CSS3

Introducción a CSS

CSS es el acrónimo de ***"Cascading Style Sheets"***, que significa **Hojas de Estilo en Cascada**. Es el lenguaje utilizado para dar estilo y formato a un documento HTML. CSS describe como los elementos deben renderizarse en una pantalla, en un hoja impresa o cualquier otro medio.

Se le denomina **en cascada** porque se lee, procesa y aplica el código desde arriba hacia abajo.

Anteriormente, el desarrollo de varias partes de la especificación CSS se hizo de forma paralela, lo que permitió que hubiera una versión de las últimas recomendaciones como por ejemplo CSS1, CSS2.1, o incluso CSS3.

Nunca habrá una versión CSS4. Más bien, todo es ahora código CSS sin un número de versión ya que CSS se divide en módulos independientes que se desarrollan a diferente ritmo.

Las especificaciones de CSS se encuentran en W3C specifications.

El documento base oficial de CSS en W3C.

Integración de estilos CSS

Estilos integrados en un elemento HTML

La regla CSS se aplica directamente sobre el elemento HTML. De esta forma la regla **sólo** se aplica al elemento HTML y no puede reutilizarse:

```
<p>Lorem <span style="color: red; padding: 8px">ipsum dolor</span></p>
```

Estilos definidos en el documento HTML

Las reglas CSS se definen en un documento HTML. Por tanto se aplican a **todo** el documento HTML únicamente. Se utiliza la etiqueta `<style>` dentro de la etiqueta `<head>` :

```
<!DOCTYPE html>
<html>
<head>
  <title>...</title>
  <style>
    div {
      background: red;
      color: white;
    }
  </style>
</head>
  ...
</html>
```

Estilos definidos en un archivo CSS externo

Las reglas CSS se definen en un **archivo CSS externo**. Estas reglas se podrán reutilizar en todos los documentos HTML que incluyan el archivo CSS.

⚠️ El atributo `type="text/css"` es opcional ya que el tipo de las hojas de estilo por defecto es CSS.

```
<!DOCTYPE html>
<html>
<head>
  <title>...</title>
  <link rel="stylesheet" href="index.css" />
</head>
  ...
</html>
```

Estilos importados

Una forma de integrar una hoja de estilo CSS es utilizar la regla `@import`, regla implementada en CSS2.

Esta regla permite importar una hoja de estilo CSS **dentro** de otra hoja de estilo CSS.

```
/* index.css */
@import url("otra-hoja.css")
```

También se puede utilizar dentro de un documento HTML aunque por rendimiento este método no es recomendable:

```
<!DOCTYPE html>
<html>
<head>
  <title>...</title>
  <style>
    @import url("otra-hoja.css");
  </style>
</head>
  ...
</html>
```

Aplicación de los estilos CSS

Más información en el sitio oficial del W3C

Noción de herencia

En la aplicación de los estilos, la noción de herencia es importante cuando existen elementos anidados.

La regla de la herencia indica que los elementos hijos **pueden heredar** propiedades del elemento padre.

Algunas propiedades de estilo en CSS se heredan de forma predeterminada, mientras que otras no. Las propiedades que se heredan son aquellas que afectan la apariencia del texto y algunas propiedades relacionadas con el modelo de caja.

Algunos ejemplos de propiedades heredadas son `font-family`, `color`, `font-size`, `line-height` o `text-align`.

Por otro lado, algunas propiedades no se heredan de forma predeterminada, como por ejemplo `border`, `margin`, `padding`, `width` o `height`.

```css
/* Estilo para el elemento padre */
.padre {
  font-family: Arial, sans-serif;
  color: #333;
}

/* Los elementos hijos heredan los estilos del elemento padre */
.padre .hijo {
  font-size: 16px;
  line-height: 1.5;
}
```

Noción de cascada

Cuando existe un conflicto de aplicación de reglas, es decir, que tenemos varias reglas que afectan a un mismo elemento HTML, se aplica el concepto de **cascada** (sin olvidar que CSS es el acrónimo de ***"Cascading Style Sheets"***).

