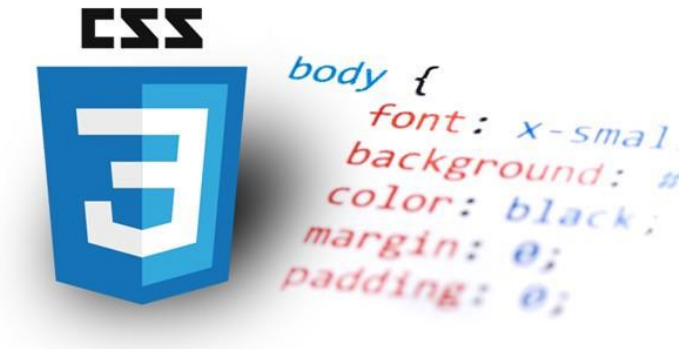
Diseño de Interfaces Web
Unidad 2 (I)

Introducción a CSS3



UNIDAD 2 (I)
Introducción a CSS3

1. Introducción a CSS

- Origen de CSS
  - En HTML se pueden usan contenidos, estructuras e instrucciones de formato en un único documento
  - A principios de los años 90, HTML no poseía todas las capacidades actuales
  - Había pocos posibilidades para controlar el aspecto de los documentos y el archivo HTML resultaba complejo al estar mezclados contenidos, estructuras e instrucciones de formato
  - Para solucionarlo, el W3C creó el estándar **CSS** en el que las instrucciones de formato se separan del resto de elementos
  - A partir de **HTML 4** están desaconsejados los elementos HTML de formato y se recomienda usar solo **CSS** para ello

1. Introducción a CSS

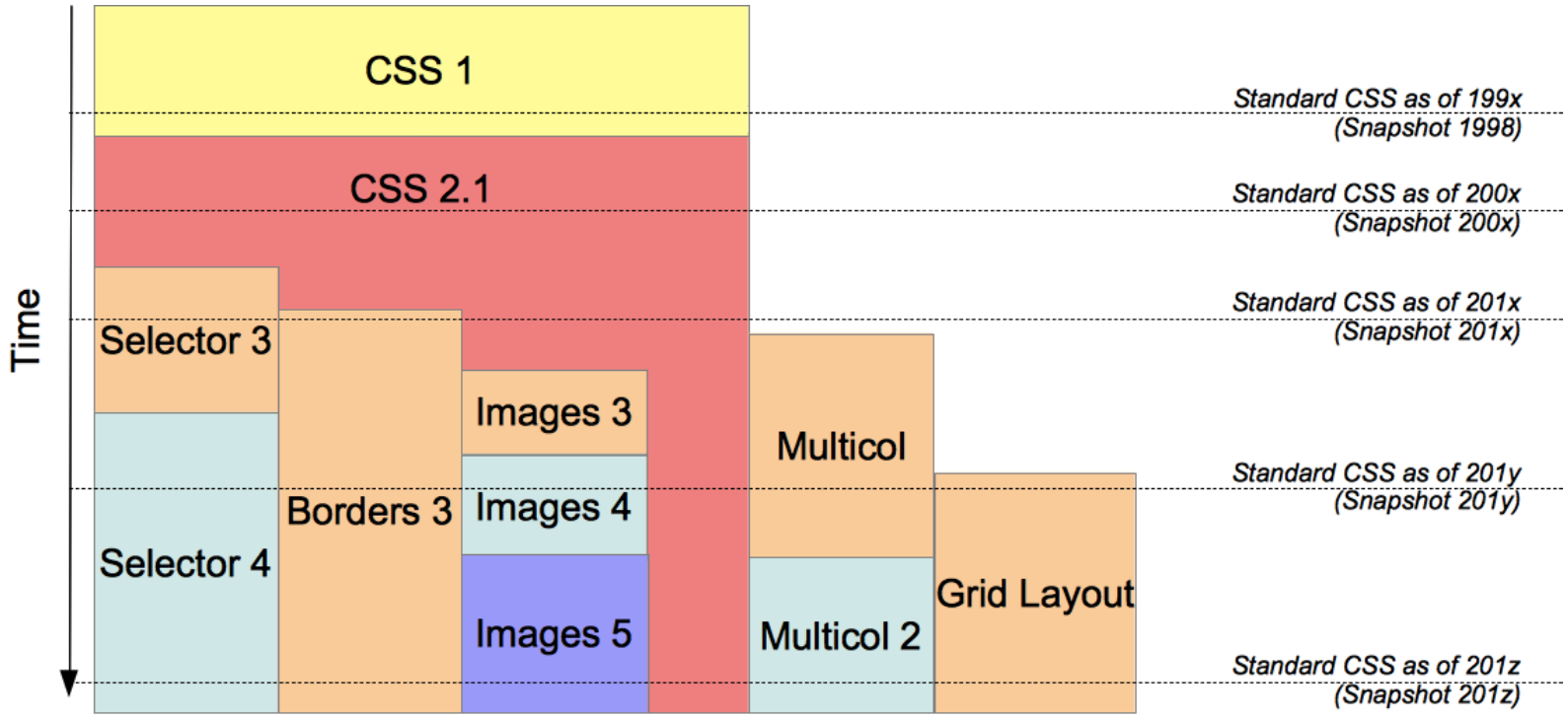
- Ventajas de CSS
  - Mediante una instrucción se puede aplicar un determinado formato a todo un sitio web en vez de a un solo elemento
  - Si tenemos que mantener un sitio entero y decidimos cambiar algún aspecto de su apariencia, basta con cambiar unas pocas líneas de código para modificar el estilo de numerosos elementos (sitios homogéneos)
  - CSS ha ido evolucionando y actualmente ofrece muchas más posibilidades que las etiquetas HTML: permite cambiar tamaño, grosor, inclinación, altura de línea, colores de fondo y primer plano, posicionamiento, gradientes, transiciones, maquetación, diseño responsive, etc.

