

Instituto Profesional AIEP Spa.

Módulo 2: Fundamentos de Desarrollo Front-End

Manual para el Participante



Dirección Nacional de Educación Continua
Instituto Profesional AIEP Spa.
Junio 2021

INDICE

INTRODUCCIÓN	5
EL DESARROLLO WEB.....	6
¿Qué se entiende por desarrollo web?	6
Servidor	6
Cliente	6
Front-End	7
Back-End	7
Fullstack	8
¿Qué es el lenguaje de marcación de hipertexto? (HTML)	8
El rol del Navegador.....	9
¿Qué es la W3C?	10
Evolución del html hacia el html5.....	10
La triada html, css y javascript: contenido, presentación y comportamiento.....	11
Descarga del editor Visual Studio Code e Instalación.	12
Utilizar el potencial de un editor de texto para el desarrollo.	14
Conociendo el inspector de elementos en un navegador.....	14
¿Como usar el inspector o DevTools?	15
Ejercicios	17
El Lenguaje HTML.....	18
Características de html5.....	18
Elementos, etiquetas y atributos.....	19
Estructura básica de un documento: html, head, body, meta, title, link.....	20
Estructura del cuerpo: header, nav, section, aside, footer.	23

Encabezados: H1 ...H6.....	25
Enlaces o hipervínculos.....	25
Imágenes.....	26
Listas y listas anidadas.....	27
Tablas.....	28
Formularios: form, input, select, radio, button, submit.....	29
Elemento DIV.....	35
Ejercicios.....	35
<i>Manejando Hojas de Estilo.....</i>	36
¿Qué es CSS, fundamentos y utilidad?.....	36
CSS y HTML.....	36
Estilos en línea, embebidos, archivos externos.....	37
Referencias y selectores, por id, por clase.....	39
El modelo de cajas.....	41
Estilos más utilizados.....	41
Buenas prácticas al construir una hoja de estilos.....	42
Manejo de assets e imágenes. Conociendo rutas absolutas y relativas.....	43
Orden jerárquico de aplicación de reglas CSS y el peso asociado a las reglas.....	44
Inspeccionando estilos con las herramientas para desarrolladores en el navegador.....	45
Responsividad.....	46
El concepto de Responsividad.....	46
Tipos de dispositivos y orientaciones.....	47
El concepto Mobile First.....	48
Utilización de Media Query.....	48
Cómo probar los distintos dispositivos.....	50
Ejercicios.....	52
<i>El Framework Bootstrap.....</i>	53
¿Qué es Bootstrap?.....	53

Beneficios de su utilización.	54
Dónde obtenerlo y cómo incorporarlo a un proyecto html.	54
Elementos y estilos básicos de Bootstrap.	56
Ejercicios	63
<i>Bases del Lenguaje JavaScript.....</i>	<i>64</i>
Breve Historia de JavaScript o JS.	64
Relevancia de Javascript.	64
Qué puede y no puede hacer en el contexto de un navegador.	65
Cómo incorporar código Javascript en un documento html.	65
Selectores básicos: getElementById.....	66
Obtención y manipulación de valores y textos de los elementos del DOM.....	66
Eventos básicos: onClick y onChange.....	67
Variables.....	68
Expresiones aritméticas.....	69
Sentencias condicionales.....	70
Funciones.....	72
Cómo ejecutar código JavaScript en la consola.....	73
Depurando el código Javascript con la consola.....	74
Ejercicios.....	75
<i>jQuery Básico.....</i>	<i>76</i>
La biblioteca jQuery, ¿Por qué y cuándo usarla?	76
Obtener JQuery. Incluir y usarlo en un sitio.....	77
¿Qué es el DOM?. Manipulación de elementos del DOM con JQuery.....	81
Eventos, tipos de evento, cómo interactuar con ellos.....	82
¿Qué es y cuándo usar un plugin?	86
Ejemplos de plugins bootstrap-jQuery más comunes.....	88
Ejercicios	88
<i>Fundamentos de Git / GitHub</i>	<i>89</i>

Necesidad de un repositorio de código fuente.	89
Instalación, configuración y comandos básicos.	89
Commits y restauración de archivos.	91
Cambios de nombres.	92
Ignorando archivos.	93
Ramas, uniones, conflictos y tags.	93
Stash y Rebase.....	96
<i>Fundamentos de GitHub</i>	97
Repositorios remotos, Push y Pull.	97
Fetch v/s Pull.....	98
Clonando un repositorio.	98
Administrando Pull Request.....	100
Flujos de trabajo con GitHub.	101
Ejercicios	101
<i>BIBLIOGRAFÍA</i>	102
<i>GLOSARIO</i>	104

INTRODUCCIÓN

En el último tiempo, la internet es la forma más utilizada para la comunicación por las personas y empresas. Esta comunicación no solamente se trata de mensajes o conversaciones sino también de información, videos, streaming, entre otros recursos.

En la actualidad la mayoría de las empresas tiene de alguna forma presencia en esta gran red, mediante páginas webs, aplicaciones u otros medios que le permiten transmitir su mensaje a los posibles clientes que no solo abarcan a una comuna o región en particular, sino que también pueden abarcar a nivel nacional e inclusive internacional si el negocio lo lleva a cabo.

En el presente documento encontrará una gran cantidad de información para el desarrollo de páginas web estáticas y/o interactivas utilizando los lenguajes o tecnologías más usadas por el mercado en la actualidad.

EL DESARROLLO WEB

¿Qué se entiende por desarrollo web?

El desarrollo web es el término que hace referencia a la creación de sitios o páginas web que utiliza la internet o intranet para publicar información de distinta índole o fines.

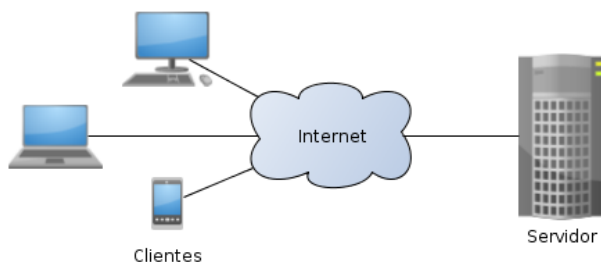
Utiliza distintas tecnologías y se compone principalmente de 2 partes:

Servidor

Utiliza aplicaciones que permiten alojar los archivos o páginas web y además de recibir y responder a las solicitudes de los usuarios.

Cliente

Utiliza cualquier dispositivo (PC, Ntbk, Celular, etc.) conectado a la red y una aplicación que realice solicitudes y procese las respuestas entregadas por el servidor.



Front-End

Es la encargada de presentar la información de forma atractiva y de manejar la interacción que el usuario tendrá con ella. Utiliza los lenguajes: HTML, CSS y JavaScript (JS).



Back-End

Es la encargada de responder a los distintos requerimientos de información de los usuarios. Utiliza distintas tecnologías para llevar a cabo esta tarea, entre las cuales están:
Lenguajes de programación: C#, PHP, Java, Python, entre otros.



Bases de datos: Oracle, PostgreSQL, SqlServer, MySQL, entre otros.



Fullstack

Es un perfil definido para el informático que maneja tanto tecnologías de Back-End como Front-End.

Para esto utiliza distintas librerías o recursos disponibles que facilitan o automatizan algunas labores para la creación de sitios web complejos, esto permite reducir tiempos y costos de desarrollo.



¿Qué es el lenguaje de marcación de hipertexto? (HTML)

El Lenguaje de etiquetas de hipertexto o HTML (sus siglas en inglés), permite, por medio de etiquetas, atributos y contenidos, la creación de páginas web.

La primera página web fue presentada por Tim Berners-Lee en 1990, que en la actualidad se puede visualizar en la siguiente URL:

<http://info.cern.ch/hypertext/WWW/TheProject.html>

Este lenguaje es interpretado o leído por el navegador, mostrando los diferentes elementos que fueron especificados en el documento HTML.

Los elementos a mostrar en el navegador pueden ser especificados mediante los etiquetas o tags.



El rol del Navegador.

El navegador web es el principal cliente web que se utiliza en la actualidad, para navegar en los sitios web existentes en la internet.

Algunos navegadores disponibles en el mercado son:



¿Qué es la W3C?

Es un consorcio internacional, creado en el año 1994, que se encarga de desarrollar las recomendaciones y estándares que permitan asegurar el funcionamiento de la internet, independiente de las empresas tecnológicas y/o herramientas que se utilicen para soportarla.

Algunos estándares publicados por la entidad son:

- HTML
- XML
- SOAP
- CSS
- DOM



El sitio oficial de este consorcio es: <https://www.w3.org/>

Evolución del HTML hacia el html5

HTML5, es la última versión de lenguaje, lanzado en el 2014.

Incluye nuevas características entre las cuales están:

- Soporte de Video y Audio.
- Creación de video juegos.
- Permite incluir gráficos vectoriales SVG (formato).
- Etiquetas para la maquetación.
- Validaciones.



La triada HTML, css y javascript: contenido, presentación y comportamiento.

En la actualidad, todas las páginas web utilizan de una u otra forma estos 3 lenguajes.

HTML dispone de una serie de etiqueta que permiten armar el “esqueleto” de la página web.

Estas etiquetas o tags se distribuyen de manera ordenada para establecer los elementos donde se quieren mostrar.



CSS, son las siglas para Hoja de estilo en cascada.

Actualmente está en la versión 3 y es un lenguaje de diseño gráfico que permite modificar la apariencia de una página web.

Este lenguaje permite modificar a los elementos de una página sus colores, tamaño, ubicación o posición, cambiar el tipo de fuente (letra), entre otros.



JavaScript o JS, es un lenguaje de programación interpretado diseñado inicialmente por Netscape (1995) y que es mantenido actualmente por la fundación Mozilla.

Funciona en el lado del cliente (navegador). La versión actual del lenguaje es la ECMAScript 2016. Permite crear páginas web dinámicas agregándole funcionalidades según las acciones que realiza el usuario en la página web.



Descarga del editor Visual Studio Code e Instalación.

Visual Studio Code, es un editor de código fuente, multiplataforma, con soporte de extensiones de tercero, creado por Microsoft y que fue lanzado en el año 2015 bajo licencia MIT.

Su Código fuente está disponible en GitHub para su descarga, lo que permite modificar y adaptar de mejor manera, el editor a nuestras necesidades.

Visual Studio Code o VS Code o también VSC, tiene soporte para múltiples lenguajes de programación, como, por ejemplo: Python, PHP, HTML, CSS, Entre otros.



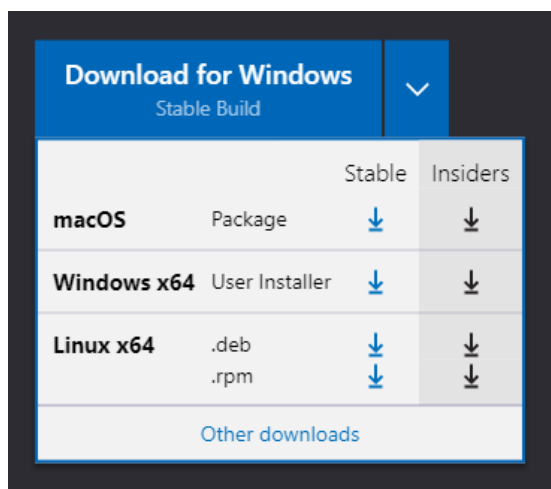
VS Code, puede ser cargado desde su sitio oficial:

<https://code.visualstudio.com>

Visual Studio Code

Y se debe seleccionar el sistema operativo que se utiliza en el equipo a usar y descargar.

Al descargar, se ejecuta el archivo y el asistente instalará los diferentes componentes para el funcionamiento del editor.

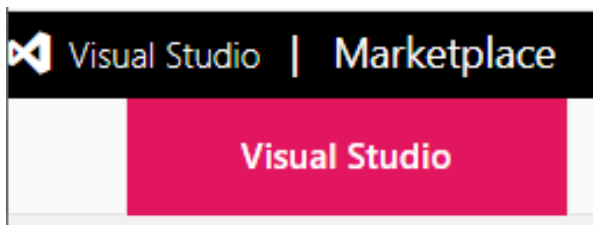


Utilizar el potencial de un editor de texto para el desarrollo.

Visual Studio, no solo permite escribir páginas web, también es un conjunto completo de extensiones o plugins que están disponibles con el editor.

Estos plugins, permiten:

- Sugerencias de códigos.
- Completar códigos.
- Destaca errores de sintaxis.
- Corrección de errores en el código.
- Depuración.
- Entre otros.



La URL para ver extensiones disponibles es: <https://marketplace.visualstudio.com/>

Conociendo el inspector de elementos en un navegador.

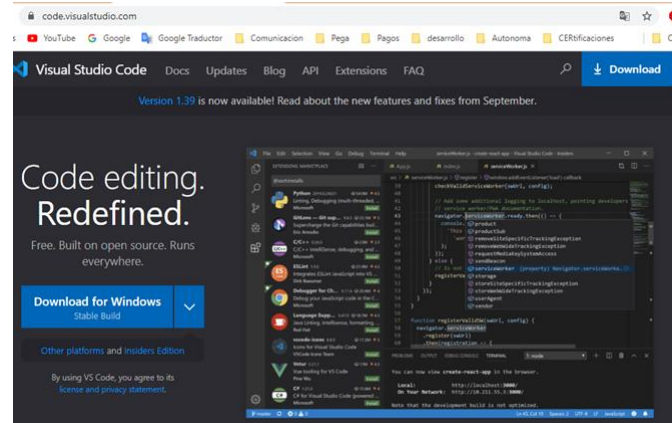
Otra herramienta a utilizar en el desarrollo de aplicaciones en entorno web es el inspector de elementos. El inspector, permite visualizar el código fuente con el que se construye una página web. Los principales códigos que se pueden visualizar con el inspector son:

- CSS o hoja de estilo en cascada, quien aplica el diseño de la página.
- JavaScript, quien determina el comportamiento de la página, según la interacción que el usuario realiza.
- HTML es el esqueleto en que se sostienen todas las páginas web.

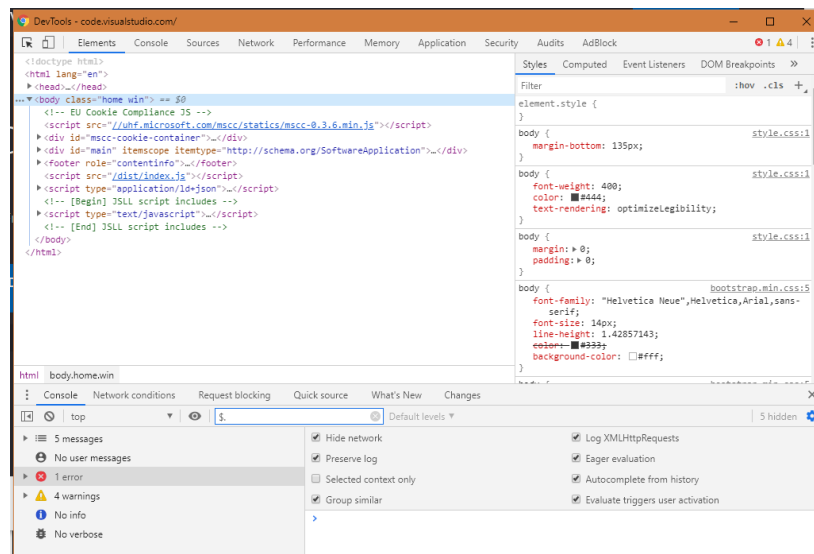
¿Como usar el inspector o DevTools?

Abrir el navegador Chrome y entrar en algún sitio de interés. Por ejemplo:

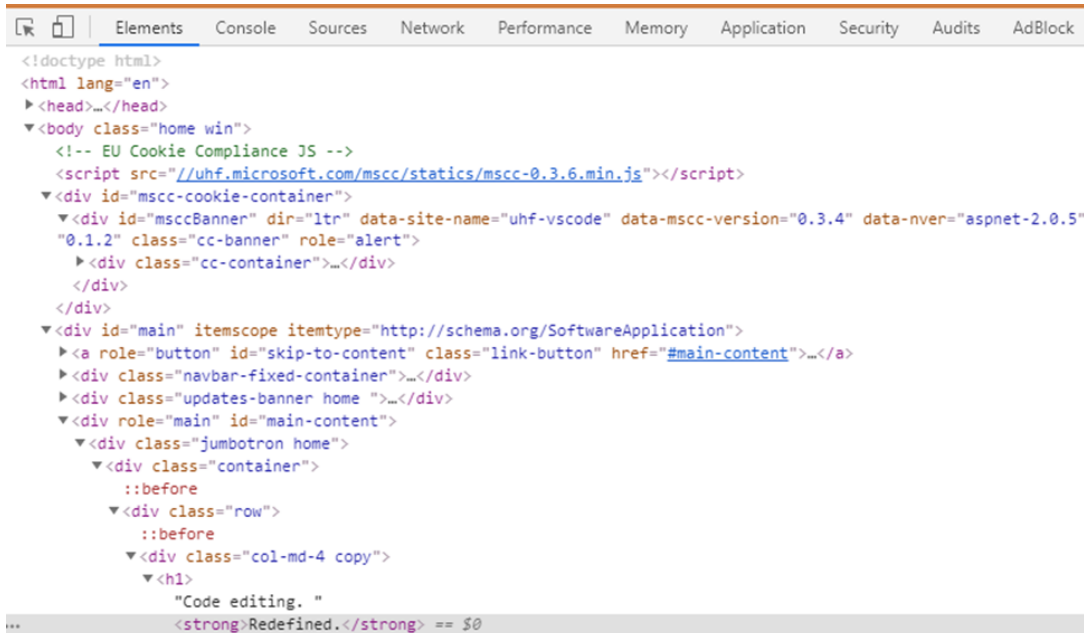
<https://code.visualstudio.com>



Una vez cargada la página en Chrome, se presiona F12 y aparecerá una ventana llamada DevTools.



