

Diseño de Interfaces Web

Fundamentos de CSS3



```
body {  
    font: x-small  
    background: /  
    color: black;  
    margin: 0;  
    padding: 0;
```





Fundamentos de CSS3

1. [Introducción a CSS](#)
2. [Hojas de estilo](#)
3. [Selectores](#)
4. [Precedencia](#)
5. [Propiedades](#)
 - [Propiedades de fuente](#)
 - [Propiedades de texto](#)
 - [Propiedades para el fondo](#)
 - [Propiedades de listas](#)
 - [Valores de las propiedades](#)
 - [Unidades absolutas](#)
 - [Unidades relativas](#)
 - [Prefijos de navegadores](#)
6. [Modelo de caja \(Box model\)](#)
7. [Box-sizing](#)
8. [Elementos flotantes](#)
9. [Posicionamiento de elementos](#)
10. [Columnas](#)
11. [Centrado del layout](#)
12. [Colores y transparencias](#)
13. [Sombras](#)
14. [Gradientes](#)
15. [Variables CSS](#)
16. [Reseteo de propiedades](#)
17. [Recomendaciones uso CSS](#)
- *. [Referencias](#)



1. Introducción a CSS

- Origen de CSS
 - Inicialmente, HTML incluía el contenido, la estructura y las instrucciones de formato en un único documento
 - A principios de los años 90, HTML no poseía todas las capacidades actuales
 - Las posibilidades para controlar el aspecto de los documentos eran pocas y el archivo HTML resultaba complejo al estar mezclados contenidos, estructuras e instrucciones de formato
 - Para solucionarlo, el W3C crea el lenguaje **CSS** (1994) 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, animaciones, maquetación, diseño responsive, etc.



1. Introducción a CSS

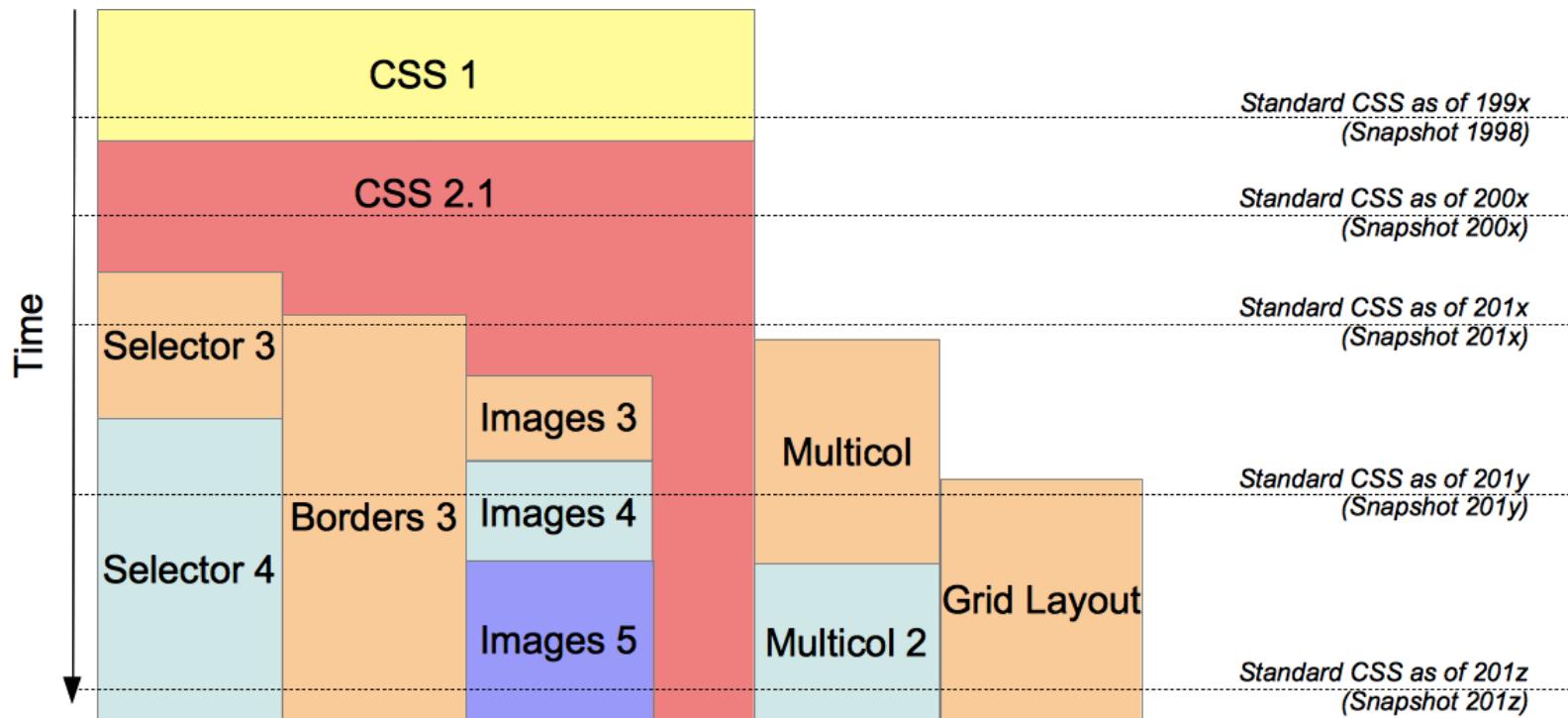
- Niveles de CSS
 - Cada nivel de CSS se construye sobre el anterior, añadiendo funciones al nivel previo
 - Los navegadores implementan estas especificaciones de forma distinta
 - ❖ CSS 1(1996): propiedades de fuente, colores, alineación,...
 - ❖ CSS 2 (1998): posicionamiento, tipos de medios,...
 -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





1. Introducción a CSS

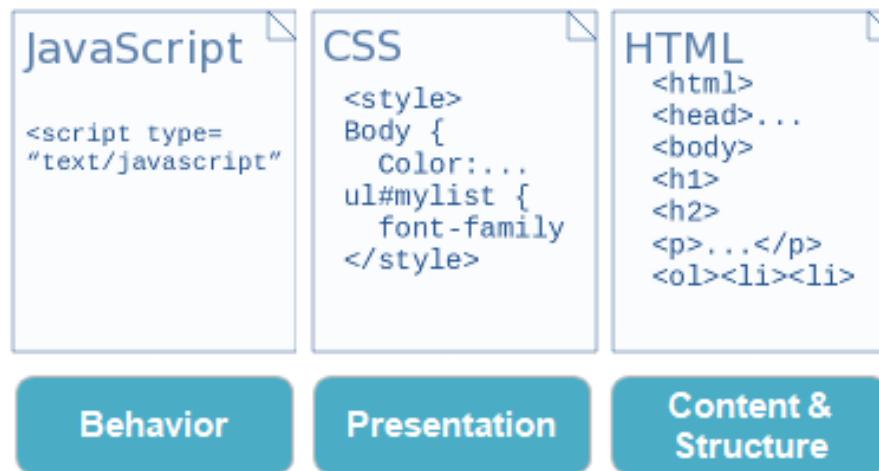
- Niveles de CSS





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

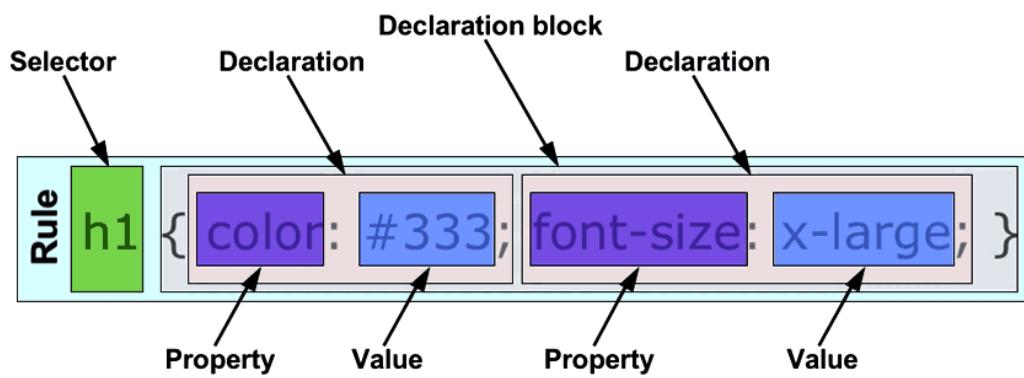
```
body {  
    background-color: lightblue;  
}  
  
h1 {  
    color: white;  
    text-align: center;  
}  
  
p {  
    font-family: verdana;  
    font-size: 20px;  
}
```