El orden de prioridad de las reglas es el siguiente, de **mayor a menor prioridad**:

1. Reglas en línea (dentro del elemento HTML)
2. Reglas en la cabecera del documento (dentro de la sección `<head>`)
3. Reglas en un fichero externo css
4. Reglas CSS por defecto aplicadas por el navegador

En el caso de que haya diferentes reglas CSS que apliquen con el mismo nivel de prioridad, como por ejemplo dentro del mismo fichero css, tendrá más prioridad **la última regla definida** ya que sobreescribe al resto.

Cálculo de la especificidad

La **especificidad** en CSS es un concepto que determina qué conjunto de reglas se aplicará a un elemento en caso de que haya conflictos.

La especificidad se calcula asignando valores a los selectores utilizados en las reglas CSS.

Cuanto más específico es un selector, mayor es su especificidad y más peso tiene sobre otras reglas que puedan aplicarse al mismo elemento.

La especificidad se mide en cuatro niveles, de menor a mayor:

1. **Especificidad de elemento**: Es el nivel más bajo y se refiere a los selectores de elementos. **Su valor es 1**.
2. **Especificidad de clase o pseudoclase**: Se refiere a los selectores de clase, pseudoclases y atributos. **Su valor es 10**.
3. **Especificidad de ID**: Se refiere a los selectores de ID. **Su valor es 100**.
4. **Especificidad de estilo en línea**: Es el nivel más alto y se refiere a las reglas de estilo en línea directamente en el elemento. **Su valor es 1000**.

Para el cálculo de la especificidad se tiene en cuenta la suma de todos los selectores de la regla:

```css
/* Tiene una especificidad de 1 (por elemento) */
p {
  color: black;
}
```

```css
/* Tiene una especificidad de 11 (1 por elemento + 10 por clase) */
p.test {
  color: black;
}

/* Tiene una especificidad de 21 (1 por elemento + 10 por clase + 10 por clase) */
p.test1.test2 {
  color: black;
}

/* Tiene una especificidad de 100 (por identificador) */
#demo {
  color: black;
}

/* Tiene una especificidad de 101 (100 por identificador + 1 por clase) */
p#demo {
  color: black;
}
```

Más información en w3schools.com

Noción de importancia

Cuando aplican las reglas CSS, intervienen el concepto de herencia y el cálculo de especificidad.

Sin embargo, es posible ignorar el orden de prioridad marcando una regla CSS como `!important`.

De hecho, al utilizar la regla `!important`, anulará TODAS las reglas de estilo anteriores para esa propiedad específica en ese elemento.

```css
#myid {
  background-color: blue;
}

.myclass {
  background-color: gray;
}

/* Todos los parráfos '<p>' tendrán fondo rojo, aunque tengan la clase o ID anteriores */
p {
  background-color: red !important;
}
```

Más información en w3schools.com

Estructura de una regla de estilo

Un estilo CSS se contruye con una **regla** que consta de un **selector** y una **declaración**.

El **selector** indica el ámbito de aplicación del estilo, como puede ser un elemento HTML.

La **declaración** indica la propiedad o propiedades CSS utilizadas y se encierra entre llaves.

Cada **propiedad** utiliza uno o varios **valores**. Se utiliza el `:` para separar la propiedad de sus valores. Cada línea de separación acaba con un `;`.

```
selector {
  propiedad: valor;
}

otro-selector {
  propiedadA: valor;
  propiedadB: valor1 valor2;
  propiedadC: valor;
}
```

El nombre de los selectores debe respetar una **nomenclatura**:

- No debe empezar por una cifra
- No debe contener espacios, caracteres acentuados ni caracteres especiales como +, *, /, etcétera…
- Puede contener guiones `-` o guiones bajos `_` en el nombre
- Es sensible a mayúsculas y minúsculas por lo que el nombre del selector debe ser idéntico tanto en el CSS como en el HTML

Comentarios

Los comentarios en CSS empiezan por `/*` y terminan con `*/`. Se pueden situar en una sola línea o al final de cualquier línea:

```
/** Comentario */
html {
  background-color: bisque;  /* Comentario */
}

body {
  /* Comentario de
  varias líneas */
  color: blue;
}
```

Unidades de medida

Hay muchas propiedades que utilizan valores numéricos como por ejemplo el tamaño de la fuente.

