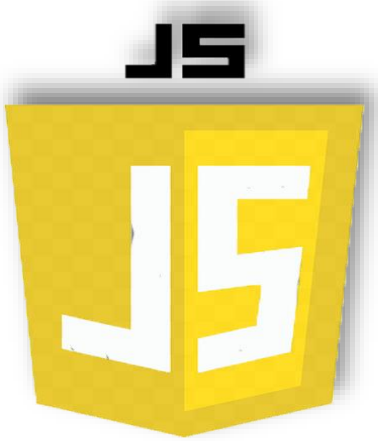
Programación

JavaScript



+

Fundamentos de la programación

Se trata de un **lenguaje de programación tipo script**, basado en objetos y guiado por eventos, diseñados específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet.

Los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

JavaScript es un **lenguaje de programación interpretado**, por lo que no es necesario compilar los programas para ejecutarlos.
Tema de debate:

https://nodejs.dev/learn/the-v8-javascript-engine
https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino/JavaScript_Compiler

Alguna características

No se tienen en cuenta espacios y saltos de línea.
Case-Sensitive (se distinguen mayúsculas y minúsculas)
No se define el tipo de variables
No es necesario terminar sentencias con ; (pero sí es conveniente hacerlo)
Se pueden incluir comentarios:

Tipo	Descripción
//	Comentario en una sola línea
/* hola mundo hola mundo*/	Comentario multilínea

/* no puedes, sin embargo, /* anidar comentarios */ SyntaxError */

Formas de referenciar:

- Dentro de las etiquetas <script></script>

```
<script>
        //Código JavaScript
</script>
```

- Link a un archivo externo

```
<script src="NombreArchivo.js></script>
```

Ver más del tema en:

https://tc39.es/ecma262/

JavaScript puede ir tanto en el HEAD, como en el BODY

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript en el body</h2>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Mi primer JavaScript";
</script>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "Hola Mundo!!!";
}
</script>
</head>

<body>

<h2>JavaScript en el Head</h2>

<p id="demo">Hola....</p>

<button type="button" onclick="myFunction()">Cambiar párrafo</button>

</body>
</html>
```

Tipos de datos

Tipo de dato	Descripción
Boolean	TRUE - FALSE
Null	Una palabra clave especial que denota un valor nulo.
undefined.	Una propiedad de alto nivel cuyo valor no es definido.
Number.	Un número entero o un número con coma flotante. Por ejemplo: 42 o 3.14159.
String.	Una secuencia de caracteres que representan un valor "Hola"

Symbol (nuevo en ECMAScript 6). Un tipo de dato cuyas casos son únicos e inmutables y Object.

Variables.

Las variables se usan como nombres simbólicos para valores en tu aplicación. Los nombres de las variables, llamados identificadores, se rigen por ciertas reglas.

- Ej. Declaración y asignación de valor.
  - var mi_dato_numerico=3;
  - Var mi_dato_texto="hola";

Ámbito de variable (global-local)

- Cuando declaras una variable fuera de una función, se le denomina variable global, porque está disponible para cualquier otro código en el documento actual. Cuando declaras una variable dentro de una función, se le denomina variable local, porque está disponible sólo dentro de esa función donde fue creada.

- Antes de ECMAScript 6 Javascript no tiene ámbito de sentencias de bloque; más bien, una variable declarada dentro de un bloque es local para la función (o ámbito global) en la que reside el bloque

Operadores matemáticos

Símbolo	Operador	Descripción
+	Suma	Suma dos números
-	Resta	Resta dos números
*	Multiplicación	Multiplica dos números
/	División	Divide dos números
%	IModulo	Devuelve el resto de dividir de dos números
++	Incremento	Suma 1 valor al contenido de una variable
--	Decremento	Resta un valor al contenido de una variable

Veamos un ejemplo

