estilos que **igual**a las diferencias existentes entre los estilos de los navegadores.
- Independientemente del que se utilice, tiene que ser la **primera hoja de estilos que se cargue en el <link>** de la página html, después, se enlaza con el resto de hojas de estilos.

Resetear Estilos predeterminados

❖ Resetear estilos predeterminados:

- Una hoja de reseto muy usada es la de Richard Clark *richclarkdesign.com*
- Ej **reset.css**:

```
html, body, div, span, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a, abbr, cite,
em, img, strong, sub, sup, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form,
label, legend, table, caption, tbody, tfoot, thead, tr, th, td, article, aside,
canvas, details, embed, figure, figcaption, footer, header, hgroup, menu,
nav, output, section, audio, video {
    margin: 0; padding: 0; border: 0; font-size: 100%;
    font: inherit; vertical-align: baseline;}
article, aside, figcaption, figure, footer, header, hgroup, menu, nav, section
    { display: block;}
body { line-height: 1;}
ol, ul { list-style: none;}
table { border-collapse: collapse; border-spacing: 0;}
```

- Para ver una hoja de ***normalize*** visita el siguiente enlace
github.com/necolas/normalize.css

Diseño de Páginas

❖ Diseño de páginas con CSS versus **tablas**:

- ✓ **Facilita el mantenimiento**
 - ✓ **Aumenta la accesibilidad**
 - ✓ **Aumenta la velocidad de carga**
 - ✓ **Aumenta valor semántico de las páginas web**
 - ✓ **Mejora el posicionamiento**
- El **diseño habitual** de una página web consiste en **dividir** su contenido en **bloques**: *cabecera, menú, contenidos y pie de página*.
 - Ej:



Diseño de Páginas

- **Deduce:** ¿cómo se mostrará una página web en una pantalla de gran resolución? ¿qué modelo de diseño elegir?
- **Modelos de diseño o *layout* habituales:**
 - ✓ **Fixed:** anchura fija de los elementos definida en **px**. Se controla la colocación de cada bloque, pero en pantallas de diferente tamaño no se adapta correctamente –uso de scroll/ demasiado espacio blanco a los lados-
 - ✓ **Elastic:** anchura de los elementos definida en **em**. Sí escala correctamente, pero se pueden producir solapamientos entre elementos adyacentes.
 - ✓ **Fluid:** anchura de los elementos definida en **%** con respecto a la etiqueta padre. Se mantiene la proporción pero en pantallas pequeñas cada bloque puede ser demasiado estrecho.
 - ✓ **Con Min/Max Sizing (híbrido):** anchura de los elementos definida entre dos valores máximo y mínimo en **px**.
 - ✓ **Responsive:** la estructura **cambia** dependiendo de las características de la pantalla (tamaño de pantalla). Nivel avanzado de CSS3.

Diseño de Páginas

- **Sol:** las líneas son demasiado largas y si la alineación es a la izquierda se verá demasiado espacio en blanco. Se debe **centrar la página horizontalmente**, con menús y contenidos a la izquierda. La estructura se divide normalmente en dos o tres columnas.
- **Proceso de diseño:**
 - 1) **Centrar páginas horizontalmente respecto al navegador**
 - a) **Definiendo una anchura fija:** modelo *fixed*
 - Se eliminan las líneas de texto demasiado largas.
 - Se definen márgenes de seguridad.
 - Aumenta la comodidad al leer –contenido legible y agradable-.
 - **¿Cómo se implementa?**
 - Crear un **contenedor** de toda la página (ahí van los <div> o en el <body>).
 - Definir los **márgenes laterales** como **auto**, para que el navegador **centre** automáticamente todo el contenido con respecto a su *elemento padre* .
 - Ej: `#wrapper { width: 800px; margin: 0 auto; }`

