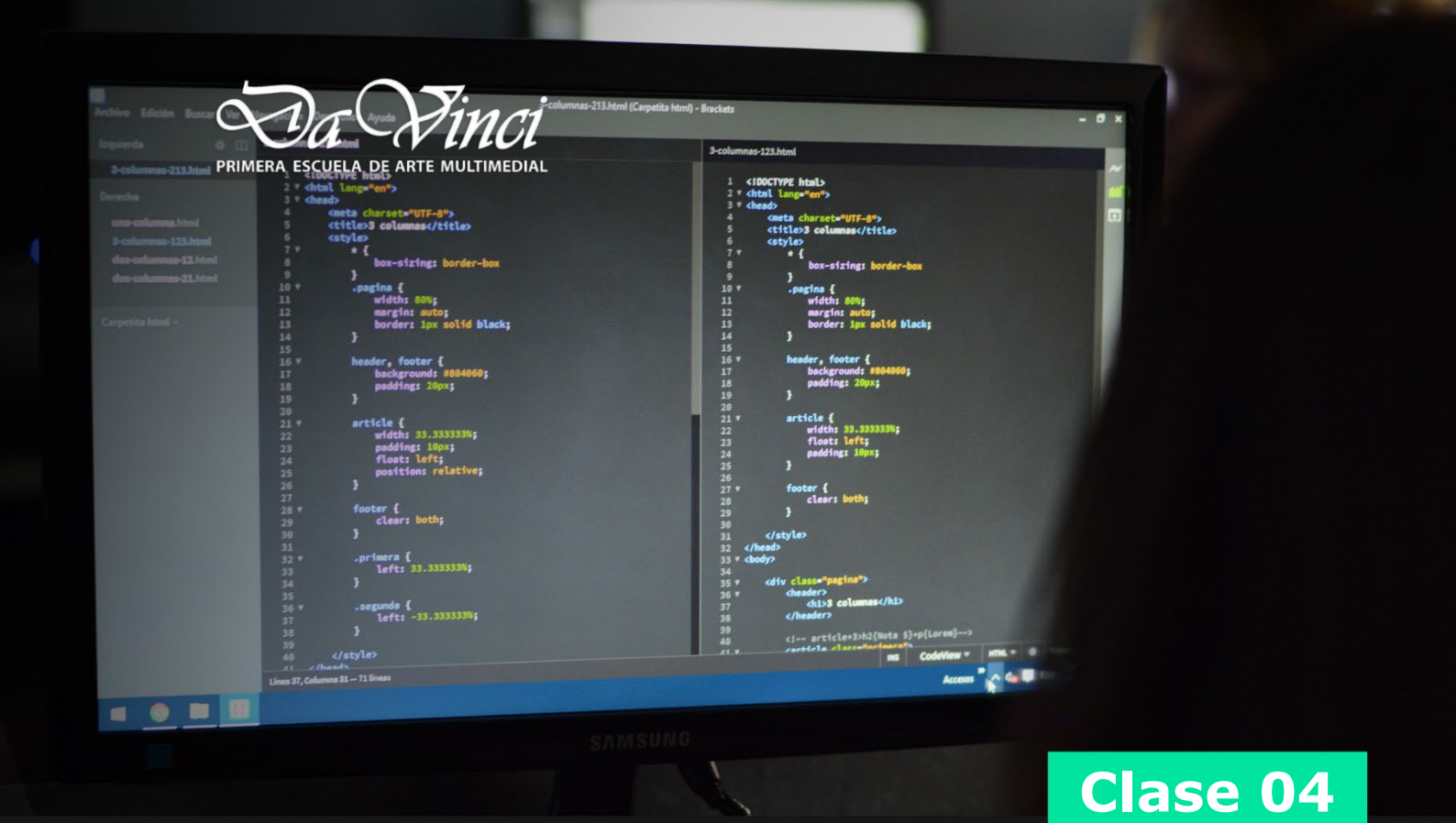


Clase 04

Diseño y Programación Web

Materia:

Programación Web I

Docentes contenidistas: GARCIA, Mabel y PEREZ, Jorge

Revisión: ROLDÁN, Hernán

Contenido

CLASE 4

¡Te damos la bienvenida a la clase 4 de la materia Programación Web I!

