



Laboratorio I - 2024



Clase Teórica 09

Docente: Myriam Ruiz

Licenciada en Informática
Profesora en Computación y Matemática
Programadora Universitaria



Iteraciones

- Hay dos tipos de iteraciones:
 - 1) **Determinadas**: Las que **conocemos** desde el principio **cuántas veces se repetirá**
 - 2) **Indeterminadas**: Las que **no conocemos** desde el principio **cuántas veces se repetirá**



Iteraciones: Elegir HACER o MIENTRAS?

No sabemos cantidad

Comer un Yogurt



Usar MIENTRAS

Si sabemos cantidad

Hacer una serie de ejercicios en el Gym



Usar HACER

Javascript: Iteraciones - for

Hacer

```
HACER n VECES  
  Acciones;  
Fin_Hacer
```

for

```
for (inicialización; condición; paso) {  
  Acciones;  
}
```

Javascript: Iteraciones - for

```
for (inicialización; condición; paso) {  
    Acciones;  
}
```

- **inicialización**: se coloca la variable con la que se va a llevar la cantidad de repeticiones. Se ejecuta solamente al comenzar la primera iteración del bucle.
- **condición**: establece lo que debe cumplir la variable para continuar la ejecución. Se evaluará cada vez que comience una iteración del bucle. Cuando deja de cumplirse, finaliza la estructura.
- **paso**: indica cómo se modificará el valor de la variable, cada vez que termina la iteración del bucle, antes de comprobar si se debe seguir ejecutando.

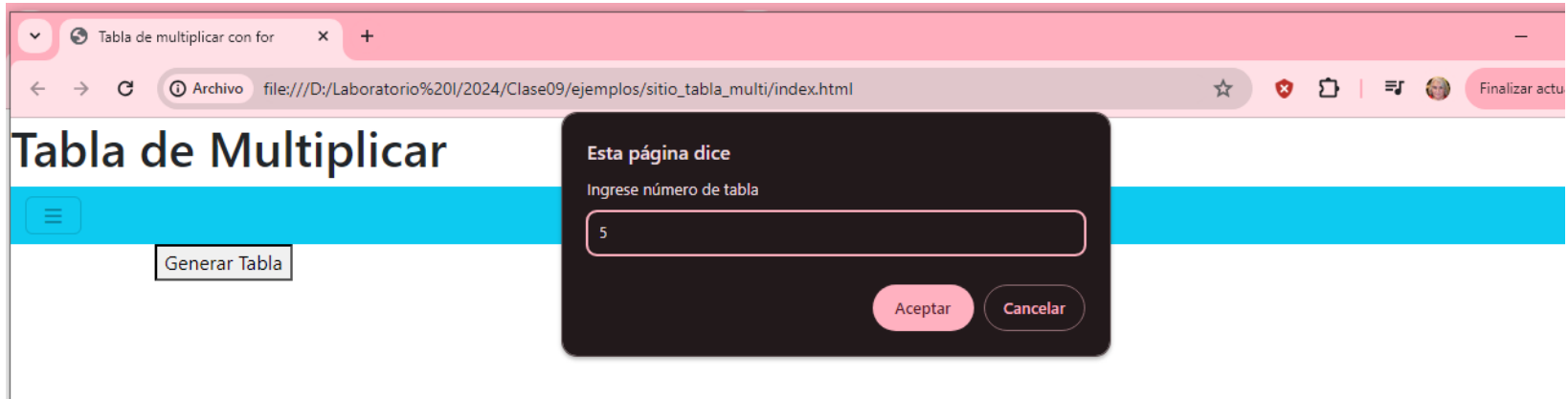
Javascript: Iteraciones - for

```
for (inicialización; condición; paso) {  
    Acciones;  
}
```

- Es común ver en el **paso** la expresión $i++$, lo que es lo mismo que $i = i + 1$
- Podría tener también $i--$, si el paso retrocediera

