PCW

PROGRAMACIÓN DEL CLIENTE WEB

Tema 03 - CSS


Dept. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial


Universitat d'Alacant
Universidad de Alicante

CSS

Cascading Style Sheets

CSS: Cascading Style Sheets

- Introducción
- Selectores CSS
- Valores y variables CSS
- Propiedades de texto, fondo y bordes
- Web fonts e Icon fonts
- Media queries
- CSS Flexbox
- CSS Grid Layout
- Gradientes, transiciones y transformaciones
- Animaciones CSS

Introducción

Introducción

¿Qué es CSS?

Es el lenguaje utilizado para especificar el estilo y diseño con el que se muestran las páginas web.

A diferencia de las dos primeras versiones de CSS (CSS1 y CSS2), que son estándares monolíticos, la tercera versión de CSS (CSS3) introduce la modularidad y pone fin a las versiones de CSS:

- Cada módulo se encarga de una parte del estilo y se puede publicar de forma independiente una vez finalizado.
- Diferentes grupos de personas pueden estar trabajando en diferentes módulos avanzando de forma paralela: desarrollo más rápido.

No todos los navegadores admiten todas las características de CSS, ni las interpretan exactamente igual

Introducción

¿Qué es CSS?

Bordes
border-color
border-image
border-radius
box-shadow

Fondos
background-origin
background-clip
background-size
Multiple background

Interfaz de usuario
box-sizing
resize
outline
nav-up, nav-right, nav-down, nav-left

Colores
Colores HSL
Colores HSLA
Opacidad
Colores RGBA

Efectos de texto
text-shadow
text-overflow
word-wrap

Selectores
Selectores de atributos

Modelo de caja básico
overflow-x, overflow-y

Contenido generado
Contenido

Otros módulos
Media queries, transiciones CSS, diseño multi-columna, web fonts, flexbox, speech

Introducción

¿Qué es CSS?

Prefijos propietarios

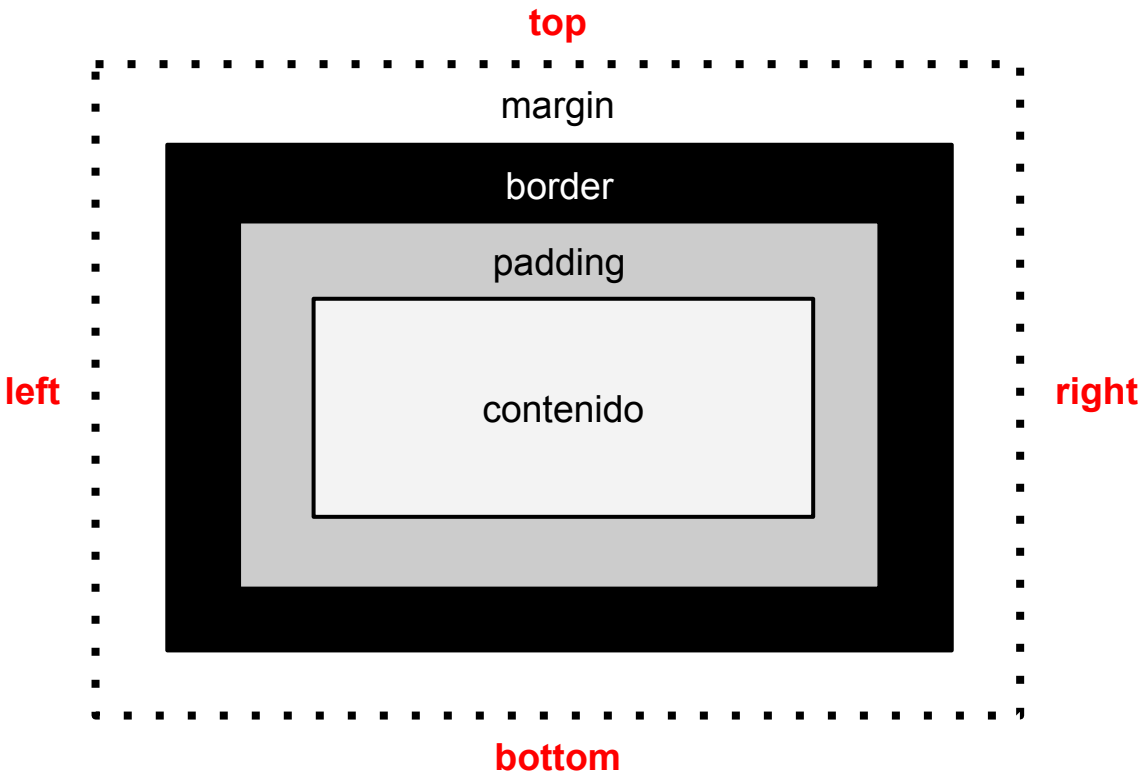
- Son necesarios para hacer funcionar algunas propiedades estándar, que todavía no están implementadas, en los distintos navegadores.
- Utilizan la implementación propietaria de la propiedad en lugar de la estándar.
- Algunos de los prefijos más conocidos y utilizados son los siguientes:

Prefijo	Organización
`-ms-`	Internet Explorer, Microsoft Edge
`-moz-`	Firefox
`-o-`	Versiones de Opera anteriores a WebKit
`-webkit-`	Chrome, Safari, versiones más nuevas de Opera, casi todos los navegadores iOS, incluyendo Firefox para iOS; básicamente, cualquier navegador basado en WebKit

Introducción

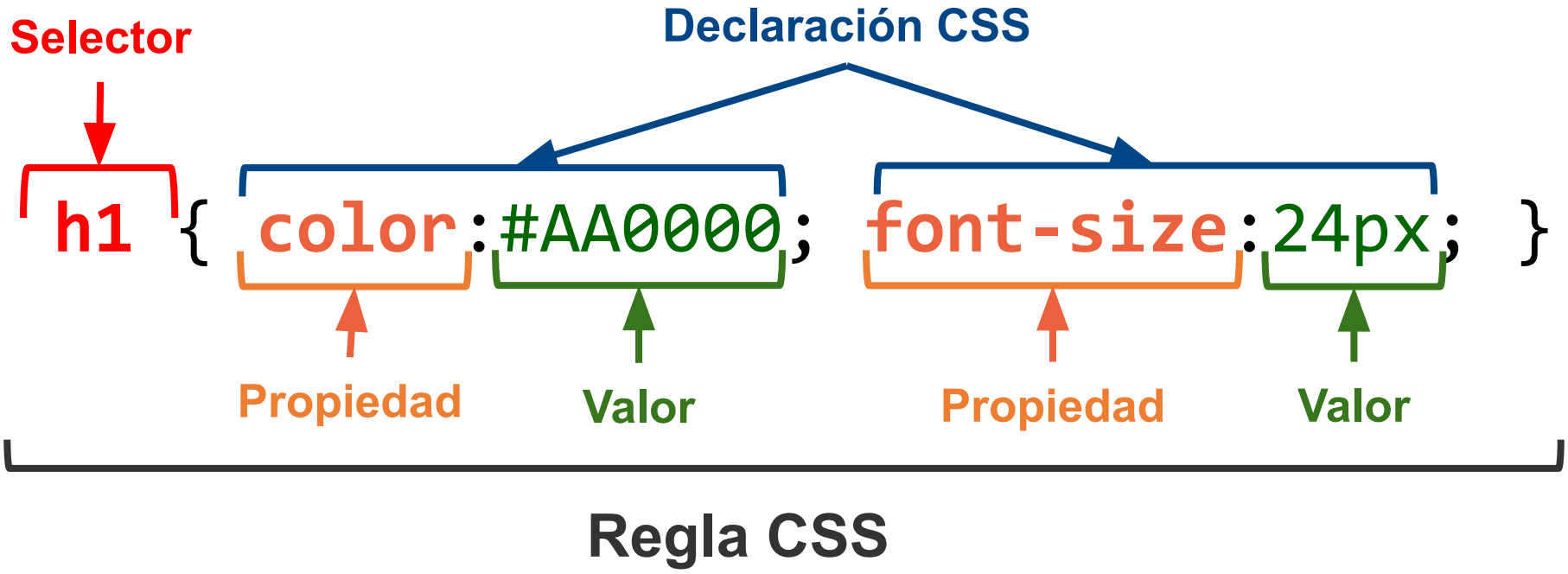
Modelo de caja

El navegador representa cada elemento HTML del documento como una caja rectangular cuyo tamaño, posición y demás propiedades (color, fondo, borde, etc) se puede configurar mediante propiedades CSS.



Introducción

Sintaxis de una regla CSS



Por legibilidad, las **declaraciones CSS** (pares `propiedad:valor`) se suelen poner una debajo de otra

```
h1 {
  color:#AA0000;
  font-size:24px;
}
```

Introducción

Cómo se utiliza CSS en un documento HTML

- Las reglas CSS se declaran en un documento de texto con extensión "`.css`".
- Se enlaza desde el documento HTML, que hará uso de dichas reglas, mediante el elemento **`<link>`**. Este elemento se incluye dentro del elemento `<head>` del documento HTML, de la siguiente manera:

```
<!DOCTYPE html>
  <html lang="es">
    <head>
      <link rel="stylesheet" href="estilo.css">
      …
    </head>
    …
```

El atributo **`href`** se utiliza para indicar la ubicación del archivo css

El atributo **`rel`** se utiliza para indicar que el archivo contiene reglas css

Nota: Aunque se puede declarar el estilo CSS en el mismo documento HTML, esta práctica no se recomienda.

Selectores CSS

Selectores CSS

Los **selectores** son patrones que buscan coincidencias en los elementos de una estructura en forma de árbol. Esto hace que sea una de las tecnologías utilizadas para seleccionar nodos en un documento HTML.

- Un selector es una expresión cuya estructura puede emplearse como una condición (por ejemplo, una regla CSS) que determina qué elementos coinciden con el selector en el árbol del documento, o fragmento del mismo.
- Los selectores pueden ser desde simples nombres de elementos, hasta ricas representaciones contextuales.
- Los selectores son sensibles a mayúsculas/minúsculas, salvo para los nombres de los elementos.

expresión * elemento ⟶ **booleano**

Selectores CSS

Selectores de elementos

- **Selectores de tipo (*tag name*)**. Seleccionan elementos en función del tipo. Por ejemplo, p seleccionaría todos los elementos **<p>** del documento.
- **Selector universal:** *. Selecciona todos los elementos del documento.

Selectores de atributos

Los selectores de atributos permiten la representación de los atributos de los elementos, por tanto, se pueden utilizar junto a los selectores de elementos para hacer la selección más fina.

- **Por valor y presencia del atributo**
  - **[atr]**. Representa cualquier elemento que tenga el atributo **atr**, independientemente del valor del mismo.
  - **[atr=*val*]**. Representa un elemento con el atributo **atr** cuyo valor es exactamente *val*.

Selectores CSS

Selectores de atributos (*y //*)

- **Por valor y presencia del atributo (*y //*)**
  - **[atr~=*val*]**. Representa un elemento con el atributo **atr** cuyo valor es una lista de palabras separadas por espacios en blanco y una de ellas es exactamente igual a *val*.
  - **[atr|=*val*]**. Representa un elemento con el atributo **atr** cuyo valor es exactamente *val*, o bien comienza con *val* seguido inmediatamente por el carácter "-".
- **Por presencia de subcadenas en el valor del atributo**
  - **[atr^=*val*]**. Representa un elemento que tiene el atributo **atr** y cuyo valor empieza por *val*. Si *val* es un valor vacío, no representa nada.
  - **[atr$=val]**. Representa al elemento que tiene el atributo **atr** y cuyo valor finaliza con el sufijo *val*. Si *val* es un valor vacío, no representa nada.

Selectores CSS

Selectores de atributos (*y ///*)

- **Por presencia de subcadenas en el valor del atributo (*y //*)**
  - **`[atr*=val]`.** Representa al elemento que tiene el atributo **`atr`** y cuyo valor contiene, al menos una vez, la subcadena *`val`*. Si *`val`* es un valor vacío, no representa nada.

Selectores de clase

Seleccionan elementos en base a su atributo de clase. Su sintaxis es un punto ("`.`") seguido inmediatamente por un identificador de clase. Por ejemplo, `.foto` seleccionaría todos los elementos que tuvieran el valor *foto* en su atributo `class`. Es equivalente a **`[class~="foto"]`**

Selectores de ID.

Seleccionan elementos en base al atributo **`id`**. Su sintaxis es un símbolo `#` seguido inmediatamente por el valor del ID. Por ejemplo, `#art001` seleccionaría al elemento con el valor *art001* en su atributo **`id`**.

Selectores CSS

Ejemplos de selectores:

- Seleccionar todos los elementos: `*`
- Seleccionar todos los elementos `<p>`: `p`
- Seleccionar todos los elementos con el atributo `data-valor`: `[data-valor]`
- Seleccionar todos los elementos `<span>` con el atributo `data-valor`: `span[data-valor]`
- Seleccionar todos los elementos `<span>` cuyo atributo `data-valor`: comienza por `idx`: `span[data-valor^="idx"]`
- Seleccionar todos los elementos de la clase `mostrar`: `.mostrar`
- Seleccionar todos los elementos `<p>` de la clase `mostrar`: `p.mostrar`
- Seleccionar todos los elementos `<p>` de la clase `mostrar` que, además, tienen el atributo `data-valor`: `p.mostrar[data-valor]`
- Seleccionar el elemento con el ID `principal`: `#principal`

Selectores CSS

Operadores de combinación (*combinators*)

Proporcionan una manera de describir las relaciones entre los elementos, combinándolos y formando la expresión que da lugar a la regla CSS.

Hay cuatro operadores de combinación:

- **Descendiente (*descendant combinator*)**. El operador se representa mediante un espacio en blanco y permite seleccionar elementos que son descendientes de otros dentro del árbol del documento.

  Ejemplos:

  - `h2 em`: Este selector representa todos los elementos **** que son descendientes de un elemento **<h2>**, independientemente del nivel.
  - `div * p`: Este selector representa todos los elementos **<p>** que son nietos o descendientes posteriores de un elemento ** **.

Selectores CSS

Operadores de combinación (*combinators*)

- **Hijo (*child combinator*).** El operador se representa mediante el carácter `>` y permite seleccionar elementos que están inmediatamente contenidos dentro de otros elementos.

  Ejemplos:

  - `section > h2`: Este selector representa a todos los elementos `<h2>` que sean descendientes de primer nivel (hijo) de un elemento `<section>`.
  - `section ul > li a`: Este selector representa a todos los elementos `<a>` que son descendientes de un elemento `<li>` que es hijo de un elemento `<ul>` que, a su vez, es descendiente de un elemento `<section>`.

Selectores CSS

Operadores de combinación (*combinators*)

- **Hermano (*sibling combinator*).** Permite relacionar elementos que tienen el mismo padre en el árbol del documento. Hay dos tipos de operador *hermano*:
  - Siguiente. Se representa mediante el carácter **+** que separa dos secuencias de selectores simples. El elemento de la izquierda del operador es un *hermano* que precede inmediatamente al de la derecha en el árbol del documento.

```
h1 + h2
```

  - Subsiguiente. Se representa mediante el carácter **~** que separa dos secuencias de selectores simples. El elemento de la izquierda del operador es un *hermano* que precede, no necesariamente de forma inmediata, al de la derecha en el árbol del documento.

```
h1 ~ p
```

Selectores CSS

Pseudo-elementos y pseudo-clases

- **Pseudo-elementos**

  Son elementos abstractos que representan porciones del árbol del documento y a los que se puede aplicar estilo mediante CSS.

  El selector de un pseudo-elemento se forma mediante dos caracteres dos puntos, "**::**", seguido del nombre del pseudo-elemento.

  ```
  ::pseudo_elemento
  ```

  Por compatibilidad con documentos escritos utilizando versiones anteriores de CSS, se acepta la notación únicamente un carácter "**:**", en lugar de dos, para los pseudo-elementos `::before`, `::after`, `::first-letter` y `::first-line`.

Selectores CSS

Pseudo-elementos y pseudo-clases

- **Pseudo-elementos. Clasificación.**

  Pseudo-elementos Tipográficos:

  - `::first-line`. Representa el contenido de la primera línea formateada del elemento que la contiene.
  - `::first-letter`. Representa la primera letra de la primera línea formateada, del elemento que la contiene, siempre y cuando no esté precedida en la línea por ningún otro contenido como pueden ser imágenes o tablas *inline*.