2. Hojas de estilo

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

CSS Terminology





2. Hojas de estilo

- Partes de una regla de estilo
 - Las reglas de estilo tienen dos componentes
 - **Selector** → selecciona el/los elementos sobre los que actuará la regla
 - **Declaración** → establece las propiedades y valores a aplicar sobre los elementos seleccionados

selector {propiedad1: valor1;}



declaración

p {color: red; }

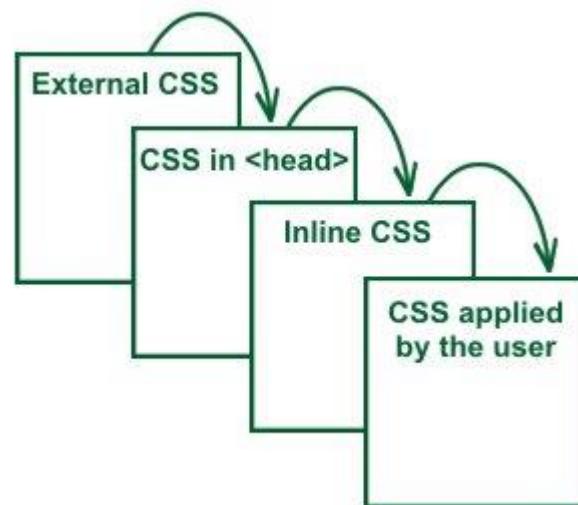
selector

propiedad

valor

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
 - La regla de estilo CSS se escribe 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. Selectores

- Selectores de tipo
 - 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. Selectores

- Selectores combinadores
 - Si se quieren seleccionar los elementos que están dentro de otros debemos separarlos con un **espacio**

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

- Para seleccionar los hijos directos (árbol DOM) de un elemento se utiliza “>”

```
ul > li {color: green;}
```



3. Selectores

- Selectores combinadores
 - Si queremos seleccionar solo el elemento que está justo **después** de otro (hermano adyacente) usaremos “+”

```
h1 + p {color: magenta; }
```
 - Para seleccionar todos los elementos hermanos que siguen a otro utilizaremos “~” (no tienen que estar inmediatamente precedidos, pero sí deben tener el mismo padre)

```
h2 ~ p {color: aqua; }
```



3. Selectores

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

```
img[alt] {border: 5px 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 incluya la palabra “valor”	a[class~="menu"]
Etiqueta[atributo\$=valor]	Selecciona aquellas etiquetas cuyo atributo acabe en ese valor	img[src\$=.png]



3. Selectores

- Selectores específicos
 - Cuando queremos aplicar un estilo determinado a unos elementos concretos y no a todos con la misma etiqueta podemos usar selectores más específicos como:
 - Identificadores
 - Clases
 - Pseudo clases (pseudo-selectores)
 - Pseudo elementos



3. 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="intro">
```

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



3. Selectores

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

```
p#intro { 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
- También es posible:

```
#intro { font-size: 14px; }
```



3. 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:

```

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

```

```



3. 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. Selectores

- Pseudo-selectores (pseudo-clases)
 - 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. Selectores

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



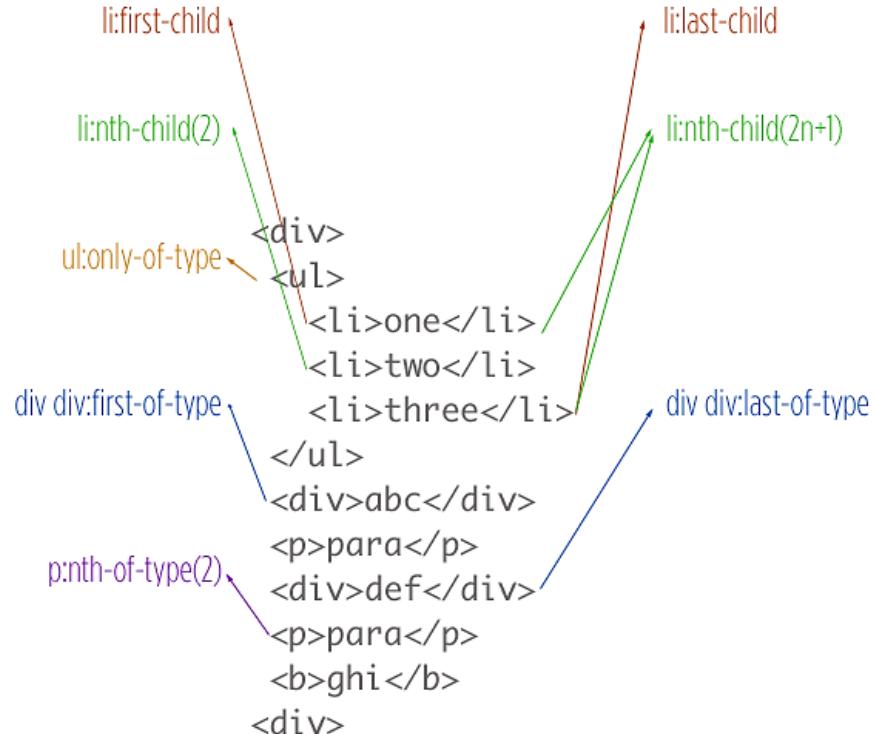
3. Selectores

- Pseudo-selectores (pseudo-clases)
 - De posición
 - :first-child
 - :last-child
 - :only-child
 - :nth-child(n) / :nth-child(odd) / :nth-child(even)
 - :nth-last-child()
 - :first-of-type
 - :last-of-type
 - :nth-of-type()
 - :nth-last-of-type()



3. Selectores

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





3. Selectores

- Pseudo-selectores (pseudo-clases)
 - Otros
 - :not(selector)
 - :is()
 - :where()
 - :has()
 - :placeholder-shown
 - ...



