

Desarrollo Web Para Principiantes

Introducción

El propósito de este documento es orientar mediante ejemplos y consejos que me han servido para adentrarme poco a poco en el mundo de las páginas web. Se iniciará con lo necesario para poder programar en HTML, pasando por CSS y JavaScript. Además de otras herramientas como NodeJS, Nginx, etc.

HTML es un lenguaje de programación basado en etiquetas usadas para declarar lo que será mostrado al usuario en el navegador. HTML no permite definir cómo será mostrado, es meramente el esqueleto de las páginas web. Para poder estilizar y dar funcionalidad a los elementos, serán necesarios otros lenguajes de programación como CSS para estilizar y JavaScript para agregar funcionalidad a la página.

Si se desea estudiar más en profundidad lo visto durante la guía, será posible descargar desde el Github todos los archivos y pdf's recomendados para iniciar.

Índice

Capítulo 1: Preparación del entorno de desarrollo.....	004
Capítulo 2: HTML básico	009
Capítulo 3: CSS básico	015
Capítulo 4: JavaScript básico	022
Capítulo 5: Nivel intermedio	031
Capítulo 6	000
Capítulo 7	000

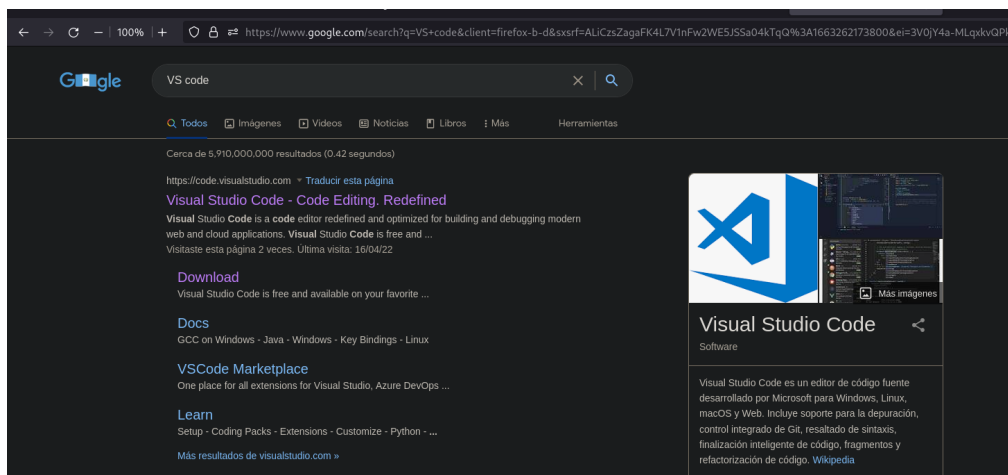
Capítulo 1: Preparación del entorno de desarrollo

Elegir un entorno de desarrollo

Para este curso será necesaria una herramienta para poder programar de forma comoda y eficiente. Se puede usar algo sencillo y ligero como el bloc de notas, pero durante la guía se usará VS Code, debido a la cantidad de extensiones y herramientas que ofrece a la hora de programar.

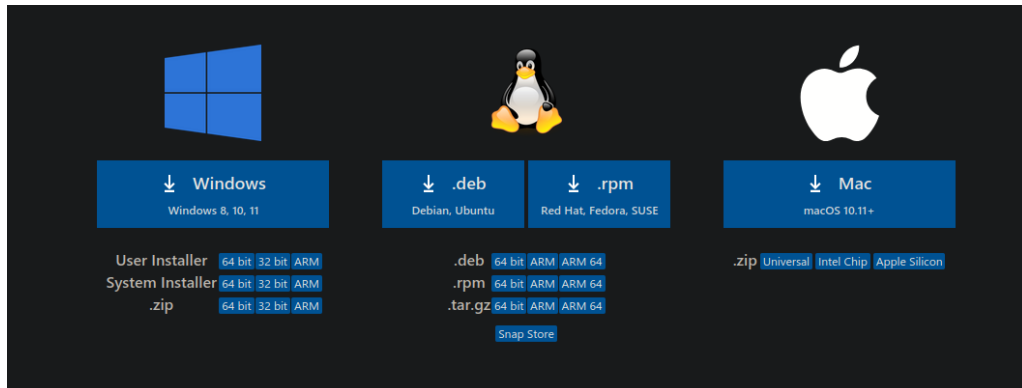
Paso 1:

- Ve al navegador de tu elección y busca VS Code o Visual Studio Code
- Ingresa a la página web



Paso 2:

- Según tu sistema operativo (Windows, Mac o Linux), selecciona el archivo para instalar, descarga e instalalo.

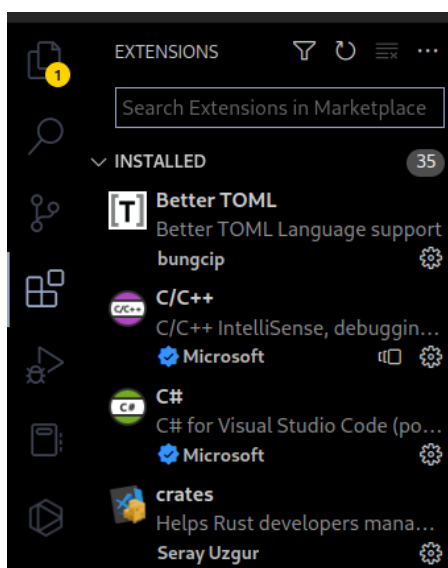


Paso 3:

- Ya instalado vamos a crear una carpeta donde iremos guardando los archivos durante toda la guía. No importa la ubicación de dicha carpeta.

Paso 4:

- Abrir VS Code
- Ir a File (Archivo)
- Open Folder (Abrir carpeta) y se selecciona la carpeta creada anteriormente



Etiquetas básicas en HTML5

DOCTYPE: Esta etiqueta no es completamente necesaria, pero como buena práctica coloca al inicio del documento html, para así hacerle saber al navegador que tipo de archivo estará leyendo.

TITLE: Permite modificar el texto del título de la pestaña del navegador.

HEAD: Etiqueta que contiene información que no será mostrada en pantalla, pero que será necesaria para el funcionamiento de la página web. Desde importación de elementos de terceros, hasta importar archivos css y javascript.

HEADER: Etiqueta que permite separar los menús principales del resto del cuerpo de la página web.

BODY: Esta etiqueta engloba a la mayoría de elementos en la página.

FOOTER: Etiqueta utilizada más como buena práctica, es usada para colocar información de contacto, direcciones físicas, etc.

P: Etiqueta para añadir párrafos completos al html de la página. (Es posible agregar párrafos con otras etiquetas, pero como buena práctica se recomienda usar la etiqueta **p**).

B: Texto en formato **bold** o **negrita**.

H1 hasta H6: Son quizás algunas de las etiquetas más utilizadas, ya que son las mejores para títulos, subtítulos y textos puntuales que no se repetirán.

BR: Etiqueta que permite agregar líneas vacías al html.

I: Etiqueta de texto en formato *cursiva*.

SUMMARY: Etiqueta utilizada en el texto principal de la etiqueta **Details**.

DETAILS: Etiqueta que permite “esconder” texto en un menú desplegable (Ver ejemplos más adelante)

STRIKE / DEL: Etiqueta que permite “tachar” un texto. (ejemplo: ~~texto tachado~~)

SMALL: Texto en un menor tamaño. (ejemplo: texto más pequeño)

SECTION: Etiqueta que permite seccionar los párrafos en varias partes. (No afecta visualmente, pero es una buena práctica).