Selectores CSS

Pseudo-elementos y pseudo-clases

- **Pseudo-elementos. Clasificación.**

  Pseudo-elementos de Resaltado:

  - **`::selection`**, **`::inactive-selection`**. Representan la porción seleccionada del documento html. Cuando la ventana del documento html deja de estar activa, la porción seleccionada pasa a ser el pseudo-elemento **`::inactive-selection`**.
  - **`::spelling-error`**. Representa una porción de texto que ha sido marcada por el navegador como mal escrita por tener errores tipográficos/ortográficos.
  - **`::grammar-error`**. Representa la porción de texto que ha sido marcada por el navegador como mal escrita gramáticamente.

Selectores CSS

Pseudo-elementos y pseudo-clases

- **Pseudo-elementos. Clasificación.**

  Pseudo-elementos *Tree-abiding*: Estos elementos existen para todos los elementos modelados como caja en el árbol del documento.

  - **`::before`**. Representa el pseudo-elemento hijo que está inmediatamente antes del contenido del elemento padre que lo genera.
  - **`::after`**. Representa el pseudo-elemento hijo que está inmediatamente después del contenido del elemento padre que lo genera.
  - **`::marker`**. Representa el marcador que se genera automáticamente con cada elemento de una lista html.
  - **`::placeholder`**. Representa el texto ejemplo que aparece en un elemento html de entrada de texto.

Selectores CSS

Pseudo-elementos y pseudo-clases

- **Pseudo-clases**

  Permiten la selección basada en la información que queda fuera del árbol del documento y que no se puede expresar mediante selectores simples.

  Una pseudo-clase se forma mediante un único carácter dos puntos, "**:**", seguido del nombre de la pseudo-clase y opcionalmente un valor entre paréntesis.

```
:pseudo_clase(n)
```

  - Las pseudo-clases se permiten en todas las secuencias de selectores simples contenidos en un selector.
  - Los nombres de las pseudo-clases son independientes a mayúsculas/minúsculas.
  - Algunas pseudo-clases son mutuamente excluyentes, mientras que otras se pueden aplicar simultáneamente a un mismo elemento.
  - Las pseudo-clases pueden ser dinámicas, es decir, un elemento puede adquirirla o perderla en función de la interacción del usuario.

Selectores CSS

Pseudo-elementos y pseudo-clases

- **Pseudo-clases**
  - Pseudo-clase de idioma
    - **`:lang`.** Permite seleccionar los elementos de un documento a los que se les ha aplicado el atributo de idioma.
  - Pseudo-clases de ubicación**.** Permiten seleccionar los elementos de un documento que son puntos de destino de enlaces dentro del mismo documento. El estilo se aplicará a los elementos cuando se pinche en el enlace que lleva a ellos.
    - **`:any-link`**. Representa cualquier elemento de tipo hiperenlace.
    - **`:link`**, **`:visited`**. Representa los elementos de tipo hiperenlace no visitados o ya visitados, respectivamente.
    - **`:local-link`**. Representa los elementos de tipo hiperenlace cuyo destino está dentro del mismo sitio web.
    - **`:target`**. Representa los elementos, dentro de un documento html, que son destino final de un hiperenlace.

Selectores CSS

Pseudo-elementos y pseudo-clases

- **Pseudo-clases**
  - Pseudo-clases de acción de usuario**.** Permiten seleccionar los elementos de un documento en base a la interacción que el usuario haga sobre ellos.
    - **`:hover`**. Se aplica a un elemento designado por el usuario mediante un puntero, por ejemplo el ratón, aunque no necesariamente lo active.
    - **`:active`**. Se aplica al elemento que está siendo activado (por ejemplo, haciendo clic con el ratón sobre él) por el usuario y sólo durante ese momento.
    - **`:focus`**. Se aplica al elemento que tiene el foco (acepta entrada por teclado, eventos de ratón o cualquier otra forma de entrada).

Selectores CSS

Pseudo-elementos y pseudo-clases

- **Pseudo-clases**
  - Pseudo-clases de elementos de entrada**.** Permiten seleccionar los elementos que permiten recoger entrada del usuario como, por ejemplo, los elementos `<input>` de HTML.
    - **`:enabled`**, **`:disabled`**. Representa elementos de interfaz de usuario que están habilitados y deshabilitados, respectivamente.
    - **`:placeholder-shown`**. Representa elementos que tienen el atributo y se está mostrado su valor.
    - **`:read-only`**, **`:read-write`**. Representa elementos de interfaz de usuario que no puede alterar el usuario y que sí puede alterar (como, por ejemplo, un elemento con el atributo *contenteditable*), respectivamente.
    - **`:default`**. Representa elementos seleccionados por defecto como, por ejemplo, el botón *submit* de un formulario.

Selectores CSS

Pseudo-elementos y pseudo-clases

- **Pseudo-clases**
  - Pseudo-clases de elementos de entrada (*y II*).
    - **`:checked`**. Representa elementos `<input>` de tipo *radio* y *checkbox* seleccionados por el usuario.
    - **`:indeterminate`**. Representa elementos cuyo valor es indeterminado, es decir, por ejemplo un grupo de botones radio en el que no hay ninguno seleccionado.
    - **`:valid`**, **`:invalid`**. Representa elementos cuyo contenido es válido o no, respectivamente, en función de la semántica de validación de datos que se les haya aplicado.
    - **`:in-range`**, **`:out-of-range`**. Representa elementos para los que se establece un rango y su valor está dentro o fuera, respectivamente. Por ejemplo, un elemento `<input>` de tipo *number* para el que se ha establecido un mínimo y un máximo.

Selectores CSS

Pseudo-elementos y pseudo-clases

- **Pseudo-clases**
  - Pseudo-clases de elementos de entrada (*y II*).
    - **`:required`**, **`:optional`**. Representan elementos de formulario para los que es obligatorio o no, respectivamente, introducir un valor antes de la acción *submit*.
  - Pseudo-clases estructurales. Permiten la selección de elementos basada en información extra del árbol del documento, pero que no se puede representar de otra manera.
    - **`:root`**. Representa el elemento raíz del documento: `<html>`.
    - **`:empty`**. Representa el elemento que no tiene hijos.
    - **`:nth-child(`*a*`n+`*b*`)`**. Representa al elemento que ocupa la posición *an+b* en la lista de hijos de su nodo padre, siendo *a* y *b* valores enteros y *n* representa a los números enteros >= 0. También admite los valores *odd* y *even*, que equivalen a impar (*2n+1*) y par (*2n*), respectivamente.

Selectores CSS

Pseudo-elementos y pseudo-clases

- **Pseudo-clases**
  - Pseudo-clases estructurales (*y II*).
    - **`:nth-last-child(`*an*`+`*b*`)`**. Representa al elemento que ocupa la posición *an*+*b* en la lista de hijos de su nodo padre, empezando a contar desde el último hijo al primero. Admite los mismos valores que `:nth-child()`.
    - **`:first-child`**. Representa el primer hijo de un elemento. Es lo mismo que `:nth-child(1)`.
    - **`:last-child`**. Representa el último hijo de un elemento. Es lo mismo que `:nth-last-child(1)`.
    - **`:only-child`**. Representa el elemento que no tiene elementos hermanos. Es lo mismo que `:first-child:last-child` o `:nth-child(1):nth-last-child(1)`, pero con una menor especificidad.

Selectores CSS

Pseudo-elementos y pseudo-clases

- **Pseudo-clases**
  - Pseudo-clases estructurales (*y III*).
    - **`:nth-of-type(`*a*`n+`*b*`)`**. Representa al elemento que ocupa la posición *an*+*b* en la lista de hijos del mismo tipo (indicado delante de ":") y con el mismo nodo padre. Admite los mismos valores que `:nth-child()`.
    - **`:nth-last-of-type(`*a*`n+`*b*`)`**. Representa al elemento que ocupa la posición *an*+*b* en la lista de hijos del mismo tipo y con el mismo nodo padre, empezando a contar desde el último hijo al primero. Admite los mismos valores que `:nth-last-child()`.
    - **`:first-of-type`**, **`:last-of-type`**. Representan al primer y último elemento, respectivamente, de la lista de nodos seleccionados del tipo indicado delante de ":". Equivalen a `:nth-of-type(1)` y `:nth-last-of-type(1)`, respectivamente.
    - **`:only-of-type`**. Representa el único hijo de un elemento, cuyo tipo es el indicado delante de ":". Es igual a `:first-of-type:last-of-type`.

Selectores CSS

Pseudo-elementos y pseudo-clases

- **Pseudo-clases**
  - Pseudo-clases de combinaciones lógicas.
    - **`:not(`***`selector`***`)`**. Es la pseudo-clase negación. Es una notación funcional que toma un selector simple como argumento. Selecciona el elemento que no está representado por dicho argumento.
    - **`:is(`***`lista_de_selectores`***`)`**. Selecciona los nodos que cumplan alguno de los selectores, separados por comas, que se indican en *`lista_de_selectores`*.
    - **`:where(`***`lista_de_selectores`***`)`**. Se utiliza para eliminar la especificidad de las reglas asociadas a los selectores indicados en *`lista_de_selectores`*, separados por comas. De esta manera, reglas posteriores para el mismo elemento en el documento css, aunque tuvieran menos especificidad, se aplicarían.

Selectores CSS

Especificidad: prioridad de selectores en reglas CSS

- **CSS asigna pesos a los selectores** de las reglas de estilo en función de los elementos utilizados para construirlos.
- Tiene más prioridad (especificidad) el peso asignado que la posición en el fichero css.
- Si dos reglas CSS se aplican al mismo elemento, la regla con el selector con mayor especificidad es la que se aplicará.

Selectores CSS

Especificidad: prioridad de selectores en reglas CSS

Jerarquía (de mayor a menor):

1. Uso de **estilos *inline*** (uso del atributo *style* en el elemento HTML).

```
<h1 style="color: #fff;">
```

2. Uso de **ID**.
3. Uso de clases, atributos y pseudo-clases (`:hover`, `:focus`, etc).
4. Uso de elementos y pseudo-elementos (`::before`, `::after`, etc).

Selectores CSS

Especificidad: prioridad de selectores en reglas CSS

Cómo saber el peso (especificidad) de un selector

Se calcula sumando el valor de cada componente del selector, teniendo en cuenta la siguiente tabla:

Componente	Valor
Estilo *inline* (uso de atributo **`style`**)	**1000**
Cada ID utilizado	**100**
Cada nombre de clase, de atributo y de pseudo-clase utilizado	**10**
Cada nombre de elemento HTML y de pseudo-elemento utilizado	**1**

Ejemplo:

```
<section>
   <h2>Lorem ipsum</h2>
   <article>
      <h3>Aliquam dignissimos</h3>
...
```

Selector	Peso
section>article	**2**
article:first-of-type	**11**

Valores y variables CSS

Valores y variables CSS

Valores CSS

El módulo de unidades y valores de CSS3 describe los habituales valores y unidades que admiten las propiedades CSS.

Unidades de medida relativas

Unidad	Relativa a:
em	Valor de la propiedad ***font-size*** del elemento
ex	Valor de *x-height* del elemento
ch	Ancho del carácter 0 (ZERO, U+0030) en la fuente del elemento
rem	Valor de la propiedad ***font-size*** del elemento root
vw	1% del ancho de la ventana del navegador
vh	1% del alto de la ventana del navegador
vmin	1% de la dimensión más pequeña de la ventana
vmax	1% de la dimensión más grande de la ventana

Valores y variables CSS

Valores CSS

Unidades de medida absolutas

Unidad	Longitud	Equivalencia:
cm	centímetro	1cm = 96px/2.54
mm	milímetro	1mm = 1/10 * 1cm
Q	1 cuarto de milímetro	1Q = 1/40 * 1cm
in	pulgada	1in = 2.54cm = 96px
pc	pica	1pc = 1/6 * 1in
pt	punto	1pt = 1/72 * 1in
px	píxel	1px = 1/96 * 1in

Unidad	Ángulo
deg	grados (un círculo tiene 360 grados)
grad	gradianes (un círculo tiene 400 gradianes)
rad	radianes (un círculo tiene 2π radianes)
turn	vueltas (un círculo tiene una vuelta)

Unidad	Tiempo
s	segundos
ms	milisegundos
Hz	Veces por segundo
kHz	1000 Hz

Unidad	Resolución
dpi	puntos por pulgada (*dots per inch*)
dpcm	puntos por centímetro (*dots per centimeter*)
dppx	puntos por unidad ‘px’

Valores y variables CSS

Valores CSS

Notaciones funcionales

Son valores de propiedades que se pueden expresar de forma más compleja aplicando una serie de funciones especiales.

- ***calc(****expresión****)***. Permite usar como valores de propiedades expresiones matemáticas con sumas (+), restas (-), multiplicaciones (*) y divisiones (/). Se puede utilizar en cualquier propiedad de longitud, ángulo, tiempo o numérica.

  Ejemplo:

CSS

```
:root {
    font-size: calc(100vw / 40);
}
section {
    float: left; margin: 1em;
    border: solid 1px;
    width: calc(100%/3 - 2*1em - 2*1px);
}
.foo {
    background: url(top.png), url(bottom.png);
    background-repeat: no-repeat;
    background-position: calc(50% + 20px) calc(50% + 20px), 50% 50%;
}
```

Valores y variables CSS

Valores CSS

Notaciones funcionales (*y //*)

- ***attr(****nombre_atributo****)***. Devuelve el valor del atributo indicado del elemento al que se aplica. Funciona solo con textos.

  Ejemplo:

HTML

```
<span data-tooltip="Cascading Style Sheets">CSS</span>
```

CSS

```
[data-tooltip]::before {
  content: attr(data-tooltip);
}

[data-tooltip]:hover::before {
  visibility: visible;
  opacity: 1;
}
```

Valores y variables CSS

Variables CSS

(CSS Custom Properties for Cascading Variables)

Valores y variables CSS

Variables CSS

- Las ***CSS Custom Properties for Cascading Variables*** o variables CSS permiten definir propiedades CSS y asignarles un valor para poder ser utilizadas en el resto del documento CSS.
- Las variables CSS se heredan.
- Las variables CSS son *case-sensitive*.
- Para declarar una variable su nombre debe tener el prefijo: `--`

  Ejemplo:

```
html {
  --color-de-fondo: #223344;
}
```

Declarándola para el elemento `<html>`, la variable CSS podrá utilizarse para todos los elementos del documento html

Valores y variables CSS

Variables CSS

- Para poder utilizar el valor de la variable en el documento CSS se utiliza la función **`var()`**.

  Ejemplo:

```
h2{
  background-color: var(--color-de-fondo);
}
```

- Como método *fallback*, por si el navegador no soporta variables CSS, se puede hacer lo siguiente:

```
h2{
  background-color: #223344;
  background-color: var(--color-de-fondo);
}
```

Propiedades de texto, fondo y bordes

Propiedades de texto, fondo y bordes

Propiedades de texto

Propiedades de texto, fondo y bordes

Propiedades de texto

Se pueden clasificar en dos tipos:

- Las que **afectan a la fuente** que se aplica al texto: tamaño, color, etc.
- Las que **afectan al diseño** del texto: alineación, espacio entre líneas, etc.

Propiedades de texto, fondo y bordes

Propiedades de texto

Propiedades de la fuente

Algunas de las propiedades más utilizadas son:

- **color**. Permite cambiar el color de la fuente.
- **font-family**. Permite elegir la fuente a utilizar para mostrar el texto.
- **font-size**. Permite especificar el tamaño de la fuente.
- **font-style**. Permite poner (o quitar) el texto en cursiva. Los valores usados son `normal` o `italic`.
- **font-weight**. Permite poner la fuente en negrita. Admite distintos valores como `normal`, `bold`, `lighter`, `bolder` o valor numérico entre `100` y `900`.

Propiedades de texto, fondo y bordes

Propiedades de texto

Propiedades de la fuente

- **text-transform**. Permite, entre otras cosas, cambiar el texto entre mayúsculas y minúsculas. Los valores posibles son:
  - `none`: Evita cualquier transformación.
  - `uppercase`: Muestra todo el texto en mayúsculas.
  - `lowercase`: Muestra todo el texto en minúsculas.
  - `capitalize`: Muestra las palabras en minúsculas con su primera letra en mayúsculas.
- **text-decoration**. Permite aplicar "decoraciones" al texto. Se pueden aplicar varios valores al mismo tiempo. Los valores posibles son:
  - `none`: Elimina cualquier decoración que tuviera el texto.
  - `underline`: Subraya el texto.
  - `overline`: Añade al texto una línea superior.
  - `line-through`: "Tacha" el texto añadiendo una línea a mitad.