3. 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. Selectores

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

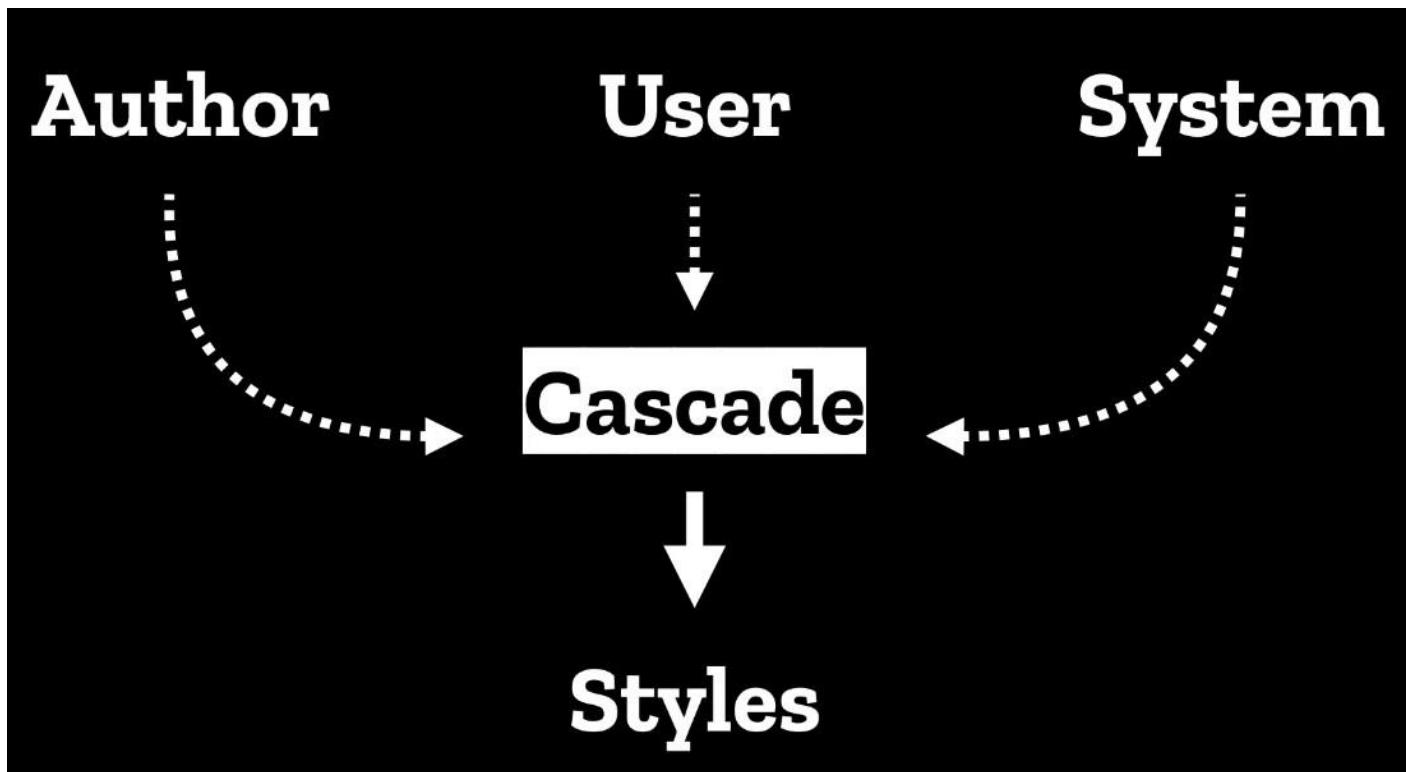
4. Precedencia

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



4. Precedencia

- Principio de la cascada





4. Precedencia

- Herencia
 - Hay propiedades de CSS que afectan a los elementos definidos por el selector y también a sus descendientes
 - **inherit**: valor permitido en todas las propiedades CSS que hace que el elemento al cual se aplica tome el valor calculado de la propiedad de su elemento padre
 - **initial**: es el valor por defecto de la propiedad (cancela a **inherit**)



4. Precedencia

- Herencia

- 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 hereda)

color



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

border



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

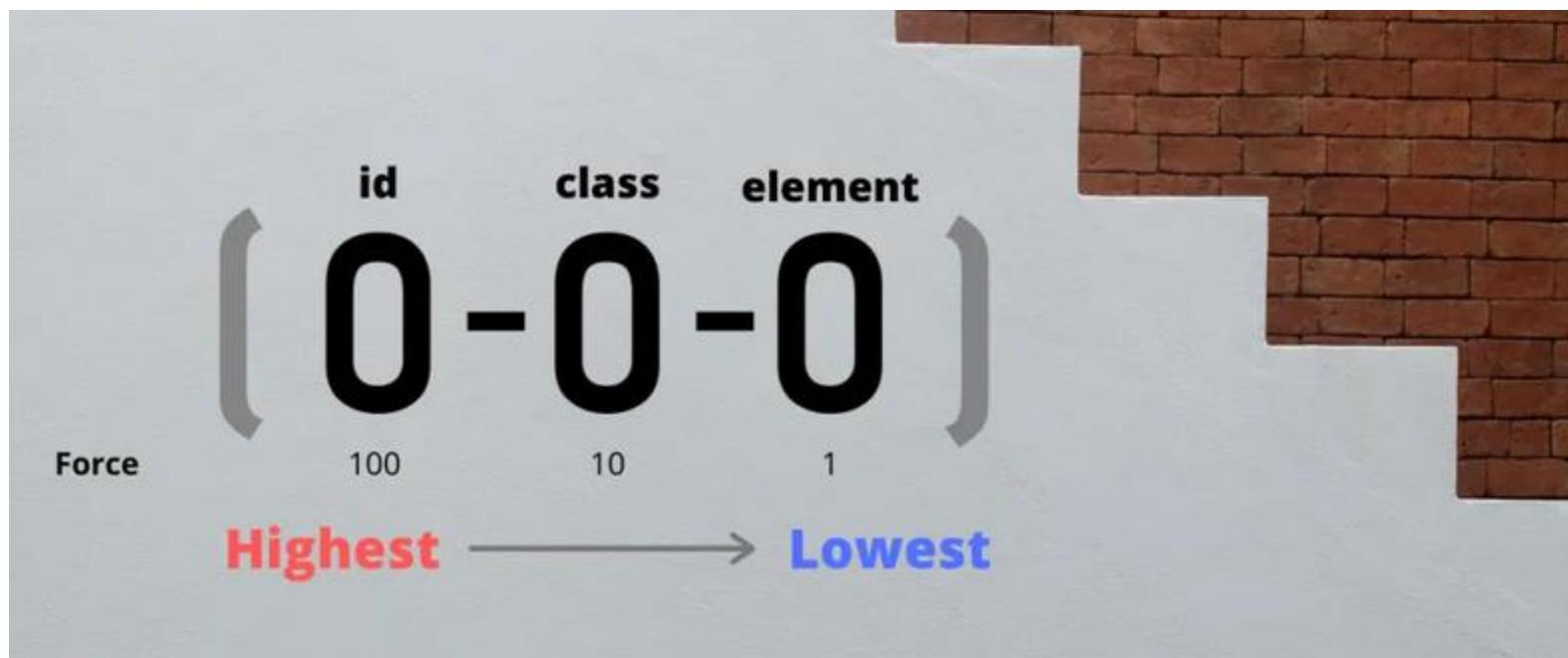


4. Precedencia

- 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 aplicadas localmente en el elemento HTML son más específicas que las que se definen en una hoja de estilo interna y estas son más específicas que las definidas en una 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

4. Precedencia

- Especificidad
 - Cálculo de la especificidad





4. Precedencia

- Ubicación
 - Las reglas que aparecen en último lugar son las que se aplican en caso de conflicto

td, p {color: green;}
→ p {color: red;}

```
<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>
```



4. 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;  
}
```



5. Propiedades CSS

- Propiedades
 - En CSS existen numerosas propiedades que afectan a los elementos HTML
 - Listado de propiedades CSS
 - https://developer.mozilla.org/es/docs/Web/CSS/Referencia_CSS
 - Listado de propiedades básicas
 - https://developer.mozilla.org/es/docs/Web/CSS/CSS_Properties_Reference
 - Se agrupan en categorías: *fuentes, texto, fondo, listas, tablas, layout, transformaciones, animaciones*, etc.



5. Propiedades CSS

- Propiedades de fuente
 - Tipo 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`



5. Propiedades CSS

- Propiedades de fuente
 - Estilo de fuente

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

- Grosor de fuente

`font-weight: valor;`

```
1  font-weight: normal;
2  font-weight: bold;
3
4  /* Relativo al padre */
5  font-weight: lighter;
6  font-weight: bolder;
7
8  font-weight: 100;
9  font-weight: 200;
10 font-weight: 300;
11 font-weight: 400;
12 font-weight: 500;
13 font-weight: 600;
14 font-weight: 700;
15 font-weight: 800;
16 font-weight: 900;
```



5. Propiedades CSS

- Propiedades de fuente
 - Estas propiedades se pueden escribir de forma abreviada (shorthand) mediante **font**

```
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;}
```



5. Propiedades CSS

- Propiedades de fuente
 - Se pueden incluir fuentes directamente en un sitio web mediante **@font-face**, incluyendo la ruta relativa del archivo de la fuente

```
@font-face {  
    font-family: 'miFuente';  
    src: url('../fonts/webfont.ttf') format('ttf');  
}
```

- Formatos de fuentes para navegadores: **.ttf** (*True Type Font*), **.eot** (*Embedded Open Type*), **.woff** (*Web Open Format*), **.svg**



5. 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>;`



5. Propiedades CSS

- Propiedades para el fondo de los elementos

- Color de fondo

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

- Opacidad

```
opacity: valor;
```



5. Propiedades CSS

- Propiedades para el fondo de los elementos

- Imagen de fondo

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

- Repetición y posición

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

