UD1 USO DE ESTILOS

Módulo de Diseño de Interfaces Web

C.F.G.S. Técnico Superior en Desarrollo de Aplicaciones Web

USO DE ESTILOS

- 1 INTRODUCCIÓN A HOJAS DE ESTILO EN CASCADA (CSS, CASCADING STYLE SHEET)
- 2 CREAR Y VINCULAR HOJAS DE ESTILOS
- 3 SELECTORES
  - 3.1. SELECTORES BASADOS EN ETIQUETAS
  - 3.2. SELECTORES BASADOS EN CLASES
  - 3.3. SELECTORES BASADOS EN IDENTIFICADORES
  - 3.4. AGRUPAMIENTO Y ANIDAMIENTO DE SELECTORES
    - 3.4.1. AGRUPAMIENTOS
    - 3.4.2. ANIDAMIENTOS
    - 3.4.3. SELECTORES ADYACENTES
  - 3.5. SELECTORES CON PSEUDO-CLASES Y PSEUDO-ELEMENTOS
  - 3.6. SELECTORES DE ENLACES
  - 3.7. VARIABLES
  - 3.8. OTROS SELECTORES
- 4. BUENAS PRÁCTICAS AL ESCRIBIR CSS

USO DE ESTILOS

- 5. UNIDADES DE MEDIDA
- 6. ELEMENTOS: COLORES, TEXTOS, ENLACES, LISTAS, TABLAS, VISIBILIDAD, IMÁGENES
  - 6.1 ATRIBUTOS DE FUENTES
  - 6.2 ATRIBUTOS DE PÁRRAFOS
  - 6.3 ATRIBUTOS DE FONDO
  - 6.4 ATRIBUTOS DE TABLAS
  - 6.5 ATRIBUTOS DE VISIBILIDAD
  - 6.6 ATRIBUTOS DE LISTAS
- 7. ATRIBUTOS. MODELO DE CAJAS
  - 7.1. ATRIBUTOS MARGIN
  - 7.2. ATRIBUTOS PADDING
  - 7.3. ATRIBUTOS BORDER
  - 7.4. OTROS ATRIBUTOS DE CSS3
  - 7.5. ATRIBUTOS DEL CONTENIDO Y DIMENSIONES

USO DE ESTILOS

- 8. MAQUETACIÓN CON CAPAS
  - 8.1. ATRIBUTOS DE POSICIÓN
  - 8.2. SUPERPOSICIÓN DE CAJAS
  - 8.3. FLOTAR Y POSICIONAR
  - 8.4. CENTRAR ELEMENTOS
- 9. PRECEDENCIA DE ESTILOS
- 10. COMPATIBILIDAD CSS NAVEGADORES
- 11. DIRECCIONES DE INTERÉS

1 INTRODUCCIÓN A HOJAS DE ESTILO EN CASCADA (CSS, CASCADING STYLE SHEET).

- CSS (*Cascading Style Sheet*, Hojas de estilos en cascada). **Permiten separar** en el desarrollo de un sitio web lo que es el **diseño** (apariencia) de lo que son los **contenidos** (información que se quiere transmitir). Esto tiene como efecto inmediato un desarrollo y mantenimiento más eficiente de sitios web.

- Filosofía: usar la etiqueta <body> de HTML para definir las estructuras de los contenidos que se muestran en el sitio (encabezados, párrafos, viñetas, capas, etc.) y luego en otro archivo (o en el <head> del HTML, aunque es mejor la opción del archivo) se define la apariencia de cada página usando el lenguaje de CSS.

1 INTRODUCCIÓN A HOJAS DE ESTILO EN CASCADA (CSS, CASCADING STYLE SHEET).

- Ventaja: cambiar los contenidos que se pongan en el <**body**> sin afectar a la apariencia definida en el archivo CSS, o que se cambie la apariencia en el CSS sin que afecte a nada de lo puesto en el <**body**> del HTML.

- Otra ventaja: **adaptación de los sitios web a los dispositivos con los que será visualizado** (iPad, un móvil con Android, un ordenador personal…). La combinación CSS y HTML permite crear sitios web personalizados para cada dispositivo. Cuando el servidor detecta el dispositivo cliente, este aplica una hoja de estilos para un mejor visionado. Evidentemente, cada una de las hojas de estilos o regla para cada dispositivo debe haber sido diseñada a propósito por el diseñador, no hay magia.

1 INTRODUCCIÓN A HOJAS DE ESTILO EN CASCADA (CSS, CASCADING STYLE SHEET).

- La organización W3C fue la encargada de estandarizar la versión CSS como hojas de estilo en cascada, ya que es posible que un mismo contenido pueda ser regido por varios archivos CSS que controlen su apariencia. Lo que el estándar **CSS1 pretendía** era **establecer los criterios** por los que se da **preferencia a un estilo respecto a otro** (por eso lo de cascada).
- Un mismo elemento como el encabezado <h1> de la página puede estar definido en un archivo CSS externo para aparecer como un texto en azul y en la propia página definirlo como texto rojo. En este caso el navegador tiene que optar por uno u otro estilo. Por lo tanto, la especificación W3C marca las pautas de preferencia que debe respetar un navegador compatible con CSS.

1 INTRODUCCIÓN A HOJAS DE ESTILO EN CASCADA (CSS, CASCADING STYLE SHEET).

- CSS1 alcanzó el status de recomendación por la W3C en 1995. Las reglas CSS1 tienen un soporte adecuado en prácticamente todos los navegadores modernos mas conocidos.
- Las reglas CSS2 alcanzaron el status de recomendación en 1998.
- En junio de 2011 la versión CSS3 fue publicada.
- El **objetivo de que W3C estandarice CSS** es **garantizar** que el creador de un sitio web no tenga que hacer uno a propósito de cada uno de los navegadores web y dispositivos existentes.

2 CREAR Y VINCULAR HOJAS DE ESTILOS

- La primera alternativa es usando el atributo *style* dentro de las etiquetas HTML (reglas de estilos integradas). Por ejemplo:

```
<p style="background: black;">El fondo de este párrafo será negro
</p>
```

- Otra alternativa es usando la etiqueta <*style*> dentro del mismo fichero (reglas de estilo incrustadas), preferiblemente en el <head> de la página.

```
<style type="text/css" media="screen,tv">
   … código CSS
</style>
```

  - Type: se usa para indicar que se aplica un estilo formato css.
  - Media: atributo que indica sobre que dispositivo se aplicarán los estilos. El valor **handheld** (para móviles), **print** (para salida de impresora), **projection** (proyectores), **screen** (pantallas), **tv** (televisores), **braille** (para dispositivos braille) o **all** (todos los dispositivos, se puede omitir). Si hay varios separar por comas (,)

2 CREAR Y VINCULAR HOJAS DE ESTILOS

- Aunque la opción más eficiente es emplear una hoja de estilos externa que se puede reutilizar en varios documentos HTML, permitiendo que todo un sitio web siga el mismo CSS.

- Una hoja de estilos externa puede ser enlazada a un documento HTML mediante la etiqueta <link> que se recomienda colocar en el <head> de la página.

```
<link rel=stylesheet href="estilo.css" type="text/css"
   media="screen, print"
```

- Cuando se define un CSS externo, la página no debe contener ninguna etiqueta HTML con style. Solo debería consistir en reglas de estilo o sentencias. Un archivo de ejemplo que solo consista en una línea podría ser:

```
p { margin:2em }
```

2 CREAR Y VINCULAR HOJAS DE ESTILOS

- De la declaración anterior

```
<link rel="stylesheet" href="estilo.css" type="text/css"
   media="screen, print"
```

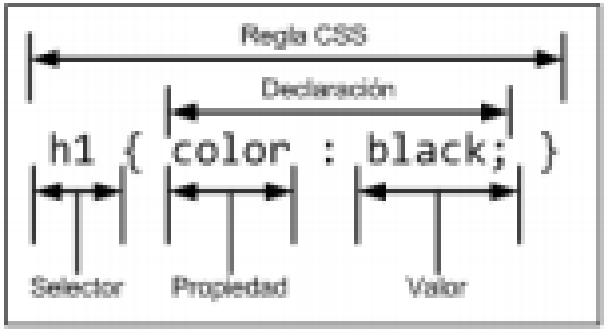
- **href** se refiere a la ruta relativa de la hoja de estilos, **type** para indicar que se aplican estilos de CSS y **media** tiene el mismo significado y valores que en la etiqueta <style>

- Es posible incluir más de una hoja de estilo en un sitio web y ofrecer al usuario (si el navegador lo permite) la posibilidad de seleccionar una u otra mediante el uso de **hojas de estilo alternativas**. Esto sería con el valor alternate stylesheet.

```
<link rel="alternate stylesheet" href="estilo.css"
type="text/css" media="screen, print"
```

3 SELECTORES

- Una hoja de estilos CSS está formada por *reglas* que indican la manera en la que se visualizará la página (reglas de estilo). Cada regla está compuesta de ***selectores***, los cuales indican a qué elemento o parte de una página se aplica un determinado estilo.
- La estructura es la siguiente:



3 SELECTORES

- Los diferentes términos se definen a continuación:

selector { propiedad: valor }

- **Selector:** Indica el elemento o elementos HTML a los que se aplica la regla CSS.
- **Propiedad:** Característica que se modifica en el elemento seleccionado, como por ejemplo su tamaño de letra, su color de fondo, etc.
- **Valor:** Establece el nuevo valor de la característica modificada en el elemento. Un archivo CSS puede contener infinitas reglas CSS, cada regla puede contener infinitos selectores y cada declaración puede estar formada por un número infinito de pares propiedad/valor, como se explicará en la definición de cada una de las propiedades.

3 SELECTORES

- CSS contempla sintaxis para definir atributos para varios selectores y selectores con varios atributos.

```
Selector1, Selector2, {propiedad1:valor1;
                       propiedad2:valor2}
```

- A cualquier etiqueta se le puede asignar un estilo pero la descripción de las reglas debe ajustarse a la sintaxis definida. **Si un navegador encuentra un selector cuya sintaxis no comprende, este ignorará la declaración entera y continúa con la siguiente,** con independencia de si el error afecta a la propiedad, al valor o a toda la descripción.

3.1. SELECTORES BASADOS EN ETIQUETAS

- La manera mas sencilla es usar las propias etiquetas HTML como selectores. Consiste en asociar a cada etiqueta una declaración del estilo que se le aplica al selector (etiqueta HTML)

- Ejemplo: **`h1 {color:blue}`**

- Si el mismo estilo se le quiere aplicar a dos selectores distintos, la sintaxis sería:

```
h1, h2 {color:blue}
```

- Y si al mismo selector se le quiere aplicar atributos diferentes:

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

3.1. SELECTORES BASADOS EN ETIQUETAS

- Una vez conocida la sintaxis de las reglas, la siguiente pregunta sería. ¿Dónde se deben poner esas declaraciones para que el navegador las interprete? En cualquiera de las opciones del apartado 2, por ejemplo para visualizarlo de forma rápida debajo se incluye en <style>, pero hay que recordar que no es lo más eficiente.

```
<head>
<title></title>
<style>
   h1 {color: blue; background-color:red }
</style>
</head>
```

- *Cuando el navegador lee la cabecera del HTML interpreta que todas las etiquetas <h1> incluidas en el <body> tendrán color azul y el color de fondo rojo.*

3.2. SELECTORES BASADOS EN CLASES

- Las **clases asociadas a etiquetas** HTML se definen:

```
Nombreetiqueta.nombreclase {propiedad:valor;
   propiedad:valor; … }
```

- Esta alternativa es más potente que la vista con selectores basados en etiquetas, ya que permite aplicar a las mismas etiquetas diferentes estilos.
- Para indicar en cada etiqueta qué estilo se le quiere aplicar se usa el atributo class de la siguiente manera:

```
<h1 class="nombre clase"> Un encabezado<h1>
```

3.2. SELECTORES BASADOS EN CLASES

- Ejemplo:

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

Un encabezado rojo

Ahora azul

y ahora verde

3.2. SELECTORES BASADOS EN CLASES

- Es posible emplear **clases genéricas** que no se aplican a ninguna etiqueta HTML, por lo que en su descripción se omite en el selector el nombre de ninguna etiqueta.

