**M1S2: CSS y Responsive Design**

**¡Hola!, ¿qué tal?, ¿cómo te sientes?, ¿qué te han parecido las UClass?**

**Recuerda que, si tienes alguna duda, cuentas con tus coaches que podrán ayudarte al respecto y a quienes les podrás escribir desde la pestaña publicaciones en Teams para aclarar las dudas que tengas.**

**A continuación, seguimos con el contenido correspondiente a la semana 2, en donde iniciaremos con CSS.**

**Iniciemos…**

**ÍNDICE**

* [CSS](https://github.com/U-Camp/BOOT-M1-SEM2#css)
  + [Reglas y declaraciones](https://github.com/U-Camp/BOOT-M1-SEM2#reglas-y-declaraciones)
    - [Estructura](https://github.com/U-Camp/BOOT-M1-SEM2#estructura)
    - [Especificidad](https://github.com/U-Camp/BOOT-M1-SEM2#especificidad)
  + [Bordes](https://github.com/U-Camp/BOOT-M1-SEM2#bordes)
  + [Márgenes](https://github.com/U-Camp/BOOT-M1-SEM2#m%C3%A1rgenes)
  + [Padding](https://github.com/U-Camp/BOOT-M1-SEM2#padding)
  + [Fondos](https://github.com/U-Camp/BOOT-M1-SEM2#fondos)
  + [Texto y Fuentes](https://github.com/U-Camp/BOOT-M1-SEM2#texto-y-fuentes)
* [CSS: Responsive Design](https://github.com/U-Camp/BOOT-M1-SEM2#css-responsive-design)
  + [Media Queries](https://github.com/U-Camp/BOOT-M1-SEM2#media-queries)
* [CSS: Frameworks](https://github.com/U-Camp/BOOT-M1-SEM2#css-frameworks)
  + [Bootstrap](https://github.com/U-Camp/BOOT-M1-SEM2#bootstrap)
* [CSS: Transformaciones](https://github.com/U-Camp/BOOT-M1-SEM2#transformaciones)
* [CSS: Transiciones](https://github.com/U-Camp/BOOT-M1-SEM2#transiciones)

**CSS**

**HTML** permite definir la forma en la que se presentarán los elementos dentro de un navegador web, como lo vimos en la clase pasada, pero, no define los colores que se usarán, ni los tamaños o formato de los textos, este estilizado de la página se define con **CSS** .

Las Hojas de Estilo en Cascada (Cascading Style Sheets) son un esquema utilizado para describir la presentación de los elementos HTML presentes en una aplicación web, por lo que CSS describe cómo debe ser renderizado el elemento en pantalla.

Éstas dos tecnologías forman parte del rendering engine que todos los navegadores web integran para poder interpretar y cargar correctamente una página web, así mismo sus reglas, sintaxis y estándares las define el mismo consorcio W3C, por lo que el estándar debe ser respetado por las compañías que desarrollan cada navegador, se puede decir que el futuro de las dos tecnologías (HTML y CSS) está ligado y tienen el mismo objetivo.

Un ejemplo de un documento HTML que hace uso de CSS para estilizar los elementos se muestra a continuación:

/\* Archivo Estilos.css \*/

/\* Se aplica directamente al elemento <body> \*/

body {

color: blue;

}

/\* Se aplica directamente a todos los elementos de tipo <p> \*/

p {

color: red;

}

/\* Se aplica en el(los) elemento(s) donde se use \*/

.clase-css {

color: black;

}

*Ejemplo de código CSS*

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Ejemplo HTML y CSS</title>

<style>

body { /\* Se aplica directamente al elemento <body> \*/

color: gray;

}

h2 { /\* Se aplica directamente a todos los elementos de tipo <h2> \*/

color: navy;

}

/\* Se aplica sólo en el elemento donde se use con el atributo `class` \*/

.clase-css-tag {

color: green;

}

</style>

</head>

<body>

<h1>

Encabezado 1 de color gris.

</h1>

<h2>

Encabezado 2 de color azul marino.

</h2>

<p>

Párrafo para encabezado 2 de color rojo.

</p>

<h5 class="clase-css">

Encabezado 5 de color negro.

</h5>

<p class="clase-css-tag">

Párrafo para encabezado 5 de color verde.

</p>

Texto de color gris dentro de body

<p>Párrafo de color rojo</p>

<div class="clase-css">

Texto de color negro.

</div>

<p class="clase-css-tag" style="color: orange;">Texto de color naranja.</p>

<img

src="https://www.utel.edu.mx/sites/default/files/ingenieria-en-computacion.jpg"

alt="Texto alternativo para la imagen"

width="400"

height="200"

/>

</body>

</html>

*Ejemplo de código HTML con código CSS integrado*

**Hola, ¿cómo te sientes hasta ahora? ¿quisierás practicar lo aprendido o ver como se ve en línea? Si tu respuesta es afirmativa, copia y pega el código anterior en Visual Studio Code o ingresando al siguiente link, <https://codepen.io/pen/> así podrás validar cómo funciona y allí puedes agregar o eliminar líneas para que vayas practicando lo aprendido.**

**Seguimos...**

Así como HTML, CSS tampoco es un lenguaje de programación, es un esquema de diseño que permite aplicar estilos de manera selectiva a elementos que se encuentran en documentos HTML. En el ejemplo anterior, la regla p en el archivo estilos.css [para ver el ejemplo estilos.css presiona aquí](https://github.com/U-Camp/BOOT-M1-SEM2/blob/main/ejemplos/index.css) aplica de manera selectiva en todos los elementos párrafo dentro del archivo HTML, lo que convierte el texto dentro de esos elementos en color rojo.

Con esto claro y en vista de cómo se integran, vamos a conocer las características de CSS.

**Reglas y Declaraciones**

**Estructura**

La estructura general del esquema es llamada regla, la cual se compone de los siguientes elementos:

* **Selector:** Comienza la regla, selecciona el(los) elemento(s) a los cuales aplicar el estilo (body, p o .clase-css en el ejemplo).
* **Declaración:** Es una directiva que define las propiedades a cambiar dentro del elemento a estilar. Esta declaración se compone de propiedades, que son los atributos del elemento que se pueden estilar, así como de los valores de esas propiedades, que son las que se asignan después de los dos puntos y especifican la apariencia. (color: blue; color: red; o color: white; en el ejemplo).

**Especificidad**

Para hablar de especificidad, primero debemos determinar cómo están enlazados los archivos.

Accede a la carpeta ejemplos dentro de este repositorio y verás dos archivos:

* index.html. Contiene nuestra estructura base, incluyendo las etiquetas y enlaces externos, es decir a un archivo index.css donde contendremos nuestros estilos.
* index.css. Contiene nuestra hoja de estilos con selectores y propiedades.

Claro esto, hablemos dentro sobre especificidad.

¿Cómo funciona?

Los selectores son aplicados bajo un esquema de prioridad, es decir, el navegador web ya tiene asociado esas prioridades dentro del motor de renderizado y aplica las reglas dependiendo del lugar en el que se encuentren definidas (especificidad):

1. Primero, se aplican las reglas definidas en archivos externos (estilos.css en el ejemplo).
2. Segundo, se aplican las reglas definidas dentro de la etiqueta <style />, las reglas aquí definidas pueden sobrescribir cualquier regla ya definida en archivos externos (en el ejemplo, dentro del archivo estilos.css se define la regla para el body con el color azul, pero dentro de la etiqueta style éste se sobrescribe definiéndose como color gris, por lo que, al tener mayor prioridad, prevalece el gris).
3. En tercer lugar, se aplican las declaraciones definidas de manera inline, esto quiere decir, aquellas que se escriban directamente dentro del atributo style en cada elemento, éstas son individuales y aplican solamente para ese elemento en particular (en el ejemplo, último párrafo con texto de color naranja, éste puede aplicar el estilo para los párrafos p que se define dentro de estilos.css, pero como tiene asignada una clase "clase-css-tag", éste tiene mayor prioridad, sin embargo también tiene asignado el estilo inline con el atributo style, por lo que éste le gana a las anteriores y prevalece el color naranja).
4. Existe la palabra reservada !important, la cual si se aplica a cualquier declaración ésta sube en precedencia y tiene prioridad sobre el resto de las reglas, pero es considerada una mala práctica porque provoca que se rompa el esquema de prioridad de cascada, por lo que no se recomienda su uso.

Como pudiste ver, a partir de esta especificidad y prioridad en la aplicación de estilos es de donde viene el nombre CSS, ya que depende de esa prioridad para aplicar los estilos de manera figurativa a una cascada.

Como parte de las reglas existen los selectores de clase, este tipo de selectores sólo aplican el estilo en aquellos elementos que se definen dentro del HTML con la palabra class, el cual es un atributo de HTML y no debe confundirse con la clase que define a un objeto, ya que son dos cosas diferentes.