1. Introducción a CSS

- Niveles de CSS
  - Cada nivel de CSS se construye sobre el anterior, añadiendo funciones al nivel previo
  - Los navegadores implementan estas especificaciones de forma distinta
    - CSS 1(1996): propiedades de fuente, colores alineación,…
    - CSS 2 (1998): posicionamiento, tipos de medios,...
    - CSS 2.1 (2005): modifica propiedades y corrige errores
    - CSS 3 (2011): va incluyendo módulos separados con nuevas funciones

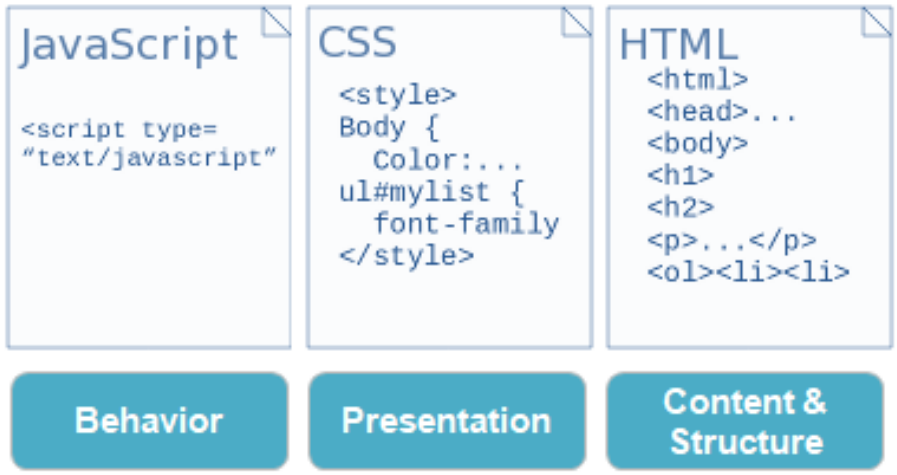
1. Introducción a CSS

- Niveles de CSS



2. Hojas de estilo

- CSS (Cascading Style Sheet)
  - Es un lenguaje que sirve para dotar de estilo a los elementos que componen una página o sitio web
  - Aunque existen diversas formas de aplicar este lenguaje, la forma habitual de hacerlo es mediante una **hoja de estilo**



2. Hojas de estilo

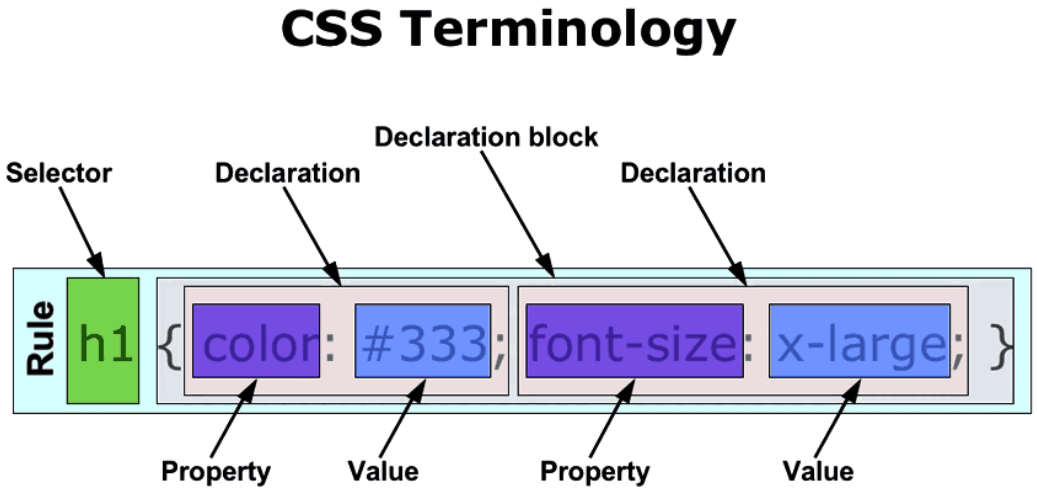
- Hojas de estilo
  - Una hoja de estilo es un conjunto de instrucciones que están vinculadas a una página HTML para darle formato a todos o algunos de los elementos que la componen
  - El código que compone la hoja de estilo está formado por una o más **reglas de estilo**

CSS Example

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

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

p {
    font-family: verdana;
    font-size: 20px;
}
```

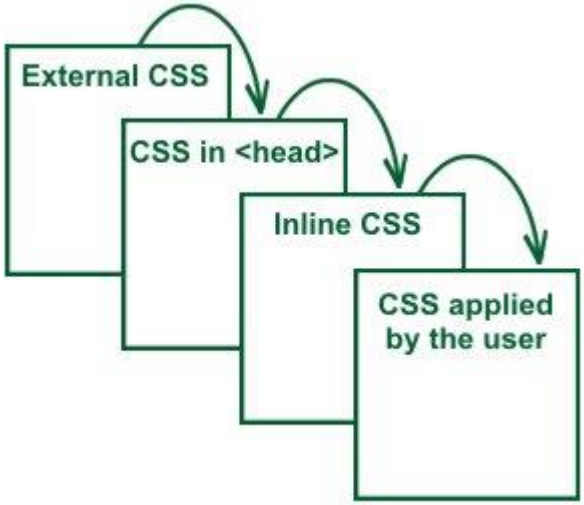
2. Hojas de estilo

- Reglas de estilo
  - Son las declaraciones de los formatos que adoptarán los elementos de la página o sitio web a la que está destinada la hoja de estilo
  - Mediante la regla se identifica el elemento HTML que se desea seleccionar y la apariencia que se le quiere dar



2. Hojas de estilo

- Formas de aplicar estilos
  - De forma local
  - Hojas de estilo internas
  - Hojas de estilo externas



2. Hojas de estilo

- Aplicar estilos de forma local
  - Se aplica el estilo a un elemento HTML directamente
  - Se escribe el código CSS dentro del atributo `style` de la etiqueta HTML
  - El único elemento afectado es aquel en el que está ubicado el estilo
  - Por ejemplo:

```
<p style="color:red;">Internet</p>
```

2. Hojas de estilo

- Hojas de estilo internas
  - Se usan cuando se quiere aplicar el estilo sólo a la página donde se ubica
  - El código de la hoja de estilo interna se encontrará entre las etiquetas:

```
<style type="text/css"> … </style>
```

  - Esta etiqueta se puede colocar en cualquier parte de la página (suele hacerse dentro de la cabecera)
  - En una página se pueden definir tantas hojas de estilo como sean necesarias

2. Hojas de estilo

- Hojas de estilo externas
  - Se usan para dar un aspecto común a varias páginas de un sitio web
  - Todos los estilos se definen en una hoja de estilo externa
  - Todas las páginas consultarán la misma hoja, obteniéndose un estilo común para todo el sitio
  - Se crean escribiendo el código en un documento de texto plano y se guarda con la extensión **.css**
  - Posteriormente se vincula a la página mediante las siguientes etiquetas colocadas en la cabecera:

```
<link rel="stylesheet" type="text/css" href="url.css">
```

2. Hojas de estilo

- Hojas de estilo externas
  - Ejemplo

```
<html>
<head>
   <title>Ejemplo hoja externa</title>
   <link rel="stylesheet" type="text/css" href="estilo.css">
</head>
<body>
   Contenido de la página
</body>
</html>
```

2. Hojas de estilo

- Hojas de estilo externas
  - Otra forma de aplicar estilos es utilizando la función **@import** en un elemento *style* dentro de la página HTML (similar al uso de librerías en lenguajes de programación)

