	ESCUELA SUPERIOR DE ECONOMÍA Y NEGOCIOS **INGENIERÍA DE SOFTWARE Y NEGOCIOS DIGITALES**
CICLO 03-2025	**GUIA DE LABORATORIO Nº 5** **Nombre de la practica:** Introducción a Javascript **Tiempo estimado:** 2 horas **Materia:** Desarrollo Web I

I. OBJETIVOS

Que al finalizar la practica el estudiante:

- Adquiera dominio de los elementos sintácticos básicos del lenguaje JavaScript.
- Sea capaz de generar código HTML desde secuencias de comando de JavaScript.
- Pueda crear páginas web utilizando métodos básicos de interacción con el usuario.
- Utilice correctamente estructuras selectivas y repetitivas en JavaScript.

II. INTRODUCCIÓN TEORICA

¿Cómo agregamos código JavaScript a nuestra página web?

JavaScript se aplica de manera similar a CSS. Mientras que CSS usa elementos <link> para aplicar hojas de estilo externas y elementos <style> para aplicar hojas de estilo internas a HTML, JavaScript solo necesita un elemento de tipo {htmlelement("script")}}. A continuación se detalla la forma en que podemos incluir JavaScript:

JavaScript interno

Podemos utilizar la etiqueta <script></script>, generalmente se agrega dentro de la etiqueta <head></head> de nuestro documento HTML, nótese la línea 7 – 9.

```
1   <!DOCTYPE html>
2   <html lang="es">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <script>
8           // AQUI IRIA NUESTRO CODIGO JAVASCRIPT
9       </script>
10      <title>Usando Javascript</title>
11  </head>
12  <body>
13
14  </body>
15  </html>
```

JavaScript Externo

Siempre utilizamos la etiqueta <script></script>, sin embargo ahora incorporamos el atributo src, el cual indicara el lugar donde se encuentra nuestro archivo .js, es importante recalcar que los archivos que son utilizados para ejecutar código JavaScript deben ser guardados con la extensión .js, caso contrario el navegador no podrá interpretar su código. Nótese la línea 7

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="example.js" defer></script>
    <title>Usando Javascript</title>
</head>
<body>

</body>
</html>
```

Controladores de JavaScript en línea

También podemos utilizar fragmentos de código JavaScript dentro de HTML. En la línea 12 estamos creando un botón en HTML y estamos utilizando un evento llamado onclick, el cual hace referencia a una función llamada crearParrafo(), está se encuentra definida dentro de la etiqueta <script></script>, lo curioso es que la etiqueta <script></script> esta fuera de la etiqueta <head></head>.

```html
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Usando Javascript</title>
</head>

<body>
    <button onclick="crearParrafo()">Pulsa aqui!</button>
    <script>
        function crearParrafo() {
            let parrafo = document.createElement('p');
            parrafo.textContent = 'Presiona el boton!';
            document.body.appendChild(parrafo);
        }
    </script>

</html>
```

Estrategias para la carga de script

Un problema común es que todo el HTML de una página se carga en el orden en que aparece. Si estamos utilizando JavaScript para manipular elementos en la página (o exactamente, el DOM), tu código no funcionará si el JavaScript se carga y procesa antes que el HTML que estás intentando haga algo.

En los ejemplos internos y externos, JavaScript se carga y se ejecuta en el encabezado del documento, antes de analizar el cuerpo HTML. Esto podría causar un error, ya que el documento HTML no ha sido construido en su totalidad. Para resolver este inconveniente podemos implementar las siguientes estrategias:

1. **Agregar el escuchador o evento DOMContentLoaded**

```html
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <script>
      document.addEventListener("DOMContentLoaded", function () {
        // AQUI IRIA NUESTRO CODIGO JAVASCRIPT
      });
    </script>
    <title>Usando Javascript</title>
  </head>
  <body></body>
</html>
```

El evento "DOMContentLoaded" del navegador, permite verificar que el cuerpo HTML está completamente cargado y analizado. Luego, el JavaScript dentro de este bloque se ejecutará sin provocar algún error de carga en la manipulación del DOM.

2. **Utilizando el atributo async**

Los scripts cargados con el atributo async descargarán el script sin bloquear el renderizado de la página y lo ejecutará tan pronto como el script se termine de descargar. No tienes garantía de que los scripts se ejecuten en un orden específico, solo que no detendrán la visualización del resto de la página. Es mejor usar async cuando los scripts de la página se ejecutan de forma independiente y no dependen de ningún otro script de la página.

Por ejemplo, si tienes los siguientes elementos script:

```html
<script async src="js/vendor/jquery.js"></script>

<script async src="js/script2.js"></script>

<script async src="js/script3.js"></script>
```

No podemos confiar que el orden que se ha colocado los archivos, también será el orden en que se cargarán los scripts, es decir primer jquery.js se puede cargar antes o después de script2.js y script3.js y si este es el caso, cualquier función en esos scripts dependiendo de jquery producirá un error porque jquery no se definirá en el momento en que se ejecute el script.

async se debe usar cuando tenemos varios scripts en segundo plano para cargar, y solo deseamos ponerlos en su lugar lo antes posible.

3. **Utilizando el atributo defer**

Los scripts cargados con el atributo defer se ejecutarán en el orden en que aparecen en la página y se ejecutara tan pronto se descarguen el script y el contenido:

```html
<script defer src="js/vendor/jquery.js"></script>

<script defer src="js/script2.js"></script>

<script defer src="js/script3.js"></script>
```

Todos los scripts con el atributo defer se cargarán en el orden en que aparecen en la página. Entonces, en el ejemplo anterior, podemos estar seguros de que jquery.js se cargará antes que script2.js y script3.js y que script2.js se cargará antes de script3.js. No se ejecutarán hasta que se haya cargado todo el contenido de la página, lo cual es útil si tus scripts dependen de que el DOM esté en su lugar.

Tipo de las variables

Hay algunos tipos de datos diferentes que podemos almacenar en variables. A continuación se detalla los más utilizados:

- **Números**
  Podemos almacenar números en variables, ya sean números enteros como 30 (también llamados enteros o integer) o números decimales como 2.456 (también llamados números flotantes, de coma flotante o number). No es necesario declarar el tipo de las variables en JavaScript, a diferencia de otros lenguajes de programación. Cuando le otorgamos a una variable un valor numérico, no se deben incluir comillas: `let edad = 17;`
- **Cadenas de caracteres (Strings)**
  Las strings (cadenas) son piezas de texto. Cuando le otorgamos a una variable un valor de cadena, se debe encerrar entre comillas simples o dobles; de lo contrario, JavaScript intenta interpretarlo como otro nombre de variable. `let mensaje = 'Bienvenido a JavaScript';`
- **Booleanos**
  Los booleanos son valores verdadero/falso, pueden tener dos valores, true o false. Estos, generalmente se utilizan para probar una condición, después de lo cual se ejecuta el código según corresponda. Así, por ejemplo, un caso simple sería: `let correcto = true;`
  En realidad se usaría más de la siguiente manera: `let verificar = 6 < 3;`, aquí se está usando el operador "menor que" (<) para probar si 6 es menor que 3. Como era de esperar, devuelve false, ¡porque 6 no es menor que 3!
- **Arreglos**
  Un arreglo es un objeto único que contiene múltiples valores encerrados entre corchetes y separados por comas. A continuación se definen dos arreglos con distintos valores:
  ```
  let arregloNombres = ['Chris', 'Bob', 'Jim'];
  let arregloNumeros = [10, 15, 40];
  ```
  Una vez que se definen estos arreglos, podemos acceder a cada valor por su ubicación dentro del arreglo. A continuación recuperamos los valores de Chris y 40 respectivamente:
  ```
  arregloNombres[0]; // debería devolver 'Chris'
  arregloNumeros[2]; // debe devolver 40
  ```
  Los corchetes especifican un valor de índice correspondiente a la posición del valor que desea devolver. Posiblemente haya notado que los arreglos en JavaScript tienen índice cero: el primer elemento está en el índice 0.
- **Objetos**
  En programación, un objeto es una estructura de código que modela un objeto de la vida real. Puedes tener un objeto simple que represente una caja y contenga información sobre su ancho, largo y alto, o podrías tener un objeto que represente una persona y contenga datos sobre su nombre, estatura, peso, qué idioma habla, cómo saludarlo, y más. El siguiente ejemplo representa un objeto.
  ```
  let perro = { nombre: 'Toto', raza: 'Dalmata' };
  ```
  Para recuperar la información guardada en el objeto, podemos utilizar la siguiente sintaxis:
  ```
  perro.nombre
  ```

Tipado dinámico

JavaScript es un "lenguaje tipado dinámicamente", lo cual significa que, a diferencia de otros lenguajes, no es necesario especificar qué tipo de datos contendrá una variable (números, cadenas, arreglos, etc.).