**Bordes**

La directiva border permite especificar el estilo, tamaño y color del borde de un elemento en particular, se define por las siguientes sub-directivas:

* **border-style:** Especifica el tipo de borde a mostrar, soporta varios valores para distintos tipos de línea de borde: dotted, dashed, solid, double, groove, ridge, inset, outset, none y hidden. Así mismo acepta 1, 2 y 4 valores distintos, dependiendo del lado del borde que se quiere especificar: top right bottom y left. Ejemplo border-style: dotted; aplica en los 4 bordes, border-style: dotted dashed; aplica dotted para el top y el bottom y dashed para los lados laterales (left y right) border-style: dotted dashed solid double; aplica para cada lado (top right bottom left).
* **border-width:** Especifica el ancho de cada uno de los 4 bordes de cada elemento, puede estar dado en un tamaño específico (px, pt, cm, em) o usando uno de los tres valores pre-definidos (thin, medium, thick). Así mismo también acepta desde 1, 2 y 4 valores. Ejemplo border-width: 1px; aplica en los 4 bordes, border-width: 1px thin; aplica 1px para el top y el bottom y thin para los lados laterales (left y right) border-width: 1px 2px 3px medium; aplica para cada lado (top right bottom left).
* **border-color:** Especifica el color con el que se mostrará el borde. Acepta los mismos valores que la directiva color, aceptando además un valor adicional transparent, el cual se aplica para transparencias y opacidades, las cuales se verán más adelante. Ejemplo border-color: #ff0000;.

Para minimizar el código se pueden especificar todos los valores del borde en una misma propiedad, en lugar de escribir cada una de las directivas, se declara una sola propiedad con todos los valores para cada una de las directivas border: 5px solid red;. El orden es border: width style color y debe ser respetado.

.borde-completo {

border: 5px solid red;

}

.bordes-laterales {

/\* El orden es top right bottom left \*/

border-style: none dotted none dotted;

border-width: 2px;

border-color: #00AA00;

}

*Ejemplo de Bordes en CSS*

**Márgenes**

Los márgenes (margin) representan espacios alrededor de los elementos, es decir, define el tamaño del espacio entre un elemento y otro, el espacio entre sus bordes correspondientes. Esta directiva acepta 4 posibles valores: **auto** con el que el navegador calcula automáticamente el margen de cada elemento (comportamiento default), **una longitud específica** (px, pt, cm, em), un **valor en porcentaje** (%) que es la proporción con relación al elemento superior que lo contiene y finalmente **inherit**, lo cual indica que el elemento hereda el margen definido en el elemento superior que lo contiene.

Se pueden usar las directivas específicas para un margen de cada lado, de la siguiente manera:

* **margin-top:** Especifica el espacio de margen superior. Ejemplo margin-top: 2px;.
* **margin-right:** Especifica el espacio de margen a la derecha del elemento. Ejemplo margin-rigth: 1.2em;.
* **margin-bottom:** Especifica el espacio de margen inferior. Ejemplo margin-bottom: 10%;.
* **margin-left:** Especifica el espacio de margen a la izquierda del elemento. Ejemplo margin-left: inherit;

Igual que con los bordes, se puede minimizar código especificando la directiva margin y asignando los valores de la propiedad en el orden en el que se especifican, es decir margin: 2px 1.2em 10% inherit;, respetando el orden top right bottom left.

.margen-completo {

/\* El orden es top right bottom left \*/

margin: 2px 2.2em 2% inherit;

}

.margen-superior-inferior {

margin-top: 2px;

margin-bottom: 2.2em;

}

*Ejemplo de Márgenes en CSS*

**Padding**

La directiva padding se usa para definir el tamaño del espacio alrededor del contenido del elemento, es decir, dentro de los límites de los bordes, desde el borde hasta el contenido en sí del elemento. De igual manera, acepta los mismos valores que la directiva margin exceptuando el valor auto, el cual no está soportado, pero sí acepta longitudes específicas (px, pt, cm, em), un valor en porcentaje (%) que es la proporción en relación con el elemento superior que lo contiene y finalmente inherit, lo cual indica que el elemento hereda el padding definido en el elemento superior que lo contiene.

Igualmente, soporta directivas para especificar cada lado padding-top, padding-right, padding-bottom y padding-left, los cuales se usan de la misma manera que los márgenes, así mismo, también soporta el minimizado de código con la directiva padding: top right bottom left.

.padding-completo {

/\* El orden es top right bottom left \*/

padding: 2px 2pt 1% inherit;

}

.padding-por-lado {

padding-top: 2px;

padding-right: 10px;

padding-bottom: 5px;

padding-left: 1px;

}

*Ejemplo de Padding en CSS*

**Colores**

Para especificar colores de texto en un elemento HTML usando CSS, se usa la declaración color, la cual acepta cualquiera de los 140 nombres de colores predefinidos que los navegadores web tienen integrados (ejemplo color: red;).

Así mismo, puede aceptar valores en RGB:

* RGB: color: rgb(255,0,0);
* Hexadecimal: color: #FF0000;
* HSL: color: hsl(0, 100%, 50%);
* HWB: color: hwb(0, 0%, 0%);

**Fondos**

Para especificar fondos se usa la declaración background , la cual se divide en cuatro directivas:

**1. background-color:** Para asignar el color de fondo de un elemento, acepta los mismos valores que acepta la directiva color. Ejemplo `background-color: #FF0000;.

**2. background-image:** Especifica una imagen para ser usada como fondo de un elemento en particular. Por default, si la imagen es más chica que el elemento que la contiene, ésta se repite de tal manera que cubra completamente el elemento. Acepta sólo texto con valores de tipo URL. Ejemplo background-image: url("path/hacia/una/imagen.jpg”);

**3. background-repeat:** Como te comenté, el comportamiento que el navegador tiene es repetir la imagen por default hasta llenar el elemento que la contiene, sin embargo, con esta directiva se puede cambiar ese comportamiento, de tal manera que se pueda repetir sólo horizontalmente (ejemplo background-repeat: repeat-x;), verticalmente (ejemplo background-repeat: repeat-y;) o mostrar la imágen sólo una vez (ejemplo background-repeat: no-repeat;).

**4. background-position:** Esta directiva permite establecer la posición de inicio de las imágenes en relación al elemento que las contiene, por default el explorador las ubica en la parte superior izquierda del elemento y a partir de ahí empieza a dibujar la imagen, pero se puede cambiar el comportamiento para que la imagen se ubique en la parte superior derecha (ejemplo background-position: right top;), esta directiva acepta 4 posibles valores indicando la posición relativa: left, top, right, bottom.

Para minimizar el código se pueden especificar todos los valores del fondo en una misma propiedad, en lugar de escribir cada una de las directivas, se declara una sola propiedad con todos los valores para cada una de las directivas:

background: #FF0000 url("path/hacia/una/imagen.jpg") no-repeat right top;.

Para que sea correctamente interpretado se tiene que respetar el orden en el que fueron explicadas (background: color image repeat position).

**Texto y Fuentes**

La mayoría de las páginas y aplicaciones web, usan fuentes de texto específicas que van con una imagen de la marca o de la empresa para la que se está desarrollando la aplicación.

Para poder configurar el texto y que el explorador cargue las fuentes específicas para esta página y las aplique, se usan varias directivas que permiten especificar la familia de la fuente, su tipo, su formato, tamaño y estilo de un texto en específico, las cuales son las siguientes:

* **font-family:** Establece la familia de la fuente a usar para el texto, puede contener varias fuentes en orden de prioridad por si el navegador no puede cargar alguna o no la soporta intentar la siguiente. Ejemplo font-family: Arial, Helvetica, sans-serif;.
* **font-style:** Especifica si se aplica un estilo itálico al texto. Tiene 3 posibles valores: normal, italic y oblique, aunque este último es menos soportado en los navegadores. Ejemplo font-style: italic;.
* **font-weight:** Especifica si se aplica un estilo “negrita” al texto. Tiene 2 posibles valores: normal y bold. Ejemplo font-weight: bold;.
* **font-size:** Especifica el tamaño del texto. Acepta valores absolutos (como son tamaños estáticos dados en pixeles) o valores relativos (que son dados en proporción al elemento que los contiene). Ejemplo font-size: 16px; o font-size: 2.5em;.

Para acortar código, es posible usar la directiva font directamente y asignar esos valores, no requiriendo estar en algún orden en específico, pero sí, al menos que se defina el tamaño y la familia (ejemplo font: 12px Arial, sans-serif;).

**¿Qué te pareció el contenido de CSS? Si deseas puedes consultar y descargar una infografía sobre este contenido, [Presiona aquí](https://github.com/U-Camp/BOOT-M1-SEM2/blob/main/infografia/M1_S2_Infografia_css.pdf)**

**CSS: Responsive Design**

Trabajar en la web nos obliga a considerar siempre a nuestros usuarios y a los dispositivos en los cuales consumirán nuestras aplicaciones.

