Desarrollo Web Frontend

Edición 2021

Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología – Sede Oro Verde

Conceptos básicos de JavaScript

Sintaxis del lenguaje

Tipos de datos

Expresiones y operadores

Estructuras de control

Conceptos básicos de JavaScript

- Es un lenguaje de programación o de secuencias de comandos.
- Creado por Netscape en 1995 con el propósito de mejorar la navegación del usuario directamente sobre el navegador.
- Permite implementar funciones complejas en páginas web.
- Conforma una de las tres capas de las tecnologías web estándar (HTML, CSS, Javascript).
- Permite crear contenido de actualización dinámica, animaciones, eventos sobre botones o formularios, efectos de estilo dinámicos, etc.

Conceptos básicos de JavaScript

Características del lenguaje

- Lenguaje del lado del cliente.
- Orientado a objetos.
- Tipado débil o no tipado.
- Muy utilizado por desarrolladores.
- Multiplataforma.

Conceptos básicos de JavaScript

Requisitos para desarrollar con JavaScript

- Entorno de desarrollo o editor de textos (eclipse, notepad plus, etc).

- Navegador web (Firefox + plugins).

~~Conceptos básicos de JavaScript~~

Sintaxis del lenguaje

Tipos de datos

Expresiones y operadores

Estructuras de control

Sintaxis del lenguaje

Podemos añadir código JavaScript en un documento HTML de la siguiente manera:

Método 1:

```
Index.html
        .....
        .....

            <script>
                //Agregamos código JS
            </script>
        </body>
```

Método 2:

```
index.html
        .....
        .....

            <script src="miScript.js"></script>
        </body>
```

Sintaxis del lenguaje

Variables en Javascript

- El nombre de una variable es **sensible a mayúsculas y minúsculas**.
- Deben declararse con la palabra reservada **let** seguida del nombre de la variable. (Se puede utilizar **var** pero es menos recomendado).
- El nombre puede contener letras, números y algunos caracteres especiales como _ .
- Permite hacer uso de una variable sin que haya sido declarada. Crea automáticamente una variable y permite su uso.
- Las variables son dinámicas, es decir, cambian su tipo de acuerdo al tipo de dato que contienen.

Sintaxis del lenguaje

Separación de instrucciones

```
<script>
//todas las líneas deben terminar con punto y coma (;) para indicar el fin de la declaración

    let miVariable;
    miVariable = '123';

    let curso = 'JavaScript';

    let suma = 1+2;

</script>
```

Sintaxis del lenguaje

Comentarios

- **Método 1**

```
<script>
      // Comentario de una línea
</script>
```

- **Método 2**

```
<script>
      /* Comentario de
         múltiples líneas.

         Otro comentario */
</script>
```

~~Conceptos básicos de JavaScript~~

~~Sintaxis del lenguaje~~

Tipos de datos

Expresiones y operadores

Estructuras de control

Tipos de datos

En javascript disponemos de los siguientes tipos de datos:

- **Undefined** → variable sin inicializar
- **Boolean** → valor booleano (verdadero o falso)
- **Number** → valor numérico (enteros, decimales, etc).
- **BigInt** → valores numéricos realmente grandes que no pueden ser representados por el tipo **number.**
- **String** → valor de texto (caracteres, cadenas de texto).
- **Object** → Es una asociación **{ clave : valor }** que permite construir estructuras de datos más complejas.
- **Null** → valor nulo o vacío. (Ausencia de valor).

Tipos de datos

Podemos determinar el tipo de dato primitivo de un operando utilizando el operador **typeof**

Ejemplo:

```
// a; //undefined
b = 1; //number
c = true; //boolean
d = 'texto'; //string
e = {clave:valor}; //Object
```

typeof a → 'undefined'

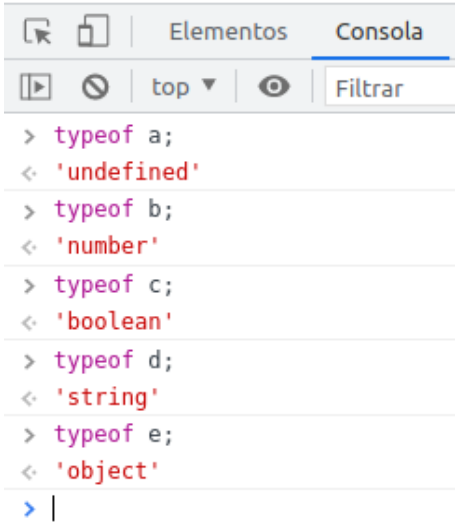
typeof b → 'number'

typeof c → 'boolean'

typeof c → 'string'

typeof d → 'object'

```
<script>

    //let a;
    let b = 1;
    let c = true;
    let d = 'text';
    let e = {nombre: 'Juan'};

</script>
```



Tipos de datos

Tipos Objeto	Tipos predefinidos de Javascript	Date (Fechas) RegExp (Expresiones regulares) Error (datos de error)
	Tipos definidos por el usuario	Funciones simples Clases
	Arrays	Elemento tipo vector o matriz. Se considera un objeto pero carece de metodos.
	Objetos especiales	Objeto global
		Objeto prototipo
		Otros

Tipos de datos

Consideraciones sobre variables de tipo String

Javascript permite definir texto utilizando comillas dobles(“ “), simples (‘’) o backticks (``).

