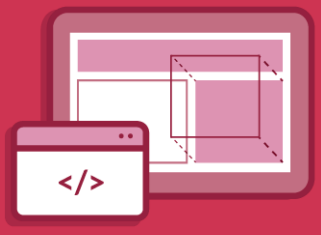
Diseño WEB

Clase 3

Incluyendo CSS a nuestro Proyecto



coderhouse

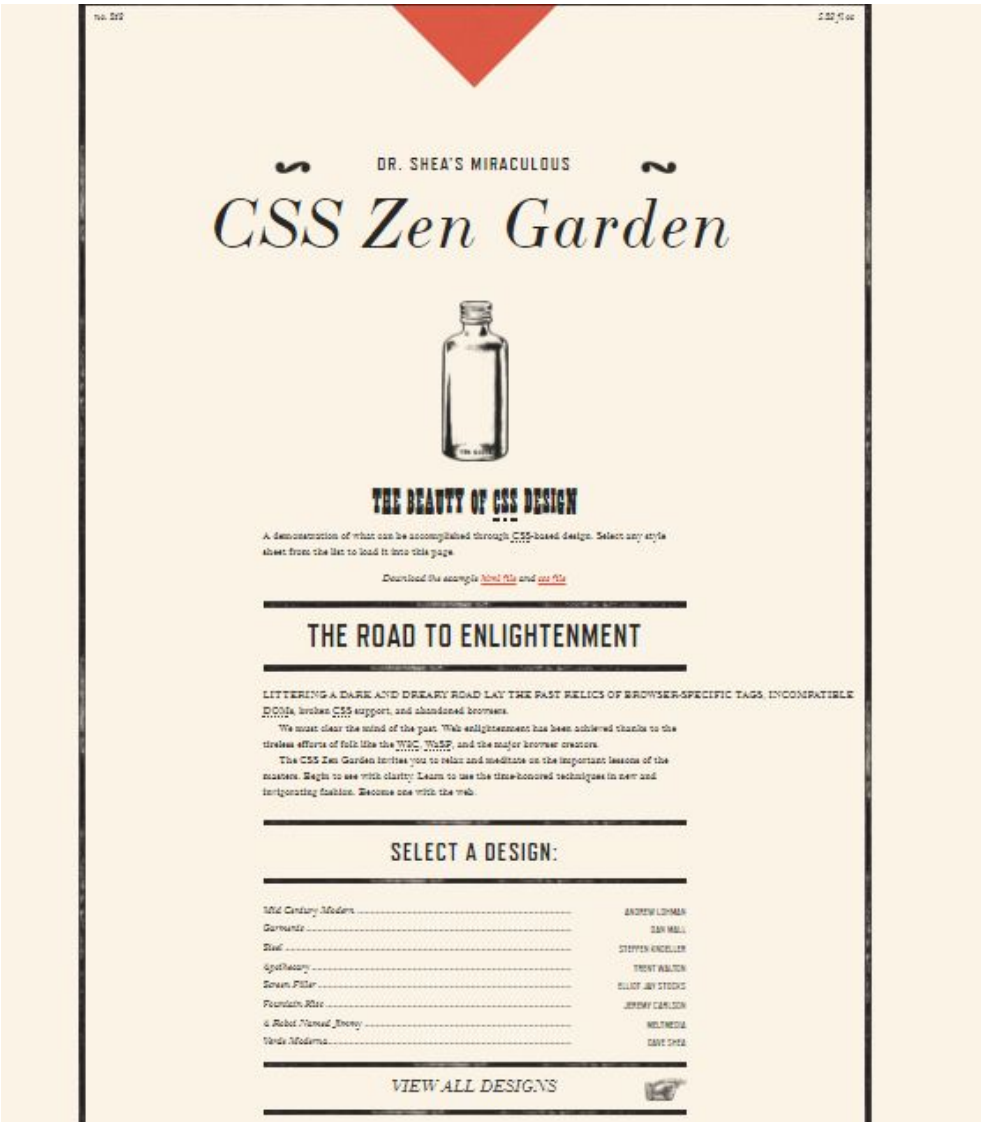
Bases de CSS

Premisas

CSS: lenguaje web para aplicar formato visual (color, tamaño, separación y ubicación) al HTML. CSS puede hacer un texto más grande, negrita o itálica, pero no reemplaza los strong, em y h1. Su objetivo es separar la semántica y estructura (el HTML) del formato con que se pretende mostrar. Sí, **con CSS podés cambiar por completo el aspecto de cualquier etiqueta HTML**.