Cabe destacar que, cuando empezamos a diseñar para diferentes dispositivos el área de trabajo es inmensa. Sólo consideraremos lo que tenemos involucrado:

* Móviles,
* Tabletas,
* Computadoras,
* Monitores largos,
* Televisiones,
* Pantallas de cine, entre otras.

Para ello, dentro de CSS, podemos encontrar estructuras que permitan flexibilidad. De acuerdo al comportamiento y el dispositivo en el cual el usuario está situado, se contendrán diferentes estilos.

A esta práctica se le conoce como *Responsive Design*, o también, Diseño Responsivo.

El elemento más importante para poder ejecutar esto son los media queries.

**Media Queries**

Un media query se refiere a una regla que incluye un bloque de propiedades CSS si se cumple una condición específica.

Está conformado por:

* @media. Indica que aplicaremos un media query.
* screen. Indica que estaremos trabajando con pantallas. Existen otras opciones que puedes observar en la [documentación](https://www.w3.org/TR/CSS2/media.html) como print o speech. Sin embargo, screen es indicada para este ejemplo.
* &. Indica que vamos a agregar una condición adicional.
* max-width: 800px. Indica que vamos a establecer un límite máximo de 800px como condición. Si la pantalla tiene 800px o menos, se ejecutará ésta. A esto se le conoce como punto de quiebre o breakpoint. También existe el contrario que es min-width, el cual es la medida mínima.

Ejemplo:

@media screen & (max-width: 800px){

body{

background-color: blue;

}

}

Observa en este siguiente ejemplo el cambio con una medida mínima.

@media screen & (min-width: 800px){

body{

background-color: blue;

}

}

Dependiendo del dispositivo, se recomienda trabajar en estos rangos:

* 320px — 480px: Dispositivos móviles.
* 481px — 768px: iPads, Tabletas.
* 769px — 1024px: Pequeñas pantallas, computadoras portátiles.
* 1025px — 1200px: Computadoras de escritorio, pantallas amplias.
* 1201px o más: Pantallas extra grandes, televisiones.

Consideraciones importantes:

* CSS trabaja en un formato en cascada. Es decir, tan pronto exista la coincidencia con un media query con respecto a las condiciones, se ejecutará esos estilos específicos.
* Puedes probar tus media queries tomando la ventana del navegador y haciendola más pequeña y más grande.

**CSS: Frameworks**

CSS cuenta con herramientas adicionales, propuestas por organizaciones o proyectos *Open Source*, que permiten desarrollar con mayor velocidad y bajo una estructura específica.

A estas herramientas se les conocen como *frameworks*.

Uno de los más populares y sólidos es **Bootstrap**.

**Bootstrap**

Bootstrap, es un framework front-end basado en CSS y JavaScript, el cual se usa para desarrollar aplicaciones web responsivas, esto es desarrollar páginas y aplicaciones web que se vean bien tanto en escritorio, como en dispositivos móviles (tabletas y celulares). Fue creado por Twitter en 2010 y desde entonces ha gozado de amplia popularidad en el mundo del desarrollo de software, ya que es un framework que es muy fácil de aprender e implementar.

Provee una serie de herramientas que permiten estilizar los elementos HTML, así como plantillas de componentes que ya se encuentran estilizados, tales como fuentes, tipografías, iconos, formularios, botones, tablas, menús de navegación, modales, carruseles de imágenes, así como un amplio abanico de herramientas de JavaScript, las cuales se pueden reusar, ampliando la funcionalidad de cualquier sitio web.

Bootstrap 5 es la versión más reciente y soporta la mayor parte de los navegadores web, excepto por versiones anteriores de IE9, incluyéndose.

Como ya se comentó, Bootstrap es muy fácil de usar, por lo que cualquier persona con conocimiento básico en HTML y CSS puede implementarla; con su esquema de filas y columnas, permite una responsividad que se ajusta fácil y automáticamente a cualquier tipo de pantalla.

Para poder implementar Bootstrap, se puede hacer de varias maneras:

**1. Usando el CDN (Content Delivery Network ) de Bootstrap:** provee la última versión del CSS y del JS de Bootstrap y se incluyen directamente en la página HTML enlazando los archivos necesarios directamente en la etiqueta de la página.

**2. Descargando los archivos directamente de la página e incluyéndolos directamente en el paquete de la aplicación**\*, esto es haciendo referencias igual en el HEAD, pero con enlaces relativos a su ubicación dentro de la página web.

**3. Instalándolo con “npm”, usando el comando install: $ npm install bootstrap.**\* Bajo este método se instala en la carpeta node\_modules del proyecto y se puede referenciar al CSS y al JS directamente importándolos en los archivos en los que se vayan a usar.

Ejemplo:

<!DOCTYPE html>

<html lang="en">

<head>

<!-- Última versión del CSS minificado -->

<link

rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<!-- Última versión del JS minificado -->

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js">

</script>

</head>

</html>

*Ejemplo de Referencias a Bootstrap*

**Filosofía mobile-first en Bootstrap**

Se establece que cualquier diseño debe funcionar y ser intuitivo primero en dispositivos móviles y luego en escritorio.

Para asegurar este comportamiento, se le indica al navegador web el tipo de diseño de la página y el zoom del mismo, por lo que se asigna un metadato en el HEAD de la página para que el explorador actúe de manera adecuada:

Ejemplo:

<meta name="viewport" content="width=device-width, initial-scale=1">

**Elementos**

La base principal de Bootstrap son los siguientes elementos:

**o Contenedores:** son elementos que envuelven otros elementos, normalmente son elementos <div /> a los cuales se le especifica la clase CSS container o container-fluid. Normalmente a este tipo de contenedores se les pueden asignar otras clases de tal manera que estilen el fondo y el texto de los elementos contenidos, como por ejemplo bg-dark o text-white.

**o Grids:** El sistema de filas y columnas de Bootstrap es su principal cualidad, ya que permite el acomodo de los elementos dentro de una rejilla virtual que se divide en 12 columnas por fila, pudiendo usar tantas filas como sean necesarias para mostrar la página.

Se usa aplicando las clases row y col, la clase col acepta modificadores dependiendo de las columnas y tamaño del dispositivo, por ejemplo col-md-4 indica que es una columna de tamaño mediano y que abarca 4 posiciones de las 12 disponibles, pudiendo poner más elementos en las siguientes 8 posiciones que quedan, cambiando el número final de la clase, es decir que por cada fila row se pueden poner hasta 12 elementos col-md-1, uno por columna o hasta 6 elementos col-md-2 y así sucesivamente hasta ocupar las 12 posiciones. Normalmente se usan elementos <div /> para manejar este sistema de filas y columnas.

Ejemplo:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Ejemplo Bootstrap</title>

<!-- Última versión del CSS minificado -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<!-- Última versión del JS minificado -->

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

</head>

<body>

<div class="container p-3 my-3 bg-dark text-white">

<h1>Contenedor</h1>

<p>Fondo negro y texto blanco.</p>

</div>

<div class="container">

<div class="row">

<div class="col-md-4">

<h3>Header de Columna 1</h3>

<p>Párrafo en columna 1</p>

</div>

<div class="col-md-4">

<h3>Header de Columna 2</h3>

<p>Párrafo en columna 2</p>

</div>

<div class="col-md-4">

<h3>Header en Columna 3</h3>

<p>Párrafo en columna 3</p>

</div>

</div>

</div>

</body>

</html>

*Ejemplo de Contenedores y Grid usando Bootstrap*

En el ejemplo, si se renderiza esta página web y luego se le cambia el tamaño, se podrá ver cómo cambian de posición los elementos de tal manera que se adaptan al tamaño del dispositivo en el que se observan. La clase p-3 indica que se dará un padding de 3 pixeles, así mismo la clase my-3 añade un margen de 3 pixeles.

Básicamente estos son los elementos principales de Bootstrap, se irán conociendo y definiendo más componentes conforme se vaya avanzando en las clases.

**¿Cómo vas?, CSS es super interesante ya que le da más forma a nuestros desarrollos.**

**Recuerda que si deseas practicar y ver cómo se ve en línea puedes hacerlo en VSC o en <https://codepen.io/pen/> solo copiando y pegando el código, así puedes hacer los cambios y pruebas que necesites. ¡Mientras más practiques, más rápido aprenderás!**

**Seguimos... ☺️**

**Componentes: Alertas**

El componente alert de Bootstrap permite mostrar mensajes al usuario de tal manera que llamen su atención y opcionalmente, que el mismo usuario los pueda quitar de su vista, a manera de notificaciones. Para lograr esto, se usa la clase alert alert-<tipo>, donde tipo es el tipo de alert que se va a mostrar, permitiendo 8 distintos tipos (alert-primary, alert-secondary, alert-success, alert-danger, alert-warning, alert-info, alert-light y alert-dark), con lo que se puede estilar cualquier tipo de mensaje.