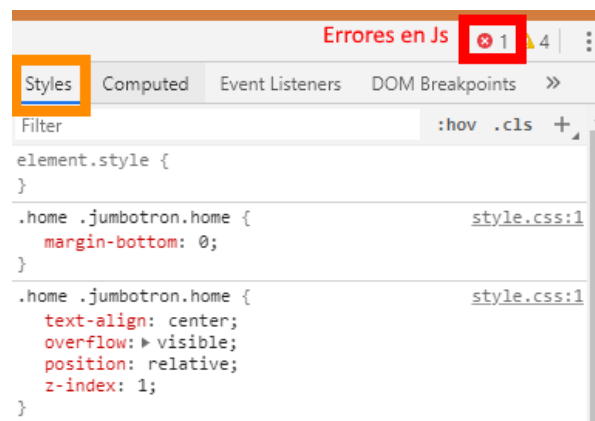
A la izquierda aparece el código HTML de la página.



```

<!doctype html>
<html lang="en">
  <head>...</head>
  <body class="home win">
    <!-- EU Cookie Compliance JS -->
    <script src="//uhf.microsoft.com/mscc/statics/mscc-0.3.6.min.js"></script>
    <div id="mscc-cookie-container">
      <div id="msccBanner" dir="ltr" data-site-name="uhf-vscode" data-mscc-version="0.3.4" data-nver="aspnet-2.0.5"
        "0.1.2" class="cc-banner" role="alert">
        <div class="cc-container">...</div>
      </div>
    </div>
    <div id="main" itemscope itemtype="http://schema.org/SoftwareApplication">
      <a role="button" id="skip-to-content" class="link-button" href="#main-content">...</a>
      <div class="navbar-fixed-container">...</div>
      <div class="updates-banner home ">...</div>
      <div role="main" id="main-content">
        <div class="jumbotron home">
          <div class="container">
            ::before
            <div class="row">
              ::before
              <div class="col-md-4 copy">
                <h1>
                  "Code editing. "
                  <strong>Redefined.</strong> == $0
  
```

A la derecha se puede ver los estilos aplicados en la página como también si existen errores en la ejecución o de sintaxis en JavaScript o JS.



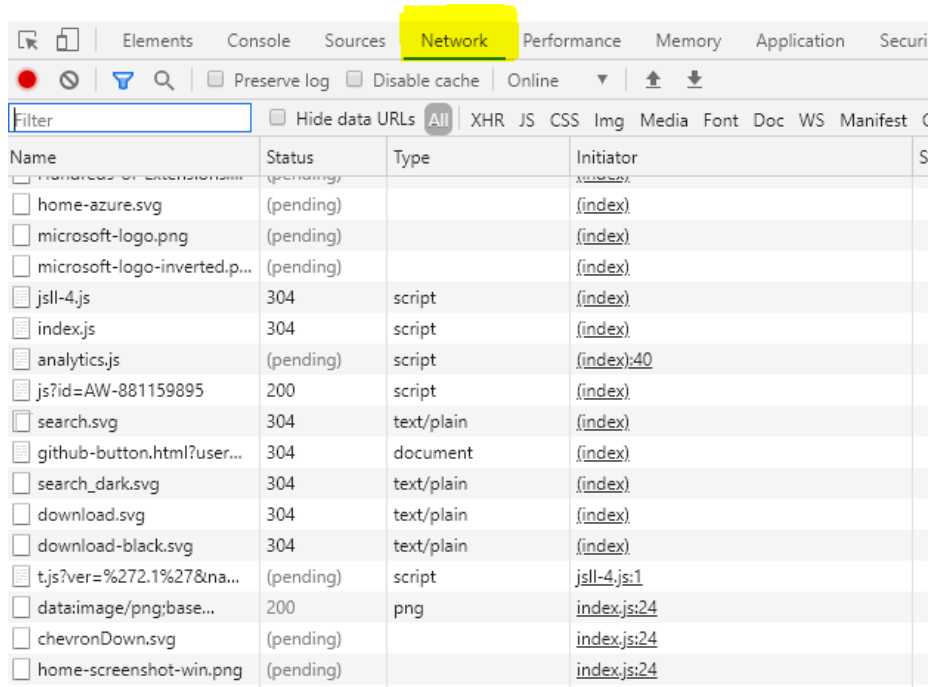
```

element.style {
}

.home .jumbotron.home {
  margin-bottom: 0;
}

.home .jumbotron.home {
  text-align: center;
  overflow: visible;
  position: relative;
  z-index: 1;
}
  
```

Otra opción importante para el desarrollo de una página web es el visor de archivos que fueron descargados cuando se realizó la solicitud de la página. La opción tiene como nombre NetWork.



Name	Status	Type	Initiator
home-azure.svg	(pending)		(index)
microsoft-logo.png	(pending)		(index)
microsoft-logo-inverted.p...	(pending)		(index)
jsll-4.js	304	script	(index)
index.js	304	script	(index)
analytics.js	(pending)	script	(index):40
js?id=AW-881159895	200	script	(index)
search.svg	304	text/plain	(index)
github-button.html?user...	304	document	(index)
search_dark.svg	304	text/plain	(index)
download.svg	304	text/plain	(index)
download-black.svg	304	text/plain	(index)
t.js?ver=%272.1%27&na...	(pending)	script	jsll-4.js:1
data:image/png;base...	200	png	index.js:24
chevronDown.svg	(pending)		index.js:24
home-screenshot-win.png	(pending)		index.js:24

Ejercicios

1. Buscar otros lenguajes del lado del cliente y BDD no nombrados.
2. Buscar algunos plugin que nos ayuden a desarrollar en HTML, JS y CSS.
3. Inspeccionar 2 páginas y analizar lo que muestra el Devtools.

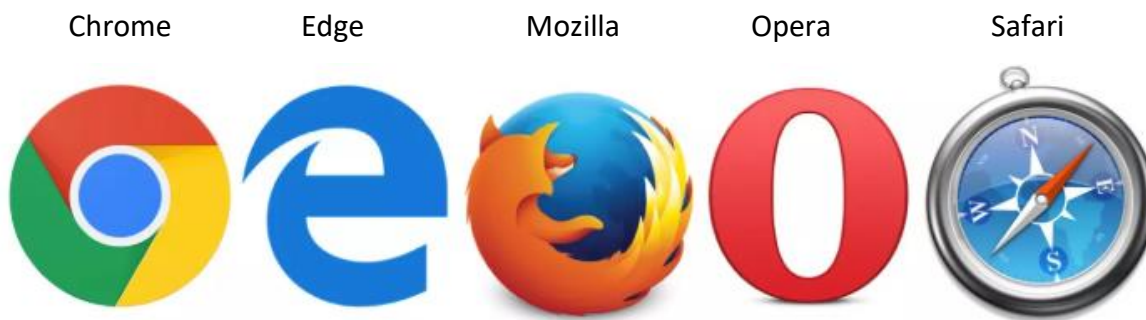
El Lenguaje HTML

Características de html5.

- Html5 es la última versión del lenguaje.
- Etiquetas semánticas.
- Almacenamiento mediante API.
- Interacción con usuarios sin conexión (usando el almacenamiento local).
- Soporte de recursos como video o audio directamente.
- Etiquetas nuevas (video, audio, nav, article, header, etc.).
- Nuevos tipos de controles (fechas, numero, correo, etc.).

La última versión disponible es HTML5 que es soportado por la mayoría y más conocidos exploradores disponibles en el mercado.

Algunos exploradores que soportan HTML son:



Para revisar el soporte de los diferentes elementos de cada navegador, pueden visitar la siguiente página: <https://caniuse.com>.

Elementos, etiquetas y atributos.

En HTML, los elementos son los diferentes controles que están disponible en una página web, por ejemplos: las cajas de texto, cajas de selección, imágenes, videos, audios, entre otros.

Las etiquetas son los códigos que permiten crear la página web. Estas etiquetas permiten crear los elementos mencionados en el punto anterior y para ello, se deben respetar la sintaxis de cada uno de ellos.

Adicionalmente, estas etiquetas cuentan con atributos o propiedades que permiten modificar la presentación de los elementos, por ejemplo: cambiar el tipo de letra, el color, el tamaño, entre otros.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <label>Su nombre es: </label>
    <input type="text" readonly=readonly value="María Ignacia">
    <input type="text" disabled>
    <input type="text" required>
  </body>
</html>
```

Buscar

Fecha

Hora

Número

Color

Rango

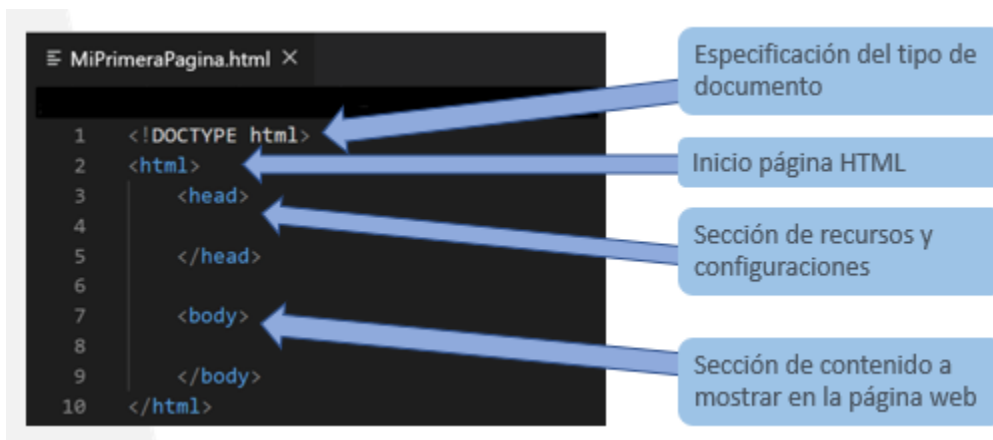
Email

Url

Telefono

Estructura básica de un documento: html, head, body, meta, title, link.

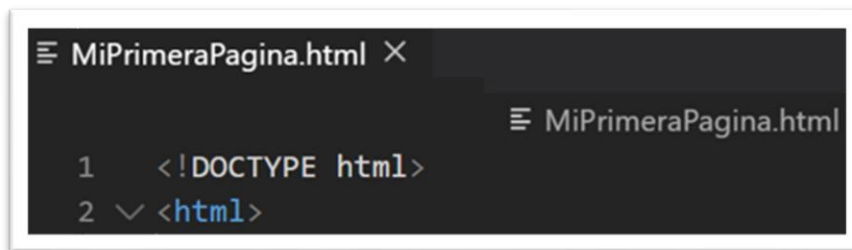
La estructura básica de un documento HTML es la siguiente:



La declaración `<!DOCTYPE html>` permite señalar al navegador, que el documento actual, es de tipo HTML. Esta declaración, debe estar en la primera línea del documento.

El tag `<html>` representa la raíz de un documento HTML. Dentro de este tag, deben estar todos los elementos que conforman la página web.

Para señalar el termino o cierre del tag se agrega el carácter `/`. Ejemplo `</html>`



El tag <meta> permite especificar información acerca del documento HTML.

Las opciones disponibles son:

- charset : define el juego de caracteres que utiliza la página.
- description : Permite añadir una descripción de la página.
- keywords : Permite definir palabras claves que se registrarán en los motores de búsquedas.
- author : Permite especificar el autor o creador de la página.
- viewport : Señala al navegador el cómo controlar las dimensiones y escala de la pagina.

La sección <head>, permite definir los recursos que conforman la página web.

Estos recursos pueden ser de los siguientes tipos:

- Favicon.
- Script (JS, jQuery, etc.).
- Estilos CSS.
- Meta data (configuración del idioma, por ejemplo).
- Título de la página.
- Entre otros.

Esta etiqueta, siempre debe estar dentro del <html>. El carácter adicional / , señala el cierre de etiqueta. Ejemplo: </head>

La sección <body>, define el cuerpo del documento HTML, en el cual, se organizan de los elementos que conforman la página web a mostrar. Estos elementos pueden ser:

- imágenes.
- Textos.
- hipervínculos.
- Tablas.
- Listas.
- Entre otros.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
  </body>
</html>
```

Esta sección debe estar dentro del bloque del tag <html> como muestra la imagen.

El carácter adicional / , señala el cierre de tag. Ejemplo: </body >

Ejemplo:



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"> <!-- Permite el uso de palabras con tilde -->
    <meta name="description" content="Programa FullStack">
    <meta name="keywords" content="Página HTML,Tag, input, meta">
    <meta name="author" content="Hernan Jaeger">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mi Página</title>
  </head>
  <body>
    <h1>Uso del tag meta</h1>
  </body>
</html>
```

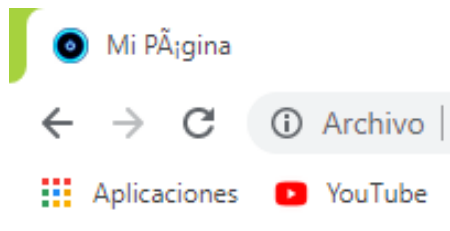
Al agregar charset utf-8. la pagina soportará tildes y eñes.

Mi Página

Uso del tag meta

El tag <title> permite establecer el título que tendrá la página.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mi Página</title>
  </head>
  <body>
  </body>
</html>
```

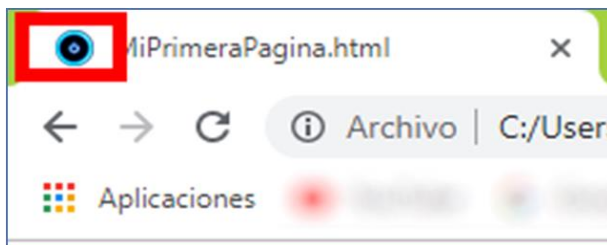


Resultado de no agregar el **charset** correspondiente

El tag <link> permite agregar diferentes recursos a la página, como, por ejemplo, el favicon o icono de página favorita, permite asociar una imagen a la página como también, al acceso directo.

Otros recursos que se pueden asociar son: CSS, librerías JavaScript, entre otros.

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="shortcut icon" href="icon.png">
  </head>
  <body>
  </body>
</html>
```

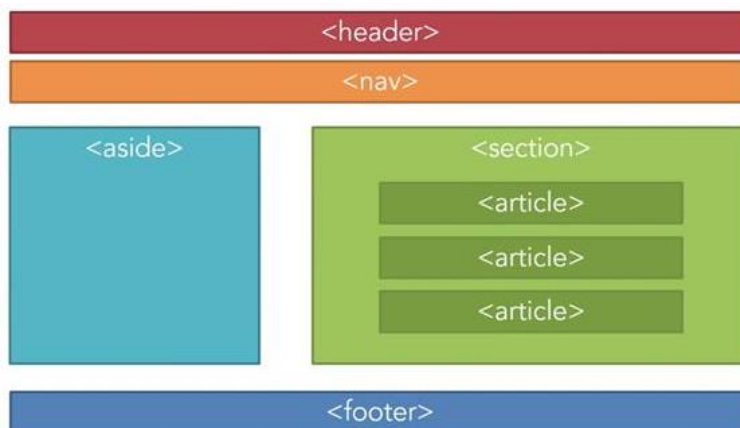


Estructura del cuerpo: header, nav, section, aside, footer.

Estas etiquetas tienen como nombre etiquetas semánticas.

Sirven para identificar las distintas secciones que conforman una página web.

En la siguiente imagen, se detalla la estructura definida en HTML5.



Tag header.

Define el encabezado o introducción de la sección en la que fue declarada. Puede estar dentro de la sección body como dentro de la sección section.

Tag nav.

Especifica las opciones o menú que tendrá disponible la página para navegar dentro del sitio. Por lo general contiene los tags ul y li.

Tag section.

Define una sección con temas en común, por ejemplo: características de los diferentes tipos de lenguajes de programación.

Tag aside.

Permite definir un contenido que con una menor importancia dentro de la página. Por lo general, su posición será en el lado izquierdo.

Tag footer.

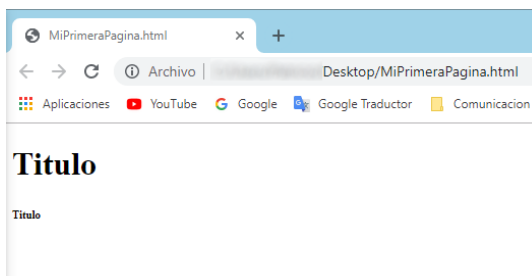
Define la sección que va al final de cada página. Es común que contenga información de contacto como, por ejemplo: el número de teléfono, correo, direcciones, entre otros.

Encabezados: H1 ...H6.

`<h1>`, permite especificar un encabezado o título en una página web. Existen 6 tipos en total y se diferencian por el tamaño de la letra.

Ejemplo:

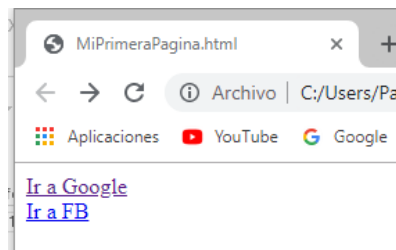
```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <h1>Titulo</h1>
    <h6>Titulo</h6>
  </body>
</html>
```



Enlaces o hipervínculos.

La etiqueta `<a>`, permite agregar hipervínculos o link a nuestro sitio. El atributo **target=_blank**, señala al navegador, que debe abrir una nueva pestaña para abrir la página señalado en el link.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <a href="http://www.google.cl">Ir a Google</a>
    <br>
    <a href="http://www.facebook.cl" target=_blank>Ir a FB</a>
  </body>
</html>
```



Imágenes.