Propiedades de texto, fondo y bordes

Propiedades de texto

Propiedades del diseño

Algunas de las propiedades más utilizadas son:

- `text-align`. Permite alinear el texto horizontalmente en su contenedor. Los valores posibles son:
  - `left`: Alinea el texto a la izquierda.
  - `right`: Alinea el texto a la derecha.
  - `center`: Centra el texto.
  - `justify`: Justifica el texto.
- `line-height`. Permite establecer la altura de cada línea de texto.
- `letter-spacing`, `word-spacing`. Permiten establecer el espacio entre letras y palabras en el texto, respectivamente.
- `text-indent`. Permite especificar el espacio horizontal a dejar en la primera línea del texto.

Propiedades de texto, fondo y bordes

Propiedades de texto

Propiedades del diseño

- `white-space`. Permite especificar cómo se manejan los espacios en blanco "de más" (incluidos tabuladores y retornos de carro) que se encuentran en el texto. Los posibles valores son:
  - `normal`: Valor por defecto. Elimina espacios en blanco "de más", juntándolos en uno solo (a veces, en ninguno). Permite mostrar el texto en más de una línea.
  - `pre`: Mantiene los espacios en blanco "de más". El texto se muestra en una sola línea salvo que tenga retornos de carro que forzarán la ruptura.
  - `nowrap`: Igual que `normal`, pero muestra el texto en una única línea.
  - `pre-wrap`: Como `pre`, pero muestra el texto en varias líneas.
  - `pre-line`: Como `normal`, pero no elimina los retornos de carro.

Propiedades de texto, fondo y bordes

Propiedades de texto

Propiedades del diseño

- `white-space`

Ejemplo:

```
<p>Do you see any Teletubbies in
here?
Do you see a slender plastic tag
clipped to my shirt with my name
printed on it?</p>
```

```
p{
  border:1px solid #000;
  white-space: normal;
  width:200px;
}
```

Con **white-space: normal**:

Do you see any Teletubbies in here? Do you see a slender plastic tag clipped to my shirt with my name printed on it?

Con **white-space: pre-wrap**:

Do you see any Teletubbies in here?

Do you see a slender plastic tag clipped to my shirt with my name printed on it?

Con **white-space: pre-line**:

Do you see any Teletubbies in here? Do you see a slender plastic tag clipped to my shirt with my name printed on it?

Con **white-space: pre**:

Do you see any Teletubbies in here?
Do you see a

Con **white-space: nowrap**:

Do you see any Teletubbies in h

Propiedades de texto, fondo y bordes

Propiedades de texto

Propiedades del diseño

- `text-shadow`. Añade una o más sombras al texto. Si se especifican más de una sombra para el texto, se utiliza la coma "," como separador.

`text-shadow:` *despl_h* *despl_v* *desenfoque* *color*

Parámetros:

- *despl_h*: desplazamiento horizontal de la sombra.
- *despl_v*: desplazamiento vertical de la sombra.
- *desenfoque*: radio del desenfoque a aplicar. Optativo.
- *color*: color de la sombra. Optativo.

Notas:

- El desplazamiento horizontal/vertical admite valores negativos.
- Si no se indica el desenfoque, su valor por defecto es 0.
- El color de la sombra se puede indicar al principio o al final de la especificación.
- Si no se indica el color de la sombra se utilizará el color del texto.

Propiedades de texto, fondo y bordes

Propiedades de texto

Propiedades del diseño

- `text-shadow`

Ejemplo:

HTML

```
<p class='fuego'>Este es un
ejemplo de text-shadow</p>
```

CSS

```
p.fuego{
   background-color:black;
   color:#fff;
   padding:40px 0 10px;
   font-size:30px;
   text-align:center;
   text-shadow: #fff 0 0 4px,
                #FF3 0 -5px 4px,
                #FD3 2px -10px 6px,
                #F80 -2px -15px 11px,
                #F20 2px -25px 18px;
}
```

Propiedades de texto, fondo y bordes

Propiedades de texto

Propiedades del diseño

- `text-overflow`. Permite especificar cómo se tratará el texto contenido en un elemento cuando no quepa en él. Los posibles valores son:
  - `clip`: Valor por defecto. Oculta el texto que sobresale.
  - `ellipsis`: Corta el texto pero utiliza los puntos suspensivos "..." para indicar que hay más.

Nota: Para poder utilizar esta propiedad y que tenga efecto sobre el texto, hay que hacerlo en combinación con las siguientes propiedades:

```
overflow: hidden;
white-space: nowrap;
width: ancho_contenedor;
```

Propiedades de texto, fondo y bordes

Propiedades de texto

Propiedades del diseño

- `text-overflow`

  Ejemplo:

HTML

```
<p>Do you see any Teletubbies in
here? Do you see a slender plastic
tag clipped to my shirt with my
name printed on it?</p>
```

CSS

```
p{
   border:1px solid #000;
   overflow:hidden;
   text-overflow:ellipsis;
   white-space: nowrap;
   width:200px;
}
```

Do you see any Teletu...

Propiedades de texto, fondo y bordes

Propiedades de fondo y bordes

Propiedades de texto, fondo y bordes

Propiedades de fondo

Permiten aplicar estilo al fondo de los elementos. Algunas de las más utilizadas son las siguientes:

- `background-color`. Permite especificar el color de fondo de un elemento. Se puede especificar un color o el valor `transparent`.
- `background-image`. Permite especificar una o más imágenes de fondo de un elemento (separadas por comas).
- `background-origin`: Punto de referencia para el posicionamiento de la imagen o imágenes de fondo: `padding-box` (por defecto), `border-box`, `content-box`.
- `background-position`: Posición de la imagen o imágenes en el fondo tomando como referencia lo indicado en `background-origin`.
- `background-repeat`: Modo de repetición de la imagen.
- `background-size`: Tamaño de la imagen, ancho y alto en valor absoluto o porcentaje (del contenedor), o bien, `cover` o `contain`.

Propiedades de texto, fondo y bordes

Propiedades de fondo

Ejemplo:

CSS

```
div.mbg{
  background-image: url(imgs/flower.png), url(imgs/ball.png),
                    url(imgs/grass.png);
  background-position: right center, 20% 80%, top left;
  background-origin: padding-box, border-box, content-box;
  background-size: 100px, 40%, 200px;
  background-repeat: no-repeat; /* Se aplica a todas las imágenes */
  border:10px solid #000;
  height:400px;
  width:600px;
}
```

HTML

```
<div class='mbg'><p>Do you see any
Teletubbies in here? Do you see a
slender plastic tag clipped to my shirt
with my name printed on it?</p></div>
```

Propiedades de texto, fondo y bordes

Propiedades de borde

Permiten aplicar estilo al borde de los elementos. Algunas de las más utilizadas son las siguientes:

- `border-top-color`, `border-right-color`, `border-bottom-color`, `border-left-color`. Permiten especificar el color de cada uno de los bordes de un elemento.
- `border-color`. Permite especificar el color para los cuatro bordes de un elemento. Por ejemplo:

```
/* border-color: top right bottom left */
   border-color: red yellow green blue;
```

- `border-top-style`, `border-right-style`, `border-bottom-style`, `border-left-style`. Permiten especificar el estilo de cada uno de los bordes de un elemento. Los valores permitidos son: `none`, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset` y `outset`.

Propiedades de texto, fondo y bordes

Propiedades de borde

- **`border-style`**. Permite especificar el estilo para los cuatro bordes de un elemento de la misma forma que se hace con el color en la propiedad **`border-color`**.
- **`border-top-width`**, **`border-right-width`**, **`border-bottom-width`**, **`border-left-width`**. Permiten especificar el ancho de cada uno de los bordes de un elemento.
- **`border-width`**. Permite especificar el ancho para los cuatro bordes de un elemento de la misma forma que se hace con el color en la propiedad **`border-color`**.
- **`outline-style`**, **`outline-color`**, **`outline-width`**. Permiten especificar el estilo, ancho y color del *pseudo-borde*, respectivamente, de cualquier elemento. A diferencia de **`border`**, este pseudo-borde no ocupa espacio, por lo que no se vuelve a *renderizar* el documento. **`outline`** es la propiedad *shorthand*:

```
outline: color style width;
```

Propiedades de texto, fondo y bordes

Propiedades de borde

- `border-image`. Permite utilizar imágenes en los bordes en lugar de color.

```
border-image: url(imagen) porción_imagen modo_aplicación
```

Parámetros:

- `url(imagen)`: En imagen se indica el path de la imagen a utilizar.
- `porción_imagen`: Indica qué cantidad de imagen, empezando desde el borde se utilizará. Hay que indicar las cantidades en píxeles o porcentaje para arriba, derecha, abajo e izquierda; y en este orden. Lo ajusta al tamaño establecido de borde.
- `modo_aplicación`: Indica cómo se utilizará el patrón seleccionado de la imagen para rellenar el borde correspondiente. Las opciones posibles son: `stretch`, `repeat`, `round` y `space`.

Para utilizar esta propiedad hay que indicar un ancho de borde. Una solución correcta sería especificar un borde por defecto, mediante la propiedad *border*, para los navegadores que no soporten esta propiedad.

Propiedades de texto, fondo y bordes

Propiedades de borde

- **border-image**.

Ejemplo:

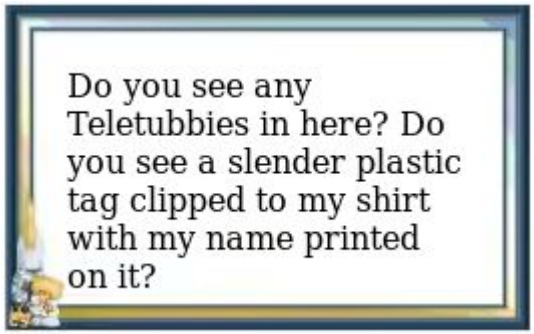
CSS

```
div.bimg{
  border:32px solid #f80; // ancho de borde por defecto
  border-image: url(imgs/marco.png) 32 32 32 32 stretch;
  height:100px;
  width:200px;
}
```

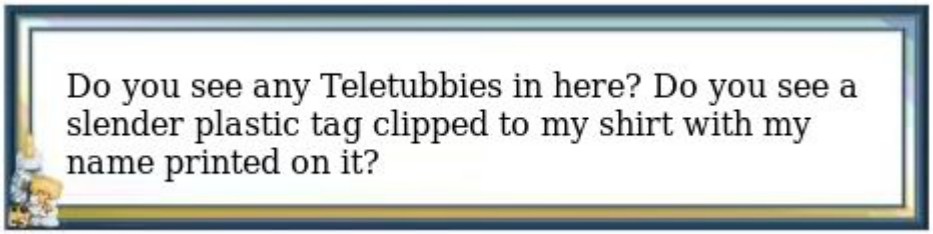
HTML

```
<div class='bimg'>Do you see any Teletubbies in here? Do you
see a slender plastic tag clipped to my shirt with my name
printed on it?</div>
```

Imagen utilizada como base para el marco



Resultado del ejemplo



Resultado del ejemplo cambiando el tamaño del div a 500px de ancho y 50px de alto

Propiedades de texto, fondo y bordes

Propiedades de borde

- **border-*-radius**. Permite redondear las esquinas del borde exterior de un elemento. Las propiedades correspondientes a las esquinas son: **border-top-left-radius**, **border-top-right-radius**, **border-bottom-right-radius** y **border-bottom-left-radius**.

```
border-nombre_esquina-radius: radio_horizontal radio_vertical
```

- **border-radius**. Permite redondear las cuatro esquinas del borde exterior de un elemento.

```
border-radius: radio_horizontal{1,4} / radio_vertical{1,4}
```

✓ El valor de *radio_horizontal* y *radio_vertical* se pueden expresar en unidades de longitud o porcentaje.
✓ Si sólo se indica un valor de radio para la esquina, lo tomará como radio vertical y horizontal.
✓ Si se utiliza la forma conjunta, cada radio horizontal se casará con su correspondiente radio vertical, si fuera el caso.

Propiedades de texto, fondo y bordes

Propiedades de borde

- `border-*-radius`.

Ejemplo:

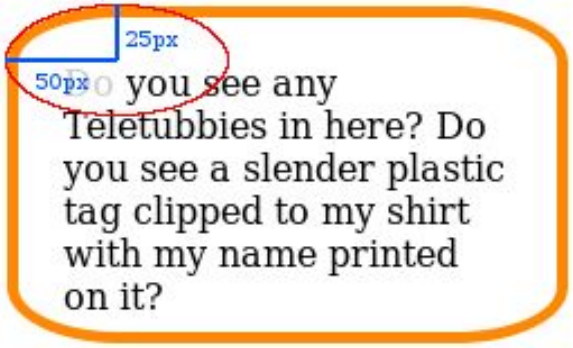
```
<div class='bradius'>Do you see any Teletubbies in here? Do you see a slender plastic tag clipped to my shirt with my name printed on it?</div>
```

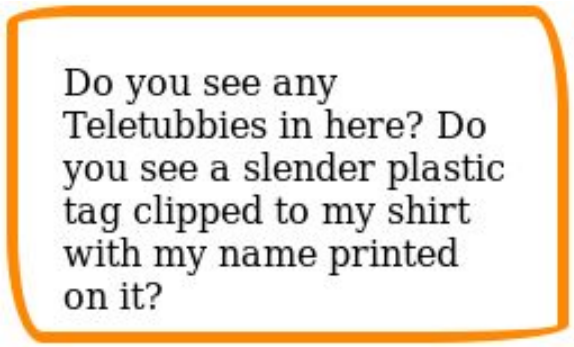
```
div.bradius{
  border:5px solid #f80;
  border-radius: 50px / 25px;
  height:100px;
  padding:20px;
  width:200px;
}
```

```
div.bradius{
  border:10px solid #f80;
  border-radius: 2em 1em 4em / 0.5em 3em;
  height:100px;
  padding:20px;
  width:300px;
}
```

Equivale a:

```
border-top-left-radius: 2em 0.5em;
border-top-right-radius: 1em 3em;
border-bottom-right-radius: 4em 0.5em;
border-bottom-left-radius:  1em 3em;
```





Propiedades de texto, fondo y bordes

Propiedades de borde

- **box-shadow**. Permite añadir una o más sombras, separadas por comas, a una "caja".

```
box-shadow: inset despl_h despl_v desenfoque expansión color
```

Parámetros:

- inset: optativo. Si se indica, la sombra que se aplica es interior.
- despl_h: desplazamiento horizontal de la sombra.
- despl_v: desplazamiento vertical de la sombra.
- desenfoque: radio del desenfoque a aplicar. Optativo.
- expansión: permite expandir o contraer la sombra.
- color: color de la sombra. Optativo.

✓ El desplazamiento horizontal/vertical admite valores negativos.
✓ Si no se indica el desenfoque, su valor por defecto es 0.
✓ El color de la sombra se puede indicar al principio o al final de la especificación.
✓ Si no se indica el color de la sombra se utilizará el color del texto.

Propiedades de texto, fondo y bordes

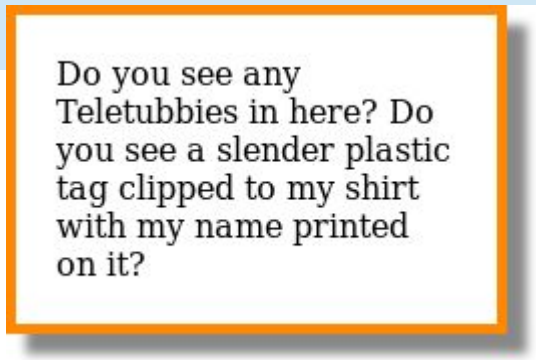
Propiedades de borde

- **box-shadow**.