Ejemplo:

<div class="mover-elemento borde alert alert-primary">

Este elemento se movió desde 0,0

<button type="button" class="close" data-dismiss="alert" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

*Ejemplo de alert con bootstrap*

**Badges**

Los badges son elementos mayormente visuales que son usados para agregar información adicional a cualquier tipo de contenido. Se usa con la clase badge seguida de una clase contextual badge-<tipo\_badge>, que como ya se ha visto en otros componentes, acepta los mismos 8 posibles valores (badge-primary, badge-secondary, badge-success, badge-danger, badge-warning, badge-info, badge-light y badge-dark).

**Transformaciones**

CSS es capaz de realizar transformaciones matemáticas a elementos HTML, esto quiere decir que, aplicando transformaciones, CSS puede mover elementos de una posición X a una posición Y o escalarlos en determinadas proporciones o rotarlos en ciertas direcciones. Para lograr esto, CSS tiene ciertas **"reglas especiales"** que funcionan como métodos de estilo, las cuales permiten realizar este tipo de acciones sobre los elementos y se definen usando la regla transform seguida del método que ejecutará la transformación. De estos métodos los más usados son los siguientes:

* **translate:** Este método permite mover un elemento a partir de su posición original, de acuerdo a los parámetros dados para el eje X y el eje Y. Ejemplo transform: translate(50px, 100px); se mueve el elemento 50 píxeles a la derecha y 100 píxeles hacia abajo. Este método se acompaña con 2 métodos extra translateX y translateY los cuales permiten mover un elemento en los ejes X y Y respectivamente de manera individual.
* **rotate:** Permite rotar el elemento en sentido horario o contra-horario, dependiendo de los grados que se le pasen como parámetro. Ejemplo transform: rotate(45deg); rota el elemento 45 grados en sentido de las manecillas del reloj (izquierda -> derecha) o transform: rotate(-45deg); rota el elemento en 45 grados en sentido contrario a las manecillas del reloj (derecha -> izquierda). Hay métodos extra de rotación, que permiten manejar los 3 ejes, de tal manera que rote el elemento en 3D:
* **rotateX:** Rota el elemento a determinada cantidad de grados sobre el eje X. Ejemplo transform: rotateX(45deg);.
* **rotateY:** Rota el elemento a determinada cantidad de grados sobre el eje Y. Ejemplo transform: rotateY(45deg);.
* **rotateZ:** Rota el elemento a determinada cantidad de grados sobre el eje Z. Ejemplo transform: rotateZ(45deg);.
* **scale:** Incrementa o decrementa el tamaño del elemento de acuerdo con la proporción dada en los parámetros para el ancho y el alto. Ejemplo transform: scale(2, 3); incrementa el tamaño del elemento dos veces su ancho y 3 veces su alto; transform: scale(0.5, 0.5); decrementa el tamaño del elemento la mitad de su ancho y la mitad de su alto. Este método se acompaña de 2 métodos extra scaleX y scaleY los cuales permiten incrementar o decrementar el ancho o el alto respectivamente de manera individual.
* **skew:** Inclina el elemento sobre el eje X o Y cierta cantidad de grados dependiendo de los parámetros dados. Ejemplo transform: skew(20deg, 10deg); inclina el elemento 20 grados en el eje X y 10 grados en el eje Y. Este método se acompaña de 2 métodos extra skewX y skewY los cuales permiten inclinar sobre el eje X o el eje Y respectivamente de manera individual.
* **matrix:** Este método combina todos los anteriores en uno solo, toma 6 parámetros que corresponden a cada una de los métodos individuales de transformación en el orden matrix(scaleX, skewY, skewX, scaleY, translateX, translateY). Ejemplo transform: matrix(1, -0.3, 0, 1, 0, 0); el ancho del elemento se queda igual, ya que la escala es 1; pero lo inclina negativamente 0.3 grados sobre el eje Y, sin inclinarlo sobre el eje X y manteniendo la misma altura ya que la escala es 1, sin moverlo sobre el eje X ni sobre el eje Y. Se usa más para minimizar código.

Ejemplo: /\* Mueve el elemento 50 píxeles a la derecha y 100 píxeles hacia abajo \*/

.mover-elemento {

transform: translate(50px, 100px);

}

/\* Rota el elemento 45 grados \*/

.rotar-elemento-positivo {

transform: rotate(45deg);

}

/\* Rota el elemento -45 grados \*/

.rotar-elemento-negativo {

transform: rotate(-45deg);

}

/\* Incrementa el elemento dos veces su ancho y 3 veces su alto \*/

.incrementar-elemento {

transform: scale(2, 3);

}

/\* Decrementa el elemento la mitad de su ancho y alto \*/

.disminuir-elemento {

transform: scale(0.5, 0.5);

}

/\* Inclina el elemento 20 grados en el eje X y 10 grados en el eje Y \*/

.inclinar-elemento {

transform: skew(20deg, 10deg);

}

/\* Transforma el elemento con el shortcut `matrix` \*/

.transformar-elemento {

/\* matrix(scaleX, skewY, skewX, scaleY, translateX, translateY) \*/

transform: matrix(1, -0.3, 0, 1, 0, 0);

}

/\* Rota el elemento en los grados dados para cada eje \*/

.rotar-elemento-3D {

transform: rotateX(150deg);

transform: rotateY(130deg);

transform: rotateZ(90deg);

}

**Transiciones**

Además de las transformaciones matemáticas que habilita CSS, éste también ofrece otros métodos matemáticos para realizar cambios en los valores de las propiedades de un elemento de manera gradual, de tal manera que se pueda observar visualmente el cambio. Para poder crear una transición, se requieren dos cosas: La propiedad CSS sobre la cual aplicar el cambio y la duración del efecto (ejemplo: transition: width 2s;, se le va a aplicar el ancho especificado al elemento en una transición que dure dos segundos), si no se especifica una duración, el valor default es cero, por lo que no ocurre ninguna transición, así mismo la transición es aplicada hasta el momento en el que el cambio de tamaño ocurre.

Las transiciones pueden tener distintos efectos, de tal manera que visualmente se note una distinta transición en cada elemento, para lo cual se ocupa la propiedad transition-timing-function, la cual permite 6 posibles valores:

**1. ease:** Es el valor predeterminado, específica que la transición empiece lenta, luego rápida y finalice lenta.

**2. linear:** Especifica que la transición tenga una misma velocidad de inicio a fin.

**3. ease-in:** Especifica una transición con un inicio lento.

**4. ease-out:** Especifica una transición con un fin lento.

**5. ease-in-out:** Especifica una transición con un inicio y fin lentos.

**6. cubic-beizer:** Permite especificar la curva de velocidad de la transición por medio de una función cubic-beizer , la cual recibe 4 parámetros (x1, y1, x2, y2) que corresponden a los dos puntos coordenados de la curva (inicial y final entre 0 y 1) con los cuales calcular la velocidad de la transición. Ejemplo: transition-timing-function: cubic-beizer(0.75, 0.56, 0.19, 0.67);.

También es posible especificar un retraso en la aplicación de la transición, usando la propiedad transition-delay la cual recibe un valor en segundos que especifica el tiempo de retraso antes de que comience la transición.

Se pueden especificar dos o más propiedades y aplicarles una transición, incluso se puede hasta aplicar transiciones a transformaciones (ejemplo: transition: width 2s, height 4s, transform 2s;).

También acepta los valores del efecto directamente, de tal manera que todo se pueda especificar en una misma propiedad de ésta manera transition: <transición> <efecto> <retraso>; (ejemplo transition: width 2s linear 1s;).

Ejemplo:

/\* Transición sobre todos los divs del documento HTML \*/

div {

width: 500px;

height: 100px;

background: gray;

transition: width 2s;

}

/\* Cambia el tamaño del div cuando se pasa el ratón sobre éste \*/

div:hover {

width: 800px;

}

.transicion-linear {

transition-timing-function: linear;

}

.transicion-ease {

transition-timing-function: ease;

}

.transicion-ease-in {

transition-timing-function: ease-in;

}

.transicion-ease-out {

transition-timing-function: ease-out;

}

transicion-ease-in-out {

transition-timing-function: ease-in-out;

}

/\* Transición asignada con propiedades separadas \*/

.transicion-separada {

transition-property: width;

transition-duration: 2s;

transition-timing-function: linear;

transition-delay: 1s;

}

/\* Transición asignada en una misma propiedad \*/

.transicion-completa {

transition: width 2s linear 1s;

}

***Ejemplo transiciones en CSS***

**Hola, espero que lo que te mostré anteriormente te haya sido fácil de comprender, de lo contrario recuerda que puedes leerlo cada vez que lo necesites. También es importante que puedas tomar un descanso, recuerda que nuestro cuerpo y nuestra mente necesitan un respiro, por lo que te invito a ver el [siguiente video](https://www.youtube.com/watch?v=tA2kT8eSjtg).**