.nombreclase {atributo:valor; atributo:valor; … }

```
<head>
   <style>
      .verde { color:green}
   </style>
</head>
```

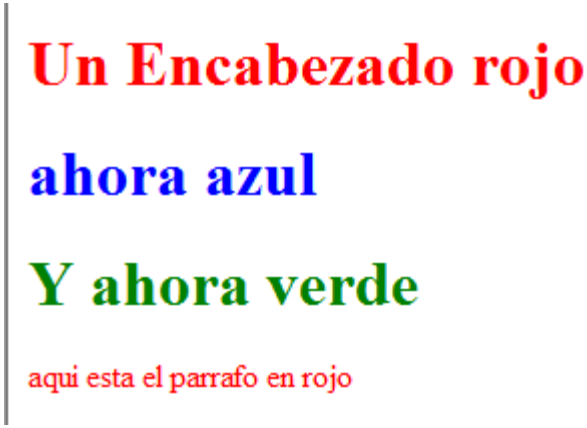
Se podría aplicar el estilo .verde a cualquier contenido en el <body> ya sea un, párrafo <p>, encabezado<h1> o bloque :

```
<p class="verde">Párrafo verde</p>
<h1 class="verde">h1 verde.</h1>
<div class="verde"> div verde </div>
```

3.2. SELECTORES BASADOS EN CLASES

- *Con la siguiente declaración:*

```
<head>
   <style>
      .rojo { color: red }
      .verde {color: green}
      .azul {color: blue}
   </style>
</head>
```



- *Y aplicando las clases a las diferentes etiquetas <h1>en el <body>:*

```
<h1 class="rojo">Un Encabezado rojo</h1>
<h1 class="azul">ahora azul</h1>
<h1 class="verde">Y ahora verde</h1>
```

- *Se pueden aplicar a cualquier otra etiqueta, por ejemplo al párrafo <p>*

```
<p class="rojo">aquí está el párrafo en rojo.</p>
```

3.2. SELECTORES BASADOS EN CLASES

- Una característica de las clases es que se puede asignar más de una clase a una misma etiqueta, siempre y cuando no hay conflicto entre ellas.

```
<head>
   <style>
      .textorojo { color: red }
      .fondoazul { background-color: blue }
   </style>
</head>
```

título en rojo, fondo azul

color en rojo; fondo: el que herede de la pagina.

Se podría hacer el siguiente contenido en el <body> separándolos con un espacio en blanco:

```
<h3 class="textorojo fondoazul">título en rojo, fondo azul</h3>
<p class="textorojo">color en rojo; fondo: el que herede de la página.</p>
```

3.3. SELECTORES BASADOS EN IDENTIFICADORES

- A diferencia de las clases, **los identificadores solo se pueden usar en un único elemento**.

```
Nombreetiqueta#nombreid {atributo:valor;
  atributo:valor; … }
#nombreid {atributo:valor; atributo:valor; … }
```

- Como puede apreciarse, esta declaración es idéntica a la usada para definir selectores de clase. La única diferencia es que en vez de usar un punto **"."** se usa una almohadilla **"#"**. Sin embargo, la semántica de ambas expresiones es muy parecida.

3.3. SELECTORES BASADOS EN IDENTIFICADORES

```
<style>
   #rojo { color:red;}
</style>
</head>
…
<body>
   <p id="rojo"> el párrafo va en rojo</p>
</body>
```

- En este código el navegador lo interpreta mostrando en color rojo el párrafo.
- Si cambiamos **id** por **class** y **#** por **.** se aplicarían selectores de clase. Aparentemente no hay diferencia entre identificadores y clases pero esto no es así. En CSS las clases se pueden usar en uno o varios elementos (<h1><h2><p>…) sin embargo y esta es la diferencia **los identificadores sólo se deben usar en un único elemento**. Esto es porque un identificador es como si hiciese referencia a un objeto de estilo y los objetos son únicos, por tanto solo un elemento puede coger ese objeto de estilo.

3.3. SELECTORES BASADOS EN IDENTIFICADORES

- Los identificadores se suelen usar en casos como el siguiente en donde los identificadores cabecera, contenidos y pie de página se definen el estilo de esas tres partes de la pagina web.
- Los tres objetos ** ** son asociados a tres identificadores diferentes ya que no tiene sentido que esos identificadores se repitan en varios elementos.

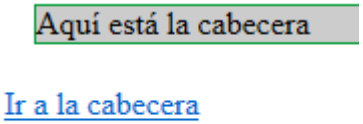
```
<div> id="cabecera"></div>
<div> id="contenido"></div>
<div> id="piepagina"></div>
```

- En definitiva las clases se usan cuando el estilo se quiere aplicar a mas de un elemento, mientras que los identificadores se definen cuando lo que se busca es **exclusividad** ya que solo será aplicada a un elemento.

3.3. SELECTORES BASADOS EN IDENTIFICADORES

- En muchos de los navegadores actuales si se usa un identificador en varios elementos estos se interpretan y devuelven un resultado idéntico al que se devuelve si se hiciese con clases. Sin embargo, son "**buenas prácticas**" de diseñador considerar el identificador para un único elemento y las clases para elementos con estilos en común

```
<head>
<style>
    #cabecera {
       background:#CCC;
       border:1px solid #093;
       margin:10px 12px 20px 15px;
    }
 </style>
 </head>
 <body>
    <div id="cabecera"> Aquí está la cabecera </div>
     <a href="#cabecera"> Ir a la cabecera </a>
 </body>
```



3.4. AGRUPAMIENTO Y ANIDAMIENTO DE SELECTORES

- Los selectores se pueden agrupar y anidar para conseguir estilos CSS, por un lado más concretos y definidos, y por otro lado para tener un fichero CSS más optimizado y fácil de entender por el equipo de desarrollo.

- Cualquier selector se puede agrupar siguiendo esta sintaxis:

```
Selector1, Selector2 {atributo1:valor1;
atributo2:valor2;…}
```

3.4.1. AGRUPAMIENTOS

- De esta manera se puede aplicar el mismo estilo a un conjunto de selectores al mismo tiempo. Por ejemplo, si se quiere aplicar un tipo de fuente Arial a etiquetas <h1>, <p> y <h3>, la regla sería:

```
h1, p, h3 {font-family:arial}
```

- La versión menos optimizada de esa misma regla sería:

```
h1 {font-family: arial;}
p {font-family: arial;}
h3 {font-family: arial;}
```

- Para identificadores y clases sería:

```
#cabecera, #piepagina {color:#00FF00}
.cabecera, h2, #micolor {color:#FF0000}
```

3.4.2. ANIDAMIENTOS

- **Selector anidado común**: se usa para crear reglas sobre elementos que están rodeados de otros elementos. La sintaxis general de este tipo de anidamiento para dos selectores es la siguiente:

```
SelectorX  SelectorY {atributo1:valor1;
  atributo2:valor2;…}
```

- El siguiente estilo aplicado al código HTML de debajo visualizará ambos textos en rojo con independencia de las etiquetas <h1> y

Un texto h1 en negrita y rojo

Un texto h1 en negrita

```
a {color:red; font-weight: bold}
```

```
<body>
   <h1><span><a> Un texto h1 en negrita y rojo </a></span> </h1>
   <h1><a> Un texto en negrita</a></h1>
</body>
```

3.4.2. ANIDAMIENTOS

Selector anidado común

- Sin embargo, usando un anidamiento en la definición de la regla se consigue el efecto deseado, ya que solo aparece en rojo aquel texto en negrita rodeado por una etiqueta a, span y otra h1

```
<head>
   <style>
      h1 span a {color:red; font-weight: bold}
   </style>
</head>
<body>
   <h1><span><a> Un texto h1 en negrita y rojo </a></span></h1>
   <h1><a> Un texto en negrita </a></h1>
</body>
```

Un texto h1 en negrita y rojo

Un texto h1 en negrita

3.4.2. ANIDAMIENTOS

Selector anidado común

- En principio no hay límite en el numero de anidamiento que se puede hacer, pero no es recomendable de mas de 4 o 5 por motivos de complejidad a la hora de interpretar el CSS, tanto por el navegador como por los diseñadores.
- Se pueden hacer agrupamientos con clases, id o combinaciones, no sólo con selectores basados en etiquetas.
- Si suponemos que hemos definido varias reglas con anidamiento h1 span a y h2 a, el agrupamiento seria así:

```
h1 span a, h2 a { color: red}
```

- Que equivaldría a esto:

```
h1 span a {color: red}
h2 a {color:red}
```

3.4.2. ANIDAMIENTOS

Selector anidado común

- El anidamiento común se comporta igual si las etiquetas están consecutivas o si hay etiquetas intermedias.

```
<head>
<style>
  h1 a {color:red}
</style>
</head>
<body>
     <h1><span><a> Un texto h1 en negrita y rojo</a></span> </h1>
     <h1><a> Un texto en h1 en negrita</a></h1>
</body>
```

Un texto h1 en negrita y rojo

Un texto h1 en negrita

3.4.2. ANIDAMIENTOS

Selector anidado común

- Los ejemplos anteriores combinan selectores a nivel de etiquetas. Sin embargo, lo más común es que el primer selector sea a nivel de identificador, clase o cualquier otro que sirva para localizar a elementos específicos de la página.
- Una vez identificados, en este caso se pueden emplear a partir del segundo nivel selectores a nivel de etiquetas que nos evitan crear más clases de lo debido, ya que tenemos identificadas las etiquetas dentro de un contenedor y solo se aplicarán en este contexto.

```
<head>
<style>
   #contenedor span a{ text-decoration: none;}
</style>
</head>
<body>
   <div id="contenedor">
    <span><a href="#">…</div>
</body>
```

3.4.2. ANIDAMIENTOS

Anidamiento de selectores hijos

- Si lo que se desea es restringir que las etiquetas, **además de estar en el mismo contexto, estén seguidas unas de otras**, entonces la sintaxis que tiene que usar en la definición es la siguiente (para anidamiento de selectores).
- De esta manera mostrará solo en rojo el texto que tiene la etiqueta <a> dentro de <h1> sin ninguna entre medias. Solo el segundo texto aparecerá en rojo al probarlo en un navegador**.**

SelectorX > SelectorY {atributo1:valor1; atributo2:valor2;…}

```
<head>
<style>
   h1>a {color:red}
</style>
</head>
<body>
   <h1><span><a> Un texto h1 en negrita</a></span> </h1>
   <h1><a> Un texto en h1 en negrita y rojo </a></h1>
</body>
```

Un texto h1 en negrita

Un texto h1 en negrita y rojo

3.4.2. ANIDAMIENTOS

Anidamiento de selectores hijos

- El siguiente código muestra un ejemplo que incluye mínimo 3 etiquetas. Solo el segundo texto aparecerá en rojo.

```
<head>
<style>
   h1>span>a {color:red}
</style>
</head>
<body>
   <h1><span><a> Un texto h1 en negrita y rojo </a></span> </h1>
   <h1><span><label><a> Un texto en h1 en
      negrita</a></label></span></h1>
</body>
```

Un texto h1 en negrita y rojo

Un texto h1 en negrita

- Los anidamientos, sean del tipo que sean, también pueden usarse con **class, id** o combinaciones. Por ejemplo “h1.clase span#id a” o “h1.clase > span#id > a”

3.4.3. SELECTORES ADYACENTES

- **Anidamiento de selectores adyacentes:** este tipo de anidamiento se usa cuando se quiere aplicar un estilo a un elemento que tiene adyacente (al lado) a otro elemento en el mismo nivel del código HTML.

