Fundamentos HTML, CSS

Introducción a las hojas de estilo con CSS3

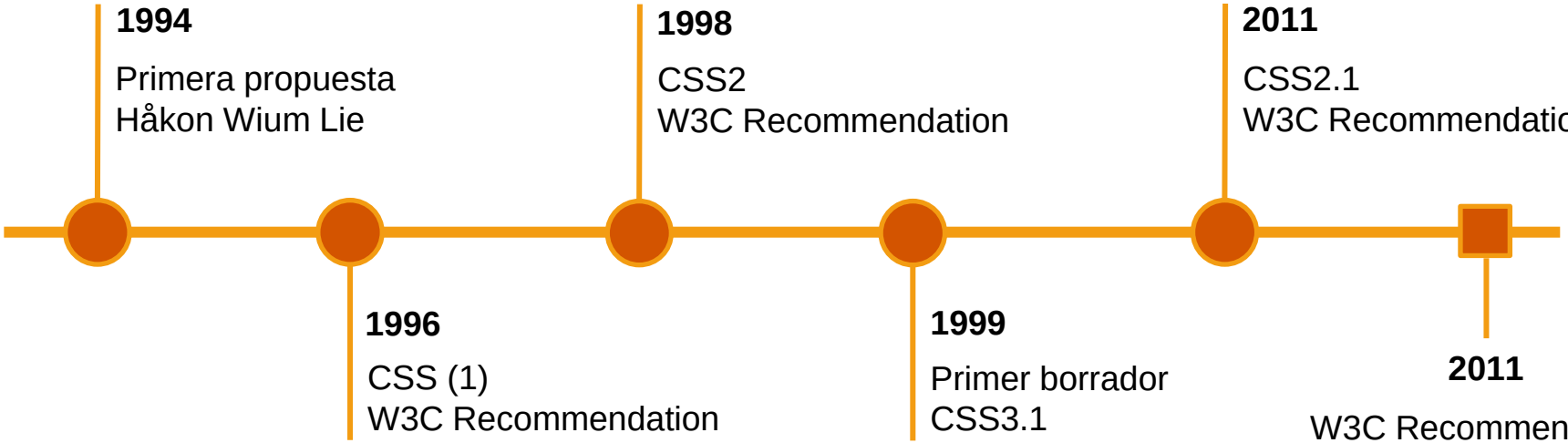
Tony G. Bolaño
@tonybolanyo – tonybolanyo@gmail.com
KeepCoding Full Stack Web Developer
Noviembre 2020

Introducción a CSS3

¿Qué es CSS?

- CSS: Cascading Style Sheets (hojas de estilo en cascada).
- Lenguaje para definir la presentación de un documento estructurado escrito en un lenguaje de marcado.
- Diseñado para permitir la separación entre el contenido del documento y la forma de presentarlo.

Un poco de historia



Fuente: https://en.wikipedia.org/wiki/HTML

¿Cómo usar CSS?

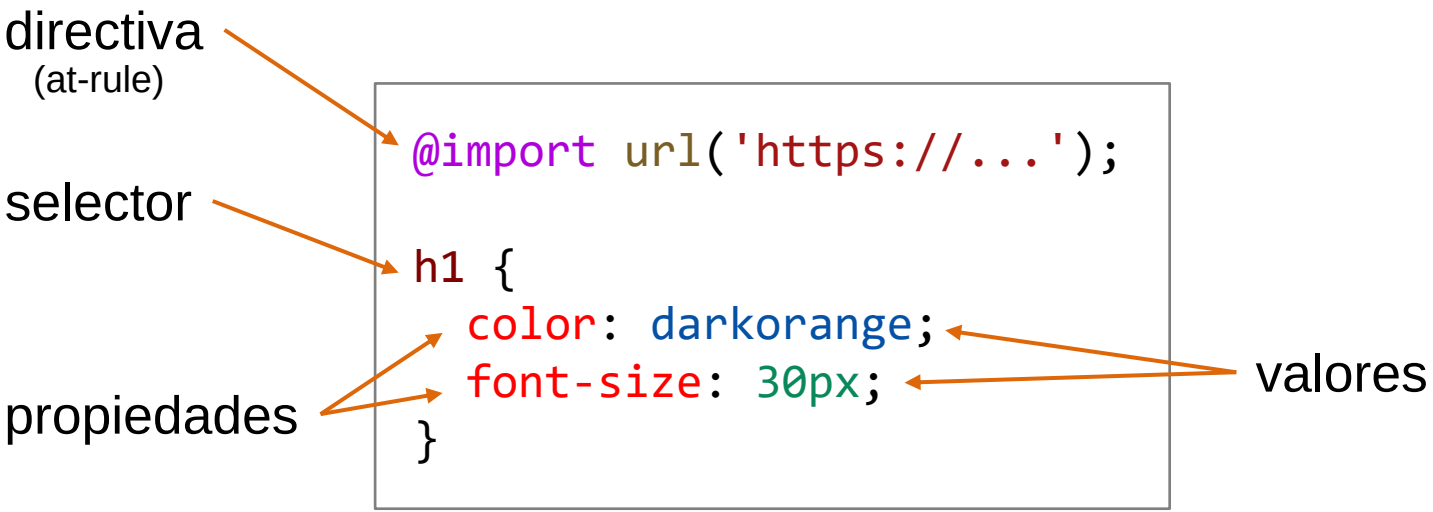
- Estilos en línea, mediante el atributo `style`:

```
<p style="propiedad: valor">
```