```
background-attachment: fixed;
```

- Disposición

```
background-position: | cover | inherit;
```

```
background-size: contain | cover | inherit;
```



5. Propiedades CSS

- Propiedades de listas

- Estilo de lista

```
list-style-type: none | disc | circle;
```

- Posición

```
list-style-position: inside | outside;
```

- Imagen

```
list-style-image: none | url;
```



5. Propiedades CSS

- Valores de las propiedades
 - Las propiedades CSS pueden aceptar distintos tipos de valores: números, porcentajes, un valor de una lista predefinida, URLs, colores,...
 - Cuando se asignen valores a propiedades de tipo numérico no se deben dejar espacios entre estos y la unidad
 - Para especificar dimensiones es posible utilizar unidades **absolutas** (*cm, mm, in, pt*) o **relativas** (*px, %, em, rem*)
 - Buscando el mejor ajuste en la mayoría de situaciones, en general son mejores las unidades relativas



5. 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



5. 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
rem	root em	Relativo al tamaño de fuente raíz del html (16 píxeles por defecto en el navegador)
em	altura de m	Tamaño relativo al tamaño de fuente actual
%	porcentaje	Porcentaje respecto al tamaño total de la pantalla
vw, vh	viewport width / viewport height	Unidades relativa al viewport



5. Propiedades CSS

- Unidades em y rem
 - **em** y **rem** son unidades **flexibles**
 - Son unidades escalables que se traducen por el navegador en valores de **pixeles**
 - La diferencia entre una y otra es cómo el navegador determina el valor en px al que se traducen
 - Aportan flexibilidad y proporcionalidad a nuestros diseños y la capacidad de que los propios usuarios puedan escalar el tamaño de los elementos



5. Propiedades CSS

- Unidades rem
 - Las unidades **rem** son relativas al valor de fuente del **elemento raíz de la página** (al elemento **html**)
 - Este tamaño de fuente raíz es multiplicado por el valor de rem usado en nuestro elemento