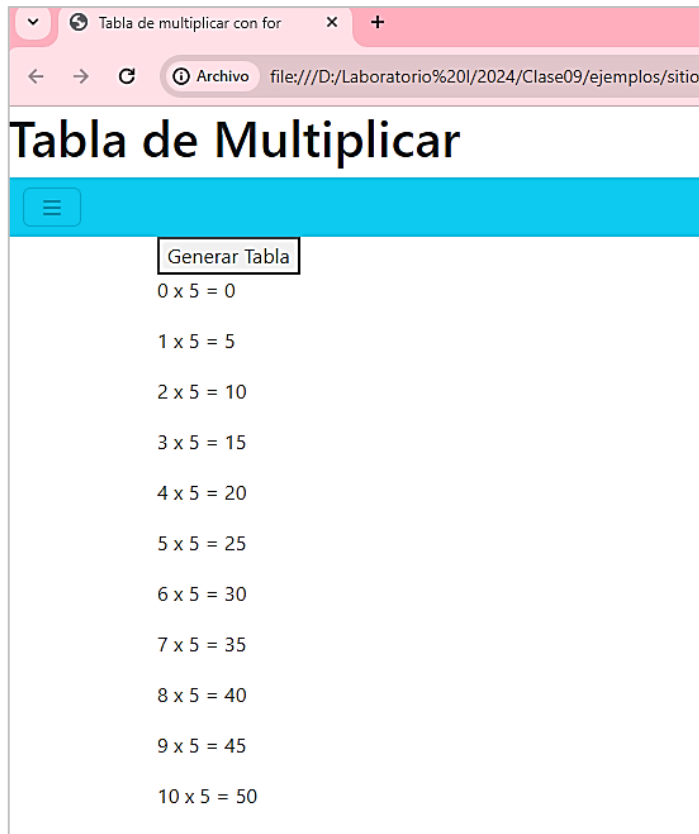
Javascript: Iteraciones - for

- Ejemplo: Realizar un script que permita introducir el número de una tabla de multiplicar, para generarla, entre 0 y 10.



Javascript: Iteraciones - for

- Ejemplo:



Javascript: Iteraciones - for

- Ejemplo: La tabla se mostrará dentro del article con id ="contenedor-tabla"

```
<section class="container">
  <main>
    <section>
      <button type="button" id="generar">Generar Tabla</button>
      <article id="contenedor-tabla">

      </article>
    </section>
  </main>
</section>
<script src="bootstrap-5.3.3-dist/js/bootstrap.min.js"></script>
<script src="js/script-tabla.js"></script>
```

Javascript: Iteraciones - for

- Ejemplo:

```
let boton = document.getElementById('generar');


function generarTabla ()
{
    let contenedor = document.getElementById('contenedor-tabla');
    let nroTabla = parseInt(prompt('Ingrese número de tabla'));
    let resultado = 0;
    for (let i = 0; i < 11; i++) {
        resultado = i * nroTabla;
        contenedor.innerHTML = contenedor.innerHTML +
            `

${i} x ${nroTabla} = ${resultado}</p>`;
    }
}

boton.addEventListener('click', generarTabla);


```

Concatenamos
(unimos) lo que ya
había con el nuevo
párrafo, para no
perder lo anterior



Javascript: Iteraciones - for

- Ejemplo la parte del mostrado se puede hacer de dos formas:
 - Incorporando el texto y la variable dentro de comillas invertidas, encerrando la variable con los símbolos `${}`:
` texto de ejemplo `${variable}` más texto de ejemplo `
 - Concatenando con el signo más, texto entre comillas y variables :
'texto de ejemplo ' + variable + ' más texto de ejemplo '

Comparativa de ambas notaciones:

```
`<p> ${i} x ${nroTabla} = ${resultado}</p>`;
'<p>' + i + ' x ' + nroTabla + ' = ' + resultado + '</p>';
```

Javascript: Concepto intuitivo de arreglo

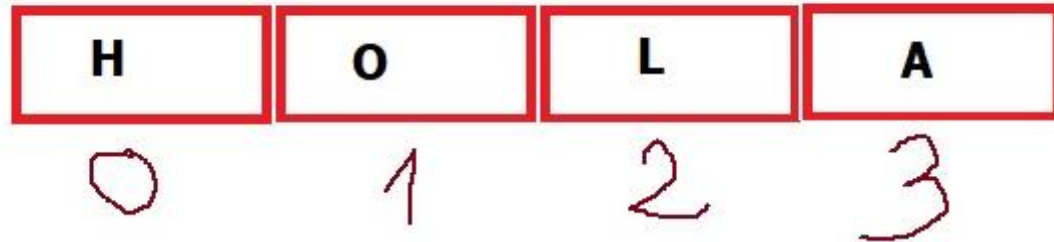
- En muchos lenguajes aparece el concepto de arreglo, que es una forma de almacenar datos. Pero que es en sí?
- Para entenderlo, podemos imaginarlo como un alhajero, un pastillero o una caja de herramientas, que sirven para guardar varias cosas en un mismo lugar, para mantenerlas juntas y organizadas, pudiéndolas sacar y poner fácilmente