  SelectorX + SelectorY {propiedad1:valor1; propiedad2:valor2;…}

```
<head>
<style>
   label+span {color: red}
   h5 + p {font-weight: bold}
</style>
</head>
<body>
   <h5> <label>Nota</label>, esto es una <span>advertencia</span> </h5>
   <p> Leer detenidamente </p>
</body>
```

Nota, esto es una **advertencia**

Leer detenidamente

- Como se aprecia solo se aplicara el estilo sobre si hay una etiqueta adyacente <label>. Solo la palabra advertencia aparece en rojo en el navegador.
- Al igual que el resto de anidamientos, en el ejemplo se han hecho con selectores basados en etiquetas, pero **se pueden hacer con class, id o combinadas** (ejemplo label.titulo + span#id1)

3.4.3. SELECTORES ADYACENTES

- El símbolo + solo se aplica al primer elemento que encuentra que cumple las condiciones indicadas anteriormente. Si se quiere aplicar un estilo a todos los elementos adyacentes, habría que utilizar el operador ~ (y sigue funcionando si hay solo un elemento)

  SelectorX ~ SelectorY {propiedad1:valor1; propiedad2:valor2;…}

```
<head>
<style>
   label~span {color: red}
   h5~p {font-weight: bold}
</style>
</head>
<body>
   <h5> <label>Nota</label>, esto es una <span>advertencia</span> </h5>
   <p> Leer detenidamente </p>
   <p>Siga leyendo detenidamente (también negrita por el operador ~)</p></p>
</body>
```

Nota, esto es una advertencia

Leer detenidamente

Siga leyendo detenidamente (también negrita por el operador ~)

- Si aplicáramos h5 + p, solamente aparecería en negrita el primer párrafo que viene después del h5

3.5. SELECTORES CON PSEUDO-CLASES Y PSEUDO-ELEMENTOS

- Las **pseudo-clases** permiten cambiar el estilo dependiendo del estado en el que se encuentra. Su sintaxis es :estilo y se puede aplicar, al igual que las clases e identificadores, sobre todo elemento (por ejemplo a:link, span:hover)
- Uno de los más empleados son los selectores a nivel de enlace sobre los que se profundizará en el apartado 3.6.
- Otro es el estilo :hover, que se aplica cuando nos posicionamos encima de un elemento. Ver ejemplo: https://www.w3schools.com/css/tryit.asp?filename=trycss_pseudo-class_hover_tooltip
- Otro es first-child, que permite aplicar un estilo al primer elemento dentro de un estilo de agrupamiento (por ejemplo p span:first-child se aplica al primer span que tenga una etiqueta p por encima, pero solo al primero)
- https://www.w3schools.com/css/tryit.asp?filename=trycss_first-child2

3.5. SELECTORES CON PSEUDO-CLASES Y PSEUDO-ELEMENTOS

- Los **pseudo-elementos** se emplean para aplicar estilos a partes específicas de una etiqueta y la sintaxis es elemento::pseudo-elemento (por ejemplo p::first-line se aplica solo a la primera línea de un párrafo, o se podría hacer a nivel de clase con p.clase::first-line)
- Ver ejemplo de selectores que se aplican a primera línea o primera letra de un elemento: https://www.w3schools.com/css/tryit.asp?filename=trycss_firstline_letter
- Otros de los más empleados son ::after y ::before que permiten insertar contenido al principio o final de una etiqueta:
  - https://www.w3schools.com/css/tryit.asp?filename=trycss_before
  - https://www.w3schools.com/css/tryit.asp?filename=trycss_after
- Y el más reciente ::selection que cambia el estilo de una parte de la página cuando la seleccionamos https://www.w3schools.com/css/tryit.asp?filename=trycss3_selection

3.6 SELECTORES DE ENLACES

Los enlaces en CSS se pueden manejar con bastante libertad comparado con HTML. CSS permite definir estilos en los enlaces, quitando el subrayado o hacer enlaces en la misma página con distintos colores. Para aplicar estilo a los enlaces debemos definirlos para los distintos tipos de enlaces, que son:

- **Enlaces normales**: `a:link {atributos}`
- **Enlaces visitados**: `a:visited {atributos}`
- **Enlaces activos**: `a:active {atributos}` *(Los enlaces están activos en el preciso momento en que se pincha sobre ellos)*
- **Enlaces hover**: `a:hover {atributos}` *(Cuando el ratón está encima de ellos)*

Por ejemplo, un atributo típico asociado a los enlaces es `text-decoration: none` para quitar el subrayado.

3.7 VARIABLES

Es posible crear variables en CSS a través del selector especial denominado **:root**. Ahí se pueden definir, siempre que tengan valores válidos en el estilo al que se asignen posteriormente. La sintaxis del nombre de las variables es “--variable”. Por ejemplo:

```
<style>
  :root {
     --main-bg-color: coral;
     --main-padding: 15px;
  }
</style>
```

Para hacer referencia a las variables posteriormente, necesitamos emplear la función var(--nombre_variable).

```
<style>
   #div1 {
      background-color: var(--main-bg-color);
      padding: var(--main-padding);
   }
</style>
```

3.8. OTROS SELECTORES

- Uno de los más empleados, no mencionado hasta ahora, es el selector a nivel de atributo: https://www.w3schools.com/cssref/tryit.asp?filename=trycss_sel_attribute

- Ver más en este enlace: https://www.w3schools.com/cssref/css_selectors.asp

4. BUENAS PRÁCTICAS AL ESCRIBIR CSS

- /* hay que poner comentarios con esta sintaxis (son multilínea) */
- Los **selectores** se nombran en **minúsculas**, **nunca empezando por caracteres especiales o numéricos**.
- El **nombre** de los selectores debe ser **específico y claro**, para que tenga una mayor capacidad expresiva.
- El nombre de **los identificadores no debe describir una característica visual**, como color, tamaño o posición.
- Los nombres deben seguir más una **visión semántica** que estructural. Para facilitar los cambios.
- **Separa** las palabras mediante guiones (_ o -) o mayúsculas. Lo más común hoy en día es emplear guiones (-) y minúsculas. Por ejemplo, las clases con nombre “letra-negrita”, “seccion-principal”, etc.
- No hacer uso excesivo de clases. En muchas ocasiones se puede organizar el código de forma más eficiente con agrupamientos o adyacencia.

4. BUENAS PRÁCTICAS AL ESCRIBIR CSS

- **Agrupar las reglas según su selector** siempre que sea posible. Agruparlas unas debajo de otras:

```
[…]
table {border:double}
table.miembros {border:solid}
table.empleados{border:grrobe}
[…]
```

- Al principio de un CSS es aconsejable definir los selectores de etiquetas. Además de usar **comentarios** para dejar claro cual es la parte que definen las clases y otros elementos.

```
/* Etiquetas html */
body {
   font-family:Arial, verdana, sans serif; font-size:13px;}
   h1,h2,h3,h4,h4,h6,form,input,text-area{
  border:0; padding:0; margin:0;
   font-family: Arial;
} /* FIN de Etiquetas HTML
```

- **Estructurar visualmente los atributos**. Si un elemento solo tiene 3 atributos se pueden poner en la misma línea. Pero si hay más se escriben en líneas diferentes sangrados con tabuladores.

5 UNIDADES DE MEDIDA

- Los valores de cualquier atributo relacionado con medidas se pueden expresar en varias unidades:
  - Absolutas:
    - Pulgadas (In). Una Pulgada = 2,54 Cm.
    - Centímetros (Cm). Milímetros (Mm).
    - Puntos (Pt). Un Punto = 1/72 De Pulgada.
    - Picas (pc). Una pica = 12 puntos.
    - Píxeles (**px**) (dependen de la resolución de pantalla. Por ejemplo, Windows tiene una resolución de 96px por pulgada y Macintosh 72px por pulgada, pero se suele considerar absoluta porque una vez que se asigna un tamaño en píxeles siempre tendrá esa medida independientemente del contenedor)
  - Relativas:
    - **em** (relativo al *font-size* del elemento en el que se usa)
    - **rem** (relativo al *font-size* del elemento raíz de la página, es decir <html>)
    - **%** es relativo al tamaño del elemento raíz donde está situada la etiqueta.
    - **vw** y **vh**. Respectivamente, el ancho y alto del viewport (área de visualización)
    - **vmin** y **vmax**. El ancho máximo o mínimo del área de visualización dependiendo de la orientación (se verá con más detalle adelante).
- Ejemplos:

```
div { font-size:15px;}
p { font-size:1.2em;}  /* 1.2 veces el tamaño de la letra */
p { width: 100vw;}  /* El 100% del área de visualización */
```

5 UNIDADES DE MEDIDA

- Aunque es la unidad más empleada, hoy se recomienda no maquetar con píxeles al diseñar para una gran variedad de pantallas y dispositivos. Como mucho emplearemos px para bordes, algún margin o padding pequeño que no afecte al diseño principal o sombra.
- Hay que tener en cuenta que si se define un tamaño en píxeles este será siempre fijo, una vez que se calcula en función de los puntos por pulgada. Las alternativas recomendadas son las siguientes:
  - **Porcentajes:** Se utilizan para definir contenedores flexibles. Es decir, si cambiamos de un dispositivo de ancho 1200px a otro de 768px, si se define un ancho del 70% se adaptará a las nuevas medidas.
  - **La unidad em:** Es escalable y siempre depende de su elemento padre. Por ejemplo, si el elemento body tiene un tamaño de fuente de 16px y un elemento hijo tiene una fuente con tamaño 1.3em, este texto se mostrará de un tamaño un 30% más grande que el del body (20.8px), mientras que si dentro de ese elemento tenemos otro hijo con un font-size de 1.3 em, el tamaño de fuente de este objeto sería un 30% más grande que el tamaño de su padre (27.04px).

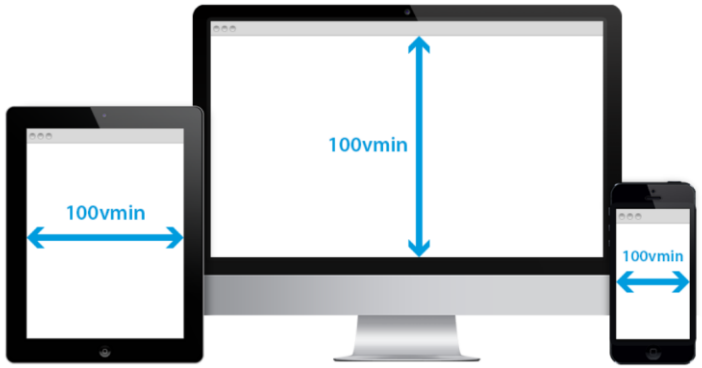
Es recomendable usar la unidad de medida **em** para definir los tamaños de fuente, los altos de línea y también para elementos de diseño que no requieran ser muy exactos o que requieran una medida que tenga relación con el tamaño del texto, como por ejemplo el margen entre párrafos. También se puede aplicar a elementos generales del layout aunque no es muy recomendable, ya que si eventualmente se cambia el tamaño de fuente de uno de ellos, se podría estropear el diseño.

5 UNIDADES DE MEDIDA

- **La unidad rem:** La unidad de medida rem es muy similar a em, con la única diferencia de que no es escalable, esto quiere decir que no depende del elemento padre, sino del elemento raíz del documento, el elemento HTML. Esto significa que si el elemento HTML tiene un tamaño de fuente de 16px (como es por defecto), entonces 1rem, sería igual a 16px en toda la página.

  Esta unidad de medida es recomendable para aplicar a elementos del layout que requieran medidas fijas sin depender del contenedor y eventualmente también para textos que deseemos que tengan un tamaño de fuente que no dependa de su elemento padre.

- **Viewport:** Su utilidad, entre muchas otras posibilidades, es lo que hasta ahora ha sido siempre complicado de conseguir, esto es, establecer una altura del 100% de altura a nuestras capas, que no siempre funcionaba con "height:100%". Otra opción interesante con las unidades relacionadas **vmin** y **vmax**, que consisten en el valor porcentual mínimo o máximo de la ventana gráfica, ya sea en altura o anchura. Es decir, que dependiendo de la orientación del dispositivo, obtenemos el valor en porcentaje de altura o de anchura, el que menos o más mida en ese momento



6. ELEMENTOS: COLORES, TEXTOS, ENLACES, LISTAS, TABLAS, VISIBILIDAD, IMÁGENES.

- A continuación se explicarán una serie de atributos para elementos de la página como colores de texto, de fondo, fuentes, enlaces, listas, tablas, imágenes o cómo ocultar elementos.

6.1 ATRIBUTOS DE FUENTES

- **Color:** RGB o nombre de color. Sirve para indicar el color del texto. Lo admiten casi todas las etiquetas de HTML. Admite nombres de colores en inglés para algunos colores concretos y valores RGB y hexadecimal para todos.
  - Listado de colores predefinidos: https://www.w3schools.com/cssref/css_colors.asp
  - Formato RGB: Es una función que recibe tres parámetros con números enteros de 0 a 255 que indican cada nivel de intensidad de rojo, verde y azul respectivamente. Ejemplo: `p {color: rgb(25,200,43)}`
  - Formato RGBA: Añade un cuarto parámetro con números decimales entre 0 y 1 que se refiere a la opacidad. Cuanto menor sea este valor, menos opaco (0 oculta el color). Ejemplo. `p {color: rgba(25,200,43,0.3)}`
  - Formato hexadecimal: Es igual que RGB, pero transformando cada valor en hexadecimal (dos dígitos hexadecimales por cada número entero de RGB). Por ejemplo, del estilo definido en RGB el equivalente sería `p {color: #19C82B}`

6.1 ATRIBUTOS DE FUENTES

- **font-size:** `unidades \ xx-small \ x-small \ small \ médium \ large \ x-large \ xx-large`. Sirve para determinar el tamaño de una fuente. Ejemplo: `span.clase {font-size: 16px}`
- **font-weight:** `normal \ bold \ bolder \ lighter \ 100 \ 200 \ 300 \ 400 \ 500 \ 600 \ 700 \ 800 \ 900`. Sirve para definir la anchura de los caracteres, es decir, efecto de negrita. Normal y 400 son el mismo valor, así como bold y 700.
- **font-style:** `normal \ italic \ oblique`. Es el estilo de la fuente. El estilo obligue es similar al italic (ambos cursiva).