```
html {  
    font-family: sans-serif;  
    font-size: 20px;  
}  
  
h1 {  
    font-size: 1rem;  
    padding: 2rem;  
}
```



5. Propiedades CSS

- Unidades em
 - Las unidades **em** son relativas al tamaño de fuente del **elemento en el que son utilizadas** (y no, como se cree habitualmente, relativas al valor de su elemento padre)
 - En algunos casos, los tamaños de la fuente del elemento padre pueden afectar a los valores em, pero cuando eso ocurre es solamente **por herencia** (a menos que explícitamente se sobreesciban con una unidad no sujeta a herencia, como **px** o **vw**)



5. Propiedades CSS

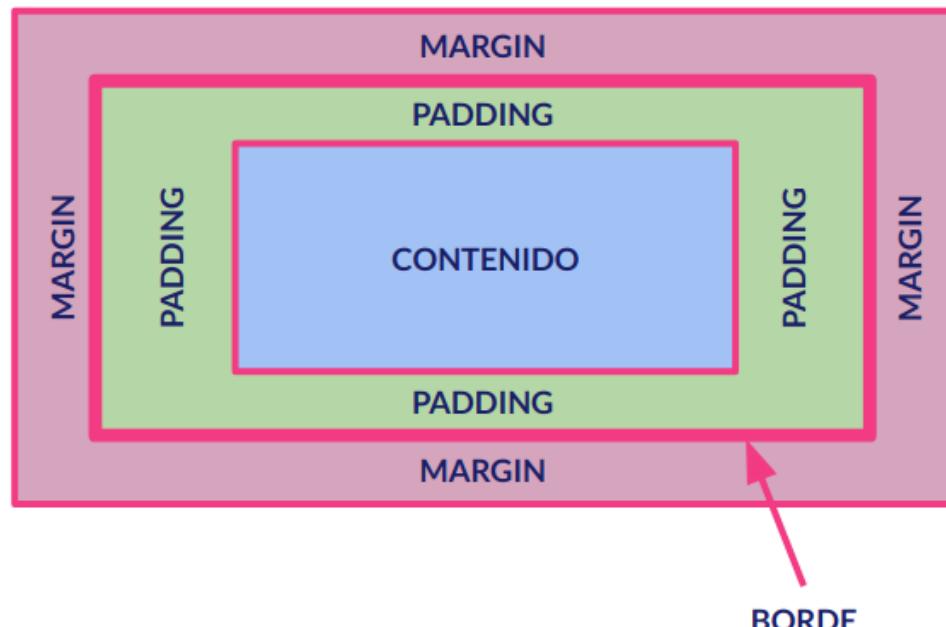
- Prefijos de navegadores (vendor prefixes)
 - Para algunos estilos, los navegadores pueden tener su propia versión si aún la propiedad no está oficialmente soportada
 - Este tipo de valores tienen un prefijo que indica para qué navegador se aplica
 - Ejemplo con propiedad **border-radius**:

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



6. Modelo de caja

- Modelo de caja CSS (Box model)
 - 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





6. Modelo de caja

- Modelo de caja CSS
 - A cualquier etiqueta HTML se le pueden aplicar propiedades de **relleno**, **margin** y **borde**
 - 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 entre 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**,)



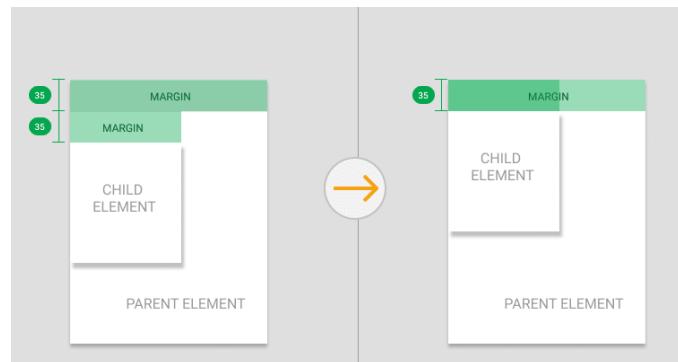
6. Modelo de caja

- Modelo de caja CSS
 - Ejemplos equivalentes:

- 1) margin: 2px 1px 2px 1px; /* top, right, bottom, left */
- 2) margin: 2px 1px; /* top y bottom: 2px, right y left: 1px */
- 3) margin-top: 2px;
margin-right: 1px;
margin-bottom: 2px;
margin-left: 1px;

6. Modelo de caja

- Modelo de caja CSS
 - Colapso de márgenes
 - Ocurre cuando dos márgenes (verticales u horizontales) entran en contacto
 - En este caso, **los márgenes no se suman**: el margen resultante es el **valor del mayor** de ellos
 - También ocurre entre elementos padre e hijos: si se le aplica un margen horizontal o vertical al hijo, este margen se aplica al padre



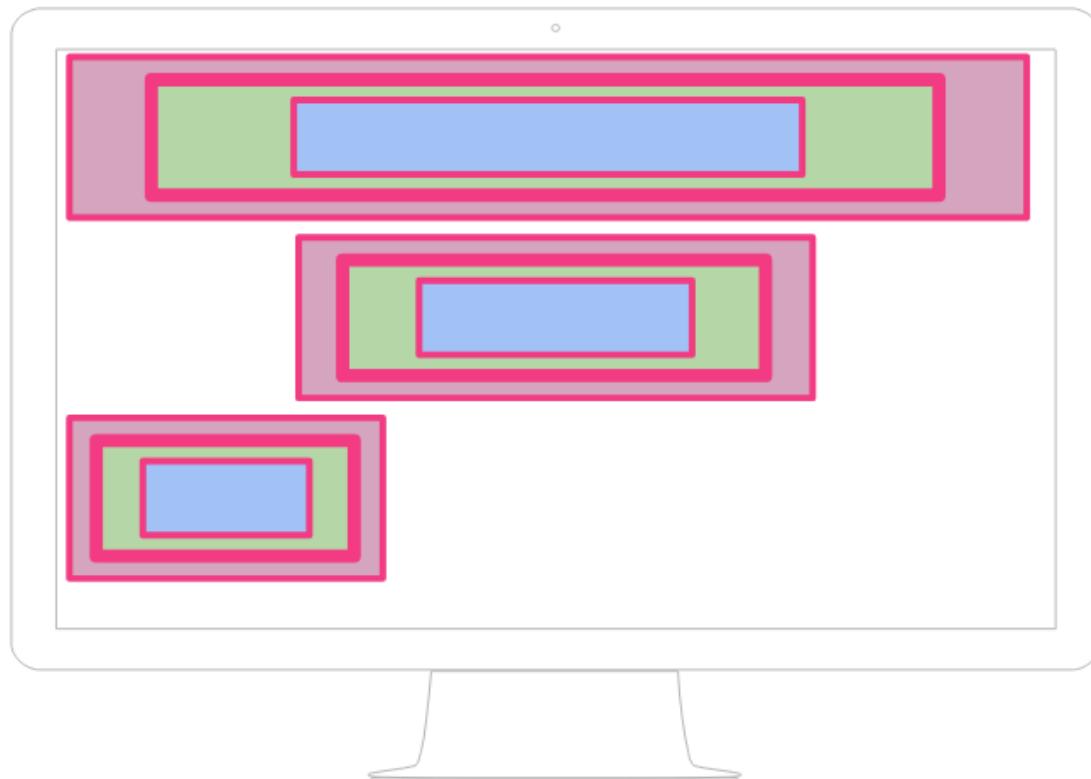


6. Modelo de caja

- Propiedad display
 - No todas las cajas se comportan igual cuando las añadimos
 - Cada elemento HTML tiene un comportamiento por defecto pero se puede modificar con los valores de **display**
 - Valores display:
 - **block**
 - **inline**
 - **inline-block**
 - **none**
 - **flex**
 - **grid**
 - Valores relacionados con **tablas**

6. Modelo de caja

- Propiedad display
 - **block (en bloque)**



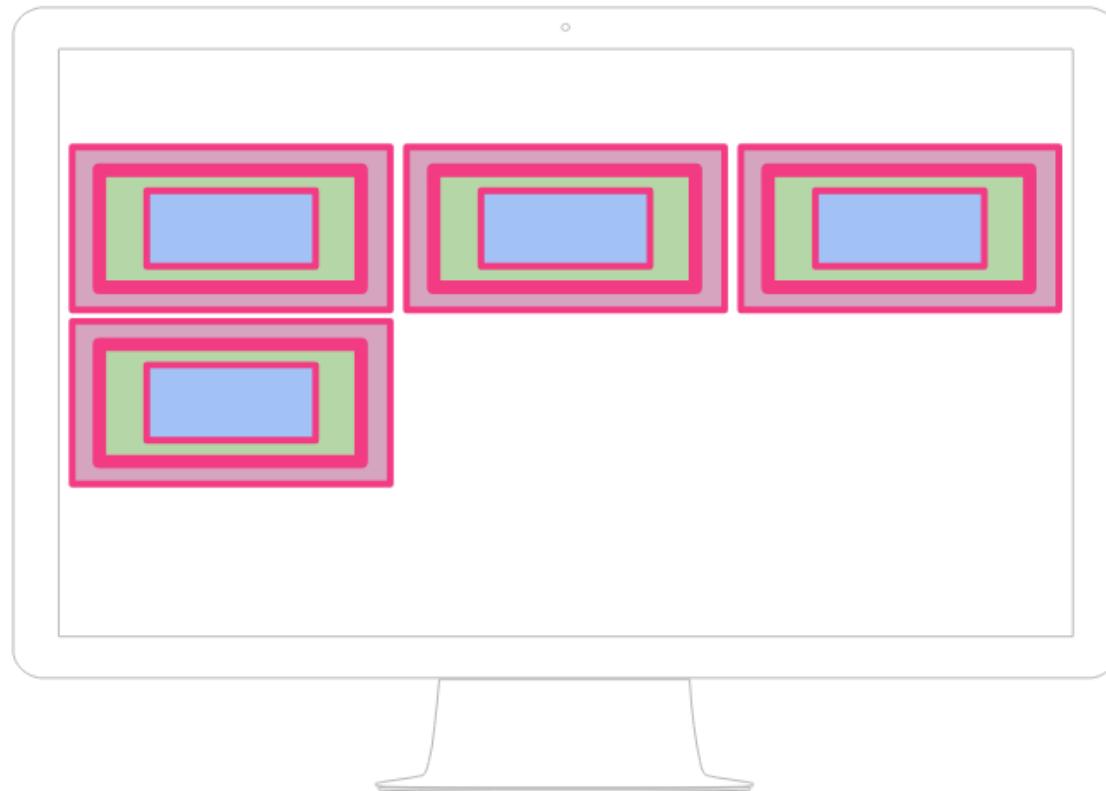


6. Modelo de caja

- Propiedad display
 - **block (en bloque)**
 - ✓ Generan saltos de línea: rompen el flujo provocando un salto de línea, tanto anterior como posterior