Javascript: Concepto intuitivo de arreglo

- En muchos lenguajes, el texto se considera un arreglo de caracteres.
- Utilizando **un único nombre de variable**, se puede acceder a cada uno de los compartimentos o **componentes**, mediante el **índice**, que a manera ilustrativa se observa en la parte inferior (inicia de cero)

saludo



Javascript: Concepto intuitivo de arreglo

- En la consola del navegador se puede observa como un texto se puede analizar componente a componente, mediante un índice, e incluso saber cuántas componentes tiene.

```
> let saludo = 'Hola';  
< undefined  
> saludo[0]  
< 'H'  
> saludo[3]  
< 'a'  
> saludo.length  
< 4
```

Javascript: `querySelectorAll()`

- Hasta ahora, si queríamos seleccionar algo, utilizábamos `getElementById('un-id')`. Pero esta forma permite seleccionar de a un elemento.
- También vimos que podíamos utilizar `getElementsByClassName()`, que seleccionaba **todos** los elementos que tenían una misma clase.
- `querySelectorAll()` es una forma de seleccionar **todo** lo que le pongamos, que coincida con un tipo selector CSS, es decir, etiquetas, clases, pseudoclases, selector de hijo, selector descendente, etc.

Javascript: `querySelectorAll()`

- `querySelectorAll()` genera como resultado un arreglo de elementos que cumplan con pertenecer a el/los selectores requeridos.
- Una vez seleccionados, se puede acceder a cada uno a través de un índice, y a partir de ahí, acceder a sus propiedades para leerlas o modificarlas, a sus eventos, etc.

Javascript: querySelectorAll()

- Ejemplo: Tenemos una página con 2 párrafos en el main y uno en el footer, haremos un script que seleccione sólo los párrafos de main y luego todos los párrafos, y veremos como lo accede JS.

Hola mundo

Título 2 - Primero

Ejemplo 1 párrafo

Título 2 - Segundo

Ejemplo 2 párrafo

Párrafo en el footer

```
<main>
  <section>
    <article>
      <h2>Título 2 - Primero</h2>
      <p>Ejemplo 1 párrafo</p>
    </article>
    <article>
      <h2>Título 2 - Segundo</h2>
      <p>Ejemplo 2 párrafo</p>
    </article>
  </section>
</main>
<footer>
  <p>Párrafo en el footer</p>
</footer>
```

Javascript: querySelectorAll()

- Continúa... Observar lo que muestra cada instrucción

```
let parrafos = document.querySelectorAll('main p'); /* párrafos en main */
let parraf = document.querySelectorAll('p'); /* todos los párrafos */
window.alert(parrafos); ①
window.alert('En main hay: ' + parrafos.length + ' párrafos'); ②
window.alert(parrafos[0]); ③ ④
window.alert(parrafos[0].innerHTML); /* muestra lo que hay entre <p> y </p> */
//parrafos[1].style.color = '#FF0000'; /* cambia color a 2do párrafo */
window.alert('Total de párrafos de la página: ' + parraf.length); ⑤
```

[object NodeList]

①

☐ No permitir que este sitio vuelva a preguntar

Aceptar

En main hay: 2 párrafos

②

☐ No permitir que este sitio vuelva a preguntar

Ejemplo 1 párrafo

④

☐ No permitir que este sitio vuelva a preguntar

[object HTMLParagraphElement]

③

☐ No permitir que este sitio vuelva a preguntar

Total de párrafos de la página: 3

⑤

☐ No permitir que este sitio vuelva a preguntar

Javascript: querySelectorAll()

- Si hacemos la intrucción :

```
parrafos[1].style.color = '#FF0000'; /* cambia color a 2do párrafo */
```