```
<style type="text/css">
<!--
 @import url(css/estilos.css)
 @import url(http://www.otraweb.com/css/estilos.css)
-->
</style>
```

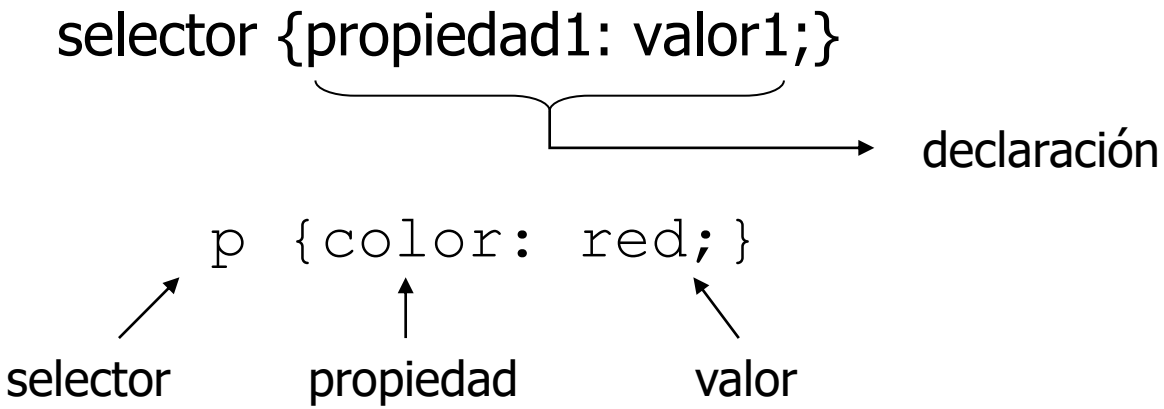
2. Hojas de estilo

- Hojas de estilo externas
  - **@import** facilita la organización de estilos en diferentes ficheros en proyectos más grandes
  - Por ejemplo, dentro de un fichero .css se pueden importar hojas externas organizadas por estructuras u otros criterios

```
@import url(“colors.css”)
@import url(“header.css”)
@import url(“navbar.css”)
@import url(“main.css”)
```

3. Reglas de estilo
3.1 Partes de una regla de estilo

- Reglas de estilo
  - Una hoja de estilo se compone de una o varias **reglas de estilo**
  - Las reglas de estilo tienen dos componentes
    - **Selector** ➔ selecciona los elementos sobre los que actuará la regla
    - **Declaración** ➔ establece las propiedades y valores a aplicar sobre los elementos seleccionados



3. Reglas de estilo
3.2 Selectores

- Selectores simples
  - Para aplicar una propiedad (o varias) a un elemento HTML se indica el nombre de la etiqueta en el selector

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

  - Mediante **“,”** se agrupan selectores dentro de la misma regla

```
td, p {color: green;}
```

  - Si se utiliza el carácter asterisco “*****” (selector universal), las propiedades afectarán a todos los elementos de la página

```
* {color: blue;}
```

3. Reglas de estilo
3.2 Selectores

- Selectores avanzados
  - Si se quiere aplicar una propiedad a los elementos que están dentro de otros debemos separarlos con un **espacio**

```
li li {color: blue;}
```

  - Para aplicar una propiedad a los hijos directos (árbol DOM) de un elemento se utiliza "**>**"

```
p > img {border: 3 px solid black;}
```

3. Reglas de estilo
3.2 Selectores

- Selectores avanzados
  - Si se quiere aplicar una propiedad a los elementos que están justo **después** de otro se utiliza el carácter "**+**"

```
p + h1 {background-color: pink;}
```

  - En el caso de que queramos aplicar la propiedad a los elementos que están precedidos de otro se utiliza "**~**" (no tienen que estar **inmediatamente precedidos**, pero sí deben de tener el mismo padre)

```
p ~ h1 {background-color: aqua;}
```

3. Reglas de estilo
3.2 Selectores

- Selectores por atributos
  - También se pueden aplicar estilos a etiquetas que tengan determinados **atributos**

```
img[alt] {border: 5 px solid black;}
```

Pseudo-clase	Significado	Ejemplo
Etiqueta[atributo^=valor]	Selecciona aquellas etiquetas cuyo atributo comience por ese valor	a[href^=https:]
Etiqueta[atributo*=valor]	Selecciona aquellas etiquetas cuyo atributo contenga ese valor (en cualquier lugar)	img[alt*=logo]
Etiqueta[atributo$=valor]	Selecciona aquellas etiquetas cuyo atributo acabe en ese valor	img[src$=.png]
Etiqueta[atributo~=valor]	Selecciona aquellas etiquetas cuyo atributo incluya la palabra “valor”	a[class~=“menu”]

3. Reglas de estilo
3.2 Selectores

- Selectores específicos
  - Puede ocurrir que sólo queramos aplicar un estilo determinado a unos elementos concretos y no a todos con la misma etiqueta
  - Para ello debemos usar selectores más específicos como:
    - Identificadores
    - Clases
    - Pseudo clases (pseudo-selectores)
    - Pseudo elementos

3. Reglas de estilo
3.2 Selectores

- Identificadores
  - Con el atributo ‘id’ podemos asignar una identificación única al elemento
  - CSS podrá seleccionarlo y aplicar un estilo de forma específica a ese elemento concreto
  - Por ejemplo:

```
<p id="despedida">
```

  - El valor que le demos a ‘id’ debe comenzar por una letra seguida por un número cualquiera de caracteres alfanuméricos

3. Reglas de estilo
3.2 Selectores

- Identificadores
  - En el selector CSS se emplea el símbolo # delante del nombre del identificador
  - Continuando con el ejemplo anterior:

```
p#despedida {font-size: 14px;}
```

    - ✓ Indicaría que la fuente de los elementos ‘p’ identificados como *despedida* debe tener un tamaño de 14px
    - ✓ El resto de párrafos tendrá el tamaño especificado en el documento HTML

3. Reglas de estilo
3.2 Selectores

- Clases
  - Mediante el atributo “class” podemos agrupar los elementos por clases o grupos para que CSS pueda seleccionarlos y distinguirlos de los demás
  - Por ejemplo:

```
<img src=“casa.jpg” class=“fotos”>
```

  - Un elemento puede pertenecer a varias clases a la vez (no son excluyentes)

```
<img src=“chalet.jpg” class=“fotos casas”>
```

3. Reglas de estilo
3.2 Selectores

- Clases
  - Si un elemento HTML ha sido identificado mediante una clase, es posible formatearlo con una regla de estilo
  - En el selector CSS se escribe el nombre de la clase precedido de un punto y del elemento afectado por la regla
  - Por ejemplo:

```
img.fotos {border-width: 1px;}
img.casas {border-width: 3px;}
```

  - Si al indicar la clase se omite el nombre del elemento, se verán afectados todos los que pertenezcan a dicha clase