 - ✓ Tamaño personalizado (admiten width y height)
 - ✓ Si no especificamos tamaño, por defecto ocuparán toda la anchura de la etiqueta que los contiene (la etiqueta contenedora)
 - <h1>, <p>, <div>, <section>, , <nav>, ...

6. Modelo de caja

- Propiedad display
 - inline (en línea)





6. Modelo de caja

- Propiedad display
- **inline (en línea)**
 - ✓ No generan saltos de línea y se van colocando uno al lado del otro (mientras caben)
 - ✓ Tamaño determinado por el contenido
 - ✓ Ignoran *width* y *height*
 - ✓ Aceptan *margin* aunque solo los **valores horizontales** (margin-right y margin-left)
- , <a>, , *, ...



6. Modelo de caja

- Propiedad display
 - **inline-block**
 - Muestra el elemento en la misma línea y con un tamaño personalizado
 - Podemos asignarle *width*, *height* y *margin vertical*



6. Modelo de caja

- Propiedad display
 - **display: none**
 - ✓ Hace desaparecer al elemento de la página
 - ✓ No dejan un espacio vacío, aunque siguen en el código HTML
 - ✓ La propiedad **visibility: hidden** sí que deja ese hueco aunque no se muestre



6. Modelo de caja

- Propiedad display
 - Valores relativos a **tablas**
 - Los elementos con estos valores simularán el comportamiento del elemento de tabla análogo
 - Valores:
 - **table**
 - **table-row**; **table-cell**
 - **table-caption**
 - **table-column**
 - **table-colgroup**; **table-header-group**; **table-footer-group**; **table-row-group**

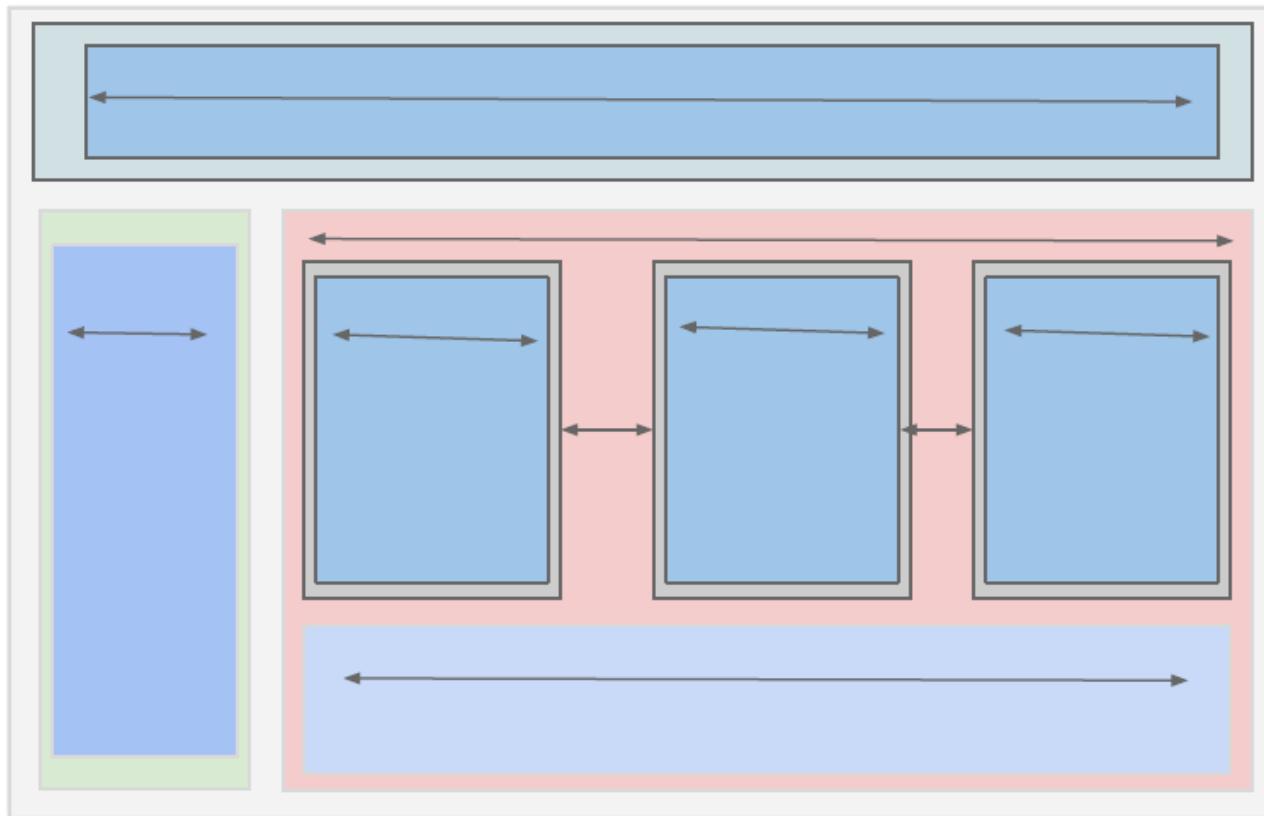


7. Box-sizing

- Dimensiones por defecto
 - Según el modelo de caja, los elementos al ser representados por el navegador ocupan el siguiente espacio:
 - **Altura:** *altura* del contenido + *padding* + *borde*
 - **Anchura:** *anchura* del contenido + *padding* + *borde*
 - En diseños complejos, para cuadrar todos los elementos HTML se hace necesario calcular manualmente sus dimensiones: sumar todos los espacios de todas las cajas, añadir los márgenes, etc.
 - Trabajar con las dimensiones por defecto a la hora de maquetar es una **tarea complicada**

7. Box-sizing

- Dimensiones por defecto





7. Box-sizing

- **box-sizing: border-box**
 - Una solución para maquetar más fácilmente es establecer la propiedad **box-sizing** con el valor **border-box**
 - De esta manera no tendremos que calcular con bordes y paddings, facilitándonos la tarea
 - El tamaño que demos al elemento será la suma de todo
 - Para ello, añadiremos en nuestro archivo CSS:

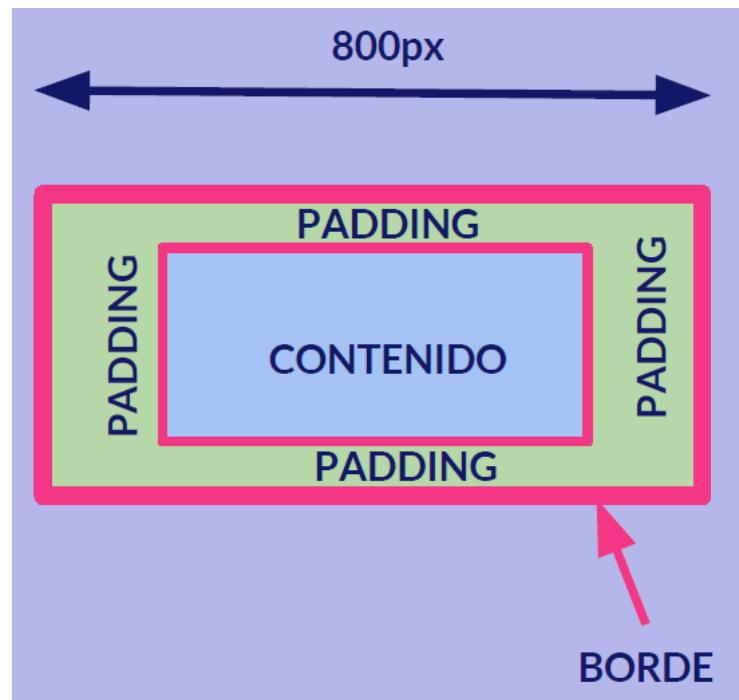
```
* {  
    -webkit-box-sizing: border-box;  
    -moz-box-sizing: border-box;  
    box-sizing: border-box;  
}
```



7. Box-sizing

- box-sizing: border-box
 - Ya estaremos incluyendo padding y border en la dimensión de la caja

