# 

TEMAS:

* HTML + CSS
  + Introducción CSS
  + Inline Styles
  + selectores

OBJETIVO:

Aprender qué es CSS y su funcionamiento básico.

MARCO TEÓRICO

## Lectura 2: ESTRUCTURA DE PROYECTO

**¿Qué estructura deberá tener tu Sitio Web?**

Ahora, veamos qué estructura debería tener nuestro sitio de prueba. Las cosas más comunes que tendremos en cualquier proyecto de sitio web que creamos son un archivo HTML índice y carpetas que contienen imágenes, archivos de estilo y archivos de script. Vamos a crear estos ahora:

**index.html**: Este archivo contendrá generalmente el contenido de tu página web inicial, es decir, el texto y las imágenes que la gente ve cuando accede por primera vez a tu sitio. Usando el editor de texto, crea un nuevo archivo llamado index.html y guardarlo justo dentro de tu carpeta sitio-prueba.

**Carpeta images:** Esta carpeta almacenará todas las imágenes que vas a usar en tu sitio. Crea una carpeta llamada images dentro de tu carpeta sitio-prueba.

**Carpeta styles:** Esta carpeta tendrá todos tus códigos CSS usados para darle estilo a tus contenidos (por ejemplo, establecer el color de tus textos y de fondos). Crea una carpeta llamada styles, dentro de tu carpeta sitio-prueba.

**Carpeta scripts**: Esta carpeta tendrá todos tus códigos JavaScript usados para darle interactividad y funcionalidad a tu sitio (por ejemplo. botones que cargan datos cuando 

## Lectura 2: CSS

CSS sirve para dar estilo y diseño a las páginas web — cambiar de tipo de letra, color, tamaño y espaciado del contenido, mostrar múltiples columnas, añadir animaciones y otras funcionalidades. Este módulo nos introducirá en el dominio de las reglas básicas de CSS, selectores, propiedades, reglas de escritura de CSS, aplicación de CSS a HTML, introducción de medidas, color y otras unidades en CSS, cascadas y herencias; y depuración de CSS.

**¿Cómo afecta CSS a un HTML?**

Los navegadores Web, al aplicar las reglas CSS a un documento, modifican la manera en que este es presentado. Una regla CSS se compone de:

* Un conjunto de propiedades (properties), con valores establecidos para actualizar la presentación del contenido HTML, por ejemplo quiero que el ancho de un elemento sea el 50% de su elemento padre, y que su fondo sea rojo.
* Un selector, que seleccionará los elementos afectados por el nuevo valor de la propiedad. Por ejemplo, quiero que mi regla CSS afecte a todos los párrafos (p) de mi documento HTML.

El conjunto de reglas CSS contenidas en el documento de estilos (stylesheet) afectará a la presentación de la página web. Profundizaremos en la sintaxis CSS en el siguiente artículo del módulo — Sintaxis CSS.

**Un ejemplo rápido de CSS**

Las definiciones anteriores pueden tener sentido o no, vamos a asegurarnos que quedan claras presentando un ejemplo rápido. Primero que nada, tomemos un documento sencillo en HTML que contiene un título <h1> y un párrafo <p> (observemos que el documento de estilos se aplicará a HTML utilizando un elemento <link> desde la sección head):

<!DOCTYPE html>  
<html>  
 <head>  
 <meta charset="utf-8">  
 <title>My CSS experiment</title>  
 <link rel="stylesheet" href="style.css">  
 </head>  
 <body>  
 <h1>Hello World!</h1>  
 <p>This is my first CSS example</p>  
 </body>  
</html>