⚠️ Cuando un valor es 0 **no se indica la unidad** de medida aunque es válido y no genera error:

```
body {
  margin: 0px; /* Poco útil aunque válido */
}
```

Hay propiedades que pueden utilizar tres valores textuales:

- **initial**: indica a los navegadores que es necesario utilizar el valor por defecto de la propiedad.
- **inherit**: informa de que el valor a utilizar es el valor del elemento HTML padre del elemento indicado.
- **unset**: especifica que el valor será *"inherit"* en aquellas propiedades que sean heredables como `color` o el valor será *"initial"* para aquellas propiedades que no sean heredables como `border`.

Más información en el sitio oficial del W3C

Valores numéricos

Hay propiedades que utilizan valores numéricos de tipo **entero** llamados *integer* en las propiedades CSS.

Otras propiedades utilizan valores **decimales** llamados *number* en las propiedades CSS. Como separador decimal se utiliza el punto `.`.

Por último tenemos propiedades que utilizan **porcentajes**. El símbolo utilizado es el `%`.

```
body {
  margin: 10px;
  line-height: 12.5px;
  padding: 2%;
}
```

Valores calculados

Algunos valores pueden depender de cálculos donde intervengan la longitud de la pantalla, el tamaño del carácter de un elementro padre, etcétera…

Para realizar los cálculos, el W3C ofrece la función `calc()`. Existen otras funciones relacionadas con expresiones matemáticas como `max()` o `min()`.

```
#div1 {
  position: absolute;
  left: 50px;
  width: calc(100% - 100px);
  border: 1px solid black;
  background-color: yellow;
  padding: 5px;
  text-align: center;
}
```

Más información en css-tricks.com

Unidades de longitud

Hay propiedades CSS como `"margin"`, `"width"` o `"padding"` que utilizan valores de longitud o *"length"* en las especificaciones del W3C.

Estos valores se dividen en dos grupos:

- valores **relativos**:
  - **em**: para indicar el tamaño de los caracteres respecto al padre
  - **rem**: para indicar el tamaño de los caracteres respecto al elemento raíz `<html>` del documento web.
  - **ex** da el tamaño de los caracteres respecto a la altura de la minúscula del carácter x (poco utilizado).
  - **vw** asigna el valor proporcional a la longitud de la pantalla
  - **vh** asigna el valor proporcional a la altura de la pantalla

- valores **absolutos**:
  - **cm** para los centímetros
  - **mm** para los milímetros
  - **in** para las pulgadas (1 pulgada = 2,54cm)
  - **pc** para las picas (1 pica = 0,4233cm)
  - **pt** para los puntos (1 punto = 0,0352cm)

Unidades de ángulos

Para las unidades de **ángulos** tenemos:

- **deg** para los grados
- **grad** para los gradientes
- **rad** para los radianes
- **turn** para un giro completo, es decir, 360º

Unidades de tiempo

Para las unidades de **tiempo**:

- **s** para los segundos
- **ms** para los milisegundos

Colores

Existen varias posibilidades para utilizar los colores en CSS:

- **notación nominativa** (por su nombre)
- **notación hexadecimal**
- **notación RGB** (o en su versión con transparencias RGBA)
- **notación HSL** (o en su versión con transparencias HSLA)

Más información en el sitio oficial del W3C

Notación nominativa

En un inicio había únicamente 16 colores que se podían utilizar por nombre. Actualmente hay **140 colores** reconocidos por los navegadores.

Más información en w3schools.com

Notación hexadecimal

Históricamente los navegadores sólo reconocían 256 colores en su forma hexadecimal. En la actualidad reconocen cualquier color en su formato hexadecimal.

La notación usa el prefijo `#` y a continuación los tres componentes de color **Rojo, Verde y Azul** es decir `#RRGGBB`.

Los valores van desde **00** hasta **FF** en hexadecimal (0-255 en decimal). Por ejemplo el color blanco se expresa como `#ffffff` y el color negro se expresa como `#000000`.

Esta notación no es sensible a mayúsculas ni minúsculas. Además, en el caso de caracteres dobles puede introducirse uno solo. Por ejemplo `#00FF00` equivaldría a `#0F0`.