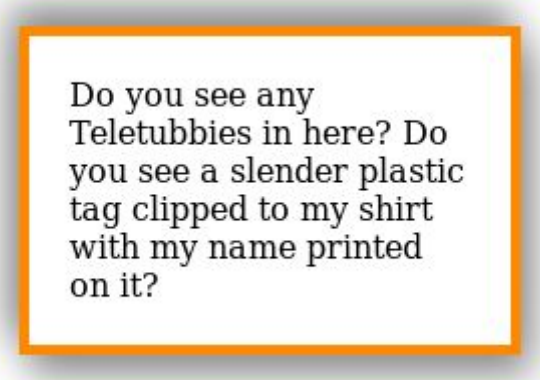
Ejemplo:

```
<div class='bshadow'>Do you see any Teletubbies in here? Do you see a slender plastic tag clipped to my shirt with my name printed on it?</div>
```

```
div.bshadow{
  border:5px solid #f80;
  box-shadow:10px 10px 5px #888888;
  padding:20px;
  width:200px;
}
```




```
div.bshadow{
  border:10px solid #f80;
  box-shadow: 5px 10px 20px #888888,
              -5px -10px 20px #888888;
  padding:20px;
  width:300px;
}
```



Web fonts
e
Icon fonts

Web fonts e Icon fonts

Web fonts

Web fonts e Icon fonts

Web fonts

CSS permite la utilización de fuentes *online*, es decir, fuentes que se encuentran disponibles en la Web, con la única inclusión de una directiva css: **@font-face**.

```
@font-face{
   font-family: nombre_de_la_fuente;
   src: url(url_donde_está_la_fuente) [format (formato_fuente)]
        [, url(url_donde_está_la_fuente) [format (formato_fuente)] ]*;
}
```

Valores:

- *nombre_de_la_fuente*: Nombre que se le asigna a la fuente y que se usará más adelante para hacer referencia a ella para utilizarla.
- *url_donde_está_la_fuente*: Dirección url en la que se encuentra del fichero correspondiente a la fuente.
- *formato_fuente*: Formato de la fuente que se va a cargar.

Web fonts e Icon fonts

Web fonts

Cadena	Formato de fuente	Extensión
“woff”, “woff2”	WOFF 1.0 y WOFF 2.0 (Web Open Font Format)	.woff
“truetype”	TrueType	.ttf
“opentype”	OpenType	.ttf, .otf
“embedded-opentype”	Embedded OpenType	.eot
“svg”	SVG Font	.svg, .svgz

Ejemplo:

```
@font-face{
   font-family: MyGentium;  /* Nombre asignado a la fuente */
   src: local(Gentium),     /* usar fuente local, si está instalada */
        url('Gentium.ttf'); /* si no, se descargará */
}
p{ font-family: MyGentium, Verdana, sans-serif; }
```

En el ejemplo se define la fuente *MyGentium* como una fuente nueva, que si está instalada en el ordenador no se descargará y si no se encuentra en el ordenador se descargará de la url indicada.

Web fonts e Icon fonts

Web fonts

URL: https://fonts.google.com/

Hay dos formas sencillas de añadir a nuestro css las fuentes que proporciona Google:

✓ **En el html del documento:** añadiendo un elemento **<link>** en el <head> del documento.

```
<link href='url_de_la_fuente_en_Google' rel='stylesheet' type='text/css'>
```

✓ **En el css del documento:** Mediante la directiva **@import** de CSS.

```
@import url(url_de_la_fuente_en_Google);
```

En caso de que no se pueda utilizar ninguna de estas dos formas, existe una tercera. Se trata de incluir la fuente mediante un código javascript.

Web fonts e Icon fonts

Web fonts. Google fonts.

Ejemplo: Utilizando un elemento **<link>** en el HTML.

```
<head>
...
   <link href='http://fonts.googleapis.com/css?family=Orienta' rel='stylesheet'
   type='text/css'>
   <style>
      .pfnt{ border:5px solid #f80; font-family: 'Orienta', sans-serif;
             padding:20px; width:200px; height:100px;}
   </style>
</head>
...
<p class='pfnt'>
   Do you see any Teletubbies in here? Do you see a slender plastic tag clipped to
   my shirt with my name printed on it?
</p>
```

Do you see any Teletubbies in here? Do you see a slender plastic tag clipped to my shirt with my name printed on it?

Web fonts e Icon fonts

Web fonts. Google fonts.

Ejemplo: Utilizando la directiva **@import** en el CSS.

```
@import url(http://fonts.googleapis.com/css?family=Orienta);

p.pfnt{ border:5px solid #f80; font-family: 'Orienta', sans-serif;
        padding:20px; width:200px; height:100px;}
```

```
<div class='dtns'>
   Do you see any Teletubbies in here? Do you see a slender plastic
   tag clipped to my shirt with my name printed on it?
</div>
```

Do you see any Teletubbies in here? Do you see a slender plastic tag clipped to my shirt with my name printed on it?

Web fonts e Icon fonts

Web fonts. Google fonts.

Si se abre en el navegador la dirección de ubicación de la fuente Google, se obtiene el css con la directiva **`@font-face`**.

Ejemplo: http://fonts.googleapis.com/css?family=Orienta

```
/* latin-ext */
@font-face {
  font-family: 'Orienta';
  font-style: normal;
  font-weight: 400;
  src: url(https://fonts.gstatic.com/s/orienta/v13/PlI9FlK4Jrl5Y9zNSy6i9URF.woff2) format('woff2');
  unicode-range: U+0100-024F, U+0259, U+1E00-1EFF, U+2020, U+20A0-20AB, U+20AD-20CF, U+2113,
U+2C60-2C7F, U+A720-A7FF;
}
/* latin */
@font-face {
  font-family: 'Orienta';
  font-style: normal;
  font-weight: 400;
  src: url(https://fonts.gstatic.com/s/orienta/v13/PlI9FlK4Jrl5Y9zNSyCi9Q.woff2) format('woff2');
  unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02BB-02BC, U+02C6, U+02DA, U+02DC, U+2000-206F,
U+2074, U+20AC, U+2122, U+2191, U+2193, U+2212, U+2215, U+FEFF, U+FFFD;
}
```

Web fonts e Icon fonts

Icon fonts

Web fonts e Icon fonts

Icon fonts.

CSS también da la opción de utilizar fuentes de iconos o *icon fonts*. No es más que una fuente pero que, en lugar de letras, lo que proporcionan son iconos.

Las fuentes de iconos se usan principalmente por dos motivos:

- Se quiere ***mejorar*** **una palabra.**
- Se quiere **utilizar sólo el icono** y se quiere que sea **funcional** e **informativo.**

Web fonts e Icon fonts

Icon fonts. Ventajas.

Algunas de las ventajas de usar fuentes de iconos en lugar de imágenes en el desarrollo de una web son las siguientes:

- Se pueden escalar muy fácilmente y sin perder calidad.
- Se les puede cambiar de color fácilmente.
- Se les puede aplicar sombras muy fácilmente.
- Se les puede aplicar propiedades de texto.
- Se puede hacer con ellos lo mismo que con una imagen teniendo un peso menor.

Web fonts e Icon fonts

Icon fonts. Cómo se utilizan.

Las fuentes de iconos se cargan igual que las web fonts, mediante la directiva `@font-face`.

```
@font-face {
    font-family: 'icomoon';
    src:url('fonts/icomoon.eot');
    src:url('fonts/icomoon.eot?#iefix') format('embedded-opentype'),
        url('fonts/icomoon.woff') format('woff'),
        url('fonts/icomoon.ttf') format('truetype'),
        url('fonts/icomoon.svg#icomoon') format('svg');
    font-weight: normal;
    font-style: normal;
}
```

Icon fonts gratuitas: http://icomoon.io/, http://fontello.com/

Web fonts e Icon fonts

Icon fonts. Uso para *mejorar una palabra*.

```
[class^="icon-"], [class*=" icon-"] {
   font-family: 'icomoon'; /* fuente de iconos incluida en el css */
   speak: none; /* indica que los screen readers no deberían leerla */
}
.icon-cart:before   {content: "\e600";}
.icon-switch:before {content: "\e601";}
.icon-user:before   {content: "\e602";}
.icon-enter:before  {content: "\e603";}
.icon-exit:before   {content: "\e604";}
.icon-home:before   {content: "\e605";}
h3{
   color:#3223EB;
}
```

`speak:none;` en el CSS y **`aria-hidden="true"`** en el HTML evitan que los *screen-readers* intenten leer el icono.


```
<h3>
  <span class="icon-user" aria-hidden="true"></span> Perfil de usuario
</h3>
```

Web fonts e Icon fonts

Icon fonts. Utilizando *sólo el icono*.

```
[class^="icon-"], [class*=" icon-"] {
   font-family: 'icomoon'; /* fuente de iconos incluida en el css */
   speak: none; /* indica que los screen readers no deberían leerla */
}
.icon-switch:before {content: "\e601";}
.icon-user:before   {content: "\e602";}
.icon-exit:before   {content: "\e604";}
.icon-home:before   {content: "\e605";}
li { list-style: none; float:left;}
li a{ color:#800; display:inline-block;
      padding:0 1em; text-decoration: none;}
.screen-reader-text{ position:absolute; top:-9999px; left:-9999px}
```

```
<ul>
   <li><a href="#"><span aria-hidden="true" class="icon-home"></span>
       <span class="screen-reader-text">Inicio</span></a></li>
   <li><a href="#"><span aria-hidden="true" class="icon-switch"></span>
       <span class="screen-reader-text">Login</span></a></li>
</ul>
```

Media queries

Media queries

¿Qué son?

Son reglas CSS que permiten aplicar estilos concretos en función del dispositivo en el que se mostrará el documento y sus características.

- En CSS2 existía ya la posibilidad de aplicar hojas de estilo en función del dispositivo.
- A partir de CSS3 se permite además aplicar estilos no sólo en función del dispositivo, sino también de las características de éste.
- Gracias a las *media queries*, la presentación de un documento se puede ajustar a cualquier dispositivo sin la necesidad de cambiar el contenido en sí. Esto se conoce como ***Responsive Design***.
- Una *media query* se compone de un tipo de dispositivo y cero o más expresiones que comprueban las condiciones de las características particulares del dispositivo.

Media queries

Sintaxis

```
@media dispositivos [ and (condición)]* { reglas CSS }
```

Donde:

- *dispositivos*: nombre del dispositivo (*all*, *print*, *screen*, *speech*). Si hay más de un dispositivo se colocan entre paréntesis y separados por comas.
- *condición*: indica la condición (si fuera el caso) que debe cumplir una característica del dispositivo para que se aplique el conjunto de reglas. Si se trata de más de una condición, cada una irá entre paréntesis y precedida por el operador **and**.

Ejemplo:

```
<link rel="stylesheet" media="screen and (color)" href="estilo.css">
```

```
@import url(estilos.css) screen and (color);
```

```
@media all and (min-width:500px) { … }
```

Media queries

Características del medio

La lista de características que puede comprobar una media query es amplia, aunque lo más utilizado es el alto (**`height`**) y ancho (**`width`**) de la ventana en la que se muestra la página. Admiten los prefijos `min` y `max`.

Ejemplo:

Para dispositivos con un ancho de pantalla igual o mayor a 400px:

```
@media screen and (min-width: 400px) { ... }
```

Para dispositivos con un alto de pantalla menor o igual a 700px:

```
@media screen and (max-height: 700px) { ... }
```

Para dispositivos con un ancho de pantalla entre 400px y 700px, ambos incluidos:

```
@media screen and (min-width: 400px) and (max-width: 700px) { ... }
```

Media queries

Ejemplo de aplicación:

Mobile First Design

Media queries

Ejemplo de aplicación: *Mobile First Design*

Responsive Web Design (RWD)

- Se trata de una técnica de **diseño adaptativo**. Consiste en hacer que **la estructura de las páginas web se adapte al tamaño del dispositivo** en el que se visualizan. De esta manera **proporciona** al usuario una **experiencia visual óptima** en todo momento.
- La estructura de las páginas se hace mediante diseños basados en **rejillas** e **imágenes flexibles** y que son optimizados gracias a un uso inteligente del CSS y ***media queries***.
- Esta técnica evita tener que crear diferentes diseños para distintas resoluciones y/o dispositivos.

Media queries

Ejemplo de aplicación: *Mobile First Design*

Mobile First Design

Más que de una técnica de diseño, se trata de una **filosofía de trabajo** basada en el *Responsive Web Design* y centrada en el *usuario móvil*. Esto último implica plantearse varias cosas:

- **La conexión a internet**. Puede ser 3G/4G/5G o WiFi, por lo que podría tener poca cobertura (o ninguna), lo cual afecta directamente en la velocidad de conexión.
- El **tamaño de la pantalla del dispositivo** con el que accede puede ser cualquiera.
- Los **dispositivos móviles no tiene las mismas interfaces** que un ordenador de escritorio, ni que un portátil, es decir, carecen de ratón, teclado, etc. Sin embargo, los dispositivos móviles poseen pantallas táctiles.

Media queries

Ejemplo de aplicación: *Mobile First Design*

Mobile First Design

Esta filosofía de trabajo es, hoy en día, la primera opción a tener en cuenta a la hora de desarrollar una nueva aplicación o sitio web, por tres razones:

- **El mercado de móviles está en plena expansión**, aumentando el número de usuarios que acceden a una web utilizando un dispositivo móvil.
- Diseñar para un dispositivo móvil obliga a **centrarse en el contenido y las acciones más importantes** de un sitio o aplicación web.
- **Los dispositivos móviles aumentan las posibilidades** gracias a las características que proporcionan, tales como GPS, *multi-touch*, acelerómetro, localización, etc.

Media queries

Ejemplo de aplicación: *Mobile First Design*

Herramientas utilizadas en *Mobile First Design*

El conjunto de herramientas a utilizar son las proporcionadas por el *RWD*. Estas herramientas van a permitir diseños que se adapten a cualquier tamaño de pantalla.

- **Rejilla flexible (*fluid grid*)**
- **Imágenes flexibles (*fluid images*)**
- **Media queries**

Sin embargo, en el contexto ***Mobile*** no es suficiente con estas herramientas, hay que tener en cuenta aspectos como ***conectividad***, ***rendimiento***, etc, además de las ***nuevas funcionalidades*** que proporcionan los dispositivos (pantallas táctiles, gps, etc).

Media queries

Ejemplo de aplicación: *Mobile First Design*

Pasos a seguir en el diseño *Mobile First*

1. Lo primero es **centrarse en el contenido**. Identificar cuál es el contenido importante que interesa al usuario e intentar presentarlo de la forma más rápida y coherente posible.
2. **Empezar el diseño pensando en** un dispositivo móvil, más concretamente en **un teléfono móvil**. Centrarse en el contenido y la funcionalidad proporcionando una buena experiencia de usuario.
3. **Adaptar el diseño al contexto**. Partiendo del diseño obtenido en el paso anterior, ir *escalándolo* para resoluciones mayores, al igual que la funcionalidad y el contenido. La regla a seguir es fácil,: "*cuando el tamaño del dispositivo haga que el diseño se rompa y/o no se vea bien, es momento de aplicar nuevas reglas css que lo corrijan*". Esto se consigue **mediante *media queries***.

Media queries

Ejemplo de aplicación: *Mobile First Design*

Cosas a tener en cuenta en *Mobile First Design*

Estructura:

- Utilizar la semántica de HTML, manteniendo el HTML simple y limpio.
- Indicar el valor de la etiqueta `<meta>` **`viewport`** para evitar que el dispositivo escale el diseño al tamaño de la pantalla:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- Dividir el contenido en fragmentos (documentos html) separados, facilitando la carga rápida del contenido esencial y haciendo accesible el contenido secundario mediante enlaces.

Media queries

Ejemplo de aplicación: *Mobile First Design*

Cosas a tener en cuenta en *Mobile First Design*

Estilo:

- Trabajar con el **CSS** para que sea lo más **ligero** y **fluido** posible.
- Empezar con el estilo básico pensado para contexto ***Mobile***.
- Utilizar ***media queries*** para especificar el estilo para resoluciones mayores.

  Ejemplo:

```
@media screen and (min-width: 40.5em) {
    .product-img {
      width: 50%;
      float: left;
    }
}
```

Media queries

Ejemplo de aplicación: *Mobile First Design*

Cosas a tener en cuenta en *Mobile First Design*

Estilo (y II):

- ✓ Utilizar **unidades relativas**. El uso de porcentajes y unidades *em* hacen el diseño más flexible y compatible con los distintos tamaños de pantalla, densidad de píxel y nivel de zoom. Se define un tamaño de fuente por defecto y se utiliza como base para calcular otras dimensiones (regla de *Ethan Marcotte*):

```
tamaño_objetivo / tamaño_fuente_contenedor = unidades em

Ejemplo: 24px / 16px = 1.5 em
```

- ✓ Utilizar CSS para reducir peticiones http. Se pueden utilizar gradientes css en lugar de imágenes para los *backgrounds*, web e icon fonts, etc.
- ✓ Dejar que el contenido indique dónde colocar *breakpoints* para las *media queries*.