```
div {  
    width: 800px;  
}
```





7. Box-sizing

- Otras valores de box-sizing
 - **content-box**: solo dimensionamos el contenido, habría que sumar todos los valores para maquetar (valor por defecto)
 - **padding-box**: no tiene en cuenta el borde pero sí el padding y el contenido (no soportado por todos los navegadores)

```
.padding {  
  ▲ box-sizing: padding-box;  
}
```



8. Elementos flotantes

- Propiedad float
 - **float** coloca un elemento a la izquierda o a la derecha de su elemento contenedor
 - Los elementos que lo rodean “fluirán” a su alrededor
 - Generalmente se utiliza 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?





8. 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



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



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



 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?





8. Elementos flotantes

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

{ 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?

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



A red arrow points from the bottom of the first text block to the bottom of the second text block.



8. Elementos flotantes

- Problema: clearfix hack
 - Posible problema al posicionar:

 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?

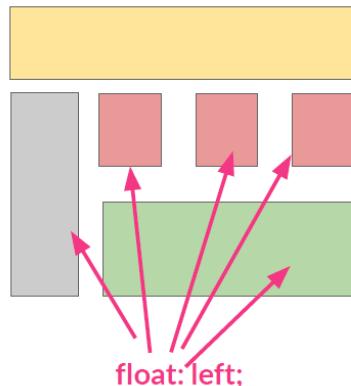


- El contenedor solo tiene en cuenta la altura del párrafo, no la de la imagen (pierde la referencia del elemento flotante)
- Solución (propiedades aplicadas al elemento contenedor):

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

8. Elementos flotantes

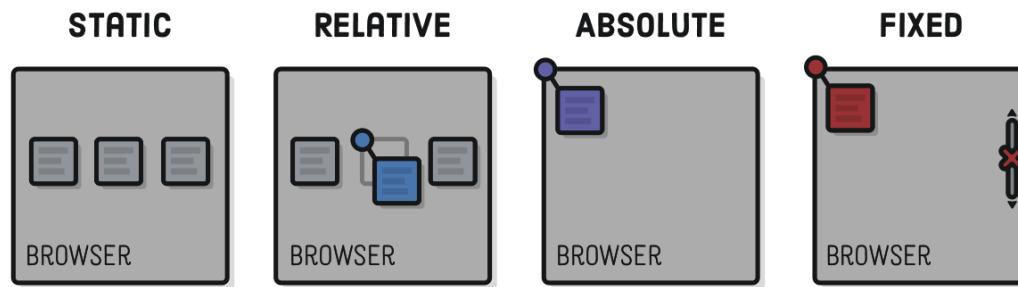
- Usando float para maquetar
 - Aunque actualmente hay mejores opciones, tradicionalmente se han usado elementos flotantes para crear estructuras
 - ¿Cómo?
 - Flotando los elementos según necesitemos -div, nav, header...- (generalmente a la izquierda)
 - Dándoles las dimensiones adecuadas





9. Posicionamiento de elementos

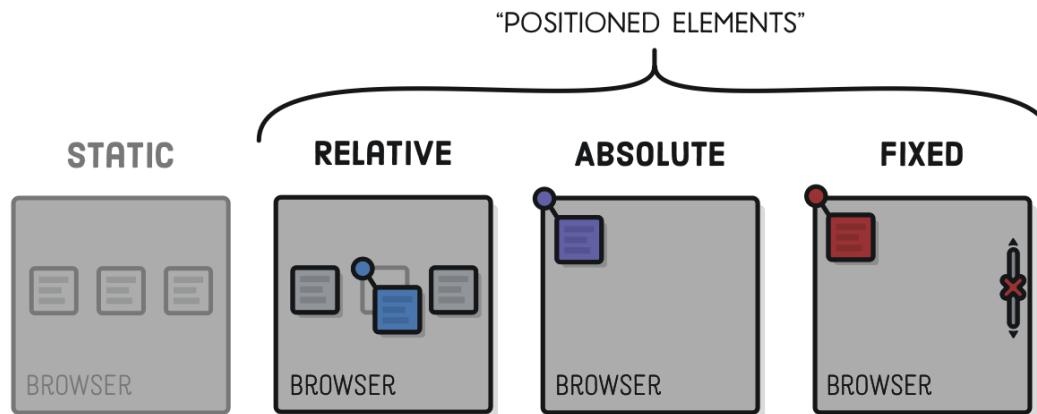
- Propiedad position
 - Es posible modificar el flujo normal de HTML con la propiedad **position**
 - Valores: **static**, **relative**, **absolute**, **fixed**, **sticky**
 - Estos valores están relacionados con las propiedades CSS **top**, **bottom**, **left** y **right** (desplazamientos conforme a un punto de referencia concreto) y **z-index** (capas)





9. Posicionamiento de elementos

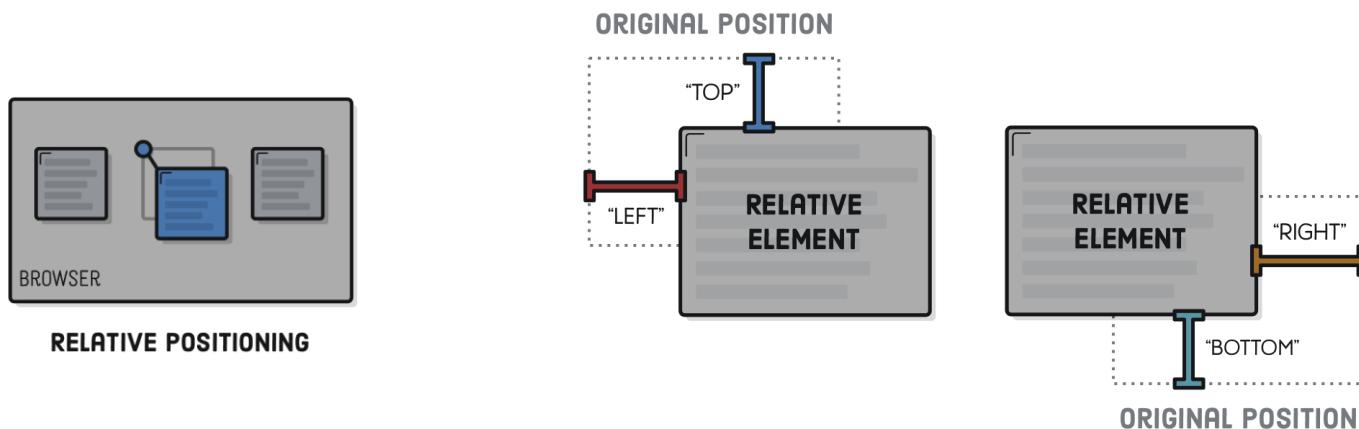
- Valores de position
 - **static** (valor por defecto)
 - El elemento sigue el flujo que le corresponde
 - Aunque tenga valores en *top*, *bottom*, *left*, *right* o *z-index* NO se aplican





9. Posicionamiento de elementos

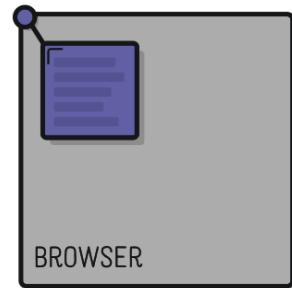
- Valores de position
 - **relative**
 - Como *static* pero SÍ atiende a los desplazamientos *top*, *bottom*, *left*, *right* o *z-index* a partir de la posición que le corresponde según el flujo normal



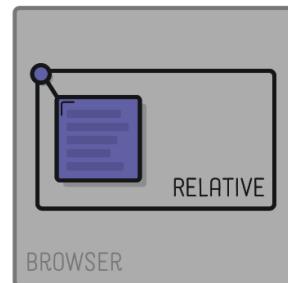


9. Posicionamiento de elementos

- Valores de position
 - **absolute**
 - El elemento se coloca en el sitio que se le indica con *top*, *bottom*, *left*, *right* o *z-index*, al margen del flujo normal de la página
 - Toma como referencia la ventana del navegador o la primera etiqueta padre que tenga *position:relative*



ABSOLUTE POSITIONING

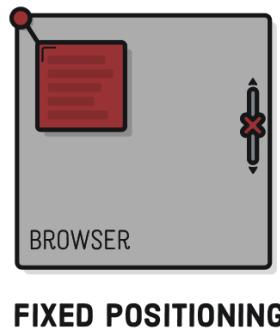


RELATIVELY ABSOLUTE POSITIONING



9. Posicionamiento de elementos

- Valores de position
 - **fixed**
 - El elemento sale del flujo y no atiende al scroll, su posición permanece fija al desplazarnos por la página
 - Se posiciona siempre en base a la ventana del navegador



FIXED POSITIONING

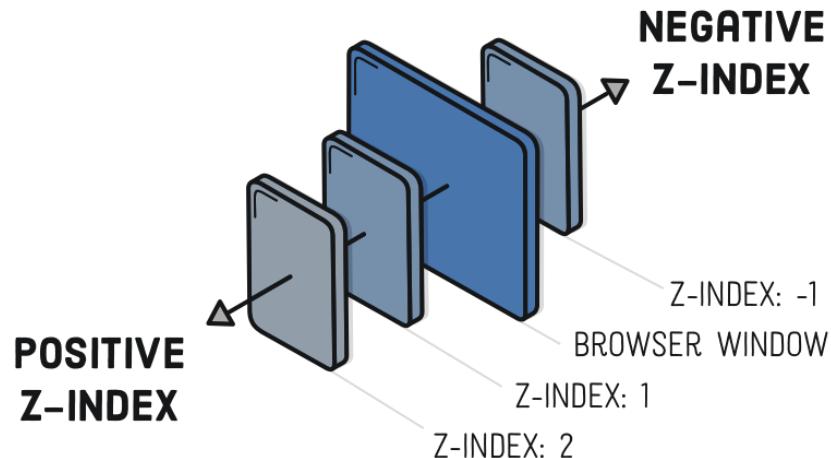


9. Posicionamiento de elementos

- Valores de position
 - **sticky**
 - Se comporta como *relative* hasta llegar a una posición de scroll determinada y a partir de entonces pasa a *fixed*
 - Necesario indicar un valor en top

9. Posicionamiento de elementos

- Propiedad z-index
 - Con **z-index** podemos establecer capas decidiendo el orden de solapamiento de los elementos
 - Se mostrará arriba aquel elemento con mayor valor z-index





10. Columnas

- División del contenido en columnas
 - CSS permiten distribuir el contenido en columnas
 - Propiedades:
 - **column-count**: número de columnas del contenedor
 - **column-width**: ancho de las columnas (el navegador calculará cuántas columnas caben con ese ancho)
 - **column-gap**: distancia entre columnas
 - **column-rule**: similar a borde; permite especificar el estilo, color y anchura de la línea que separa las columnas



10. Columnas

- División del contenido en columnas
 - Propiedades (continuación)
 - **column-span**: valores *all* o *none*; indica si el elemento que estará dentro del contenedor donde hemos especificado que habrá columnas, sigue el flujo en columnas o no
 - **column-fill**: cómo se rellenan las columnas; el contenedor debe tener altura. Valores: *auto* o *balance* (todas las columnas la misma altura)
 - **break-inside**: *void* si queremos que el elemento no quede roto de una columna a otra



11. Centrado del layout

- Centrar un elemento
 - Cuando hablamos de centrar un elemento nos referimos siempre a un contexto
 - El contexto de referencia siempre es su **etiqueta padre** o su **etiqueta contenedora**
 - Suele ser una tarea algo compleja inicialmente usando CSS tradicional (flex lo facilita mucho)
 - Se distingue **centrado horizontal** (más habitual) y **centrado vertical**



11. Centrado del layout

- Centrado horizontal
 - Elementos **inline**:
 - ✓ **text-align: center;** (al contenedor padre)
 - Elementos **block**:
 - ✓ **margin: X auto;** al elemento que queremos centrar
 - ✓ El elemento debe tener anchura
 - Varios elementos en bloque dentro de un contenedor:
 - ✓ **text-align: center;** (al padre)
 - ✓ **display: inline-block;** a los elementos a centrar y anchura



11. Centrado del layout

- Centrado vertical
 - Elementos **inline**
 - Con el mismo **padding** arriba y abajo
 - **vertical-align: middle** si estamos dentro de una celda de una tabla o lo estamos simulando con la propiedad **display** (el parent debe tener una altura fija)



11. Centrado del layout

- Centrado vertical
 - Elementos **block**
 - Cuando conocemos la altura del elemento (por ejemplo 150px)