CSS, ¿Por qué?

CSS bien implementado permite cambiar TODO el diseño de un sitio web sin modificar el HTML. Las siguientes dos imágenes **corresponden al mismo código HTML pero distinto CSS**:



Esto se logra aprendiendo a separar la estructura (el HTML) del aspecto visual (el CSS).

Historia de CSS

Las hojas de estilos aparecieron poco después que el lenguaje de etiquetas SGML, alrededor del año 1970. Desde la creación de SGML, se observó la necesidad de **definir un mecanismo que permitiera aplicar de forma consistente diferentes estilos a los documentos electrónicos.**

El gran impulso de los lenguajes de hojas de estilos se produjo con el boom de Internet y el crecimiento exponencial del lenguaje HTML para la creación de documentos electrónicos. La guerra de navegadores y **la falta de un estándar para la definición de los estilos dificultaban la creación de documentos con la misma apariencia en diferentes navegadores.**

El organismo W3C (World Wide Web Consortium), encargado de crear todos los estándares relacionados con la web, propuso **la creación de un lenguaje de hojas de estilos específico para el lenguaje HTML** y se presentaron nueve propuestas. Las dos propuestas que se tuvieron en cuenta fueron la CHSS (Cascading HTML Style Sheets) y la SSP (Stream-based Style Sheet Proposal).

La propuesta CHSS fue realizada por Håkon Wium Lie y SSP fue propuesto por Bert Bos. **Entre finales de 1994 y 1995 Lie y Bos se unieron para definir un nuevo lenguaje que tomaba lo mejor de cada propuesta y lo llamaron CSS** (Cascading Style Sheets).

En 1995, el W3C decidió apostar por el desarrollo y estandarización de CSS y lo añadió a su grupo de trabajo de HTML. **A finales de 1996, el W3C publicó la primera recomendación oficial, conocida como "CSS nivel 1".**

A principios de 1997, el W3C decide separar los trabajos del grupo de HTML en tres secciones: el grupo de trabajo de HTML, el grupo de trabajo de DOM y el grupo de trabajo de CSS.

El 12 de Mayo de 1998, el grupo de trabajo de CSS publica su segunda recomendación oficial, conocida como "CSS nivel 2". **La versión de CSS que utilizan todos los navegadores de hoy en día es CSS 2.1, una revisión de CSS 2 que aún se está elaborando** (la última actualización es del 8 de septiembre de 2009). Al mismo tiempo, la siguiente recomendación de CSS, conocida como **"CSS nivel 3", continúa en desarrollo desde 1998 y hasta el momento sólo se han publicado borradores.**

La adopción de CSS por parte de los navegadores ha requerido un largo periodo de tiempo. El mismo año que se publicó CSS 1, Microsoft lanzaba su navegador Internet Explorer 3.0, que disponía de un soporte bastante reducido de CSS. El primer

navegador con soporte completo de CSS 1 fue la versión para Mac de Internet Explorer 5, que se publicó en el año 2000.

CSS, tipo de formato

Hay 3 grupos de declaraciones CSS que podemos implementar:

FORMATO DE TEXTO	FORMATO DE CAJAS	UBICACIÓN DE ELEMENTOS
Familia tipográfica, color y tamaño de fuente, interlineado, negritas, itálicas, subrayados	Ancho, alto, bordes, color e imágenes de fondo, márgenes sombras, rotación.	Posicionamiento, flotaciones, mostrar/ocultar cajas.
HEREDADAS	NO HEREDADAS	

Insertar CSS en el HTML

Todas las etiquetas HTML tienen el atributo **`style=""`** y entre comillas se escribirán las reglas CSS para formatear **únicamente ese elemento.**

```
<h1>Un encabezado sin formato</h1>
<h2 style="CODIGO CSS">H2 con formato CSS</h2>

<p>Párrafo sin formatear</p>
<p style="CODIGO CSS">Párrafo formateado</p>
<p>Otro párrafo sin formatear</p>
```