Hola mundo

Título 2 - Primero

Ejemplo 1 párrafo

Título 2 - Segundo

Ejemplo 2 párrafo


Párrafo en el footer

Javascript: Iteraciones - for

Ejemplo: Seleccionar todos los botones que tienen la clase .btn y recorrerlos, para asignarles un addEventListener, para que cuando le hagan click reaccionen con una función que muestre a cual le hicieron click

```
const botones = document.querySelectorAll('.btn');

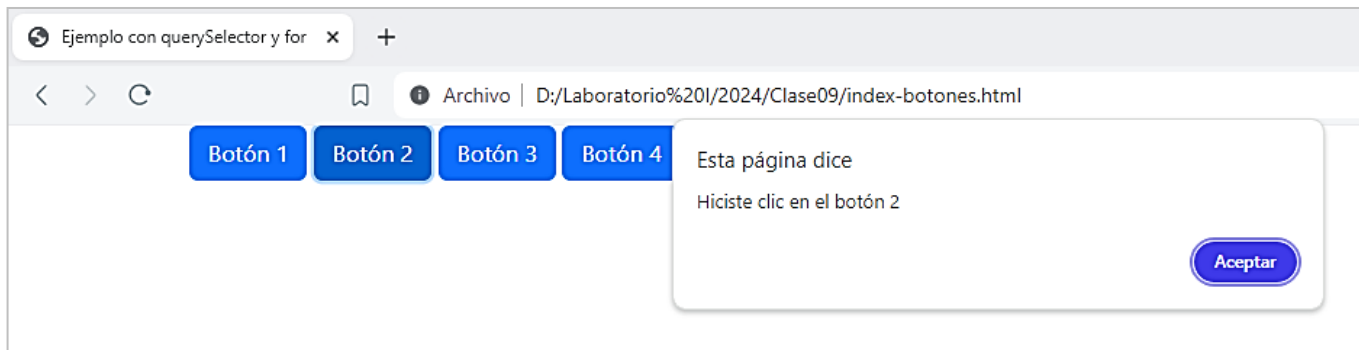
// Itera sobre los botones y añade un manejador de eventos a cada uno
for (let i = 0; i < botones.length; i++) {
  botones[i].addEventListener('click', function() {
    alert(`Hiciste clic en el botón ${i + 1}`);
  });
}
```



Función anónima (sin nombre)

Javascript: Iteraciones - for

Ejemplo:



Javascript: Funciones anónimas

- Las funciones anónimas se utilizan para que el código sea más legible y fácil de mantener al reducir la cantidad de funciones nombradas.
- Pueden ayudar a encapsular la lógica específica que no necesita ser reutilizada en otras partes del código.

Javascript: Funciones anónimas

- Ejemplo: Al tocar los botones, mediante JS cambiar la forma en que se muestran los artículos, agregando clases de Bootstrap

Párrafo

Grilla

Primer título
Primer párrafo

Segundo título
Segundo párrafo

Tercer título
Tercer párrafo

Cuarto título
Cuartopárrafo

Quinto título
Quinto párrafo

Párrafo

Grilla

Primer título
Primer párrafo

Segundo título
Segundo párrafo

Tercer título
Tercer párrafo

Cuarto título
Cuartopárrafo

Quinto título
Quinto párrafo

* Tenga en cuenta que no está cambiando el tamaño de pantalla

Javascript: Funciones anónimas

- Ejemplo:

```
<main class="container">
  <section class="container" id="botonera">
    <button class="btn btn-primary" value="opc1">Párrafo</button>
    <button class="btn btn-primary" value="opc2">Grilla</button>
  </section>
  <section id="contenedor-parrafos">
    <article style = "width: 18rem;">
      <h2 class="card-title">Primer título</h2>
      <p class="card-text">Primer párrafo</p>
    </article>
    <article style = "width: 18rem;">
      <h2 class="card-title">Segundo título</h2>
      <p class="card-text">Segundo párrafo</p>
    </article>
  </section>
</main>
```


Javascript: Funciones anónimas

- Ejemplo: inicio del código JS

```
let botones = document.querySelectorAll('#botonera button');
let contenedor = document.getElementById('contenedor-parrafos');
let articulos = document.querySelectorAll('#contenedor-parrafos article');

for (let i = 0; i < botones.length; i++) {
  botones[i].addEventListener('click', function() {
    if (botones[i].value === 'opc1') {
      contenedor.className = 'd-block';
      for (let j = 0; j < articulos.length; j++) {
        articulos[j].className = 'col-12 m-2 p-2 bg-white';
      }
    } else {
      contenedor.className = 'd-flex flex-wrap';
      for (let j = 0; j < articulos.length; j++) {
        articulos[j].className = 'card col-4 m-2 p-2 bg-primary
        bg-opacity-50';
      }
    }
  });
}
```