SUP y SUB: Permite colocar texto arriba y debajo de otros elementos. (Ejemplo: $hola^{exponente} y base_{base}$)

ASIDE: Etiqueta para indicarle al navegador que es un texto secundario (No afecta visualmente, más es una buena práctica).

ARTICLE: Etiqueta mayormente utilizada como una buena práctica ya que funciona similar a la etiqueta **P**, ya que permite añadir grandes textos. (Es usada para hacerle saber que el texto es solamente una pequeña parte del contenido de los párrafos).

NAV: Etiqueta utilizada para englobar grandes bloques de links (No afecta visualmente, es usada como buena práctica).

UL: Lista sin un orden específico. Un buen ejemplo sería una lista con viñetas.

OL: Lista ordenada, lista que tiene un orden específico, mediante distintos elementos.

LI: Etiqueta para agregar elementos a la lista.

A: Etiqueta utilizada para crear elementos con links.

IMG: Etiqueta para poder crear elementos de tipo imagen. Será posible hacer ciertas modificaciones mediante el atributo width (se recomienda usar porcentajes para poder mantener las proporciones). Y el atributo alt para indicar que debería ir en caso de que la imagen no pueda ser desplegada.

VIDEO: Etiqueta para añadir elementos multimedia de tipo vídeo. Será posible modificar elementos mediante los atributos controls, loop y autoplay.

AUDIO: Etiqueta multimedia de tipo audio. Permite modificar elementos mediante los atributos autoplay, controls y loop.

TABLE: Etiqueta que permite crear un elemento de tipo tabla (ejemplo: tablas de Excel).

TR: Etiqueta para generar filas en las tablas. (Table row)

TH: Etiqueta para generar los encabezados de las columnas en las tablas. (Table header)

TD: Etiqueta para añadir elementos a las tablas. (Table data).

CAPTION: Etiqueta usada para añadir títulos a las tablas.

DIV: Etiqueta usada para englobar secciones de código (No afecta visualmente, es usada como una buena práctica, para mantener lo más organizado el código).

LABEL: Etiqueta usada como buena práctica al lado de elementos de formularios.

FORM: Etiqueta usada para englobar y crear elementos de input en la página web. Existen varios atributos que le indican al navegador que tipo de input se realizará.

- **Type text:** Cuadros de inputs vacíos para textos simples
- **Type password:** Le indica al navegador que se ingresará una contraseña, lo que sustituye los inputs por asteriscos.

- **Type number:** Indica que se deberán ingresar números solamente.
- **Type email:** Indica que deberá ingresar un texto en formato email (hola@algo.algo).
- **Type Color:** Crea un recuadro que permite elegir un color.
- **Type range:** Crea una barra para deslizar en vez de ingresar el número directamente.
- **Type date:** Crea elementos que permiten seleccionar una fecha.
- **Type time:** Crea un recuadro que permite seleccionar una hora
- **Type textarea:** Permite generar un recuadro para ingresar textos grandes.
 - Además, se puede crear el recuadro con un tamaño ya establecido con los atributos row y cols

- **Type radio:** Permite crear un menú del tipo rellenar un círculo. Mediante los atributos name para agrupar y value para definir la opción dentro del menú.
- **Type checkbox:** Permite producir un menú de opciones que permite seleccionar más de un elemento. Las opciones se generarán con los atributos name y value.
- **Type button:** Elemento que genera un botón para crear un botón e interactuar con el formulario.
- **Type submit:** Elemento que genera un botón para enviar los resultados del formulario.

SELECT: Etiqueta que creará un menú de tipo acordeon.

OPTION: Etiqueta para añadir elementos al menú en la etiqueta **SELECT**.

<!-- -->: Etiqueta que permite “comentar” una sección de código.

PROGRESS: Permite mostrar una barra de “progreso” o carga.

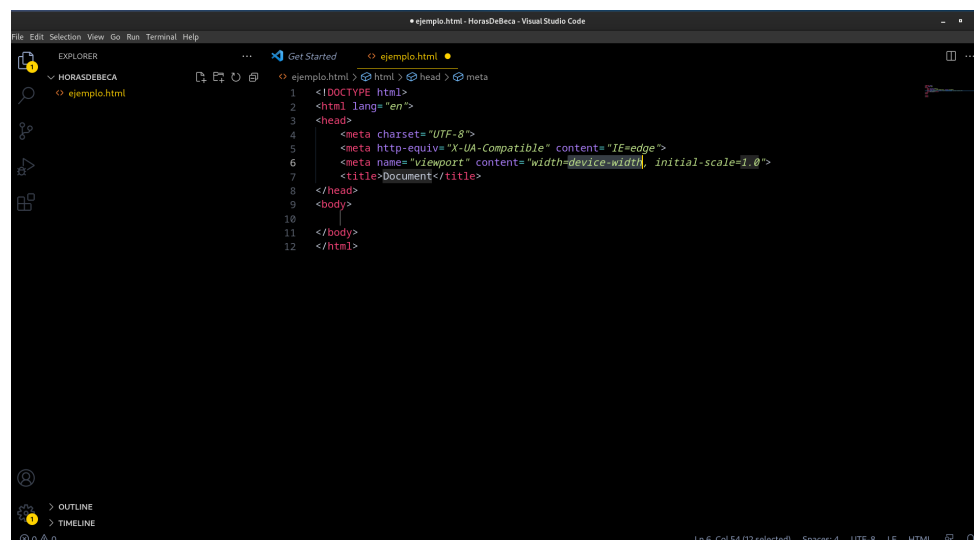
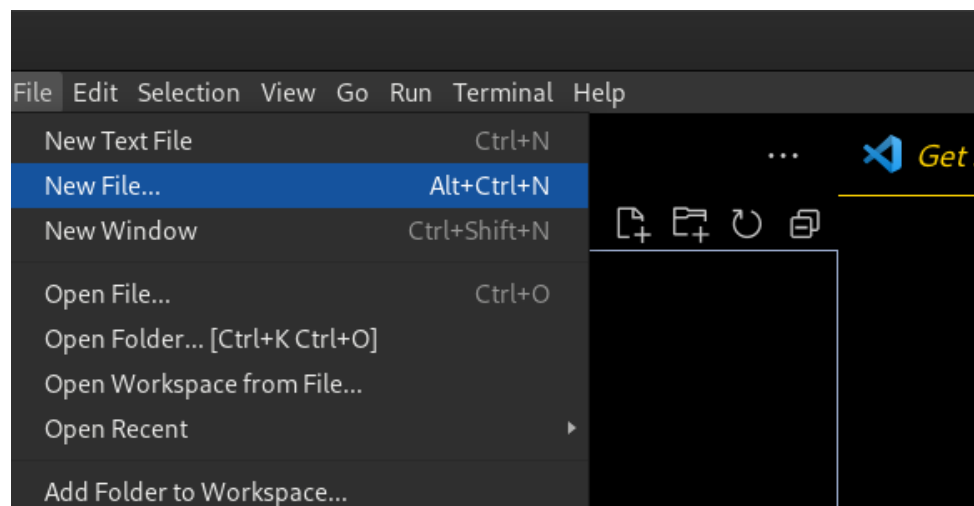
Capítulo 2: HTML básico

A lo largo de este capítulo iremos creando diversos mini-proyectos escritos solamente en html. Comenzando con una pequeña historia interactiva, pasando por una página web de una tienda y por último un blog, siempre haciendo uso de varias de las etiquetas vistas en el capítulo anterior.