- Estilos independientes. Uso de selectores:
  - Embebidos con la etiqueta `<style>`
  - En ficheros CSS independientes

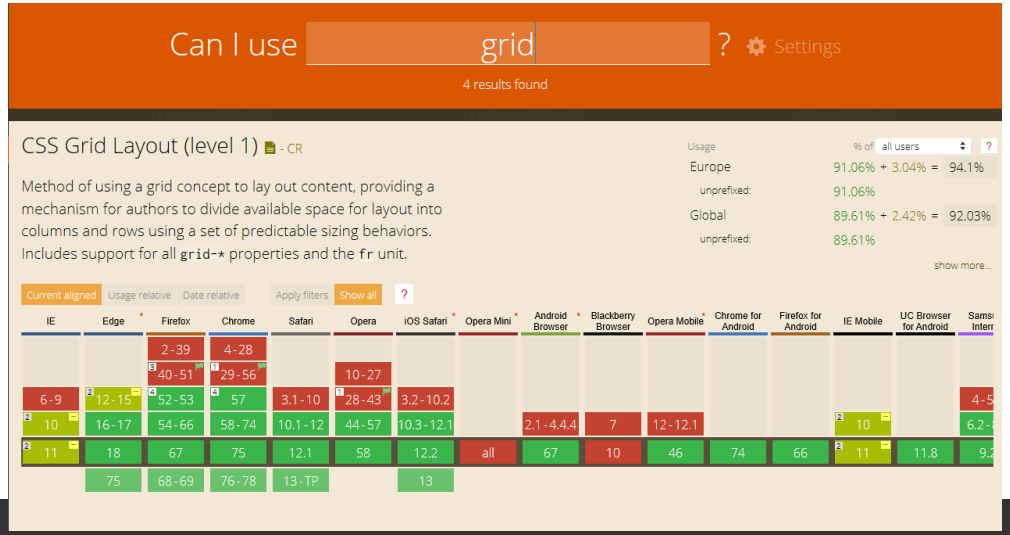
```
<link rel="stylesheet" href="./styles.css">
```