Por ejemplo, si declara una variable y le das un valor entre comillas, el navegador trata a la variable como una cadena (string): `let saludo = 'Hola';`

Incluso si el valor contiene números, sigue siendo una cadena, en el siguiente ejemplo se hace una demostración:

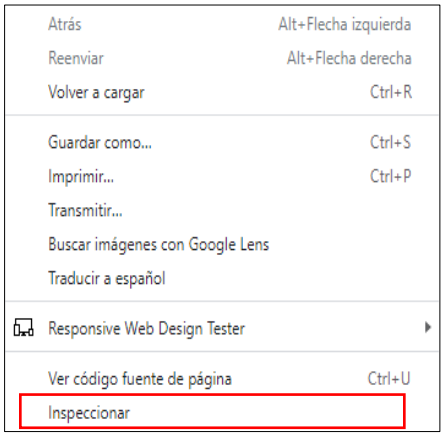
```
let numero = '500'; // Vaya, esto sigue siendo una cadena
typeof numero;
numero = 500; // mucho mejor — ahora este es un número
typeof numero;
```

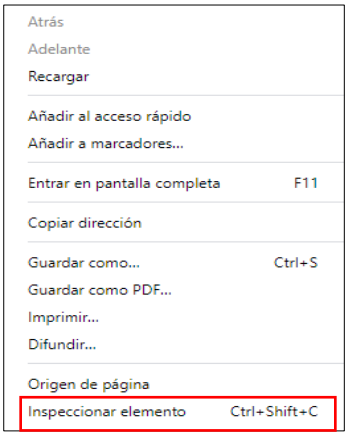
Notarás que estamos usando un operador especial llamado typeof, esto devuelve el tipo de datos de la variable. La primera vez que se llama, debe devolver string, ya que en ese punto la variable numero contiene una cadena, '500'. La segunda llamada de typeof debería de volver number.

Depuración de errores con el navegador

Frecuentemente nos encontraremos con situaciones en donde será necesario corregir los scripts realizados con JavaScript. Es una tarea común en programación que la página web que contiene código JavaScript no se comporte de la manera esperada o no produzca el resultado que de acuerdo a nuestra lógica es el correcto. Generalmente, esto sucede cuando se produce algún error en la etapa de interpretación ejecutada por el navegador, ya sea debido a algún error de sintaxis o a un error en la lógica de programación.

Hoy en día todos los navegadores actuales poseen sus propias “herramientas del desarrollador”, generalmente dando clic derecho del mouse dentro de la página web podremos acceder a estas herramientas, en algunos navegadores observaremos que se llama “inspeccionar” y en otros “inspeccionar elementos”.

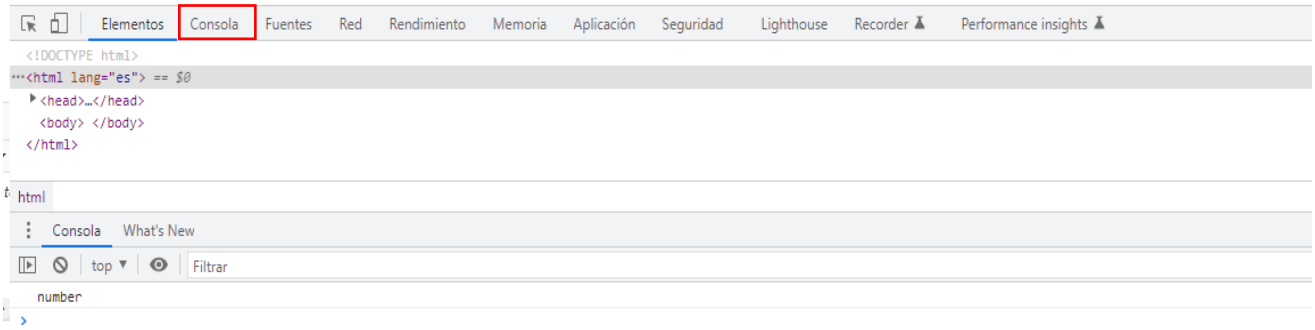




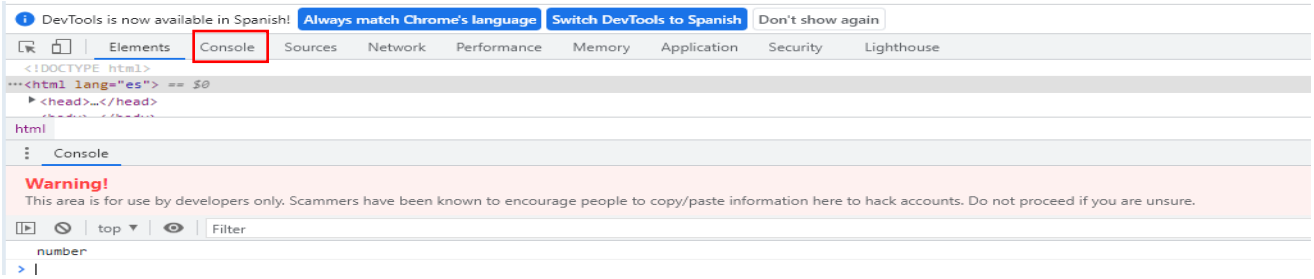
Posteriormente de su selección, se nos mostrara diferentes opciones para los desarrolladores de páginas web, desde la estructura HTML, CSS y todos los archivos cargados en nuestro sitio.

En nuestro caso nos enfocaremos en la opción “Consola”, a continuación se muestra la herramienta en diferentes navegadores web.

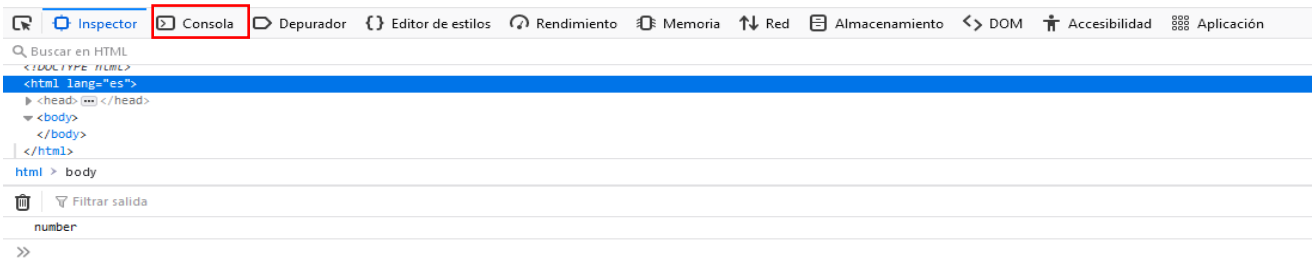
Consola de Google Chrome Versión: 105.0.5195.102



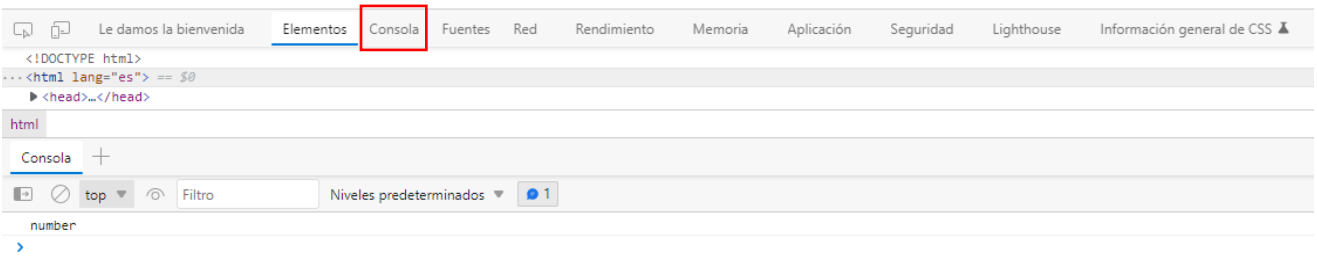
Consola de Opera Versión: 90.0.4480.80



Consola de Firefox Versión: 104.0.1



Consola de Microsoft Edge Versión 105.0.1343.27



II. DESARROLLO DE LA PRÁCTICA

Cree una carpeta con el nombre Guia5_[su número de carnet], y luego proceda a crear las siguientes carpetas: jsy css y pages.

PARTE I: FUNDAMENTOS DE JAVASCRIPT

EJEMPLO 1: INTERACCIÓN BÁSICA CON EL USUARIO

JavaScript posee algunas funciones predefinidas que nos permiten interactuar con el usuario, estas son: alert, prompt y confirm.

- La función alert, muestra un mensaje en pantalla y espera que el usuario presione "Aceptar".
- La función prompt, muestra una ventana modal con un mensaje de texto, un campo de entrada para capturar información del usuario y dos botones (Ok y Cancelar). La función prompt acepta dos argumento: prompt(title, [default]);
  - Title -> el texto a mostrar al usuario
  - Default -> un segundo argumento que es opcional, permite colocar un valor inicial.
- La función confirm, muestra una ventana modal con una pregunta y dos botones (Ok y Cancelar).

1. Proceda a crear descargar el archivo style.css de los recursos de la guía y guárdelo en la carpeta css.
2. Proceda a crear un archivo con el nombre de buttons.js y guárdelo en la carpeta js.
3. Luego proceda a añadir el siguiente código en el archivo creado en el paso anterior:

```
function aviso() {
  alert("Bienvenido al mundo JavaScript");
}

function confirmacion() {
  // Los valores que puede almacenar la variable confirmacion
  // son true o false
  let confirmacion = confirm("¿Desea salir de la Sesión?");
  /* para imprimir una variable en una cadena podemos
    utilizar las comillas simples inversas `` y luego hacemos el llamado
    de la variable con ${aqui debera de escribir el nombre de la variable}
    */
  alert(`Valor seleccionado ${confirmacion}`);
}

function capturarDatos() {
  let nombre = prompt("¿Cual es su nombre?");
  // Notese que en campo del promt se mostrara 0 por defecto
  let edad = prompt("¿Cual es su edad?", 0);

  alert(`Su nombre es ${nombre} y su edad ${edad}`);
}
```

```
23
24  function dibujarParrafo() {
25    let parrafo = prompt(
26      "Escriba la  información que desea visualizar en el parrafo"
27    );
28
29    /* Utilizaremos la API DOM para acceder al elemento
30      <p id="idParrafo"></p> que hemos creado en nuestro documento HTML*/
31
32    const p = document.querySelector("#idParrafo");
33    p.innerHTML = parrafo;
34  }
```

4. Cree un archivo con el nombre index.html y guárdelo en la carpeta raíz.
5. Luego proceda a crear la siguiente estructura del documento:

```
1   <!DOCTYPE html>
2   <html lang="es">
3     <head>
4       <meta charset="UTF-8" />
5       <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6       <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7       <title>Interactuando con JavaScript</title>
8       <link href="./css/style.css" rel="stylesheet" />
9       <script src="./js/buttons.js" defer></script>
10    </head>
11
12    <body>
13      <div class="container">
14        <h1>Interactuando con JavaScript</h1>
15        <div class="flex-container">
16          <button id="idBtnAlert" onclick="aviso()">Funcion alert</button>
17          <button id="idBtnPromt" onclick="capturarDatos()">Funcion promt</button>
18          <button id="idBtnConfirm" onclick="confirmacion()">Funcion confirm</button>
19          <button id="idBtnParrafo" onclick="dibujarParrafo()">Dibujar parrafo</button>
20        </div>
21        <h4>Parrafo de información</h4>
22        <p id="idParrafo"></p>
23      </div>
24    </body>
```

6. Proceda a visualizar el resultado en el navegador web de su preferencia.

EJEMPLO 2: CREANDO TABLAS CON JAVASCRIPT

1. Edite el archivo index.html y agregue la siguiente línea después de

```
22        <p id="idParrafo"></p>
23        <a href="pages/tablas.html">Crear tablas</a>
```

2. Cree un archivo llamado tablas.html y guárdelo en la carpeta pages.
3. Ahora proceda a crear la siguiente estructura de código.