Inicio del proyecto

Para iniciar, crearemos el tan conocido “Hola Mundo!”. Por lo que debemos crear un archivo, los pasos a seguir son:

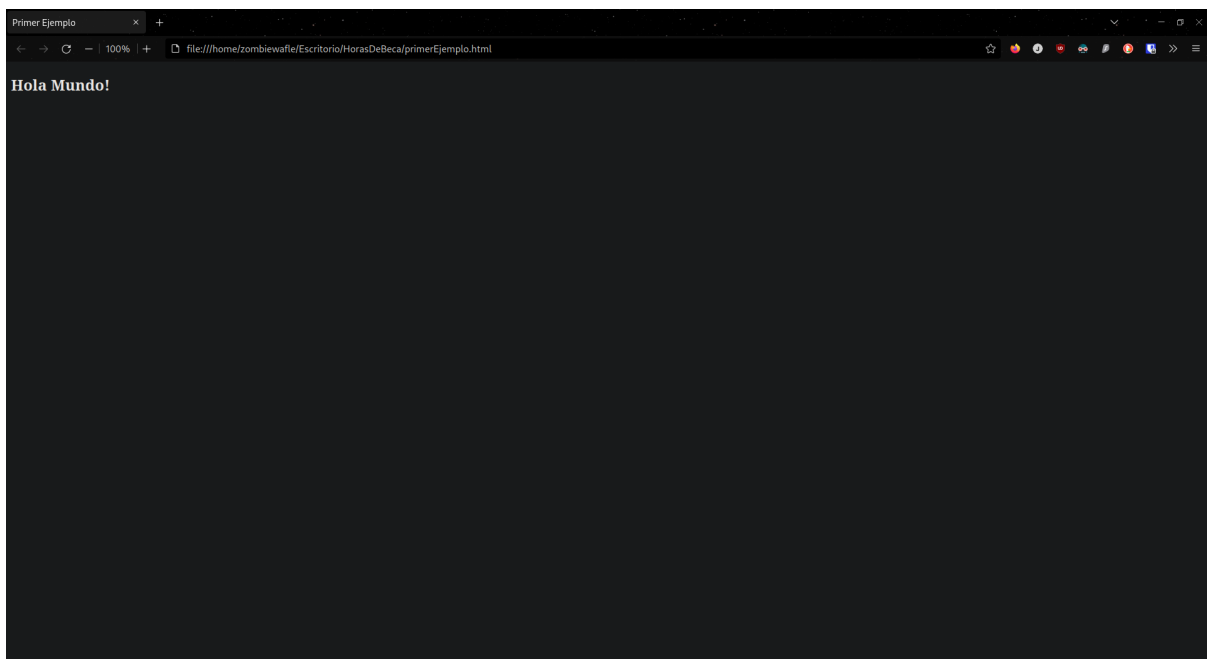
- Ir a crear un archivo desde **Archivo** (File)
- Ir a **Nuevo Archivo** (New File)
- Se ingresa el nombre del documento
- Para generar la estructura básica de un html podemos escribir html y seleccionar la opción html:5



Ya generada la estructura básica, ha llegado el momento de crear nuestra primera página web en HTML5 (solamente con HTML). Para este ejemplo solamente tendremos una etiqueta H2 con el texto “¡Hola Mundo!”.

Dentro de la etiqueta **body** manejaremos la mayor parte del tiempo en este primer proyecto. Esto solamente para seguir las buenas prácticas de la programación de páginas web. Además de que usaremos la etiqueta **title** para indicar el título en la pestaña del navegador.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Primer Ejemplo</title>
8  </head>
9  <body>
10     <h2>Hola Mundo!</h2>
11
12 </body>
13 </html>
```



Historia Interactiva

Esta vez vamos a crear una historia de fantasía (el tema queda a discreción de cada uno), en donde se tengan múltiples opciones. Dependiendo de la situación se deberán crear varios caminos y finales.

Nota:

Para este ejemplo haremos uso de las etiquetas `<p>`, `<a>` y del atributo `href` para poder redirigir a los distintos archivos en el proyecto.

Primero necesitamos generar la estructura básica de la primera sección de la página.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Historia Interactiva No.1</title>
</head>
<body>

</body>
</html>
```

Para facilitar la comprensión será necesaria una página de inicio, con instrucciones básicas de cómo moverse a lo largo de la historia.

```
start.html index.html x
index.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Historia Interactiva No.1</title>
8 </head>
9 <body>
10
11   <h2>¡Bienvenidos</h2>
12   <p>A continuación encontrarás una historia interactiva, la cual cambiará basandose en tus decisiones. Algunas te llevaran al mismo punto,
13     además de que siempre podrás regresar. Este es solamente un ejemplo basado en los videojuegos basados en texto, por lo que si deseas
14     seguir expandiendo la historia, siempre podrás acceder al código fuente e ir agregando contenido.
15   </p>
16   ¡Espero que lo disfrutes y suerte!
17 </p>
18
19 </body>
20 </html>
```

Para mantener un orden se irá construyendo la historia en distintos archivos. Empezando siempre con el archivo `index.html`. Será muy importante saber nombrar los archivos que vayamos creando, para facilitar el trabajo, además de hacer más legible nuestro código.

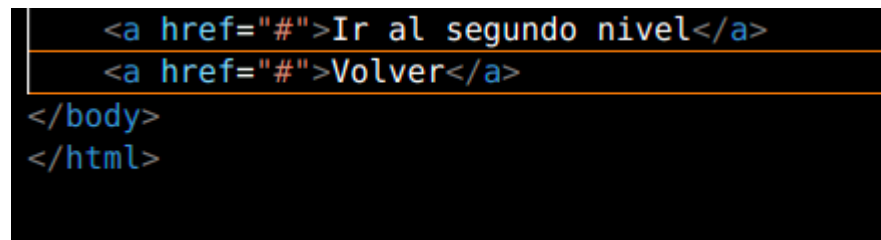
Nota: Se aconseja usar una estructura de nombrado de archivos del tipo CamelCase, ya que facilita su lectura, pero esto queda a discreción de cada uno.



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <title>Entrada</title>
8 </head>
9 <body>
10
11 <p>Has avanzado por el estrecho pasadizo, avanzas como puedes y llegas a la parte trasera de la casa. ¿Ahora que harás? ¿Intentarás entrar a
12
13 <a href="#">Romper la puerta corrediza y entrar a la casa</a>
14 <a href="#">Volver</a>
15
16 </body>
17 </html>
```

Para facilitar el trabajo más adelante, solamente iremos agregando el símbolo “#” en el atributo **href**, esto para que el navegador reconozca que es un link, pero sin funcionar realmente.

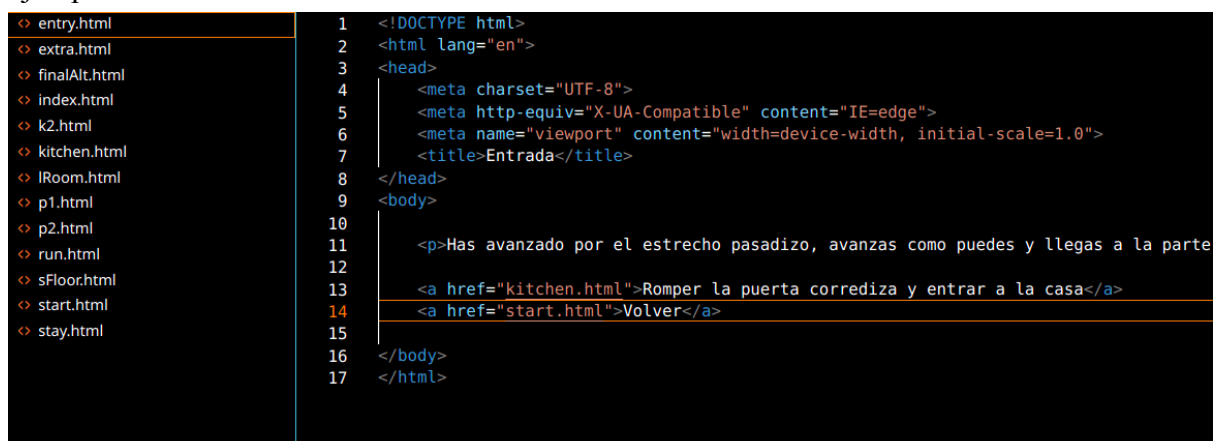
Ejemplo:



```
<a href="#">Ir al segundo nivel</a>
<a href="#">Volver</a>
</body>
</html>
```

Para poder redirigir a diversos archivos html. Será necesario agregar el nombre y dirección de dicho archivo. Repetir hasta haber completado la historia.

Ejemplo:



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <title>Entrada</title>
8 </head>
9 <body>
10
11 <p>Has avanzado por el estrecho pasadizo, avanzas como puedes y llegas a la parte
12
13 <a href="kitchen.html">Romper la puerta corrediza y entrar a la casa</a>
14 <a href="start.html">Volver</a>
15
16 </body>
17 </html>
```

Formulario en Línea

Para este ejemplo, crearemos un formulario hecho enteramente en HTML, comenzaremos creando el archivo de HTML,

En pos de mantener una estructura que sea legible para otros programadores, además de para nosotros mismos más adelante. Iremos dividiendo la página en diversos contenedores **div**, y lo identificamos con la clase *form*, para que en capítulos futuros nos sea más sencillo darle estilo.

```
<body>
|   <div class="form">|
|   </div>
```

Para crear un formulario serán necesarias unas cuentas etiquetas, comenzando con la etiqueta **form**, la cual le indica al navegador que todo lo que está encapsulado en ella, será un formulario.

```
<div class="form">
|   <form>|
|   </form>
| </div>
```

Para que el usuario sepa qué tipo de información será necesaria en el formulario, agregamos la etiqueta **label**, la cual nos permite colocar un texto al lado de lo que sea que agreguemos al html, siempre y cuando no sea un elemento que ocupe toda la línea, o que mueva al resto de etiquetas una línea. (Por ejemplo: **div** o **p**). Adicionalmente será necesario el atributo **for**, el cual sirve para asociar dicho label con un id, que en este caso será *nombre*.

```
<label for="nombre">Nombre:</label>|
```

La segunda etiqueta necesaria para la creación de un formulario será **input**, siendo la que nos permite crear un elemento interactivo de varios tipos en las páginas. Tipo que podremos ir cambiando mediante el atributo *type*, (para mayor referencia, visitar la página 5). Para finalizar si se desea se pueden agregar un par de etiquetas **br** para separar los elementos.

```
<form>
|   <label for="nombre">Nombre:</label>
|   <input type="text" id="nombre" name="nombre" required>|
| </form>
```

Repetimos el proceso con los elementos del correo electrónico, el número de teléfono y una sección extra llamada mensaje, la cual es básicamente un input vacío el cuál cuenta con una

diferencia notable respecto al resto de inputs, el tamaño es dinámico aunque esto puede ser delimitado manualmente, usando el atributo *size* y el tamaño tamaño deseado.

```
<form>
  <label for="nombre">Nombre:</label>
  <input type="text" id="nombre" name="nombre" required>
  <br>
  <br>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required>
  <br>
  <br>
  <label for="telefono">Teléfono:</label>
  <input type="tel" id="telefono" name="telefono" required>
  <br>
  <br>
  <label for="mensaje">Mensaje:</label>
  <textarea id="mensaje" name="mensaje"></textarea>
  <br>
  <br>
```

Para finalizar el formulario será necesario un input de tipo submit, ya que es lo que nos permitirá enviar todos los datos a donde sea que queramos.

```
<input type="submit" value="Enviar">
```

Notas adicionales:

En capítulos futuros se verá cómo darle funcionalidad al formulario mediante javascript.

Capítulo 3: CSS básico

Para este capítulo veremos cómo estilizar un HTML de manera básica, mediante el uso de CSS. Durante el capítulo iremos repasando paso a paso y con una descripción de los atributos más recurrentes en la mayoría de páginas web de hoy en día. Además de una breve descripción de lo que es el lenguaje CSS y sus derivados.

En el caso de CSS, los elementos que deseamos modificar serán seleccionados mediante su ID, clase, o etiqueta. Podremos acceder a ellos diversas maneras, en el caso de los ID que hayamos colocado en el html, será mediante el # seguido del ID. Las clases mediante un punto seguido del nombre de la clase (.claseEjemplo). Y por último tendremos las etiquetas nativas del lenguaje, que serán representadas por el nombre de la misma. (body, p, div, etc...)

En CSS y derivados se programa en un formato de cascada, ya que conforme vamos bajando, se podrán sobrescribir los cambios que vayamos haciendo. Por ejemplo:

```
# general.css > ...  
1  body{  
2  |    background-color: □black;  
3  |}  
4  
5  body{  
6  |    background-color: ■white;  
7  |}  
8  
9  |
```

El navegador tomará en cuenta solamente el último cambio que hayamos hecho. En este caso sería cambiar el color del fondo a blanco.

En el caso de CSS, tendremos varias opciones a la hora de implementarlo en nuestros proyectos, pero la más recomendable sería tener un archivo separado dedicado exclusivamente al manejo de los estilos de nuestros programas. Todo con la idea de mantener un orden en la estructura del proyecto, haciendo más sencillo para otros programadores leer y comprender nuestro código.

Atributos Básicos en CSS

COLOR: Permite cambiar el color de un texto.

LINE-HEIGHT: Permite cambiar la distancia entre las líneas.

LETTER-SPACING: Permite cambiar el espaciado entre caracteres.

TEXT-ALIGN: Permite posicionar el texto en distintos puntos dentro de su contenedor.

TEXT-DECORATION: Permite cambiar la decoración del texto, como subrayados, viñetas, etc.

TEXT-INDENT: Permite alterar la sangría o indentado del primer carácter de un texto.

TEXT-TRANSFORM: Permite modificar cosas como todo en mayúsculas, minúsculas, primera en mayúsculas o sin ningún tipo de alteraciones.

LIST-STYLE: Altera el estilo de las listas.

LIST-STYLE-IMAGE: Permite convertir una imagen en un elemento de la lista.

LIST-STYLE-POSITION: Permite cambiar la posición de los elementos de la lista.

LIST-STYLE-TYPE: Permite cambiar el estilo de la lista.

BORDER: Permite alterar elementos de los bordes como color, estilo, etc.

BORDER-BOTTOM: Permite cambiar las propiedades del borde inferior del contenedor.

BORDER-TOP: Permite cambiar las propiedades del borde superior del contenedor.

BORDER-LEFT: Permite cambiar las propiedades del borde izquierdo del contenedor.

BORDER-RIGHT: Permite cambiar las propiedades del borde derecho del contenedor.

BORDER-COLOR: Permite cambiar el color del borde de todos los lados del contenedor.

BORDER-LEFT-COLOR: Permite cambiar el color del borde izquierdo.

BORDER-RIGHT-COLOR: Permite cambiar el color del borde derecho.

BORDER-TOP-COLOR: Permite cambiar el color del borde superior.

BORDER-BOTTOM-COLOR: Permite cambiar el color del borde inferior.

BORDER-BOTTOM-STYLE: Permite cambiar el estilo del borde inferior.

BORDER-BOTTOM-WIDTH: Permite cambiar el grosor del borde inferior.

BORDER-TOP-STYLE: Permite cambiar el estilo del borde superior.