```
var edadGrover, edadPablo, diferenciaEdad, sumaEdades, yearActual, yearGrover, yearPablo;

edadGrover = 34;
edadPablo = 28;
yearActual = 2019;

diferenciaEdad = edadGrover - edadPablo;
sumaEdades = edadGrover + edadPablo;

yearGrover = yearActual - edadGrover;
yearPablo = yearActual - edadPablo;

console.log(diferenciaEdad);
console.log(sumaEdades);
```

Operadores Lógicos

Operador	Descripción	Ejemplo
Igualdad (==)	Devuelve **Verdadero (true)** si ambos operandos son iguales.	3 == var1 "3" == var1
Desigualdad(!=)	Devuelve **Verdadero (true)** si ambos operandos no son iguales.	var1 != 4 var2 != "3"
Mayor que (>)	Devuelve **Verdadero (true)** si el operando de la izquierda es mayor que el operando de la derecha.	var2 > var1 "12" > var1
Mayor o igual que (>=)	Devuelve **Verdadero (true)** si el operando de la izquierda es mayor o igual que el operando de la derecha.	var2 >= var1 var1 >= 3
Menor que (<)	Devuelve **Verdadero (true)** si el operando de la izquierda es menor que el operando de la derecha.	var1 < var2 "2" < 12
Menor o igual que (<=)	Devuelve **Verdadero (true)** si el operando de la izquierda es menor o igual que el operando de la derecha.	var1 <= var2 var2 <= 5

Operador condicional (ternario)

El operador condicional es el único operador de JavaScript que necesita tres operandos. El operador asigna uno de dos valores basado en una condición. La sintaxis de este operador es:

condición ? valor1 : valor2

Si la condición es true, el operador tomará el valor1, de lo contrario tomará el valor2. Puedes usar el operador condicional en cualquier lugar que use un operador estándar.

Por ejemplo,

```
var estado = (edad >= 18) ? "adulto" : "menor";
```

Esta sentencia asigna el valor adulto a la variable estado si edad es mayor o igual a 18, de lo contrario le asigna el valor menor.

Operador coma

El operador coma (,) simplemente evalúa ambos operandos y retorna el valor del último. Este operador es ante todo utilizado dentro de un ciclo for, permitiendo que diferentes variables sean actualizadas en cada iteración del ciclo.

Por ejemplo, si a es un Array bi-dimensional con 10 elementos en cada lado, el siguiente código usa el operador coma para actualizar dos variables al mismo tiempo. El código imprime en la consola los valores correspondientes a la diagonal del Array:

```
for (var i = 0, j = 9; i <= j; i++, j--)
  console.log("a[" + i + "][" + j + "]= " + a[i][j]);
```

La Consola

La consola es un panel que muestra mensajes importantes, como errores, para desarrolladores. Gran parte del trabajo que hace la computadora con nuestro código es invisible para nosotros por defecto. Si queremos que las cosas aparezcan en nuestra pantalla, podemos imprimir o iniciar sesión en nuestra consola directamente.

En JavaScript, la palabra clave ***console*** se refiere a un objeto, una colección de datos y acciones, que podemos usar en nuestro código. Las palabras clave son palabras que están incorporadas en el lenguaje JavaScript, por lo que la computadora las reconocerá y las tratará especialmente.

Una acción, o método, que está integrado en el objeto de la consola es el método .log (). Cuando escribimos console.log (), lo que ponemos dentro de los paréntesis se imprimirá, o se registrará, en la consola.

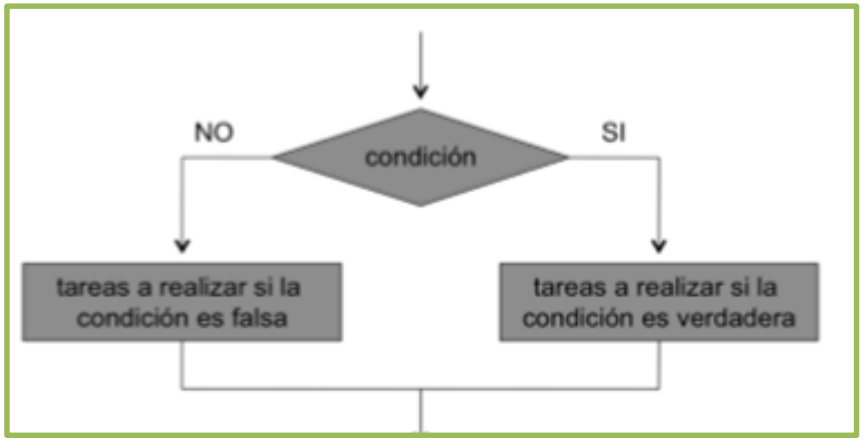
Sentencias

Es un conjunto de comandos que se ejecutan para ver si se cumple, o no, cierta condición.

Sentencia if...else

Se utiliza la sentencia if para comprobar si la condición lógica es verdadera. Se utiliza la opción else para ejecutar un sentencia si la condición es falsa. A continuación se muestra un ejemplo de if...else:

```
if (condición) {
    sentencia_1;
 } else {
    sentencia_2;
}
```



Switch

Una sentencia switch permite a un programa evaluar una expresión e intentar igualar el valor de dicha expresión a una etiqueta de caso (case). Si se encuentra una coincidencia, el programa ejecuta la sentencia asociada. Una sentencia switch se describe como se muestra a continuación:

```
switch (expresión) {
 case etiqueta_1:
   sentencias_1
    [break;]
  case etiqueta_2:
   sentencias_2
•      [break;]
•        ...
•      default:
•        sentencias_por_defecto
•        [break;]
•    }
```

Explicación

El programa primero busca una cláusula case con una etiqueta que coincida con el valor de la expresión y, entonces, transfiere el control a esa cláusula, ejecutando las sentencias asociadas a ella. Si no se encuentran etiquetas coincidentes, el programa busca la cláusula opcional default y, si se encuentra, transfiere el control a esa cláusula, ejecutando las sentencias asociadas. Si no se encuentra la cláusula default, el programa continúa su ejecución por la siguiente sentencia al final del switch. Por convención, la cláusula por defecto es la última cláusula, aunque no es necesario que sea así.

La sentencia opcional break asociada con cada cláusula case asegura que el programa finaliza la sentencia switch una vez que la sentencia asociada a la etiqueta coincidente es ejecutada y continúa la ejecución por las sentencias siguientes a la sentencia switch. Si se omite la sentencia break, el programa continúa su ejecución por la siguiente sentencia que haya en la sentencia switch.

Estructuras repetitivas

Las **estructuras repetitivas o cíclicas** nos permiten ejecutar varias veces un conjunto de instrucciones. A estas repeticiones se las conoce con el nombre de ciclos o bucles.

Estructuras repetitivas en JavaScript:

- FOR
- WHILE
- DO WHILE

Sentencia for

Un bucle for se repite hasta que la condición especificada se evalúa como false.

for ([expresionInicial]; [condicion]; [expresionIncremento])

Explicación

Cuando un bucle for se ejecuta, ocurre lo siguiente:

La expresión de inicialización expresionInicial, si existe, se ejecuta. Esta expresión habitualmente inicializa uno o más contadores del bucle, pero la sintaxis permite una expresión con cualquier grado de complejidad. Esta expresión puede también declarar variables.

Se evalúa la expresión condición. Si el valor de condición es true, se ejecuta la sentencia del bucle. Si el valor de condición es false, el bucle for finaliza. Si la expresión condición es omitida, la condición es asumida como verdadera.

Se ejecuta la sentencia. Para ejecutar múltiples sentencias, use un bloque de sentencias ({ ... }) para agruparlas.

Se ejecuta la expresión expresionIncremento, si hay una, y el control vuelve al paso 2.

Para ver el ejemplo, ingresa al siguiente link:

https://www.w3schools.com/js/tryit.asp?filename=tryjs_loop_for

Sentencia while

Una sentencia while ejecuta sus sentencias mientras la condición sea evaluada como verdadera. Una sentencia while tiene el siguiente aspecto:

while (condición)

Si la condición cambia a falsa, la sentencia dentro del bucle deja de ejecutarse y el control pasa a la sentencia inmediatamente después del bucle.

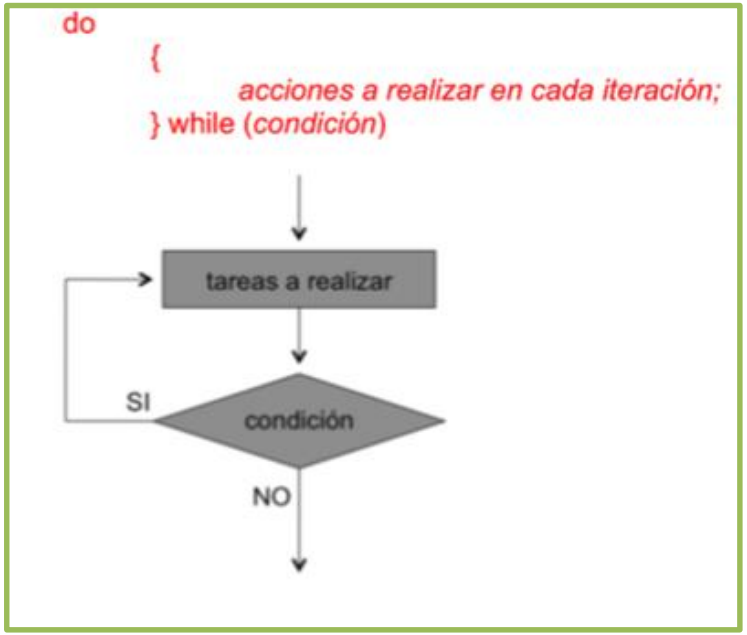
La condición se evalúa antes de que la sentencia contenida en el bucle sea ejecutada. Si la condición devuelve verdadero, la sentencia se ejecuta y la condición se comprueba de nuevo. Si la condición es evaluada como falsa, se detiene la ejecución y el control pasa a la sentencia siguiente al while.

Para ejecutar múltiples sentencias, use un bloque de sentencias ({ ... }) para agruparlas.

Sentencia While-do

Se utiliza para repetir instrucciones un número indefinido de veces, hasta que se cumpla una condición.

A diferencia de la estructura mientras (while), la estructura hasta (do while) se ejecutará al menos una vez.



Sentencia break

Use la sentencia break para salir de un bucle, switch, o en conjunto con una sentencia label.

Cuando use break sin un label, finaliza inmediatamente el código encerrado en while, do-while, for, o switch y transfiere el control a la siguiente sentencia.

Cuando usted use break con un label, termina la sentencia especificada por label.

La sintaxis de la sentencia break es la siguiente:

```
break;
```

Sentencia continue

La sentencia continue puede usarse para reiniciar una sentencia while, do-while, for, o label.

Cuando use continue sin un label, este termina la iteración en curso del código encerrado en una sentencia while, do-while, o for y continúa la ejecución del bucle con la siguiente iteración. A diferencia de la sentencia break, continue no termina completamente la ejecución del bucle.

La sintaxis de la sentencia continue es la siguiente:

- continue;
- continue label;
-

Funciones

Es un **conjunto de instrucciones** o sentencias que se agrupan para realizar una tarea concreta y que se pueden reutilizar fácilmente.

Realizan varias operaciones invocando un nombre. Esto puede **simplificar el código**.

Puedes crear tus propias funciones y usarlas cuando sea necesario.

La declaración de una función consiste:

- Un nombre
- Una lista de parámetros o argumentos encerrados entre paréntesis.
- Conjunto de sentencias JavaScript encerrada entre llaves.

Ejemplo:

```
function nombre (parámetro1, parámetro2)
{
codigo ha ser ejecutado;
}
```

Para más información consulte:
https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Funciones

Veamos un ejemplo

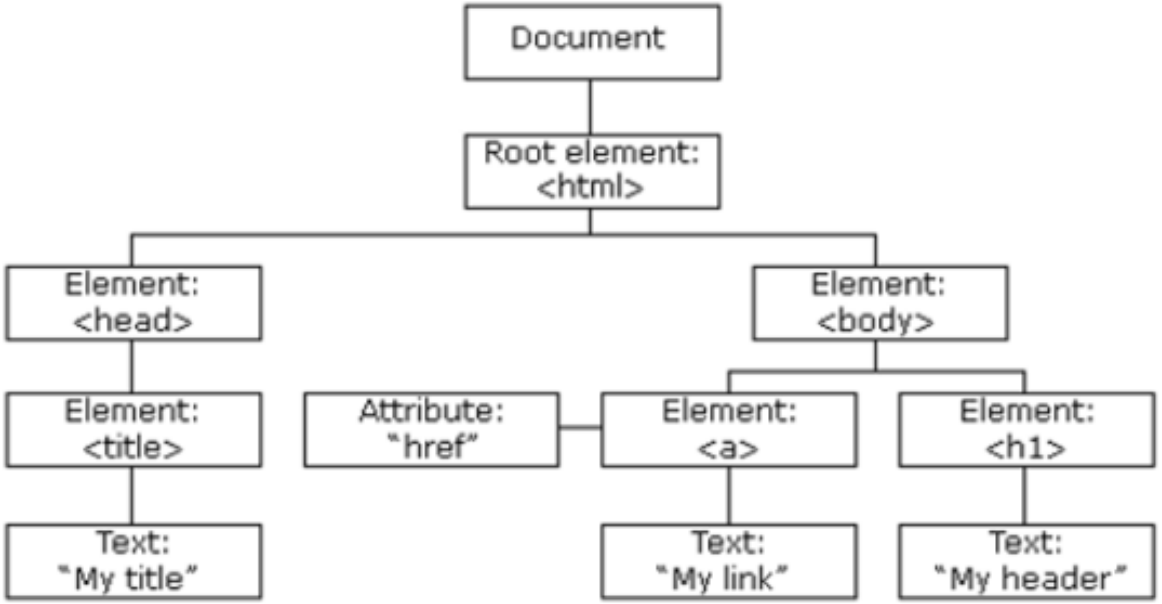
```
1    function bienvenido(){
2        return 'Hola Bienvenido a la sección de funciones';
3    }
4    var mensaje = bienvenido();
5    console.log(mensaje);
```

El DOM Define:

Los elementos HTML como **objetos**.
Las **propiedades** de todos los elementos HTML.
Los **métodos** para acceder a todos los elementos.
Los **eventos** para todos los elementos.

DOM es la forma de obtener, cambiar, agregar o borrar elementos HTML

HTML DOM (Document Object Model)



Es un modelo de objetos de documento (DOM) del W3C es una plataforma y una interface de lenguaje-neutral que permite crear programas y scripts para acceder y actualizar el contenido de una página web dinámicamente.

Tipos de nodos

- **Document**, nodo raíz del que derivan todos los demás nodos del árbol.
- **Element**, representa cada una de las etiquetas HTML. Se trata del único nodo que puede contener atributos y el único del que pueden derivar otros nodos.
- **Attr**, se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas HTML, es decir, uno por cada par atributo=valor.
- **Text**, nodo que contiene el texto encerrado por una etiqueta HTML.

Document

Representa la página web.