Más información en w3schools.com

Notación RGB y RGBA

Los diseñadores gráficos utilizan **la notación RGB** ya que suele ser la notación utilizada en las aplicaciones de diseño gráfico.

En esta notación, cada color se expresa con la forma `rgb(red, green, blue)`.

Cada parámetro (*red, green y blue*) define la intensidad del color entre **0 y 255**.

Para mostrar el color negro se utiliza `rgb(0, 0, 0)` mientras que para un color blanco se utiliza `rgb(255, 255, 255)`.

En la **notación RGBA** tenemos un cuarto componente que es la transparencia (de *alpha*).

Este valor va desde 0 para indicar **total transparencia** hasta **1** para indicar **total opacidad**. Todos los valores intermedios se muestran en decimal. Por ejemplo, un rojo semitransparente sería `rgb(255, 0, 0, 0.5)`.

Más información en w3schools.com

Notación HSL y HSLA

Otra notación utilizada por diseñadores gráficos es la **notación HSL** acrónimo de *Hue, Saturation y Lightness* (Tinta, Saturación y Luminosidad).

La tinta es la noción de color, la saturación indica la cantidad de gris en la tinta y la luminosidad especifica la cantidad de luz.

En esta notación, cada color se expresa con la forma `hsl(hue, saturation, lightness)`.

La tinta es una rueda cromática desde el 0 hasta el 360, donde el 0 es rojo, 120 es verde y 240 es azul.

La saturación es un porcentaje. Va desde el 0% sin color y solo con gris hasta el 100% para un color puro sin gris.

La luminosidad también es un porcentaje. Va desde el 0% para el negro (por lo tanto sin luz) hasta el 100% para el blanco, con luz total.

Se puede agregar el componente **transparencia** con la **notación HSLA**. El **0** indica transparencia mientras que el **1** indica opacidad. Todos los valores intermedios se muestran en decimal.

Más información en w3schools.com

Selectores

En una regla de estilo CSS, el selector indica sobre qué elemento HTML se va a aplicar la declaración.

Los selectores se dividen en cinco categorías:

- **Selectores simples** basados en su nombre, clase o id
- **Selectores de combinación** basados en una relación específica entre ellos
- **Selectores de pseudo-clases** basados en un determinado estado
- **Selectores de pseudo-elemento** para seleccionar una única parte de un elemento
- **Selectores de atributos** para seleccionar elementos según su atributo o su valor

Más información en el sitio oficial del W3C

Selectores simples

Selector universal

Este selector universal indicado por `*` permite apuntar a todos los elementos HTML de un documento y aplicar un estilo a toda la página.

```
/* Esta regla CSS se aplica a todos los elementos HTML de la página */
* {
  margin: 0;
  padding: 0;
}
```

Selector de tipo

Los selectores de tipo o también llamados selectores de elemento permiten seleccionar un elemento HTML específico.

Este selector hace uso simplemente del nombre del elemento HTML, como por ejemplo `<div>`, `<span>`, `<p>`, `<stront>`, etcétera…

```
p {
  color: rgb(255, 0, 0);
}
```

Selector de clase

El selector de clase permite seleccionar elementos HTML que tengan el mismo atributo de clase.

⚠ El nombre de una clase **no puede empezar con un número**.

El nombre del selector empieza **siempre** por `.` seguido del nombre de la clase:

```
/* Esta regla se aplica a todos los elementos con 'class="center"' */
.center {
  text-align: center;
  color: red;
}
```

También se puede especificar que sólo determinados elementos HTML deben ser afectados por una clase:

```css
/* Esta regla se aplica solo a los elementos <p> con 'class="center"' */
p.center {
  text-align: center;
  color: red;
}
```

Un elemento HTML puede tener varias clases como atributo, por lo que se verá afectado por todas ellas:

```html
<!-- El elemento <p> se verá afectado por las reglas de las clases 'center' y 'large' -->
<p class="center large">This paragraph refers to two classes.</p>
```

Selector de identificación

Un selector de identificación permite crear un estilo que solo se aplicará a un **único elemento HTML** identificado gracias al valor de su atributo `id`. Esto es así debido a que cada identificador debe ser **único** en cada documento.