BORDER-TOP-WIDTH: Permite cambiar el grosor del borde superior.

BORDER-LEFT-STYLE: Permite cambiar el estilo del borde izquierdo

BORDER-LEFT-WIDTH: Permite cambiar el grosor del borde izquierdo.

BORDER-RIGHT-STYLE: Permite cambiar el estilo del borde derecho.

BORDER-RIGHT-WIDTH: Permite cambiar el grosor del borde derecho.

BORDER-STYLE: Permite cambiar el estilo del borde, en todos sus lados.

FONT: Permite cambiar el estilo de la letra, desde su grosor, estilo, tipografía, etc.

FONT-FAMILY: Permite crear o cambiar la tipografía.

FONT-SIZE: Permite especificar el tamaño de la letra.

FONT-STYLE: Permite especificar el estilo de la letra, ya sea normal, itálica, negrita, etc.

FONT-VARIANT: Permite especificar si el texto deberá ser mostrado en minúsculas o no.

FONT-WEIGHT: Permite especificar el grosor de la letra, desde normal, bold (negrita), lighter(delgada), etc.

POSITION: Establece la posición de un elemento en relación al documento o a su contenedor padre.

- **STATIC:** Es la configuración por defecto, se verá afectada por el flujo normal del documento.

- **RELATIVE:** Permite alterar la posición del elemento, afectando el flujo del documento en el proceso.
- **ABSOLUTE:** Permite alterar la posición del elemento, pero será ignorado por el resto de elementos en el flujo del documento.

MARGIN: La propiedad **margin** establece el espacio en blanco que hay alrededor del contenido de un elemento.

PADDING: La propiedad padding define el espacio en blanco que hay entre el contenido de un elemento y su borde.

DISPLAY: La propiedad display define cómo un elemento debe ser mostrado en la página. Los valores comunes incluyen block, inline, inline-block y none.

BACKGROUND-COLOR: La propiedad background-color establece el color de fondo de un elemento.

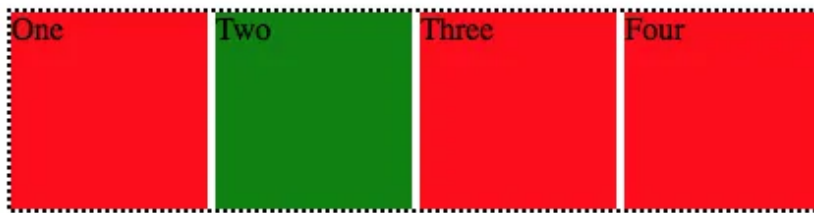
WIDTH: La propiedad width define la anchura de un elemento.

HEIGHT: La propiedad height define la altura de un elemento.

OPACITY: La propiedad opacity establece la opacidad de un elemento. Un valor de 1 es totalmente opaco y un valor de 0 es totalmente transparente.

Notas adicionales:

position: static



position: relative



position: absolute

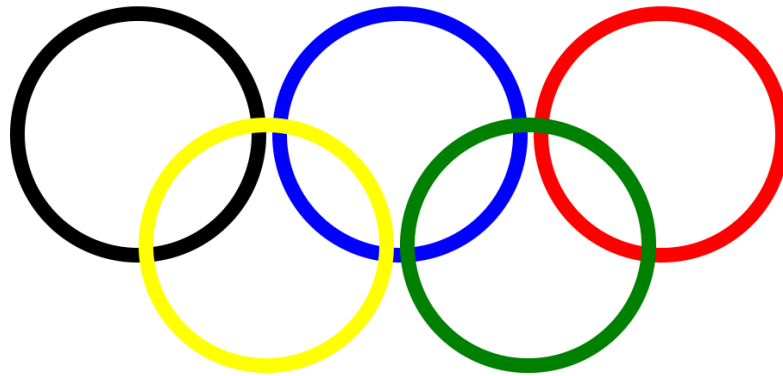


Un ejemplo visual de cómo funciona el selector **position**

Ejemplos varios:

1. Logo de los juegos olímpicos.

Logo de los Juegos Olímpicos



Ejemplo:

```
logo.html x # style.css
Arte en CSS > Logo > logo.html > html > body > div.logo > div#first-row
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>Logo de los Juegos Olímpicos</title>
5    <link rel="stylesheet" type="text/css" href="style.css">
6  </head>
7  <body>
8    <h1>Logo de los Juegos Olímpicos</h1>
9    <div class="logo">
10
11      <div id="first-row">
12        <div class="circle" id="black"></div>
13        <div class="circle" id="blue"></div>
14        <div class="circle" id="red"></div>
15      </div>
16
17      <div id="second-row">
18        <div class="circle" id="yellow"></div>
19        <div class="circle" id="green"></div>
20      </div>
21
22    </div>
23  </body>
24  </html>
25
```

Para este ejemplo serán necesarios algunas etiquetas vistas en capítulos anteriores, comenzando con los div. Será necesario agruparlos en un div que los engloba a todos, luego dos más que agruparán los aros en primera y segunda fila, respectivamente. Luego creamos los siguientes div para representar cada uno de los aros del logo, los clasificamos como círculos usando una clase y luego los identificamos con un id específico para cada color.

Ejemplo:

```
logo.html # style.css x
Arte en CSS > Logo > # style.css > #second-row
1 .logo{
2   width: fit-content;
3 }
4
5 #second-row{
6   position: relative;
7   left: 85px;
8   top: 100px;
9   z-index: 1;
10 }
11
12 #first-row{
13   z-index: 2;
14   position: absolute;
15 }
16
17
18 #blue{
19   width: 150px;
20   height: 150px;
21   border: 10px solid blue;
22   border-radius: 50%;
23   display: inline-block;
24 }
25
26 #yellow{
27   width: 150px;
28   height: 150px;
29   border: 10px solid yellow;
30   border-radius: 50%;
31   display: inline-block;
32 }
```

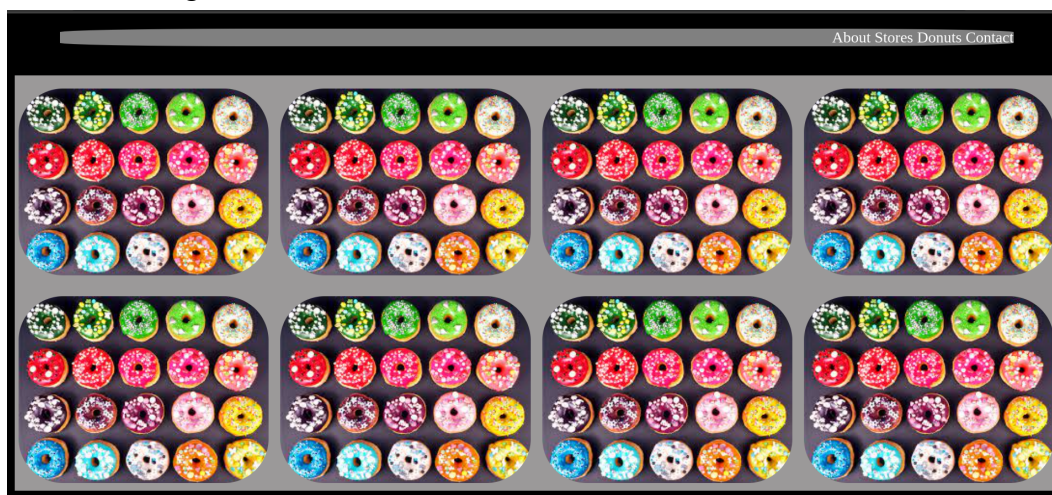
Para poder estilizar los div que fuimos creando vamos a usar los ID 's de cada uno de los círculos, además de colocar la clase logo (para hacer que el contenedor tenga el ancho de la página). Luego los ID' s de first y second row.

