JS-I

Introducción a JavaScript

JavaScript es un lenguaje de programación que fue creado originalmente para ser usado en el front-end de una página web.

La idea original era poder dar dinamismo a las páginas webs, que en un principio eran estáticas.

La introducción del "motor V8" de Google ha mejorado la velocidad y el funcionamiento de JS. Haciendo que JS (javascript) sea la lengua franca de la web, llegando inclusive al Back-End a través de NodeJS.

Variables

Una variable es una forma de almacenar el valor de algo para usar más tarde.

JavaScript tiene tipado dinámico, esto es, las variables se pueden configurar y reestablecer para cualquier tipo de dato.

Para crear una variable en JavaScript utilizamos la palabra clave var, seguida de un espacio y el nombre de la variable (con este nombre, llamado identificador, podremos hacer referencia a ella luego).

Además de declarar una variable, podemos asignarle un valor usando el signo =.

```
var identificador = valor
```

Los identificadores pueden iniciar con una letra, un guión bajo (_) o signo pesos ($), además de que puede contener también números.

Los siguientes son nombres válidos de variables: Nombre_usuario, Email002, $ahorros y _direccion.

También, JS distingue entre mayúsculas y minúsculas para nombrar variables, así que Nombre y nombre serían dos nombres de variables DIFERENTES.

Formas de declarar una variable:

```
1    var nombre = 'Juan'; // Vamos a usar principalemente esta forma
2    let apellido = 'Perez';
3    const comidafavorita = 'Pizza';
```

Tipos de datos

Strings

Son bloques de texto los cuales siempre se definirán entre comillas, ya sean simples (' ') o dobles (" ").

```
1 var nombrePerro = 'firulais';
2 var usuario = 'Alejo123';
3 var emailUsuario = "emailusuario@mail.com";
4 var password = 'EhrthbSDTG5ytrVBDFn%/89'
```

Numbers

Son los números tal cual los conocemos. NO se escriben entre comillas. Pueden ser tanto negativos como positivos.

```
1 var edad = 25;
2 var gradosCentigrados = -40;
3 var longitud = 25.785;
```

Booleans

Los booleanos provienen de la lógica de Boole.
Esencialmente, significa que tiene dos opciones: verdadero (*true*) o falso (*false*).

```
1 var meEncantaJavascript = true;
2 var soyMalProgramador = false;
```

Undefined

Obtenemos undefined cuando busques *algo* que no existe, como una variable que aún no tiene un valor. undefined simplemente significa que lo que estás pidiendo no existe.

```
1 var hola; //undefined
```

Null

Se usa para establecer que el elemento que estamos buscando existe, pero no hay ningún valor asociado a este.

```
1 var numeroTelefono = '11-1234-5678';
2 numeroTelefono = null;
3
4 numeroTelefono; // null
```

undefined vs null

El valor undefined está configurado por JS, mientras que null lo configuramos nosotros para indicar que no hay un valor específico para esa variable, pero esta sí existe.

Operadores

Son símbolos que le indican al intérprete de JS las operaciones que debe realizar.

```
2 + 3 = 5
3 / 3 = 1
2 * 2 = 4
7 - 4 = 3
2 ** 3 = 8 (potenciación)
```

Módulo (%)

Este dividirá los dos números entre los que se encuentra y devolverá el resto o residuo de la división:

```
21 % 5 = 1;
47 % 6 = 5;
56 % 7 = 0;
```

Precedencia de Operadores y Asociatividad

La *precedencia de operadores* es básicamente el orden en que se van a llamar las funciones de los operadores. O sea que, si tenemos más de un operador, el intérprete va a llamar al operador de mayor precedencia primero y después va a seguir con los demás.

La *Asociatividad de operadores* es el orden en el que se ejecutan los operadores cuando tienen la misma precedencia, es decir, de izquierda a derecha o de derecha a izquierda.

Podemos ver la documentación completa sobre Precedencia y Asociatividad de los operadores de JavaScript acá

```
1 // Precedencia
2 console.log(3 + 4 * 5) // 23
3 console.log(3 * 5 + 4 ** 2) // 31
4
5 // Asociatividad
6 var a = 1, b = 2, c = 3;
7 a = b = c;
8
9 console.log(a,b,c); // 3
```

Operadores de comparación

Se usan para comparar dos expresiones y devuelven un valor booleano, true o false, que representa la relación entre estas dos expresiones.

- == (igual a...)
- != (diferente a...)
- === (igual a... Compara también el tipo de dato)
- !== (diferente a... Compara también el tipo de dato)
- >= (mayor o igual que...)
- <= (menor o igual que...)
- > (mayor (estricto) que...)
- < (menor (estricto) que...)

```
3 === 3 // true
3 == '3' // true
3 = 3 // Uncaught SyntaxError: Invalid left-hand side in assignment

3 === '3' // false
3 !== '3' // true

4 != 3 // true
3 >= '3' // true
3 <= 4 // true
3 > '3' // false
4 < '3' // false
```

Operadores lógicos

También podemos combinar dos expresiones de igualdad y preguntar si alguna de las dos es verdadera, si ambas son verdaderas o si ninguna de ellas es verdadera. Para hacer esto, utilizaremos operadores lógicos.

NOT (!)

AND (&&)

OR (||)

Operador AND (&&)

Esto evaluará ambas expresiones y devolverá true si y solamente si AMBAS expresiones son true. Si una de ellas (o ambas) son false, este operador devolverá false:

```
1 true  && true;  // true
2 true  && false; // false
3 false && true;  // false
4 false && false; // false
5
6 // Ejemplos
7 (100 > 10) && (10 === 10) // true
8 (10 === 9) && (10 > 9) // false
```

Operador OR (||)

Devolverá true si al menos una de las expresiones (o ambas) son true. Devolverá false si AMBAS expresiones son false:

```
true  || true;  // true
true  || false; // true
false || true;  // true
false || false; // false

// Ejemplos
(100 > 10) || (10 === 10) // true
(100 < 10) || (10 === 10) // true
(100 > 10) || (10 !== 10) // true
(10 === 9) || (10 < 9) // false
```

Operador NOT (!)

El operador NOT devolverá el valor booleano opuesto de lo que se le pasa:

```
1 !true; // false
2 !false;  // true
3
4
5 // Ejemplos
6 !(1 === 1) // false
7 !(10 <= 9) // true
```

Métodos

Algunos tipos de datos tienen métodos incorporados, los cuales realizan una acción en concreto SOLO para este tipo de dato. Uno de los más comunes es el método *.length* definido para el tipo de dato "string" y lo que hace es devolver la cantidad de caracteres que tiene una cadena en particular:

```
1 var nombreGato = 'felix';
2 console.log(nombreGato.length); // 5
```

Veremos muchos otros métodos integrados en otros tipos de datos a lo largo de este curso.