Selecciona sólo los article dentro
del id #contenedor-parrafos

Javascript: Funciones anónimas

- Ejemplo: detalle del for

```
for (let i = 0; i < botones.length; i++) {  
  botones[i].addEventListener('click', function() {  
    if (botones[i].value === 'opc1') {  
      contenedor.className = 'd-block';  
      for (let j = 0; j < articulos.length; j++) {  
        articulos[j].className = 'col-12 m-2 p-2 bg-white';  
      }  
    } else {  
      contenedor.className = 'd-flex flex-wrap';  
      for (let j = 0; j < articulos.length; j++) {  
        articulos[j].className = 'card col-4 m-2 p-2 bg-primary  
        bg-opacity-50';  
      }  
    }  
  });  
}
```



Reemplazamos las
clases de cada artículo

Javascript: Iteraciones - while

Mientras

```
MIENTRAS (condición)  
    Acciones;  
Fin_Mientras
```

while

```
while (condición) {  
    Acciones;  
}
```

Javascript: Iteraciones - while

```
while (condición) {  
    Acciones;  
}
```

- Para ingresar por primera vez a realizar las acciones del while, lo que se establezca como condición debe cumplirse al inicio (de no ser así no ingresará).
- Debe cambiar algo, dentro del while, para que en algún momento deje de repetir, sino se entra en un bucle infinito. Por ejemplo, debe ingresarse un valor, tener una fórmula que vaya cambiando, etc.

Javascript: Iteraciones - while

- Ejemplo: Realizar un script, que se encargue de preguntar la distancia que se tiene desde que inicia el recorrido. Dejará de preguntar al estar a 1 metro o menos del anfiteatro

```
const DIST_ACEPTADA = 1; /* valor fijo hasta finalizar */
let distancia = 50; /* en metros */
while (distancia > DIST_ACEPTADA) {
    distancia = parseInt(window.prompt('Ingrese Distancia: '));
    /* seguir caminando */
}

window.alert('Estoy en la zona. Espero instrucciones');
```

Javascript: Iteraciones - while

- Ejemplo: Desarrollar un script, que permita ir sumando los importes de una compra del súper, teniendo en cuenta que tendrá un tope de 10000 pesos. (Cambiando el tope sirve para cualquier otro caso con valores más altos)

```
let compra = 0;
let costo = 0;
const TOPE = 10000; /* valor fijo */
while (compra < TOPE) {
    costo = window.prompt('Ingrese costo:');
    if (isNaN(costo)) { /* controla si no es un número */
        window.alert('debe ser un número');
    } else {
        compra = compra + parseFloat(costo); /* Convierte a número real */
    }
}
```

Javascript: Iteraciones - while

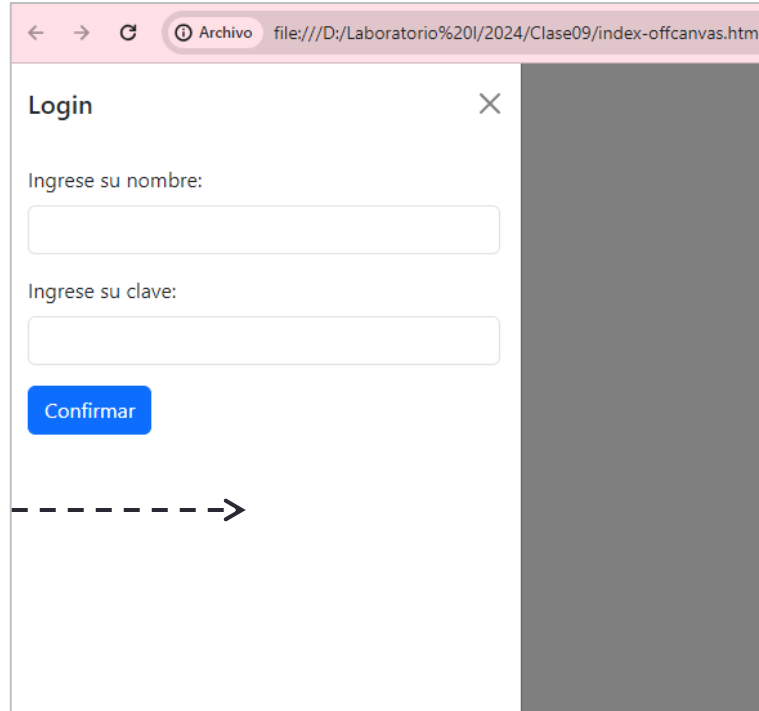
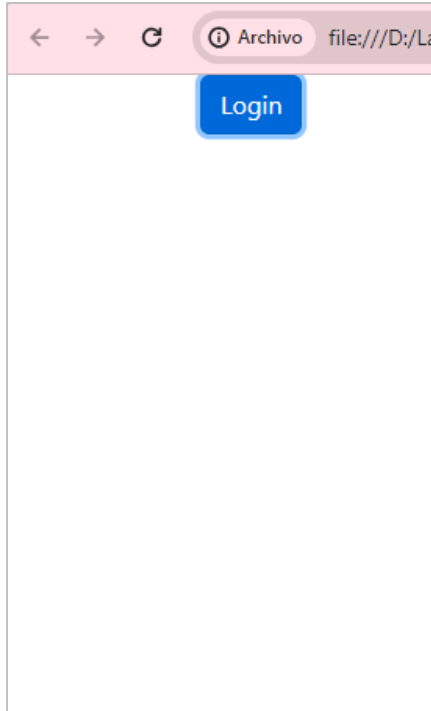
- Continuación...

```
if (compra > TOPE) {  
    compra = compra - costo; /* resta de la compra si se excedió */  
    window.alert('El último producto superaba el presupuesto y se quitó');  
}  
  
window.alert('Total compra: ' + compra);
```