Media queries

Ejemplo de aplicación: *Mobile First Design*

Cosas a tener en cuenta en *Mobile First Design*

Comportamiento:

- Una vez se tiene la estructura y el diseño, se puede añadir **JavaScript** para mejorarlo y añadir funcionalidad.

Navegación:

- El menú de navegación en pantallas pequeñas puede pasar a ser secundario, por lo que por defecto permanece oculto y sólo se hace visible cuando el usuario lo requiere (mediante un enlace, botón, etc).

Imágenes adaptativas:

- Si se utilizan imágenes grandes, considerar la posibilidad de cargar por defecto versiones optimizadas en el contexto *Mobile*, utilizando carga condicional de las imágenes más grandes cuando el dispositivo lo permita.

CSS Flexbox
(Flexible Box)

CSS Flexbox

- Proporciona un modo más eficiente de diseñar y distribuir el espacio en blanco entre los elementos de un contenedor.
- Funciona con contenedores de tamaño fijo, dinámico o desconocido.
- Sistema de **diseño unidimensionales**: fila o columna.
- Es apropiado para **diseños a pequeña escala**.

CSS Flexbox

Los nodos hijo de un contenedor de tipo ***flex*** se pueden:

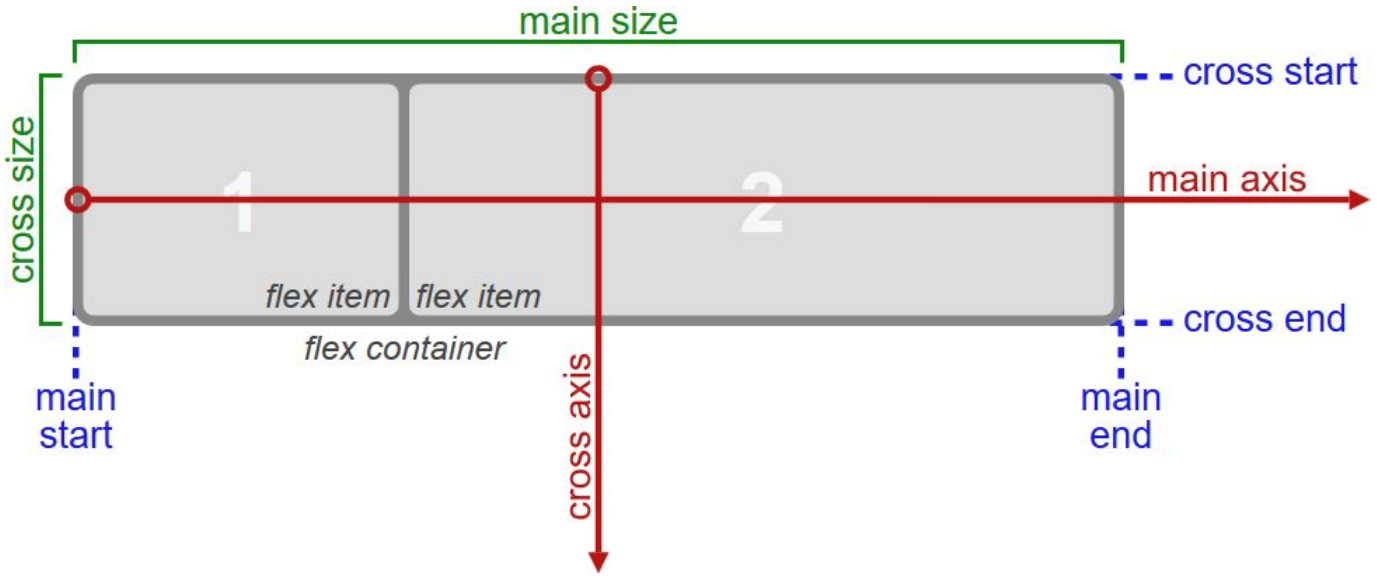
- colocar en cualquier dirección: hacia la izquierda, hacia la derecha, hacia arriba o hacia abajo;
- reordenar en la capa de estilo (css);
- cambiar su ancho/alto para aprovechar de la mejor forma posible todo el espacio disponible;
- alinear con respecto al contenedor o a sí mismo;
- colapsar o mostrar a lo largo del eje principal (*main axis*) o del secundario (*cross axis*).

CSS Flexbox

- Un contenedor ***flex*** es un elemento caja con su propiedad *display* a ***flex*** o ***inline-flex***.

`display: flex;`	Hace que el contenedor sea *flex* de tipo bloque.
`display: inline-flex;`	Hace que el contenedor sea *flex* de tipo *inline*.

- La disposición de los elementos dentro de un contenedor flex puede ser a lo largo del eje secundario (*cross axis*, por defecto el vertical) o del eje principal (*main axis*, por defecto el horizontal):



CSS Flexbox

Propiedades de un contenedor *flex*

- `flex-direction`. Indica el eje en el que se aplicará la disposición *flex*. Los posibles valores son:
  - `row`: de izquierda a derecha (valor por defecto).
  - `row-reverse`: de derecha a izquierda.
  - `column`: de arriba a abajo.
  - `column-reverse`: de abajo a arriba.
- `flex-wrap`. Por defecto, los nodos hijo de un contenedor *flex* se mostrarán todos en la misma línea. Esta propiedad permite cambiar este comportamiento mediante los siguientes valores:
  - `nowrap`: todos los elementos en la misma línea, de izquierda a derecha (valor por defecto).
  - `wrap`: múltiples líneas, empezando de izquierda a derecha.
  - `wrap-reverse`: múltiples líneas, empezando de derecha a izquierda.

CSS Flexbox

Propiedades de un contenedor *flex*

- `flex-flow`. Permite indicar los valores de *flex-direction* y *flex-wrap* en la misma propiedad. El valor por defecto es:

```
flex-flow: row nowrap;
```

- `justify-content`. Permite definir la alineación de los elementos (nodos hijo) a lo largo del eje indicado con *flex-direction*. Admite los siguientes valores:
  - `flex-start`: Los elementos se alinean al inicio del eje (valor por defecto).
  - `flex-end`: Los elementos se alinean al final del eje.
  - `center`: Los elementos se centran en el eje.
  - `space-between`: Los elementos se distribuyen uniformemente por todo el eje de manera que el primer elemento se coloca al inicio del eje y el último elemento al final.
  - `space-around`: Los elementos se distribuyen uniformemente por todo el eje de manera que el espacio a la derecha e izquierda de cada uno es el mismo.

CSS Flexbox

Propiedades de un contenedor *flex*

- `align-items`. Permite indicar cómo se colocan los elementos a lo largo del eje secundario. Admite los valores:
  - `flex-start`: Los elementos se alinean al inicio del eje.
  - `flex-end`: Los elementos se alinean al final del eje.
  - `center`: Los elementos se centran en el eje.
  - `baseline`: Los elementos se alinean de manera que la primera línea de texto de cada elemento está en la misma línea.
  - `stretch`: Estira los elementos para que rellenen todo el eje (valor por defecto).

CSS Flexbox

Propiedades de un contenedor *flex*

- `align-content`. Permite alinear las líneas del contenedor *flex* en función del espacio libre en el eje secundario. Admite los siguientes valores:
  - `flex-start`: Las líneas se sitúan al inicio del eje.
  - `flex-end`: Las líneas se sitúan al final del eje.
  - `center`: Las líneas se centran en el contenedor.
  - `space-between`: Las líneas se distribuyen uniformemente por todo el eje de manera que la primera línea se coloca al inicio del contenedor y la última línea al final.
  - `space-around`: Las líneas se distribuyen uniformemente por todo el eje de manera que el espacio encima y debajo de cada una es el mismo.
  - `stretch`: Las líneas se estiran para utilizar todo el espacio restante (valor por defecto).

CSS Flexbox

Propiedades de los nodos hijo de un contenedor *flex*

- `order`. Permite cambiar el orden en el que se muestran los nodos hijo a lo largo del eje indicado por *flex-direction*. Por defecto se muestran en el orden en el que aparecen en el código fuente. Admite un número entero positivo o negativo, como valor.
- `flex-grow`. Permite asignar a los nodos hijo la posibilidad de crecer si fuera necesario. Acepta como valor números enteros que sirven para indicar el factor de crecimiento. Sólo se aplica el factor de crecimiento cuando hay espacio disponible en el contenedor *flex* padre. El valor por defecto es 0.
- `flex-shrink`. Permite a un nodo hijo encogerse si fuera necesario. Admite un número entero positivo. El valor por defecto es 1.

CSS Flexbox

Propiedades de los nodos hijo de un contenedor *flex*

- **`flex-basis`**. Define el tamaño por defecto de un elemento antes de que se distribuya el espacio disponible libre. Admite valores de longitud (30%, 2em, etc) y *auto*, que hará que se utilice el valor de la propiedad *width* o *height* de css, en función de la dirección.
- **`flex`**. Permite indicar los valores de *flex-grow*, *flex-shrink* y *flex-basis* en la misma propiedad. El segundo y tercer parámetro son opcionales. El valor por defecto de esta propiedad es:

```
flex: 0 1 auto;
```

- **`align-self`**. Permite modificar la alineación por defecto para elementos de forma individual. Admite los mismos valores que la propiedad *align-items* (flex-start, flex-end, center, baseline, stretch) y *auto*.

CSS Grid Layout

CSS Grid Layout

- Se trata del sistema de **diseño más potente** en CSS.
- Sistema de **diseño bidimensional**: filas y columnas.
- Es apropiado para **diseños a gran escala**.
- Las reglas CSS se aplican tanto al elemento padre (**contenedor Grid**) como a los elementos hijos (**elementos Grid**).

CSS Grid Layout

Un contenedor **`grid`** es un elemento caja con su propiedad *display* a ***`grid`*** o ***`inline-grid`***.

`display: grid;`	Hace que el contenedor sea `grid` de tipo bloque.
`display: inline-grid;`	Hace que el contenedor sea `grid` de tipo *inline*.

CSS Grid Layout

Propiedades de un contenedor *grid*

- **grid-template-rows**, **grid-template-columns**. Definen el número de filas y columnas del grid con una lista de valores separadas por espacios en blanco.

```
.contenedor{
  display: grid;
  grid-template-rows: <tamaño> ... | <nombre> <tamaño> ...;
  grid-template-columns: <tamaño> ... | <nombre> <tamaño> ...;
}
```

  - **<tamaño>**. Puede ser una longitud, un porcentaje o una fracción del espacio libre en el contenedor grid (se utiliza la unidad `fr` para esto último).
  - **<nombre>**. Es un nombre arbitrario que se puede dar a la fila/columna para poder hacer referencia a ella en otras reglas de CSS pertenecientes a Grid Layout.

CSS Grid Layout

Propiedades de un contenedor *grid*

- **grid-template-rows**, **grid-template-columns**.

  Ejemplo:

```
.contenedor{
  display: grid;
  grid-template-rows: [row1-start] 25%
                      [row1-end] 100px
                      [third-line] auto
                      [last-line];
  grid-template-columns: [first] 40px
                         [line2] 50px
                         [line3] auto
                         [col4-start] minmax(50px, 1fr)
                         [five] 40px
                         [end];
}
```

La función CSS **minmax()** permite definir un rango para el ancho/alto de una columna/fila. Los valores admitidos son longitud, porcentaje, fracción (`fr`), `max-content`, `min-content` y `auto`. Los valores `max-content` y `min-content` definen como ancho/alto de columna/fila el mismo del elemento con tamaño máximo/mínimo en la fila/columna, respectivamente

CSS Grid Layout

Propiedades de un contenedor *grid*

- `grid-template-rows`, `grid-template-columns`.

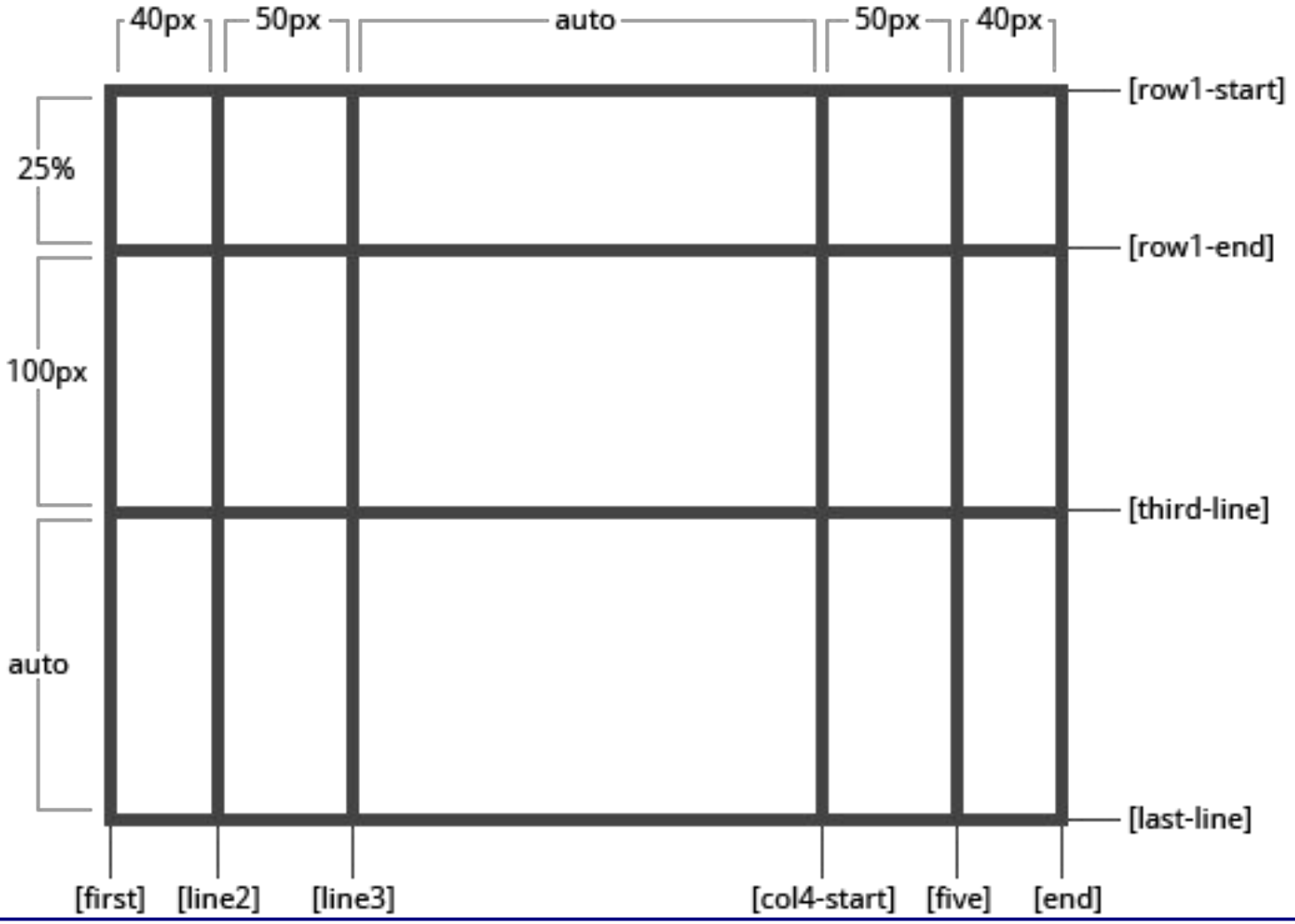
  Ejemplo:

  (Resultado)

```
.contenedor{
  display: grid;

  grid-template-rows:
    [row1-start] 25%
    [row1-end] 100px
    [third-line] auto
    [last-line];

  grid-template-columns:
    [first] 40px
    [line2] 50px
    [line3] auto
    [col4-start] 50px
    [five] 40px
    [end];
}
```



CSS Grid Layout

Propiedades de un contenedor *grid*

- **grid-template-rows**, **grid-template-columns**.

  Una fila/columna puede tener más de un nombre.

  Ejemplo:

```
.contenedor{
  display: grid;
  grid-template-rows: [row1-start] 25% [row1-end row2-start] 25% [row2-end];
}
```

  Se puede utilizar la función **repeat()** para repetir partes de una regla:

```
  grid-template-rows: repeat(3, [row-start] 25%) 5%;
```

  Como primer argumento de **repeat()** también se pueden utilizar los valores `auto-fill` y `auto-fit` para que realice de forma automática el cálculo del número de repeticiones

  Equivale a repetir 3 veces "`[row-start] 25%`":

```
  grid-template-rows: [row-start] 25% [row-start] 25% [row-start] 25% 5%;
```

CSS Grid Layout

Propiedades de un contenedor *grid*

- **grid-template-areas**. Define un grid referenciando los nombres de las áreas grid especificadas por la propiedad grid-area.

```
.contenedor{
  display: grid;
  grid-template-areas: "<grid-area-name> | . | none | ... " "...";
}
```

  - **<grid-area-name>**. El nombre de una de las áreas grid especificada en la propiedad grid-area.
  - **“.”**. El punto significa una celda vacía del grid.
  - **none**. No se define ninguna área grid.