Existe una etiqueta **<style></style>** que va en el **<head>** y contendrá las reglas CSS para **formatear únicamente el archivo HTML donde se haya insertado**.

```
<style>
Aca va el codigo CSS, hay que indicarle el listado de
etiquetas que se quieren formatear y por cada etiqueta el
CSS a aplicarle.
</style>
```

Existe una etiqueta **<link/>** que va en el **<head>** y se usa para cargar un archivo externo –con extensión .css- que **permite formatear múltiples archivos HTML.** El **<link/>** no funciona si no tiene el atributo "rel". Debe tener el valor stylesheet (hoja de estilos).

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

Formatear un elemento

Si el CSS está en el head (**<style></style>** o **<link/>**) hay que indicar el elemento a afectar. Se debe **indicar el elemento y entre llaves todo el código CSS** a aplicar al elemento seleccionado:

POR ETIQUETA	POR ATRIBUTO ID

p{
formato para todos los párrafos
}
h1{
formato para todos las etiquetas h1
	#ppal{
	solo formatea la etiqueta con id="ppal"
	}
	#caja{
	solo formatea la etiqueta con id="caja"

}	}
Si encuentra más de uno, le aplica el mismo formato a todos.	**Sólo afecta a un elemento por cada archivo HTML, lleva numeral por delante.**

Reglas sintácticas

1- **Cada declaración CSS está formada por un juego de pares `propiedad:valor;`** no es con igual (como pasa con los atributos HTML), sino con dos puntos. Si a un elemento se le aplica más de una propiedad se deben separar con punto y coma. La última propiedad puede tener punto y coma (pero en este caso, no es obligatorio).

```
p{ propiedad1: valor; propiedad2: valor; }
```

2- Tampoco se ve afectado por el espacio en blanco. **Las propiedades se pueden escribir de corrido o una debajo de la otra**. Los comentarios se hacen como en Javascript. Lo que esté comentado, será ignorado por CSS.

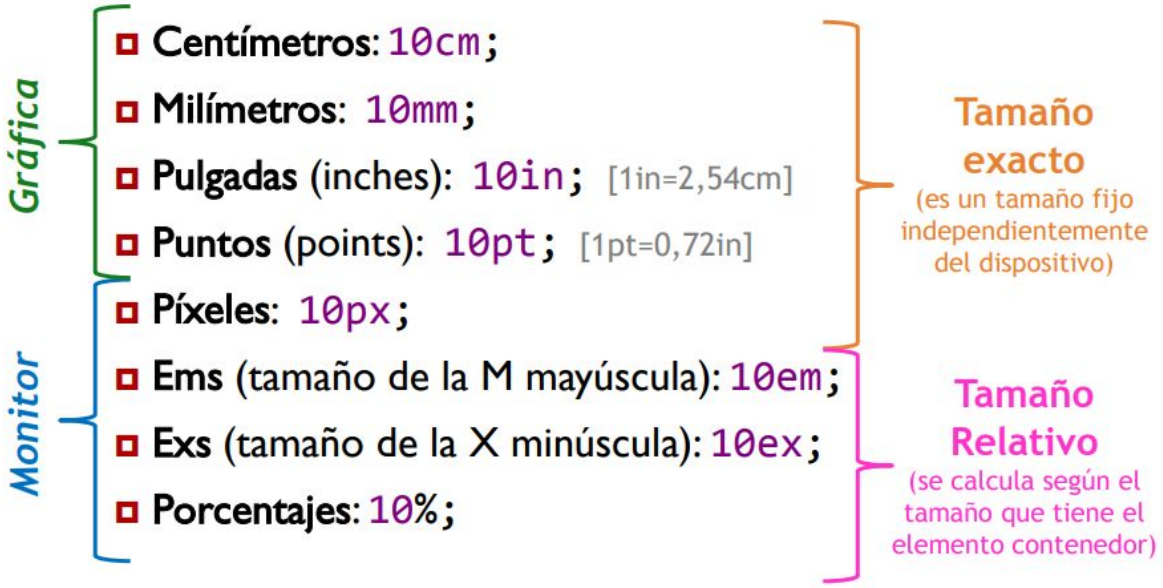
```
P{ propiedad1: valor; propiedad2: valor;
 propiedad3         :             valor;

     /* propiedad_ignorada: valor_ignorado */
}
```