6.1 ATRIBUTOS DE FUENTES

- **font-family:** `serif \ sans-serif \ cursive \ fantasy \ monospace`. Con este atributo indicamos la familia de tipografía del texto. Los primeros valores son genéricos, es decir, los navegadores los comprenden y utilizan las fuentes que el usuario tenga en su sistema. También se pueden definir con tipografías normales (Arial, Times, Courier, etc.). Si el nombre de una fuente tiene espacios se utilizan comillas para que se entienda bien.
- Se pueden indicar varias alternativas entre comas por si el navegador no entiende alguna de ellas. Por ejemplo:

```
p.a {font-family: "Times New Roman", Times, serif}
```

6.1 ATRIBUTOS DE FUENTES

- Con la regla @ **font-face**, diseñadores web ya no tienen que utilizar una de las fuentes predefinidas:
  - En primer lugar hay que incluir un archivo de fuente en la regla font-face.

    `src: url ('Sansation_Light.ttf')` ruta relativa o se puede emplear una dirección URL completa en su lugar como se indica:

    ```
    src: url
    ('http://www.w3schools.com/css3/Sansation_Light.ttf').
    ```

  Ej:

```
@font-face{
    font-family: myFirstFont; src:
    url('Sansation_Light.ttf'),
    url('Sansation_Light.eot');
    /* Se pueden indicar dos URL por si falla alguna */
}
a {font-family: myFirstFont} /* Aquí se referencia a la
fuente creada arriba */
```

6.1 ATRIBUTOS DE FUENTES

- **text-decoration:** *none* \ *underline* \ *overline* \ *line-through.* Para establecer la decoración de un texto, es decir, si está subrayado, sobre-rayado o tachado.
- **text-align:** *left* \ *right* \ *center* *justify.* Sirve para indicar la alineación del texto. Es interesante destacar que las hojas de estilo permiten el justificado de texto, aunque no funciona en todos los navegadores.
- **text-shadow:** Efecto de sombra. Los parámetros son: sombra horizontal, sombra vertical, borrosidad, color. Ejemplo:

```
text-shadow: 5px 5px 5px #FF0000;
```

http://www.w3schools.com/cssref/playit.asp?filename=playcss_text-shadow

6.2 ATRIBUTOS DE PÁRRAFOS

- **text-indent:** Unidades. Un atributo que sirve para hacer sangrado o márgenes en las páginas.

http://www.w3schools.com/cssref/tryit.asp?filename=trycss_text-indent

- **line-height:** normal \ unidades. El alto de una línea, y por tanto, el espaciado entre líneas. Ejemplo: `p {line-height: 10px}`

https://www.w3schools.com/cssref/tryit.asp?filename=trycss_line-height

6.3 ATRIBUTOS DE FONDO

- Se pueden resumir los atributos que veremos a continuación en una línea con el siguiente formato:

```
background: color image position/bg-size bg-repeat
bg-origin bg-clip bg-attachment
```

  - **Background-color:** RGB, RGBA o nombre de color. Sirve para indicar el color de fondo de un elemento de la página.
  - **Background-image:** Nombre de la imagen con su camino relativo o absoluto. `background-image:url('smiley.gif');`

    Es posible incluir varias imágenes superpuestas de la siguiente manera:

    ```
    background:url(img_tree.gif),url(img_flwr.gif);
    ```
  - **background-repeat:** Si se repite o no, siempre que la imagen ocupe más que su contenedor. Por defecto, repeat. El valor no-repeat es para que la imagen aparezca solo una vez

6.3 ATRIBUTOS DE FONDO

- Las propiedades que vienen a continuación están especialmente relacionadas con background-image con el valor no-repeat.
  - **background-position:** Posición horizontal y vertical de la imagen. Los valores pueden ser left, center, right, top o bottom o una unidad de medida que indica lo que se quiere desplazar la imagen desde su posición inicial.

    `background-position: left right` (o 25% 50% con unidades, siendo 25% la coordenada horizontal y 50% la vertical. También valdría con otras unidades como px o em, por ejemplo 100px 200px)
  - **background-origin:** Donde se empieza a posicionar la imagen. A partir del propio contenido del elemento (content-box), a partir de padding (padding-box) y a partir del borde (border-box).
  - **background-clip:** Con los mismos valores que background-origin, pero en este caso especifica cuánto se extiende el fondo definido.

6.3 ATRIBUTOS DE FONDO

- **background-size:** Cuánto ocupa la imagen de ancho y alto (100% 100% en porcentajes, o 100px 200px con otra unidad de medida). Otros valores importantes son contain y cover. El valor contain indica que la imagen se adapta al contenedor sin ocultarse nada de la imagen, mientras que el valor cover cubre todo el contenedor con la imagen aunque no se vea en su totalidad.
- **background-blend-mode:** Cómo se mezcla la imagen de fondo con el resto del contenido. normal|multiply|screen|overlay|darken|lighten|color-dodge|saturation|color|luminosity
- **background-attachment:** A tener en cuenta el valor fixed, que permite que la imagen de fondo se muestre aunque se haga scroll.

6.4 ATRIBUTOS DE TABLAS

- **caption-side:** *valores top \ bottom.* Posición del título (arriba o abajo).
- **table-layout:** *auto \ fixed.* Control del algoritmo usado para el formato de las celdas, filas y columnas.
- **border-collapse:** *collapse \ separate.* Si los diferentes bordes de cada celda se unen, o aparecen por separado.

border-collapse: separate

Author Name	Contact No
Geek	XXXXXXXXXX
GFG	XXXXXXXXXX

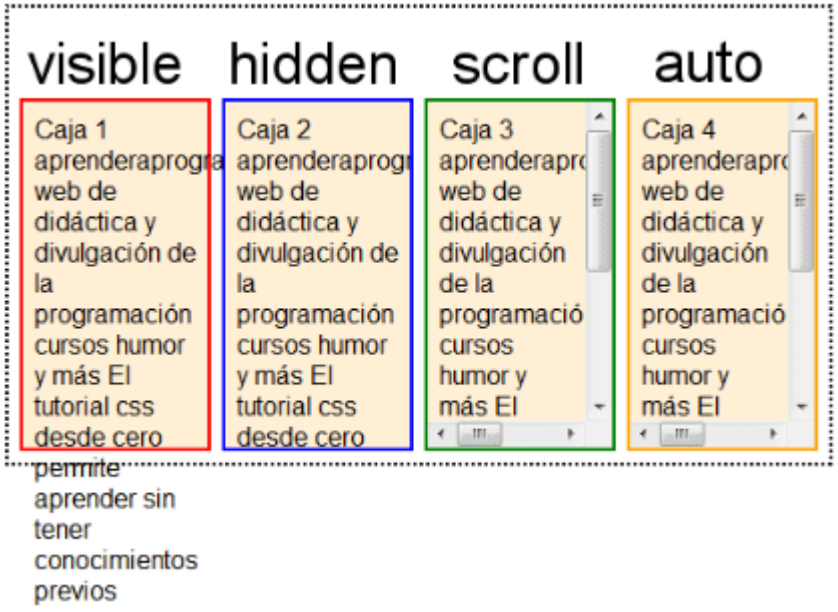
border-collapse: collapse

Author Name	Contact No
Geek	XXXXXXXXXX
GFG	XXXXXXXXXX

- **border-spacing:** *horizontal vertical.* Espaciado entre los bordes de celdas adyacentes. Solo funciona con `border-collapse: separate`
- **empty-cells:** *show \ hide.* Visibilidad de los bordes de celdas sin contenido (ocultar o mostrar).
- **vertical-align:** *top \ bottom \ middle.* Si una celda tiene un alto con height, especifica si el texto se muestra arriba, debajo o en el medio.

6.5 ATRIBUTOS DE VISIBILIDAD

- **Overflow:** *visible* \ *hidden* \ *scroll* \ *auto.* Comportamiento del contenido si se desborda el contenedor. Con visible desborda, con hidden se oculta el contenido, con scroll siempre muestra el scroll (vertical y horizontal), mientras que con auto solo muestra el scroll cuando es necesario. Es posible definir los valores overflow-x u overflow-y, para el desbordamiento horizontal o vertical, respectivamente.

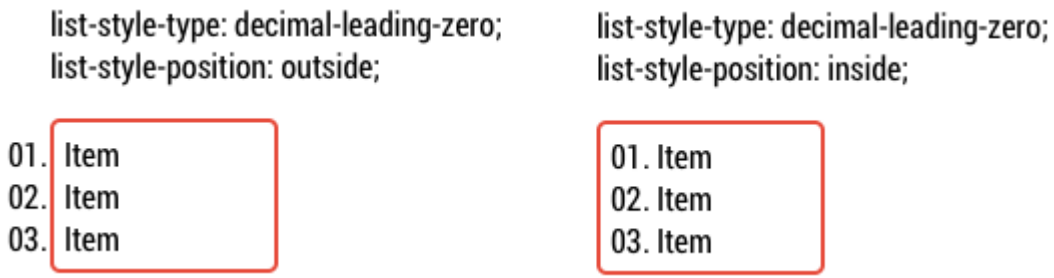


6.5 ATRIBUTOS DE VISIBILIDAD

- **Display:** *inline* \ *block* \ *contents* \ *flex* \ *grid* \ *inline-block* \ *inline-flex* \ *inline-grid* \ *inline-table* \ *list-item* \ *run-in* \ *table* \ *table-caption* \ *table-column-group* \ *table-header-group* \ *table-footer-group* \ *table-row-group* \ *table-cell* \ *table-column* \ *table-row.* Por un lado, `display: none` oculta un elemento en la página siendo ocupado por el siguiente. Por otro lado, esta propiedad permite cambiar el comportamiento por defecto de las etiquetas de HTML. Por ejemplo, un tiende a ocupar un bloque (`display: block`) o un lo que ocupe su contenido (`display: inline`). Esta propiedad puede hacer que un se comporte inline o incluso una combinación de block e inline (inline-block).
- **Visibility:** *visible* \ *hidden.* Visibilidad de las cajas. Si el valor es hidden, el espacio del elemento no visible no se ocupa por otro y se queda en blanco.

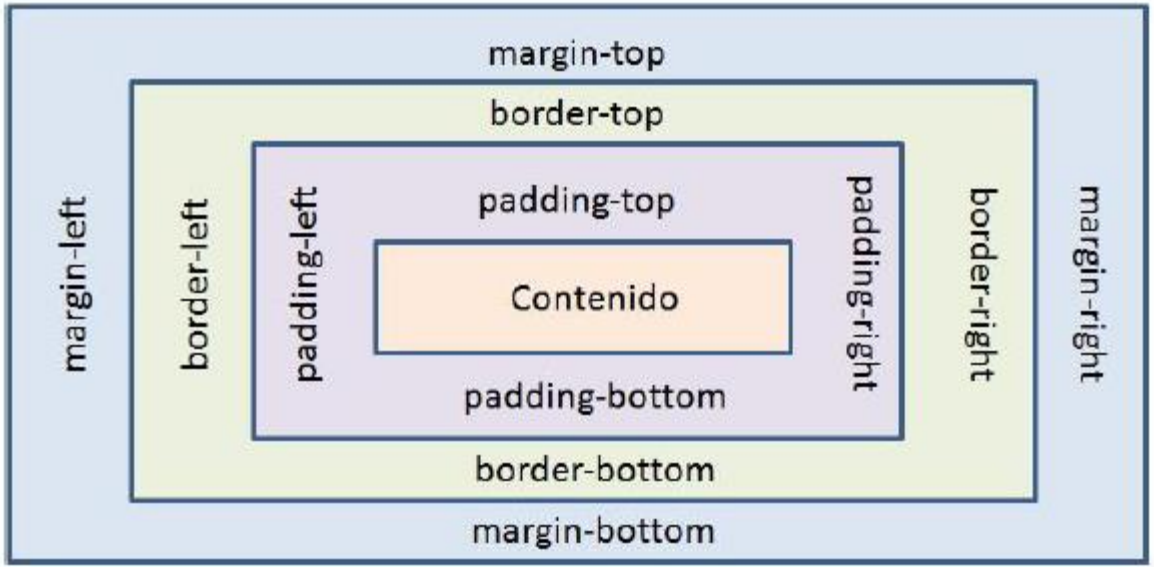
6.6 ATRIBUTOS DE LISTAS

- **list-style-type:** *disc* \ *circle* \ *square* \ *decimal* \ *decimal-leading-zero* \ *lower-roman* \ *upper-roman* \ *lower-greek* \ *lower-latin* \ *upper-latin* \ *armenian* \ *georgian* \ *lower-alpha* \ *upper-alpha* \ *none.* Estilo aplicable a los marcadores visuales de las listas. Se puede quitar el marcador con `list-style-type: none`.
- **list-style-image:** *url("../jpg")* \ *none.* Imagen aplicable a los elementos de las listas.
- **list-style-position:** *inside* \ *outside.* Posición dentro de la lista de los elementos marcadores de las listas.