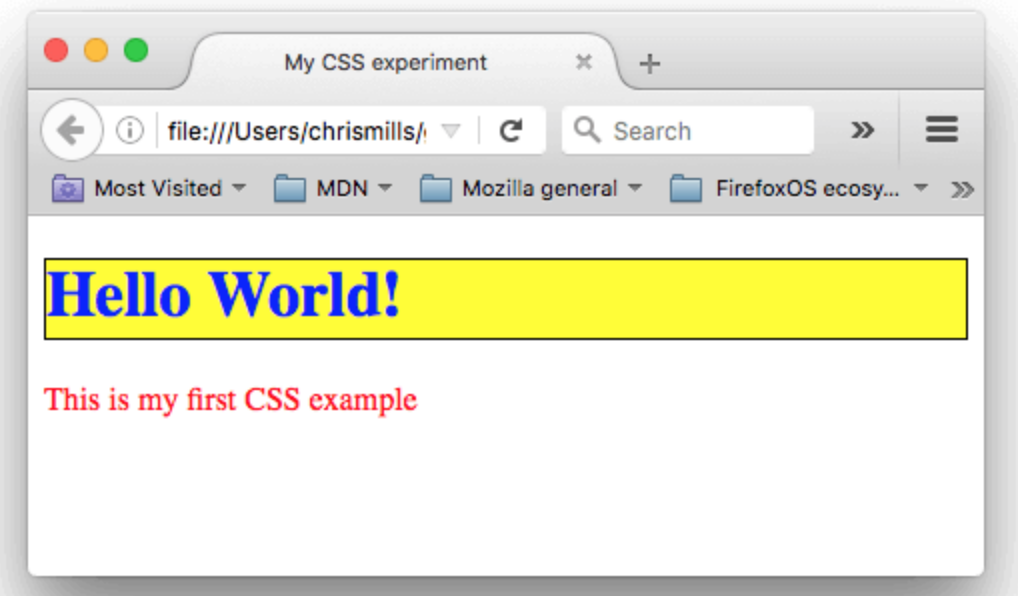
Ahora echemos un vistazo a un sencillo CSS con dos reglas:

h1 {  
 color: blue;  
 background-color: yellow;  
 border: 1px solid black;  
}  
  
p {  
 color: red;  
}

La primera regla comienza con un selector h1, que hará que los valores de sus propiedades se apliquen a los elementos <h1>. Contiene 3 propiedades con sus valores correspondientes (a cada par propiedad:valor se le llama declaración):

1. La primera establece el color del texto en azul.
2. La segunda establece el fondo en amarillo.
3. La tercera establece un marco alrededor de la cabecera de 1 pixel de ancho, sólido (no punteado, o con guiones etc.) y de color negro.
4. La segunda regla comienza con el selector p, que implica que los valores de esta propiedad afectarán a los elementos <p>. Solo contiene una declaración que establece el color en rojo.

En un navegador, el código anterior producirá el siguiente resultado:



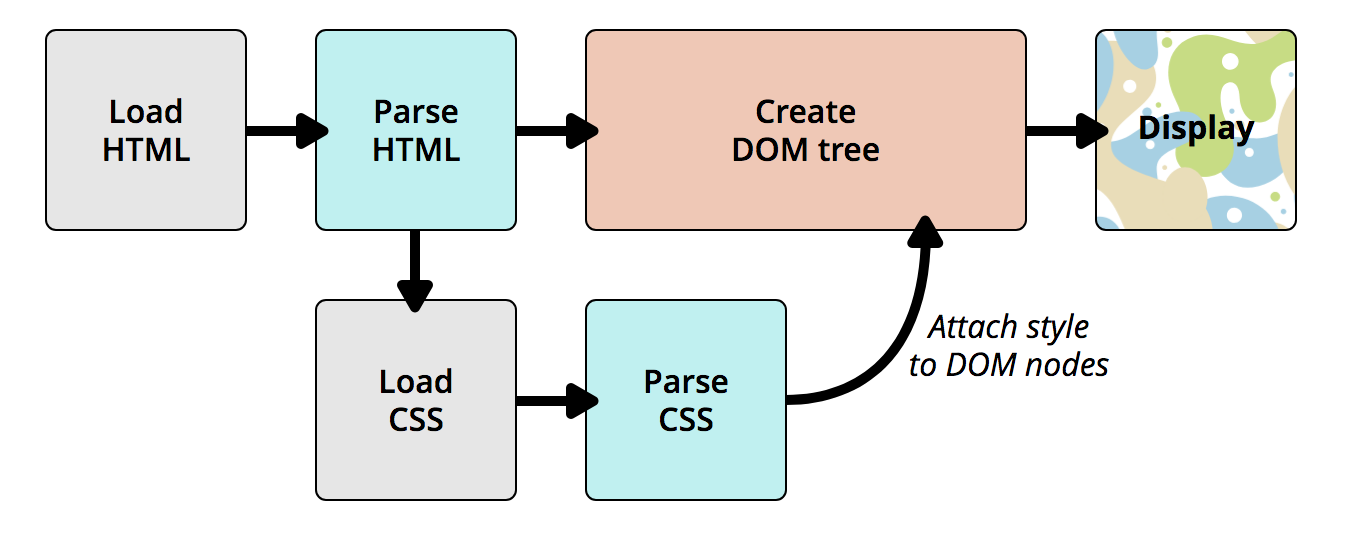
No queda muy bonito, pero basta para hacernos una idea del funcionamiento de CSS.