Sintaxis: selectores, propiedades y directivas

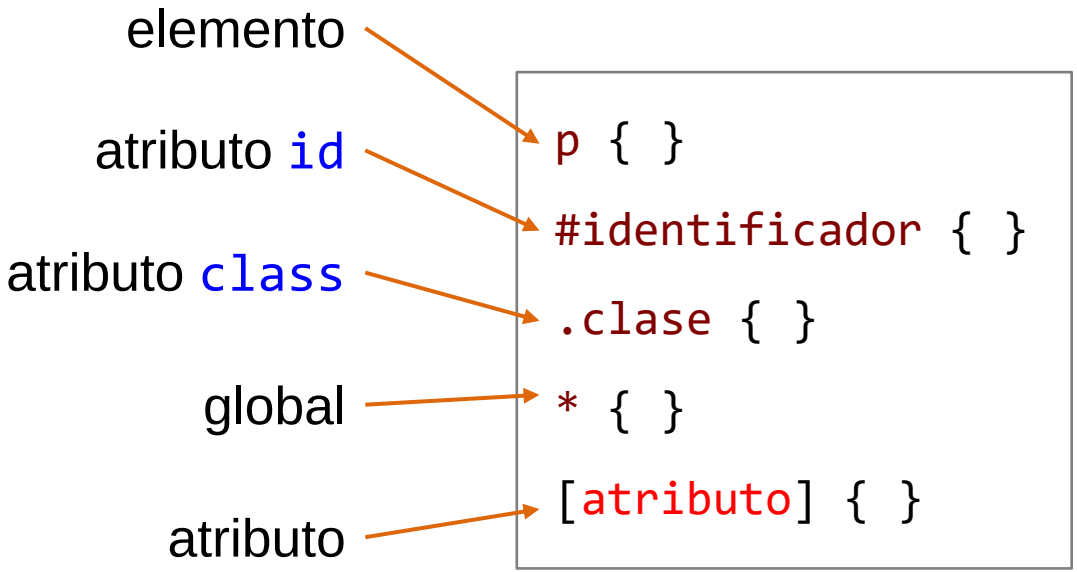


Herramientas y referencias

- MDN: https://developer.mozilla.org/en-US/docs/Web/CSS/Reference
- Can I Use: https://caniuse.com
- Navegador: inspector



Selectores básicos




```
a[href$=".org"] {
font-style: italic;
}
```

`[atributo]`
elemento que tienen el atributo

`[atributo="valor"]`
elementos con ese valor

`[atributo^="valor"]`
elementos que comienzan con el valor

`[atributo$="valor"]`
elementos que terminan con ese valor

`[atributo*="valor"]`
elementos que contienen el valor

`[atributo|="valor"]`
valor exacto o principio seguido de guión

`[atributo~="valor"]`
lista de palabras y una es valor

Selectores múltiples y anidados

`h1, h2, h3 { }`	• Aplica estilos a múltiples selectores
`div p { }`	• Descendientes de cualquier nivel
`div > p { }`	• “Hijos” directos
`div + p { }`	• “Hermanos” adyacentes (inmediatos)
`div ~ p { }`	• “Hermanos” en general

Pseudo-elementos y Pseudo-clases

- **Pseudoelementos**: elementos definidos desde CSS.
  - `::first-letter`
  - `::first-line`
  - `::before {content=""}`
  - `::after {content=""}`

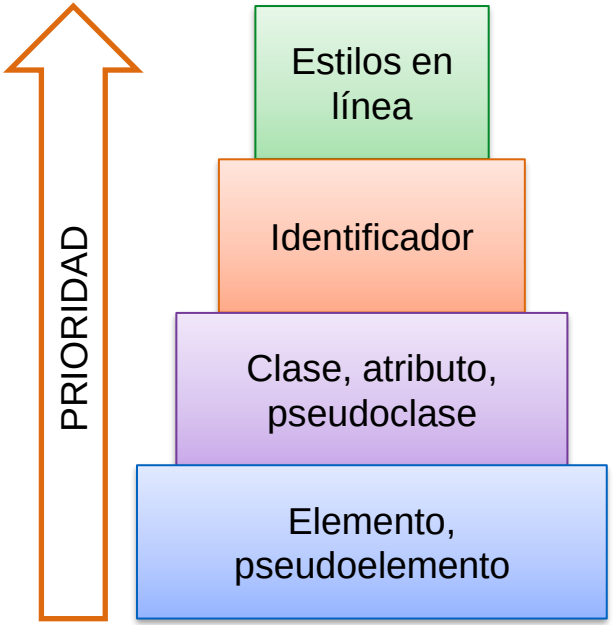
- **Pseudoclases**: clases aplicadas dinámicamente a elementos reales de HTML en función de su estado
  - `:link, :visited, :active, :focus :hover, :target, :lang()`
  - `:nth-child(n), :first-child, :last-child, :only-child`
  - `:nth-of-type(n), :first-of-type, :last-of-type,`
  - `:only-of-type, :not()`

Aplicación de múltiples estilos

- **Herencia**
  - Propiedades que se heredan en los elementos hijos
    `font-family`, `color`… => afectan al contenido
  - Propiedades que no se heredan
    `margin`, `padding`, `border`… => afectan al contenedor
- **Cascada**
  - agente de usuario: por defecto / definidos por el usuario
  - bloques de CSS: externos / embebidos
  - estilos en línea
- **Especificidad**