En esta clase vamos a ver los siguientes temas:

- ¿Qué es CSS?
- Concepto de Herencia.
- Formas de insertar CSS al HTML.
- Afectar a elementos desde CSS, (etiqueta, class, id).
- Reglas sintácticas.
- Manejo de color, notaciones de distintos modelos.
- Concepto de Cascada, precedencia. Regla **!important**.
- CSS para texto (color, align, size, family, spacing, indent, height, weight, style, decoration, transform, shadow).

Introducción a CSS

¿Qué es CSS?

Las siglas **CSS** (*Cascading Style Sheets*) significan "Hojas de estilo en cascada" y parten de un concepto simple pero muy potente: aplicar **estilos** (colores, formas, márgenes, etc...) a uno o varios **documentos** HTML.

Se lo denomina **estilos en Cascada** porque se lee, procesa y aplica el código desde arriba hacia abajo. La cascada en CSS es el proceso por el cual se aplican los estilos a un documento HTML, permitiendo que los estilos más específicos se prioricen por sobre los estilos menos específicos.

En este escenario, debemos tener claro la división de responsabilidades:

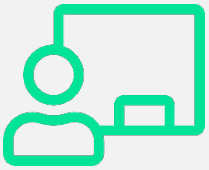
- Los **documentos HTML** tendrán los datos, la información que se le va a transmitir al usuario (contenido por medio de etiquetas).
- Los **documentos CSS** serán responsables de los aspectos que tengan relación con el estilo visual de los documentos HTML (diseño, colores, espacios entre elementos, etc.).

Para crear un documento css, hay que crear un archivo con el nombre deseado y guardarlo con **extensión .css**

Tipos de formatos

Hay 3 grupos de declaraciones CSS que podemos implementar:

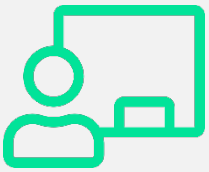
- **Formato de texto**: Afectaremos la familia tipográfica, color, tamaño de la fuente, interlineado, negritas, itálicas, subrayados **(se heredan)**.
- **Formato de cajas**: Afectaremos ancho, alto, bordes, color de fondo, imágenes de fondo, márgenes, etc. **(NO se heredan)**.
- **Ubicación de elementos**: Afectaremos la posición y distribución dentro del diseño **(NO se Heredan)**.



Estas declaraciones las iremos viendo a medida que vayamos avanzando.
Lo importante a tener es cuenta es el concepto de **HERENCIA**.

¿Qué es la Herencia?

Hay ciertas **propiedades que se heredan desde los elementos HTML padres a los elementos HTML hijos**, modificando el valor que tienen por defecto.



Por ejemplo, si queremos que todo nuestro documento html tenga un color de tipografía gris, podemos declararla en el body, si no hay otra regla que diga lo contrario, todo el texto será gris.

Si a lo largo de nuestro css, decidimos que un elemento, por ejemplo, un párrafo, lo queremos de otro color, podremos darle a este elemento un color diferente, por ejemplo, rojo.
Esta nueva declaración que le estamos dando a ese párrafo sobrescribe a la declaración que estaba heredando cuando establecimos que dentro del body todo tendría color gris.

A medida que vayamos avanzando en la técnica tendremos que ser cuidadosos con las reglas css que implementemos para tener control sobre el resultado visual de los elementos.

Insertar CSS en el HTML

Hay 3 formas de aplicar CSS al HTML:

1. TODAS las etiquetas HTML aceptan el **atributo style** y entre comillas se escribirán las reglas CSS para formatear UNICAMENTE a ESE elemento. Se lo conoce como estilo inline (no recomendado).

```
<h2 style="propiedad:valor;">Juegos vendidos</h2>
   <p>Este párrafo no tiene estilo</p>
   <p style="propiedad:valor;">Con estilo</p>
```

2. Por medio de la **etiqueta <style></style>** que va dentro del head y contendrá las reglas CSS para formatear ÚNICAMENTE a ESE ARCHIVO HTML.

```
 <style>
 /*Hay que indicar que elementos se quieren formatear y por cada elemento las propiedades css a aplicar */
 </style>
```

3. Por medio de la **etiqueta <link>** que va dentro del **head** y se usa para cargar un archivo EXTERNO, con la extensión .css, que permite formatear MÚLTIPLES ARCHIVOS HTML.
   Esta es la forma más profesional y nos ahorra código, si hay que cambiar alguna propiedad, se recurre a este archivo donde tenemos todas las declaraciones para el sitio.

```
 <link rel="stylesheet" href="estilos.css" >
```

El atributo **rel** es obligatorio, e indica cuál es la relación de ese archivo importado, con nuestro html. Su valor siempre será "**stylesheet**" (hoja de estilos).