```
1   <!DOCTYPE html>
2   <html lang="es">
3     <head>
4       <meta charset="UTF-8" />
5       <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6       <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7       <script src="../js/tablas.js"></script>
8       <link href="../css/style.css" rel="stylesheet" />
9       <title>Crear tablas con JavaScript</title>
10    </head>
11    <body>
12      <div class="container">
13        <h1>Creando una Tabla con JavaScript</h1>
14        <div id="idContenedor"></div>
15        <a href="../index.html">Regresar</a>
16      </div>
17    </body>
18  </html>
```

4. Cree un archivo llamado tablas.js y guárdelo en la carpeta js. Añada el siguiente código:

```
// Creación de la tabla utilizando concatenación de cadenas
let table = "<table>";
table += "<thead>";
table += "<tr>";
table += "<th scope='col'>#</th>";
table += "<th scope='col'>Nombre</th>";
table += "<th scope='col'>Apellido</th>";
table += "<th scope='col'>Correo electrónico</th>";
table += "</tr>";
table += "</thead>";
table += "<tbody>";

// Datos de los alumnos
const alumnos = [
    { id: 1, nombre: "Marcos Antonio", apellido: "Alas", correo:
"marcos.alas@estudiante.esen.edu.sv" },
```

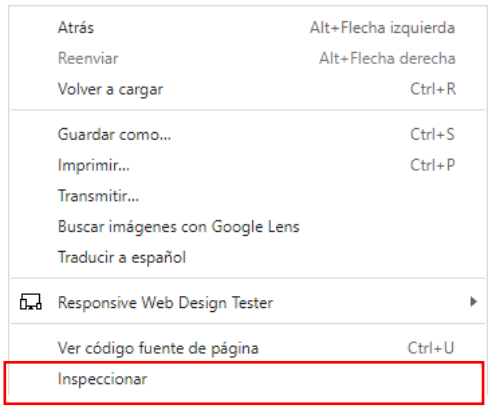
```
    { id: 2, nombre: "Ana Paola", apellido: "Rivas Polanco", correo:
"paola.rivas@estudiante.esen.edu.sv" },
    { id: 3, nombre: "Alexis Armando", apellido: "Quintanilla Peña", correo:
"alexis.quintanilla@estudiante.esen.edu.sv" },
    { id: 4, nombre: "Vanessa Alejandra", apellido: "Bermudez Urquilla",
correo: "vanessa.bermudez@estudiante.esen.edu.sv" },
    { id: 5, nombre: "Oscar Armando", apellido: "López Rodriguez", correo:
"oscar.lopez@estudiante.esen.edu.sv" }
];

// Agregar filas de los datos al cuerpo de la tabla
alumnos.forEach(alumno => {
    table += "<tr>";
    table += `<td scope='row'>${alumno.id}</td>`;
    table += `<td>${alumno.nombre}</td>`;
    table += `<td>${alumno.apellido}</td>`;
    table += `<td>${alumno.correo}</td>`;
    table += "</tr>";
});

table += "</tbody>";
table += "</table>";
// Agregar la tabla al contenedor
const contenedor = document.querySelector("#idContenedor");
contenedor.innerHTML = table;
```

5. Proceda a actualizar en el navegador la página index.html, podrá observar los cambios que hemos realizado hasta este momento. Ahora procesa a dar clic en opción “Crear tablas”.

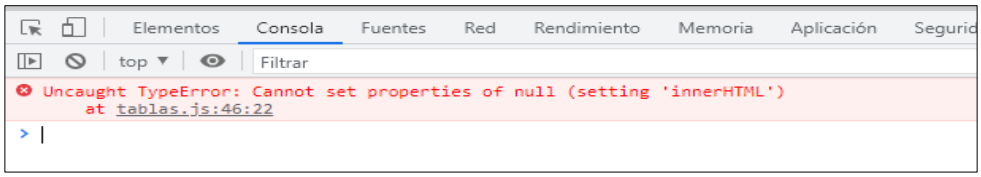
6. Podrá observar que nos ha llevado a la página llamada “tablas.html” y se visualiza el siguiente resultado:

¿Por qué no se ha creado la tabla si la hemos definido en nuestro archivo tablas.js?, la respuesta es simple, el script tablas.js se está cargando antes que finalice la carga de nuestro documento HTML, esto hace que el DOM (Document Object Model) no esté disponible al momento de querer acceder desde el document.querySelector("#idContenedor");

7. Da clic derecho del mouse en la página web (tablas.html) y selecciona “inspeccionar” o “inspeccionar elemento”.



8. Luego selecciona consola y podrás observar que se nos está mostrando un error referente al DOM.



Nota: Siempre que tengamos inconvenientes con el funcionamiento de nuestros script js, lo primero que debes hacer es verificar la consola para verificar los posibles errores.

9. Para solucionar este inconveniente aplicaremos el atributo “defer” en la etiqueta <script></script> del documento tablas.html

```
7    <script src="../js/tablas.js" defer></script>
```

El atributo defer permite cargar el script js después de haber cargado el DOM.

10. Ahora proceda a actualizar la página “tablas.html” en el navegador. Obtendrá el siguiente resultado:

EJEMPLO 3: NÚMERO ALEATORIO

1. Edite el archivo index.html y agregue la siguiente línea después del enlace “Crear tablas”.