CSS Grid Layout

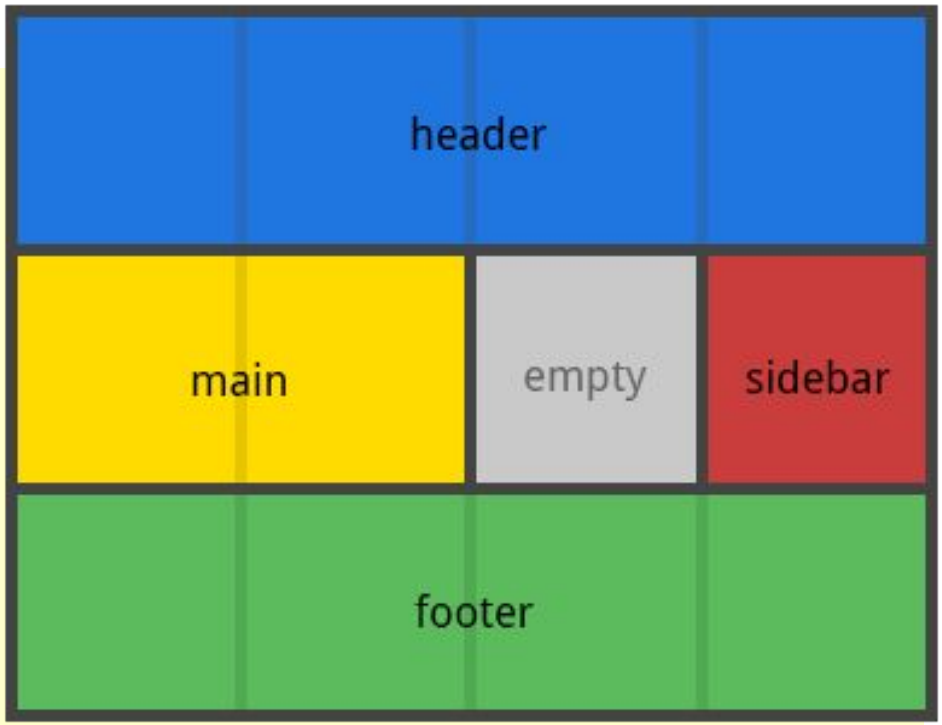
Propiedades de un contenedor *grid*

- **grid-template-areas**.

Ejemplo 1:

```
.item-a {grid-area: header;}
.item-b {grid-area: main;}
.item-c {grid-area: sidebar;}
.item-d {grid-area: footer;}

.container {
  grid-template-areas:
    "header header header header"
    "main main . sidebar"
    "footer footer footer footer";
  grid-template-rows: auto;
  grid-template-columns: 50px 50px 50px 50px;
}
```



CSS Grid Layout

Propiedades de un contenedor *grid*

- **grid-template-areas**.

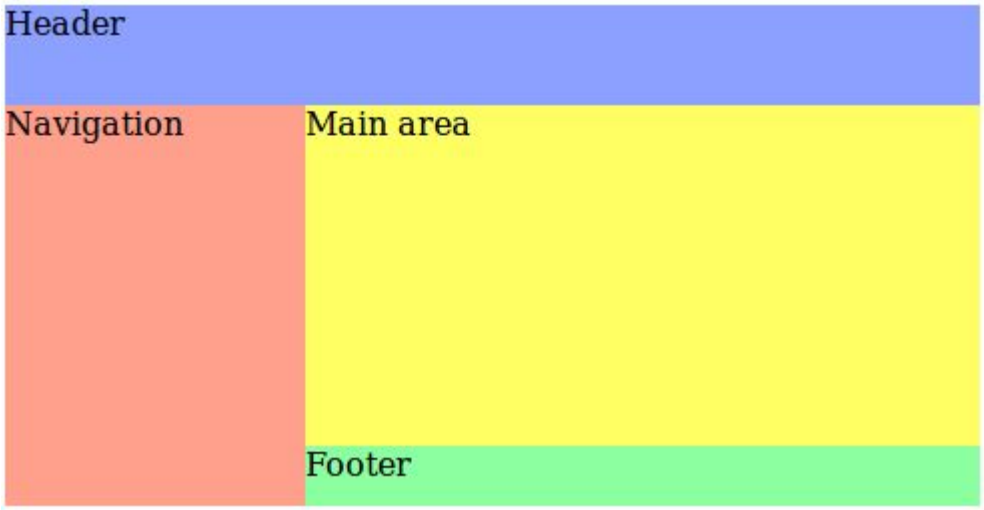
Ejemplo 2:

CSS

```
#page{
  display: grid;
  width: 100%;
  height: 250px;
  grid-template-areas: "head head"
                       "nav  main"
                       "nav  foot";
  grid-template-rows: 50px 1fr 30px;
  grid-template-columns: 150px 1fr;
}
#page>header{
  grid-area: head;
  background-color: #8ca0ff;}
#page>nav{
  grid-area: nav;
  background-color: #ffa08c;}
#page>main{
  grid-area: main;
  background-color: #ffff64;}
#page>footer{
  grid-area: foot;
  background-color: #8cffa0;}
```

HTML

```
<section id="page">
  <header>Header</header>
  <nav>Navigation</nav>
  <main>Main area</main>
  <footer>Footer</footer>
</section>
```



CSS Grid Layout

Propiedades de un contenedor *grid*

- **grid-template**. Es la propiedad *shorthand* que permite especificar las propiedades `grid-template-rows`, `grid-template-columns` y `grid-template-areas` en una sola.

```
.contenedor{
  display: grid;
  grid-template: none | subgrid |
                 <grid-template-rows> / <grid-template-columns>;
}
```

  - **none**. Las tres propiedades toman su valor inicial.
  - **subgrid**. Establece `grid-template-rows` y `grid-template-columns` a `subgrid`, y `grid-template-areas` a su valor inicial .
  - **<grid-template-rows> / <grid-template-columns>**. Establece el valor de **<grid-template-rows>** y **<grid-template-columns>** a los valores especificados y el valor de **<grid-template-areas>** a none.

CSS Grid Layout

Propiedades de un contenedor *grid*

- **grid-template**.

  Ejemplo:

```
.contenedor {
  grid-template:
    [row1-start] "header header header" 25px [row1-end]
    [row2-start] "footer footer footer" 25px [row2-end]
    / auto 50px auto;
}
```

es equivalente a:

```
.contenedor {
  grid-template-rows: [row1-start] 25px [row1-end row2-start] 25px
                      [row2-end];
  grid-template-columns: auto 50px auto;
  grid-template-areas:
    "header header header"
    "footer footer footer";
}
```

CSS Grid Layout

Propiedades de un contenedor *grid*

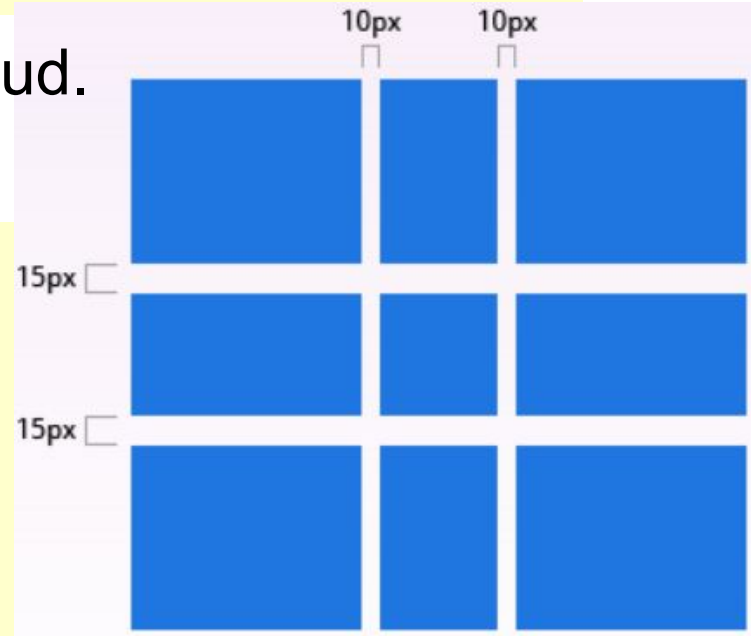
- **grid-row-gap**, **grid-column-gap**. Permiten especificar la anchura de las líneas que delimitan las celdas del grid.

```
.contenedor{
  display: grid;
  grid-column-gap: <tamaño-línea>;
  grid-row-gap: <tamaño-línea>;
}
```

  - **<tamaño-línea>**. Un valor de longitud.

Ejemplo:

```
.contenedor{
  display: grid;
  grid-template-columns: 100px 50px 100px;
  grid-template-rows: 80px auto 80px;
  grid-column-gap: 10px;
  grid-row-gap: 15px;
}
```



CSS Grid Layout

Propiedades de un contenedor *grid*

- **grid-gap**. Propiedad *shorthand* para grid-row-gap y grid-column-gap.

```
.contenedor{
  display: grid;
  grid-gap: <grid-row-gap> <grid-column-gap>;
}
```

  - **<grid-row-gap>**, **<grid-column-gap>**. Valores de longitud.

Ejemplo:

```
.contenedor{
  display: grid;
  grid-template-columns: 100px 50px 100px;
  grid-template-rows: 80px auto 80px;
  grid-gap: 10px 15px;
}
```

Si sólo se especifica un valor se asume el mismo para grid-row-gap y para grid-column-gap.

CSS Grid Layout

Propiedades de un contenedor *grid*

- **justify-items**. Permite alinear el contenido del grid en el eje **horizontal**. El valor es heredado por todos los elementos grid que haya en el contenedor.

```
.contenedor{
  display: grid;
  justify-items: start | end | center | stretch;
}
```

  - **start**. Alinea el contenido a la izquierda del área grid.
  - **end**. Alinea el contenido a la derecha del área grid.
  - **center**. Alinea el contenido en el centro del área grid.
  - **stretch**. Rellena todo el contenido del área grid. Este es el valor por defecto.

CSS Grid Layout

Propiedades de un contenedor *grid*

- **align-items**. Permite alinear el contenido del grid en el eje **vertical**. El valor es heredado por todos los elementos grid que haya en el contenedor.

```
.contenedor{
  display: grid;
  align-items: start | end | center | stretch;
}
```

  - **start**. Alinea el contenido a la parte superior del área grid.
  - **end**. Alinea el contenido a la parte inferior del área grid.
  - **center**. Alinea el contenido en el centro del área grid.
  - **stretch**. Rellena todo el contenido del área grid. Este es el valor por defecto.

CSS Grid Layout

Propiedades de un contenedor *grid*

- **justify-content**. Permite alinear la rejilla dentro del contenedor grid **horizontalmente**. Esto es útil cuando el tamaño de los elementos grid se ha definido con unidades no flexibles como px.

```
.contenedor{
  display: grid;
  justify-content: start | end | center | stretch |
                   space-around | space-between | space-evenly;
}
```

  - **start**. Alinea el contenido a la izquierda del área grid.
  - **end**. Alinea el contenido a la derecha del área grid.
  - **center**. Alinea el contenido en el centro del área grid.
  - **stretch**. Rellena todo el contenido del área grid.
  - **space-around**. Sitúa un número par de espacios entre los elementos grid. Los de los extremos de la fila tiene la mitad de tamaño.
  - **space-between**. Igual que space-around, pero sin espacios en los extremos.
  - **space-evenly**. Igual que space-around, pero incluyendo los de los extremos.

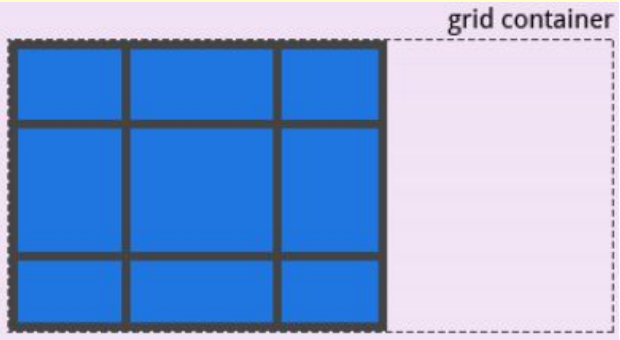
CSS Grid Layout

Propiedades de un contenedor *grid*

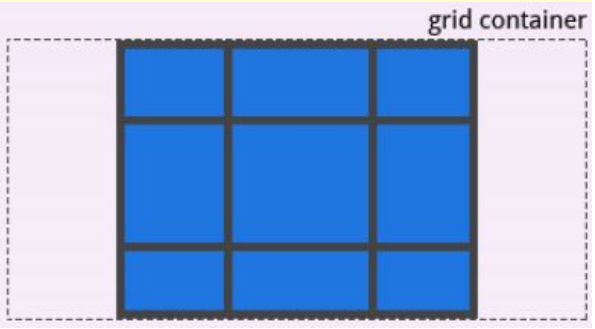
- **`justify-content`**.

  Ejemplo:

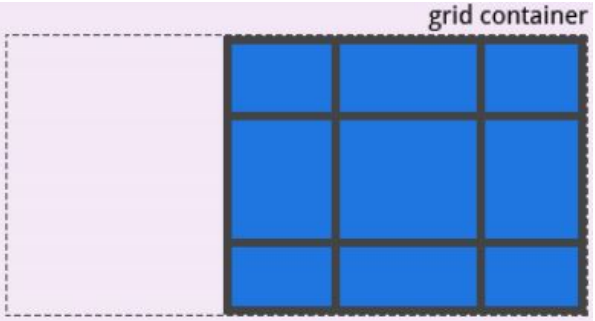
`justify-content: start;`



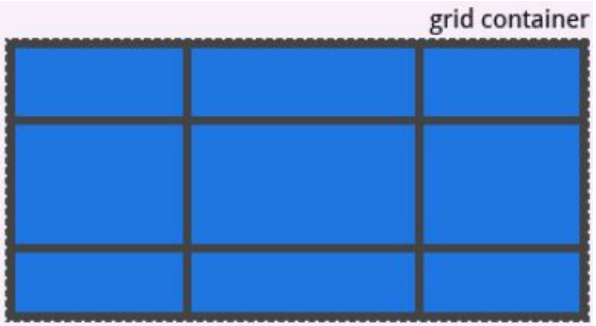
`justify-content: center;`



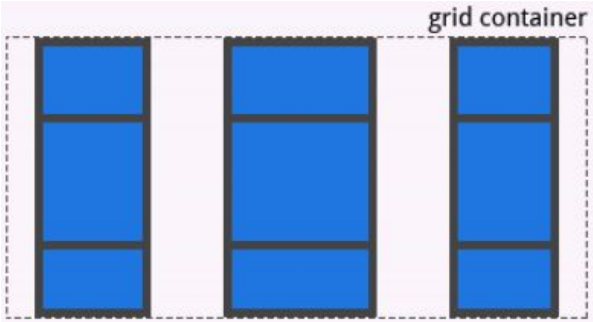
`justify-content: end;`



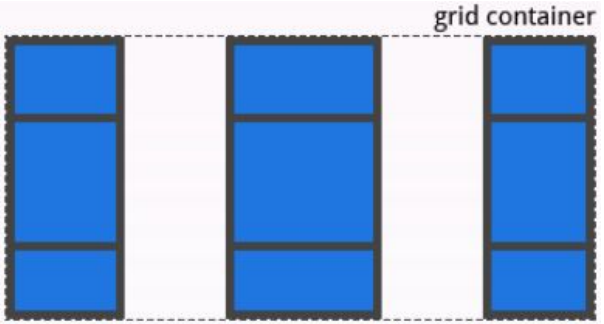
`justify-content: stretch;`



`justify-content: space-around;`



`justify-content: space-between;`



CSS Grid Layout

Propiedades de un contenedor *grid*

- **align-content**. Permite alinear la rejilla dentro del contenedor grid **verticalmente**. Esto es útil cuando el tamaño de los elementos grid se ha definido con unidades no flexibles como px.

```
.contenedor{
  display: grid;
  align-content: start | end | center | stretch |
                 space-around | space-between | space-evenly;
}
```

  - **start**. Alinea el contenido a la parte superior del área grid.
  - **end**. Alinea el contenido a la parte inferior del área grid.
  - **center**. Alinea el contenido en el centro del área grid.
  - **stretch**. Rellena todo el contenido del área grid.
  - **space-around**. Sitúa un número par de espacios entre los elementos grid. Los de los extremos de la columna tiene la mitad de tamaño.
  - **space-between**. Igual que space-around, pero sin espacios en los extremos.
  - **space-evenly**. Igual que space-around, pero incluyendo los de los extremos.