El atributo **href** es obligatorio y debemos indicar la ruta desde nuestro archivo html hacia donde tenemos creada la hoja de estilos (en este caso, el documento .html y el archivo estilos.css están en el raíz del sitio, si nuestra hoja de estilos estuviese en una carpeta llamada css, su href tendría como valor = "css/estilos.css").

Otro aspecto importante a tener en cuenta y muy relacionado con las 3 formas de aplicar css es el orden de prioridad, más conocido como "orden de cascada".

- Si un elemento tiene un estilo en línea definido, éste tiene la máxima prioridad y se aplicará al elemento.
- Si un elemento no tiene un estilo en línea, pero su estilo está definido en el estilo interno, se aplicará al elemento.
- Si un elemento no tiene un estilo en línea ni un estilo interno definido, se aplicará el estilo definido en el archivo CSS externo.

Conclusión: si un elemento tiene definido un estilo en línea, éste tendrá prioridad sobre cualquier otro estilo definido. Si no hay estilo en línea definido, se utilizará el estilo interno si está presente en la página, y si no hay estilo interno, se utilizará el estilo externo.

Dar formato a elementos HTML

Para aplicar un estilo en CSS a un elemento específico, es necesario indicar el nombre del elemento seguido por llaves que contengan el código CSS a aplicar en él.
Hay varias formas de lograr esto y empezaremos por las más sencillas: **Por etiqueta, por atributo id, por atributo class**.

Nuestro html, a esta altura tendrá todas las etiquetas necesarias para la estructura de la información. Por ejemplo:

```
<h1>Mi sitio web</h1>
   <p>Este párrafo no tiene estilo</p>
```

Por etiqueta

Podemos hacer referencia al elemento por su nombre (strong, em, p, h1, h2, etc), por ejemplo, si queremos afectar a todos los h1 que tenemos y queremos darle un color en especial usaremos la propiedad **color**, por ahora usaremos como valor su nombre en inglés (keyword), pero hay otras formas que ya veremos.

Hay que tener presente la sintaxis para usar una propiedad y un valor.
Nuestra hoja de estilos tendrá la regla:

```
h1 {
   color:red;
    /* Acá irán las propiedades css y sus
    valores separadas por punto y coma */
}
```

Nuestro browser renderizará el h1 de color rojo. Si tuviéramos más de un documento html que posean h1, todos tendrían el mismo color.

Este parrafo no tiene estilo

Por atributo id

Solo afecta a **un elemento** por cada archivo HTML. Los id, son únicos por documento HTML y pueden colocarse sobre cualquier etiqueta.

Desde nuestra hoja de estilos lleva un numeral por delante #. Este tipo de declaración es más específica que la declaración por etiqueta.

Desde nuestro documento html le pondremos ese identificador al elemento que queremos afectar y un nombre representativo:

```
 <p id="especial">Este párrafo va a tener un id como atributo, le daremos un nombre, y vamos a darle una propiedad color para destacarlo a nivel visual.</p>
```

En nuestra hoja de estilos tendremos la siguiente sintaxis para darle el color deseado:

```
 #especial {
    color:magenta;
}
```

Por atributo class

Al igual que ocurre con el id que puede colocarse sobre cualquier etiqueta, todas etiquetas aceptan el atributo **class=" "**.

Esa clase se usa cuando queremos **aplicar el mismo estilo a más de un elemento** y la búsqueda por etiqueta no sirve para lograrlo (porque no queremos darle a todos los párrafos las mismas características visuales) y usar id no nos sirve porque no se pueden repetir en el documento (no validaría nuestro html)

```
<ul>
  <li><a href="#" class="rojizo">Minecraft</a></li>
  <li><a href="#">League of Legends</a></li>
  <li><a href="#" class="rojizo">Mortal Kombat</a></li>
</ul>
```

En nuestro css, para referenciar al atributo class, lo buscaremos por punto:

```
.rojizo{
  color:indianred;
}
```

En nuestro browser se mostrarán las clases aplicadas a esos 2 vínculos, con el color asignado:

- Minecraft
- League of Legends
- Mortal Kombat

A tener presente:
El id es único por documento,
y el class se puede repetir

Reglas sintácticas

- Cada declaración CSS, está formada por un juego de pares, propiedad:valor; y debe estar dentro de las llaves {}
- No se ven afectadas por el espacio en blanco o tabulaciones.
- Los comentarios se hacen como en javascript /* comentario */ y será ignorado lo que tenga dentro.
- Siempre que la propiedad represente un número, el valor debe indicar bajo qué unidad se expresa.
  Entre el número y la unidad no pueden existir espacios.
- Para 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.

```
em, strong{ propiedad1: valor; propiedad2: valor; }
```

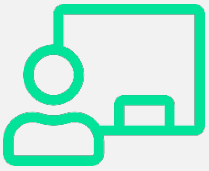
Color en CSS

Para aplicar valores de color a una propiedad, podemos hacerlo de distintas formas:

- Por su **keyword** (nombre del color en inglés). Por ejemplo: red
- Por notación **hexadecimal** (numeral + 6 caracteres de [0-9] de [a-f]). Por ejemplo: #ff0000
- Por notación del **modelo rgb** (red, green, blue) tres números de 0 a 255 separados por coma (es su forma más común).
  Admite un cuarto valor que es su canal alpha que nos permite manejar valores decimales donde 0 es completamente transparente y valores cercanos al 1 son más opacos, siendo ésta la notación para **rgba**(255, 0, 0, 0.8)
- Por notación del **modelo hsl** (hue, saturation, lightness.) –tono, saturación, luminosidad.
  El matiz es un entero entre 0 y 360, recorre todos los colores, mientras que la saturación y luz son porcentajes. Si queremos trabajar con el canal alpha podemos usar el **modelo hsla**, en donde le pasaremos un cuarto valor que será un número decimal entre 0 y 1, donde 0 significa completamente transparente y 1 significa completamente opaco. Por ejemplo: hsla(360, 100%, 50%, 0.8).
- Los colores habitualmente los obtenemos de la herramienta que usamos para trabajar con nuestros diseños (Photoshop, etc) con el picker de color podremos elegir las notaciones que nos ofrezca.

Cascada CSS

Es muy común escribir mal las reglas css y ver que no se están aplicando- Esto puede ocurrir por querer afectar al mismo elemento, pero usando distintos selectores, puede ser que no estemos aplicando las nociones básicas de **cascada, herencia y precedencia o de especificidad.**



Por ejemplo, si tenemos en siguiente código:

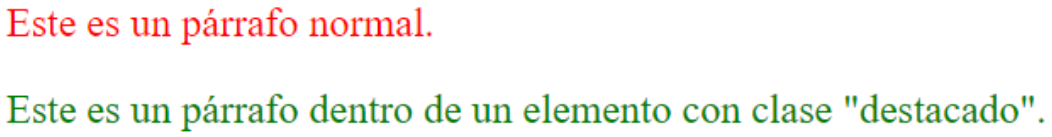
```
<p>Este es un párrafo normal.</p>

<div class="destacado">
  <p>Este es un párrafo dentro de un elemento con clase
"destacado".</p>
</div>
```

En nuestra hoja de estilos:

```
p {
   color: red;
 }
 .destacado p {
   color: green;
 }
```

La primera regla CSS establece el color de todas las etiquetas <p> en rojo. La segunda regla establece el color de todas las etiquetas <p> dentro de un elemento con clase "destacado" en verde.

La segunda regla se llama **selector de descendiente o selector de contexto**, especifica una regla que se aplicará a un elemento sólo si se encuentra dentro de un elemento padre específico.



El primer párrafo tiene un color rojo, mientras que el segundo párrafo tiene un color verde. Esto se debe a que **la regla más específica** (la regla para etiquetas <p> dentro de un elemento con clase "destacado") **se prioriza sobre la regla menos específica** (la regla para todas las etiquetas <p>).

La **cascada** nos permite especificar diferentes estilos para elementos similares y **tener control sobre cómo se aplican esos estilos** en función de la especificidad y la ubicación de los elementos dentro del documento HTML.

La **precedencia** se refiere a qué regla CSS se aplicará cuando existan varias reglas que pueden aplicarse a un mismo elemento. En otras palabras, la precedencia determina qué regla "gana" cuando hay conflictos entre varias reglas.