Bootstrap: Component: Offcanvas

- Barras de navegación laterales ocultas
- Utiliza Javascript
- No se puede usar `margin` o `translate` en un elemento `.offcanvas`. En su lugar, utilizar la clase en una etiqueta envolvente independiente.

Bootstrap: Component: Offcanvas



Bootstrap: Component: Offcanvas

```
<section class="container">
  <a class="btn btn-primary" data-bs-toggle="offcanvas" href="#offcanvas1">
    Login
  </a>
  <section class="offcanvas offcanvas-start" tabindex="-1" id="offcanvas1">
    <section class="offcanvas-header">
      <h2 class="offcanvas-title">Login</h2>
      <button type="button" class="btn-close text-reset"
        data-bs-dismiss="offcanvas"></button>
    </section>
    <section class="offcanvas-body">
      <form action="php/procesar.php" method="post">
        <section class="mb-3">
          <label for="nombre" class="form-label">Ingresa su nombre:</label>
```

Aparece desde izquierda

se posicionará primero ahí cuando se haga tab

Cerrará el offcanvas

Bootstrap: Component: Offcanvas

```
<label for="nombre" class="form-label">Ingrese su nombre:</label>
<input type="text" class="form-control" id="nombre" name="nombre">
</section>
<section class="mb-3">
  <label for="clave" class="form-label">Ingrese su clave:</label>
  <input type="password" class="form-control" id="clave" name="clave">
</section>
<button type="submit" class="btn btn-primary">Confirmar</button>
</form>
</section>
</section>
</section>

<script src="bootstrap-5.3.3-dist/js/bootstrap.min.js"></script>
```

Uso de íconos en la web

- **Mejora de la Experiencia del Usuario:** Los íconos pueden ayudar a los usuarios a comprender rápidamente la función de un elemento o una sección de una página web.
 - Por ejemplo, un ícono de una casa puede indicar la página de inicio, un ícono de un sobre puede representar la sección de mensajes o correo electrónico, etc.
- **Ahorro de Espacio:** Los íconos ocupan menos espacio visual que el texto, lo que permite mostrar más información en una página web sin que luzca cargada (en el caso de decidir que no haya texto).
- **Estética y Diseño:** Los íconos pueden agregar un toque visual atractivo y estilizado al diseño de una página web. Pueden ayudar a crear una identidad visual coherente y atractiva para una marca o sitio web.

Bootstrap: icons

- Bootstrap tiene una biblioteca de iconos gratuita, de alta calidad y de código abierto con más de 2000 iconos.
- Se pueden usar como: SVG, sprites SVG o fuentes web.
- Ventaja de trabajar con íconos a partir de fuentes: las fuentes suelen pesar menos que las imágenes, lo que puede mejorar los tiempos de carga de la página web.
- Este tiempo de carga de un sitio se puede medir, por ejemplo, usando:

Bootstrap: icons

- Encontramos el enlace de descarga de las fuentes y como se utiliza en:

<https://icons.getbootstrap.com/>


- Enlace de descarga:

<https://github.com/twbs/icons/releases/latest/>

Bootstrap: icons

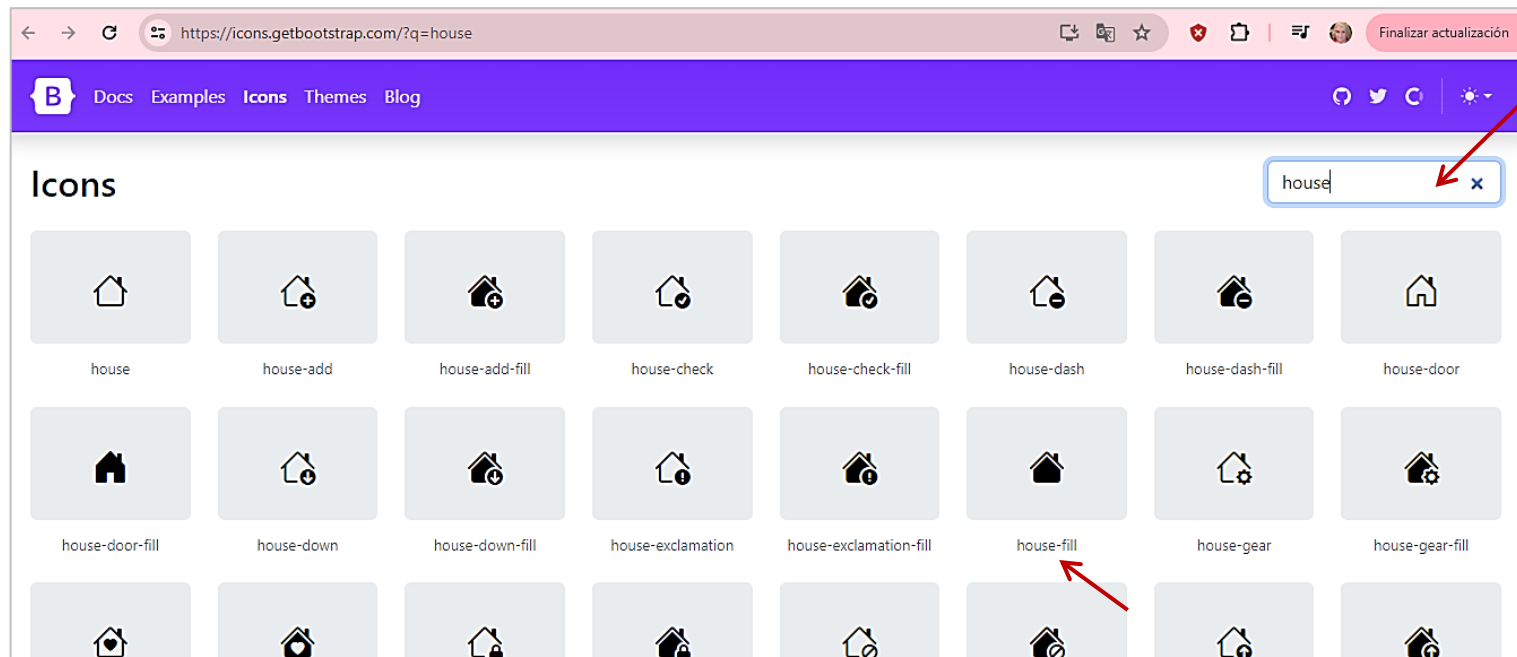
- Se debe linkear el archivo de fuentes de íconos en el head de nuestras páginas

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Clase 09</title>
  <link rel="stylesheet" href="bootstrap-icons-1.11.3/font/bootstrap-icons.min.css">
  <link rel="stylesheet" href="bootstrap-5.3.3-dist/css/bootstrap.min.css">
</head>
```



Bootstrap: icons

- En la página de Bootstrap se buscará la clase que usaremos



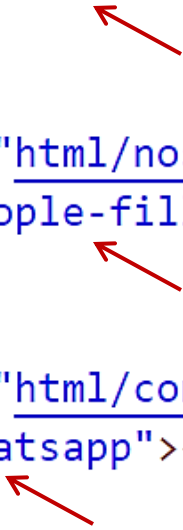
Bootstrap: icons

- Utilizaremos la etiqueta span y le colocaremos la clase bi y la clase del ícono elegido anteponiendo **bi-** al nombre elegido.
- Por ejemplo, si el nombre de la clase es house, deberemos poner:

```
<span class="bi bi-house">
```




Bootstrap: icons

```
<ul class="navbar-nav fs-4">
  <li class="nav-item">
    <a class="nav-link" href="#">
      <span class="bi bi-house-fill"></span>Inicio</a>
    </li>
  <li class="nav-item">
    <a class="nav-link" href="html/nosotros.html">
      <span class="bi bi-people-fill"></span>Nosotros</a>
    </li>
  <li class="nav-item">
    <a class="nav-link" href="html/contacto.html">
      <span class="bi bi-whatsapp"></span>Contacto</a>
    </li>
</ul>
```






Bootstrap: icons

- Sin retocar el css, se verá así:




 Inicio  Nosotros  Contacto

- Con nuestro css se verá así:

 Inicio  Nosotros  Contacto

Bootstrap: icons

- El CSS:

 Inicio  Nosotros  Contacto