```
.fotos {border-color:yellow;}
```

3. Reglas de estilo
3.2 Selectores

- Resumen de selectores

Selector	Significado
`*`	Selector universal (selecciona todas las etiquetas)
`#id`	Selecciona el elemento con ese valor en el atributo id
`.class`	Selecciona los elementos que pertenezcan a la clase
`etiqueta`	Selecciona esa etiqueta
`selector1, selector2`	Se seleccionan todos los elementos indicados en selector1 y selector2
`selector 1 selector 2`	Se seleccionan todos los elementos indicados en selector2 que estén **dentro** de aquellos seleccionados por selector1

3. Reglas de estilo
3.2 Selectores

- Resumen de selectores

Selector	Significado
`selector1>selector2`	Se seleccionan todos los elementos indicados en selector2 que son **hijos directos** (en el árbol DOM) de los elementos seleccionados por selector1
`selector1+selector2`	Se seleccionan todos los elementos indicados en selector2 que están justo **después** de aquellos seleccionados con selector1
`selector1~selector2`	Se seleccionan todos los elementos indicados en selector2 que están precedidos por los elementos indicados por selector1 Ambos elementos deben de tener el mismo padre, pero los elementos indicados por selector2 no tienen que estar **inmediatamente precedidos** de los elementos indicados por selector1.
`[atribut expre valor]`	siendo expre (=,~=,|=,$=,*=..) para seleccionar elementos atendiendo al valor de sus atributos

3. Reglas de estilo
3.2 Selectores

- Pseudo-clases (pseudo-selectores)
  - Son palabras clave que se añaden a los selectores y sirven para clasificar a los elementos según el estado en el que se encuentran

```
selector:pseudoselector {
        propiedad: valor;}
```

  - Por ejemplo, la siguiente regla indica que los vínculos, una vez visitados, aparezcan en azul:

```
a:visited {color: blue;}
```

3. Reglas de estilo
3.2 Selectores

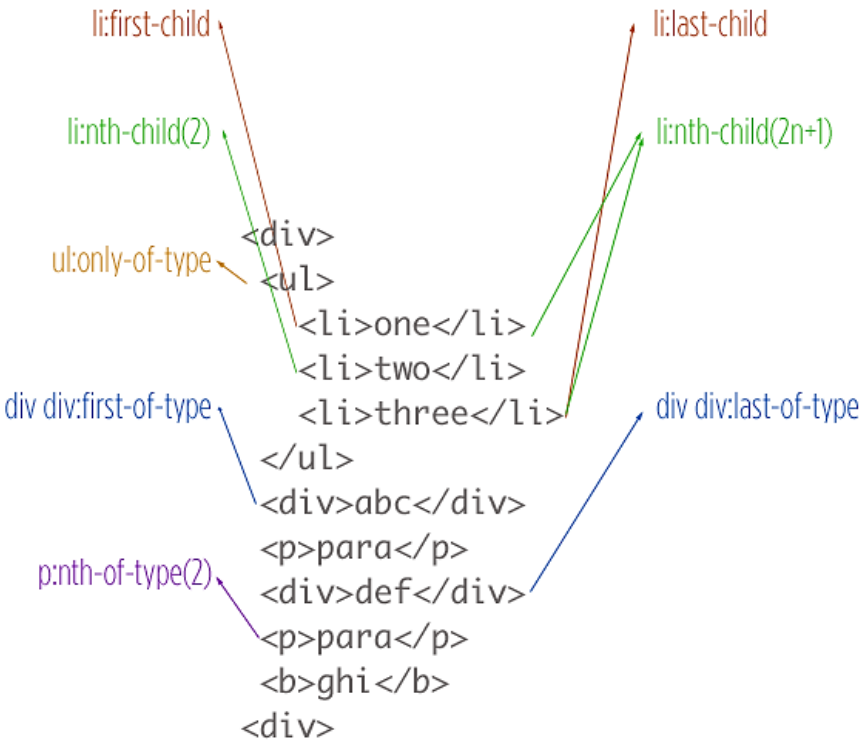
- Pseudo-clases (pseudo-selectores)
  - De estado
    - :link
    - :visited
    - :enabled
    - :disabled
    - :checked
    - :required
    - :optional
    - :focus
    - :hover
    - :empty
    - …

3. Reglas de estilo
3.2 Selectores

- Pseudo-clases (pseudo-selectores)
  - De posición
    - :first
    - :first-child
    - :first-of-type
    - :last-child
    - :only-child
    - :last-of-type
    - :nth-child(n) / :nth-child(odd) / :nth-child(even)
    - :nth-last-child()
    - :nth-last-of-type()
    - :not(selector)
    - …

3. Reglas de estilo
3.2 Selectores

- Pseudo-clases (pseudo-selectores)
  - De posición



3. Reglas de estilo
3.2 Selectores

- Pseudo-elementos
  - Son palabras clave que se añaden a los selectores que sirven para seleccionar **partes** de elementos HTML

```
selector::pseudoelemento {
        propiedad: valor;}
```

  - Mediante reglas de estilo podemos identificar estos elementos y formatearlos de manera distinta a la de los elementos a los que pertenecen

```
p::first-letter {color: brown;}
```

3. Reglas de estilo
3.2 Selectores

- Pseudo-elementos
  - Tipos
    - ::first-line
    - ::first-letter
    - ::selected
    - ::after (asociada a la propiedad *content*)
    - ::before (asociada a la propiedad *content*)
    - …

3. Reglas de estilo
3.3 Precedencia

- Principio de la cascada
  - Si en un sitio web se han definido varias reglas de estilo para un mismo elemento, ¿cuál se aplica?
  - CSS sigue el "**principio de la cascada**"
  - Sirve para decidir cómo resolver conflictos generados por problemas de **herencia**, **especificidad** o **ubicación**

3. Reglas de estilo
3.3 Precedencia

- Principio de la cascada
  - **Herencia**: hay propiedades de CSS que afectan a los elementos definidos por el selector y también a sus descendientes
    - Por ejemplo, si se formatean los elementos h1 en *azul* con un borde *rojo*, cualquier elemento dentro de h1 será también *azul* (la propiedad *color* se hereda) pero no tendrán el borde *rojo* (*border* no se transmite)

Valor inicial	Varies from one browser to another
Applies to	all elements. It also applies to `::first-letter` and `::first-line`.
Heredable	yes
Valor calculado	If the value is translucent, the computed value will be the `rgba()` corresponding one. If it isn't, it will be the `rgb()` corresponding one. The `transparent` keyword maps to `rgba(0,0,0,0)`.

- `valor inicial`: ver propiedades individuales
- Se aplica a: todos los elementos
- `herencia`: no
- Porcentajes: N/A
- Medio: `visual`
- `valor calculada`: ver propiedades individuales

3. Reglas de estilo
3.3 Precedencia

- Principio de la cascada
  - **Especificidad**: cuando se aplica más de una regla a un mismo elemento existe un conflicto de especificidad
  - La ley de especificidad indica que “mientras más concreto es el selector, más valor tiene la regla”
  - Las reglas que se aplican localmente en el elemento HTML son más específicas que las que se definen en una hoja de estilo interna y estas a su vez son más específicas que las definidas en una hoja externa
  - Además, un selector con el atributo **id** prevalecerá sobre un selector con el atributo **class** y éste sobre cualquier selector **simple** o sin atributos

3. Reglas de estilo
3.3 Precedencia

- Principio de la cascada
  - **Ubicación**: las reglas que aparecen en último lugar son las que se aplican en caso de conflicto

