AIEP de la Universidad Andrés Bello

Certified B Corporation

MÓDULO 2:
Fundamentos de Desarrollo Front-End.

Parte 5: JavaScript.

Aprendizajes Esperados

Utilizar código Javascript para la personalización de eventos sencillos dentro de un documento html dando solución al problema planteado.

Contenidos

I. Bases del Lenguaje JavaScript.

1. Breve historia de JavaScript.
2. Relevancia de Javascript.
3. Qué puede y no puede hacer en el contexto de un navegador.
4. Cómo incorporar código Javascript en un documento html.
5. Selectores básicos: getElementByld.
6. Obtención y manipulación de valores y textos de los elementos del DOM.
7. Eventos básicos: onClick y onChange.
8. Variables.
9. Expresiones aritméticas.
10. Sentencias condicionales.
11. Funciones.
12. Cómo ejecutar código Javascript en la consola.
13. Depurando el código Javascript con la consola.

I. Bases del Lenguaje JavaScript.

1. 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 enc regular el lenguaje.

I. Bases del Lenguaje JavaScript.

2. 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).

ECMAScript

I. Bases del Lenguaje JavaScript.

3. 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.

ECMAScript

I. Bases del Lenguaje JavaScript.

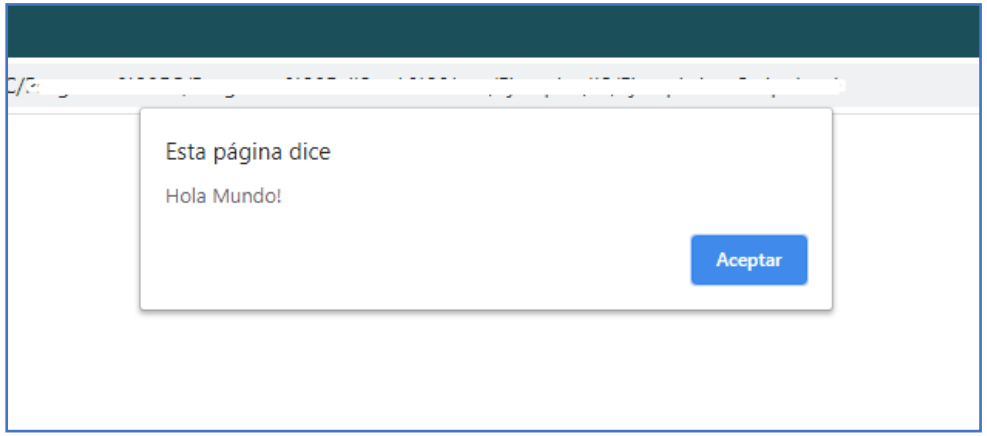
4. Cómo incorporar código Javascript en un documento html.

Para agregar JavaScript a nuestra pagina 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>
```



I. Bases del Lenguaje JavaScript.

5. Selectores básicos: getElementByld.

- 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 pagina.
- Utiliza la propiedad id para enlazar el elemento a intervenir.
- Este método funciona una vez cargada toda la pagina 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.

I. Bases del Lenguaje JavaScript.

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

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

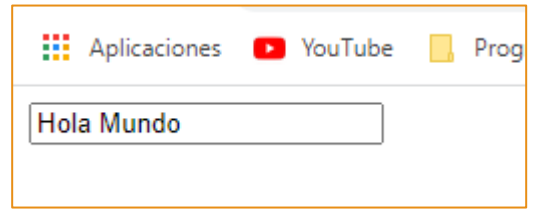
```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>JavaScript</title>
</head>
<body>
<input type="text" id="txtValor" value="Chao Mundo">
<script>
    alert(document.getElementById("txtValor").value);
</script>
</body>
</html>
```

I. Bases del Lenguaje JavaScript.

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

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