Hay varios factores que determinan la **precedencia** de una regla de estilo:

- **Especificidad**:
  La regla con un selector más específico tendrá mayor precedencia que una regla con un selector menos específico.
- **Orden en el archivo CSS**:
  Las reglas que aparecen más abajo en el archivo CSS tendrán precedencia sobre las reglas que aparecen antes de ellas.
- **Importancia**:
  Las reglas con la propiedad "!important" tendrán mayor precedencia que las reglas sin esta propiedad.

La combinación de estos factores determina la precedencia de una regla y por lo tanto, cuál de las reglas se aplica a un elemento en particular. En general, la regla con mayor precedencia "gana" y se aplica al elemento.

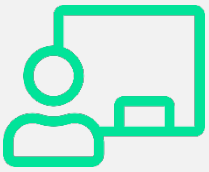
Por ejemplo, las reglas por id “ganan” a las reglas que llevan class o por su nombre de etiqueta. Las reglas por class “ganan” a las reglas que afectan a los elementos por el nombre de etiqueta.

Regla !important

Si tenemos un css mediano es poco probable que nuestras reglas "choquen".
La regla !important corta la precedencia y se escribe después del valor de la propiedad que se quiere convertir en importante.

```
p { color: red !important; }
.destacado p {color: green; }
```

Aplicado al ejemplo anterior, la regla que termina aplicándose es la primera y la segunda se anula.



El uso de **!important** es una mala práctica y debería evitarse porque hace que el código sea más difícil de depurar al romper la cascada.
Salvo que tengamos que romper la cascada por el uso de algún framework de css.

Se recomienda hacer un mejor uso de las propiedades en cascada de CSS o usar reglas más específicas.

Propiedades CSS para Texto

Para seguir profundizando, vamos a afectar a elementos que contengan texto y veremos su comportamiento y los posibles valores que aceptan estas propiedades.

Font-family

CSS tiene la propiedad **font-family** para indicar qué fuente utilizar en un texto. El valor de la tipografía debería estar entre comillas (principalmente si el nombre tiene espacios).
Tiene una restricción: La tipografía seleccionada debe estar instalada en la PC del usuario que entra a la web. Si esto no sucede, el browser sustituirá la fuente por alguna "de sistema".

Ejemplos de algunas de estas fuentes: Arial, Comic Sans, Courier New, Impact, Lucida Sans, Tahoma, Trebuchet, Times New Roman, Verdana.
Se pueden indicar varias familias separadas por coma, la primera de la lista que esté instalada será usada.

```
p { font-family: 'Courier New', Courier, monospace; }
```

Color

El color de fuente se define con la propiedad color, acepta nombre (en inglés), hexadecimal, notación rgb, etc.

```
h1 { color: rgb(0, 0, 255); }
```

Font-size

El tamaño de fuente se define con la propiedad font-size.
Es numérica, acepta como unidad px. (por ahora, hay otras unidades).

```
span { font-size: 20px; }
```

line-height

El tamaño de fuente va acompañado de un interlineado.
El **line-height** indica cuánto mide cada línea de texto si se llega al final y hay una segunda línea.
Debería ser igual o mayor al tamaño de fuente.

```
p { font-family: 'Courier New', Courier, monospace;
    font-size:20px;
     color:cornflowerblue;
    line-height: 26px;
}
```

Mayúsculas y minúsculas

La propiedad **text-transform** acepta como valores:

- **uppercase**: Pasa todo el texto a mayúsculas.
- **lowercase**: Pasa todo el texto a minúsculas.
- **capitalize**: Primera letra de cada palabra en mayúsculas.
- **none**: Elimina esta propiedad (se ve como fue escrito).

```
span { text-transform: uppercase; }
```

Font-weight

Para manipular la negrita se usa **font-weight**. Se quita con el valor **normal** y se aplica con **bold**.

```
span {
   text-transform: uppercase;
   font-weight: bold;
}
```

Font-style

La itálica se indica con **font-style**, que puede ser **normal** (la quita) o **italic** (lo hace itálica). Sí, podemos sacarle el bold a un encabezado o a un strong, lo mismo que la itálica de un em.

```
em { font-style: normal; }
```

Ahora que sabemos cómo modificar el tamaño/negrita de un texto, no nos olvidemos de respetar la semántica.

Espaciado de texto horizontal

Hay **3 propiedades que manejan el espaciado horizontal.** En todos los casos se usan **valores numéricos con su unidad** y aceptan tanto valores positivos como negativos.

