JavaScript

Agregar javascript a una página web.

Modo interno

Se añade en la cabecera (head) del documento html lo siguiente:

```
<script>
        // JavaScript va aquí
</script>
```

Modo externo

En head añadimos la siguiente línea:

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

script.js es un documento con nuestro código javascript.

Comentarios

```
// This is an in-line comment.

/* This is a
multi-line comment */
```

Variables

Javascript es un lenguaje de **tipado dinámico**, es decir que **no hace falta definir el tipo de una variable**. Ya que la variable **recibe el tipo en el momento que se le asigna un valor**.

Declarar variables

Podemos declarar variables usando var o let.

```
var myVar;
var myName = "your name";

let camper = "James";
let numero = 5;
```

La diferencia entre var y let es que con este último solo se puede declarar una variable con un determinado nombre una sola vez, con var podemos declarar dos variables con el mismo nombre, lo que nos puede llevar a error.

Nota: Var ya no se usa, usar siempre let o const para declarar variables.

Nota: JavaScript es case sensitive, esto significa que se distingue entre mayúsculas y minúsculas. Una buena práctica es escribir los nombres en camelCase.

Variables con valor constante

Para declarar una variable cuyo valor no queremos que cambie una vez inicializada con un valor.

```
const FAV_PET = "Cats";
```

Inicializar variables

Cuando declaramos una variable en Javascript tiene un valor inicial undefined.

Podemos dar un valor inicial justo al crear la variable.

Tipos de variables

Sobre las variables podemos definir **9 tipos variables** Javascriptque son los tipos básicos del lenguaje:

- Boolean, variables que puedan contener un valor de verdadero o falso. (true o false).
- null, en el caso de que la variable no tome ningún valor.
- undefined, cuando la variable no ha sido inicializada tiene un valor indefinido.
- Number, para los números enteros o decimales. (8 o 3.1416)
- BigInt, es un tipo para almacenar números grandes más allá del tamaño de Number.
- String, para las cadenas de texto. (“Soy una cadena”)
- Symbol, para valores que sean únicos e inmutables.
- Object, para cuando la variable contiene un objeto.
- Function, podremos asignar a una variable una función.

El **tipo de dato de la variable se calcula atendiendo al valor que le asignemos a dicha variable**. El **tipo de una variable puede cambiar a lo largo de la vida de un programa Javascript** atendiendo a los valores que se le vayan asignando

Con las **conversiones dinámicas** el tipo de la variable irá cambiando, por lo que será necesario en ciertos momentos saber el tipo de una variable javascript en un determinado momento.

Para ello tenemos el operador typeof cuya estructura es la siguiente: **typeof** operando;

https://www.manualweb.net/javascript/operadores-javascript/

Operadores aritméticos

Operador	Descripción	Ejemplo
`+`	Suma	`6 + 9`
`-`	Resta	`20 - 15`
`*`	Multiplicación	`3 * 7`
`/`	División	`10 / 5`

https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/A_first_splash

Incremento y decremento con ++ , -- .

i++ ==> i = i+1;

i-- ==> i = i-1;

Operador += , -= , *=.

Sumar: `myVar = myVar + 5; es equivalente a lo siguiente:`

```
let myVar = 1;
myVar += 5;
console.log(myVar);  resultado = 6;
```

restar: `myVar = myVar - 5 equivalente a myVar = myVar - 5;`

`multiplicar:` `myVar = myVar * 5; equivale a myVar *= 5;`

Operador para obtener el resto de una división: %

5 % 2 = 1

17 % 2 = 1

Cadenas de texto

Las cadenas de texto se definen usando comillas dobles o simples.

Operador para concatenar cadenas

Puedes usar el operador **+** para unir cadenas de texto.

Tambien podemos usar **+=** para concatenar cadenas ejemplo:

```
let ourStr = "I come first. ";
ourStr += "I come second.";

      ourStr now has a value of the string "I come first. I come second."
```

Secuencias de escape en Strings.

Code	Output
\'	single quote
\"	double quote
\\	backslash
\n	newline
\t	tab
\r	carriage return
\b	backspace
\f	form feed

Longitud de una cadena.

Cadena.length

Ejemplo:

```
console.log("Alan Peter".length);        resultado = 10;
```

Primer y último carácter en una cadena u otro cualquiera.

```
Nota: en Javascript el index comienza en cero.

Primer carácter:       miCadena[0];

Segunda letra:         miCadena[1];

último carácter:       miCadena[miCadena.length - 1];

tercer carácter por el final:      miCadena[miCadena.length - 3];
```

Inmutabilidad en cadenas.

No podemos cambiar una letra de la cadena usando la notación de brakets, pero si podemos cambiar la cadena asignando un nuevo valor a la variable.

```
No se puede:      myStr[0] = "j";   da error.

Si se puede:      myStr = "nuevo valor";
```

JavaScript Arrays

Crear arrays.

Ejemplo:
```
const sandwich = ["peanut butter", "jelly", "bread"];

const teams = [["Bulls", 23], ["White Sox", 45]];  Multidimensional
```

Acceso a los arrays mediante indice.

Nota: Array index comienza por cero.

Ejemplo:

```
const array = [50, 60, 70];

console.log(array[0]);              resultado en consola: 50

const data = array[1];              resultado en consola: 60
```

En arrays multidimensionales es igual, solo que indicamos tantos [indice] como dimensiones tiene el array.

Modificar arrays.

```
Ejemplo:

const ourArray = [50, 40, 30];
ourArray[0] = 15;

Ahora la vble ourArray contiene los valores [15,40,30]
```

Manipular Arrays con PUSH.

Para **añadir elementos al final**, podemos pasar varios argumentos y la función Push devuelve la longitud una vez añadidos los elementos.

Ejemplo:

```
const arr1 = [1, 2, 3];

arr1.push(4, 5);

arr1 ahora contiene los valores [1, 2, 3, 4, 5]
```

Método POP.

Elimina el último elemento de un array y devuelve dicho elemento.

Ejemplo:

```
const threeArr = [1, 4, 6];
const oneDown = threeArr.pop();
console.log(oneDown);                    muestra el valor 6
console.log(threeArr);                   muestra el valor [1,4]
```

Método SHIFT.

Eliminar el primer elemento.

Método UNSHIFT.

Añadir elementos al principio del array.

```
function functionName() {
      console.log("Hello World");
}
```

Función con argumentos:

```
function testFun(param1, param2) {
      console.log(param1, param2);
}
```

Función que devuelve un valor:

```
function plusThree(num) {
      return num + 3;
}

const answer = plusThree(5);
```

Global vs. Local variables en Funciones

Es posible tener una variable local y otra global con el mismo nombre, cuando hacemos esto la variable local tiene preferencia sobre la variable global.

Ejemplo:

```
const someVar = "Hat";

function myFun() {
  const someVar = "Head";
  return someVar;
}
```

The function myFun will return the string Head because the local version of the variable is present