```html
<!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);*/
        document.getElementById("txtValor").value = "Hola Mundo";
    </script>
</body>
</html>
```



I. Bases del Lenguaje JavaScript.

7. Eventos básicos: onClick y onChange.

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

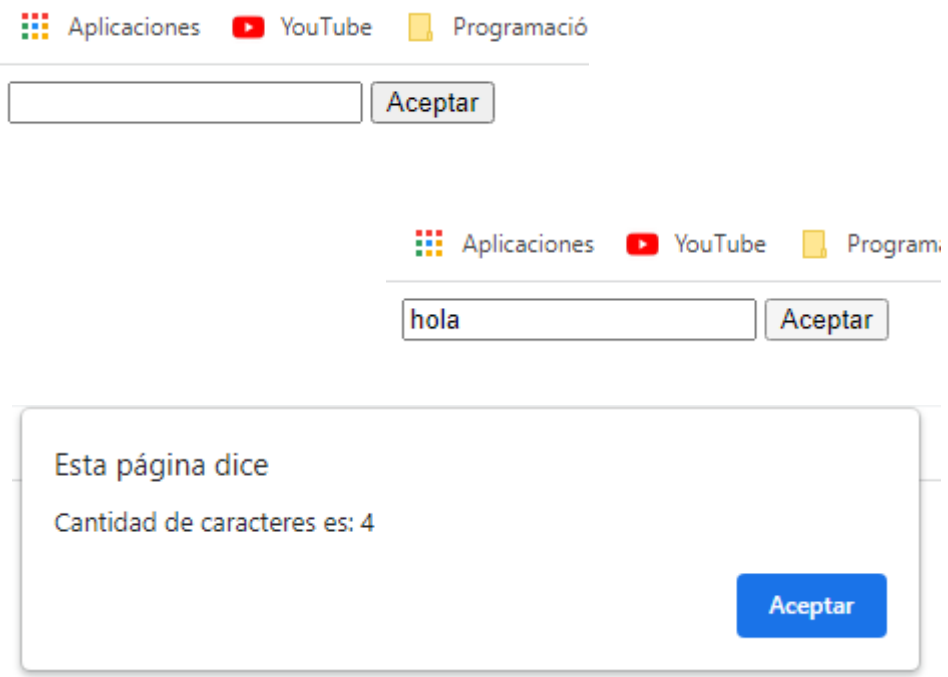
El evento **onChange**, en cambio, se ejecuta cuando el usuario realiza algún cambio en un elemento de la pagina.

I. Bases del Lenguaje JavaScript.

7. Eventos básicos: onClick y onChange.

Ejemplo código onClick:

```html
<!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>
```



I. Bases del Lenguaje JavaScript.

8. Variables.

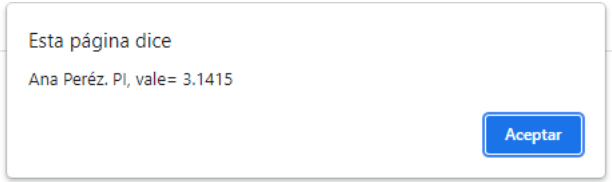
- Las variables permiten almacenar valores temporalmente en JS.
- Existen 3 maneras de declarar variables, usando **var, let** o **const**.
- Las variables definidas con **var**, tienen un alcance global dentro de una rutina a diferencia de **let** y **const**, que solamente están disponible en un bloque 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.

I. Bases del Lenguaje JavaScript.

8. Variables.

Ejemplo:

```html
<!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>
```



I. Bases del Lenguaje JavaScript.

9. Expresiones aritméticas.

Para realizar operaciones aritméticas en JavaScript, se utilizan las siguientes simbología:

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.

I. Bases del Lenguaje JavaScript.

9. Expresiones aritméticas.

Ejemplo:

```
<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>
```

I. Bases del Lenguaje JavaScript.

10. 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

I. Bases del Lenguaje JavaScript.

10. Sentencias condicionales.

Ejemplo:

```
<!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>
```

I. Bases del Lenguaje JavaScript.

10. Sentencias condicionales.

Ejemplo:

```
<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>
```

I. Bases del Lenguaje JavaScript.

11. Funciones.

Las funciones son un conjunto de sentencias agrupadas bajo un nombre, que realizan cálculos o tareas especificas. 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**.

I. Bases del Lenguaje JavaScript.

11. Funciones.

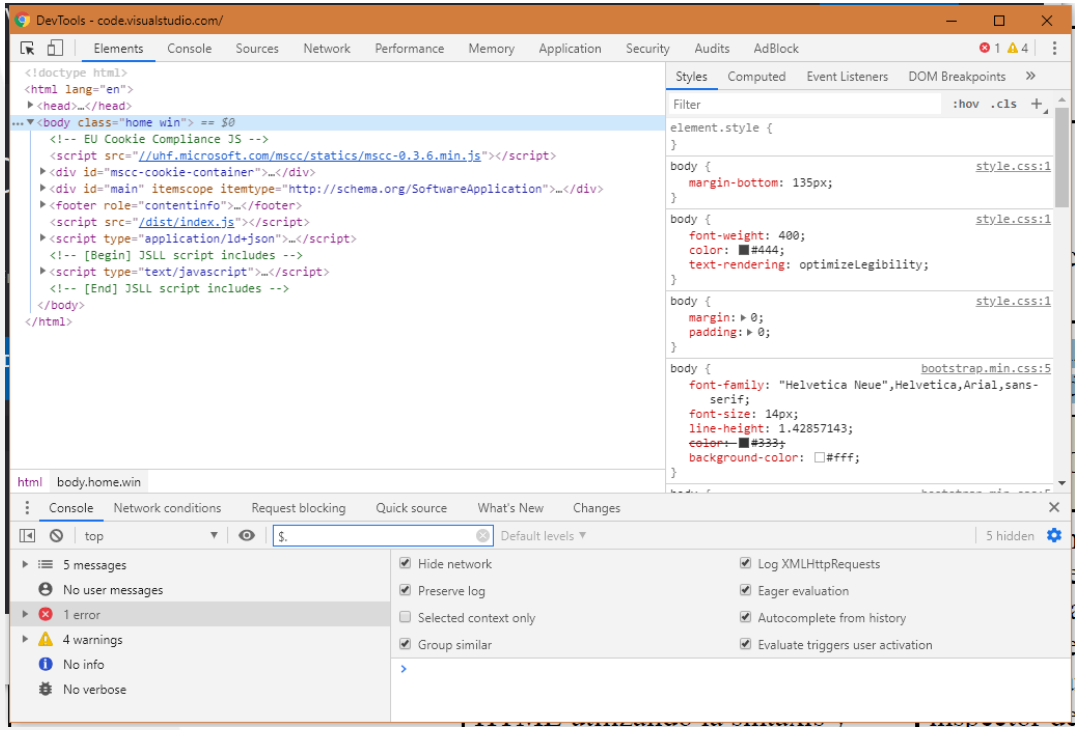
Ejemplo:

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

I. Bases del Lenguaje JavaScript.

12. 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:

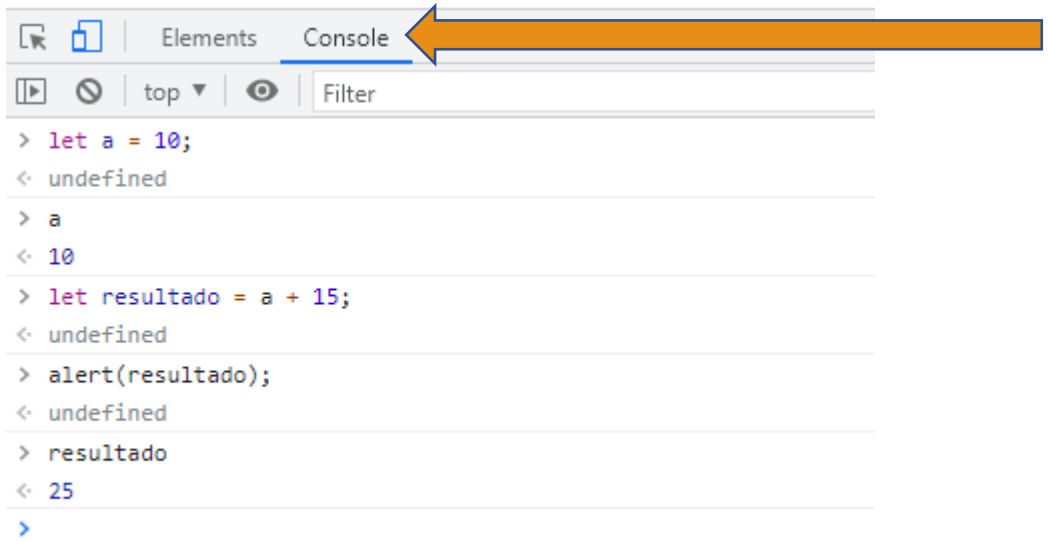