**¿Qué te pareción el video?, espero te sientas mucho mejor, y ahora si a descansar.**

**¡Nos vemos pronto!**

**M1S2: CSS y Responsive Design**

**¡Hola!, ¿qué tal?, ¿cómo te sientes?, ¿qué te han parecido las UClass?**

**Recuerda que, si tienes alguna duda, cuentas con tus coaches que podrán ayudarte al respecto y a quienes les podrás escribir desde la pestaña publicaciones en Teams para aclarar las dudas que tengas.**

**A continuación, seguimos con el contenido correspondiente a la semana 2, en donde iniciaremos con CSS.**

**Iniciemos…**

**ÍNDICE**

* [CSS](https://github.com/U-Camp/BOOT-M1-SEM2#css)
  + [Reglas y declaraciones](https://github.com/U-Camp/BOOT-M1-SEM2#reglas-y-declaraciones)
    - [Estructura](https://github.com/U-Camp/BOOT-M1-SEM2#estructura)
    - [Especificidad](https://github.com/U-Camp/BOOT-M1-SEM2#especificidad)
  + [Bordes](https://github.com/U-Camp/BOOT-M1-SEM2#bordes)
  + [Márgenes](https://github.com/U-Camp/BOOT-M1-SEM2#m%C3%A1rgenes)
  + [Padding](https://github.com/U-Camp/BOOT-M1-SEM2#padding)
  + [Fondos](https://github.com/U-Camp/BOOT-M1-SEM2#fondos)
  + [Texto y Fuentes](https://github.com/U-Camp/BOOT-M1-SEM2#texto-y-fuentes)
* [CSS: Responsive Design](https://github.com/U-Camp/BOOT-M1-SEM2#css-responsive-design)
  + [Media Queries](https://github.com/U-Camp/BOOT-M1-SEM2#media-queries)
* [CSS: Frameworks](https://github.com/U-Camp/BOOT-M1-SEM2#css-frameworks)
  + [Bootstrap](https://github.com/U-Camp/BOOT-M1-SEM2#bootstrap)
* [CSS: Transformaciones](https://github.com/U-Camp/BOOT-M1-SEM2#transformaciones)
* [CSS: Transiciones](https://github.com/U-Camp/BOOT-M1-SEM2#transiciones)

**CSS**

**HTML** permite definir la forma en la que se presentarán los elementos dentro de un navegador web, como lo vimos en la clase pasada, pero, no define los colores que se usarán, ni los tamaños o formato de los textos, este estilizado de la página se define con **CSS** .

Las Hojas de Estilo en Cascada (Cascading Style Sheets) son un esquema utilizado para describir la presentación de los elementos HTML presentes en una aplicación web, por lo que CSS describe cómo debe ser renderizado el elemento en pantalla.

Éstas dos tecnologías forman parte del rendering engine que todos los navegadores web integran para poder interpretar y cargar correctamente una página web, así mismo sus reglas, sintaxis y estándares las define el mismo consorcio W3C, por lo que el estándar debe ser respetado por las compañías que desarrollan cada navegador, se puede decir que el futuro de las dos tecnologías (HTML y CSS) está ligado y tienen el mismo objetivo.

Un ejemplo de un documento HTML que hace uso de CSS para estilizar los elementos se muestra a continuación:

/\* Archivo Estilos.css \*/

/\* Se aplica directamente al elemento <body> \*/

body {

color: blue;

}

/\* Se aplica directamente a todos los elementos de tipo <p> \*/

p {

color: red;

}

/\* Se aplica en el(los) elemento(s) donde se use \*/

.clase-css {

color: black;

}

*Ejemplo de código CSS*

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Ejemplo HTML y CSS</title>

<style>

body { /\* Se aplica directamente al elemento <body> \*/

color: gray;

}

h2 { /\* Se aplica directamente a todos los elementos de tipo <h2> \*/

color: navy;

}

/\* Se aplica sólo en el elemento donde se use con el atributo `class` \*/

.clase-css-tag {

color: green;

}

</style>

</head>

<body>

<h1>

Encabezado 1 de color gris.

</h1>

<h2>

Encabezado 2 de color azul marino.

</h2>

<p>

Párrafo para encabezado 2 de color rojo.

</p>

<h5 class="clase-css">

Encabezado 5 de color negro.

</h5>

<p class="clase-css-tag">

Párrafo para encabezado 5 de color verde.

</p>

Texto de color gris dentro de body

<p>Párrafo de color rojo</p>

<div class="clase-css">

Texto de color negro.

</div>

<p class="clase-css-tag" style="color: orange;">Texto de color naranja.</p>

<img

src="https://www.utel.edu.mx/sites/default/files/ingenieria-en-computacion.jpg"

alt="Texto alternativo para la imagen"

width="400"

height="200"

/>

</body>

</html>

*Ejemplo de código HTML con código CSS integrado*

**Hola, ¿cómo te sientes hasta ahora? ¿quisierás practicar lo aprendido o ver como se ve en línea? Si tu respuesta es afirmativa, copia y pega el código anterior en Visual Studio Code o ingresando al siguiente link, <https://codepen.io/pen/> así podrás validar cómo funciona y allí puedes agregar o eliminar líneas para que vayas practicando lo aprendido.**

**Seguimos...**

Así como HTML, CSS tampoco es un lenguaje de programación, es un esquema de diseño que permite aplicar estilos de manera selectiva a elementos que se encuentran en documentos HTML. En el ejemplo anterior, la regla p en el archivo estilos.css [para ver el ejemplo estilos.css presiona aquí](https://github.com/U-Camp/BOOT-M1-SEM2/blob/main/ejemplos/index.css) aplica de manera selectiva en todos los elementos párrafo dentro del archivo HTML, lo que convierte el texto dentro de esos elementos en color rojo.

Con esto claro y en vista de cómo se integran, vamos a conocer las características de CSS.

**Reglas y Declaraciones**

**Estructura**

La estructura general del esquema es llamada regla, la cual se compone de los siguientes elementos:

* **Selector:** Comienza la regla, selecciona el(los) elemento(s) a los cuales aplicar el estilo (body, p o .clase-css en el ejemplo).
* **Declaración:** Es una directiva que define las propiedades a cambiar dentro del elemento a estilar. Esta declaración se compone de propiedades, que son los atributos del elemento que se pueden estilar, así como de los valores de esas propiedades, que son las que se asignan después de los dos puntos y especifican la apariencia. (color: blue; color: red; o color: white; en el ejemplo).

**Especificidad**

Para hablar de especificidad, primero debemos determinar cómo están enlazados los archivos.

Accede a la carpeta ejemplos dentro de este repositorio y verás dos archivos:

* index.html. Contiene nuestra estructura base, incluyendo las etiquetas y enlaces externos, es decir a un archivo index.css donde contendremos nuestros estilos.
* index.css. Contiene nuestra hoja de estilos con selectores y propiedades.

Claro esto, hablemos dentro sobre especificidad.

¿Cómo funciona?

Los selectores son aplicados bajo un esquema de prioridad, es decir, el navegador web ya tiene asociado esas prioridades dentro del motor de renderizado y aplica las reglas dependiendo del lugar en el que se encuentren definidas (especificidad):

1. Primero, se aplican las reglas definidas en archivos externos (estilos.css en el ejemplo).
2. Segundo, se aplican las reglas definidas dentro de la etiqueta <style />, las reglas aquí definidas pueden sobrescribir cualquier regla ya definida en archivos externos (en el ejemplo, dentro del archivo estilos.css se define la regla para el body con el color azul, pero dentro de la etiqueta style éste se sobrescribe definiéndose como color gris, por lo que, al tener mayor prioridad, prevalece el gris).
3. En tercer lugar, se aplican las declaraciones definidas de manera inline, esto quiere decir, aquellas que se escriban directamente dentro del atributo style en cada elemento, éstas son individuales y aplican solamente para ese elemento en particular (en el ejemplo, último párrafo con texto de color naranja, éste puede aplicar el estilo para los párrafos p que se define dentro de estilos.css, pero como tiene asignada una clase "clase-css-tag", éste tiene mayor prioridad, sin embargo también tiene asignado el estilo inline con el atributo style, por lo que éste le gana a las anteriores y prevalece el color naranja).
4. Existe la palabra reservada !important, la cual si se aplica a cualquier declaración ésta sube en precedencia y tiene prioridad sobre el resto de las reglas, pero es considerada una mala práctica porque provoca que se rompa el esquema de prioridad de cascada, por lo que no se recomienda su uso.

Como pudiste ver, a partir de esta especificidad y prioridad en la aplicación de estilos es de donde viene el nombre CSS, ya que depende de esa prioridad para aplicar los estilos de manera figurativa a una cascada.

Como parte de las reglas existen los selectores de clase, este tipo de selectores sólo aplican el estilo en aquellos elementos que se definen dentro del HTML con la palabra class, el cual es un atributo de HTML y no debe confundirse con la clase que define a un objeto, ya que son dos cosas diferentes.