`` Permite mostrar una imagen en la página.

Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <label>Funciones de cada tecnología</label><br>
    
  </body>
</html>
```



`` + `<a>`

Permite que una imagen sea un link.

En el cuadro destacado aparece la URL a la que apunta el tag `<a>`

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <label>imagen como link</label><br>
    <a href="http://www.google.cl">
      
    </a>
  </body>
</html>
```



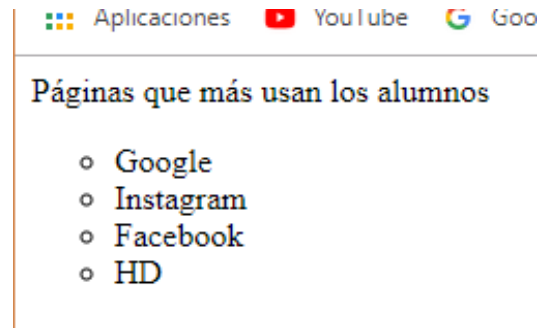
Listas y listas anidadas.

 y

Permite crear listas no ordenas. Tiene la opción de establecer el signo modificando la propiedad *type*, las opciones son: *disk*, *square* y *circle*

Ejemplo:

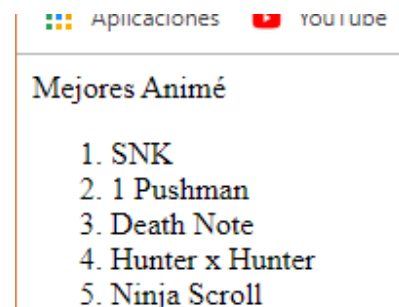
```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <!-- Esto es un comentario.
    No es ejecutado o leído por el navegador-->
    <label>Páginas que más usan los alumnos </label>
    <!-- tipos disponibles circle, disk, square -->
    <ul type=“circle”>
      <li>Google</li>
      <li>Instagram</li>
      <li>Facebook</li>
      <li>HD</li>
    </ul>
  </body>
</html>
```



 y Permite crear listas ordenas.

Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <label>Mejores Animé </label>
    <ol type=“1” <!-- tipos disponibles a, 1, i, I -->
      <li>SNK</li>
      <li>1 Pushman</li>
      <li>Death Note</li>
      <li>Hunter x Hunter</li>
      <li>Ninja Scroll</li>
    </ol>
  </body>
</html>
```



Tablas.

<table> permite crear una tabla para mostrar la información de forma ordenada, en filas y columnas.

Para crear una tabla se requieren las siguientes etiquetas:

- Las filas son creadas mediante el tag <tr>.
- Las columnas mediante el tag <td>.
- Los títulos de cada columna se pueden crear con el tag <th>.

```
<table>
  <tr>
    <th>Encabezado 1</th>
    <th>Encabezado 2</th>
    <th>Encabezado 3</th>
  </tr>
  <tr>
    <td>dato A 1</td>
    <td>dato A 2</td>
    <td>dato A 3</td>
  </tr>
  <tr>
    <td>dato B 1</td>
    <td>dato B 2</td>
    <td>dato B 3</td>
  </tr>
</table>
```

Formularios: `form`, `input`, `select`, `radio`, `button`, `submit`.

El tag `<form>`, permite crear un formulario que contiene distintos inputs, los cuales, conforman la información requerida por el servidor para ser procesada.

Los atributos asociados al form son:

action : pagina que procesará la información, por lo general, también devolverá una respuesta.

method: especifica la forma como se envía la información. Las opciones más usadas son:

GET : la información es enviada por medio de la url, lo que permite verlos directamente. Esta opción, en lo posible, se debe evitar.

POST : la información es enviada de forma oculta al servidor, por ende, no se ven.

name : asigna un nombre al formulario.

enctype: especifica como serán codificado los datos que se envían al servidor si el método usado es POST.

`<input>`

Permite la captura de información mediante una caja de texto, como también, el envío de información.

Este tag no requiere cierre.

Existen diferentes tipos de input y propiedades útiles para ayudar a la captura de información.

El input, al ser usado como botón, se debe agregar el tag **`<form>`**.

El tag **<form>**. debe contener todos los inputs que conforman la página, lo que permitirá, enviar toda la información al servidor cuando el usuario presiona el input de tipo submit (botón).

<input> tipo **text**:

type = Especificar el comportamiento (y forma) del **<input>**. Existen una variedad de opciones que se verán más adelante.

name = Asigna un nombre que permitirá identificar al tag input. Este nombre será usado por el servidor para capturar el valor que el usuario a especificado. Por convención, el nombre siempre empieza con txt.

La sintaxis de tag **<input>** es:

```
<body>
  <label>Su nombre es: </label>
  <input type="text" name="txtNombre">
</body>
</html>
```

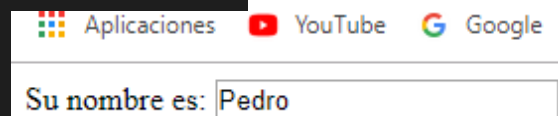


Aplicaciones YouTube Google

Su nombre es:

value es un atributo opcional que permite especificar un valor por defecto.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <label>Su nombre es: </label>
    <input type="text" name="txtNombre" value="Pedro">
  </body>
</html>
```

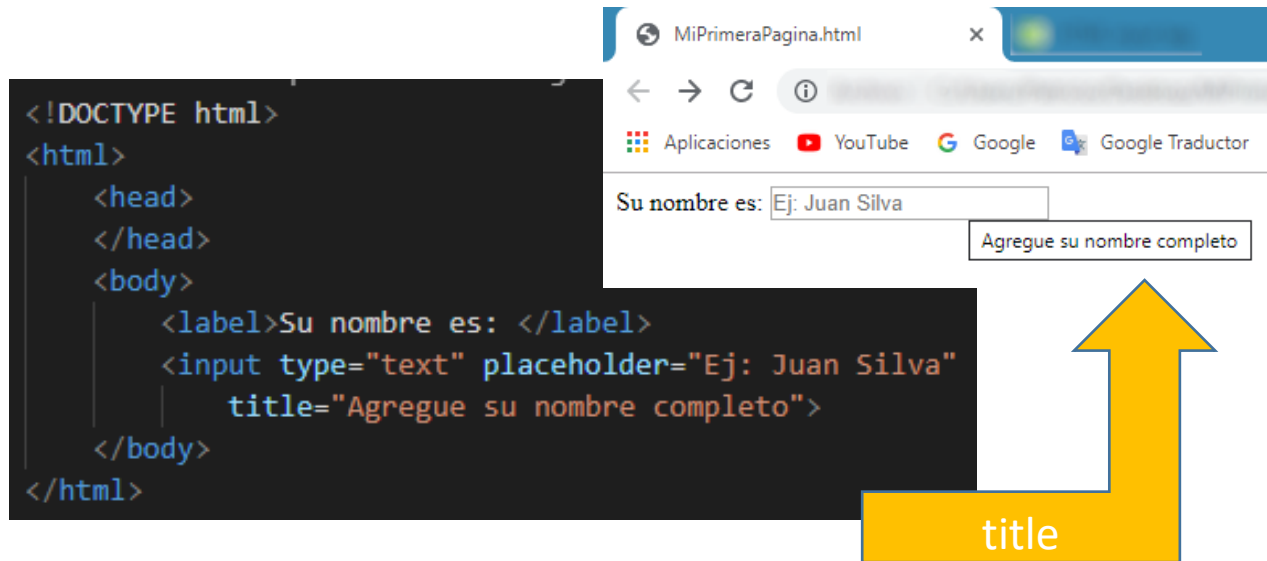


Aplicaciones YouTube Google

Su nombre es:

Los atributos **placeholder** y **title**, permite entregar una guía o ayuda al usuario.

Ejemplo:



Otros atributos disponibles:

- El atributo **readonly** permite restringir la escritura en el input.
- El atributo **disabled** permite bloquear al input y evita el envío al servidor.
- El atributo **required** permite obligar al usuario a especificar un valor (validar).

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <label>Su nombre es: </label>
    <input type="text" readonly=readonly value="María Ignacia">
    <input type="text" disabled>
    <input type="text" required>
  </body>
</html>
```


Input tipo **password**, oculta o encripta lo que el usuario escribe.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <h2>Ingrese clave</h2>
    <input type="password" name="txtClave" maxlength="10" title="señale su clave">
  </body>
</html>
```



Otros inputs disponibles para la captura de datos:

```
<!-- campo oculto. No se dibuja en la página -->
<br><input type=hidden name=txtId>
<!-- agrega un botón para limpiar el input-->
<br><input type=search name=txtBuscar>
<!-- Crea un calendario para seleccionar una fecha -->
<br><input type=date name=txtFecha>
<!-- Crea un seleccionador de hora -->
<br><input type=time name=txtHora>
<!-- Crea un input que solo permite numeros -->
<br><input type=number name=txtNumero>
<!-- Crea una paleta de colores -->
<br><input type=color name=txtColor>
<!-- Crea un ajustador de valores -->
<br><input type=range name=txtRango>
<!-- Otros input para validar email, url y teléfono
por ahora sin soporte -->
<input type=email name=txtMail>
<input type=url name=txtUrl>
<input type=tel name=txtTel>
```

Otros inputs

Buscar

Fecha

Hora

Número

Color

Rango

Email

Url

Telefono

Otra forma de mostrar opciones al usuario es con el tag select.

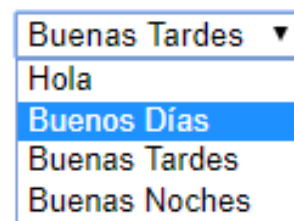
El tag select despliega una serie de opciones para que el usuario seleccione sola una.

El atributo selected, permite “dejar” seleccionado una opción por defecto.

Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <h2>Listado</h2>
    <select name=combo>
      <option value=1>Hola</option>
      <option value=2>Buenos Días</option>
      <option value=3 selected>Buenas Tardes</option>
      <option value=4>Buenas Noches</option>
    </select>
  </body>
</html>
```

Listado



Input tipo radio, permite dar una serie de opciones al usuario y para que este, pueda seleccionar solo uno. Este efecto se logra si todos los radios tienen en mismo nombre en el atributo name.

El atributo checked, permite “dejar” marcado un radio por defecto.

Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <label>cantidad de opción:</label><br>
    <input type=radio name=opcion value=1> Opción Uno<br>
    <input type=radio name=opcion value=2 checked> Opción Dos<br>
    <input type=radio name=opcion value=3> Opción Tres
  </body>
</html>
```

cantidad de opción:

- ☐ Opción Uno
- ☒ Opción Dos
- ☐ Opción Tres

Input tipo checked, permite dar una serie de opciones al usuario y contrario al anterior, puede seleccionar más de uno. El atributo checked, permite “dejar” marcado un radio por defecto.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <h2>Lista de opciones</h2>
    <input type=checkbox name=opcion value=1>Opcion 1
    <input type=checkbox name=opcion value=2>Opcion 2
    <input type=checkbox name=opcion checked value=3>Opcion 3
  </body>
</html>
```

Lista de opciones

☐ Opcion 1 ☐ Opcion 2 ☒ Opcion 3

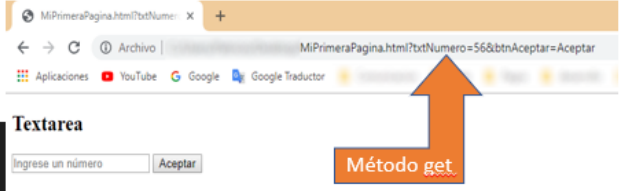
Input tipo **submit**, permite el envío de la información al servidor. Para esta acción, se debe especificar junto a un formulario, usando el tag **<form></form>**.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <h2>Botón</h2>
    <input type=submit name=btnAceptar value=Aceptar>
  </body>
</html>
```



Ejemplo de envío de información al servidor.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <h2>Textarea</h2>
    <form action="MiPrimeraPagina.html" method="get">
      <input type="input" name="txtNumero" placeholder="Ingrese un número">
      <input type="submit" name="btnAceptar" value="Aceptar">
    </form>
  </body>
</html>
```



Método get

Nota: Se debe tener el tag form e input, de tipo submit, para el envío de los datos.

Elemento DIV.

La etiqueta <DIV>, se utiliza para dividir o seccionar elementos dentro de una página web.

Este elemento no se ve en el navegador.

Es posible aplicar distintas propiedades mediante CSS, como, por ejemplo: color, posición, entre otros.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <div>

  </div>
</body>
</html>
```

Ejercicios

Crear una página que solicite los siguientes datos:

Rut, dígito verificador, nombre, nacionalidad, selección de grado académico, fecha de nacimiento, título o cargo, si está o no con trabajo(cesante), años de servicios, hijos.

Manejando Hojas de Estilo.

¿Qué es CSS, fundamentos y utilidad?

Las Hojas de estilo CSS, es una tecnología que permite aplicar colores, formas, márgenes, entre otros efectos para crear paginas visualmente más atractivas.

Los estilos definidos pueden ser aplicado a una o varias páginas de forma masiva.

CSS son las siglas de Cascading Style Sheets o Hoja de estilo en cascada en español.

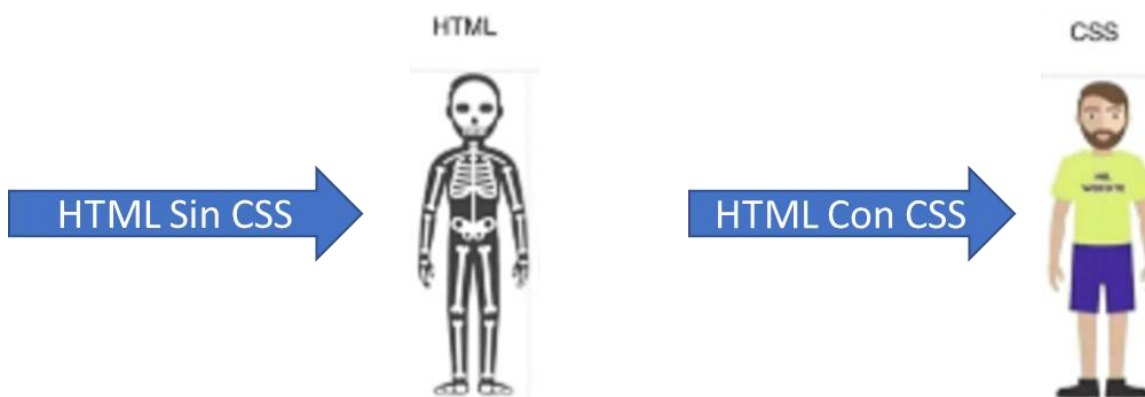
La última versión disponible es la CSS3.



CSS y HTML.

Al comprender el uso de CSS, podemos dar una estética más amigable a nuestro esqueleto HTML.

Principalmente, CSS se utiliza para aplicar distintas propiedades a los elementos HTML para que la página web sea atractiva visualmente para los usuarios que la visitan.



Estilos en línea, embebidos, archivos externos.

Existen 3 formas de aplicar estilos con CSS.

En línea: Es cuando se especifica el estilo directamente en el elemento.

Por ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Jerarquía</title>
  </head>
  <body>
    <h1 style="color: #0000ff;">Ejemplo de CSS </h1>
  </body>
</html>
```

Ejemplo de CSS

Incrustado o interna.

Es cuando se especifica el estilo dentro del documento HTML.

Por ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Jerarquía</title>
    <style>
      .titulo { color : #ff0000; }
    </style>
  </head>
  <body>
    <h1 class="titulo">Ejemplo de CSS </h1>
  </body>
</html>
```

Ejemplo de CSS

Vinculado o Externa.