**¿Cómo funciona CSS en realidad?**

Cuando un navegador muestra un documento, debe combinar su contenido con su información de estilos. Procesa el documento en dos fases:

El navegador convierte HTML y CSS en un DOM (Objeto Documento Modelo). Este DOM representa el documento en la memoria del ordenador. Combinando el contenido del documento con su estilo.

El navegador muestra el contenido del DOM.



**El DOM**

Un DOM tiene una estructura de árbol. Cada elemento, atributo y sección de texto en el lenguaje de marcado se convierte en un nodo DOM node en la estructura del árbol. Los nodos se definen por su relación con otros nodos DOM. Algunos elementos son padres de nodos hijos, y los hijos tienen nodos hermanos.

Entender el DOM nos ayudará a la hora de diseñar, depurar y mantener el CSS, ya que DOM es donde se unen el CSS y el contenido del documento.

**Representación del DOM**

En lugar de realizar un larga y aburrida explicación, mejor veamos un ejemplo de cómo trabajan juntos el DOM y CSS.

Supongamos el siguiente ejemplo de código HTML:

<p>  
 Let's use:  
 <span>Cascading</span>  
 <span>Style</span>  
 <span>Sheets</span>  
</p>

En el DOM, el nodo que corresponde a nuestro elemento <p> es un padre. Sus hijos son nodos de texto y nodos correspondientes a elementos <span>. Los nodos SPAN son también padres, cuyos hijos son nodos de texto:

P  
├─ "Let's use:"  
├─ SPAN  
 └─ "Cascading"  
├─ SPAN  
 └─ "Style"  
└─ SPAN  
 └─ "Sheets"

**¿Cómo aplicar un CSS a nuestro HTML?**

Hay tres maneras distintas de aplicar CSS a un documento HTML, unas más prácticas que otras. Veamos cada una de ellas:

**Documento de estilos externo (CSS externo)**

Ya hemos visto CSS externos en este artículo, pero no con este nombre. Un CSS externo es cuando el CSS se encuentra en un archivo separado con una extensión .css, y lo referenciamos desde HTML con un elemento <link>. El archivo HTML se parecerá a lo siguiente:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>My CSS experiment</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h1>Hello World!</h1>

<p>This is my first CSS example</p>

</body>

</html>

Y el archivo CSS:

h1 {

color: blue;

background-color: yellow;

border: 1px solid black;

}

p {

color: red;

}

Este método es sin duda el mejor, pues podemos usar un único documento de estilos en múltiples documentos y, si queremos cambiarlo, solo necesitamos actualizar el CSS desde un sitio.

**Documento de estilos interno (CSS interno)**

En un CSS interno no tenemos un CSS externo, sino que lo ubicamos dentro de un elemento <style>, en el apartado HTML head. Así, el HTML se escribirá:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>My CSS experiment</title>

<style>

h1 {

color: blue;

background-color: yellow;

border: 1px solid black;

}

p {

color: red;

}

</style>

</head>

<body>

<h1>Hello World!</h1>

<p>This is my first CSS example</p>

</body>

</html>

Este método puede ser práctico en algunas ocasiones (puede que estemos trabajando con un gestor de contenido que no permite modificar el archivo CSS directamente), pero no es tan eficiente como el CSS externo — en una web, el CSS estará repetido en todas las páginas y si se necesita hacer cambios, estos deberán ser hechos en múltiples sitios diferentes.