3- Siempre que la propiedad represente un número, **el valor debe indicar en qué unidad se expresa.** Entre el número y la unidad no pueden existir espacios.

```
P{
  propiedad1: 14px;
  propiedad2: 80%;
  propiedad3: 20cm;
}
```

Unidades de medida centrales

Los números pueden expresarse en:



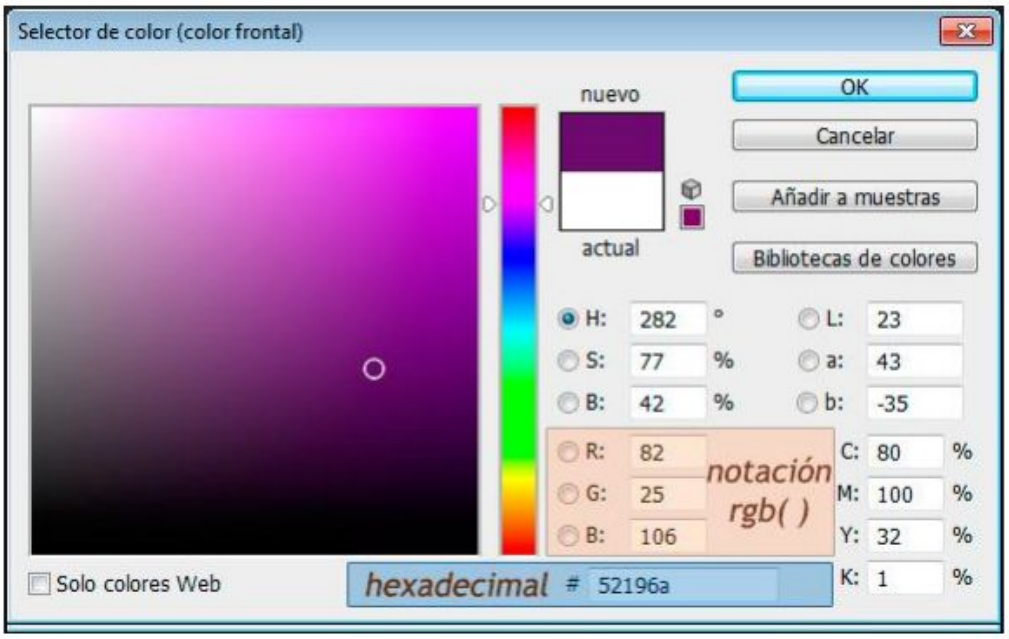
4- Siempre que la propiedad represente un color, el valor se puede expresar de tres maneras distintas: **Por nombre del color (en inglés). Por hexadecimal (numeral + 6 caracteres). Por rgb (red, green, blue)**, tres números de 0 a 255, separados por coma.

```
P{
  propiedad1: red;
  propiedad2: #FF0000;
  propiedad3: rgb( 255, 0, 0 );
}
```

Las componentes RGB de un color también se pueden indicar mediante un porcentaje. El funcionamiento y la sintaxis de este método es el mismo que el del RGB decimal. La única diferencia es que en este caso el valor de las componentes RGB puede tomar valores entre 0% y 100%. Por tanto, para transformar un valor RGB decimal en un valor RGB porcentual, es preciso realizar una regla de tres considerando que 0 es igual a 0% y 255 es igual a 100%.

Obtener el valor del color

Sí, también se hace con cualquier soft fotográfico. Usar cualquier picker de color y junto al círculo cromático te aparecen los valores.



5- Si se necesita aplicar el mismo formato CSS a más de un elemento diferente, no hace falta escribir dos veces todas las propiedades. **Si se escribe más de un elemento, separado por comas, aplica el mismo formato a todos.**

```
#caja, p{
  /* Formato en común para el objeto con id="caja" y
todos los párrafos */
}
```

Desafío

- Crear un archivo CSS y un párrafo dentro de un documento HTML (con la estructura base HTML5). Dentro del archivo CSS formatear el párrafo con la propiedad “color” y como valor el color rojo *(Duración : 30 min)*.