**Bordes**

La directiva border permite especificar el estilo, tamaño y color del borde de un elemento en particular, se define por las siguientes sub-directivas:

* **border-style:** Especifica el tipo de borde a mostrar, soporta varios valores para distintos tipos de línea de borde: dotted, dashed, solid, double, groove, ridge, inset, outset, none y hidden. Así mismo acepta 1, 2 y 4 valores distintos, dependiendo del lado del borde que se quiere especificar: top right bottom y left. Ejemplo border-style: dotted; aplica en los 4 bordes, border-style: dotted dashed; aplica dotted para el top y el bottom y dashed para los lados laterales (left y right) border-style: dotted dashed solid double; aplica para cada lado (top right bottom left).
* **border-width:** Especifica el ancho de cada uno de los 4 bordes de cada elemento, puede estar dado en un tamaño específico (px, pt, cm, em) o usando uno de los tres valores pre-definidos (thin, medium, thick). Así mismo también acepta desde 1, 2 y 4 valores. Ejemplo border-width: 1px; aplica en los 4 bordes, border-width: 1px thin; aplica 1px para el top y el bottom y thin para los lados laterales (left y right) border-width: 1px 2px 3px medium; aplica para cada lado (top right bottom left).
* **border-color:** Especifica el color con el que se mostrará el borde. Acepta los mismos valores que la directiva color, aceptando además un valor adicional transparent, el cual se aplica para transparencias y opacidades, las cuales se verán más adelante. Ejemplo border-color: #ff0000;.

Para minimizar el código se pueden especificar todos los valores del borde en una misma propiedad, en lugar de escribir cada una de las directivas, se declara una sola propiedad con todos los valores para cada una de las directivas border: 5px solid red;. El orden es border: width style color y debe ser respetado.

.borde-completo {

border: 5px solid red;

}

.bordes-laterales {

/\* El orden es top right bottom left \*/

border-style: none dotted none dotted;

border-width: 2px;

border-color: #00AA00;

}

*Ejemplo de Bordes en CSS*

**Márgenes**

Los márgenes (margin) representan espacios alrededor de los elementos, es decir, define el tamaño del espacio entre un elemento y otro, el espacio entre sus bordes correspondientes. Esta directiva acepta 4 posibles valores: **auto** con el que el navegador calcula automáticamente el margen de cada elemento (comportamiento default), **una longitud específica** (px, pt, cm, em), un **valor en porcentaje** (%) que es la proporción con relación al elemento superior que lo contiene y finalmente **inherit**, lo cual indica que el elemento hereda el margen definido en el elemento superior que lo contiene.

Se pueden usar las directivas específicas para un margen de cada lado, de la siguiente manera:

* **margin-top:** Especifica el espacio de margen superior. Ejemplo margin-top: 2px;.
* **margin-right:** Especifica el espacio de margen a la derecha del elemento. Ejemplo margin-rigth: 1.2em;.
* **margin-bottom:** Especifica el espacio de margen inferior. Ejemplo margin-bottom: 10%;.
* **margin-left:** Especifica el espacio de margen a la izquierda del elemento. Ejemplo margin-left: inherit;

Igual que con los bordes, se puede minimizar código especificando la directiva margin y asignando los valores de la propiedad en el orden en el que se especifican, es decir margin: 2px 1.2em 10% inherit;, respetando el orden top right bottom left.

.margen-completo {

/\* El orden es top right bottom left \*/

margin: 2px 2.2em 2% inherit;

}

.margen-superior-inferior {

margin-top: 2px;

margin-bottom: 2.2em;

}

*Ejemplo de Márgenes en CSS*

**Padding**

La directiva padding se usa para definir el tamaño del espacio alrededor del contenido del elemento, es decir, dentro de los límites de los bordes, desde el borde hasta el contenido en sí del elemento. De igual manera, acepta los mismos valores que la directiva margin exceptuando el valor auto, el cual no está soportado, pero sí acepta longitudes específicas (px, pt, cm, em), un valor en porcentaje (%) que es la proporción en relación con el elemento superior que lo contiene y finalmente inherit, lo cual indica que el elemento hereda el padding definido en el elemento superior que lo contiene.

Igualmente, soporta directivas para especificar cada lado padding-top, padding-right, padding-bottom y padding-left, los cuales se usan de la misma manera que los márgenes, así mismo, también soporta el minimizado de código con la directiva padding: top right bottom left.

.padding-completo {

/\* El orden es top right bottom left \*/

padding: 2px 2pt 1% inherit;

}

.padding-por-lado {

padding-top: 2px;

padding-right: 10px;

padding-bottom: 5px;

padding-left: 1px;

}

*Ejemplo de Padding en CSS*

**Colores**

Para especificar colores de texto en un elemento HTML usando CSS, se usa la declaración color, la cual acepta cualquiera de los 140 nombres de colores predefinidos que los navegadores web tienen integrados (ejemplo color: red;).

Así mismo, puede aceptar valores en RGB:

* RGB: color: rgb(255,0,0);
* Hexadecimal: color: #FF0000;
* HSL: color: hsl(0, 100%, 50%);
* HWB: color: hwb(0, 0%, 0%);

**Fondos**

Para especificar fondos se usa la declaración background , la cual se divide en cuatro directivas:

**1. background-color:** Para asignar el color de fondo de un elemento, acepta los mismos valores que acepta la directiva color. Ejemplo `background-color: #FF0000;.

**2. background-image:** Especifica una imagen para ser usada como fondo de un elemento en particular. Por default, si la imagen es más chica que el elemento que la contiene, ésta se repite de tal manera que cubra completamente el elemento. Acepta sólo texto con valores de tipo URL. Ejemplo background-image: url("path/hacia/una/imagen.jpg”);

**3. background-repeat:** Como te comenté, el comportamiento que el navegador tiene es repetir la imagen por default hasta llenar el elemento que la contiene, sin embargo, con esta directiva se puede cambiar ese comportamiento, de tal manera que se pueda repetir sólo horizontalmente (ejemplo background-repeat: repeat-x;), verticalmente (ejemplo background-repeat: repeat-y;) o mostrar la imágen sólo una vez (ejemplo background-repeat: no-repeat;).

**4. background-position:** Esta directiva permite establecer la posición de inicio de las imágenes en relación al elemento que las contiene, por default el explorador las ubica en la parte superior izquierda del elemento y a partir de ahí empieza a dibujar la imagen, pero se puede cambiar el comportamiento para que la imagen se ubique en la parte superior derecha (ejemplo background-position: right top;), esta directiva acepta 4 posibles valores indicando la posición relativa: left, top, right, bottom.

Para minimizar el código se pueden especificar todos los valores del fondo en una misma propiedad, en lugar de escribir cada una de las directivas, se declara una sola propiedad con todos los valores para cada una de las directivas:

background: #FF0000 url("path/hacia/una/imagen.jpg") no-repeat right top;.

Para que sea correctamente interpretado se tiene que respetar el orden en el que fueron explicadas (background: color image repeat position).

**Texto y Fuentes**

La mayoría de las páginas y aplicaciones web, usan fuentes de texto específicas que van con una imagen de la marca o de la empresa para la que se está desarrollando la aplicación.

Para poder configurar el texto y que el explorador cargue las fuentes específicas para esta página y las aplique, se usan varias directivas que permiten especificar la familia de la fuente, su tipo, su formato, tamaño y estilo de un texto en específico, las cuales son las siguientes:

* **font-family:** Establece la familia de la fuente a usar para el texto, puede contener varias fuentes en orden de prioridad por si el navegador no puede cargar alguna o no la soporta intentar la siguiente. Ejemplo font-family: Arial, Helvetica, sans-serif;.
* **font-style:** Especifica si se aplica un estilo itálico al texto. Tiene 3 posibles valores: normal, italic y oblique, aunque este último es menos soportado en los navegadores. Ejemplo font-style: italic;.
* **font-weight:** Especifica si se aplica un estilo “negrita” al texto. Tiene 2 posibles valores: normal y bold. Ejemplo font-weight: bold;.
* **font-size:** Especifica el tamaño del texto. Acepta valores absolutos (como son tamaños estáticos dados en pixeles) o valores relativos (que son dados en proporción al elemento que los contiene). Ejemplo font-size: 16px; o font-size: 2.5em;.

Para acortar código, es posible usar la directiva font directamente y asignar esos valores, no requiriendo estar en algún orden en específico, pero sí, al menos que se defina el tamaño y la familia (ejemplo font: 12px Arial, sans-serif;).

**¿Qué te pareció el contenido de CSS? Si deseas puedes consultar y descargar una infografía sobre este contenido, [Presiona aquí](https://github.com/U-Camp/BOOT-M1-SEM2/blob/main/infografia/M1_S2_Infografia_css.pdf)**

**CSS: Responsive Design**

Trabajar en la web nos obliga a considerar siempre a nuestros usuarios y a los dispositivos en los cuales consumirán nuestras aplicaciones.