Es cuando se especifica el estilo por medio de la **invocación** de un archivo CSS.

Por ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Jerarquía</title>
    <link rel="stylesheet" href="jerarquia.css">
  </head>
  <body>
    <h1 class="titulo">Ejemplo de CSS </h1>
  </body>
</html>
```

Archivo CSS con nombre: jerarquía.css

```
> # jerarquia.css > .titulo
.titulo { color : #00ff2afb; }
```

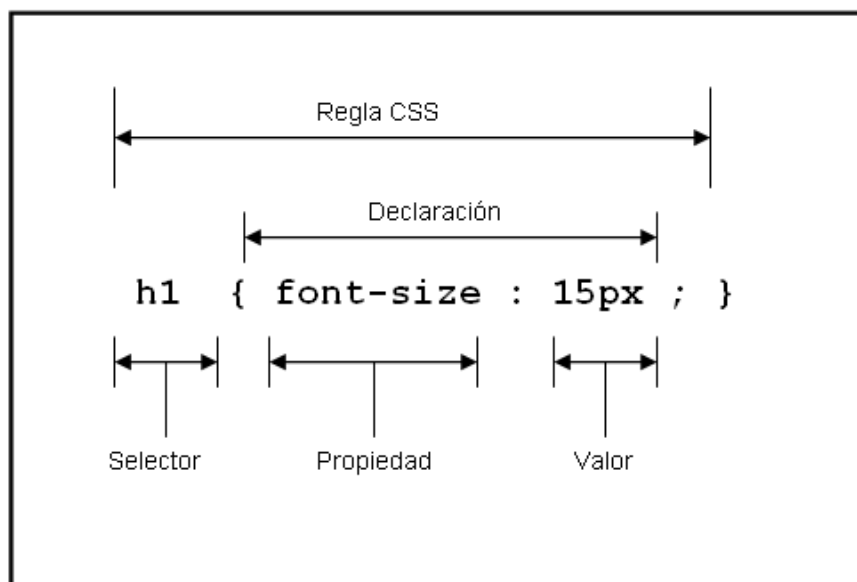
Resultado:

Ejemplo de CSS

Referencias y selectores, por id, por clase.

La sintaxis básica de CSS3 es la siguiente:

```
selector { propiedad : valor; propiedad : valor; }
```



Donde:

- **Selector:** es donde se especifica el elemento donde se quiere aplicar algún estilo.

Para identificar al elemento, se puede utilizar:

- **Por Tipo o Nombre del tag del elemento:**

Esta acción afectará a todos los elementos de la página.

Ejemplo: para el tag <h1>, se nombre como **h1**.

Para aplicar o referenciar propiedades a un elemento HTML, se utilizan las palabras reservadas:

- **Id:** El nombre a utilizar debe ser único dentro de la página, por lo que no se debe asignar a otro elemento dentro de la página. El símbolo para referenciar el id en CSS es # (gato).

```
</head>
<body>
  <div id="divImagen">

  </div>
</body>
</html>
```

```
#divImagen { width: 50%; }
```

- **Class:** Permite tener más de un nombre asociado. Lo que implica que se pueden agregar más de un estilo o propiedades.

El nombre de una clase se puede repetir en varios elementos, donde se aplica las mismas propiedades. El símbolo para referenciar una clase en CSS es el . (punto).

```
</head>
<body>
  <div class="titulo">

  </div>
</body>
</html>
```

```
</head>
<body>
  <div class="titulo colores formas">

  </div>
</body>
</html>
```

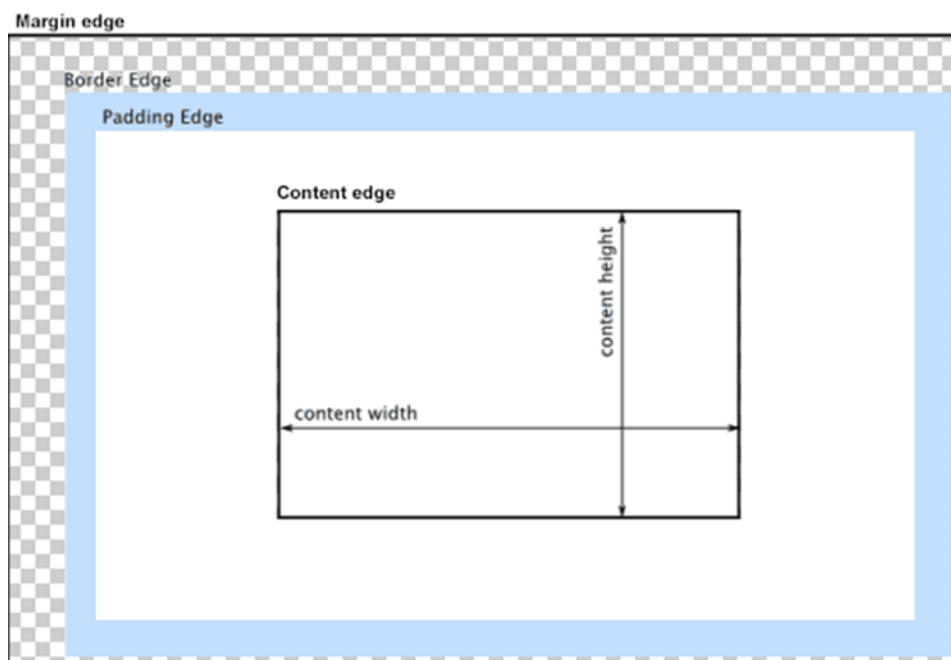
```
> # jerarquia.css > .titulo
.titulo { color : #00ff2afb; }
```

El modelo de cajas.

El modelo de cajas es la representación, con forma de rectángulo, que se da, a cada elemento dibujado por el navegador.

Esta forma tiene asociado las siguientes propiedades **content**, **padding**, **border** y **margin**.

Diagrama:



Estilos más utilizados.

Estilos o propiedades más utilizadas son las siguientes:

- background : Define una imagen o color, del fondo de un elemento.
- color : Define el color de la fuente(letra).
- font-size : Define el tamaño de la fuente (letra).
- font-family : Define el tipo de letra a utilizar.
- text-align : Define la alineación del texto.
- Border : Define el estilo del borde del elemento.
- Height : Define el alto que tendrá el objeto.
- Width : Define el ancho que tendrá el objeto.

Otras propiedades visitar el siguiente link:
https://developer.mozilla.org/es/docs/Web/CSS/Referencia_CSS

Buenas prácticas al construir una hoja de estilos.

Buenas prácticas, hace referencia a la seguir una serie de reglas establecidas por la industria. Estas reglas son opcionales de seguir, pero es bueno tenerlas presente al momento de construir un archivo CSS. Algunas buenas prácticas son:

- Utilizar tabulación.
- Nombres consistentes con el elemento o zona a afectar
- Organizar los estilos que se aplicaran de arriba hacia abajo
- Combinar elementos que tendrán el mismo estilo.
- En lo posible, utilizar varias clases en los elementos.
- Al finalizar la página, comprimir el documento para reducir su tamaño.

Manejo de assets e imágenes. Conociendo rutas absolutas y relativas.

Assets:

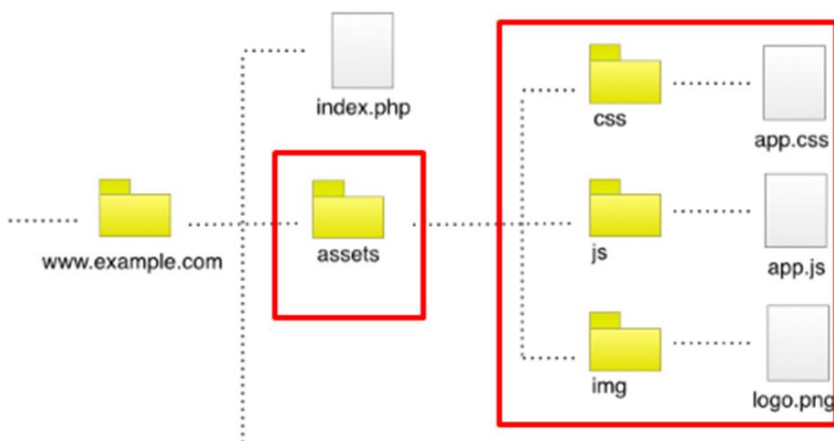
Son carpetas que contienen archivos o recursos adicionales al documento HTML que son utilizados en la página para complementar la interfaz gráfica de la página web.

Estos recursos son agrupados, por lo general, según el tipo u objetivo del recurso.

Algunos recursos utilizados son:

- Hoja de estilos en cascada o CSS
- Imágenes en distintos formatos. Por ejemplo: PNG, JPG, GIF, etc.
- JavaScript
- Videos en formato MP4, AVI, etc.

Ejemplo:



Rutas

Existen 2 tipos de rutas que se utilizan en las paginas web:

Rutas absolutas

Usada principalmente para acceder a recursos fuera de nuestro dominio.

Ejemplo: `https://www.dominio.cl/assert/css/`

Rutas relativas

Usado principalmente para adjuntar recursos disponibles localmente a nuestro sitio.

Ejemplo: `/assert/images/`

Orden jerárquico de aplicación de reglas CSS y el peso asociado a las reglas.

En CSS, existe la jerarquía de aplicación, el que consiste en que solo un estilo se aplica, dependiendo de la precedencia que tenga la forma, en la siguiente muestra desde la primera prioridad hasta la de menor prioridad.

1. **En línea.**
2. **Incrustada o interna.**
3. **Vinculado o externa.**
- 4.

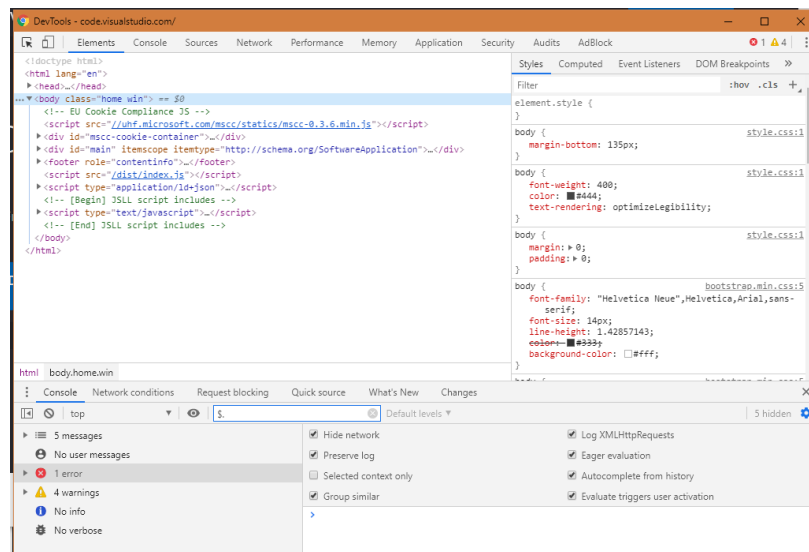
Si por algún motivo se repite una propiedad para un mismo elemento, el que se aplica o perdura, será, la **última** propiedad señalada.

Inspeccionando estilos con las herramientas para desarrolladores en el navegador.

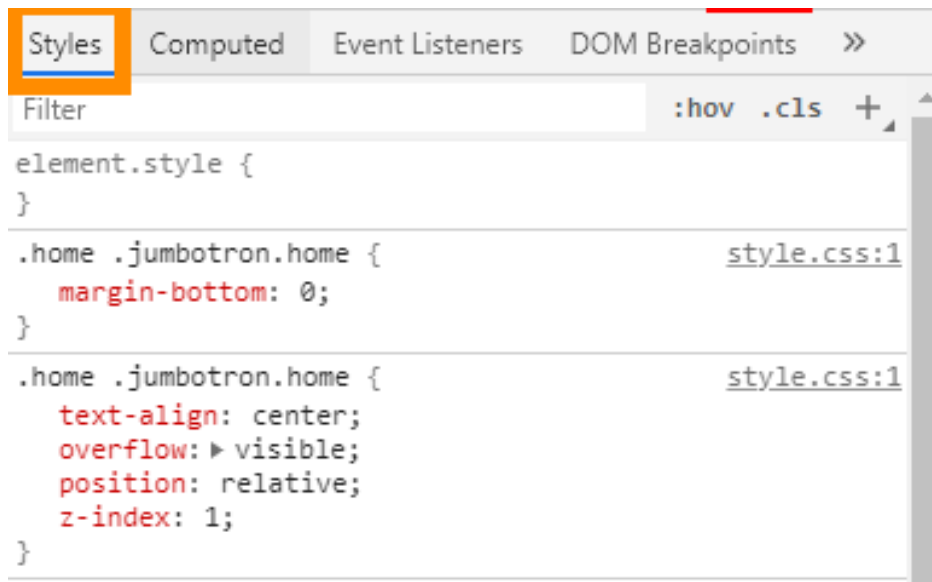
Al utilizar Chrome como navegador el inspector, aparece cuando se presiona F12 o con el botón derecho del mouse, la opción inspeccionar. Esta herramienta tiene el nombre de Dev Tools.

En ella se puede ver:

- El código HTML
- La consola (JS)
- Los archivos descargados
- Almacenamiento local
- Errores
- Entre otras informaciones acerca de la página.



Para ver los estilos aplicados a una página, se debe seleccionar la opción “Styles”.



Responsividad.

El concepto de Responsividad.

El concepto de responsividad está relacionado al diseño de interfaces gráfica y tiene como objetivo, que la visibilidad de una página web, se adapte a cualquier tipo de dispositivo que el usuario use para visitarla.

Por lo tanto, la página debería estar disponible, sin errores gráficos, en todos los dispositivos disponibles en el mercado, como por ejemplo: móviles, Tablet, PC, notebook, o cualquier otro con conexión al sitio.

En la actualidad existen varios “Framework” que ayudan al desarrollo de páginas webs responsivas, entre las cuales están:

- Bootstrap (Ampliamente usado y recomendado)
- Foundation
- Materialize
- jQuery Mobile
- PureCSS



Pure



[Tipos de dispositivos y orientaciones.](#)

Los tipos de dispositivos a considerar para el desarrollo de una página web son:

- Teléfonos móviles.
- Tablet.
- PC y/o Notebooks.
- Televisores.



Las orientaciones se consideran las siguientes:

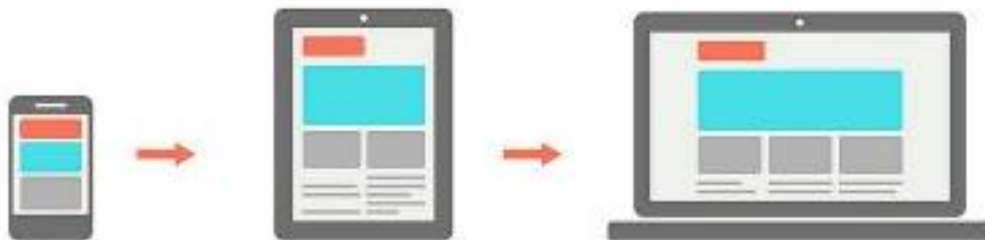
- Horizontal o Landscape
- Retrato o Portrait.



El concepto Mobile First.

El concepto **Mobile First** hace referencia a los “Framework” que ayudan a crear página webs responsivas y que, por defecto, el diseño inicial es para un **dispositivo móvil**, sin perjuicio de que posteriormente, se adapte a pantallas más grandes si es requerido.

Mobile First Web Design



Utilización de Media Query.

Las “**media query**”, permiten aplicar un estilo en función del tipo de dispositivo o según la resolución o ancho de la pantalla en que se ve la página.

Se utiliza la instrucción `@medio` para definir la resolución y estilo por aplicar.

Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>@media</title>
    <style>
      .titulo { color: grey; }
      @media (max-width: 800px) {
        .titulo { color:blue; }
      }
    </style>
  </head>
  <body>
    <h1 class="titulo">Ejemplo de @media </h1>
  </body>
</html>
```

Para abarcar todas las pantallas de los dispositivos mencionados, se puede utilizar la siguiente tabla de resoluciones:

Dispositivo	Resolución
Móviles	< 576px con orientación portrait.
Móviles	mínimo 576px con orientación horizontal.
Tablet	mínimo 768px o más.
Notebooks y Equipos de escritorio	mínimo 992px o más.
Pantallas Grandes (tv, etc.)	mínimo 1200px o superior.