En el caso del ID second row, le indicamos que su posición será relativa al resto de los elementos en pantalla (por lo que su posición no se verá afectada por otros elementos en pantalla). Haciendo posible colocar anillos unos detrás de otros.

La primera fila tendrá una posición absoluta(este si se verá afectado por los demás elementos en pantalla). Por lo que sea que le pase a su contenedor padre, le afectará.

Más adelante empezamos a modificar a los aros del logo, empezando por el tamaño que estos tendrán, luego indicamos el borde del div, dándole un ancho de 10 pixeles, estilo solido y de color azul. Para luego cambiar la forma del div a uno circular y para finalizar colocamos el div en la misma línea que el resto de aros. Repetimos este proceso para cada uno de los aros, solamente cambiando el color.

2. Menú de navegación:



Para este ejemplo serán necesarios varios elementos. Comenzando con la creación del menú de la página, menú que haremos utilizando la etiqueta nav, para luego indicar que será una lista sin orden (ul) y por último los ítems de la lista (li), y su texto.

```
menu.html X # menu.css
Arte_en_CSS > Menu > menu.html > html > body > nav
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <link rel="stylesheet" href="menu.css">
8   <title>Menu de navegación</title>
9 </head>
10 <body>
11   <nav>
12     <ul>
13       <li> <a href="#">About</a> </li>
14       <li> <a href="#">Stores</a> </li>
15       <li> <a href="#">Donuts</a> </li>
16       <li> <a href="#">Contact</a> </li>
17     </ul>
18   </nav>
19
20   <div id="body-site">
21     
22     
23     
24     
25     
26     
27     
28     
29   </div>
30 </body>
31 </html>
```

Tras haber colocado los ítems del menú, procederemos a darles estilo en un archivo de CSS. Los cambios que iremos haciendo son:

- Body: se cambia el color de fondo a negro.

```
1 body{
2   background-color: black;
3 }
```

- Nav: El contenedor del menú recibirá estos cambios.
 - Indicamos el ancho del contenedor de nuestro menú

- Centramos el nav en el centro del contenedor. (0 representa a los márgenes superiores e inferiores y auto es el centrado, en el eje horizontal).
- Se posiciona el texto en el lado derecho del nav.
- Se colorea el fondo de color gris.

```

4
5  nav{
6      width: 90vw;
7      margin: 0 auto;
8      text-align: right;
9      background-color: gray;
10 }
11

```

- ul li: Todos los ítems de la lista sin orden recibirán estos cambios.
 - Se agrega una separación (desde el contenido hacia los bordes).
 - Se indica que el contenido será desplegado en una misma línea.

```

11
12  ul li{
13      padding: 5px;
14      display: inline-block;
15  }

```

- ul li a: Todos los textos de los ítems dentro del menú recibirán estos cambios.
 - La decoración del texto es removida. (subrayado, tachado, etc...).
 - Se indica que el color del texto será blanco.
 - Adicionalmente se asignaron ciertos comportamientos dependiendo del estado del texto. (si el puntero está sobre él, si ya fué seleccionado anteriormente, etc.).
 - En caso de que el puntero esté sobre él, el texto será de color azul y estará subrayado.
 - Si ya fué visitado, permanecerá de color blanco.
 - Si ya fue visitado y el puntero está sobre el, será de color azul y estará subrayado.

```

17  ul li a{
18      text-decoration: none;
19      color: white;
20  }
21
22  ul li a:hover{
23      color: blue;
24      text-decoration: underline;
25  }
26
27  ul li a:visited{
28      color: white;
29  }
30
31  ul li a:visited:hover{
32      color: blue;
33      text-decoration: underline;
34  }

```

- body-site: El contenedor del cuerpo de la página web recibirá estos cambios.
 - El texto estará separado de los bordes laterales.
 - Todo el contenedor estará separado de lo que esté arriba de él.
 - El color del fondo será cambiado a un tono de gris.
 - La altura del contenedor irá variando dependiendo del contenido que esté dentro de él.

```

35
36  #body-site{
37      padding-left: 200px;
38      padding-right: 200px;
39      margin-top: 30px;
40      background-color: rgb(155, 153, 153);
41      height: fit-content;
42  }

```

- body-site img: Todas las imágenes dentro del contenedor del cuerpo de la página web recibirán estos cambios.
 - Si el puntero está sobre ellas, el contorno se iluminará.
 - La opacidad cambiará para representar que está siendo seleccionada.

```
43
44 #body-site img{
45     padding: 4px;
46     margin-top: 10px;
47     border-radius: 50px;
48 }
49
50 #body-site img:hover{
51     background-color: ■aliceblue;
52     opacity: 30%;
53 }
```

Capítulo 4: JavaScript básico

JavaScript es un lenguaje de programación que permite agregar funcionalidad y robustez a las páginas web creadas a partir de uno o varios HTML, (entre otros lenguajes de programación). Este lenguaje nos da la posibilidad de crear animaciones, permitir el paso de información entre la página web y el servidor, entre otras cosas.

Al igual que con CSS, se tienen varias formas de implementar JavaScript en los proyectos en HTML. Pero la forma más utilizada es la de mantener archivos separados para luego importarlo en el HTML, ya que nos permite tener una estructura más limpia.

Notas adicionales:

A día de hoy hay un lenguaje similar que ha estado ganando popularidad, llamado TypeScript. Se puede decir que es una extensión de JavaScript ya que agrega características como clases y funciones. A pesar de que no tendremos un capítulo dedicado a typescript, si se verá muy superficialmente más adelante.

Otras consideraciones que deben tomarse en cuenta:

- Es posible construir una página web en su totalidad solamente utilizando JavaScript y CSS, para luego solamente importar los elementos en el html.
- La mejor manera de crear una página web robusta es utilizando algún empaquetador.

Los más utilizados son:

- Webpack (será visto en profundidad en otro capítulo).
- Adicionalmente será necesario un framework de JavaScript, algunos de los más conocidos son:

- React (será visto en profundidad en otro capítulo).
- Angular
- Vue.js
- Svelte

Atributos Básicos en JavaScripts

Declarado de variables:

- **VAR:** Variable que tiene la capacidad de ser global o estar contenido dentro de un scope.
- **LET:** Solamente son accesibles en el scope en el que fueron creadas.
- **CONST:** Una vez se le asigna un valor, no es posible asignarle otro, además de ser accesible solamente en su scope.

Condicionales:

- **IF/IF-ELSE:** Estructura condicional que se ejecuta solamente si la instrucción se cumple. O en caso de que se cumpla una que no esté en la instrucción if inicial.
- **SWITCH:** Estructura condicional de control que se utiliza para ejecutar distintos bloques de código, según el valor de una variable. Dependiendo de la situación, puede ser una forma más eficiente y legible que utilizar varios if-else.
- **TERNARY:** Es una forma abreviada de escribir un if-else en una sola línea. Siendo útil cuando se le quiere asignar un valor a una sola variable.

Loops:

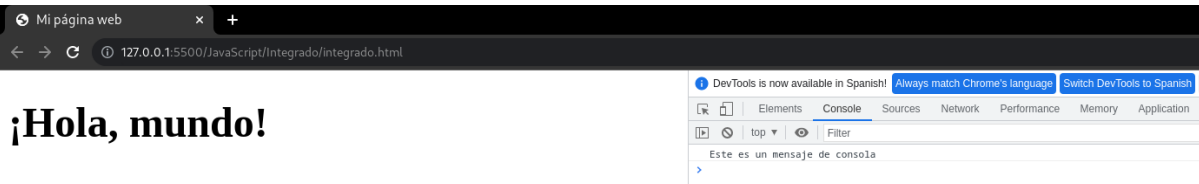
- **FOR:** Ciclo que nos permite ir incrementando o disminuyendo un valor hasta que se cumpla una condición.
- **WHILE:** Ciclo que permite que un bloque de código continúe ejecutándose hasta que una condición sea cumplida.
- **DO-WHILE:** Similar al ciclo while, pero en este caso la condición se evalúa al final de cada iteración. Permitiendo que el código se ejecute por lo menos una vez.