```
<p id="idParrafo"></p>
<a href="pages/tablas.html">Crear tablas</a>
<a href="pages/numeroAleatorio.html">Adivinando el número</a>
```

2. Cree un archivo llamado numeroAleatrio.html y guárdelo en la carpeta pages.

3. Proceda a agregar la siguiente información en el archivo creado anteriormente.

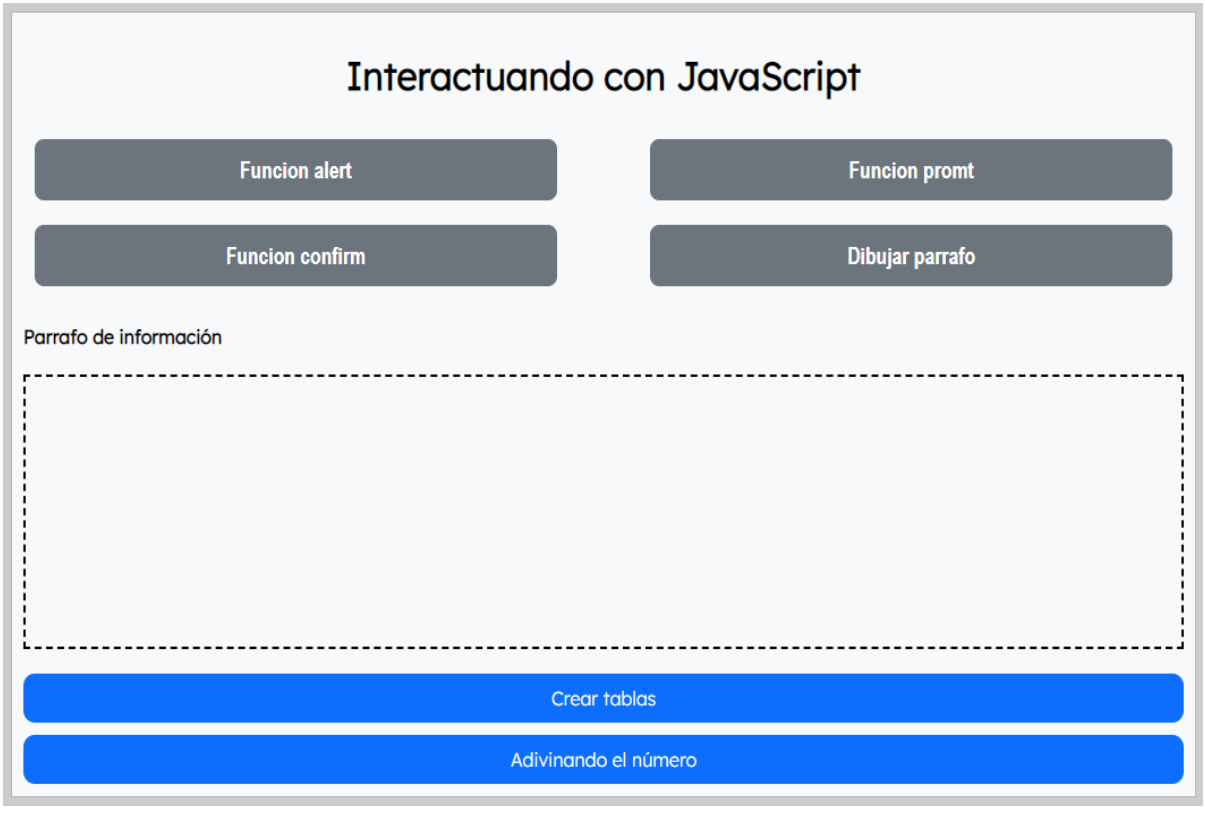
```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <script src="../js/numeroAletorio.js"></script>
    <link href="../css/style.css" rel="stylesheet" />
    <title>Adivinando un número</title>
  </head>
  <body>
    <div class="container">
      <h1>Adivinando un número (rango 1 - 25)</h1>
      <div class="flex-container">
        <button id="idBtnNumero" onclick="generarNumeroAleatorio()">Adivina el número</button>
      </div>
      <p id="idParrafo"></p>
      <a href="../index.html">Regresar</a>
    </div>
  </body>
</html>
```

4. Cree un archivo llamado numeroAletorio.js y guárdelo en la carpeta js. Agregue el siguiente código:

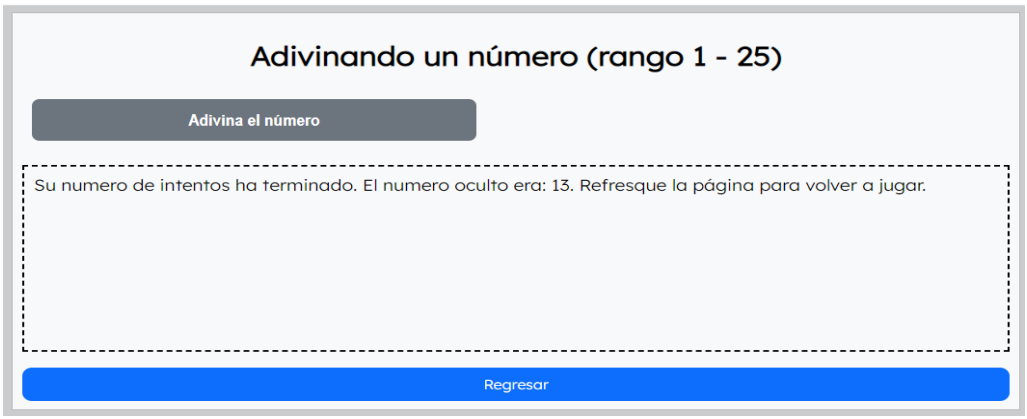
```
//Generamos un numero aleatorio que se encuentre en el rango del 1 al 25
const numeroAleatorio = Math.floor(Math.random() * 25) + 1;
// Creamos una constante que permite identificar el maximo de intentos
const numeroIntentos = 3;
// Guardara el numero de intentos que realiza el usuario
let intentos = 1;
function generarNumeroAleatorio() {
  //Definimos una variable para impresion de mensajes
  let mensaje;
  // Utilizamos el dom para acceder al parrafo creado
  const parrafo = document.querySelector("#idParrafo");