```
<head>
   <title>Ejemplo hoja externa</title>
   <link rel="stylesheet" type="text/css" href="estilos1.css">
   <link rel="stylesheet" type="text/css" href="estilos2.css">
   <link rel="stylesheet" type="text/css" href="estilos3.css">
</head>
```

3. Reglas de estilo
3.3 Precedencia

- Principio de la cascada
  - En resumen
    - A la hora de aplicar reglas de estilo debemos tener en cuenta que hay estilos que se heredan entre los elementos
    - Cuando dos reglas afectan al mismo elemento, prevalece la más específica y entre dos reglas con la misma especificidad, se aplicará la última en aparecer
    - Hay que evitar el uso de la declaración **!important**

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

4. Propiedades CSS

- Propiedades
  - En CSS existen numerosas propiedades que afectan a los elementos HTML
    - https://developer.mozilla.org/es/docs/Web/CSS/Referencia_CSS
  - Propiedades básicas
    - https://developer.mozilla.org/es/docs/Web/CSS/CSS_Properties_Reference
  - Las propiedades están agrupadas en varias categorías: *fuentes*, *texto*, *tablas, posicionamiento*, etc.

4. Propiedades CSS

- Propiedades de fuente y texto
  - Fuente de letra

```
font-family: (nombre_familia | familia_genérica),
(nombre_familia | familia_genérica);
```

  - Tamaño de fuente

```
font-size: valor;
```

`font-size:` `xx-small` | `x-small` | `small` | `medium` | `large` | `x-large` | `xx-large`

`font-size:` `smaller` | `larger`

`font-size:` <longitud> | <porcentaje> | `inherit`

4. Propiedades CSS

- Propiedades de fuente y texto
  - Estilo de fuente

    `font-style: normal | italic | oblique;`

  - Grosor de fuente

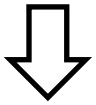
    `font-weight: valor;`

```css
font-weight: normal;
font-weight: bold;

/* Relativo al padre */
font-weight: lighter;
font-weight: bolder;

font-weight: 100;
font-weight: 200;
font-weight: 300;
font-weight: 400;
font-weight: 500;
font-weight: 600;
font-weight: 700;
font-weight: 800;
font-weight: 900;
```

4. Propiedades CSS

- Propiedades de acceso rápido
  - Algunas propiedades como **font** se pueden escribir de forma compacta

```
p {
    font-style: italic;
    font-variant: small-caps;
    font-weight: bold;
    font-size: 1em;
    line-height: 140%;
    font-family: ”Arial, sans-serif”; }
```



```
p {font: italic small-caps bold 1em/140% ”Arial”, sans-serif;}
```

4. Propiedades CSS

- Fuentes CSS3
  - Se pueden incluir fuentes directamente en un sitio web mediante @**font-face**, incluyendo la ruta relativa de la fuente
  - Formatos de fuentes para navegadores (no hay estándar, hay que proporcionar todas las posibles):

Formato	Descripción
eot	Embedded Open Type, Explorer
ttf	True Type Font, iOS
svg	Scalar Vector Graphics
woff	Web Open Font Format, Chrome

4. Propiedades CSS

- Fuentes CSS3
  - Ejemplo de aplicación:

```
@font-face {
      font-family: 'miFuente';
      font-weight: 400;
      font-style: normal;
      src: url('webfont.eot'); /* IE9 Compat Modes */
      src: url('webfont.eot?#iefix') format('embedded-opentype'),
      / *IE6-IE8 */
      url('webfont.woff2') format('woff2'), /*Super Modern Browsers*/
      url('webfont.woff') format('woff'), /*Pretty Modern Browsers*/
      url('webfont.ttf')  format('truetype'), /*Safari, Android, iOS */
      url('webfont.svg#svgFontName') format('svg'); /* Legacy iOS */
}
```

4. Propiedades CSS

- Propiedades de texto
  - Interlineado

    `line-height: valor;`
  - Alineación de texto

    `text-align: left | right | center | justify | initial;`
  - Decoración de texto

    `text-decoration: text-decoration-line | text-decoration-color | text-decoration- style | initial;`
  - Tabulado del texto

    `text-indent: <medida> | <porcentaje>;`

4. Propiedades CSS

- Propiedades para el fondo
  - Color de fondo

    ```
    background-color: color | transparent | inherit;
    ```

  - Imagen de fondo

    ```
    background-image: url | none | inherit;
    ```

  - Repetición y posición

    ```
    background-repeat: repeat | repeat-x | repeat-y |
    no-repeat | inherit;
    ```

  - Opacidad

    ```
    opacity: valor;
    ```

4. Propiedades CSS

- Valores
  - Cada propiedad CSS posee reglas diferentes sobre los valores que puede aceptar
  - Algunas aceptan un valor de una lista predefinida, otras números, porcentajes, URLs, colores,…
  - Cuando se asignen valores a propiedades de tipo numérico no se deben dejar espacios entre éstos y la unidad
  - Es posible utilizar unidades **absolutas** (*cm, mm, in, pt*)
  - O **relativas** (*px, em, rem, ex, %*)

4. Propiedades CSS

- Unidades absolutas
  - Se aplican siempre igual, independientemente de cómo sea la pantalla o la resolución

Dimensión	Significado	Descripción
pt	puntos	Utilizada para fuentes. Por norma general un punto equivale a 1/72 pulgadas
pc	picas	Equivale a 12 puntos o 1/6 pulgadas
mm	milímetros	Medida métrica
cm	centímetros	Centímetros
in	pulgadas	Pulgadas

4. Propiedades CSS

- Unidades relativas
  - Estas unidades se ajustan a cada tipo de dispositivo

Dimensión	Significado	Descripción
px	píxeles	Cada punto de la pantalla que forma una imagen es un pixel. La cantidad de píxeles que pueden ser representados varía según la resolución
ex	altura de x	Representa la altura del carácter x en la pantalla. Es relativo al tamaño base del navegador
em	altura de M	Tamaño del carácter M en la pantalla (**16 píxeles por defecto** en el navegador)
%	porcentaje	Porcentaje respecto al tamaño total de la pantalla

4. Propiedades CSS

- Uso de unidades
  - Recomendaciones
    - Buscando el mejor ajuste en la mayoría de situaciones, en general son mejores las unidades relativas
    - El dimensionamiento en píxeles y ems nos permite un mejor control y adaptabilidad a cualquier dispositivo
    - El tamaño de los ems se establece según el tamaño base que tenga el navegador (eso es lo que lo otorga su adaptabilidad a cualquier tipo de pantalla)
    - Problema del píxel: no todas las pantallas son iguales y un píxel no es igual en todas las pantallas (se soluciona siempre que se pueda redimensionar el texto en las páginas, recomendación en las guías de *accesibilidad*)

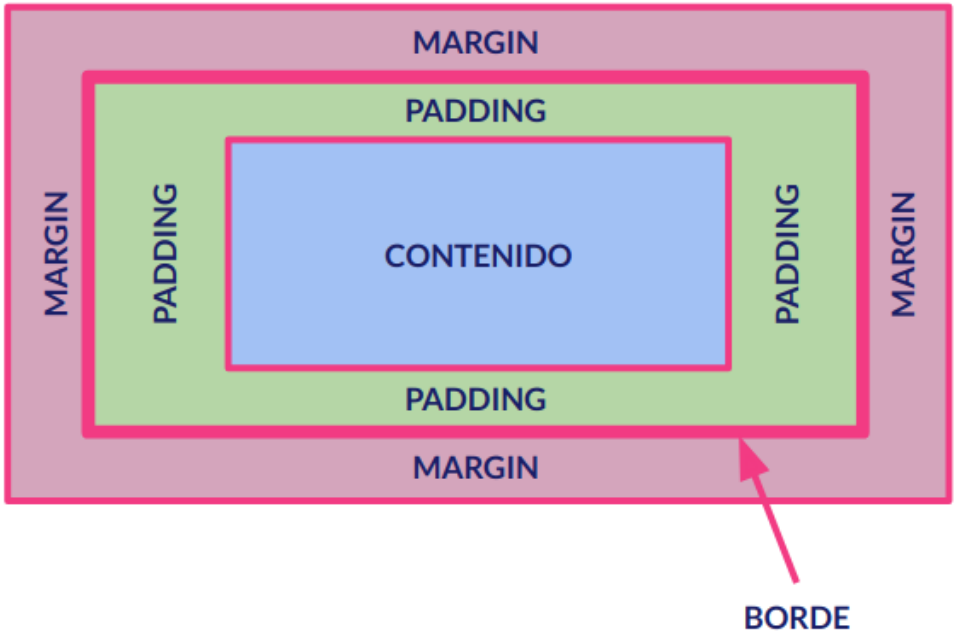
4. Propiedades CSS

- Prefijos de navegadores (vendor prefixes)
  - Para algunos estilos, los navegadores pueden tener su propia versión si aún la propiedad no está oficialmente soportada
  - Este tipo de valores tienen un prefijo que indica para qué navegador se aplica
  - Ejemplo con propiedad **border-radius**:

```
.round {
    -moz-border-radius: 8px; /* Mozilla Firefox, IceWeasel */
    -webkit-border-radius: 8px; /* Webkit: Safari, Chrome, Chromiun */
    -ms-border-radius: 8px; /* Microsoft */
    -o-border-radius: 8px; /* Opera */
    border-radius: 8px; /* W3C standard */
}
```

5. Modelo de caja

- Modelo de caja CSS
  - En HTML se considera que todo elemento es una “caja”
  - Los navegadores crean una caja virtual alrededor de todos los elementos HTML para determinar el área que ocupan



5. Modelo de caja

- Modelo de caja CSS
  - A cualquier etiqueta HTML se le pueden aplicar propiedades de relleno, margen, borde, etc.
    - Todo elemento HTML tiene un tamaño y un límite visible implícito, que puede hacerse visible con **border**
    - El relleno antes de ese límite puede aplicarse con **padding**
    - La distancia con los elementos vecinos se especifica con **margin**
    - Estas propiedades se pueden aplicar de forma conjunta **(border, padding, margin)** o de forma separada (**-top, -right -bottom, -left,**)

5. Modelo de caja

- Modelo de caja CSS
  - Ejemplos equivalentes:

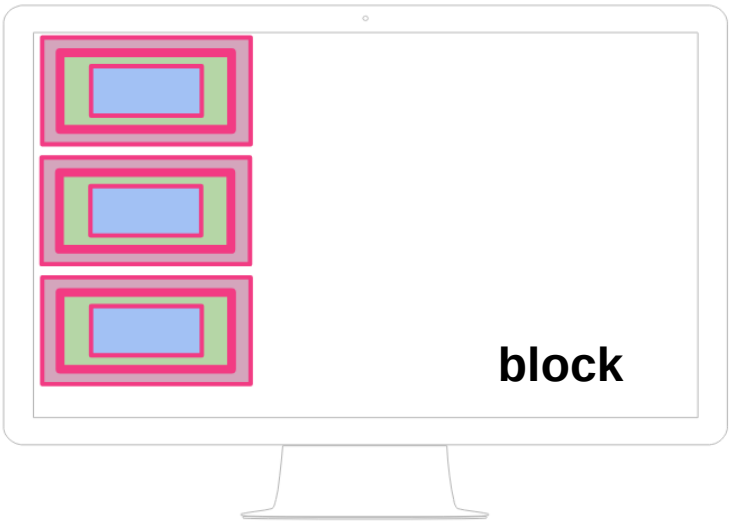
1)
```
padding: 2px 1px 2px 1 px;
```
/* top, right, bottom, left */

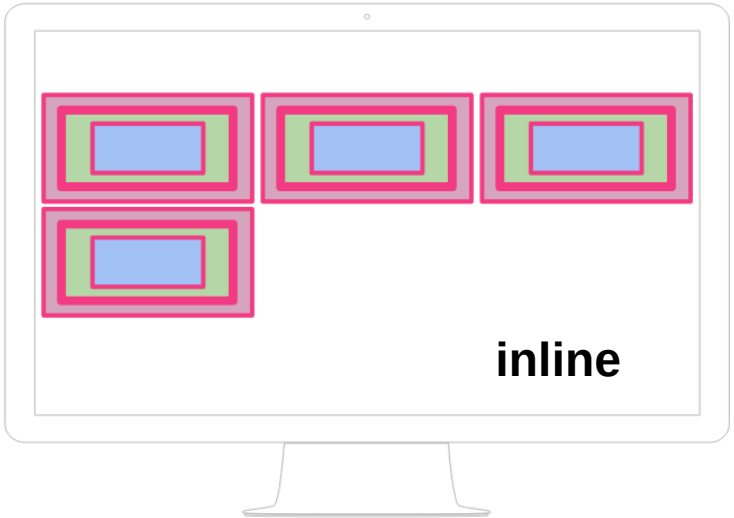
2)
```
padding: 2px 1px;
```
/* top y bottom: 2px, right y left: 1px */

3)
```
padding-top: 2px;
padding-right: 1px;
padding-bottom: 2px;
padding-left: 1px;
```

5. Modelo de caja

- Propiedad display
  - Para organizar las cajas en pantalla, los elementos se clasifican en dos tipos básicos:
    - **block** → tamaño personalizado; generan saltos de línea
    - **inline** → tamaño determinado por su contenido y no generan saltos de línea



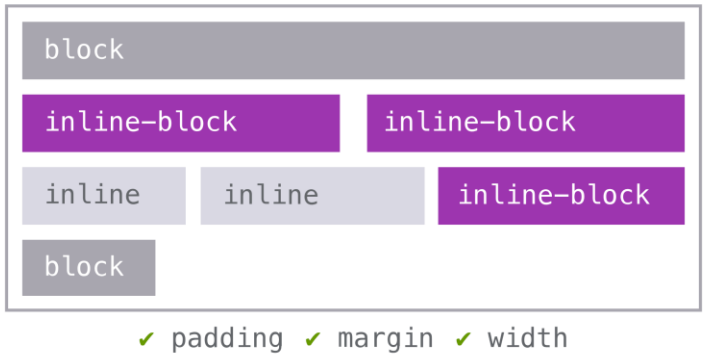


5. Modelo de caja

- Propiedad display
  - Debido a sus características, los elementos **block** son apropiados para crear columnas y secciones en la página y los **inline** son adecuados para representar contenido
  - Por esto, las etiquetas semánticas que definen la estructura de un documento (<section>, <nav>, <header>, <footer>, ,…) por defecto son tipo **block**
  - Las que representan contenido (, o) son **inline**
  - Este comportamiento por defecto se puede cambiar con la propiedad **display** que define el tipo de caja usado para representar el elemento en pantalla

5. Modelo de caja

- Propiedad display
  - Valores de **display**
    - **none** → elimina el elemento (no ocupa espacio)
    - **block** → muestra el elemento en una nueva línea y con un tamaño personalizado
    - **inline** → muestra el elemento en la misma línea
    - **inline-block** → muestra el elemento en la misma línea y con un tamaño personalizado (permiten crear secciones del tamaño que deseemos y ubicarlas en la misma línea si es necesario)



5. Modelo de caja

- Propiedad display
  - Visible/invisible
    - Mediante CSS podemos hacer que un elemento aparezca o desaparezca con la propiedad `visibility` (valores `visible` y `hidden`)
    - Puede vincularse a `display` para controlar el comportamiento de un elemento no visible
    - Si un elemento se hace invisible dejará su hueco en la página; para evitarlo hay que añadir la propiedad `display:none;`

6. Elementos flotantes

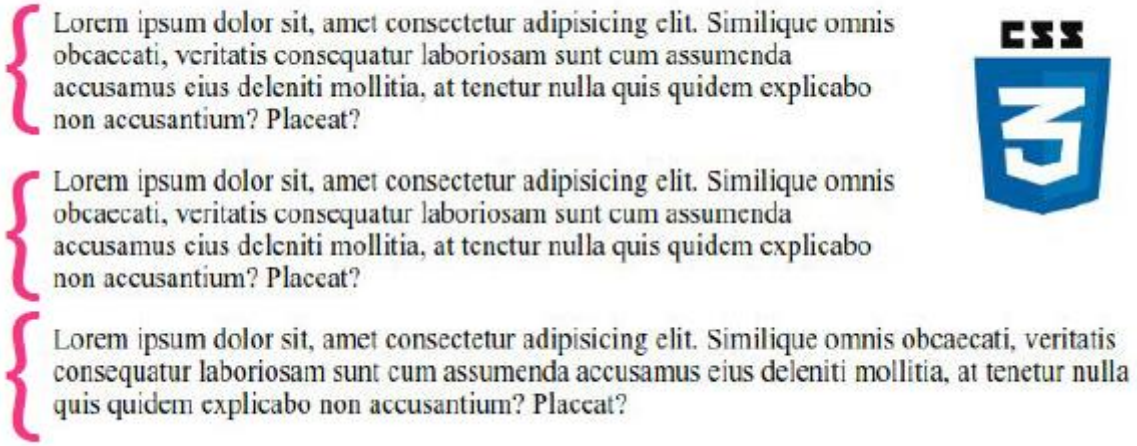
- Propiedad float
  - float (**left** o **right**) es una propiedad CSS pensada para especificar cómo se dispone el texto alrededor de una imagen