Gramática de JavaScript:

- **Document:** Es un objeto de JavaScript que representa el contenido de la página web. Es usado para manipular su contenido, agregar, modificar y eliminar elementos de la página. Algunas de las funciones básicas con las que cuenta el objeto document son:
 - **getElementById:** La función recibe como argumento el ID de un elemento en el html, accediendo así a dicho elemento.
 - **createElement:** Recibe como argumento el nombre de alguna etiqueta de HTML, permitiéndonos crear elementos de HTML desde el js.
 - **querySelector:** Recibe como argumento un selector de CSS y devuelve el primer el objeto que encuentra con el selector.
 - **querySelectorAll:** Similar al querySelector, pero este puede retornar un grupo de elementos que coincidan con el selector de CSS.
 - **write:** Escribe un texto en el documento de HTML. Aunque es importante tener en cuenta que esta función reescribirá todo el contenido de la página web.
 - **title:** Permite crear y/o modificar el título de la página web.

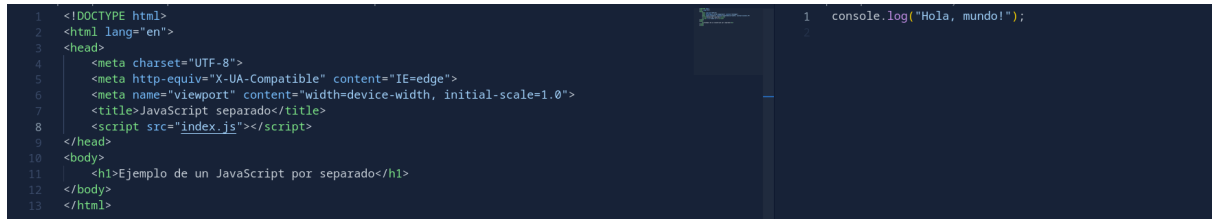
Ejemplo JavaScript integrado:

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Mi página web</title>
5    </head>
6    <body>
7      <h1>¡Hola, mundo!</h1>
8
9      <script>
10       // Código JavaScript aquí
11       console.log("Este es un mensaje de consola");
12     </script>
13   </body>
14 </html>
15
```



The screenshot shows a web browser window with the title 'Mi página web'. The address bar shows the URL '127.0.0.1:5500/JavaScript/Integrado/integrado.html'. The page content displays '¡Hola, mundo!'. The Chrome DevTools console is open, showing a message 'Este es un mensaje de consola'.

Ejemplo JavaScript separado:

A screenshot of a code editor with a dark background. The editor is split into two panes. The left pane shows an HTML file with the following code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>JavaScript separado</title>
8   <script src="index.js"></script>
9 </head>
10 <body>
11   <h1>Ejemplo de un JavaScript por separado</h1>
12 </body>
13 </html>
```

 The right pane shows a JavaScript file with the following code:

```
1 console.log("Hola, mundo!");
```

En JavaScript, al igual que en CSS, será necesario importar el archivo en el HTML. En este caso utilizando la etiqueta **script**, con su atributo *src* que nos indica en donde está almacenado el documento de JavaScript.

A la hora de manejar las importaciones de los archivos .js, es necesario tomar en cuenta que dependiendo de donde se coloque la etiqueta script hará que la página web ejecute el código antes o después. Si se desea cargarlo antes de cualquier cosa, puede colocarse en la sección de head, pero debe tomarse en cuenta que nada será cargado hasta que el código haya sido ejecutado.

Ejemplos prácticos:

Comenzamos creando el documento de HTML y el de JavaScript. Importamos el archivo de JS. Ahora será necesario agregar todos los elementos con los que vayamos trabajando, en el contenedor creado en el HTML.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta http-equiv="X-UA-Compatible" content="IE=edge">
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">
7    <title>JavaScript separado</title>
8    <script src="index.js"></script>
9  </head>
10 <body>
11   <h1>Ejemplo de un JavaScript por separado</h1>
12   <div id="container"></div>
13   <script src="form.js"></script>
14 </body>
15 </html>
```

```
const container = document.getElementById('container');
```

Ya habiendo conectado la variable container con el **div** en el HTML podemos empezar a añadir las etiquetas necesarias para la creación de un formulario. Lo primero que se debe hacer es empezar, es crear el div que va a contener a todo el formulario.

```
// Crear el formulario
const form = document.createElement('form');
form.id = 'myForm';
```

Luego iremos creando cada uno de los inputs del formulario, con su respectivo label. Primero usando la función createElement agregaremos a la página la etiqueta **label**, luego con la función textContent agregamos el texto que el usuario verá.

```
// Crear el campo de nombre
const nameLabel = document.createElement('label');
nameLabel.textContent = 'Nombre:';
```

Ya habiendo creado el **label** crearemos el **input** y le asignaremos su tipo de entrada que recibirá, id y el tipo de input que es. Por último agregaremos los elementos creados a la lista de hijos del contenedor form.

```
// Crear el campo de nombre
const nameLabel = document.createElement('label');
nameLabel.textContent = 'Nombre: ';
const nameInput = document.createElement('input');
nameInput.type = 'text';
nameInput.id = 'name';
nameInput.name = 'name';
form.appendChild(nameLabel);
form.appendChild(nameInput);
```

Repetiremos el mismo proceso con cada uno de los inputs en el formulario.

```
// Crear el campo de correo electrónico
const emailLabel = document.createElement('label');
emailLabel.textContent = 'Correo electrónico: ';
const emailInput = document.createElement('input');
emailInput.type = 'email';
emailInput.id = 'email';
emailInput.name = 'email';
form.appendChild(emailLabel);
form.appendChild(emailInput);
```

```
// Crear el campo de número de teléfono
const phoneLabel = document.createElement('label');
phoneLabel.textContent = 'Número de teléfono: ';
const phoneInput = document.createElement('input');
phoneInput.type = 'tel';
phoneInput.id = 'phone';
phoneInput.name = 'telefono';
form.appendChild(phoneLabel);
form.appendChild(phoneInput);
```

Tras haber creado los inputs, será necesario agregar el **textarea** para poder enviar mensajes en el formulario.

```
// Crear el campo del mensaje
const messageLabel = document.createElement('label');
messageLabel.textContent = 'Mensaje: ';
const messageInput = document.createElement('textarea');
messageInput.id = 'message';
messageInput.name = 'mensaje';
form.appendChild(messageLabel);
form.appendChild(messageInput);
```

Para terminar la sección de creación de elementos, necesitaremos agregar el botón para enviar todos los datos ingresados en el formulario.

```
// Crear el botón de envío
const submitButton = document.createElement('button');
submitButton.type = 'submit';
submitButton.textContent = 'Enviar';
form.appendChild(submitButton);
```

Para finalizar la creación de la página web tendremos que añadir todo lo que fuimos creando en el contenedor form, al contenedor container, el cual fue creado desde el mensaje desde el HTML. Adicionalmente le agregaremos funcionalidad al botón de enviar mediante un `addEventListener`, en donde crearemos variables locales referenciando dos **inputs** creados anteriormente, de los cuales solamente sacaremos el valor de lo que se ingrese en estos. Y por último el valor se imprimirá en la consola.