Atributo class=””

Sabemos que podemos ponerle ID a cualquier elemento HTML para darle un "nombre". Y así como el ID, todos los elementos también aceptan el atributo **`class=""`**. Esa clase se usa cuando querés aplicar el mismo estilo a más de un elemento (y la búsqueda por etiqueta no sirve para lograrlo). No necesitas escribir varias veces el mismo CSS. No necesitas repetir el ID.

Cuando una regla empieza con numeral, es un ID. **Si el elemento buscado, empieza con un punto, buscará en el HTML un class con ese valor:**

HTML:

```
<a class="word">Ver documento</a>
```

CSS:

```
.word{
  /* Formato CSS para los class="word" */
}
```

¿La mayor diferencia entre class e ID? **El ID es único por documento y el class se puede repetir.**

Cascada CSS

Define **una ruta que deben cumplir los elementos dentro del HTML**. Se trata de una lista de elementos (ya sea por etiqueta, ID o clase) separados por espacios. **Si algún elemento cumple esta "ruta" le aplica el formato CSS al último elemento de la lista** (el que está justo antes de la apertura de llaves). Se usa para no plagar el HTML de atributos CLASS e ID.

Si tenemos dos listas distintas (un **`<ol>`** y un **`<ul>`**) necesitamos que los **`<li>`** de cada lista sean de distintos colores. Podemos aplicarle un class a cada list-item (pero esto nos obliga a copiar y pegar el class tantas veces como ítems tenga cada lista) O hacer

una cascada discriminando los **<li>** adentro de cada lista.

Si se hace con Class **hay que repetir el class por cada ítem del mismo tipo**:

HTML:

```
<ul>
  <li class="rojo">Item R1</li>
  <li class="rojo">Item R2</li>
  <li class="rojo">Item R3</li>
  <li class="rojo">Item R4</li>
</ul>
<ol>
  <li class="azul">Item A1</li>
  <li class="azul">Item A2</li>
</ol>
```

CSS:

```
.rojo{ color: red; }
.azul{ color: blue; }
```

Si se hace con cascada **todo <li> dentro de un <ul>, color rojo. Y todo <li> dentro de un <ol>, color azul:**

HTML:

```
<ul>
  <li class="rojo">Item R1</li>
  <li class="rojo">Item R2</li>
  <li class="rojo">Item R3</li>
  <li class="rojo">Item R4</li>
</ul>
<ol>
  <li class="azul">Item A1</li>
  <li class="azul">Item A2</li>
</ol>
```

CSS:

```
ul li{ color: red; }
ol li{ color: blue; }
```

Precedencia de declaraciones

Cuando reglas distintas apuntan al mismo objeto:

Si son propiedades distintas se suman (se combinan).

Si tienen alguna propiedad repetida, solo una queda.

Esto es lo que se llama *precedencia*.

- Las reglas por etiqueta tienen menos precedencia (porque son genéricas).
- La declaración de Class sobreescribe las de etiqueta (porque tenés que darle a mano esa clase)
- La declaración de ID pisa cualquier otra regla (porque el ID es único, por lo cual es más específico).

!important;

Si tenés 3 reglas CSS, es poco probable que *"choquen"*, pero en un CSS extenso es más común.

La declaración ***!important;*** corta la precedencia. Se escribe después del valor de la propiedad CSS, que se quiere convertir en la más importante. Sí, es un ***!important;*** por cada valor a pisar.

Si necesitás más de 5 *!important;* en todo tu CSS, algo estás haciendo mal.

Desafío

- Crear un archivo CSS y un elemento Div dentro de un documento HTML (con la estructura base HTML5). Dentro del archivo CSS formatear el div pero esta vez con la clase “para1”; las propiedades para la misma: “color” y como valor el color verde *(Duracion : 30 min).*

Desafío

- Aplicar un documento CSS externo al HTML del proyecto (creado en el desafío 2). Insertar las reglas vistas hasta el momento: atributos para textos, listas, fondos.