CSS Grid Layout

Propiedades de un contenedor *grid*

- **grid-auto-rows**, **grid-auto-columns**. Especifican el tamaño de las filas/columnas *implícitas*, es decir, aquéllas cuyo tamaño no se ha especificado en las propiedades grid-template-rows y/o grid-template-columns.

```
.contenedor{
  display: grid;
  grid-auto-rows: <tamaño>;
  grid-auto-columns: <tamaño>;
}
```

  - **<tamaño>**. Puede ser una longitud, un porcentaje o una fracción del espacio libre en el contenedor grid (se utiliza la unidad `fr` para esto último).

CSS Grid Layout

Propiedades de un contenedor *grid*

- **grid-auto-flow**. Permite especificar cómo se colocan automáticamente en el grid aquellos elementos grid que no se han colocado específicamente mediante las correspondientes propiedades.

```
.contenedor{
  display: grid;
  grid-auto-flow: row | column | row dense | column dense;
}
```

  - **row**. Rellena cada fila por orden, añadiendo nuevas si es necesario.
  - **column**. Rellena cada columna por orden, añadiendo nuevas si es necesario.
  - **dense**. Intenta rellenar los huecos que queden disponibles en el grid una vez colocados los elementos para los que se ha especificado una ubicación en el grid.

CSS Grid Layout

Propiedades de un contenedor *grid*

- **grid**. Propiedad *shorthand* que permite especificar las propiedades `grid-template-rows`, `grid-template-columns`, `grid-template-areas`, `grid-auto-rows`, `grid-auto-columns` y `grid-auto-flow`. También establece implícitamente los valores de `grid-row-gap` y `grid-column-gap` a su valor inicial.

```
.contenedor{
  display: grid;
  grid: none | <grid-template-rows> / <grid-template-columns> |
        <grid-auto-flow> [<grid-auto-rows> [/<grid-auto-columns>]];
}
```

  - **none**. Establece todas las sub-propiedades a su valor inicial.
  - **<grid-template-rows>/<grid-template-columns>**. Establece las correspondientes propiedades a los valores especificados y el resto de sub-propiedades a sus valores iniciales.
  - **<grid-auto-flow> [<grid-auto-rows> [/<grid-auto-columns>]]**. Establece las correspondientes propiedades a los valores especificados. Si `grid-auto-columns` se omite se le asigna el valor establecido por `grid-auto-rows`. Si se omiten las dos, entonces se establecen a sus valores iniciales.

CSS Grid Layout

Propiedades de los hijos de un contenedor *grid*

- **grid-column-start**, **grid-column-end**, **grid-row-start**, **grid-row-end**. Permite establecer la ubicación de un elemento grid dentro del grid especificando filas y columnas concretas. `grid-column-start / grid-row-start` es la celda del grid en la que empieza y `grid-column-end / grid-row-end` es la celda del grid en la que termina.

```
.elemento{
  grid-column-start: <número> | <nombre> | span <número> | span <nombre> | auto;
  grid-column-end: <número> | <nombre> | span <número> | span <nombre> | auto;
  grid-row-start: <número> | <nombre> | span <número> | span <nombre> | auto;
  grid-row-start: <número> | <nombre> | span <número> | span <nombre> | auto;
}
```

  - **<número>**. Número o un nombre de una fila/columna del grid.
  - **span <número>**. El elemento se expandirá ocupando tantas filas/columnas como indique <número>.
  - **span <nombre>**. El elemento se expandirá hasta la fila/columna llamada <nombre>.
  - **auto**. Indica auto-ubicación: se expandirá automáticamente o por defecto una celda.

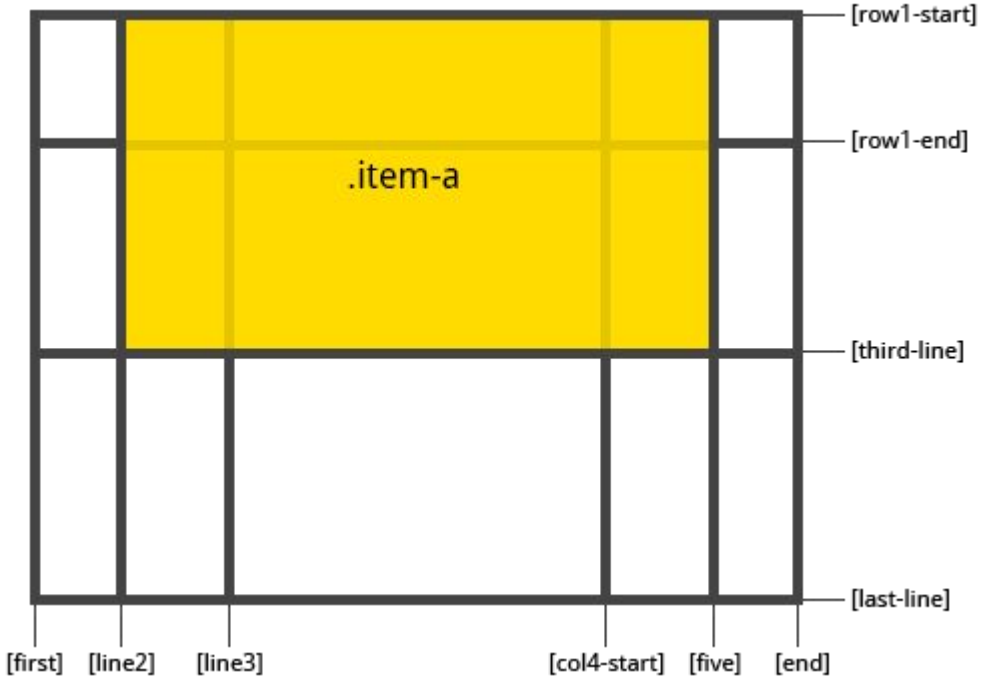
CSS Grid Layout

Propiedades de los hijos de un contenedor *grid*

- **grid-column-start**, **grid-column-end**, **grid-row-start**, **grid-row-end**.

  Ejemplo:

```
.elemento{
  grid-column-start: 2;
  grid-column-end: five;
  grid-row-start: row1-start;
  grid-row-end: 3;
}
```



CSS Grid Layout

Propiedades de los hijos de un contenedor *grid*

- **grid-column**, **grid-row**. Propiedad *shorthand* para `grid-column-start` + `grid-column-end` y `grid-row-start` + `grid-row-end`.

```
.elemento{
  grid-column: <start-line> / <end-line> |
               <start-line> / span <value>;
  grid-row: <start-line> / <end-line> |
            <start-line> / span <value>;
}
```

  - **<start-line> / <end-line>**. Cada uno de los parámetros acepta los mismos valores que en la versión extendida de las propiedades, incluido `span`.

CSS Grid Layout

Propiedades de los hijos de un contenedor *grid*

- **grid-area**. Permite asignar un nombre a un elemento para que pueda ser referenciado más tarde en la propiedad `grid-template-areas`. Alternativamente, esta propiedad se puede utilizar como una propiedad *shorthand*, todavía más corta, de `grid-column-start` + `grid-column-end` y `grid-row-start` + `grid-row-end`.

```
.elemento{
  grid-area: <nombre> | <row-start> /
             <column-start> / <row-end> / <column-end>;
}
```

  - **<nombre>**. Nombre del elemento grid.
  - **<row-start> / <column-start> / <row-end> / <column-end>**. Número o nombre de fila/columna.

CSS Grid Layout

Propiedades de los hijos de un contenedor *grid*

- **justify-self**. Alinea horizontalmente el contenido dentro de un elemento grid.

```
.elemento{
  justify-self: start | end | center | stretch;
}
```

  - **start**. Alinea el contenido a la izquierda del área grid.
  - **end**. Alinea el contenido a la derecha del área grid.
  - **center**. Alinea el contenido en el centro del área grid.
  - **stretch**. Rellena todo el área grid con el contenido.

CSS Grid Layout

Propiedades de los hijos de un contenedor *grid*

- **align-self**. Alinea verticalmente el contenido dentro de un elemento grid.

```
.elemento{
  align-self: start | end | center | stretch;
}
```

  - **start**. Alinea el contenido a la parte superior del área grid.
  - **end**. Alinea el contenido a la parte inferior del área grid.
  - **center**. Alinea el contenido en el centro del área grid.
  - **stretch**. Rellena todo el área grid con el contenido.

Gradientes, transiciones y transformaciones

Gradientes, transiciones y transformaciones

Gradientes CSS

CSS3 permite utilizar gradientes a la hora de especificar colores de fondo, cosa que antes debía hacerse mediante imágenes.

- **linear-gradient**. Permite especificar un gradiente lineal.

  linear-gradient(to *dirección*, *color_parada* [,*color_parada*]+ **)**

  Parámetros:

  - *dirección*: indica la dirección que debe seguir el gradiente. Se puede indicar mediante el par *dir_horizontal dir_vertical*; cuyos valores pueden ser **top**, **right**, **left** y **bottom**. Si se omiten, el valor por defecto es top.
  - La *dirección* también se puede indicar mediante un ángulo, por ejemplo, 45deg, 90deg, -45deg, 180deg, etc ...
  - *color_parada*: Color por el que debe pasar el gradiente. Puede acompañarse de un punto de parada expresado en porcentaje.

Gradientes, transiciones y transformaciones

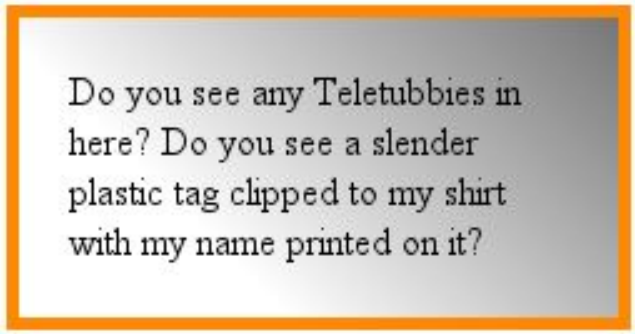
Gradientes CSS

- `linear-gradient`

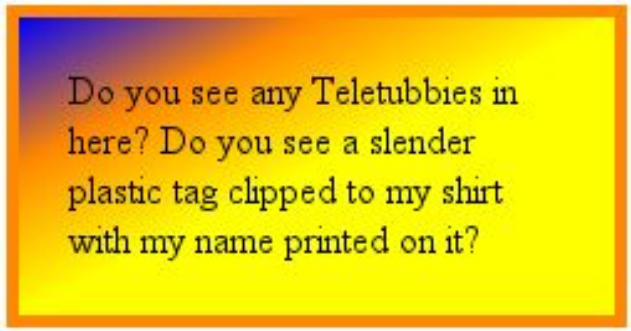
Ejemplo:

```
<p class='bgrad'>Do you see any Teletubbies in here? Do you see a slender plastic tag clipped to my shirt with my name printed on it?</p>
```

```
.bgrad{
  background:linear-gradient(45deg,#fff 30%,#777);
  border:5px solid #f80;
  padding:20px;
  width:200px;
}
```




```
.bgrad{
  background: linear-gradient(to top left,
              #ff0 50%, #f80, #00f);
  border:5px solid #f80;
  padding:20px;
  width:200px;
}
```



Gradientes, transiciones y transformaciones

Gradientes CSS

- `radial-gradient`. Permite especificar un gradiente circular.

  radial-gradient(*forma tamaño* **at** *centro*, *color_parada* [,*color_parada*]+ **)**

  Parámetros:

  - *forma*: forma de la elipse del gradiente. Puede ser **circle** o **ellipse**.
  - *tamaño*: tamaño de la elipse. Los valores que admite son: **closest-side**, **closest-corner**, **farthest-side**, **farthest-corner**. Si la forma es *circle*, este valor indica el radio.
  - *centro*: indica dónde se ubicará el centro del círculo que envuelve el gradiente. Se puede indicar mediante porcentaje, o bien mediante una combinación de **center**, **top**, **right**, **left** y **bottom**. Si se omite, el valor por defecto es center.
  - *color_parada*: Color por el que debe pasar el gradiente. Puede acompañarse de un punto de parada expresado en porcentaje.

Gradientes, transiciones y transformaciones

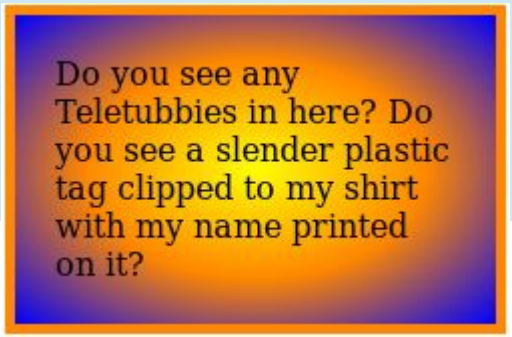
Gradientes CSS

- `radial-gradient`

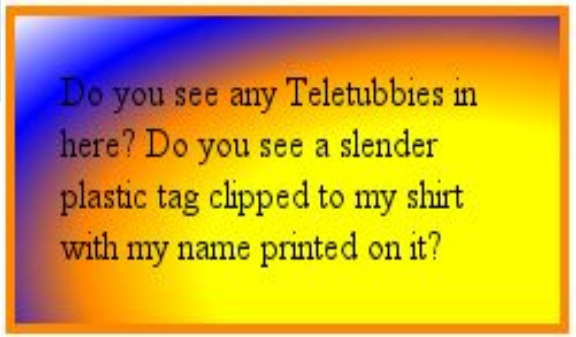
Ejemplo:

```
<div class='bgrad'>Do you see any Teletubbies in here? Do you see a slender plastic tag clipped to my shirt with my name printed on it?</div>
```

```
.bgrad{
  background:radial-gradient(#ff0, #f80, #00f);
  border:5px solid #f80;
  padding:20px;
  width:200px;
}
```




```
radial-gradient(ellipse farthest-corner at 80% 80%,
                #ff0 40%,#f80,#00f,#fff);
```



Gradientes, transiciones y transformaciones

Gradientes CSS

- `conic-gradient`. Permite especificar un gradiente cónico.

  conic-gradient([**from** *ángulo*] [**at** *centro*]**,** *color_parada* [,*color_parada*]+ **)**

  Parámetros:

  - *ángulo*: ángulo de rotación del gradiente. El valor por defecto es **`0deg`**.
  - *centro*: indica dónde se ubicará el centro del gradiente. Se puede indicar mediante porcentaje, o bien mediante una combinación de **center**, **top**, **right**, **left** y **bottom**. Si se omite, el valor por defecto es center.
  - *color_parada*: Color por el que debe pasar el gradiente. Puede acompañarse de un punto de parada expresado en porcentaje.

Gradientes, transiciones y transformaciones

Gradientes CSS

- `conic-gradient`

Ejemplo:

```
<div class="bgrad">
</div>
```

```
.bgrad{
  background: conic-gradient(white -50%, black 150%);
  border:5px solid #f80;
  padding:20px;
  width:100px;
  height:100px;
}
```

```
.bgrad2{
  background: conic-gradient(yellowgreen 0deg 40%, gold 0deg 75%, #f06 0deg);
  border-radius: 50%;
  border:5px solid #f80;
  width: 100px;
  height: 100px;
}
```

Gradientes, transiciones y transformaciones

Transiciones CSS

Las transiciones CSS permiten que los cambios en los valores de las propiedades CSS se hagan de manera suave y en un intervalo de tiempo determinado. Consta de varias propiedades CSS:

- **`transition-property`**. Especifica el nombre de la propiedad CSS a la que aplicar la transición. Se puede indicar más de una propiedad CSS separándolas con ",". También admite los valores **`none`** y **`all`**.
- **`transition-duration`**. Especifica el intervalo de tiempo (en segundos) en el que se hará la transición.
- **`transition-timing-function`**. Permite especificar cómo se aplicará la transición, pudiendo cambiar la velocidad durante el intervalo de tiempo. Admite numerosos valores y posibilidades: **`ease`**, **`linear`**, **`ease-in`**, **`ease-out`**, **`ease-in-out`**, **`cubic-bezier(n,n,n,n)`**.
- **`transition-delay`**. Define el tiempo de demora de comienzo de la transición.