```
// Agregar el formulario al contenedor
container.appendChild(form);
```

```
// Agregar el evento de envío al formulario
form.addEventListener('submit', function(event) {
  event.preventDefault();
  const name = document.getElementById('name').value;
  const email = document.getElementById('email').value;
  console.log(`Nombre: ${name}, Correo electrónico: ${email}`);
});
```

Capítulo 5: Nivel intermedio

Para este capítulo será necesario tener una comprensión sólida de los conceptos y atributos básicos de HTML y CSS, para así poder aplicarlos eficazmente en la creación de páginas cada vez más complejas y detalladas. Entre las cosas que se necesitan para el nivel intermedio en HTML son:

- Compresión sólida de las etiquetas mayormente utilizadas en la creación de páginas web. como **html**, **head**, **h1 - h6**, **p**, **img**, **a**, **ul**, **ol**, **li**, **table**, **th**, **tr** y **td**, entre otros.
- Anidación y jerarquías. Se debe comprender como se anidan las etiquetas y como se establece la jerarquía de elementos en una página web.
- Uso de CSS. Se debe tener una comprensión básica de los selectores en CSS, con tal de dar estilo a las páginas de HTML.
- Tener una noción sobre la compatibilidad entre navegadores, de cómo se deben de hacer las páginas compatibles con la mayoría de ellos.

En este capítulo empezaremos creando páginas cada vez más atractivas, pero sin ninguna funcionalidad. Para luego ir agregando con JavaScript más y más características. Algunos de los elementos que se verán son:

KEYFRAMES: Permite definir un conjunto de reglas para implementar una animación específica.

ANIMATION: Selector que que nos permite especificar propiedades como:

- **ANIMATION-NAME:** Define el nombre de la animación.
- **ANIMATION-DURATION:** EL tiempo que la animación durará.
- **ANIMATION-DELAY:** El tiempo le tomará ejecutarse desde que se cumple la condición de ejecución.
- **ANIMATION-ITERATION-COUNT:** Define el número de veces que se ejecutará dicha animación.

TRANSITION: Define una transición gradual de una propiedad de CSS de un estado a otro.

- **TRANSITION-PROPERTY:** Define la propiedad que será cambiada. Por ejemplo: **color, padding, opacity**, etc.
- **TRANSITION-DURATION:** Define la duración de la transición, expresado en segundos o en milisegundos.
- **TRANSITION-DELAY:** Atributo opcional que permite retrasar la ejecución de la transición.
 - **EASE:** Comienza lentamente, acelera en el medio y desacelera al final. Es el elemento por defecto.
 - **LINEAR:** Transición uniforme en todo momento.
 - **EASE-IN:** Comienza lentamente y acelera al final.
 - **EASE-OUT:** Comienza rápidamente y desacelera al final.
 - **EASE-IN-OUT:** Comienza lentamente, acelera en el medio y desacelera hacia el final, pero todo de manera más pronunciada.
 - **VALOR NUMÉRICO:** Además se puede especificar un tiempo específico para el retardo que tendrá la transición.

Ejemplo:

Para este ejemplo se usará el código del menú de navegación visto en capítulos anteriores. Solamente que ahora se agregaron varios selectores de CSS para estilizar la página mediante animaciones y transiciones.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="n2.css">

  <title>Menu de navegación</title>
</head>
<body>
  <nav>
    <ul>
      <li> <a href="#">Stores</a> </li>
      <li> <a href="#">Donuts</a> </li>
      <li> <a href="#">About</a> </li>
      <li> <a href="#">Contact</a> </li>
    </ul>
  </nav>

  <div id="body-site">
    
    
    
    
    
    
    
    
    
  </div>
</body>
</html>
```

Al igual que antes, se dividió el body en varios contenedores, todo en pos de mantener una estructura limpia y organizada. Luego en sus respectivos contenedores y ID 's, se añaden las etiquetas **nav**, **ul**, **li** y **img** respectivamente.

Comenzando con el body, solamente será necesario cambiar el color del contenedor al color deseado. En este caso, será de color negro.

```
body{  
  background-color: black;  
}
```

Para el selector **nav**, se especificarán:

- Ancho del contenedor.
- Márgenes del contenedor.
- Alineación del texto dentro del contenedor.
- Color del fondo del contenedor.
- La animación que será ejecutada y su duración.

```
nav{  
  width: 90vw;  
  margin: 0 auto;  
  text-align: right;  
  background-color: gray;  
  animation: slideDown 1s;  
}
```

Se define una nueva animación personalizada que indica la posición en la que el elemento comenzará, además de especificar su transparencia. Y por último como y donde terminará dicho elemento.

```
@keyframes slideDown {  
  from {  
    transform: translateY(-50px);  
    opacity: 0;  
  }  
  to {  
    transform: translateY(0);  
    opacity: 1;  
  }  
}
```

Continuamos estilizando cada uno de los elementos en el HTML, en este caso será con padding para que los textos dentro del **nav** no estén demasiado cerca de los bordes. Para finalizar se usa la regla keyframes para definir una nueva animación que durará un segundo.

```
ul li{
  padding: 5px;
  display: inline-block;
  animation: slideRight 1s;
}

@keyframes slideRight {
  from {
    transform: translateX(-50px);
    opacity: 0;
  }
  to {
    transform: translateX(0);
    opacity: 1;
  }
}
```

Seguiremos alterando el estilo del menú de navegación. Comenzando con el texto dentro de cada una de las opciones del menú.

- Al iniciar la página el texto cambiara de color cada vez que algo le suceda, regresará a su color original.
- En caso de que el mouse esté sobre el texto del menú de navegación, cambiará a color azul de manera gradual.
- Los elementos visitados permanecerán de color blanco.
- Los elementos visitados y que en caso de que el mouse esté sobre él, cambiará a color azul y estará subrayado.

```
ul li a{
  text-decoration: none;
  color: white;
  transition: color 0.3s;
}

ul li a:hover{
  color: blue;
  text-decoration: underline;
  transition: color 0.3s;
}
```

```
ul li a:visited{
  color: white;
}

ul li a:visited:hover{
  color: blue;
  text-decoration: underline;
}
```

El contenedor de todo el cuerpo de la página será ligeramente alterado. Todos los cambios son:

- Alejar el contenido de los bordes de la página,
- Separar el menú de navegación del cuerpo de la página.
- Cambiar el color del fondo.
- Definir la altura del contenedor de manera dinámica, dependiendo del contenido dentro de él.
- El cuerpo aparece gradualmente en 1 segundo, tras haber cargado la página.

```
#body-site{
  padding-left: 200px;
  padding-right: 200px;
  margin-top: 30px;
  background-color: rgb(155, 153, 153);
  height: fit-content;
  animation: fadeIn 1s;
}

@keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}
```

Por último se estilizan las imágenes en el cuerpo de la página. Los cambios son:

- Separar todas las imágenes entre sí y del borde superior.
- En caso de que el mouse esté sobre alguna de las imágenes, la escala de la imagen se incrementará ligeramente y la opacidad se reducirá para mostrar que la imagen está siendo seleccionada correctamente.

```
#body-site img{
  padding: 4px;
  margin-top: 10px;
  border-radius: 50px;
}

#body-site img:hover{
  background-color: #aliceblue;
  opacity: 30%;
  animation: pulse 0.7s ease;
  /* padding: 6px ; */
}

@keyframes pulse {
  0% {
    transform: scale(1);
  }
  50% {
    transform: scale(1.05);
  }
  100% {
    transform: scale(1);
  }
}
```