**Estilos en una línea**

Los estilos en una línea son declaraciones CSS que afectan solo a un elemento que está contenido dentro de un atributo style:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>My CSS experiment</title>

</head>

<body>

<h1 style="color: blue;background-color: yellow;border: 1px solid black;">Hello World!</h1>

<p style="color:red;">This is my first CSS example</p>

</body>

</html>

Por favor, no lo hagamos a menos que sea estrictamente necesario. Su mantenimiento es verdaderamente complicado (podríamos tener que actualizar la misma información muchas veces en cada documento), además de mezclar la información de estilo con la información estructural del HTML, haciendo el CSS difícil de leer y de entender. Manteniendo los distintos tipos de código separados y puros, facilitará la tarea a aquellos que vayan a trabajar posteriormente en el código.

Solamente está justificado el uso de estilos en la misma línea cuando los entornos de trabajo son muy restrictivos (el CMS solo permite editar el cuerpo del HTML).

**Un poco de vocabulario**

En el nivel más básico, CSS se compone de dos bloques:

* **Propiedades:** Son identificadores que indican a las personas qué característica de estilo (ancho, color de fondo, fuente) queremos cambiar.
* **Valores:** A cada propiedad se le da un valor, que indica cómo queremos cambiar esta característica (por ejemplo qué fuente, qué ancho o qué color usar)

El par formado por una propiedad y un valor se denomina declaración CSS. Las declaraciones CSS forman bloques declarativos. Los bloques de declaraciones CSS emparejados con selectores forman Conjuntos de Reglas CSS (o simplemente Reglas CSS).

Antes de profundizar más en el tema, veamos un ejemplo concreto (vimos algo parecido en nuestro artículo anterior):

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>My CSS experiment</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h1>Hello World!</h1>

<p>This is my first CSS example</p>

<ul>

<li>This is</li>

<li>a list</li>

</ul>

</body>

</html>

Y el archivo CSS:

h1 {

colour: blue;

background-color: yellow;

border: 1px solid black;

}

p {

color: red;

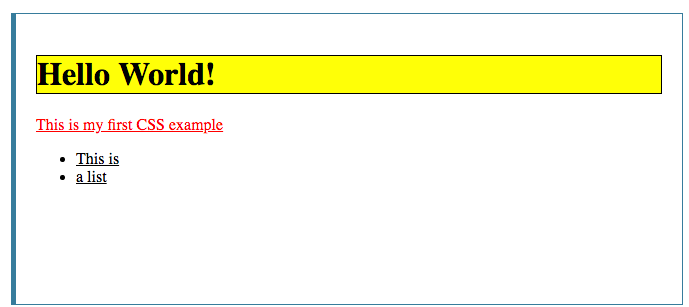
}

p, li {

text-decoration: underline;

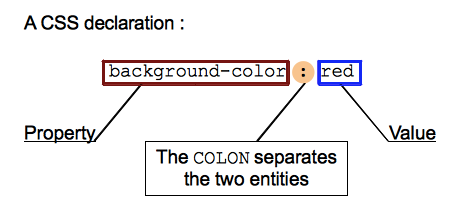
}

Combinando estos dos se producirá el siguiente resultado:



**Declaraciones CSS**

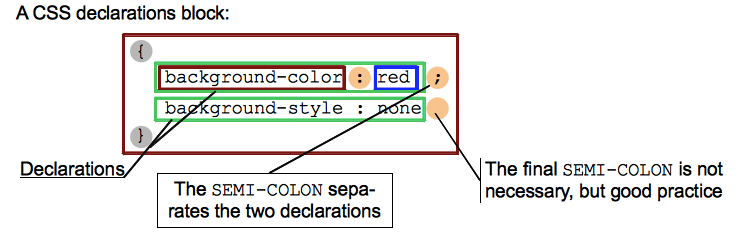
La principal función del lenguaje CSS es asignar valores a las propiedades CSS. El motor CSS calcula qué declaraciones debe aplicar a cada elemento de una página para mostrarla y darle estilo de manera adecuada. Lo importante es recordar que tanto las propiedades como los valores son modificables en CSS. Los pares propiedad y valor, se separan por dos puntos (:).



**Bloques declarativos en CSS**

Las declaraciones están agrupadas por bloques, en los que el conjunto de declaraciones se encuentra enumerado entre llaves, uno inicial ( { ) y otro final ( } ).