Referencia de resolución: <https://getbootstrap.com/docs/4.3/layout/overview/>

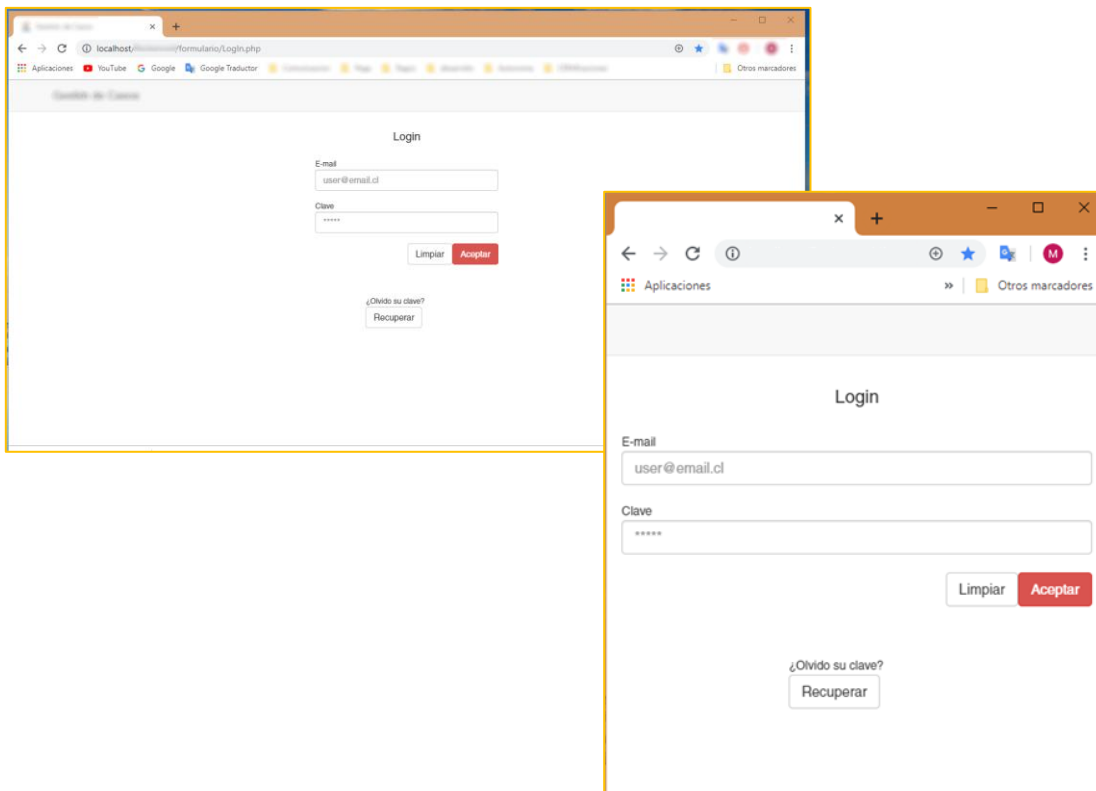
Cómo probar los distintos dispositivos.

Al diseñar una página responsiva, se debe disponer de la herramienta adecuada para probar los diferentes formatos de dispositivo de forma virtual y así, evitar disponer del equipo físico para hacer las pruebas.

Existen dos formas de probar los formatos con el navegador (Chrome). La primera, es cambiando el tamaño del navegador para que aumente o reduzca la resolución y utilizar el DevTools.

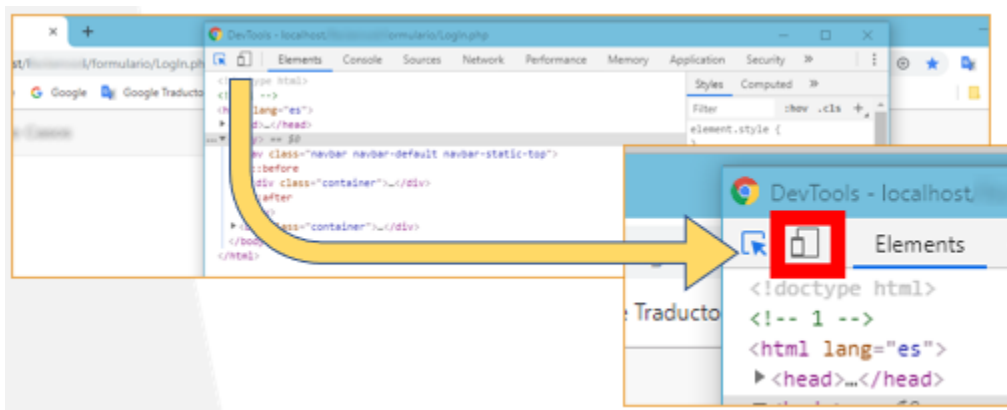
DevTools, viene integrado por defecto varios dispositivos móviles, también, está la opción de agregar otros dispositivos que se requieran para probar la página.

Reduciendo el tamaño del navegador.

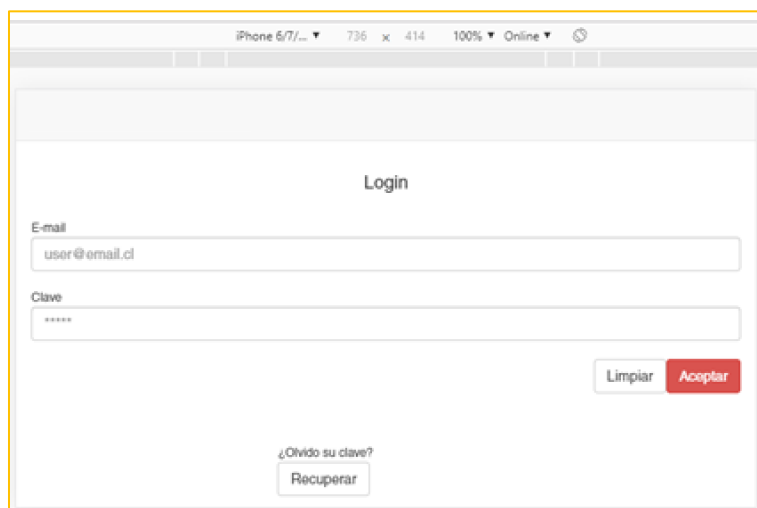
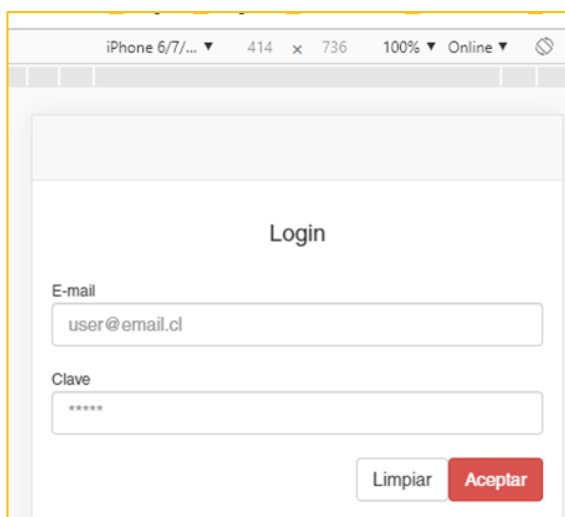


DevTools.

Para acceder a **DevTools**, se debe presionar botón derecho y escoger la opción inspeccionar o también presionando **ctr + shift + i**.



Ejemplo del navegador con vista de un iPhone, tanto en formato *portrait* como *landscape*



Ejercicios

- Crear una página web y aplicar estilos usando class o id.
- Ingresar a una página y ver con el inspector del navegador las propiedades del modelo de cajas.
- Crear una página y aplicar una propiedad con las diferentes opciones presentadas y ver el resultado.
- Con el inspector o Devtools, ingresar a una página y registre los estilos que tiene.
- Buscar otros **framework** que permitan responsividad.
- Crear una página y probar las media query.
- Probar con una página web, con distintos dispositivos y orientaciones.

El Framework Bootstrap.

¿Qué es Bootstrap?

Bootstrap es un *framework* de código abierto para el diseño de páginas webs enfocado en el Font-End o la gráfica de una página.

Provee de plantillas de diseño para formularios, botones, menús de navegación, títulos, entre otros elementos utilizando para ello, HTML y CSS. Existen algunas funcionalidades que requieren JavaScript para su implementación.

La versión actual en uso es la 5.x, pero también está ampliamente en uso la versión 4.x y 3.4.

Página web oficial de Bootstrap: <https://getbootstrap.com/>



Beneficios de su utilización.

Principales beneficios del uso de Bootstrap:

- Incluye una serie de componentes ya diseñados y reutilizables.
- Es fácil de implementar y utilizar.
- Usa un sistema de grilla que adapta de forma automática a diferentes resoluciones la página creada.
- No requiere de un diseñador, a menos que el proyecto lo requiera.
- Reduce el tiempo de diseño de la interfaz.
- Cuenta con una extensa documentación.

Dónde obtenerlo y cómo incorporarlo a un proyecto HTML.

Para utilizar Bootstrap:

Agregar CDN a nuestra página desde la ruta <https://getbootstrap.com/docs/4.5/getting-started/download/>

```
<html>
  <head>
    <meta charset="UTF-8">
    <title>Jerarquía</title>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js">
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com
  </head>
  <body>
    <h1>Integración de Bootstrap</h1>
  </body>
</html>
```

Descargar e integrar de forma local los archivos de Bootstrap.

Obtener ruta <https://getbootstrap.com/docs/4.5/getting-started/download/>

Compiled CSS and JS

Download ready-to-use compiled code for **Bootstrap v4.5.0** to easily drop into your project, which includes:

- Compiled and minified CSS bundles (see [CSS files comparison](#))
- Compiled and minified JavaScript plugins

This doesn't include documentation, source files, or any optional JavaScript dependencies (jQuery and Popper.js).

Download

Archivos requeridos localmente y código para agregarlo al proyecto.

Nombre	Tipo	Tamaño con
bootstrap.css	Documento de hoja de es...	
bootstrap.css.map	Linker Address Map	
bootstrap.min.css	Documento de hoja de es...	
bootstrap.min.css.map	Linker Address Map	
bootstrap-grid.css	Documento de hoja de es...	
bootstrap-grid.css.map	Linker Address Map	
bootstrap-grid.min.css	Documento de hoja de es...	
bootstrap-grid.min.css.map	Linker Address Map	
bootstrap-reboot.css	Linker Address Map	
bootstrap-reboot.css.map		
bootstrap-reboot.min.css		
bootstrap-reboot.min.css.map		

Nombre	Tipo
bootstrap.bundle.js	Linker Address Map
bootstrap.bundle.js.map	Linker Address Map
bootstrap.bundle.min.js	Linker Address Map
bootstrap.bundle.min.js.map	Linker Address Map
bootstrap.js	Archivo JavaScript
bootstrap.js.map	Linker Address Map
bootstrap.min.js	Archivo JavaScript
bootstrap.min.js.map	Linker Address Map


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Bootstrap</title>
    <script src="js/bootstrap.min.js"></script>
    <link rel="stylesheet" href="css/bootstrap.min.css">
  </head>
  <body>
    <h1>Integración de Bootstrap</h1>
  </body>
</html>
```


Nombre
css
js
Ejemplo.html

Elementos y estilos básicos de Bootstrap.

Container

Clase que permite envolver la página a crear, limita el ancho máximo de visibilidad de los estilos de los componentes siguientes a él. Se utilizará para implementar el sistema de grilla.

Los 2 clases más usados son: **container** y **container-fluid**.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Bootstrap</title>
    <script src="js/bootstrap.min.js"></script>
    <link rel="stylesheet" href="css/bootstrap.min.css" >
  </head>
  <body>
    <div class="container-fluid">
      <div class="alert alert-primary" role="alert">
        Código para un alerta simple
      </div>
    </div>
  </body>
</html>
```

Grilla

Bootstrap utiliza un sistema de grillas para la división de la página y sirve como referencia para la visualización de los elementos en la página.

Algunas características del sistema de grilla son:

	Extra pequeño <576px	Pequeño ≥576px	Mediano ≥768px	Grande ≥992px	Extra grande ≥1200px
Ancho máximo del contenedor	Ninguno (auto)	540px	720px	960px	1140px
Prefijo de clase	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# de columnas	12				

La adaptación de los contenidos de una página a las diferentes pantallas se realiza mediante la especificación de las siguientes clases:

- col
- col-sm
- col-md
- col-lg
- col-xl

Considerar que Bootstrap 4 en adelante es mobile-first.

Extra pequeño ≤ 576px	Pequeño ≥ 576px	Mediano ≥ 768px	Grande ≥ 992px	Extra grande ≥ 1200px
Ninguno (auto)	540px	720px	960px	1140px
.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
12				

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Bootstrap</title>
    <script src="js/bootstrap.min.js"></script>
    <link rel="stylesheet" href="css/bootstrap.min.css" >
  </head>
  <body>
    <div class="container">
      <div class="row"> <!-- declaración de fila -->
        <!-- Por defecto (mobile-first) cada div tendrá 12 columnas (m--)
        <div class="col-md-3"> <!-- la columna tendrá 3 de 12 columnas -->
          <div class="form-group">
            <label for="Nombre">Nombre</label> <input type="text" class="form-control" placeholder="Juan">
          </div>
        </div>
        <div class="col-md-9"> <!-- la columna tendrá 9 de 12 columnas -->
          <div class="form-group">
            <label for="EstadoCivil">Estado Civil</label>
            <select class="form-control">
              <option>Soltero</option><option>Casado</option><option>Viudo</option><option>Divorciado</option>
            </select>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```

Tabla

La clase definida en Bootstrap para las tablas es **table** y existen varios tipos de estilos disponibles. Por defecto:

#	Primero	Último	Resolver
1	Marcos	Otón	@mdo
2	Jacob	Thornton	@grasa
3	Larry	el pájaro	@gorjeo

Al agregar otra clase se puede modificar, por ejemplo: `.table-dark`.

#	Primero	Último	Resolver
1	Marcos	Otón	@mdo
2	Jacob	Thornton	@grasa
3	Larry	el pájaro	@gorjeo

Existen otras opciones disponibles para cambiar la apariencia como, por ejemplo:

- `.thead-ligth`.
- `.thead-dark`.
- `.table-striped`.
- `.table-bordered`.
- `.table-borderless`.
- `.table-primary` (o cualquier clase con el color definido en Bootstrap).
- `.table-responsive` (aplicar en un div).

Para más info acerca de clases o estilos que se puede aplicar a una tabla, visitar:

<https://getbootstrap.com/docs/4.1/content/tables/>

JumboTron

```

<html>
  <head>
    <meta charset="UTF-8">
    <title>Bootstrap</title>
    <script src="js/bootstrap.min.js"></script>
    <link rel="stylesheet" href="css/bootstrap.min.css" >
  </head>
  <body>
    <div class="container">
      <div class="jumbotron">
        <h1 class="display-4">¡Hola Mundo!</h1>
        <p class="lead">Componente JumboTron</p>
        <hr class="my-4">
        <p>texto amplio para expresar una idea</p>
        <a class="btn btn-primary btn-lg" href="#" role="button">
          Presionar aquí
        </a>
      </div>
    </div>
  </body>
</html>

```

¡Hola Mundo!

Componente JumboTron

texto amplio para expresar una idea

Presionar aquí

Alert

A simple primary alert—check it out!

```

<html>
  <head>
    <meta charset="UTF-8">
    <title>Bootstrap</title>
    <script src="js/bootstrap.min.js"></script>
    <link rel="stylesheet" href="css/bootstrap.min.css" >
  </head>
  <body>
    <div class="alert alert-primary" role="alert">
      Código para un alerta simple
    </div>
  </body>
</html>

```

Button



Para probar, pegar dentro del BODY, el botón deseado.