7. ATRIBUTOS. MODELO DE CAJAS.

Las cajas tienen varios atributos relacionados con colores imágenes etc.



- Atributos de posición de la caja.
- Los atributos de los márgenes, que asignan un borde externo a la caja.
- Los atributos de relleno (padding) asignan un espacio interno dentro de la caja para separar el contenido de los márgenes.
- Los atributos de los bordes, que definen las líneas gráficas alrededor de la caja.

7.1 ATRIBUTOS *MARGIN*

- Los atributos margin-left, margin-right, margin-top, margin-bottom marcan la separación entre los diferentes elementos y se aplica por fuera de los bordes.

```
<head>
<style>
   div#contenedor {
        margin-top: 100px;
        margin-right: 100px;
        margin-bottom: 100px;
        margin-left: 50px;
        border: 3px dotted blue;
   }
   #contenedor div {
        margin: 15px;
        border: 3px dotted red;
   }
</style>
</head>
<body>
   <div id="contenedor">
        <div>Texto de 50 píxeles</div>
   </div>
</body>
```

7.1 ATRIBUTOS *MARGIN*

- En el ejemplo muestra los atributos definidos en dos cajas: una interna y otras externa. Se hace para las dos porque de esa manera se puede ver que las medidas se hacen relativas a la caja que la contiene. Para visualizar mejor el efecto se han incluido bordes, cuyo funcionamiento se verá a continuación.

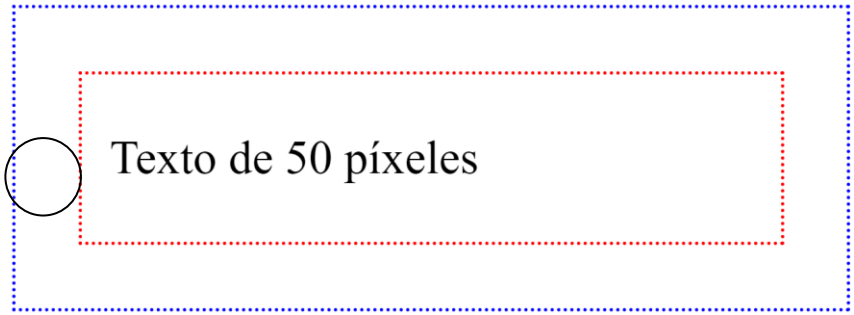
- Al no haber relleno (padding) la distancia entre el borde del cuadrado exterior (azul) y el mas interno(rojo) es de unos 15 px. La distancia de 100px es entre el borde exterior (azul) y los bordes del navegador, al estar justo debajo de la etiqueta <body>
- Para simplificar se puede usar un atributo margin que tiene 4 valores separados por espacios en blanco y cuyo orden es en el sentido de las agujas del reloj: margin-top, margin-right, margin-bottom y margin-left. Se puede resumir en uno, dos valores o tres valores teniendo las siguientes posibilidades.

```
margin: 100px 100px 100px 50px; /* Top:100, Right:100, Bottom:50, Left:50 */
margin: 100px 75px 50px; /* Top:100, Left y right:75, Bottom:50 */
margin: 100px 50px; /* Top y bottom: 00, Left y right: 0 */
margin: 100px; /* Las cuatro coordenadas 100 */
```

7.2 ATRIBUTOS *PADDING*

Los atributos son: *padding-top* (superior), *padding-right* (derecho), *padding-bottom* (inferior) y *padding-left* (izquierdo). Distancia entre el borde y los elementos que se encuentran en el interior. Se expresan con la misma sintaxis que margin.

```
<head>
<style>
    div#contenedor {
        margin: 100px 100px 100px 50px;
        padding: 20px;
        border: 3px dotted blue ;
    }
    #contenedor div {
        margin: 15px;
        padding: 20px 10px; /* 20px top y bottom, 10px left y right */
        border: 3px dotted red;
}
 </style>
 </head>
<body>
          <div id="contenedor">
               <div>Texto de 50 píxeles</div>
          </div>
</body>
```

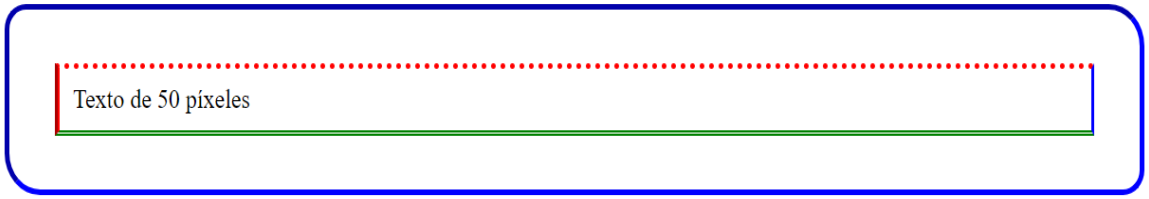


7.3 ATRIBUTOS *BORDER*

- **Los atributos son:**
  - **Border-width:** thick (ancho), medium(medio), thin(estrecho) o cualquier valor en las unidades de medida de CSS.
  - **Border-style:** Definen el estilo y color del borde de la caja. Palabras clave: *none*, *dotted (punteado)*, *dashed (discontinua)*, *solid (línea sólida)*, *doublé (doble linea)*, *groove (ranura)*, *ridge (cresta)*, *inset (recuadro)* y *outset.*
  - **Border-color:** Cualquier color válido en RGB, hexadecimal o con palabras reservadas.
  - Se pueden resumir estas propiedades por cada coordenada, es decir, `border-top: 2px solid blue` (border-width, border-style y border-color)
  - Incluso se pueden resumir las tres propiedades para todas las coordenadas con `border: 1px solid black` (las cuatro coordenadas con ancho de 1px, negro y sólido continuo)

7.3 ATRIBUTOS *BORDER*

- **Ejemplo:**




```
<head>
<style>
    div#contenedor {
          margin: 100px;
          border-style: inset;
          border-color: blue; /* Color del borde azul */
          border-radius: 15px; /* Borde redondeado de radio 15px */
          border-width: thick; /* Un borde grueso */
          padding: 15px;
    }
    #contenedor div {
          margin: 15px;
          padding: 10px;
          border-top : 3px dotted red ; /* Estilo, tamaño y color incluidos en el mismo
    atributo */
          border-right : 2px solid blue ;
          border-bottom : 3px double green ;
          border-left : 3px groove red ;
    }
 </style>
</head>
<body>
    <div id="contenedor">
          <div>Texto de 50 píxeles</div>
    </div>
</body>
```

7.4 OTROS ATTRIBUTOS DE CSS3

- Existen muchos más atributos en CSS3 como:
  - **border-radius**: Se usa para hacer esquinas redondeadas. Tienen un único parámetro que es el grado de curvatura. Al igual que el resto de propiedades del modelo de cajas se puede definir por cada coordenada o con una única regla, aunque la sintaxis cambia ligeramente:

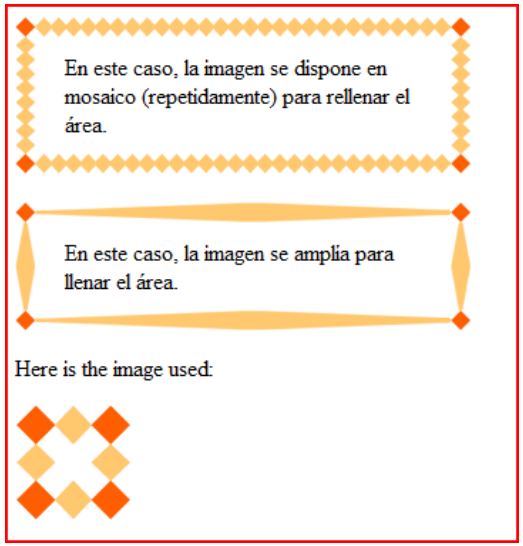
```
border-bottom-left-radius: 25px; /* Esquina inferior izquierda */
border-top-left-radius: 25px; /* Esquina superior izquierda */
border-bottom-right-radius: 25px; /* Esquina inferior derecha */
border-top-right-radius: 25px; /* Esquina superior derecha */

border-radius: 25px; /* Las cuatro esquinas */
border-radius: 25px 10px; /* top-left y bottom-right = 25px y el resto 10px*/
border-radius: 25px 10px 20px 5px /*  Orden: top-left, top-right corner, bottom-right bottom-left */
```

7.4 OTROS ATRIBUTOS DE CSS3

- Existen muchos más atributos en CSS3 como:
  - **border-image:** utiliza una imagen para definir el borde. Los valores numéricos indican el tamaño de la imagen origen que se tomará para formar el borde.

```
border-image: url(border.png) 30 30 round; /* round es
redondeada, en mosaico  y strech es estirada, la imagen se
amplía para llenar el área como se puede ver en la imagen. */
```



7.4 OTROS ATRIBUTOS DE CSS3

- **box-shadow**: que se usa para hacer dar sombra.
  - Los dos primeros parámetros son el lado derecho e inferior, respectivamente (y si los valores son negativos la sombra aparecería a la izquierda y arriba). El tercer parámetro es el nivel de opacidad y el último el color de la sombra.

```
<!DOCTYPE html>
<html>
<head>
    <style>
    div
    {
    width:300px;
    height:100px;
    background-color:yellow;
    box-shadow: 10px 10px 5px #888888;
    }
</style>
</head>
<body>

<div></div>

</body>
</html>
```

7.5 ATRIBUTOS DEL CONTENIDO Y DIMENSIONES

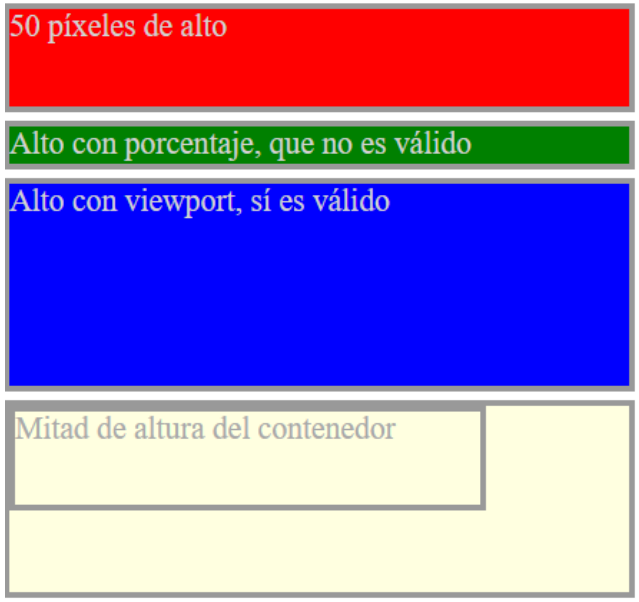
- También se puede concretar el ancho y alto de un contenido, con los atributos **width** y **height** respectivamente (con un valor con cualquier unidad de medida válida, ya sean px, % o em, entre otros).
- Además, las propiedades **max-width**, **max-height**, **min-width** y **min-height** permiten delimitar el tamaño máximo y mínimo cuando se emplear unidades relativas o que calcula el navegador. Por ejemplo:

https://www.w3schools.com/cssref/tryit.asp?filename=trycss_dim_min-width

https://www.w3schools.com/cssref/tryit.asp?filename=trycss_dim_max-height

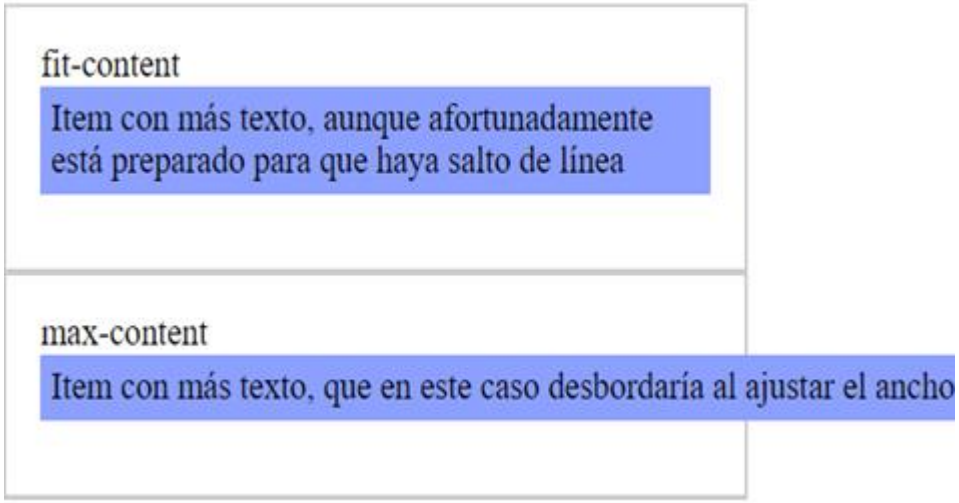
7.5 ATRIBUTOS DEL CONTENIDO Y DIMENSIONES

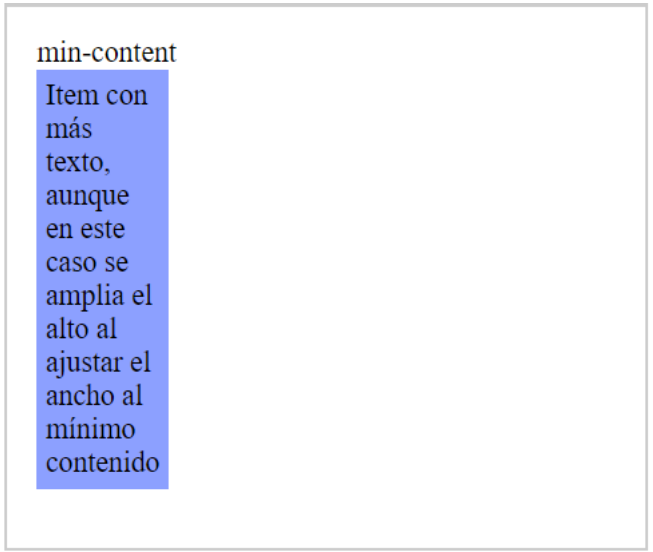
- Cabe destacar que **height** no funciona con porcentajes cuando se aplica en un elemento que no tiene dimensiones absolutas o relativas de manera estática, tales como px, em, rem o incluso unidades del viewport.
- Por ejemplo, al elemento verde de la imagen de debajo se le aplica el selector `#green {height: 50%}` pero no da resultado. Sin embargo, en el subcontenedor de debajo funciona el alto con porcentajes porque su contenedor tiene definido un alto de `6rem`



7.5 ATRIBUTOS DEL CONTENIDO Y DIMENSIONES

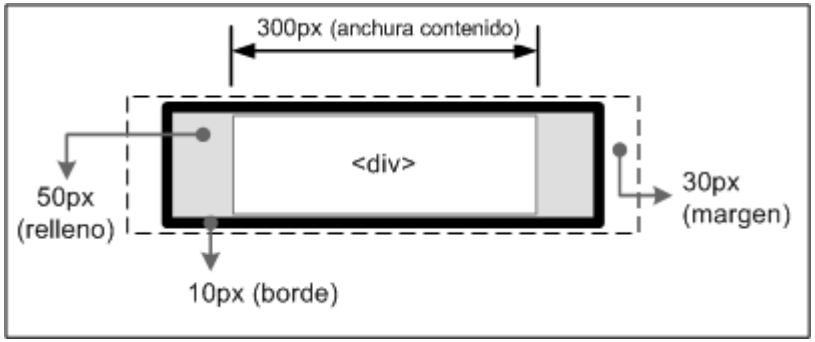
- En las últimas especificaciones de CSS se han incluido una serie de palabras reservadas para adaptar el alto o ancho al contenido. Algunas de las más empleadas son:
  - max-content: La altura del contenido siempre, aunque desborde.
  - min-content: La altura mínima posible.
  - fit-content: Se adapta al contenido sin desbordar el contenedor.





7.5 ATRIBUTOS DEL CONTENIDO Y DIMENSIONES

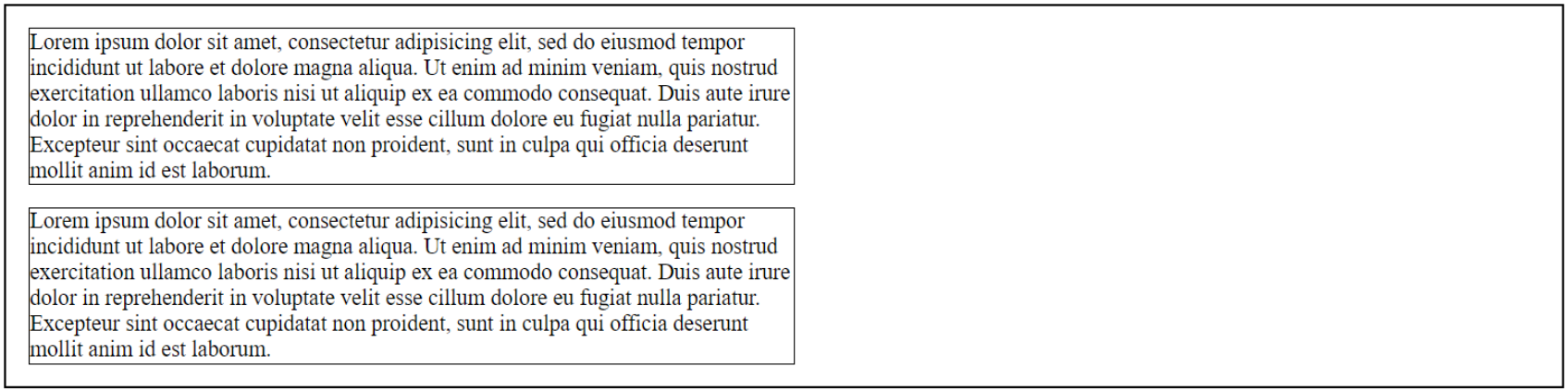
- Cuando se definen propiedades del modelo de cajas, ya sean margin, padding o borde, estas se suman al tamaño de un elemento. Por ejemplo, en la imagen de debajo, el contenido realmente ocupa 390px de ancho (300px + 50px+ 10px + 30px). Esto presenta una serie de problemas al diseñador.



- El atributo box-sizing con la propiedad border-box de soluciona este problema, ya que en el ancho (o alto) tiene en cuenta el padding y el borde (no el margin, ya que es una separación externa al elemento). Por tanto, el div de la imagen de arriba ocuparía 300px más los 30 px de margen. Ejemplo: https://www.w3schools.com/cssref/tryit.asp?filename=trycss3_box-sizing

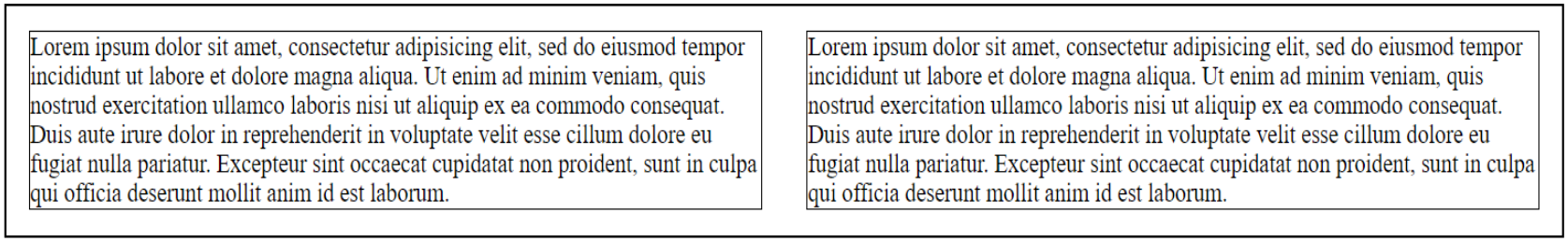
7.5 ATRIBUTOS DEL CONTENIDO Y DIMENSIONES

- En definitiva, combinar las dimensiones y el modelo de cajas es uno de los aspectos más complejos del diseño. Supongamos la siguiente página con un contenedor principal. Los dos cuadrados internos tienen 50% de ancho y margen de 1em para intentar situar ambos en la misma línea (con float que veremos más adelante).
- Entonces las dimensiones desbordan porque el margen no cuenta en las dimensiones y tenemos 50% + 2 em (uno a cada lado) + 50%. Un total de 100% + 2em que hacen que el segundo cuadrado aparezca debajo.



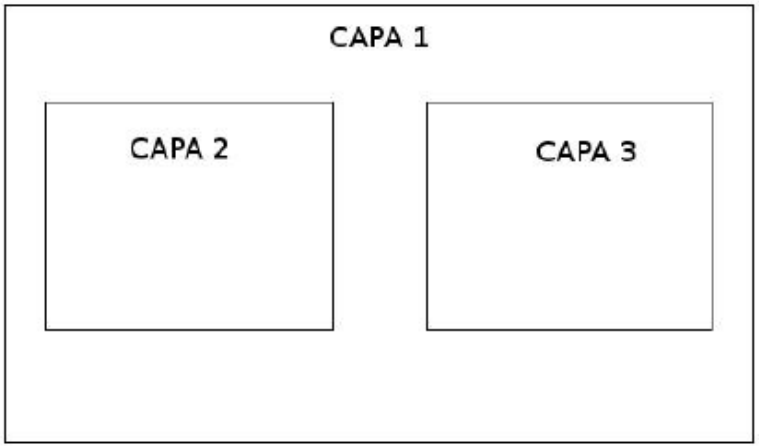
7.5 ATRIBUTOS DEL CONTENIDO Y DIMENSIONES

- Una propiedad interesante para asignar al ancho o al alto es el valor **calc.** Lo interesante es que se puede mezclar con porcentajes y em. Por lo que en el ejemplo anterior si tenemos un ancho con `width: calc(50% - 2em)` entonces lograremos el diseño de debajo sin desbordar y empleando márgenes. En los ejemplos adjuntos a la presentación puedes observar con más detalle esta técnica de diseño.
- La idea de la función **calc** es poder asignar dimensiones más complejas y evitar desbordamientos cuando no se emplean px y no tenemos certeza del espacio real que van a ocupar los elementos.



8. MAQUETACIÓN CON CAPAS

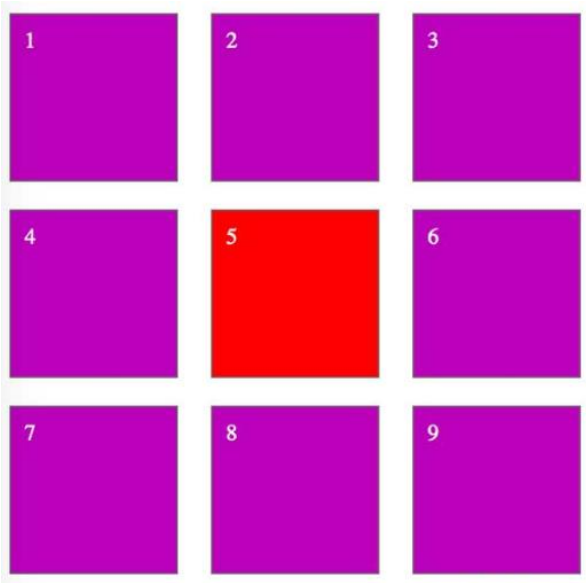
- Las capas, layouts o divs son el mismo concepto, pero con distinto nombre. Se suelen imaginar como contenedores donde podemos incluir lo que queramos dentro (imágenes, texto, animaciones, otro bloque, o todo al mismo tiempo) a los que se le asigna un ancho, alto y posición, de esta manera se van a ir posicionando consiguiendo la estructura que queremos. Por ejemplo, en la siguiente imagen:



- Capa 1: Es la capa principal y contenedora
- Capa 2: Capa dentro de la capa contenedora 1 y alineada a la izquierda (float: left;)
- Capa 3: Igual que la capa 2, pero a la derecha y con un margen (float: right; margin: 10px).