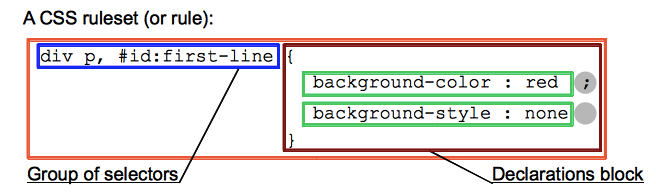
Cada declaración contenida en un bloque declarativo debe estar separada por un punto y coma (;), sino el código no funcionará (o producirá resultados inesperados). La última declaración de un bloque no necesita llevar (;), aunque se considera buena práctica añadirlo, para prevenir olvidos cuando se añaden más declaraciones al bloque.



Los bloques pueden estar anidados. En estos casos los corchetes de apertura y cierre deben definir el anidamiento lógicamente, de la misma manera en que las etiquetas anidan a los elementos HTML. El ejemplo más usual son las reglas-@, que son bloques que comienzan por el identificador @ como @media, @font-face, etc. (ver las declaraciones CSS más abajo).

**Selectores y reglas CSS**

Pero, en este rompecabezas falta una pieza — y es aprender a identificar los elementos a los que afectará nuestro bloque declarativo. Esto lo conseguiremos añadiendo a cada bloque declarativo un prefijo a modo de selector — un módulo que identificará a ciertos elementos de nuestra página. Las declaraciones asociadas se aplicarán sólo a estos elementos. El selector más el bloque declarativo se llama regla o regla-conjunto.



Los selectores (filtros) pueden llegar a ser bastante complicados — se puede establecer una regla que afecte a múltiples elementos separándolos en el selector por comas, y estos pueden ser identificados en conjunto, por ejemplo: Seleccionar los elementos de la clase "blah", pero solo los que estén dentro de un <artículo>, y solo cuando pase por encima el puntero del ratón. Tranquilidad — esto lo iremos viendo más claro a medida que ganemos experiencia con CSS, veremos los selectores con más detalle en el siguiente artículo dedicado a los Selectores.

Un elemento puede estar afectado por varios selectores, por tanto, una propiedad puede ser asignada en distintas ocasiones por distintas reglas. CSS definirá cuál tiene preferencia sobre las otras para ser aplicada: esto se conoce como algoritmo en cascada, y veremos cómo funciona en Cascada y herencia.

**Declaraciones CSS**

Las Reglas CSS son los bloques principales de un documento de estilos — el bloque más usual que veremos en un CSS. Pero nos podemos encontrar con otros tipos de bloques — Las reglas CSS son uno de los tipos de declaraciones CSS. Los otros son:

**Reglas-@:** Usadas en CSS para definir metadatos, información condicional u otra información descriptiva. Comienzan con el símbolo (@), seguido del identificador del tipo de regla, luego un bloque sintáctico correspondiente y terminará con un (;). Cada tipo de Regla-@ definido por un identificador dispondrá de su propia sintaxis y semántica internas. Ejemplos:

* @charset y @import (metadatos)
* @media o @document (información condicional, llamada también declaraciones anidadas, ver más abajo).
* @font-face (información descriptiva)

Ejemplo concreto de sintaxis:

@import 'custom.css';

Esta Regla-@ importará un archivo CSS externo hasta nuestro CSS actual.

**Declaraciones anidadas:** son un subtipo de Reglas-@, su sintaxis es un bloque de reglas CSS anidadas que solo afectará al documento bajo ciertas condiciones:

* El contenido de la regla-@ @media se aplicará solo si el dispositivo que ejecuta el navegador cumple la condición expresada.
* El contenido de la regla-@ @supports se aplicará solo si el navegador soporta la característica mencionada.
* El contenido de la regla-@ @document se aplicará solo si la página actual cumple ciertas condiciones.