Especificidad

- Orden de prioridad (menor a mayor especificidad):
  - Selectores de tipo (`p`, `h1`) y pseudo-elementos (`::before`, `::after`)
  - Selectores de clase (.ejemplo), selectores de atributos (`[type="text"]` y pseudo-clases (`:hover`)
  - Selectores ID (`#ejemplo`)
- Los estilos inline (`style="color: darkorange"`) tienen prioridad sobre los externos



Unidades de medida

UNIDADES ABSOLUTAS	
Unidad	**Medida**
`in`	Pulgada (2.54cm)
`cm`	Centímetro
`mm`	Milímetro
`pt`	Punto. 1pt = 1/72in
`pc`	Pica. 1pc=12pt
`px`	Relativos a la pantalla

UNIDADES RELATIVAS	
Unidad	**Relativo a**
`em`	Tamaño de letra del elemento
`ex`	Altura de la “x” del elemento
`vw`	Anchura del viewport
`vh`	Altura del viewport
`%`	% del tamaño del elemento
`rem`	Tamaño de letra en el root `<html>`

```
.container {
  font-size: 2rem;
  width: 800px;
  height: 100vh;
}
```

Variables. Uso de `calc()`

- Variables definidas globalmente:

```
:root {
  --color-principal: #06c;
}
```

- Se utilizan mediante la función `var()`

```
#foo h1 {
  color: var(--color-principal);
}
```

- La función `calc()` opera con variables y números

```
.container {
  --separacion: 20;
  margin-top: calc(var(--separacion) * 2px);
}
```

Propiedades básicas

Tipografía

- Podemos incluir nuevas fuente de varias formas:
  - Mediante la propiedad `@font-face`

```
@font-face {
  font-family: miFuente;
  src: url();
}
```

  - Importándola de webs como Google Fonts

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

```
<link href='' type='text/css'>
```

Propiedades para fuentes

Propiedad	Significado
`font-style`	Define el estilo de un texto. Normal, italic,…
`font-variant`	Variante del tipo de letra. Normal, small-caps
`font-weight`	Peso de la letra. Normal, bold, 100, 900,…
`font-size`	Tamaño de la fuente.
`font-family`	Familia de la fuente
`font`	Abreviatura de todas las anteriores en el mismo orden de aparición

Otras propiedades:

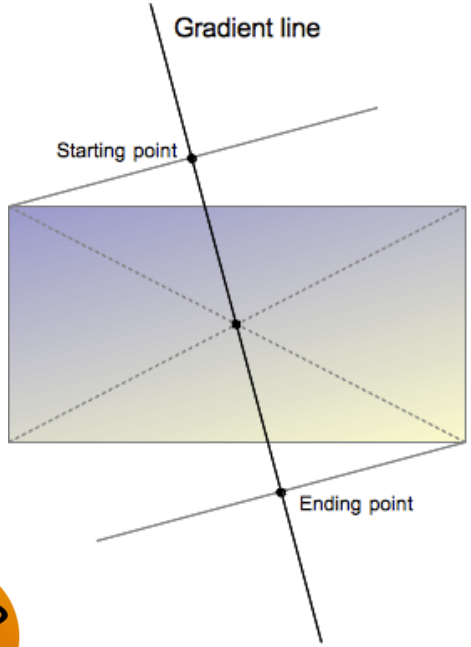
- `text-decoration, text-transform`
- `line-height, letter-spacing, word-spacing`
- `text-align, vertical-align, text-indent`

Colores

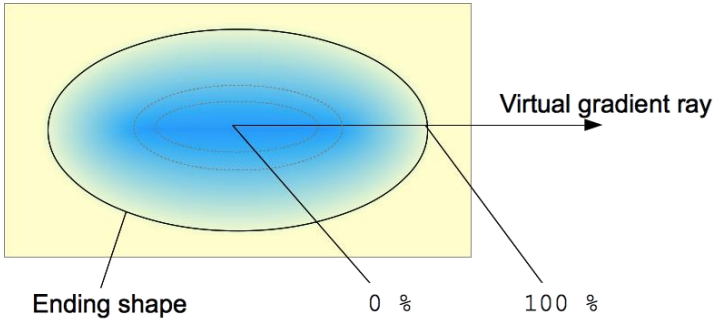
- Sistemas de colores
  - Nombres
  - rgb() - red, green, blue
  - hexadecimal (#)
  - hsl()
  - hueº (matiz), saturation%, luminosity%
- Transparencia canal alfa entre 0 (transparente) y 1 (opaco)
  - rgba()
  - hsla()
  - Opacity / se hereda