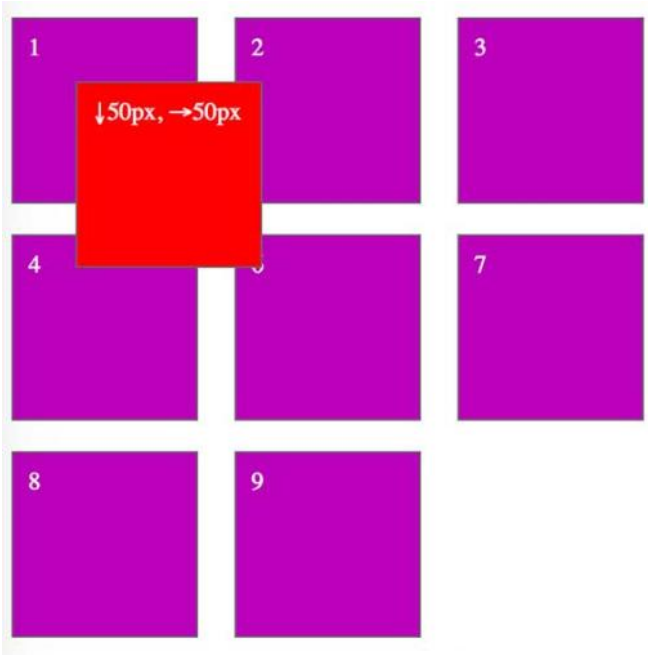
8.1. ATRIBUTOS DE POSICIÓN

- A continuación veremos una serie de atributos normalmente asociados a la maquetación por capas, empezando por el posicionamiento,:
  - **position: static** es el valor predeterminado del atributo y el posicionamiento normal de los elementos en la página. Quiere decir que los elementos se colocarán según el flujo normal del HTML.



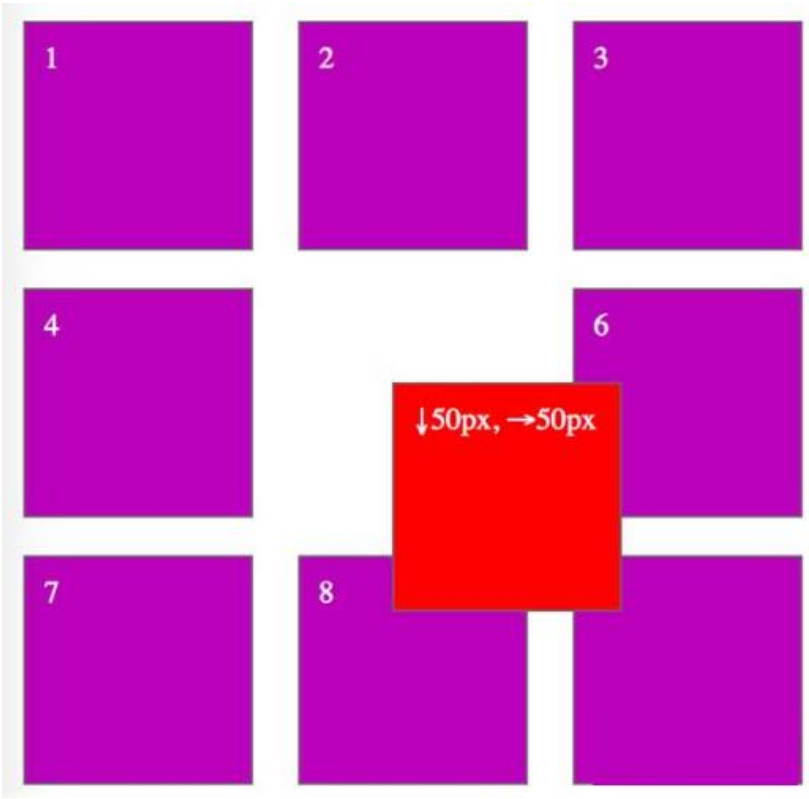
8.1. ATRIBUTOS DE POSICIÓN

- **position: absolute** indica la distancia del borde superior de la capa con respecto al borde superior de la última capa posicionada. Es decir, si hay un por encima con posicionamiento se tienen en cuenta este último, si no el borde del navegador. En la imagen los 9 recuadros podrían estar rodeados por un con posicionamiento y se tendría en cuenta este elemento. Si no, las coordenadas del navegador.



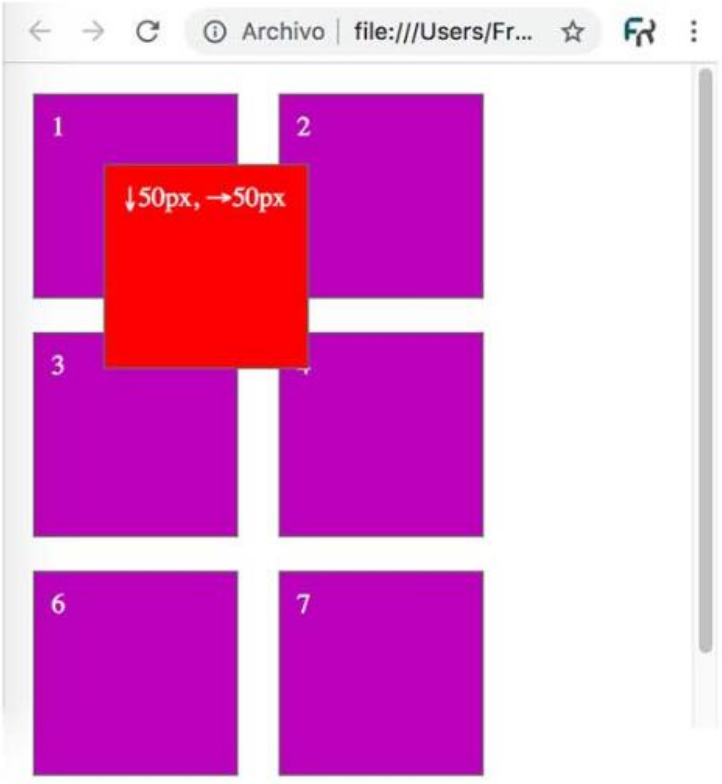
8.1. ATRIBUTOS DE POSICIÓN

- **position: relative** indica la distancia desde donde correspondía el posicionamiento en ese momento.



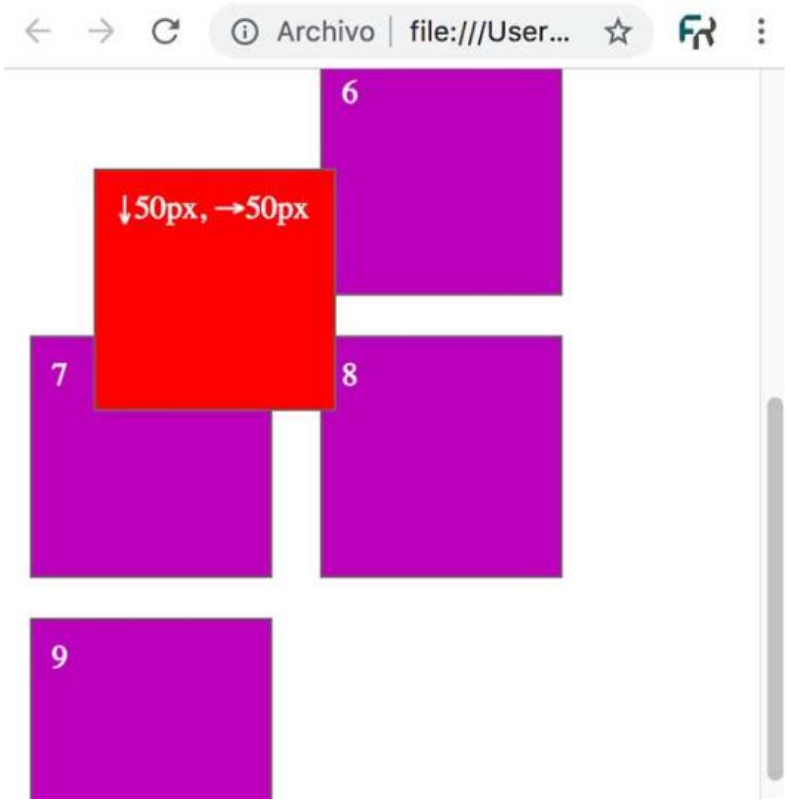
8.1. ATRIBUTOS DE POSICIÓN

- **position: fixed** La posición final de la capa será siempre fija, es decir, aunque se desplace el documento con las barras de desplazamiento del navegador, siempre aparecerá en la misma posición.



8.1. ATRIBUTOS DE POSICIÓN

- **position: sticky:** En principio se comporta como posicionamiento relativo, pero si se hace scroll se queda como posicionamiento fijo con los valores definidos.



8.1. ATRIBUTOS DE POSICIÓN

- Teniendo en cuenta los diferentes valores de posicionamiento, para que realmente tengan efecto hay que combinarlos con desplazamientos según las coordenadas que se indican a continuación:
  - top
  - bottom
  - left
  - right
- Por ejemplo:

```
a.caja {
   position:absolute;
   top:50px;
   left:50px;
   background-color:red;
} /* Se desplaza 50px desde la izquierda y 50px desde arriba
de donde corresponde el posicionamiento */
```

8.2 SUPERPOSICIÓN DE CAJAS

- Controlar los aspectos de superposición y precedencia de estilos supone entender bien el funcionamiento del CSS y conseguir resultados muy precisos y personales.

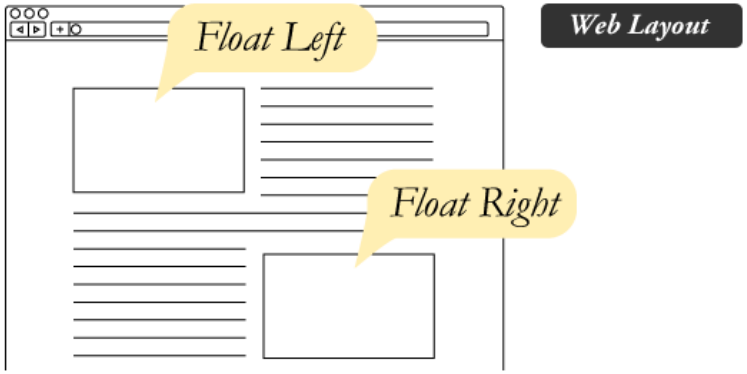
- El atributo `z-index` permite definir el nivel de profundidad de una caja. Su valor es un número entero. En principio el estándar W3C permite números negativos, pero generalmente el valor 0 suele tomarse como en nivel más bajo. Cuanto más alto sea el valor, más cerca se mostrará la capa al usuario en la web, es decir, una caja con `z-index=10` se mostrará por encima de una con `z-index=9`).

- **El atributo z-index solo tiene efecto si va acompañado del atributo position con un valor diferente a static.**

8.3. FLOTAR Y POSICIONAR

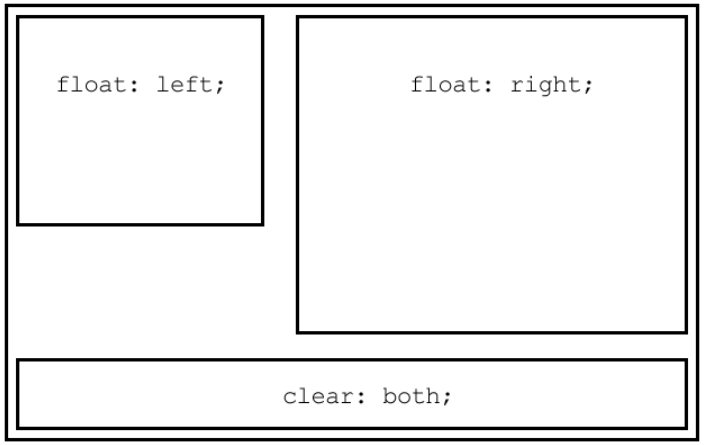
Flotar sirve para mover una caja a la izquierda o a la derecha hasta que su borde exterior toque el borde de la caja que lo contiene o toque otra caja flotante. Para que un elemento pueda flotar debe tener definido implícita o explícitamente su tamaño. Las cajas flotantes no se encuentran en el "flujo normal" del documento por lo que las cajas que sí siguen el flujo normal se comportan como si las flotantes no estuviesen ahí.