Cabe destacar que, cuando empezamos a diseñar para diferentes dispositivos el área de trabajo es inmensa. Sólo consideraremos lo que tenemos involucrado:

* Móviles,
* Tabletas,
* Computadoras,
* Monitores largos,
* Televisiones,
* Pantallas de cine, entre otras.

Para ello, dentro de CSS, podemos encontrar estructuras que permitan flexibilidad. De acuerdo al comportamiento y el dispositivo en el cual el usuario está situado, se contendrán diferentes estilos.

A esta práctica se le conoce como *Responsive Design*, o también, Diseño Responsivo.

El elemento más importante para poder ejecutar esto son los media queries.

**Media Queries**

Un media query se refiere a una regla que incluye un bloque de propiedades CSS si se cumple una condición específica.

Está conformado por:

* @media. Indica que aplicaremos un media query.
* screen. Indica que estaremos trabajando con pantallas. Existen otras opciones que puedes observar en la [documentación](https://www.w3.org/TR/CSS2/media.html) como print o speech. Sin embargo, screen es indicada para este ejemplo.
* &. Indica que vamos a agregar una condición adicional.
* max-width: 800px. Indica que vamos a establecer un límite máximo de 800px como condición. Si la pantalla tiene 800px o menos, se ejecutará ésta. A esto se le conoce como punto de quiebre o breakpoint. También existe el contrario que es min-width, el cual es la medida mínima.

Ejemplo:

@media screen & (max-width: 800px){

body{

background-color: blue;

}

}

Observa en este siguiente ejemplo el cambio con una medida mínima.

@media screen & (min-width: 800px){

body{

background-color: blue;

}

}

Dependiendo del dispositivo, se recomienda trabajar en estos rangos:

* 320px — 480px: Dispositivos móviles.
* 481px — 768px: iPads, Tabletas.
* 769px — 1024px: Pequeñas pantallas, computadoras portátiles.
* 1025px — 1200px: Computadoras de escritorio, pantallas amplias.
* 1201px o más: Pantallas extra grandes, televisiones.

Consideraciones importantes:

* CSS trabaja en un formato en cascada. Es decir, tan pronto exista la coincidencia con un media query con respecto a las condiciones, se ejecutará esos estilos específicos.
* Puedes probar tus media queries tomando la ventana del navegador y haciendola más pequeña y más grande.

**CSS: Frameworks**

CSS cuenta con herramientas adicionales, propuestas por organizaciones o proyectos *Open Source*, que permiten desarrollar con mayor velocidad y bajo una estructura específica.

A estas herramientas se les conocen como *frameworks*.

Uno de los más populares y sólidos es **Bootstrap**.

**Bootstrap**

Bootstrap, es un framework front-end basado en CSS y JavaScript, el cual se usa para desarrollar aplicaciones web responsivas, esto es desarrollar páginas y aplicaciones web que se vean bien tanto en escritorio, como en dispositivos móviles (tabletas y celulares). Fue creado por Twitter en 2010 y desde entonces ha gozado de amplia popularidad en el mundo del desarrollo de software, ya que es un framework que es muy fácil de aprender e implementar.

Provee una serie de herramientas que permiten estilizar los elementos HTML, así como plantillas de componentes que ya se encuentran estilizados, tales como fuentes, tipografías, iconos, formularios, botones, tablas, menús de navegación, modales, carruseles de imágenes, así como un amplio abanico de herramientas de JavaScript, las cuales se pueden reusar, ampliando la funcionalidad de cualquier sitio web.

Bootstrap 5 es la versión más reciente y soporta la mayor parte de los navegadores web, excepto por versiones anteriores de IE9, incluyéndose.

Como ya se comentó, Bootstrap es muy fácil de usar, por lo que cualquier persona con conocimiento básico en HTML y CSS puede implementarla; con su esquema de filas y columnas, permite una responsividad que se ajusta fácil y automáticamente a cualquier tipo de pantalla.

Para poder implementar Bootstrap, se puede hacer de varias maneras:

**1. Usando el CDN (Content Delivery Network ) de Bootstrap:** provee la última versión del CSS y del JS de Bootstrap y se incluyen directamente en la página HTML enlazando los archivos necesarios directamente en la etiqueta de la página.

**2. Descargando los archivos directamente de la página e incluyéndolos directamente en el paquete de la aplicación**\*, esto es haciendo referencias igual en el HEAD, pero con enlaces relativos a su ubicación dentro de la página web.

**3. Instalándolo con “npm”, usando el comando install: $ npm install bootstrap.**\* Bajo este método se instala en la carpeta node\_modules del proyecto y se puede referenciar al CSS y al JS directamente importándolos en los archivos en los que se vayan a usar.

Ejemplo:

<!DOCTYPE html>

<html lang="en">

<head>

<!-- Última versión del CSS minificado -->

<link

rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<!-- Última versión del JS minificado -->

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js">

</script>

</head>

</html>

*Ejemplo de Referencias a Bootstrap*

**Filosofía mobile-first en Bootstrap**

Se establece que cualquier diseño debe funcionar y ser intuitivo primero en dispositivos móviles y luego en escritorio.

Para asegurar este comportamiento, se le indica al navegador web el tipo de diseño de la página y el zoom del mismo, por lo que se asigna un metadato en el HEAD de la página para que el explorador actúe de manera adecuada:

Ejemplo:

<meta name="viewport" content="width=device-width, initial-scale=1">

**Elementos**

La base principal de Bootstrap son los siguientes elementos:

**o Contenedores:** son elementos que envuelven otros elementos, normalmente son elementos <div /> a los cuales se le especifica la clase CSS container o container-fluid. Normalmente a este tipo de contenedores se les pueden asignar otras clases de tal manera que estilen el fondo y el texto de los elementos contenidos, como por ejemplo bg-dark o text-white.

**o Grids:** El sistema de filas y columnas de Bootstrap es su principal cualidad, ya que permite el acomodo de los elementos dentro de una rejilla virtual que se divide en 12 columnas por fila, pudiendo usar tantas filas como sean necesarias para mostrar la página.

Se usa aplicando las clases row y col, la clase col acepta modificadores dependiendo de las columnas y tamaño del dispositivo, por ejemplo col-md-4 indica que es una columna de tamaño mediano y que abarca 4 posiciones de las 12 disponibles, pudiendo poner más elementos en las siguientes 8 posiciones que quedan, cambiando el número final de la clase, es decir que por cada fila row se pueden poner hasta 12 elementos col-md-1, uno por columna o hasta 6 elementos col-md-2 y así sucesivamente hasta ocupar las 12 posiciones. Normalmente se usan elementos <div /> para manejar este sistema de filas y columnas.

Ejemplo:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Ejemplo Bootstrap</title>

<!-- Última versión del CSS minificado -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<!-- Última versión del JS minificado -->

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

</head>

<body>

<div class="container p-3 my-3 bg-dark text-white">

<h1>Contenedor</h1>

<p>Fondo negro y texto blanco.</p>

</div>

<div class="container">

<div class="row">

<div class="col-md-4">

<h3>Header de Columna 1</h3>

<p>Párrafo en columna 1</p>

</div>

<div class="col-md-4">

<h3>Header de Columna 2</h3>

<p>Párrafo en columna 2</p>

</div>

<div class="col-md-4">

<h3>Header en Columna 3</h3>

<p>Párrafo en columna 3</p>

</div>

</div>

</div>

</body>

</html>

*Ejemplo de Contenedores y Grid usando Bootstrap*

En el ejemplo, si se renderiza esta página web y luego se le cambia el tamaño, se podrá ver cómo cambian de posición los elementos de tal manera que se adaptan al tamaño del dispositivo en el que se observan. La clase p-3 indica que se dará un padding de 3 pixeles, así mismo la clase my-3 añade un margen de 3 pixeles.

Básicamente estos son los elementos principales de Bootstrap, se irán conociendo y definiendo más componentes conforme se vaya avanzando en las clases.

**¿Cómo vas?, CSS es super interesante ya que le da más forma a nuestros desarrollos.**

**Recuerda que si deseas practicar y ver cómo se ve en línea puedes hacerlo en VSC o en <https://codepen.io/pen/> solo copiando y pegando el código, así puedes hacer los cambios y pruebas que necesites. ¡Mientras más practiques, más rápido aprenderás!**

**Seguimos... ☺️**

**Componentes: Alertas**

El componente alert de Bootstrap permite mostrar mensajes al usuario de tal manera que llamen su atención y opcionalmente, que el mismo usuario los pueda quitar de su vista, a manera de notificaciones. Para lograr esto, se usa la clase alert alert-<tipo>, donde tipo es el tipo de alert que se va a mostrar, permitiendo 8 distintos tipos (alert-primary, alert-secondary, alert-success, alert-danger, alert-warning, alert-info, alert-light y alert-dark), con lo que se puede estilar cualquier tipo de mensaje.