Por ende, si deseas acceder a cualquier elemento de tu página web, debes primero acceder a document.

Métodos para encontrar elementos

Métodos que se utiliza para buscar un elementos o etiquetas HTML y permiten cambiar las propiedades de dicho elemento.

getElementById (identificador).
getElementsByTagName(nombre).
getElementsByTagClass(nombre).

Modificar elementos de HTML

element.innerHTML = nuevo_contenido	Cambia el contenido de un elemento HTML
element.attribute = nuevo_valor	Cambia el valor del atributo de un elemento HTML
element.setAttribute(atributo, valor)	Cambia el valor del atributo de un elemento HTML
element.style.property = nuevo_estilo(CSS)	Cambia el estilo de un elemento HTML.

Eventos

Los eventos hacen posible que el usuario **interactúe** con el programa.

Cada elemento HTML tiene una lista de eventos que se le puede asignar.

El mismo evento puede ser asignado a varios elementos HTML.

Veamos una ejemplo

En HTML

```
<body>
  <header id="main-header" class="bg-info text-white p-4 mb-3">
    <div class="container">
      <h1 id="header-title">Lista de Items</h1>
    </div>
```

En JavaScript

```
var headerTitle = document.getElementById('header-title');
var header = document.getElementById('main-header');
//console.log(header);
headerTitle.textContent = 'Hola';
```

- onchange : se modificó un elemento HTML
- onclick : el usuario hizo click en un
- elemento HTML onmouseover /
- onmouseout : el usuario mueve el mouse sobre / fuera de un elemento HTML
- onkeydown : el usuario presiona una tecla
- onload : el navegador terminó de cargar la página

Veamos un ejemplo

Actividad:

A fin de interactuar con el usuario y, utilizando javascript, agrega en tu sitio web una adivinanza (relacionada al tema del sitio) con hasta 4 intentos.

El apartado de adivinanza, deberá:

- contener un cuadro texto y un botón
- facilitar al usuario 2 pistas si el usuario no acierta en el intento 2 y 3.
- informar en todo momento al usuario el nro intentos que le quedan.
- informar el resultado final (si acierta o si perdió los 4 intentos).

(opcional) usar los componentes de Alertas de Bootstrap 4 para informar al usuario.

Se valorará:

- inclusión de todos los ítems mencionados en la consigna
- aplicación de los contenidos teóricos desarrollados
- creatividad en la implementación de la actividad
- funcionalidad del sitio web responsive solicitado
- coherencia
- prolijidad