Degradados

```
background: linear-gradient(
  dirección, color-stop1,
  color-stop2, color-stop-n);
```




```
background: radial-gradient(
  figura at posición,
  color-stop1, color-stop2,
  color-stop-n);
```



Sombras

- Sombra de caja:

```
.container {
  box-shadow: offset-x offset-y blur-radius spread-radius color
}
```

- Sombra de texto:

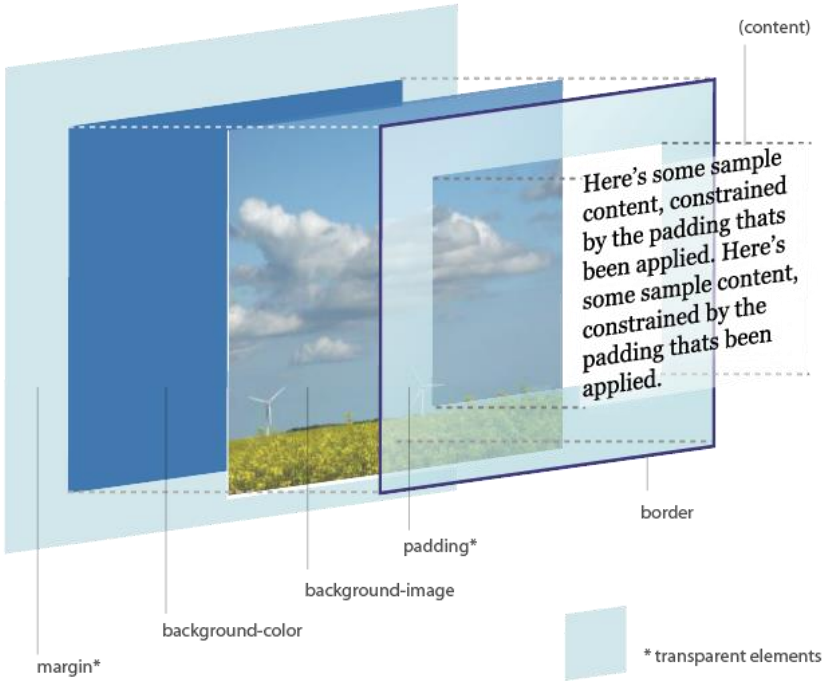
```
.foo {
  text-shadow: offset-x offset-y blur-radius color
}
```

Backgrounds: imágenes y patrones para el fondo

```
background-image: url(./assets/about-bg.jpg);
background-repeat: no-repeat;
background-size: cover;
background-position: center center;
background-attachment: fixed;
background-clip: border-box;
background-blend-mode: normal;
```

Modelo de caja

Modelo de caja en CSS



- Contenido
- Espaciado interno (padding)
- Borde
- Imagen de fondo
- Color de fondo
- Margen

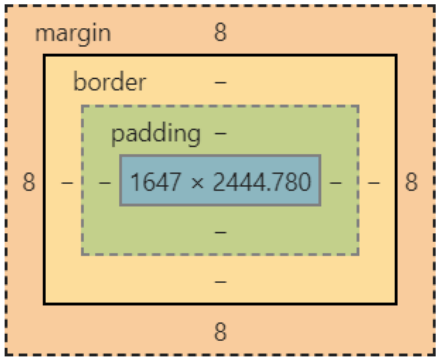
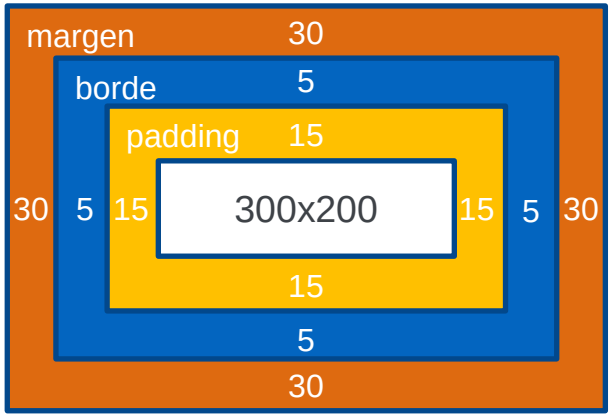