Ejemplo:

<div class="mover-elemento borde alert alert-primary">

Este elemento se movió desde 0,0

<button type="button" class="close" data-dismiss="alert" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

*Ejemplo de alert con bootstrap*

**Badges**

Los badges son elementos mayormente visuales que son usados para agregar información adicional a cualquier tipo de contenido. Se usa con la clase badge seguida de una clase contextual badge-<tipo\_badge>, que como ya se ha visto en otros componentes, acepta los mismos 8 posibles valores (badge-primary, badge-secondary, badge-success, badge-danger, badge-warning, badge-info, badge-light y badge-dark).

**Transformaciones**

CSS es capaz de realizar transformaciones matemáticas a elementos HTML, esto quiere decir que, aplicando transformaciones, CSS puede mover elementos de una posición X a una posición Y o escalarlos en determinadas proporciones o rotarlos en ciertas direcciones. Para lograr esto, CSS tiene ciertas **"reglas especiales"** que funcionan como métodos de estilo, las cuales permiten realizar este tipo de acciones sobre los elementos y se definen usando la regla transform seguida del método que ejecutará la transformación. De estos métodos los más usados son los siguientes:

* **translate:** Este método permite mover un elemento a partir de su posición original, de acuerdo a los parámetros dados para el eje X y el eje Y. Ejemplo transform: translate(50px, 100px); se mueve el elemento 50 píxeles a la derecha y 100 píxeles hacia abajo. Este método se acompaña con 2 métodos extra translateX y translateY los cuales permiten mover un elemento en los ejes X y Y respectivamente de manera individual.
* **rotate:** Permite rotar el elemento en sentido horario o contra-horario, dependiendo de los grados que se le pasen como parámetro. Ejemplo transform: rotate(45deg); rota el elemento 45 grados en sentido de las manecillas del reloj (izquierda -> derecha) o transform: rotate(-45deg); rota el elemento en 45 grados en sentido contrario a las manecillas del reloj (derecha -> izquierda). Hay métodos extra de rotación, que permiten manejar los 3 ejes, de tal manera que rote el elemento en 3D:
* **rotateX:** Rota el elemento a determinada cantidad de grados sobre el eje X. Ejemplo transform: rotateX(45deg);.
* **rotateY:** Rota el elemento a determinada cantidad de grados sobre el eje Y. Ejemplo transform: rotateY(45deg);.
* **rotateZ:** Rota el elemento a determinada cantidad de grados sobre el eje Z. Ejemplo transform: rotateZ(45deg);.
* **scale:** Incrementa o decrementa el tamaño del elemento de acuerdo con la proporción dada en los parámetros para el ancho y el alto. Ejemplo transform: scale(2, 3); incrementa el tamaño del elemento dos veces su ancho y 3 veces su alto; transform: scale(0.5, 0.5); decrementa el tamaño del elemento la mitad de su ancho y la mitad de su alto. Este método se acompaña de 2 métodos extra scaleX y scaleY los cuales permiten incrementar o decrementar el ancho o el alto respectivamente de manera individual.
* **skew:** Inclina el elemento sobre el eje X o Y cierta cantidad de grados dependiendo de los parámetros dados. Ejemplo transform: skew(20deg, 10deg); inclina el elemento 20 grados en el eje X y 10 grados en el eje Y. Este método se acompaña de 2 métodos extra skewX y skewY los cuales permiten inclinar sobre el eje X o el eje Y respectivamente de manera individual.
* **matrix:** Este método combina todos los anteriores en uno solo, toma 6 parámetros que corresponden a cada una de los métodos individuales de transformación en el orden matrix(scaleX, skewY, skewX, scaleY, translateX, translateY). Ejemplo transform: matrix(1, -0.3, 0, 1, 0, 0); el ancho del elemento se queda igual, ya que la escala es 1; pero lo inclina negativamente 0.3 grados sobre el eje Y, sin inclinarlo sobre el eje X y manteniendo la misma altura ya que la escala es 1, sin moverlo sobre el eje X ni sobre el eje Y. Se usa más para minimizar código.

Ejemplo: /\* Mueve el elemento 50 píxeles a la derecha y 100 píxeles hacia abajo \*/

.mover-elemento {

transform: translate(50px, 100px);

}

/\* Rota el elemento 45 grados \*/

.rotar-elemento-positivo {

transform: rotate(45deg);

}

/\* Rota el elemento -45 grados \*/

.rotar-elemento-negativo {

transform: rotate(-45deg);

}

/\* Incrementa el elemento dos veces su ancho y 3 veces su alto \*/

.incrementar-elemento {

transform: scale(2, 3);

}

/\* Decrementa el elemento la mitad de su ancho y alto \*/

.disminuir-elemento {

transform: scale(0.5, 0.5);

}

/\* Inclina el elemento 20 grados en el eje X y 10 grados en el eje Y \*/

.inclinar-elemento {

transform: skew(20deg, 10deg);

}

/\* Transforma el elemento con el shortcut `matrix` \*/

.transformar-elemento {

/\* matrix(scaleX, skewY, skewX, scaleY, translateX, translateY) \*/

transform: matrix(1, -0.3, 0, 1, 0, 0);

}

/\* Rota el elemento en los grados dados para cada eje \*/

.rotar-elemento-3D {

transform: rotateX(150deg);

transform: rotateY(130deg);

transform: rotateZ(90deg);

}

**Transiciones**

Además de las transformaciones matemáticas que habilita CSS, éste también ofrece otros métodos matemáticos para realizar cambios en los valores de las propiedades de un elemento de manera gradual, de tal manera que se pueda observar visualmente el cambio. Para poder crear una transición, se requieren dos cosas: La propiedad CSS sobre la cual aplicar el cambio y la duración del efecto (ejemplo: transition: width 2s;, se le va a aplicar el ancho especificado al elemento en una transición que dure dos segundos), si no se especifica una duración, el valor default es cero, por lo que no ocurre ninguna transición, así mismo la transición es aplicada hasta el momento en el que el cambio de tamaño ocurre.

Las transiciones pueden tener distintos efectos, de tal manera que visualmente se note una distinta transición en cada elemento, para lo cual se ocupa la propiedad transition-timing-function, la cual permite 6 posibles valores:

**1. ease:** Es el valor predeterminado, específica que la transición empiece lenta, luego rápida y finalice lenta.

**2. linear:** Especifica que la transición tenga una misma velocidad de inicio a fin.

**3. ease-in:** Especifica una transición con un inicio lento.

**4. ease-out:** Especifica una transición con un fin lento.

**5. ease-in-out:** Especifica una transición con un inicio y fin lentos.

**6. cubic-beizer:** Permite especificar la curva de velocidad de la transición por medio de una función cubic-beizer , la cual recibe 4 parámetros (x1, y1, x2, y2) que corresponden a los dos puntos coordenados de la curva (inicial y final entre 0 y 1) con los cuales calcular la velocidad de la transición. Ejemplo: transition-timing-function: cubic-beizer(0.75, 0.56, 0.19, 0.67);.

También es posible especificar un retraso en la aplicación de la transición, usando la propiedad transition-delay la cual recibe un valor en segundos que especifica el tiempo de retraso antes de que comience la transición.

Se pueden especificar dos o más propiedades y aplicarles una transición, incluso se puede hasta aplicar transiciones a transformaciones (ejemplo: transition: width 2s, height 4s, transform 2s;).

También acepta los valores del efecto directamente, de tal manera que todo se pueda especificar en una misma propiedad de ésta manera transition: <transición> <efecto> <retraso>; (ejemplo transition: width 2s linear 1s;).

Ejemplo:

/\* Transición sobre todos los divs del documento HTML \*/

div {

width: 500px;

height: 100px;

background: gray;

transition: width 2s;

}

/\* Cambia el tamaño del div cuando se pasa el ratón sobre éste \*/

div:hover {

width: 800px;

}

.transicion-linear {

transition-timing-function: linear;

}

.transicion-ease {

transition-timing-function: ease;

}

.transicion-ease-in {

transition-timing-function: ease-in;

}

.transicion-ease-out {

transition-timing-function: ease-out;

}

transicion-ease-in-out {

transition-timing-function: ease-in-out;

}

/\* Transición asignada con propiedades separadas \*/

.transicion-separada {

transition-property: width;

transition-duration: 2s;

transition-timing-function: linear;

transition-delay: 1s;

}

/\* Transición asignada en una misma propiedad \*/

.transicion-completa {

transition: width 2s linear 1s;

}

***Ejemplo transiciones en CSS***

**Hola, espero que lo que te mostré anteriormente te haya sido fácil de comprender, de lo contrario recuerda que puedes leerlo cada vez que lo necesites. También es importante que puedas tomar un descanso, recuerda que nuestro cuerpo y nuestra mente necesitan un respiro, por lo que te invito a ver el**[**siguiente video**](https://www.youtube.com/watch?v=tA2kT8eSjtg)**.**

**¿Qué te pareción el video?, espero te sientas mucho mejor, y ahora si a descansar.**

**¡Nos vemos pronto!**