- El **letter-spacing** define el espacio que habrá entre cada letra tipeada.
- El **word-spacing** indica cuál es el espacio de cada palabra, pero los caracteres se ven con su espaciado normal.
- La tabulación de la primera línea de texto (como en los libros) se define con el **text-indent.**

```
P {
   font-family: 'Courier New', Courier, monospace;
   font-size:20px;
   color:cornflowerblue;
   line-height: 26px;
   text-indent:20px;
}
```

Text-align

Al igual que pasa con los editores de texto, que permiten que los usuarios manejen la alineación de un texto, la propiedad **text-align** permite alinear elementos de texto. Sus valores posibles son: **left**, **right** y **center** (las más usadas en web).

```
h1 {
   color: rgb(0, 0, 255);
   text-align: center;
};
```

Subrayados

Podemos manipular el subrayado del texto con la propiedad **text-decoration-line,** sus posibles valores son**:**

- **underline** (debajo del texto)
- **overline** (sobre el texto)
- **line-through** (tachado)
- **none** (saca el subrayado)

Con la propiedad **text-decoration-color**, se puede modificar el color de ese subrayado.

También se puede definir el estilo que tendrá ese subrayado con la propiedad **text-decoration-style,** sus posibles valores son:

- solid (sólida)
- dashed (guiones)
- dotted (punteada)
- doublé (doble)
- wavy (ondulada)

Si quisiéramos modificar el grosor de ese subrayado podemos usar la propiedad **text-decoration-thickness, su valor es numérico con su unidad.**

```
em {
    font-style: normal;
    text-decoration-line: underline;
    text-decoration-color: red;
    text-decoration-style: solid;
    text-decoration-thickness: 4px;
}
```

Tenemos la posibilidad de **abreviar** el uso de estas propiedades con la propiedad **text-decoration** y pasarle los valores para las propiedades: **text-decoration-line, text-decoration-color, text-decoration-style y text-decoration-thickness** (no necesariamente en ese orden, como mínimo requiere especificar el valor para **text-decoration-line** que deseemos usar).

Text-shadow

Esta propiedad nos permite generar sombra en el texto y acepta 4 valores:

- **distancia** de la sombra y el elemento en el **eje X (valor numérico y su unidad)**
- **distancia** de la sombra y el elemento en el **eje Y (valor numérico y su unidad)**
- Una **expansión** de la sombra (se esfumará) **(valor numérico y su unidad).** Esta expansión es opcional, pero permite manejar el efecto que queramos lograr con la sombra.
- Un **color** de sombra (por hexadecimal, rgb, rgba o nombre). La elección del color de la sombra es opcional, por lo tanto, si no está definida tomará como color de sombra el color que posea el texto.

  Los valores numéricos pueden ser positivos o negativos dependiendo del efecto que deseamos lograr.
  El text-shadow permite aplicar más de una sombra a un mismo elemento.

```
span {
    text-transform:uppercase;
    font-weight:bold;
    letter-spacing: 3px;
    text-shadow: 2px 2px 2px coral;
 }
```

¿Cómo nos fue quedando?
¡Veamos el antes y el después de CSS!

Mortal Kombat

Durante nueve generaciones, **Shang Tsung, ha llevado a un poderoso príncipe a la victoria contra sus mortales enemigos.** Si vence el décimo torneo Mortal Kombat, la desolación y el mal que han florecido en el mundo reinarán la tierra por siempre. *¡Es hora de pelear por la Tierra!*

Mortal Kombat

Durante nueve generaciones, **Shang Tsung, ha llevado a un poderoso príncipe a la victoria contra sus mortales enemigos.** Si vence el décimo torneo **MORTAL KOMBAT**, la desolación y el mal que han florecido en el mundo reinarán la tierra por siempre. ¡Es hora de pelear por la Tierra!

Llegamos así al final de esta primera clase en la que vimos:

- ¿Qué es CSS?
- Concepto de Herencia.
- Formas de insertar CSS al HTML.
- Afectar a elementos desde CSS, (etiqueta, class, id).
- Reglas sintácticas.
- Manejo de color, notaciones de distintos modelos.
- Concepto de Cascada, precedencia. Regla **!important**.
- CSS para texto (color, align, size, family, spacing, indent, height, weight, style, decoration, transform, shadow).

Recuerden visualizar el **desafío semanal** que encontrarán en el aula y conectarse a la clase en vivo de esta semana.

¡Hasta la próxima clase!

Para ampliar la información:

Mozilla Developers Network, Documentación sobre CSS:
https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/CSS_basics