- **Float:** La propiedad float puede tener los siguientes valores:
  - *none* hará que el objeto no sea flotante (por defecto)
  - *left* hace que el elemento flote a la izquierda.
  - *right* hace que el elemento flote a la derecha.



8.3. FLOTAR Y POSICIONAR

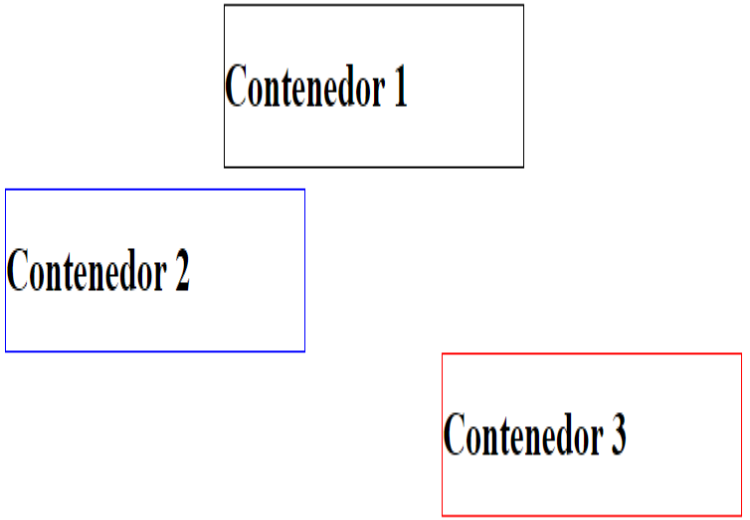
- **Clear:** Sirve para mantener limpia el área que está al lado del elemento flotante y que el siguiente elemento comience en su posición normal dentro del bloque que lo contiene. La propiedad clear puede tener los siguientes valores:
  - left indica que el elemento comienza por debajo de cualquier otro elemento del bloque al que pertenece que estuviese flotando a la izquierda.
  - right funciona como el left pero en este caso el elemento deberá estar flotando a la derecha.
  - both mueve hacia abajo el elemento hasta que esté limpio de elementos flotantes a ambos lados.
  - none permite elementos flotantes a ambos lados. Es el valor por defecto.



8.4. CENTRAR ELEMENTOS

- Hasta ahora se han estudiado propiedades que tienen que ver con la alineación de texto o de celdas en tablas (text-align o vertical-align). Si queremos alinear contenedores, el procedimiento es algo diferente.
- Por un lado, la **alineación horizontal** es muy sencilla. Se trata de agrupar todos los contenidos de la página en un elemento y asignarle a ese (u otra etiqueta en bloque equivalente unos márgenes laterales automáticos con el valor auto en las coordenadas left y right.
- De la misma forma, con el valor margin-left: auto un elemento se alinea a la derecha (deja el resto de espacio al margen izquierdo) y con el valor margin-right: auto a la izquierda (dejando el resto del espacio al margen derecho).

```
#contenedor1 {
   margin: 0 auto; /* centrado */
}
#contenedor2 {
   margin-right: auto;  /* izquierda */
}
#contenedor3 {
   margin-left: auto; /* derecha */
}
```



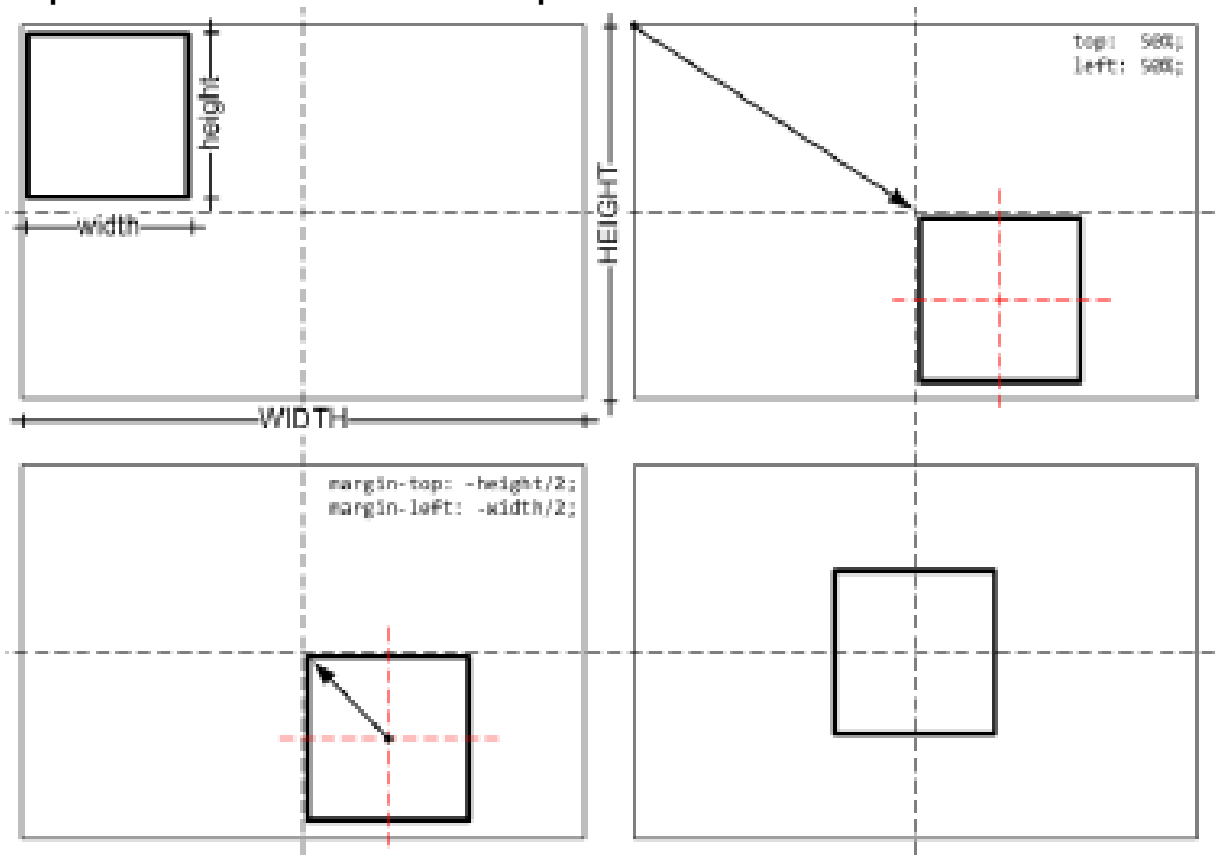
8.4. CENTRAR ELEMENTOS

- Aunque centrar una página web horizontalmente es muy sencillo, centrarla verticalmente es mucho más complicado. Afortunadamente, no es muy común que una página web aparezca centrada de forma vertical. El motivo es que la mayoría de páginas web son más altas que la ventana del navegador, por lo que no es posible centrarlas verticalmente de manera sencilla.
- Desafortunadamente, la propiedad margin: auto no funciona tal y como se espera para los márgenes verticales y la página no se muestra centrada.
- La solución correcta para **centrar verticalmente** una página web se basa en el posicionamiento absoluto e implica realizar un cálculo matemático sencillo. A continuación se muestra el esquema gráfico de los cuatro pasos necesarios para centrar una página web en la ventana del navegador:

```
#contenedor {
    width: 500px;
    height: 500px;
    position: absolute;
    top: 50%;
    left: 50%;
    margin-top: -250px; /* height/2 = 500px / 2 */
    margin-left: -250px; /* width/2 = 500px / 2 */
}
```

8.4. CENTRAR ELEMENTOS

- Como se desprende de la imagen anterior, la página web debe moverse hacia arriba una cantidad igual a la mitad de su altura y debe desplazarse hacia la izquierda una cantidad equivalente a la mitad de su anchura



8.4. CENTRAR ELEMENTOS

- Y estos cálculos se pueden hacer automáticamente empleando la propiedad transform: translate(-50%, -50%) que desplaza un elemento la mitad de su tamaño hacia la izquierda y hacia arriba.

```
#contenedor {
    width: 500px;
    height: 500px;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%).
}
```

9. PRECEDENCIA DE ESTILOS

- La precedencia de estilos va asociado con el concepto de *especificidad* de una regla.
- La especificidad se refiere al peso que toman cada uno de los elementos de una hoja de estilo. Cuanto más peso más especificidad. Cuanta más especificidad tenga un regla menos problemas a la hora de garantizar que será esa y no otra la regla que se aplique sobre un determinado contenido.
- Un cálculo sencillo para calcular la especificidad de una regla es sumar los puntos según el tipo de selectores que contenga:
- Se da un valor de 1 punto a un *selector de etiqueta* (por ejemplo, h1, p, div).
- A un *selector de clase* se le da el valor de 10 puntos.
- A un *selector de identificador* se le da un valor de 100 puntos.
- A un *atributo de estilo* (style) a los que se les da un valor de 1000 puntos.

9. PRECEDENCIA DE ESTILOS

- Ejemplo:

```
<style>
   p{background: crimson;} /* Especificidad de 1 puntos */
   .parrafo{background: pink;} /*Especificidad de 10
      puntos*/
   p.parrafo{background: brown;} /*Especificidad de 11
      puntos*/
   #id-parrafo{background: orange;}/*Especificidad de 100
      puntos*/
   p#id-parrafo{background: red;}/*Especificidad de 101
      puntos*/
   p.parrafo#id-parrafo{background:green;}/*Especificidad
      de 111 puntos*/
</style>
```

9. PRECEDENCIA DE ESTILOS

- Otro enfoque a tener en cuenta:
  - Se cuenta 1 si la declaración está incluida en un atributo style; si no, se cuenta 0. (Peso "a", mayor importancia).
  - Se cuenta el número de identificadores (ID) que aparezcan en el selector. (Peso "b").
  - Se cuenta el número de atributos (distintos al ID) , clases y pseudo-clases en el selector. (Peso "c").
  - Se cuenta el número de nombres de elementos y pseudo-elementos en el selector. (Peso "d", menor importancia).

9. PRECEDENCIA DE ESTILOS

- Otro enfoque a tener en cuenta:

```
*               {}  /* a=0 b=0 c=0 d=0 -> especificidad = 0,0,0,0 */
 li             {}  /* a=0 b=0 c=0 d=1 -> especificidad = 0,0,0,1 */
 li:first-line  {}  /* a=0 b=0 c=0 d=2 -> especificidad = 0,0,0,2 */
 ul li          {}  /* a=0 b=0 c=0 d=2 -> especificidad = 0,0,0,2 */
 ul ol+li       {}  /* a=0 b=0 c=0 d=3 -> especificidad = 0,0,0,3 */
 h1 + *[rel=up]{}   /* a=0 b=0 c=1 d=1 -> especificidad = 0,0,1,1 */
 ul ol li.red   {}  /* a=0 b=0 c=1 d=3 -> especificidad = 0,0,1,3 */
 li.red.level   {}  /* a=0 b=0 c=2 d=1 -> especificidad = 0,0,2,1 */
 #x34y          {}  /* a=0 b=1 c=0 d=0 -> especificidad = 0,1,0,0 */
 style=""           /* a=1 b=0 c=0 d=0 -> especificidad = 1,0,0,0 .
                       Éste último es el que se aplica primero*/
```

9. PRECEDENCIA DE ESTILOS

- Por último, estas reglas tienen una excepción que viene dada por el identificador `!important`
- Por ejemplo: `p{color: blue !important}`
- Esta última regla tendría prioridad sobre otras más específicas.
- En caso de que haya varias reglas con `!important`, se tienen en cuenta los mismos parámetros y se opta por la más específica.

10. COMPATIBILIDAD CSS NAVEGADORES

- A continuación se incluye un enlace con la compatibilidad de las diferentes propiedades de CSS en los navegadores más empleados:
  - https://www.w3schools.com/cssref/css3_browsersupport.asp
- Mientras que la página siguiente contiene un buscador por propiedad que permite verificar su soporte en una amplia gama de navegadores:
  - https://caniuse.com/

11. DIRECCIONES DE INTERÉS

- http://www.w3.org/
- http://www.w3schools.com/
- http://jigsaw.w3.org/css-validator/
- http://xhtml-css.com/
- https://desarrolloweb.com/css/
- https://uniwebsidad.com/libros/css