```
img {
   float: right;
   margin: 0px 1em;
}
```

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

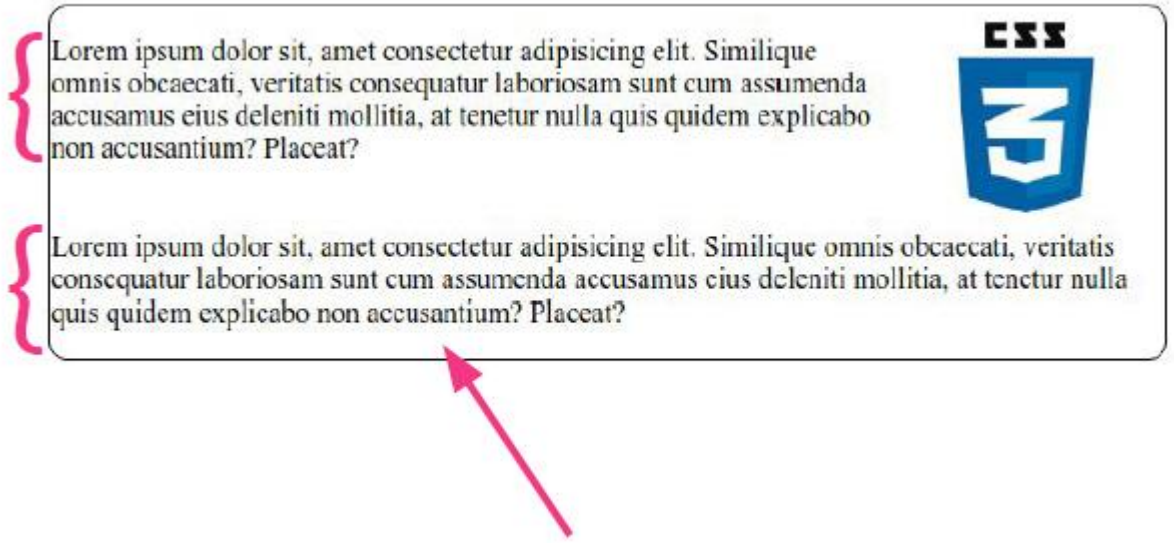
6. Elementos flotantes

- Propiedad float
  - Si sigue habiendo “hueco vertical” en el lugar contrario al que se ha flotado la imagen, los elementos se siguen añadiendo ahí hasta que sobrepasan en la “vertical” al elemento flotante



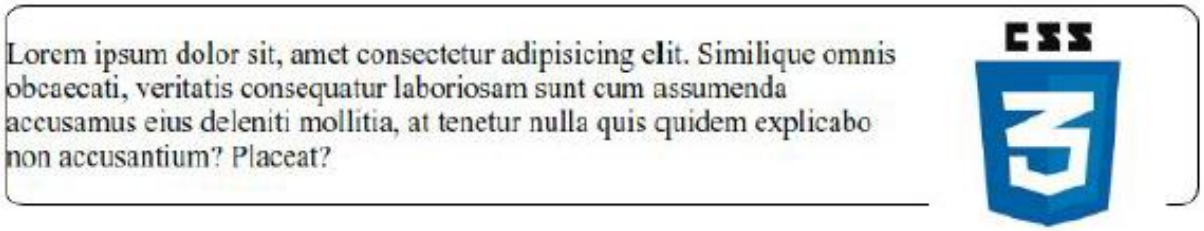
6. Elementos flotantes

- Propiedad clear
  - Si queremos forzar a que un elemento deje de flotar respecto a otro hay que añadir la propiedad **clear: right** (o **left** o **both**) y ese elemento ya se añadirá tras el fin en vertical del elemento flotante



6. Elementos flotantes

- Problema: clearfix hack
  - Posible problema al posicionar:



  - El contenedor solo tiene en cuenta la altura del párrafo, no la de la imagen (pierde la referencia del elemento flotante)
  - Solución (propiedades aplicadas al elemento contenedor):

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

7. Colores y transparencia

- Definición de colores en CSS
  - Opciones para definir colores en las hojas de estilo (fuentes, fondos, líneas, etc.):
    - Nombre del color en inglés