```
css {  
    .contenedor {  
        position: relative;  
    }  
  
    .elemento_a_centrar {  
        height: 150px;  
        margin-top: -75px; /* La mitad de la altura */  
        position: absolute;  
        top: 50%;  
    }  
}
```



11. Centrado del layout

- Centrado vertical
 - Elementos **block**
 - Cuando desconocemos conocemos la altura del elemento

```
.contenedor {  
    position: relative;  
}  
.elemento_a_centrar {  
    position: absolute;  
    top: 50%;  
    transform: translateY(-50%);  
}
```



11. Centrado del layout

- Centrado con Flexbox
 - Más fácil
 - Centrado en ambos ejes

```
.parent {  
    display: flex  
    align-items: center  
    justify-content: center  
}
```



12. 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 # o con valores decimales (0 a 255)

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

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



12. 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);  
}
```

- También podemos utilizar la propiedad *opacity* para aplicar transparencias



12. Colores y transparencia

- Color HSL

- HSL (*Hue, Saturation, Lightness*): color, saturación, luz
- El tono (primer valor) 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;  
}
```

- Admite transparencia mediante canal alfa



13. Sombras

- Sombras
 - En CSS3 se pueden aplicar sombreados tanto a **texto** (text-shadow) como a **cajas** (box-shadow)

CSS: Shadows box-shadow

- Formato:

```
box-shadow: 2px 2px 1px 1px #808080;
```

h-offset: tamaño sombra aplicada a la derecha

v-offset: sombra que se proyecta hacia abajo

blur: difuminado (opc.)

spread: propagación, cantidad de sombra (opc., solo aplica a box-shadow)

color sombra



13. Sombras

- Sombras
 - Si se necesitan sombras arriba y a la izquierda, hay que aplicar valores negativos
 - También es posible aplicar **más de una sombra**, poniendo varios valores separados por comas

```
text-shadow: 3px 3px 1px #ccc, 3px 4px 1px #3f1234;
```

CSS: Shadows



14. Gradientes

- Gradientes CSS (degradados)
 - Es una transición suave entre dos o más colores especificados
 - Usando CSS para la creación de gradientes evitamos utilizar imágenes para estos efectos, lo que reduce el tiempo de descarga y el uso de ancho de banda
 - Tipos: lineales y radiales; se pueden repetir





14. Gradientes

- Gradientes lineales



```
selector {  
    background-image: linear-gradient(to right,  
        red , blue);  
}
```



14. Gradientes

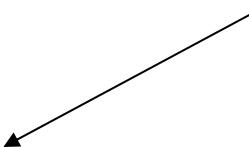
- Gradientes lineales
 - Formato:

```
background-image: linear-gradient (90deg, #ccc 0%,  
#121212 50%, #fff 100%);
```



Color final

Color intermedio al que cambia el gradiente; puede crearse lista con varias paradas para ir cambiando de gradientes; se puede omitir



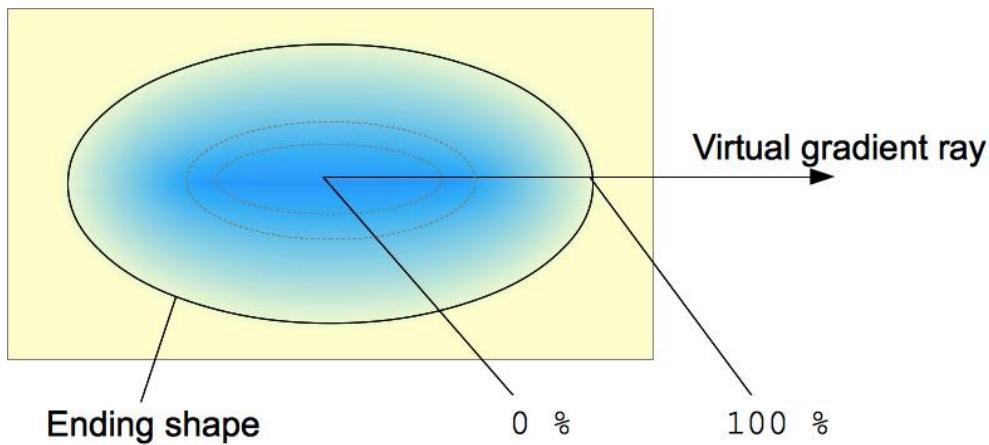
Ángulo de salida en grados que toma el gradiente; si se omite, lo hace en vertical; otra forma: “**to left**”, “**to right**”; Para diagonales: “**to bottom right**”, **to top right**”, ...

Color de inicio del gradiente y punto de inicio; si se usa un valor negativo, el color se inicia antes de su aparición

14. Gradientes

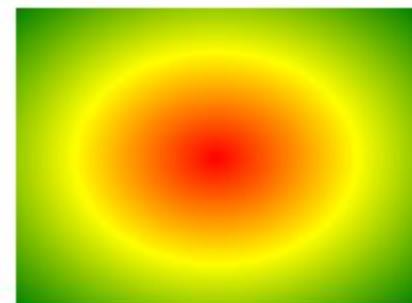
- Gradientes radiales
 - Formato:

`background-image: radial-gradient (forma, posición,
color1, ... , último_color);`



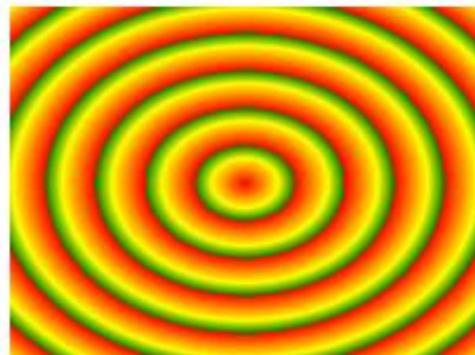
Forma: **circle** o **ellipse** (por defecto)

Posición: **center** por defecto



14. Gradientes

- Repetición de gradientes



- background-image: repeating-linear-gradient (red, yellow 10%, green 15%)
- background-image: repeating-radial-gradient (red, yellow 10%, green 15%);



15. Variables CSS

- Variables CSS (custom properties)
 - 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);
```



15. 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 **sobreescibir** 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



16. 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



16. 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



17. 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
 - Organizar las reglas poniendo las que tengan relación juntas
 - Documentar los ficheros CSS
 - Uso de preprocesadores CSS en proyectos grandes



17. Recomendaciones uso de CSS

- Recomendaciones maquetación
 - Esquema previo
 - ✓ “A mano”
 - ✓ Software: *MockFlow*, *NinjaMock*, *Wireframe.cc*, ...
 - De lo grande a lo pequeño
 - ✓ No colocar lo que hay dentro de una zona del diseño hasta que no se ha colocado el contenedor principal (contenedor padre)
 - Uso de las ayudas disponibles en navegadores
 - Probar en distintos navegadores



*. 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>
 - Sitio web <https://internetingishard.com/html-and-css/>