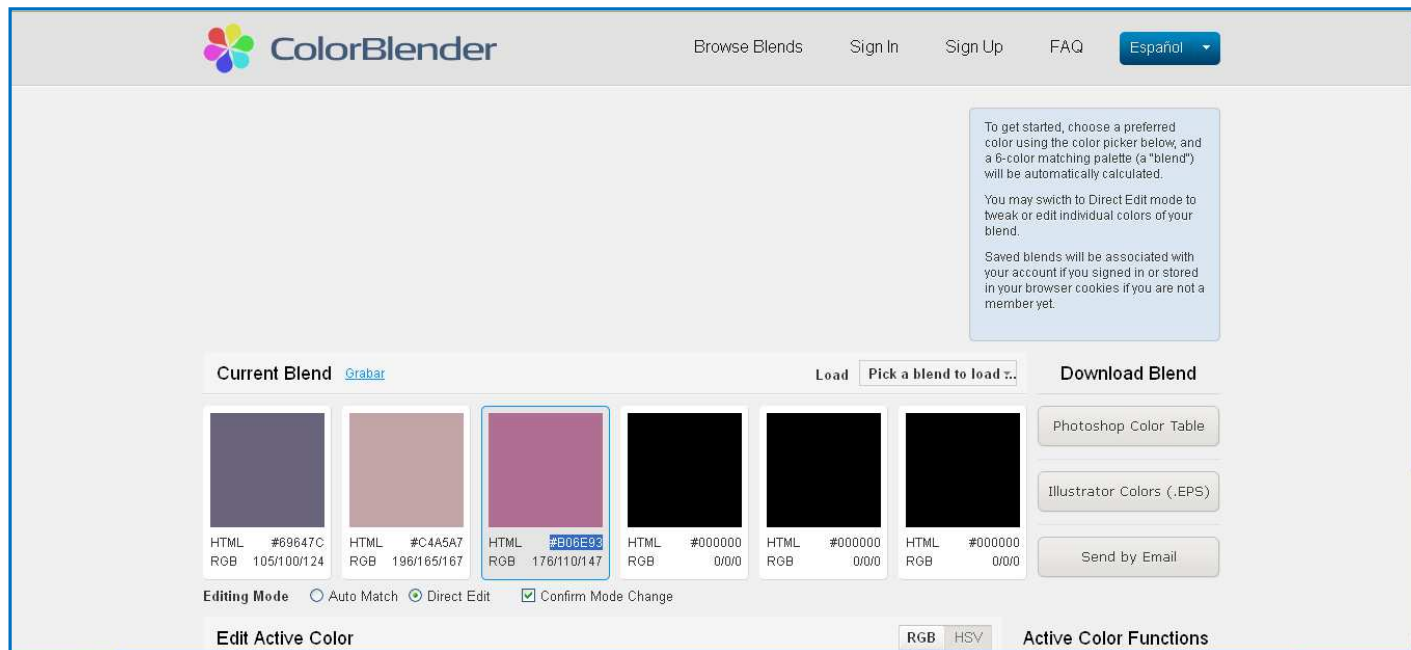
Diseño de Páginas

- Proceso de diseño:

- 1) Centrar páginas horizontalmente respecto al navegador

- b) Definiendo una anchura *dinámica*: modelo *fluid*

- Hay que indicar el ancho utilizando el sistema porcentual
- Ej: `#wrapper { width: 80%; margin: 0 auto; }`
- Ej de página web que está centrada horizontalmente:



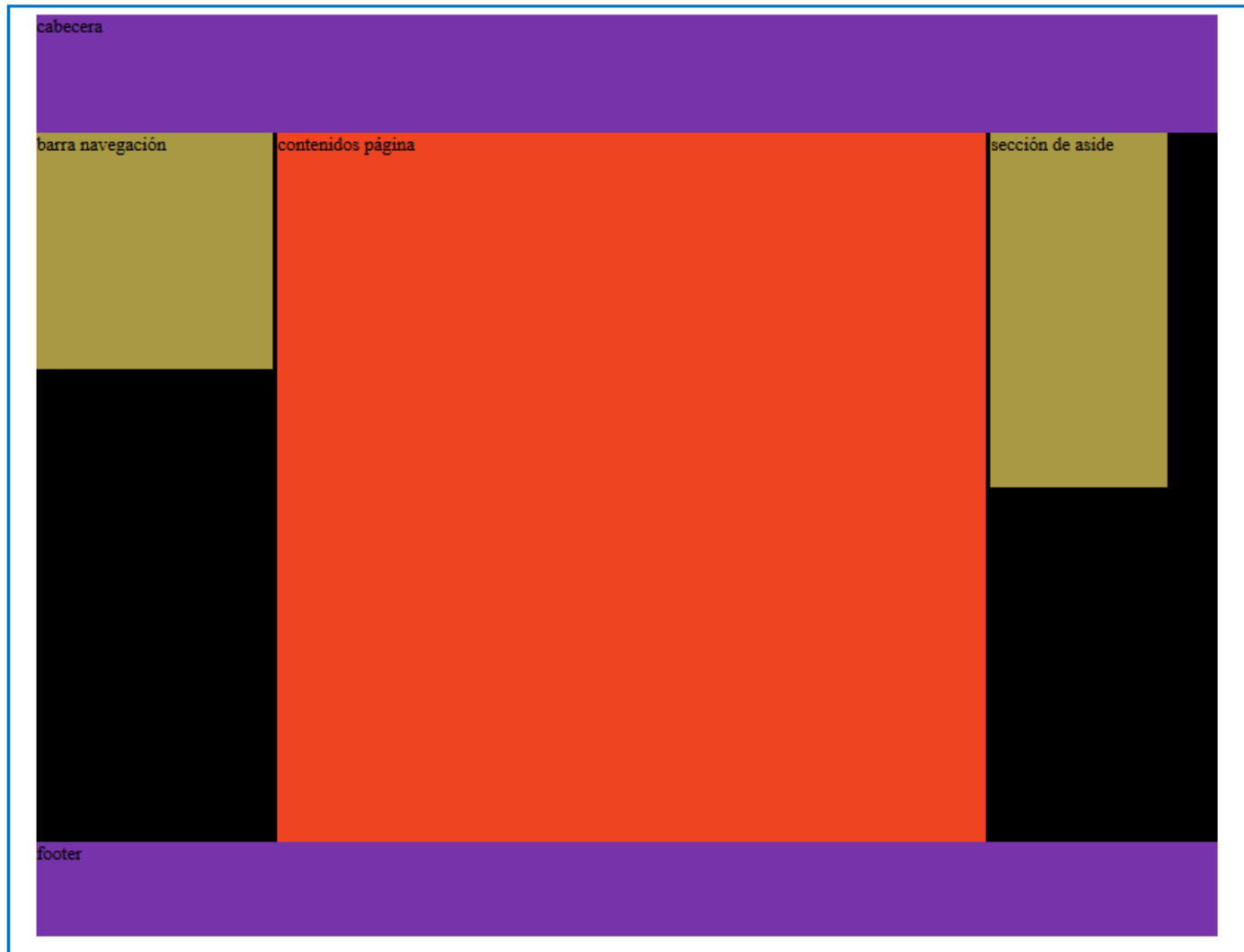
Diseño de Páginas

- Proceso de diseño:
 - 2) Dividir estructura en un nº de columnas
 - a) En 2 columnas: menú y contenido flotando a la izquierda



Diseño de Páginas

- Proceso de diseño:
 - Dividir estructura en un nº de columnas
 - En 3 columnas: menú, contenido y aside flotando a la izquierda



Diseño de Páginas

- Proceso de diseño:
 - 3) Posicionar los elementos contenedores con **display** o con **float/clear**
 - 4) Indicar tamaño **fijo** o **dinámico**
 - 5) Definir **tamaño máximo** y **mínimo** de los elementos:
 - a) Determinar anchuras máximas y mínimas
 - ✓ **max-width**: para definir la **anchura máxima** de un elemento
 - **Valores**: unidades de medida, porcentaje, none
 - ✓ **min-width**: para definir la **anchura mínima** de un elemento
 - **Valores**: unidades de medida, porcentaje, 0
 - b) Determinar alturas máximas y mínimas
 - Menos utilizado
 - ✓ **Max-height**: para definir la **altura máxima** de un elemento
 - **Valores**: unidades de medida, porcentaje, none
 - ✓ **Min-height**: para definir la **altura mínima** de un elemento
 - **Valores**: unidades de medida, porcentaje, 0

Diseño de Páginas

- Ej: partiendo de una misma página web con los contenidos habituales –cabecera, menú de navegación, contenidos, ...- varía su diseño en dos o tres columnas, manteniendo los contenidos centrados horizontalmente para una correcta visualización.