  // Verificamos en que intento esta el usuario
  if (intentos <= numeroIntentos) {
    let numero = prompt(
      "¿Que número se ha generado (Intento " + intentos + ")?"
    );

```

```
    //verificamos el numero aleatorio con el ingresado por el usuario
    if (numero == numeroAleatorio) {
      mensaje = `¡Es sorprente, pudiste adivinar el numero oculto (${numeroAleatorio}).
        Refresque la página para volver a jugar.`;
    } else if (intentos == numeroIntentos) {
      mensaje = `Su numero de intentos ha terminado.
          El numero oculto era: ${numeroAleatorio}. Refresque la página para volver a jugar.`;
    } else {
      mensaje = `Vuelve a intentar. Quedan ${
        numeroIntentos - intentos
      } intentos`;
    }

    //aumentamos el valor de los intentos
    intentos++;
  } else {
    mensaje = `Su numero de intentos ha terminado.
      El numero oculto era: ${numeroAleatorio}. Refresque la página para volver a jugar.`;
  }

  parrafo.innerHTML = mensaje;
}
```

5. Actualice en su navegador la página index.html y proceda a verificar los cambios. Deberá de tener el siguiente resultado.



6. Comience a interactuar con la página numeroAleatorio.html, tendría que tener un resultado como el siguiente.



EJEMPLO 4: OPERACIONES BÁSICAS

1. Edite el archivo index.html y agregue la siguiente línea después del enlace “Adivinando el número”.

```
<p id="idParrafo"></p>
<a href="pages/tablas.html">Crear tablas</a>
<a href="pages/numeroAleatorio.html">Adivinando el número</a>
<a href="pages/operacionesBasicas.html">Operaciones básicas</a>
```

2. Cree un archivo llamado operacionesBasicas.html y guárdelo en la carpeta pages.
3. Edite el archivo con la siguiente estructura HTML.

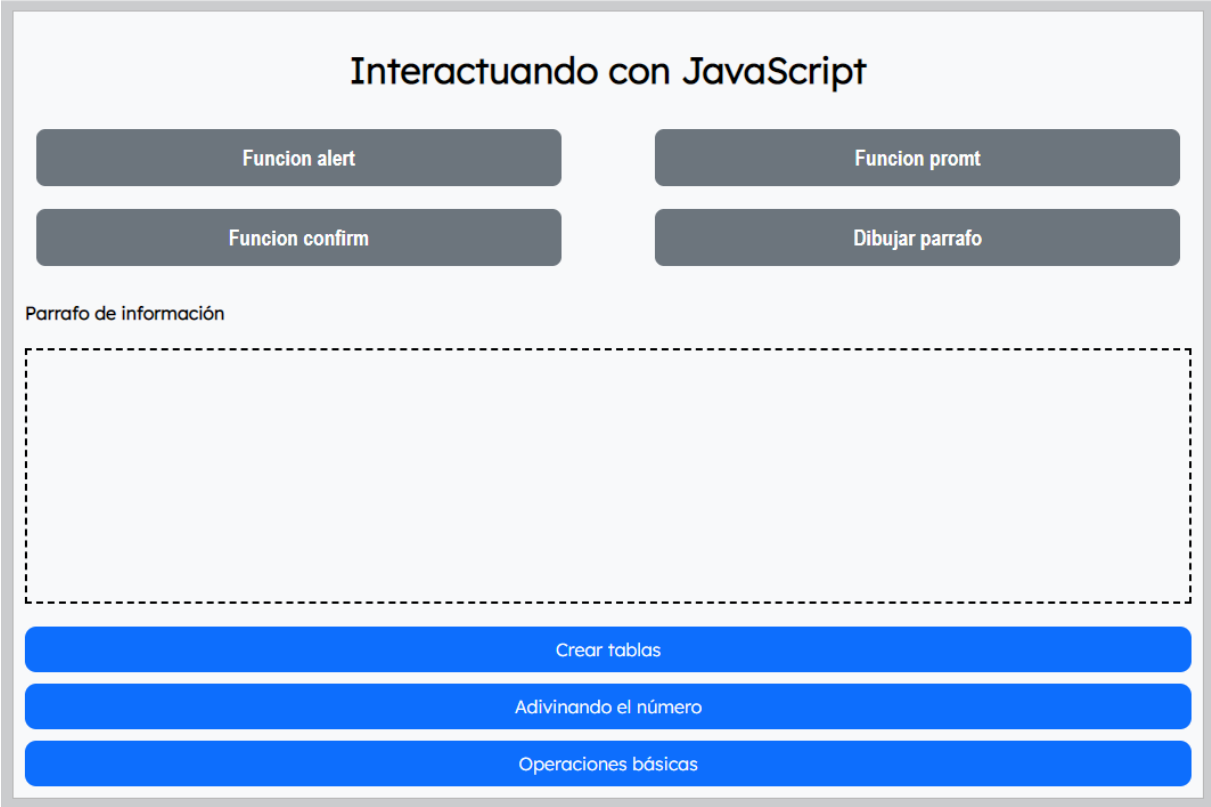
```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <script src="../js/operacionesBasicas.js"></script>
    <link href="../css/style.css" rel="stylesheet" />
    <title>Operaciones básicas con JavaScript</title>
  </head>
  <body>
    <div class="container">
      <h1>Operaciones básicas con JavaScript</h1>
      <div class="flex-container">
        <button id="idBtnSumar">Sumar</button>
        <button id="idBtnRestar">Restar</button>
        <button id="idBtnMultiplicar">Multiplicar</button>
        <button id="idBtnDividir">Dividir</button>
      </div>
      <p id="idParrafo"></p>
      <a href="../index.html">Regresar</a>
    </div>
  </body>
</html>
```

4. Cree un archivo llamado operacionesBasicas.js y guárdelo en la carpeta js. Ahora proceda a editarlo con el siguiente código.