```
.blue {
    background-color: dodgerblue;
    border: 1px solid blue;
}
```

    - RGB (Red, Green, Blue): 3 pares de valores hexadecimales representando la mezcla de rojo, verde y azul precedidos de #

```
#lista3 a:hover {
    color: #000000;
    background-color: #FFF;
}
```

    - RGB con valores decimales (0 a 255)

```
#lista3 li:hover {
    background-color: rgb(0, 128, 250);
}
```

7. Colores y transparencia

- RGB con transparencia
  - Desde CSS3 se ofrece la posibilidad de aplicar transparencia a un color (a través del *canal alfa*)
  - Se especifica añadiendo un cuarto parámetro (de 0 a 1) al valor RGB

```
#lista3 li:hover {
    background-color: rgb(0, 128, 250, 0.2);
}
```

7. Colores y transparencia

- Color HSL
  - HSL (*Hue*, *Saturation*, *Lightness*): color, saturación, luz
  - Ejemplo de aplicación: hsl (60º, 90%, 60%)
  - Se basa en una rueda de color de 360º, donde cada color se sitúa en:
    - Amarillo → 60º
    - Verde → 120º
    - Cyan → 180º
    - Azul → 240º
    - Magenta → 300º
    - Rojo → 360º

```
.title {
    color: hsl(60, 90%, 60%);
    text-align: center;
    font-weight: bold;
}
```

  - Se le puede aplicar transparencia mediante canal alfa

8. Estilos en formularios

- Estilos en formularios
  - Al igual que con otros elementos, se ha dejado de utilizar tablas para maquetar formularios
  - Hay etiquetas HTML específicas de formularios que pueden resultar útiles para aplicarle reglas CSS:
    - `legend` → agrupa todo el formulario
    - `fieldset` → agrupa cada elemento del formulario
    - `label` → etiqueta asociada a cada campo
  - Otra opción es utilizar bloques para agrupar elementos y al

8. Estilos en formularios

- Estilos en formularios
  - Existen pseudo-clases para elementos de formularios que facilitan la introducción de datos al usuario, como cuando un campo es requerido, cuando recibe el foco o cuando es válido o no

```
input:required {
    border: 1px solid black;
    font-weight: bold;
}
input:focus, textarea:focus {
    border: 2px solid cadetblue;
    background-color: lightblue;
}

input:focus:valid, textarea:focus:valid {
    color: green;
    background-color: lightgreen;
}

input:focus:invalid, textarea:focus:invalid {
    color: darkred;
    border: 2px solid red;
    background-color: indianred;
}
```

9. Reseteo de propiedades

- Hojas de reseteo CSS
  - Los navegadores tienen sus propias hojas de estilos por defecto que se aplican a los distintos elementos de la página con el objetivo de hacer las páginas sin estilos más atractivas
  - Estos estilos no tienen porqué coincidir de un navegador a otro y si queremos consistencia en todos los navegadores se deben usar **hojas de reseteo CSS**
  - Eliminan ciertas características que imponen los navegadores
  - En Internet existen varias hojas de estilo para reseteo que podemos utilizar en nuestros proyectos

9. Reseteo de propiedades

- Hojas de reseteo CSS
  - Aspectos sobre este tema que debemos considerar:
    - Las hojas de estilo de resetear son usadas por *frameworks CSS* (como *BootStrap*) que buscan que todas las páginas siempre luzcan igual, independientemente del navegador
    - Si se usan, deben ser adaptadas a cada proyecto
    - Hay que usarlas con cuidado porque podemos eliminar estilos que dábamos por sentado
    - El trabajo posterior tras usarlas es mayor pero el resultado es más personalizado

10. Variables CSS

- Variables CSS
  - Se declaran como una propiedad más y se escriben prefijadas de un doble guion: **--nombreVariable: …;**

```
--mi_color: #A5B5C5;
```

  - Se asignan mediante la función var():

    propiedad: var(--nombreVariable);

```
background-color: var(--mi_color);
```

  - Permiten asignar como argumento un valor alternativo para el caso en el que la variable no tuviera un valor asignado:

```
background-color: var(--mi_color, #A3B3C3);
```

10. Variables CSS

- Alcance de las variables CSS
  - Las variables que se declaran dentro de **:root** o la etiqueta **html** tienen un **alcance global**
  - Las variables que se declaran en otros nodos o selectores están **acotadas a ese nodo** y sus **descendientes**
  - Las variables se pueden **sobreescribir** en nodos inferiores, haciendo que tengan un valor a nivel global (etiqueta html) y un valor diferente a nivel local (por ejemplo, en una clase concreta)
  - **:root {… }**
    - El selector *:root* equivale a la etiqueta *html* y se utiliza habitualmente para establecer variables globales
    - La única diferencia entre :root {} y html {} es que el primero tiene mayor especificidad

11. Recomendaciones uso de CSS

- Recomendaciones CSS
  - Minimizar CSS para producción
  - Borrar reglas innecesarias y duplicadas
  - Ordenar las propiedades por orden alfabético
  - CSS en la cabecera
  - Si el CSS es muy grande partirlo en varios
  - Organiza las reglas poniendo las que tengan relación juntas
  - Documentar los ficheros CSS
  - Uso de preprocesadores CSS en proyectos grandes

*. Referencias

- Bibliografía y referencias
  - Cursos de Openwebinars.net "*Curso de HTML5 y CSS3*", "*Maquetación web con CSS"* de Juan Diego Pérez
  - Libro "Diseño de Interfaces Web" de Eugenia Pérez Martínez / Pello Xabier Altadill Izura – Ed. Garceta
  - Libro "Diseño de Interfaces Web" de Diana García-Miguel López – Ed. Síntesis
  - https://developer.mozilla.org/es/docs/Web/CSS