Gradientes, transiciones y transformaciones

Transiciones CSS

Ejemplo:

```
<p class="textoS">Este texto cambia de color al pasar el ratón</p>
```

```
.textoS{
   color: #00f; /* azul */
   transition-property: color; /* sólo se aplica a la propiedad color */
   transition-duration: 2s; /* el cambio de color durará 2 segundos */
   transition-timig-function: linear; /* la transición será linear */
   transition-delay: 0.5s; /* comenzará pasados 0.5 segundos */
}
.textoS:hover{
   color: #f00; /* rojo */
}
```

Gradientes, transiciones y transformaciones

Transiciones CSS

Existe una última propiedad que define todas las características de la animación en una única línea:

- **`transition`**. Combina las cuatro propiedades anteriores en una única propiedad.

```
transition: tp td ttf tD [, tp td ttf tD]*
```

Parámetros:

- ***tp***: valor de la propiedad *transition-property*.
- ***td***: valor de la propiedad *transition-duration*.
- ***ttf***: valor de la propiedad *transition-timing-function*.
- ***tD***: valor de la propiedad *transition-delay*.

En el caso del ejemplo anterior, definir la transición se hubiese reducido a escribir:

```
...
   transition: color 2s linear 0.5s;
...
```

Gradientes, transiciones y transformaciones

Transiciones CSS

- **transition**. A la hora de especificar los parámetros de la transición se pueden omitir valores para las propiedades, lo que hará que se tomen los valores por defecto. Al omitir valores, el primer valor numérico que encuentre lo interpretará como *transition-duration* y el segundo valor numérico (si lo hubiera) como *transition-delay*.

Propiedad	Valor por defecto
`transition-property`	**`all`** (todas las propiedades)
`transition-duration`	**`0s`** (cero segundos, no hay transición)
`transition-timing-function`	**`ease`**
`transition-delay`	**`0s`** (cero segundos, empieza inmediatamente)

```
/* Aplica transición al color durante 2 segundos */
transition: color 2s;
```

```
/* Define dos transiciones: una para el ancho (width) y otra para
   la transparencia (opacity) */
transition: width 3s, opacity 2s 3s ease-in;
```

Gradientes, transiciones y transformaciones

Transformaciones

- CSS3 permite transformaciones 2D y 3D mediante la propiedad **`transform`**.

  `transform:` *función_transformación* [*función_transformación*]*

  Donde *función_transformación* es una de las posibles funciones de transformación 2D o 3D.

- Junto a la propiedad **`transform`** se suele utilizar la propiedad **`transform-origin`** que permite especificar el punto de referencia para realizar la transformación.

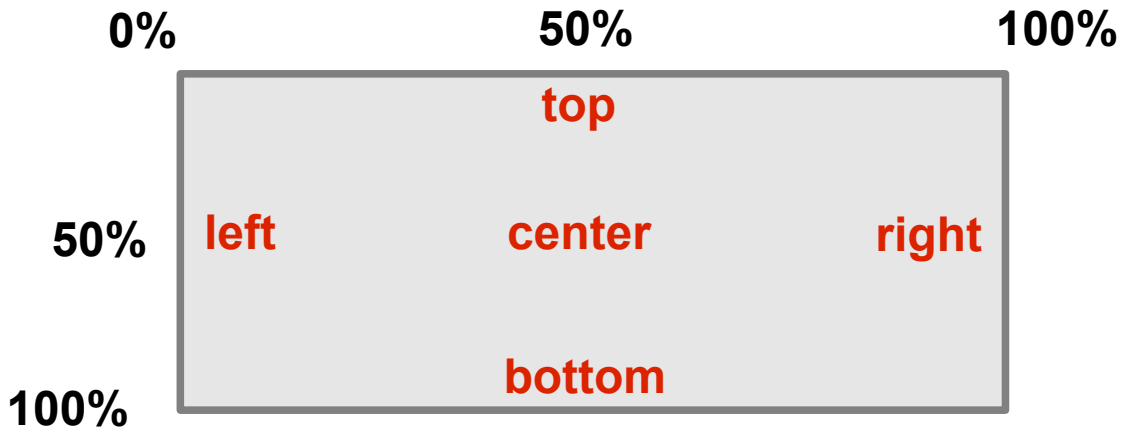
  `transform-origin:` *posición_x* [*posición_y*]

  Donde *posición_x* y *posición_y* pueden tomar los valores **`center`**, **`left`**, **`top`**, **`bottom`**, **`right`**; o bien, valores porcentuales. Todos estos valores se establecen con referencia al centro del objeto.

Gradientes, transiciones y transformaciones

Transformaciones

- `transform-origin:`
  - Los parámetros *posición_x* y *posición_y* también admiten valores absolutos del tipo 30px, 100pt, etc; para los que se tomará como referencia la esquina superior izquierda del objeto.
  - Si sólo se especifica un valor, éste se toma como *posición_x* y *posición_y* se deja en el centro del objeto.
  - Por defecto el origen de la transformación es el centro del elemento.



Gradientes, transiciones y transformaciones

Transformaciones

- **Transformaciones 2D**:
  - `translate( tx [, ty] )`

Especifica una traslación 2D mediante el vector de desplazamiento (*tx*,*ty*), donde *tx* se corresponde con la translación horizontal y *ty* con la translación vertical. Si sólo hubiera un valor para el par (*tx*,*ty*), éste se tomaría como *tx* y a *ty* se le asignaría el valor 0.

```
div.dtns{
  transform: translate(25%,50%);
}
```

```
<div class="dtns">Do you see any
Teletubbies in here? Do you see a
slender plastic tag clipped to my
shirt with my name printed on
it?</div>
```

Gradientes, transiciones y transformaciones

Transformaciones

- **Transformaciones 2D**:
  - `rotate( ángulo )`

Especifica una rotación 2D en el sentido de las agujas del reloj. Los valores que admite *ángulo* son del tipo: 30deg, 45deg, 90deg, -60deg, -180deg, etc...

```
div.dtns{
  transform: rotate(-30deg);
  transform-origin: right top;
}
```

```
<div class="dtns">Do you see any
Teletubbies in here? Do you see a
slender plastic tag clipped to my
shirt with my name printed on
it?</div>
```

Gradientes, transiciones y transformaciones

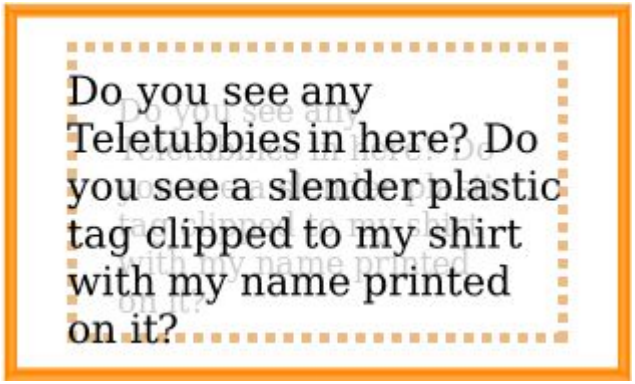
Transformaciones

- **Transformaciones 2D**:
  - `scale( sx [,sy] )`

Escala el tamaño del objeto utilizando como factor de escalado el indicado por (*sx*,*sy*). Si sólo se especifica un valor, éste se utilizará para *sx* y para *sy*. Por ejemplo, un factor de escalado de (1,1) dejaría al objeto con el mismo tamaño; mientras que un factor de escalado (2,2) duplicaría el tamaño del objeto en alto y ancho.

```
div.dtns{
  transform: scale(1.25,1.25);
}
```

```
<div class="dtns">Do you see any
Teletubbies in here? Do you see a
slender plastic tag clipped to my shirt
with my name printed on it?</div>
```



Gradientes, transiciones y transformaciones

Transformaciones

- **Transformaciones 2D**:
  - `skew(` *ángulo_x*`,` *ángulo_y*`)`

Inclina el objeto en los ejes ***x*** e ***y*** según el ángulo especificado por *ángulo_x* y *ángulo_y*, respectivamente. Los valores que admiten los ángulos son del tipo: `30deg`, `-45deg`, `90deg`, `-180deg`, etc...

```
div.dtns{
  transform: skew(35deg, 5deg);
  transform-origin: 50% 0%;
}
```

```
<div class="dtns">Do you see any
Teletubbies in here? Do you see a
slender plastic tag clipped to my
shirt with my name printed on
it?</div>
```

Gradientes, transiciones y transformaciones

Transformaciones

- **Transformaciones 3D**:
  - Es una mejora de las transformaciones CSS vistas, que permite hacerlas en un espacio tridimensional.
  - Se utilizan, igualmente, las propiedades `transform` y `transform-origin`, pero utilizando un sistema de coordenadas tridimensional (x,y,z). Además, se tienen las siguientes propiedades:
    - ✓ `transform-style`. Indica cómo se renderizan los elementos en 3D.
    - ✓ `perspective`. Indica la perspectiva de los elementos en 3D.
    - ✓ `perspective-origin`. Indica el punto inferior de los elementos en 3D en la perspectiva (perspectiva del usuario).
    - ✓ `backface-visibility`. Indica si para un elemento que se esté transformando en 3D es visible su parte trasera.

Gradientes, transiciones y transformaciones

Transformaciones

- **Transformaciones 3D**:

Las funciones de transformación 3D son las siguientes:

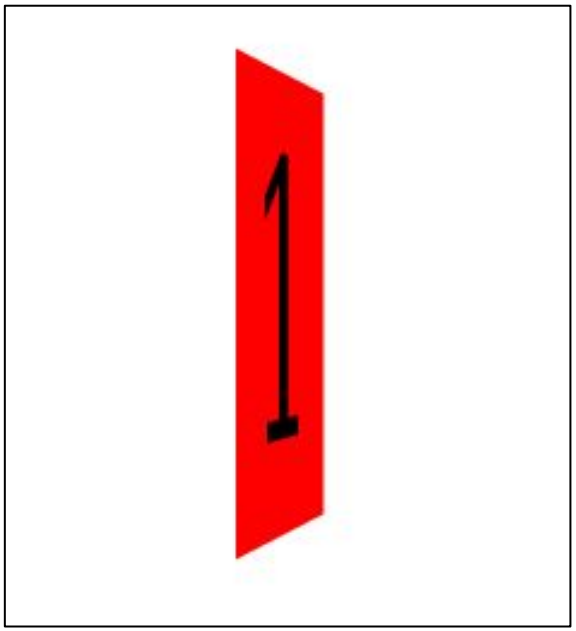
- ✓ `matrix3d(n,n,...,n)`. Define una matriz 3D de transformación, utilizando una matriz 4x4 de 16 valores.
- ✓ `translate3d(x,y,z)`, `translateX(x)`, `translateY(y)`, `translateZ(z)`. Definen translaciones en 3D.
- ✓ `scale3d(x,y,z)`, `scaleX(x)`, `scaleY(y)`, `scaleZ(z)`. Define escalados en 3D.
- ✓ `rotate3d(x,y,z,angle)`, `rotateX(x)`, `rotateY(y)`, `rotateZ(z)`. Define rotaciones en 3D.

Gradientes, transiciones y transformaciones

Transformaciones

- **Transformaciones 3D**:

```
<div id="card">
  <div class="front">1</div>
  <div class="back">2</div>
</div>
```




```
#card {
  height: 12em;
  perspective: 800px;
  position: relative;
  transform-style: preserve-3d;
  width: 10em;
}
#card div {
  transition:all 1s ease-in;
  backface-visibility: hidden;
  font-size: 10em;
  text-align: center;
  height: 100%;
  width: 100%;
}
#card .front{ background: red; }
#card .back{
  top:0;
  background: blue;
  position: absolute;
  transform: rotateY( -180deg );
}
#card:hover .front{ transform: rotateY( 180deg ); }
#card:hover .back{ transform: rotateY( 0deg ); }
```

Animaciones CSS

Animaciones CSS

- Gracias a las *animaciones CSS*, las transiciones de un estilo CSS a otro sobre un elemento se puede configurar proporcionándoles un efecto.
- Para poder aplicar una animación sobre un elemento sólo hay que realizar dos cosas:
  - Definir el conjunto de pasos, o ***keyframes***, de la animación.
  - Definir el estilo que se aplicará al elemento y que le asignará la animación definida mediante los ***keyframes***.

Animaciones CSS

Ventajas de utilizar animaciones CSS

- ✓ Son fáciles de usar para crear animaciones simples sin necesidad de conocer JavaScript.
- ✓ Las animaciones corren fluidas, incluso bajo una moderada carga de sistema, cosa que en JavaScript no suele ocurrir.
- ✓ Al dejar que sea el navegador el que controla la animación, éste se encargará de que su funcionamiento esté optimizado y sea eficiente.

Animaciones CSS

Configuración básica de la animación

- ✓ La creación de los distintos *keyframes* de la animación se hace mediante la directiva CSS `@keyframes`.
- ✓ Cada *keyframe* será como un *fotograma* de la animación y en él hay que indicar cuál es el estilo que tendrá el objeto en ese momento.
- ✓ Los keyframes están ordenados, indicando el punto de la animación en el que se aplicarán, mediante porcentaje. Así pues, el primer keyframe estará asociado al **0%** y el último al **100%**. Se pueden utilizar como alias de estos dos valores **`from`** y **`to`**, respectivamente.

Animaciones CSS

Configuración básica de la animación

El uso de la directiva CSS **@keyframes** es el siguiente:

```
@keyframes nombre_de_la_animación{
    0% { // estilo a aplicar al inicio de la animación
        ...
    }
    20% { // estilo a aplicar en el 20% de la animación
        ...
    }
    ...
    100% { // estilo a aplicar al final de la animación
        ...
    }
}
```

La animación debe tener un nombre al que luego se pueda hacer referencia a la hora de asignarla a un elemento del documento.

Animaciones CSS

Configuración básica de la animación

Una vez construida la animación sólo queda crear la regla CSS que haga uso de ella y configure la forma en que se aplicará.

Las principales propiedades CSS de las que disponemos para configurar cómo aplicar la animación son las siguientes:

- `animation-name`: especifica el nombre de la secuencia de keyframes a aplicar, es el nombre que acompaña a la directiva **`@keyframes`**.
- `animation-duration`: especifica el tiempo que la animación debe tardar en ejecutar un ciclo completo.
- `animation-timing-function`: especifica la animación de la transición a aplicar entre keyframes, estableciendo curvas de aceleración. Algunos de los valores admitidos son: `ease`, `ease-in`, `ease-out`, `ease-in-out` y `linear`.

Animaciones CSS

Configuración básica de la animación

Principales propiedades CSS de la animación:

- **`animation-direction`**: permite especificar si la animación debe alternar la dirección en la que recorre los keyframes en cada ejecución, o bien ir directamente al inicio y repetirse. Los valores que admite son: `normal`, `alternate`, `reverse`, `alternate-reverse`.
- **`animation-delay`**: especifica el tiempo que debe esperar la animación para iniciarse.
- **`animation-iteration-count`**: especifica el número de veces que se debe repetir una animación antes de pararse. Los valores que admite son: `infinite`, o un número mayor o igual que 1. Este número admite valores decimales, por ejemplo, un valor 0.5 haría que la animación se parara cuando llegara al 50%.

Animaciones CSS

Configuración básica de la animación

Como muchas otras propiedades CSS, la configuración de la animación se puede resumir y hacerla en una única regla CSS:

```
animation an-Nam an-Dur an-Tim-Func an-Del an-It-Co an-Dir
```

Donde:

- ***an-Nam***: valor de la propiedad ***animation-name***.
- ***an-Dur***: valor de la propiedad ***animation-duration***.
- ***an-Tim-Func***: valor de la propiedad ***animation-timing-function***.
- ***an-Del***: valor de la propiedad ***animation-delay***.
- ***an-It-Co***: valor de la propiedad ***animation-iteration-count***.
- ***an-Dir***: valor de la propiedad ***animation-direction***.

Animaciones CSS

Configuración básica de la animación

Ejemplo:

```
<img class="rotar" id="logo" src="logo100.png" alt="Logo">
```

```
@keyframes rotacion360 {
    0% {
        transform: rotateY(0);
    }
    100% {
        transform: rotateY(360deg);
    }
}
.rotar{
    animation: rotacion360 2s linear infinite alternate;
}
```