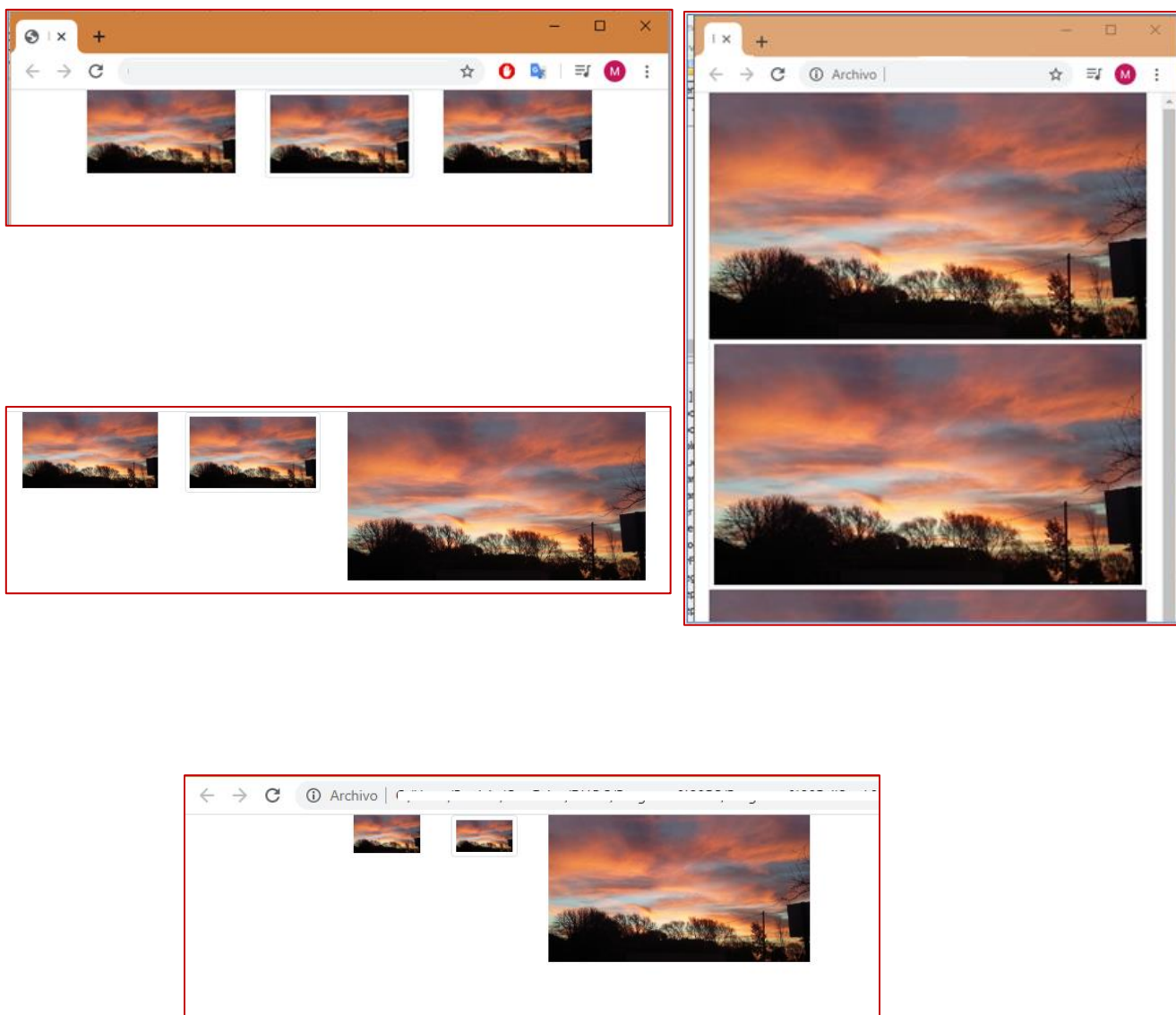
Imágenes.

Para agregar imágenes a nuestro sitio y que no presente problemas a los cambios de resolución de la página, es utilizando las siguientes clases para que automáticamente se adapte a cada una de ellas. La imagen usada tiene una resolución de 3264x1836.

Código de Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Bootstrap</title>
    <script src="js/bootstrap.min.js"></script>
    <link rel="stylesheet" href="css/bootstrap.min.css">
  </head>
  <body>
    <div class="container">
      <div class="row">
        <div class="col-12 col-sm-4 col-md-3 col-xl-1 text-center">
          
        </div>
        <div class="col-12 col-sm-4 col-md-3 col-xl-1 text-center">
          
        </div>
        <div class="col-12 col-sm-4 col-md-6 col-xl-3 text-center">
          
        </div>
      </div>
    </div>
  </body>
</html>
```

La visualización de las diferentes configuraciones realizadas con col-X col-sm-X col-md-X col-xl-X y la clase img-fluid.



Navbar

Permiten presentar un menú con las opciones que tendrá la página.

Navbar Home Features Pricing Disabled

Código de ejemplo:

Para probar, pegar dentro del BODY, el botón deseado.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav"
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>

      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Features</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Pricing</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
      </li>
    </ul>
  </div>
</nav>
```

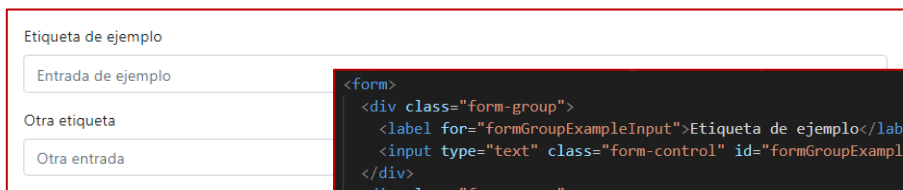
Forms:

Los formularios permiten crear páginas webs que permitan la captura de datos y posterior envío de estos al servidor.

En BootStrap, existe dos tipos de form, form-group y form-control:

- form-group : permiten agregar una estructura a los elementos que formaran el formulario.
- Form-control : permite ajustar la medida del control según las medidas de la celda donde se encuentra, agrandando o achicando según en la resolución en que se ve la página.

Ejemplo:



```
<form>  
  <div class="form-group">  
    <label for="formGroupExampleInput">Etiqueta de ejemplo</label>  
    <input type="text" class="form-control" id="formGroupExampleInput" placeholder="Example input">  
  </div>  
  <div class="form-group">  
    <label for="formGroupExampleInput2">Otra etiqueta</label>  
    <input type="text" class="form-control" id="formGroupExampleInput2" placeholder="Another input">  
  </div>  
</form>
```

Ejercicios

- Crear formulario para solicitar datos nombre, apellido, teléfono, lugar de trabajo, cargo, años en la actual empresa, entre otros.
- Las paginas deben adaptarse a los distintos tipos de pantallas.
- Investigar en la documentación oficial las diferentes clases y componentes para utilizarlas en este ejercicio.

Bases del Lenguaje JavaScript.

Breve Historia de JavaScript o JS.

- JavaScript es un lenguaje de programación del lado del cliente (se ejecuta en el navegador).
- Fue diseñado en 1995, por la empresa Netscape para su navegador con el mismo nombre.
- La sintaxis es similar al lenguaje C y Java, pero tiene otro propósito.
- Es un lenguaje interpretado, débilmente tipado y orientado a objetos.
- La última versión es la ECMAScript 6, publicado por ECMA que es la encargada de regular el lenguaje.



Relevancia de JavaScript.

- Es el estándar de la mayoría de los navegadores web en la actualidad.
- Hoy en día es común utilizar JS para realizar solicitudes y recibir respuesta del servidor mediante AJAX.
- Soporta el formato Json o XML.
- Según la última información entregada por ECMA, el lenguaje no tendrá más modificaciones.
- Se utiliza principalmente para reducir las peticiones al servidor, mediante la validación de los formularios (campos vacíos, verificar rut correcto y/o correo, restricción de caracteres, limitar caracteres, limitar valores, entre otros tipos de validaciones).

Qué puede y no puede hacer en el contexto de un navegador.

Con JavaScript, las tareas que se pueden realizar son:

- Aplicar efectos visuales como aparecer o desaparecer elementos.
- Aplicar estilos de forma dinámica.
- Animaciones.
- Asociar eventos (clic, keypress, etc.).
- Realizar peticiones al servidor.
- Gestionar el teclado.
- Mensajería.
- Entre otros.



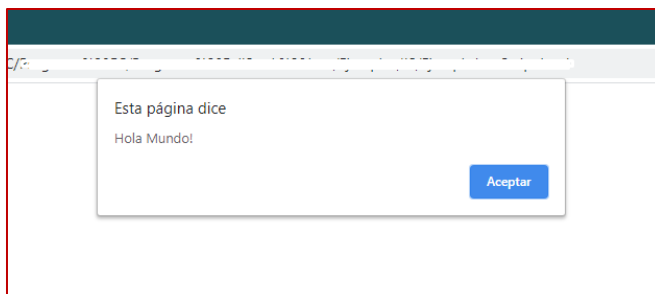
Cómo incorporar código JavaScript en un documento HTML.

Para agregar JavaScript a nuestra página se deben utilizar los tag `<script>` y `</script>` para cerrar el bloque.

Otra forma es importar un documento con extensión JS, al igual como se realiza con CSS.

Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>JavaScript</title>
    <script>
      alert("Hola Mundo!");
    </script>
  </head>
  <body>
  </body>
</html>
```



Selectores básicos: getElementById.

- Es un método disponible dentro del documento HTML.
- La sintaxis es document.getElementById("nombreID").value.
- Permite la captura del valor actual de un elemento disponible en la página.
- Utiliza la propiedad id para enlazar el elemento a intervenir.
- Este método funciona una vez cargada toda la página y un evento que dispare la ejecución de la sentencia o que el bloque script este después del elemento. Caso contrario no captura ningún valor.

Obtención y manipulación de valores y textos de los elementos del DOM.

Ejemplo para capturar un valor de un elemento (input tipo text).

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>JavaScript</title>
</head>
<body>
  <input type="text" id="txtValor" value="Chao Mundo">
  <script>
    alert(document.getElementById("txtValor").value);
  </script>
</body>
</html>
```



Ejemplo para cambiar un valor de un elemento (input tipo text).



Eventos básicos: onClick y onChange.

Los eventos nos permiten detectar las interacciones que realiza el usuario en nuestra página.

Los eventos requieren de una función con el respectivo código.

El código, es una rutina que realizará alguna tarea según los requerimientos del problema a resolver.

- El evento onClick se “dispara” cuando el usuario realiza un clic en algún elemento, por lo general, en un botón.
- El evento onChange, en cambio, se ejecuta cuando el usuario realiza algún cambio en un elemento de la página.

Ejemplo código onClick

The image shows a code editor on the left and a web browser on the right. The code editor contains the following HTML and JavaScript code:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>JavaScript</title>
    <script>
      function capturar(){
        let variable = document.getElementById("txtPalabra").value;
        if(variable.length > 0){
          alert("Cantidad de caracteres es: " + variable.length);
        }
      }
    </script>
  </head>
  <body>
    <input type="text" id="txtPalabra">
    <input type="button" onclick="capturar()" value="Aceptar">
  </body>
</html>
```

Two green arrows point from the code to the browser. One arrow points to the `onclick="capturar()"` attribute in the button tag, and the other points to the `let variable = document.getElementById("txtPalabra").value;` line in the JavaScript function. The browser shows a navigation bar with links to 'Aplicaciones', 'YouTube', and 'Programació'. Below the navigation bar is a text input field with the value 'hola' and an 'Aceptar' button. Below the input field is a message box that says 'Esta página dice' and 'Cantidad de caracteres es: 4', with an 'Aceptar' button.

Variables.

- Las variables permiten almacenar valores temporalmente en JS.
- Existen 3 maneras de declarar variables, usando **var**, **let** o **const**.
- Las variables definidas con **var**, tienen un alcance global dentro de una rutina a diferencia de **let** y **const**, que solamente están disponible en un bloque de código (por ejemplo, en una función).
- Tanto **var** como **let**, permiten sobre escribir el valor que contiene (variables).
- **const**, se utiliza para almacenar valores constantes o que no cambian, por ej: constantes matemáticas, astronómicas, porcentajes de impuestos, etc.
- **let** y **const** con las variables que se recomiendan para optimizar el código.

Ejemplos:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>JavaScript</title>
  <script>
    var nombre;
    const PI = 3.1415;
    const IVA= 1.19;
    function enviar()
    {
      let apellido = "Peréz";
      nombre = "Ana " + apellido;
      alert(nombre + ". PI, vale= " + PI);
    }
  </script>
</head>
<body>
  <input type="button" value="Aceptar" onClick="enviar()">
</body>
</html>
```

Esta página dice

Ana Pérez, PI, vale= 3.1415

Aceptar

Expresiones aritméticas.

Para realizar operaciones aritméticas en JavaScript, se utilizan las siguientes simbologías

Símbolo (operador)	Descripción
+	Suma.
-	Resta.
*	Multiplicación.
**	Exponente.
/	División.
%	Resto / Modulo.
++	Incremento de 1 al valor de la variable.
--	Decremento de 1 al valor de la variable.

Ejemplo de uso:

```
<script>
  let numero1 = 2;
  let numero2 = 4;
  let resultado = 0;
  resultado = numero1 + numero2;
  alert("resultado: " + resultado);
  resultado = numero1 - numero2;
  alert("resultado: " + resultado);
  resultado = numero1 * numero2;
  alert("resultado: " + resultado);
</script>
```

Sentencias condicionales.

Existe una serie de instrucciones disponible en JavaScript los cuales se resumen a continuación.

If: para especificar un bloque de código que se ejecutará, si una condición especificada es verdadera.

else: es para especificar un bloque de código que se ejecutará, si la misma condición es falsa.

else if: para especificar una nueva condición para probar, si la primera condición es falsa.

Ejemplos de uso:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>JavaScript</title>
    <script>
      function capturar(){
        let variable = document.getElementById("txtPalabra").value;
        if(variable.length > 0)
        {
          alert("Cantidad de caracteres es: " + variable.length);
        }
      }
    </script>
  </head>
  <body>
    <input type="text" id="txtPalabra">
    <input type="button" onclick="capturar()" value="Aceptar">
  </body>
</html>
```

```
<script>
  function capturar(){
    let variable = document.getElementById("txtPalabra").value;
    if(variable.length > 10)
    {
      alert("Mayor a 10 caracteres");
    }else if(variable.length > 0)
    {
      alert("Menor a 10 caracteres");
    }
    else
    {
      alert("No tiene texto para contar");
    }
  }
</script>
```


Funciones.

Las funciones son un conjunto de sentencias agrupadas bajo un nombre, que realizan cálculos o tareas específicas. También son conocidas como subprograma. Las funciones ejecutarán las sentencias solo cuando sean llamadas o invocadas. Las funciones distinguen entre mayúsculas y minúsculas, lo que influye en la asignación del nombre y llamada (deben ser iguales). Permiten parámetros o valores de entradas para ser usados en los cálculos internos de la función. La palabra reservada para declarar una función es **function**.

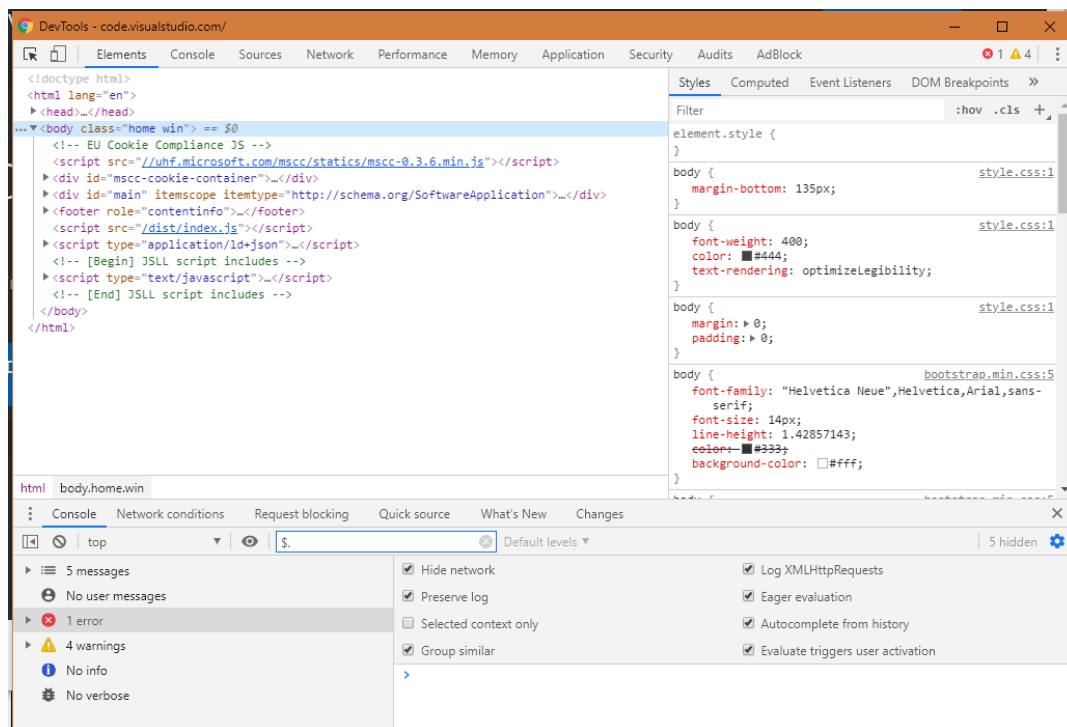


Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>JavaScript</title>
    <script>
      // declaración de función
      function saludo1() {
        alert("Hola");
      }
      // declaración de función con parámetros
      function saludo2(mensaje, nombre) {
        alert(mensaje + " " + nombre);
      }
      // llamada de la función (ejecución)
      saludo1();
      saludo2("Adiós", "Eren");
    </script>
  </head>
  <body>
  </body>
</html>
```

Cómo ejecutar código JavaScript en la consola.

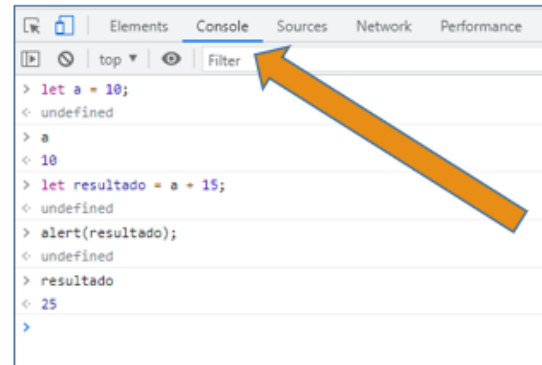
Al utilizar Chrome como navegador, el inspector aparece cuando se presiona F12 o con el botón derecho del mouse, la opción inspeccionar, la cual permite abrir DevTools.