I. Bases del Lenguaje JavaScript.

12. Cómo ejecutar código JavaScript en la consola.

- 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().

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



I. Bases del Lenguaje JavaScript.

13. 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.

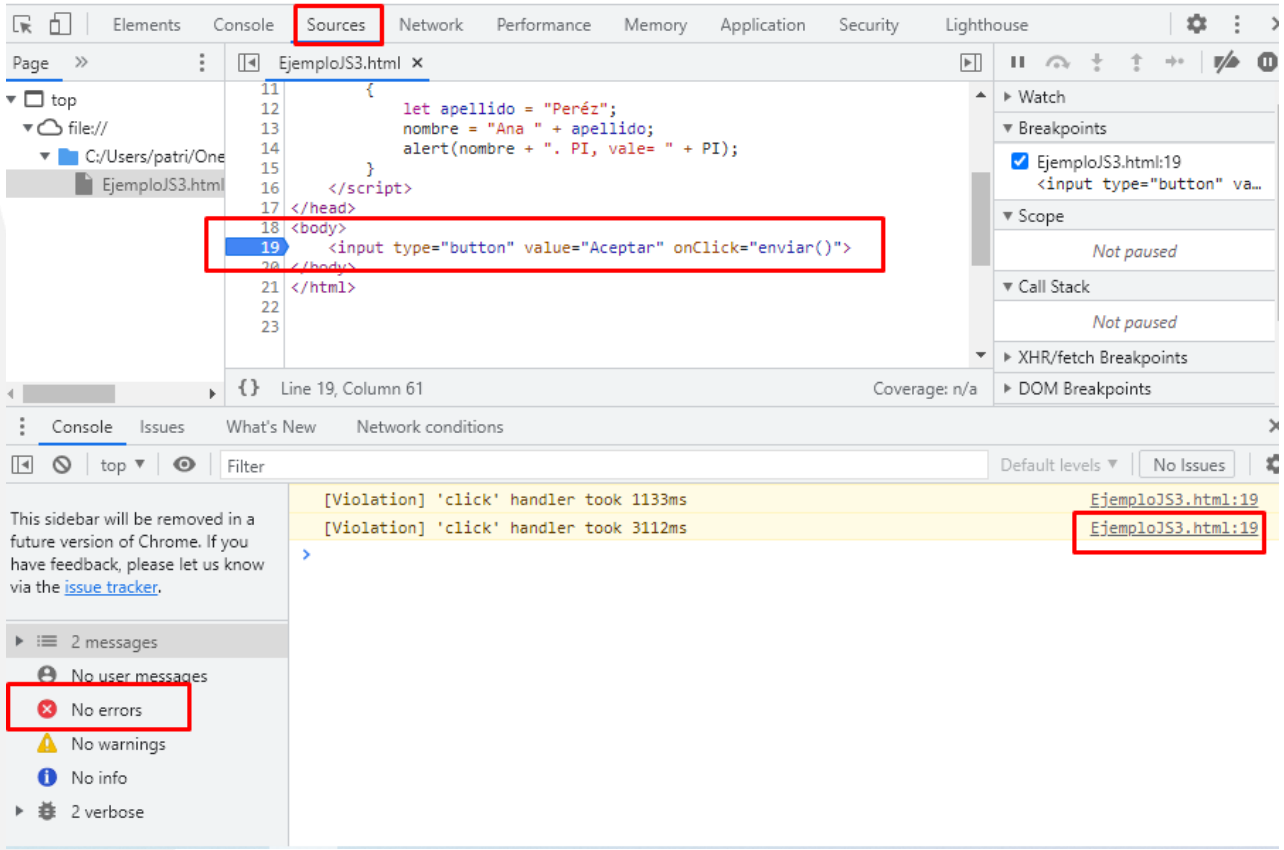
La depuración se activa en la opción “Sources” de DevTools (usando Chrome), el cual, mostrará:

- 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.

I. Bases del Lenguaje JavaScript.

13. Depurando el código Javascript con la consola.

Ejemplo:



Actividades.