⚠️ El nombre usado como identificador **no puede empezar con un número**.

El nombre del selector empieza **siempre** por el carácter almohadilla `#` seguido del identificador:

```css
/* Esta regla se aplicará al elemento HTML cuyo 'id="para1"' */
#para1 {
  text-align: center;
  color: red;
}
```

Selector de agrupación

Para minimizar el uso de reglas, se pueden **agrupar** distintos selectores que apliquen los mismos estilos. Por ejemplo:

```css
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
  color: red;
}

p {
  text-align: center;
  color: red;
}
```

Dado que las tres definiciones de estilos son iguales, podemos agruparlas en una única definición. Para ello agrupamos los selectores separándolos con una `,`:

```css
h1, h2, p {
  text-align: center;
  color: red;
}
```

Selectores de atributos

Los selectores de atributo van a permitir apuntar a los elementos HTML que utilizan un atributo específico a nivel del nombre o de los valores.

El nombre del atributo se indica entre corchetes `[atributo]`.

Se puede utilizar únicamente el nombre del atributo como por ejemplo `[lang]` o combinarlo con un elemento HTML como por ejemplo `p[lang]`.

```
/* Selecciona todos los elementos <a> con el atributo 'target' sin importar su valor */
a[target] {
  background-color: yellow;
}

/* Selecciona todos los elementos de cualquier tipo con el atributo 'title' sin importar su valor */
[title] {
  background-color: yellow;
}

/* Selecciona todos los elementos con el atributo 'target="_blank"' */
[target="_blank"] {
  background-color: yellow;
}

/* Selecciona todos los elementos con el atributo 'title' y que contengan
la palabra indicada como valor y separada por espacios de otras palabras */
[title~="flower"] {
  background-color: yellow;
}
/* The example above will match elements with title="flower", title="summer flower"
 and title="flower new", but not title="my-flower" or title="flowers" */

/* Selecciona todos los elementos con el atributo 'lang' cuyo valor sea exactamente 'es'
o seguido de un guión como 'es-es' o 'es-ca' */
[lang|="es"] {
  background-color: yellow;
}

/* Selecciona todos los elementos <a> con el atributo 'href' que comiencen con 'https' */
a[href^="https"] {
  background-color: yellow;
}

/* Selecciona todos los elementos <a> con el atributo 'href' que finalicen con '.pdf' */
a[href$=".pdf"] {
  background-color: yellow;
}

/* Selecciona todos los elementos <a> con el atributo 'href' que contengan la cadena 'w3schools' */
a[href*="w3schools"] {
  background-color: yellow;
}
```

Más información en w3schools.com

Selector de pseudo-clase

Las pseudo-clases permiten apuntar a elementos que no son accesibles con los selectores clásicos y que toman un **estado particular**, como por ejemplo los enlaces visitados.

En lo que respecta a la sintaxis, una pseudo-clase empieza siempre por `:` y va seguida del nombre de la pseudo-clase.

```
selector:pseudo-class {
  property: value;
}
```

Las pseudoclases se pueden combinar con clases HTMl, por ejemplo:

```
/* Sólo aplica al elemento <a class="highlight"> y no al resto de <a>*/
a.highlight:hover {
  color: #ff0000;
}
```

Más información en w3schools.com

Pseudo-clases de enlaces

- :link: selecciona los enlaces no visitados
- :visited: selecciona los enlaces visitados (por razones de seguridad los navegadores sólo aceptan algunos formatos CSS)

Pseudo-clase de acciones del usuario

- :hover: cuando se pasa el ratón por encima
- :active: cuando el usuario activa un elemento
- :focus: cuando el elemento obtiene el "foco"

Pseudo-clase de ancla

- :target: seleccionar el destino del enlace

Pseudo-clase de idioma

- :lang(language): selecciona el elemento con el atributo "lang" y el valor indicado

Pseudo-clase de estado

- :enabled: selecciona los elementos con el estado activado
- :disabled: selecciona los elementos con el estado desactivado
- :checked: selecciona los elementos seleccionados
- :optional: selecciona los elementos sin el atributo "required"
- :valid: cuando tiene un valor válido
- :invalid: cuando tiene un valor no válido
- :in-range: cuando el elemento tiene un valor dentro del rango
- :out-range: cuando el elemento tiene un valor fuera del rango
- :placeholder: selecciona los elementos con el atributo "placeholder"