- Seleccionar la opción “Console”, para ingresar a la consola de JS.
- La consola se utiliza principalmente para probar las rutinas que se están desarrollando y si estas, funcionan de forma adecuada.
- La consola permite escribir código JS directamente o mediante el comando `console.log()`.

Ejemplo:

```
<script>
  let numero = 2;
  numero = numero + 10;
  console.log("El resultado es: " + numero);
</script>
```



Depurando el código JavaScript con la consola.

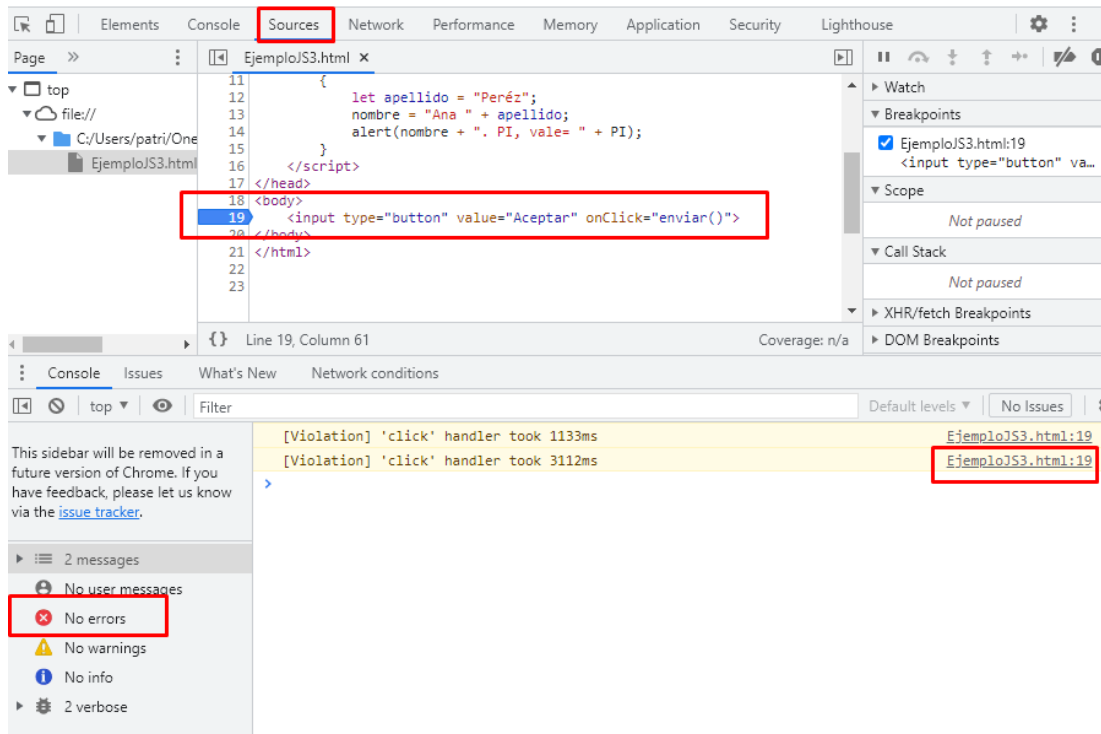
La depuración permite revisar la ejecución del código línea a línea para buscar algún error que pueda estar ocurriendo en nuestro código.

La depuración se activa en la opción “Sources” de DevTools (usando Chrome), el cual, mostrará:

- Cantidad de errores.
- Errores de sintaxis.
- Advertencias o sugerencias (no son errores).
- Pausar el código según la o las líneas que se marquen (break point).
- Descripción del error que se ha provocado y línea que la que se encuentra.
- Controles para avance de ejecución paso a paso, ejecución normal, ingresar a una función, entre otras.



Ejemplo de opción:



Ejercicios.

Parte 1:

- Crear una calculadora que realice las operaciones básicas (+, -, *, /).
- Usar funciones para cada una de las operaciones.
- Verificar que los text contengan número que sumar.
- Restringir que solo se permitan números en las cajas de texto a utilizar.

Partes 2:

- Solicitar nombre y restringir la cantidad y los caracteres que puede utilizar el usuario.
- Mostrar el nombre, operación realizada y el resultado en un mensaje.

jQuery Básico.

La biblioteca jQuery, ¿Por qué y cuándo usarla?

jQuery es una librería de JavaScript, creada en 2006, desarrollada principalmente para simplificar la programación de JavaScript.

Permite manipular elementos de una página, animarlos, integrar AJAX y manejar eventos, lo que permite mejorar la experiencia de usuario.

Permite manejar CSS.

jQuery es una librería libre y de código abierto.

Soporta la separación del código del documento HTML (archivo externo).

Se recomienda su uso cuando se requiere crear páginas web dinámicas y que requieran la creación o integración de librerías de terceros.



Obtener JQuery. Incluir y usarlo en un sitio.

Utilizar un CDN valido (Google Hosted Libraries por ejemplo). Acceder a la URL:
<https://code.jquery.com/>



Clic en la opción señalada en la imagen.

La URL oficial de descarga de jQuery: <https://jquery.com/>

En la actualidad la versión disponible es: 3.6.0 :

<https://code.jquery.com/jquery-3.6.0.min.js>

o del Google Hosted Libraries:

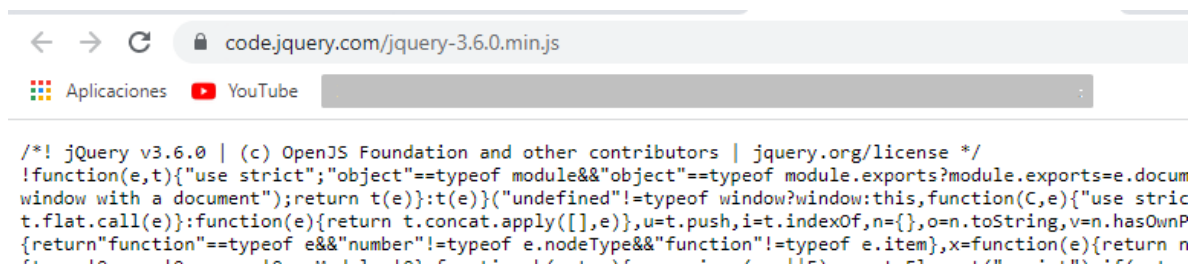
<https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js>



Existen 2 formas de integrar jQuery en un sitio web:

1. Bajando la librería y agregarlo al proyecto.

Descargarlo de la URL <https://code.jquery.com/jquery-3.6.0.min.js>, y guardar el código en la misma carpeta que el documento HTML.



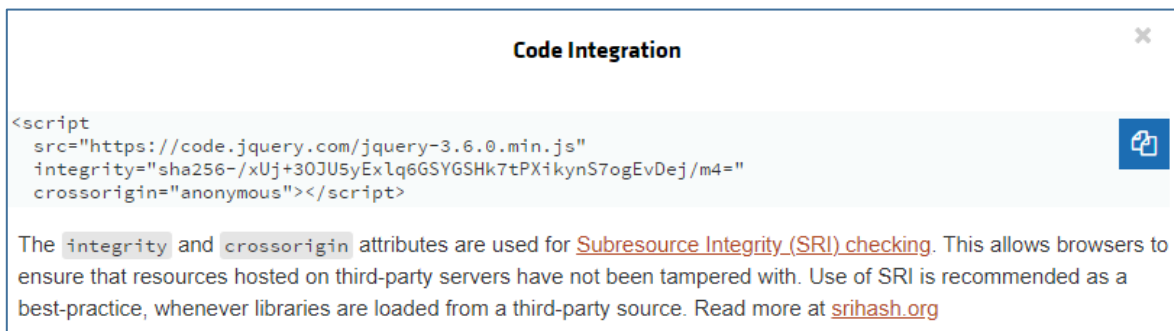
2. Bajando la librería y agregarlo al proyecto.

En el documento HTML, en la sección HEAD, agregar el siguiente código:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>jQuery</title>
  <script src="jquery-3.6.0.min.js"></script>
</head>
<body>

</body>
</html>
```

Aparecerá la siguiente ventana:



Copiar el código y pegarlo en la sección **HEAD** del documento HTML. Notar que el código es similar, solo cambia la propiedad **SRC** que señala la url desde donde se **descargará** el núcleo jQuery.

Resultado final del documento HTML

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
  <title>jQuery</title>  
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"  
    integrity="sha256-/xUj+30JU5yExlq6GSGYSHk7tPXikynS7ogEvDej/m4="  
    crossorigin="anonymous"></script>  
</head>  
<body>  
  
</body>  
</html>
```


¿Cómo usar jQuery?

Para usar la librería jQuery, siempre se debe integrar el núcleo, como fue descrito en el punto anterior.

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"  
    integrity="sha256-/xUj+30JU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=""  
    crossorigin="anonymous">  
</script>
```

Declarar una sección **dentro** del documento HTML con los tags **<script>** o también, se puede crear un archivo con extensión **JS (recomendado)**. En cualquiera de los dos casos anteriores, se debe declarar el siguiente bloque:

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
    <title>jQuery</title>  
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"  
        integrity="sha256-/xUj+30JU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=""  
        crossorigin="anonymous">  
    </script>  
    <script>  
        $(function()  
        {  
            // bloque de código para jQuery  
        })  
    </script>  
</head>  
<body>  
</body>  
</html>
```

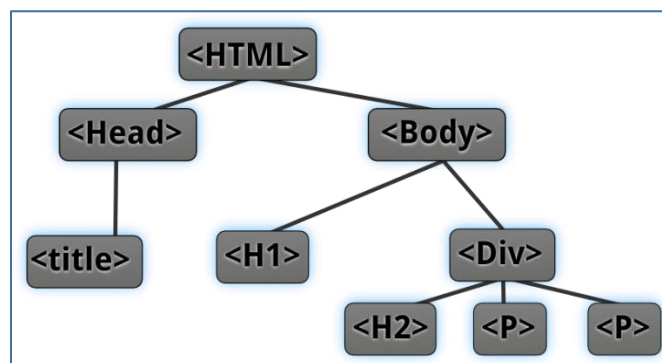
Ejemplo 2:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>jQuery</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"
    integrity="sha256-/xUj+30JU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
    crossorigin="anonymous">
  </script>
  <script>
    $(function()
    {
      alert("Hola Mundo");
    })
  </script>
</head>
<body>
</body>
</html>
```

¿Qué es el DOM?. Manipulación de elementos del DOM con JQuery.

El DOM son las siglas de Modelo de Objetos de Documento y es la representación jerárquica de los objetos existentes o que conforman una página web.

jQuery como JS, accede a los distintos elementos de una página, por medio del DOM.



Para afectar algún elemento de la página, se debe usar el **selector** más la propiedad **id** o **class** del elemento. Ejemplo de selector:



```
<script>
$(function()
{
    $(".btnSaludos").click(function()
    {
        alert("Hola");
    })
});
</script>
```

Eventos, tipos de evento, cómo interactuar con ellos.

Los eventos permiten modificar el comportamiento de una página mediante la detección de las acciones que un usuario realiza sobre la página.

Ejemplo de acción son:

- Clic en un botón, link, imagen u otro elemento.
- Movimientos del mouse.
- Presionar teclado.
- Presionar teclas del mouse.
- Entre otros.



jQuery proporciona una serie de eventos que se pueden asociar a los distintos elementos del documento HTML.

En jQuery se tienen disponible una serie de eventos que permiten detectar la acción realizada por el usuario, con el fin de asociar rutinas de programación que realicen alguna tarea en especial.

Eventos del Mouse:

- click
- dblclick
- mousedown
- mouseleave
- mouseup
- mouseover
- mouseout

Eventos del Teclado:

- Keydown
- keypress
- keyup

Para más eventos, visitar documentación oficial: <https://api.jquery.com/category/events/>

Ejemplo 1:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>jQuery</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"
    integrity="sha256-/xUj+30JU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
    crossorigin="anonymous">
  </script>
  <script>
    $(function()
    {
      $(".btnAceptar").click(function()
      {
        alert("Hola Mundo");
      })
    })
  </script>
</head>
<body>
  <input type="button" class="btnAceptar" value="Aceptar">
</body>
</html>
```

Ejemplo 2:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>jQuery</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"
    integrity="sha256-/xUj+30JU5yEx1q6GSYGS7k7tPXikynS7ogEvDej/m4="
    crossorigin="anonymous">
  </script>
  <script>
    $(function()
    {
      $(".btnAceptar").click(function()
      {
        let nombre = $('#txtNombre').val();
        alert("Hola " + nombre);
      })
    })
  </script>
</head>
<body>
  <input type="text" class="txtNombre" placeholder="Ingrese su nombre">
  <input type="button" class="btnAceptar" value="Aceptar">
</body>
</html>
```

Ejemplo 3: (verifique resultado).

```
<body>
  <input type="text" class="txtNumero1" placeholder="Ingrese 1er número">
  <input type="text" class="txtNumero2" placeholder="Ingrese 2do número">
  <input type="button" class="btnAceptar" value="Aceptar">
</body>
```

```
<script>
  $(function()
  {
    $(".btnAceptar").click(function()
    {
      let numero1 = $('#txtNumero1').val();
      let numero2 = $('#txtNumero2').val();
      let resultado = numero1 + numero2;
      alert("El resultado es: " + resultado);
    })
  });
</script>
```

Ejemplo con resultado correcto:

```
$(".btnAceptar").click(function()  
{  
    let numero1 = $('#txtNumero1').val();  
    let numero2 = $('#txtNumero2').val();  
    let resultado = Number(numero1) + Number(numero2);  
    alert("El resultado es: " + nombre);  
})
```

La función **Number**, cambia el tipo de dato de texto a un numérico, si el texto en cuestión representa a un número válido. Caso contrario, devolverá un **NaN**.

¿Qué es y cuándo usar un plugin?

Los plugins, son librerías de terceros o creados por la comunidad que permiten extender la funcionalidad de Bootstrap con jQuery.

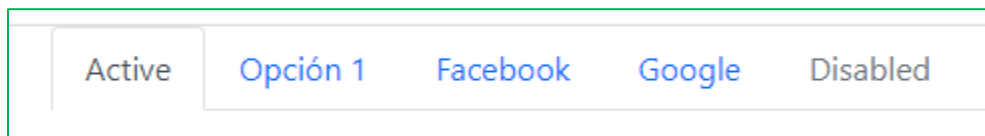
- Permiten enriquecer la interacción de los usuarios.
- Mejora en la interfaz gráfica.
- Mayor atractivo para usar el sitio.
- En la última versión de Bootstrap, se encuentran como componentes y ya no depende de jQuery, sino que solo usa JS de forma nativa (vanilla).

URL <https://getbootstrap.com/docs/5.1/components>.

También existen otros plugins que entregan otros tipos de elementos, como, por ejemplo, gráficos, tablas con filtros, mapas y otros tipos de funcionalidades.

Ejemplo de uso de tab.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>JavaScript</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js" integrity="sha384-7tRN421sk31zLzdhm1Kq3zoz6pp9m6D19o7Z12iWtdq6v42f2+f7X7zGz+gt8" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js" integrity="sha384-ke7885Ms8hKv1Py9U2aoC9PP6h210kJ782tu6AtD9P0tWtleIun6F3/v0m1FXpq3" crossorigin="anonymous"></script>
</head>
<body>
  <div class="container">
    <ul class="nav nav-tabs">
      <li class="nav-item">
        <a class="nav-link active" aria-current="page" href="#">Active</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Opción 1</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="https://www.facebook.com/">Facebook</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="https://www.google.cl/">Google</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled">Disabled</a>
      </li>
    </ul>
  </div>
</body>
</html>
```



Ejemplos de plugins Bootstrap-jQuery más comunes.

Las funcionalidades que podemos encontrar son:

- Modal.
- DropDown.
- Scrollspy.
- Tab.
- Tooltip.
- Button.
- Collapse.
- Carousel.
- Entre otros.

Igual considerar dado que existen páginas que utilizan versiones anteriores a la actual (5.x).

Ejercicios

- Crear una calculadora básica, pero usando jQuery.
- Buscar 5 funciones integradas o build-in para ser usadas.
- Utilice 3 de estas funciones.
- Buscar 3 librerías de tercero no pertenecientes a Bootstrap o jQuery que permitan facilitar la implementación de efectos visuales a la información o imágenes a mostrar.
- Buscar otros plugin o componentes e implementarlo en una página web.

Fundamentos de Git / GitHub

Necesidad de un repositorio de código fuente.