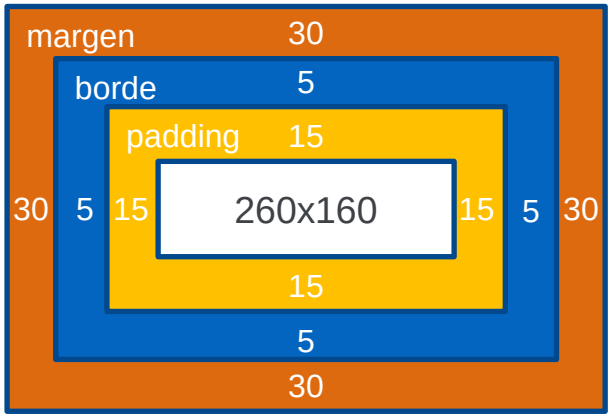
Imagen: https://hicksdesign.co.uk/boxmodel/

Modelo de caja. Sistemas de medida

box-sizing: content-box;



box-sizing: border-box;




```
.container {
  width: 300px;
  height: 200px;
  padding: 15px;
  border: 5px solid gray;
  margin: 30px;
}
```

Modelo de caja. Propiedades y usos.

- Propiedades
  - margin
  - padding
  - border
  - border-radius
  - border-image
- Usos
  - wrappers
  - centrado de bloques

Centrado de bloques

- Horizontal:
  - width: %
  - margin-left / margin-right: auto
- Vertical:
  - line-heigth = height (elementos de 1 linea)
  - display table
  - position absolute 50% / 50% margin -w/2 -h/2
  - position absolute 50% / 50% transform translate -50% -50%

Alternativa: display: flex

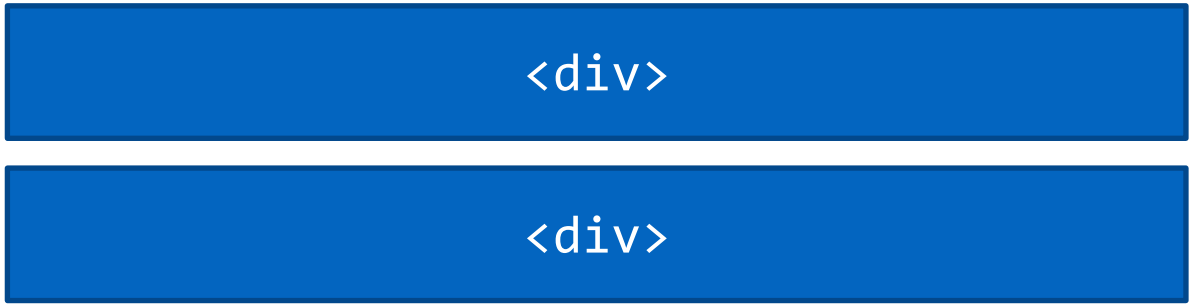
Display y posicionamiento

Display. Opciones básicas

- La propiedad `display` en la más importante para controlar la estructura de la web.
- Cada elemento HTML tiene un valor por defecto.
- Algunos elementos de “nivel bloque” son: `<div>`, `<h1>`…`<h6>`, `<p>`, `<form>`, `<section>`
- Algunos elementos de “nivel inline” son: `<span>`, `<a>`, `<img>`

display: block

- Un elemento block comienza en una nueva linea y se expande en el eje horizontal tanto como pueda.



display: inline

- Un elemento *inline* continua en la misma línea que el resto del contenido.
- No respeta ni márgenes, ni los paddings **top** ni **bottom**.

Esto es `<span>un elemento</span>` inline.

display: inline-block

- Se comporta igual que un elemento *inline*, pero a diferencia de él, puede tener tanto un **width** como un **height**.
- También respeta los márgenes y los paddings.

Esto es `<span>un elemento</span>` inline.

display: none

- Oculta el elemento sin dejar espacio en el sitio que debería ocupar.
- Usado por algunos elementos como la etiqueta <script>.

Posicionamiento

- La propiedad `position` nos permite crear estructuras más complejas en nuestros diseños.
- Por defecto, los navegadores posicionan todos los elementos de forma automática.
- Puede tomar lo valores:
  - `static` (por defecto): sin posicionamiento específico.
  - `relative`: se posiciona como static y luego se desplaza
  - `absolute`: referencia al ancestro más cercano (no static).
  - `fixed`: variante del absoluto. Su posición es inamovible.