```
//Accedemos al parrafo que nos permitira imprimir el resultado
const parrafo = document.querySelector("#idParrafo");
console.log(parrafo);

//Accedemos a cada boton por medio de la API DOM
const btnSumar = document.querySelector("#idBtnSumar");
const btnRestar = document.querySelector("#idBtnRestar");
const btnMultiplicar = document.querySelector("#idBtnMultiplicar");
const btnDividir = document.querySelector("#idBtnDividir");

//Agregamos el evento click a los botones, adicionalmente
//se le asigna la funcion que realizará la operación
btnSumar.addEventListener("click", sumar);
btnRestar.addEventListener("click", restar);
btnMultiplicar.addEventListener("click", multiplicar);
btnDividir.addEventListener("click", dividir);

//Creamos la variable que tendra el valor del resultado de la operación matematica
let resultado;

//Funcion de operaciones
function sumar() {
  let numero1 = parseFloat(prompt("Ingrese el primer numero a sumar"));
  let numero2 =  parseFloat(prompt("Ingrese el segundo numero a sumar"));
  resultado = numero1 + numero2;
  parrafo.innerHTML = `${numero1} + ${numero2} = ${resultado}`;
}
```

```
function restar() {
  let numero2 =  parseFloat(prompt("Ingrese el segundo numero a restar"));
  resultado = numero1 - numero2;
  parrafo.innerHTML = `${numero1} - ${numero2} = ${resultado}`;
}

function multiplicar() {
  let numero1 =  parseFloat(prompt("Ingrese el primer numero a multiplicar"));
  let numero2 =  parseFloat(prompt("Ingrese el segundo numero a multiplicar"));
  resultado = numero1 * numero2;
  parrafo.innerHTML = `${numero1} x ${numero2} = ${resultado}`;
}

function dividir() {
  let numero1 =  parseFloat(prompt("Ingrese el primer numero a dividir"));
  let numero2 =  parseFloat(prompt("Ingrese el segundo numero a dividir"));
  let mensaje;
  if (numero2 != 0) {
    resultado = numero1 / numero2;
    mensaje = `${numero1} / ${numero2} = ${resultado}`;
  } else {
    mensaje = `El valor ${numero1} / ${numero2} no se puede dividir`;
  }

  parrafo.innerHTML = mensaje;
}
```

5. Proceda a actualizar en su navegar la página index.html, tendría que tener el siguiente resultado. Luego proceda a seleccionar la opción “Operaciones básicas”.



6. Aparentemente nuestra página “operacionesBasicas.html” funciona perfectamente. Pero si le damos clic a cualquier botón, no realizara ninguna acción. (vea la consola de errores)
Esto se debe a que en nuestra etiqueta <script src="../js/operacionesBasicas.js"></script> no hemos agregado el atributo defer. Agregue dicho atributo y revise si se ha corregido el error.

III. EJERCICIOS COMPLEMENTARIOS

1. Modifique el ejemplo III sobre números aleatorios de forma que cada vez que un usuario falle al adivinar el número, el programa le diga si el número que busca adivinar es más alto o más bajo.