Ejemplo de sintaxis:

@media (min-width: 801px) {

body {

margin: 0 auto;

width: 800px;

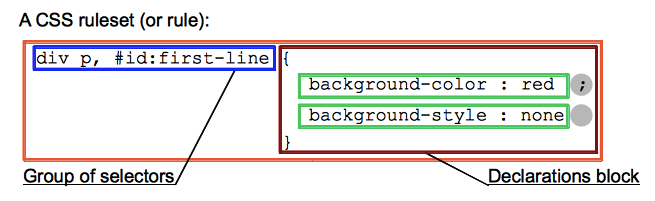
}

}

Esta regla anidada sólo se aplicará cuando el ancho de la página sea superior a 800 pixels.

**SELECTORES**

Los selectores forman parte de las reglas CSS y van justo antes de los bloques declarativos.



**Tipos de selectores**

Podemos dividir los selectores en las siguientes categorías:

* **Selectores simples:** Seleccionan los elementos por el nombre del tipo de elemento, class, o su id.
* **Selectores de atributos:** Seleccionan los elementos por los valores de sus atributos.
* **Pseudo-clases:** Seleccionan los elementos por el estado en que se encuentran, cómo haber aparecido al pasar el ratón, o el tic deshabilitado o seleccionado, o por ser el primer hijo de su padre en el árbol DOM.
* **Pseudo-elementos:** Selecciona los elementos por su situación en relación a otro elemento, por ejemplo: la primera palabra de cada párrafo, o el contenido que se encuentra justo después de un elemento.
* **Combinaciones**: No son en sí mismos selectores, sino formas de combinar dos o más selectores de forma práctica para una selección especial. Por ejemplo, se pueden seleccionar párrafos que sean descendientes de divs, o párrafos situados justo después de títulos.
* **Selectores múltiples:** Tampoco son selectores en sí mismos; podemos agrupar múltiples selectores en la misma regla CSS separados por comas, para aplicarlos a una de las declaraciones o a todos los elementos seleccionados por estos selectores.

**Selectores simples**

Este selector hace referencia directamente a un tipo de elemento HTML. Es la manera más sencilla para referirse a todos los elementos de un mismo tipo. Veamos el ejemplo:

Dado el siguiente código HTML:

<p>What color do you like?</p>

<div>I like blue.</div>

<p>I prefer red!</p>

Una sencilla hoja de estilos:

/\* Todos los elementos p son rojos \*/

p {

color: red;

}

/\* Todos los elementos div son azules \*/

div {

color: blue;

}

**Selectores de clases**

El selector de clase se forma con un punto, '.', seguido de un nombre de clase. Un nombre de clase puede ser cualquier valor sin espacios usado dentro de un atributo HTML class. Podemos elegir el nombre que deseemos para la clase. Es conveniente saber que en un mismo documento varios elementos pueden compartir el mismo valor de clase y que un elemento puede tener varios nombres de clases separados por un espacio en blanco. Veamos un rápido ejemplo:

Sea el siguiente código HTML:

<ul>

<li class="first done">Create an HTML document</li>

<li class="second done">Create a CSS style sheet</li>

<li class="third">Link them all together</li>

</ul>

Y un sencillo documento de estilos:

/\* El elemento con la clase "first" está en negrita \*/

.first {

font-weight: bold;

}

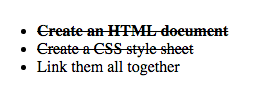
/\* Todos los elementos con la clase "done" están tachados \*/

.done {

text-decoration: line-through;

}

Obteniendo el siguiente resultado:



**Selectores ID**

El selector ID está formado por una almohadilla (#), seguida del nombre ID de determinado elemento. Cualquier elemento solo puede tener un único nombre ID fijado con el atributo id. Podemos usar cualquier nombre para el ID. Es la forma más efectiva de selecionar un solo elemento.

Veamos un rápido ejemplo — dado el código HTML:

<p id="polite"> — "Good morning."</p>

<p id="rude"> — "Go away!"</p>

Y un sencillo documento de estilos:

#polite {

font-family: cursive;

}

#rude {

font-family: monospace;

text-transform: uppercase;

}

Herramientas

Para el desarrollo de las clases recomienro usar <https://codepen.io/pen/> para las demostraciones de codigo.

La confirmación de los ejercicios y seguimiento a la guía debe ser en el editor de su preferencia **Sublime Text** (https://www.sublimetext.com/)

Ejercicios

Crear una lista de elementos HTML y explorar el uso de selectores