Posicionamiento

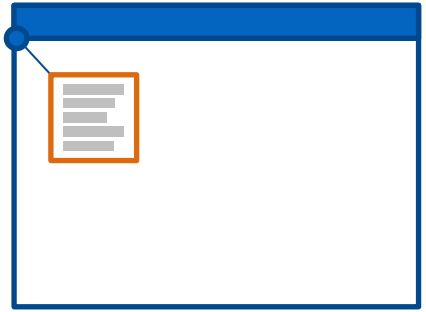
position: static;

ESTÁTICO O
NO POSICIONADO

position: relative;

RELATIVO

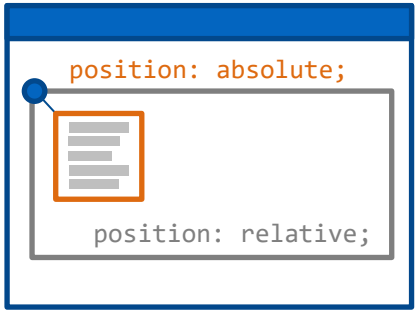
position: absolute;



ABSOLUTO

position: fixed;

FIJO



(RELATIVAMENTE) ABSOLUTO

Cambios del flujo estático

- Float: left / right
- Clear: left / right / both

- Overflow: hidden / auto

Técnica de clear-fix con ::after

```
::after {
  content : "";
  display: block;
  clear: both;
}
```

The Last Great Time War. My people fought a race called the Daleks, for the sake of all creation. And they lost. We lost. Everyone lost.

I'm the Doctor, I can save the world with a kettle and some string! And look! I'm wearing a vegetable!

"Cuando un elemento tiene a todos sus hijos flotando, pierde su altura"

Transiciones y transformaciones

Transiciones

- Las transiciones nos permiten realizar cambios sobre las propiedades CSS de forma suave de el estado inicial al estado final. No son animaciones.

- En una transición podemos especificar la propiedad que queremos que cambie (width), el tiempo de la transición (2s), y la curva de aceleración de la transición (linear).

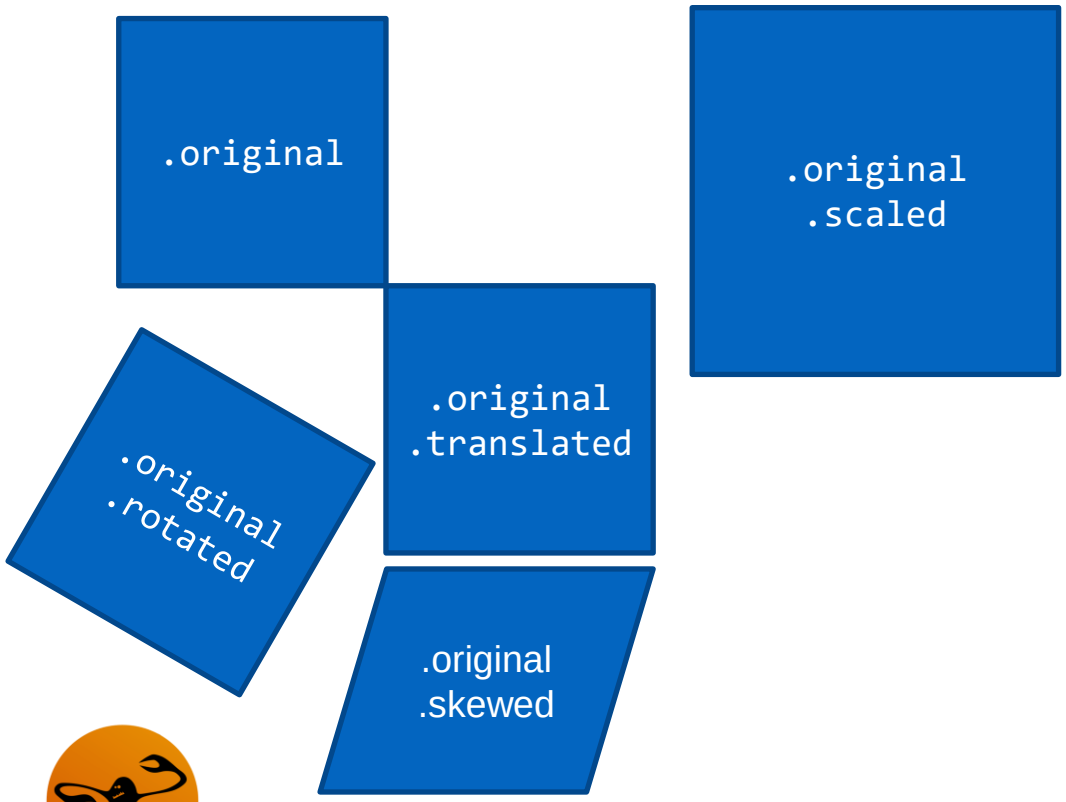
```
<style>
  .class  {
    border: 2px solid black;
    transition: width 2s ease-in,
        border-color 2s linear 1s;
  }

  .class  {
    width: 200px;
    border-color: red;
  }
</style>
```