La necesidad de un repositorio de código fuente se produce cuando:

- El producto desarrollado o por desarrollar consta de una gran cantidad de archivos y configuraciones.
- El equipo de desarrollo es muy grande.
- Distribución del proyecto que se está desarrollando a los distintos participantes del equipo.
- Unión de diferentes módulos o partes del producto que desarrollan los participantes del equipo.



Instalación, configuración y comandos básicos.

Para instalar GIT, se debe descargar el ejecutable desde la siguiente URL:

<http://git-scm.com/download/win>

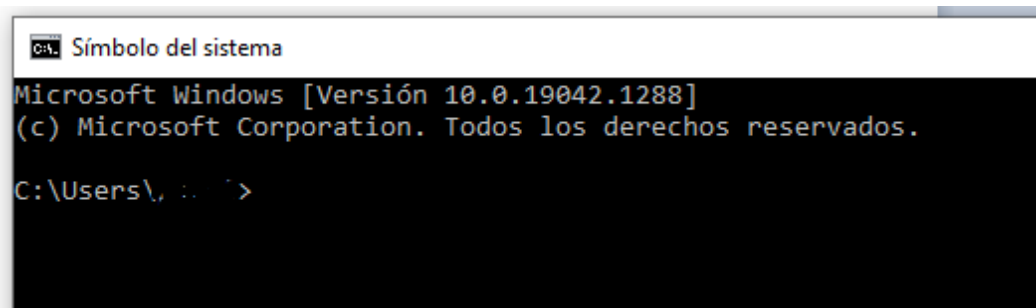
Descargar la versión según sistema operativo a utilizar. Ejecutar el archivo descargado y esperar la instalación.



Luego debes establecer nombre de usuario y correo. Para esto, requieres ejecutar algunos de los siguientes comandos en la consola.

Para acceder a la consola presiona la tecla de Windows y Símbolo de sistema.

Aparecerá la consola como muestra la siguiente imagen:



Escribir los siguientes comandos:

```
>git config --global user.name "Maria Ignacia"
```

```
>git config --global user.email mimail@servidor.com
```

Esta información es para asociar todas las actualizaciones de forma global a el usuario especificado.

Commits y restauración de archivos.

Los Commits son la confirmación de los cambios realizados sobre los archivos.

Para realizar esta confirmación, se utiliza el siguiente comando:

```
>git commit -m "cambios realizados con éxito"
```

Este commit creara una instantánea del proyecto el cual se puede usar para comparar o volver a la versión previa.

Si no todos los archivos fueron subidos, ejecutar el siguiente comando previamente:

```
>git add .
```

Para ver un listado de los commit del más nuevo al más antiguo utilizar:

```
>git log
```

Para restaurar o volver a una versión x de un archivo, se debe conocer el código del commit.

Para ver el listado, usar

```
>git log
```

Identificar el código o id para obtener la versión guardada y utilizar el siguiente comando:

```
>git checkout a1fefbA
```

Donde a1fefbA, es el id registrado en algún commit realizado previamente.

Para volver al estado actual del proyecto, ejecutar:

```
>git checkout master
```

Cambios de nombres.

Para cambiar nombre o renombrar un archivo, utilizar el siguiente comando:

```
>git mv nombreOriginal.html nombreNuevo.html
```

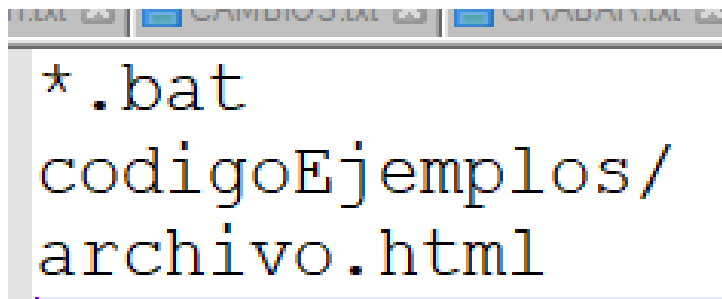
Ignorando archivos.

Para ahorrar tiempo en subida o descarga de los archivos, se puede crear un archivo de configuración que permitirá ignorar a los archivos o carpetas nombrados en el.

Los pasos para realizar esta configuración son:

- 1.- Crear un archivo en la carpeta raíz del proyecto con nombre: .gitignore
Puede utilizar algún editor, por ejemplo: NotePad, VS Code o similar.
- 2.- Escribir los archivos y/o carpetas que necesitemos ignorar y guardar cambios.

Ejemplo de archivos:



```
*.bat  
codigoEjemplos/  
archivo.html
```

Ramas, uniones, conflictos y tags.

Las ramas, permiten realizar trabajos diferentes sobre la misma base, pero no afectando la base desde donde se creó. Un uso, por ejemplo, es probar una librería nueva que se quiere agregar al proyecto.

El atributo de GIT, es que su creación es rápida con respecto a otros sistema de respaldo/repositorio.

Los siguientes comandos, permiten:

Crear una nueva rama:

```
>git branch nuevaRama
```

Mostrar la rama actual:

```
>git branch
```

Para cambiarnos de rama:

```
>git checkout nombreDeLaRama
```

Eliminar la rama señalada:

```
>git branch -d nombreRamaParaBorrar
```

Existe la posibilidad de unir 2 ramas. Para esto, se debe utilizar el siguiente comando:

```
>git merge ramaParaUnirConMaster
```

Los conflictos surgen cuando se intenta unir 2 ramas y sus respectivos archivos. En el caso de que 2 archivos no tengan el mismo contenido, GIT, no resuelve el conflicto de forma automática.

Para resolver, se deben abrir los archivos que están en conflicto y dejar la porción de código idénticos en ambos.

Git entrega un reporte señalando las líneas de contenido o códigos que tienen conflictos.

Ejemplo de conflicto y su solución (ambos archivos tendrán el mismo código).

```
<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=====
<div id="footer">
  please contact us at support@github.com
</div>
>>>>>> iss53:index.html
```



```
<div id="footer">
  please contact us at email.support@github.com
</div>
```

Los proyectos pueden ser etiquetados (tags), para marcar los avances importantes o de liberación de una versión para producción.

Para crear una etiqueta, se puede utilizar los siguientes códigos:

```
>git tag v1.0.12
```

```
>git tag v12312 -m "Primera versión producción"
```

Para obtener el listado de etiquetas registradas y su información, utilizar:

```
>git tag -l 'versionMinimaParaVer'
```

```
>git show nombreTag
```


Stash y Rebase

Stash, permite crear un directorio limpio cuando se requiere volver a un punto anterior del proyecto, sin eliminar los cambios realizados posterior al **commit**.

Estos cambios no deben ser confirmados para que al momento de volver al punto anterior y regresar al trabajo (cambios posteriores) no se vean afectados.

El comando para realizar esta acción es:

```
>git stash
```

Para volver al trabajo incompleto por el cambio a la versión anterior, se utiliza:

```
>git stash pop
```

Los **stash** realizados, generan un historial, el cual puedes acceder a ver con el comando:

```
>git stash list
```

- a. Rebase, permite mover una rama completa a un extremo de otra rama, formando una sola rama extensa.
- b. Reescritura del historial al fusionar los commit de ambas ramas.
- c. El historial es lineal, permitiendo una ir directo al inicio sin desviaciones.
- d. La navegación es más sencilla dentro del proyecto.
- e. Tiene como desventaja la perdida de trazabilidad y seguridad.

Para reorganizar una rama se debe posicionar en la rama a fusionar y ejecutar lo siguiente:

```
>git checkout ramaParaFusionar
```

```
>git rebase master
```

Fundamentos de GitHub

Repositorios remotos, Push y Pull.

Los repositorios remotos son servidores que están disponibles para alojar repositorios GIT. Para este curso, se utilizará GitHub como proveedor de alojamiento git.

Para iniciar el uso de repositorio remoto en GitHub, se debe crear una cuenta en la URL:

<https://github.com> y seguir los pasos solicitados en dicha página.

Push se utiliza para subir los commits o confirmaciones de cambios al repositorio remotos en la rama actual local.

Pull se utilizar para descargar las actualizaciones disponibles en el repositorio remoto y las fusiona con el trabajo local. Como recomendación, usar commits antes de usar pull.

Ejemplos:

```
>git push usuario@host:/ruta/repositorio miRama
```

```
>git pull usuario@host:/ruta/repositorio miRama
```

Fetch v/s Pull.

Fetch permite recuperar un trabajo nuevo realizado por otras personas, donde se obtiene todas las etiquetas y ramas de seguimiento remoto **sin fusionar** estos cambios en tus propias ramas.

A diferencia del **Pull** que realiza la fusión de ambos trabajos.

Ejemplo Fetch:

```
>git fetch usuario@host:/ruta/repositorio
```

Clonando un repositorio.

Clonar un repositorio es copiar tu repositorio remoto a uno local, con la finalidad de sincronizarlos.

Ejemplo:

```
>git clone usuario@host:/ruta/repositorio
```

Salida esperada:

```
$ git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY  
> Cloning into `Spoon-Knife`...  
> remote: Counting objects: 10, done.  
> remote: Compressing objects: 100% (8/8), done.  
> remote: Total 10 (delta 1), reused 10 (delta 1)  
> Unpacking objects: 100% (10/10), done.
```

Documentando un proyecto con Markdown.

Markdown es una forma de estilo de texto en la web.

Controla la visualización del documento como por ejemplo:

1. Formatear palabras como negrita o cursiva.
2. Agregar imágenes.
3. Crear listas.
4. Entre otros.

La sintaxis de toda la escritura y formatos se pueden encontrar en la documentación oficial de GitHub. La url es:

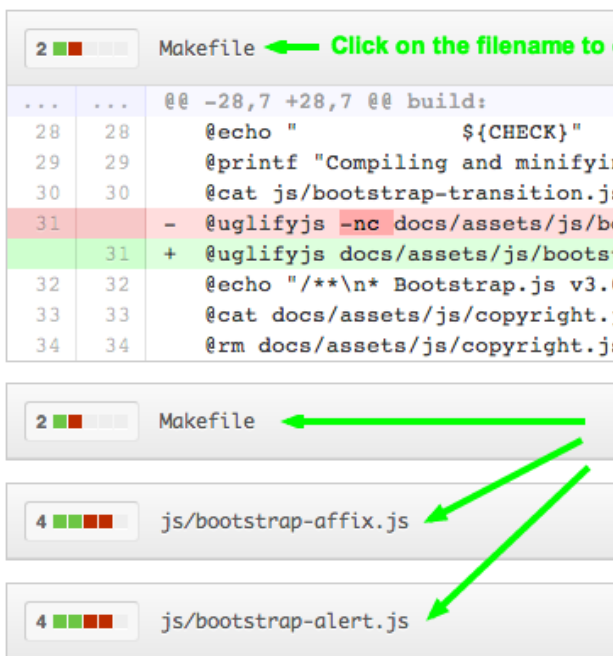
<https://help.github.com/es/github/writing-on-github/basic-writing-and-formatting-syntax>

Administrando Pull Request.

Un Pull Request es una solicitud realizada al repositorio cuando se envía nuevos códigos con cambios y donde el encargado o dueño del repositorio, se encarga de revisar este código y decide si es apto para ser integrado.

Esta funcionalidad promueve la colaboración de los distintos integrantes en un proyecto. También es un foro para debatir, publicar bug o errores, entregar feedback, modificar el código propuesto, entre otras tareas.

Los Pull Request, permiten comparar los códigos del archivo original con la propuesta realizada, mostrando en forma gráfica el nuevo código o sus modificaciones.



```

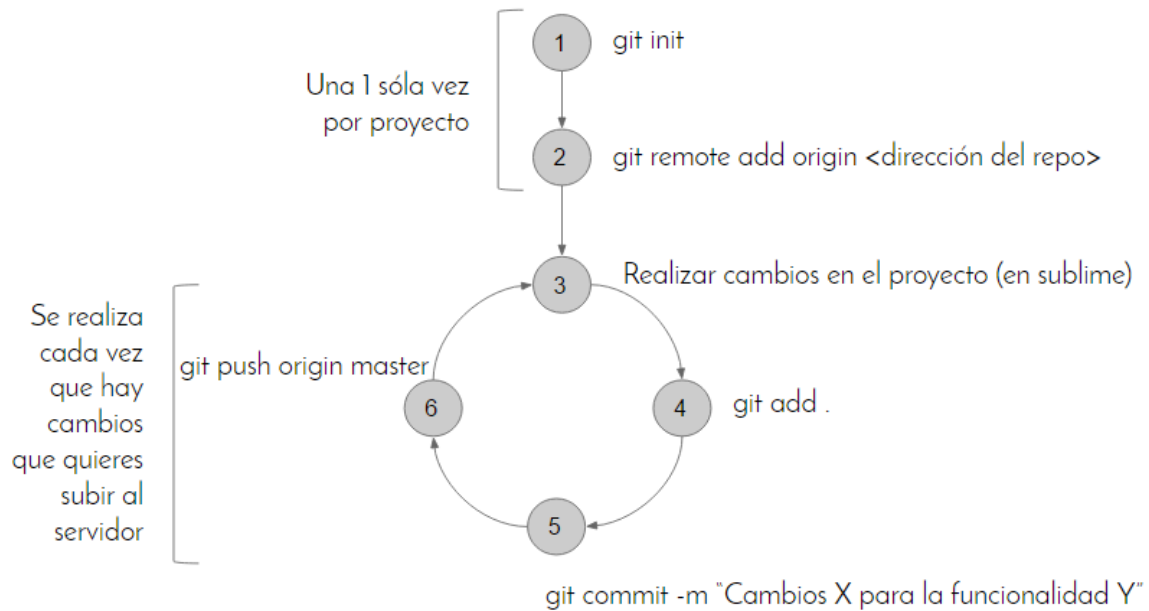
2 █ █ █ █ █ Makefile ← Click on the filename to
...  ... @@ -28,7 +28,7 @@ build:
28 28  @echo "                ${CHECK}"
29 29  @printf "Compiling and minifyi
30 30  @cat js/bootstrap-transition.js
31 31  - @uglifyjs -nc docs/assets/js/b
31 31  + @uglifyjs docs/assets/js/boots
32 32  @echo "/*\n* Bootstrap.js v3.0
33 33  @cat docs/assets/js/copyright.
34 34  @rm docs/assets/js/copyright.js

2 █ █ █ █ █ Makefile
4 █ █ █ █ █ js/bootstrap-affix.js
4 █ █ █ █ █ js/bootstrap-alert.js

```

Flujos de trabajo con GitHub.

A continuación, se muestra un diagrama con el flujo de trabajo sugerido cuando se utiliza GitHub como repositorio de los proyectos desarrollados.



Ejercicios

- Instalar Git y crear repositorio.
- Ingresar al sitio de GitHub, registrarse y crear un repositorio con un documento.
- Clone el repositorio creado.
- Registre los cambios presentados.

BIBLIOGRAFÍA

HTML

<https://www.w3schools.com/html/default.asp>

<https://developer.mozilla.org/en-US/docs/Glossary/HTML5>

CSS

<https://www.w3schools.com/css/default.asp>

<https://developer.mozilla.org/en-US/docs/Web/CSS>

JavaScript

<https://www.w3schools.com/js/default.asp>

<https://developer.mozilla.org/en-US/docs/Web/JavaScript>

jQuery

<https://api.jquery.com/>

<https://www.w3schools.com/jquery/default.asp>

Bootstrap

<https://getbootstrap.com/docs/4.6/getting-started/introduction/>

<https://getbootstrap.com/docs/5.1/getting-started/introduction/>

https://www.w3schools.com/bootstrap/bootstrap_ver.asp

Git

<https://www.w3schools.com/git/default.asp>

<https://git-scm.com/doc>

GitHub

<https://github.com/>

GLOSARIO

- **HTML:** HyperText Markup Language o Lenguaje de marcado de hipertexto, permite crear páginas webs simples.
- **JS/JavaScript:** Lenguaje de programación del lado del cliente.
- **DevTools:** Herramienta multifuncional para el desarrollo de páginas web.
- **Framework:** Conjunto de herramientas o tecnologías que operan de cierta forma y que permiten agilizar el desarrollo de aplicaciones.
- **Bootstrap:** Framework que dispone de diferentes componentes web y que facilitan la creación de páginas web.
- **jQuery:** librería, basado en JavaScript para simplificar la sintaxis de esta.
- **CSS:** Cascade style sheet o hoja de estilo en cascada, permite dar mayor atractivo visual a las páginas web creadas con HTML.
- **Git:** Aplicación que permite respaldar, versionar el Código fuente de uno o varios proyectos, además permite la colaboración de distintas personas que intervienen en el proyecto.
- **GitHub:** aplicación web que provee de un entorno similar a Git.
- **Navegador:** aplicación que permite realizar peticiones a un servidor web.
- **FullStack:** desarrollador que cumple con varios roles dentro de un proyecto(back-end y Front-end), utiliza distintas herramientas para agilizar el desarrollo de una pagina web o programa.
- **W3C:** consorcio internacional que vela por la estandarización de la internet y su buen funcionamiento a largo plazo.