- :read-only: selecciona los elementos con el atributo "readonly"
- :read-write: selecciona los elementos sin el atributo "readonly"

Pseudo-clase de estructura

- :root: selecciona la raíz del documento web (`:root` > `html`)
- :first-child: selecciona el primer hijo del elemento padre
- :last-child: selecciona el último hijo del elemento padre
- :nth-child(n)
- :nth-last-child(n)
- :first-of-type
- :last-of-type
- :nth-of-type(n)
- :nth-last-of-type(n)
- :only-child
- :only-of-type
- :empty
- :not(selector)

Selector de pseudo-elemento

Los pseudo-elementos se utilizan para seleccionar una parte del elemento, como por ejemplo la primera letra o la primera linea, etcétera…

En lo que respecta a la sintaxis, un pseudo-elemento empieza siempre por `::` y va seguida del nombre del pseudo-elemento:

```
selector::pseudo-element {
  property: value;
}
```

⚠ La notación con doble `::` fue introducida por W3C para CSS3. En CSS1 y CSS2 se utiliza la misma notación `:` que para las pseudo-clases.
Por compatibilidad se puede utilizar las dos notaciones.

- ::after: insertar contenido después con `content`
- ::before: insertar contenido antes con `content`
- :first-letter: seleccionar la primera letra
- ::first-line: seleccionar la primera línea

- ::marker: selecciona el "marker" de una lista de items
- ::selection: selecciona el texto seleccionado por el usuario con el ratón

Más información en w3schools.com

Combinación de selectores

Con la combinación de selectores simples se puede explicar la relación entre selectores.

Más información en w3schools.com

Combinación descendente

Permite seleccionar todos los elementos HTML que son descendientes del primer elemento HTML indicado por el selector, independientemente del nivel de profundidad.

Esta combinación se establece con un espacio entre los selectores.

En este ejemplo se seleccionan todos los elementos `<p>` descendientes de `<div>`, sin importar si son hijos directos o son hijos de otras etiquetas:

```
div p {
  background-color: yellow;
}
```

Combinación de hijos

Permite seleccionar los elementos HTML que son hijos directos del primer elemento HTML indicado por el selector.

Esta combinación se establece con el símbolo `>` entre los selectores.

En este ejemplo se seleccionan todos los elementos `<p>` que son hijos directos de `<div>`:

```
div > p {
  background-color: yellow;
}
```

Combinación de hermanos adjacentes

Esta combinación permite seleccionar un elemento HTML que sigue inmediatamente a otro elemento HTML, es decir, que son "hermanos", lo cual significa que están al mismo nivel.

Esta combinación se establece son el signo `+` entre los selectores.

Esn este ejemplo se selecciona el primer elemento `<p>` que es hermano de la etiqueta `<div>` y por tanto están al mismo nivel:

```
div + p {
  background-color: yellow;
}
```

Combinación de hermanos

Esta combinación permite seleccionar todos los elementos HTML que son "hermanos" de otro elemento HTML, sin importar si va a continuación.

Esta combinación se establece con el caracter "virgulilla" `~` entre los selectores.

Esn este ejemplo se seleccionan todos los elementos `<p>` que son hermanos de la etiqueta `<div>`:

```
div ~ p {
  background-color: yellow;
}
```

Enlaces de interés

- https://www.w3.org/Style/CSS/
- https://www.w3.org/Style/CSS/specs.en.html
- https://lenguajecss.com/css/
- https://developer.mozilla.org/en-US/docs/Web/CSS
- https://web.dev/learn/css?hl=es
- https://htmlcheatsheet.com/css/
- https://overapi.com/css
- https://cheatsheets.shecodes.io/css
- https://cssreference.io/
- https://www.w3schools.com/css/
- https://css-tricks.com/almanac/ -- https://css-tricks.com/guides/
- https://devhints.io/
- https://www.theodinproject.com/
- https://roadmap.sh/frontend
- https://jsfiddle.net

Licencia