Transformaciones

- Nos permiten modificar las coordenadas en las que se encuentra un elemento.
- Los elementos pueden ser:
  - Trasladados: `translate`, `translateX`, `translateY`
  - Rotados: `rotate`, `rotateX`, `rotateY`
  - Escalados: `scale`, `scaleX`, `scaleY`
  - Distorsionados: `skew`, `skewX`, `skewY`

Transformaciones




```css
.original {
  width: 200px;
  height: 200px;
}

.translated {
  transform: translate(206px);
}

.rotated {
  transform: rotate(45deg);
}

.scaled {
  transform: scale(1.5, 1.5)
}

.skewed {
  transform: skewX(-10deg);
}
```

Animaciones

Animaciones

- Nos permiten cambiar de un estilo a otro de forma elegante.
- Podemos cambiar tantas propiedades de CSS como queramos y tantas veces como queramos
- Las reglas de la animación las vamos a definir mediante `@keyframes`
- Se deben usar a través de la propiedad `animation`.
- Son muy fáciles de crear, incluso sin conocimientos de JS.
- Las animaciones se muestran correctamente incluso en equipos poco potentes.
- Al ser el navegador el que maneja la secuencia de animación, nos permite optimizar el rendimiento y eficacia de la misma

Cómo construir una animación

Keyframes (cuadros clave)	Propiedades de la animación
Define las fases y estilos de la animación	Asigna los *keyframes* a un elemento CSS y define *cómo* se anima

Cómo construir una animación

Keyframes (cuadros clave)

Define las fases y estilos de la animación

- Nombre de la animación
- Fases de la animación (%)
- Propiedades CSS

Cómo construir una animación

```
@keyframes rebote {
  0% {
    transform: scale(0.1);
    opacity: 0;
  }
  60% {
    transform: scale(1.2);
    opacity: 1;
  }
  100% {
    transform: scale(1);
  }
}
```

- Nombre de la animación
- Fases de la animación (%)
- Propiedades CSS

Cómo construir una animación

```
@keyframes rebote {
  0% {
    transform: scale(0.1);
    opacity: 0;
  }
  60% {
    transform: scale(1.2);
    opacity: 1;
  }
  100% {
    transform: scale(1);
  }
}
```

Propiedades de la animación

Asigna los *keyframes* a un elemento CSS y define *cómo* se anima

`animation-name` | `animation-duration`

Cómo construir una animación

```css
@keyframes rebote {
  0% {
    transform: scale(0.1);
    opacity: 0;
  }
  60% {
    transform: scale(1.2);
    opacity: 1;
  }
  100% {
    transform: scale(1);
  }
}
```

```css
div {
  animation-duration: 1.5s;
  animation-name: rebote;
}

/* Forma abreviada */
div {
  animation: rebote 1.5s;
}
```

Configurando la animación

- `animation-name`: nombre de la animación
- `animation-delay`: retardo con el que empezará la animación
- `animation-duration`: duración de la animación
- `animation-fill-mode`: estado final de la animación
- `animation-iteration-count`: nº de repeticiones o `infinite`
- `animation-timing-function`: establece curvas de aceleración
- `animation-direction`: indica si la dirección debe ser invertida en cada iteración o volver al inicio
- `animation-play-state`: permite pausar y reanudar la animación

Animación en forma abreviada

```
animation: [animation-name] [animation-duration]
  [animation-timing-function] [animation-delay]
  [animation-iteration-count] [animation-direction]
  [animation-fill-mode] [animation-play-state];
```

Definiendo la secuencia

```
@keyframes rainbow {
  0% {
    background-color: darkcyan;
  }
  50% {
    background-color: darkorange;
  }
  75% {
    background-color: darkviolet;
  }
}
```

```
@keyframes slidein {
  from {
    margin-left: 100%;
    width: 300%;
  }

  to {
    margin-left: 0%;
    width: 100%;
  }
}
```

GRACIAS

www.keepcoding.io