Internet: Interconnected y Network  
Red de computadoras conectadas en todo el mundo.

Tim Berners-Lee Invento World Wide Web.  
Fundo el consorcio W3C para estandarizar el desarrollo de las tecnologías base Internet

* HTTP Hyper Text Transfer Protocol 😕/ (Permite la comunicación de datos entre dispositivos)
* URL Uniform Resource Locator: (Dirección de un sitio web)
* HTML Hyper Text Markup Language (Lenguaje para describir la estructura de un sitio web)

En 1994 apareció CSS Cascade Style Sheets.

* **HTML**: Es un lenguaje de marcado usado para decirle a tu navegador cómo estructurar las páginas web que visitas. No es un lenguaje de programación.
* **CSS**: Es un lenguaje que nos permite crear páginas web con un diseño agradable para los usuarios. Tampoco es un lenguaje de programación.
* **DOM**: Document Object Model. Es una transformación del código HTML escrito por nosotros a objetos entendibles para el navegador.
* **CSSOM**: así como el DOM para el HTML, EL CSSOM es una representación de objetos de nuestros estilos en CSS.
* **RENDER TREE**: es la unión entre el DOM y el CSSOM para renderizar todo el código de nuestra página web.

Pasos que sigue el navegador para construir las páginas web:

1. Procesa el HTML para construir el DOM.
2. Procesa el CSS para construir el CSSOM.
3. El DOM se une con el CSSOM para crear el Render Tree.
4. Se aplican los estilos CSS en el Render Tree.
5. Se ““pintan”” los nodos en la pantalla para que los usuarios vean el contenido de la página web

Nuestros elementos HTML se componen de:

* **ETIQUETA DE APERTURA**: el nombre de nuestra etiqueta encerrado entre símbolos de mayor o menor. Por ejemplo: **<h1>**.
* **CONTENIDO**: dentro de nuestras etiquetas podemos añadir texto u otros elementos HTML, lo que conocemos como **anidamiento**.
* **ETIQUETA DE CIERRE**: son casi iguales que las etiquetas de apertura, pero también necesitan un slash (**/**) antes del nombre de la etiqueta. Por ejemplo: **</h1>**.

Las etiquetas de apertura también pueden tener atributos. Los atributos nos permiten definir características especiales para nuestros elementos: **<etiqueta atributo=""valor del atributo"">**. Por ejemplo: **<h1 class=""saludo"">**.

Tabla

Descripción generada automáticamenteTambién existen elementos vacíos. Estos elementos no tienen contenido ni etiqueta de cierre, solo etiqueta de apertura y atributos. Por ejemplo: **<img src=""puppy.png"" alt=""mi mascota"">**.

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

* **ERRORES SINTÁCTICOS**: Son errores de escritura en el código y evitan que el programa funcione. Pueden ser errores de tipado.
* **ERRORES LÓGICOS**: En estos la sintaxis es correcta, pero el código no hace lo que debería. El programa funciona, pero de forma incorrecta.
* **SELECTOR**: son la referencia a los elementos HTML que queremos estilizar. Los nombres de estas etiquetas van seguidas de una llave de apertura y otra de cierre (**{}**). Por ejemplo: **h1 {}**.
* **PROPIEDADES**: son el tipo de estilo que queremos darle a nuestros elementos. Van seguidas de dos puntos (**:**). Las propiedades deben estar dentro de las llaves del selector que definimos anteriormente. Podemos escribir diferentes propiedades en un mismo selector. Por ejemplo: **h1 { color: }**.
* **VALORES**: son el estilo que queremos que tomen nuestros elementos HTML con respecto a una propiedad. Van seguidas de un punto y coma (**;**). Por ejemplo: **h1 { color: red; }**.

**h1** {

color: red;

}

\***(asterisco)**: Es el selector universal. Las propiedades se aplicaran a todos los elementos de nuestro HTML.

**Tipo**: Son selectores que se aplican a cierto elemento HTML en específico. Las propiedades se aplicaran a la etiqueta que queremos, por ejemplo p, body, html, div, etc.

**Clase**: Si nuestras etiqueta de HTML tienen un atributo de class podemos usar ese valor o identificador para que los cambios en el CSS afecten únicamente a ese elemento.

**ID**: Es similar al anterior, si la etiqueta HTML tiene un ID podemos afectar solo ese elemento.

Las **Pseudo-clases** y **Pseudo-elementos** nos permiten ser aún más específicos con qué elemento o partes de nuestros elementos deben recibir los estilos.

Para usarlas debemos definir el selector base (por ejemplo, **p**) seguido de dos puntos y la pseudo-clase que queremos estilizar (por ejemplo: **p:first-child**). En el caso de los pseudo-elementos debemos usar el dos puntos 2 veces (**p::first-letter**).

/\* Asterisco (universal) \*/

\* {

margin: 0;

}

/\* Tipo \*/

**h1** {

color: red;

}

/\* Clase \*/

.saludo {

font-size: 2em;

}

/\* ID \*/

**#id** {

border-radius: 20px;

}

/\* Pseudo-clases \*/

**p**:first-child {

color: white;

}

**p**:last-child {

color: purple;

}

**p**:nth-child(2n) {

color: red;

}

Tabla

Descripción generada automáticamente

Imagen que contiene Patrón de fondo

Descripción generada automáticamente

Texto

Descripción generada automáticamente  
Imagen que contiene Texto

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

**CSS ARQUITECTURA**

Los objetivos son:

* Predecible > Escribir reglas claras.
* Reutilizable > No escribir codigo redundante.
* Mantenible > Que sea facil de leer y adaptarnos a los estandares.
* Escalable > Que pueda crecer facilmente pero sin afectar el rendimiento.

Buenas practicas

* Establecer reglas para que el equipo se entienda.
* Explicar la estructura base o dar los fundamentos del proyecto a un nuevo integrante.
* Evitar hojas de estilo muy extensas
* Tener una buena documentación explicando ciertos aspectos del codigo.

Un componente, tanto en diseño como desarrollo web, es un elemento muy pequeño que tiene la capacidad de ser reutilizado en diferentes partes de una aplicación. Por ejemplo: botones, iconos, cards, entre otras

* **WHITE-SPACE:**

Permite que el contenido fluya y no lo acomoda para que quede en la pantalla (carrusel)

white-space: nowrap;