```
.bi {  
  margin-right: 0.5em;  
}  
  
.bi-whatsapp {  
  color:  #25D366;  
}  
  
.bi-people-fill {  
  color:  #F58C27;  
}  
  
#menu a {  
  background-color:  antiquewhite;  
  margin-right: 0.1em;  
  color:  black;  
}
```

Anexo 1: Implementación de Según



- Cuando ante una pregunta existen más de dos respuestas posibles, que no se pueden manejar de manera sencilla con una estructura Si Sino Si, utilizaremos Según.

Anexo 1: Implementación de Según (switch)

```
SEGUN variable
    valor1:
        respuesta1
    valor2:
        respuesta2
...

SINO
    respuesta_por_defecto
Fin-SEGUN
```

```
switch (variable) {
    case valor1:
        respuesta1;
        break;
    case valor2:
        respuesta2;
        break;
    ...
    default:
        respuesta_por_defecto;
}
```

Anexo 1: Implementación de Según (switch)

- Ejemplo: Crear una página que tenga un formulario con un select para las carreras de Agrimensura, PU, Lic. en Informática e Ing. en Informática, depende la carrera que se seleccione, se mostrará una mensaje distinto

Selecciona tu carrera:

Carrera:

Programador Universitario



Los programadores universitarios desarrollan software y aplicaciones informáticas.

Anexo 1: Implementación de Según (switch)

- Ejemplo:

```
<section class="container py-5 col-6 fs-2">
  <form id="formulario" action="#">
    <section class="mb-3">
      <label for="carrera" class="form-label">Carrera:</label>
      <select id="carrera" class="form-select fs-4">
        <option value="---">--- Seleccione ---</option>
        <option value="agrimensura">Agrimensura</option>
        <option value="programador">Programador Universitario</option>
        <option value="licenciatura">Licenciatura en Informática</option>
        <option value="ingenieria">Ingeniería en Informática</option>
      </select>
    </section>
  </form>
</section>
```


Anexo 1: Implementación de Según (switch)

- Ejemplo:

```
        </form>
    </section>
    <main class="container fs-3">
        <p id="mensaje" class="mt-4">

        </p>
    </main>
    <script src="js/mi-script"></script>
</body>
```

Anexo 1: Implementación de Según (switch)

- Ejemplo: inicio del JS

```
function mostrarMensaje() {  
  let carreraSeleccionada = selectCarrera.value;  
  let mensaje = '';  
  switch (carreraSeleccionada) {  
    case "agrimensura":  
      mensaje = "La agrimensura se dedica a la medición y delimitación de terrenos.";   
      break;  
    case "programador":  
      mensaje = "Los programadores universitarios desarrollan software y aplicaciones informáticas."  
      break;  
    case "licenciatura":  
      mensaje = "La licenciatura en informática se enfoca en la teoría y práctica de la computación."  
      break;  
    case "ingenieria":  
      mensaje = "La ingeniería en informática se centra en el diseño y desarrollo de sistemas y
```

Anexo 1: Implementación de Según (switch)

- Ejemplo: detalle del switch

```
switch (carreraSeleccionada) {  
    case "agrimensura":  
        mensaje = "La agrimensura se dedica a la medición y delimitación de terrenos.";  
        break;  
    case "programador":  
        mensaje = "Los programadores universitarios desarrollan software y aplicaciones informáticas."  
        ";  
        break;  
    case "licenciatura":  
        mensaje = "La licenciatura en informática se enfoca en la teoría y práctica de la  
        computación.";  
        break;  
    case "ingenieria":  
        mensaje = "La ingeniería en informática se centra en el diseño y desarrollo de sistemas y  
        tecnologías informáticas.";  
        break;  
    default:  
        mensaje = "Selecciona una carrera válida.";  
}  
mensajeElement.innerHTML = mensaje;
```

Anexo 1: Implementación de Según (switch)

- Ejemplo: debajo de la función

```
    }  
    mensajeElement.innerHTML = mensaje;  
}  
  
let selectCarrera = document.getElementById('carrera');  
let mensajeElement = document.getElementById('mensaje');  
selectCarrera.addEventListener('change', mostrarMensaje);
```