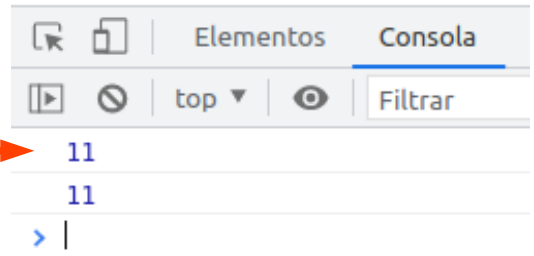
En este ejemplo las definiciones son válidas:

```
<script>

  let cadenaHola = 'Hola';
  let cadenaMundo = "Mundo";
  let cadena = `${cadenaHola} ${cadenaMundo}`;

</script>
```

→ **Hola Mundo**

Podemos obtener la longitud de una cadena utilizando **.lenght** después de la definición de la cadena. Por ej:

```
<script>

    console.log( "Hola Mundo!".length );

    let cadena = "Hola Mundo!";
    console.log( cadena.length );

</script>
```



Tipos de datos

Consideraciones sobre variables de tipo String

HTML también permite usar comillas simples o dobles. Y veremos que esto puede generar el siguiente problema:

```
<button onclick="alert("Hola Javascript");">Saludar</button>
```

alert() *es una función de JavaScript que permite lanzar un mensaje en la ventana del usuario. En este caso asociamos esa función a un evento* ***onclick*** *del botón.*

La forma correcta seria utilizar comillas simples (‘’) para la definición de la cadena en JavaScript.

```
<button onclick="alert('Hola Javascript');">Saludar</button>
```

Tipos de datos

Consideraciones sobre variables de tipo String

Las secuencias de escape se utilizan para representar símbolos que no pueden ser incluidos de forma normal dentro de un texto. A continuación veremos las secuencias de escape mas utilizadas.

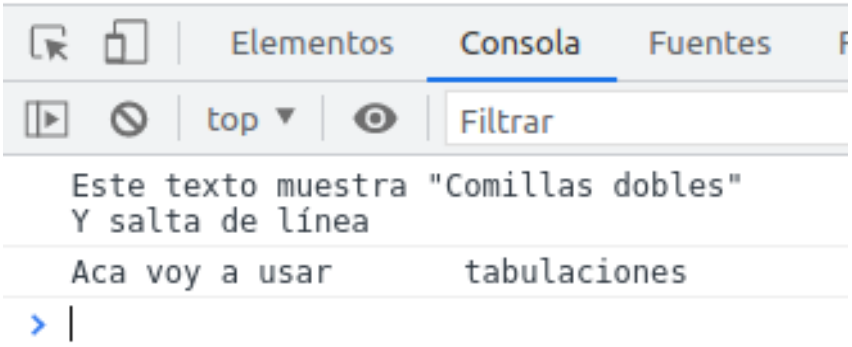
Secuencia de escape	Resultado
\’	Comilla simple
\”	Comilla doble
\\	Barra invertida \
\n	Salto de línea
\t	Tabulación

Tipos de datos

Consideraciones sobre variables de tipo String

Veamos un ejemplo:

```
<script>

  let texto;
  let otroTexto;

  texto = "Este texto muestra \"Comillas dobles\" \nY salta de línea";
  console.log(texto);

  otroTexto = "Aca voy a usar \t \t tabulaciones";
  console.log(otroTexto);

</script>
```



~~Conceptos básicos de JavaScript~~

~~Sintaxis del lenguaje~~

~~Tipos de datos~~

Expresiones y operadores

Estructuras de control

Operadores aritméticos

Operador	Símbolo	Ejemplo
Suma / Concatenación	**+**	5 + 4; “Hola” + “Mundo”; → “Hola Mundo”
Resta Multiplicación División	**-** ***** **/**	10 – 3; 9 * 8; 15 / 5;
Resto de la división	**%**	let resto = 10%3; → 1

Operadores lógicos

Operador	Símbolo	Ejemplo
Asignación	**=**	let nombre = ‘Juan’;
Igualdad	**==**	let a = 2; a == “2” → true
Igualdad estricta	**===**	let a = 2; a === “2” → false
Menor, menor igual, mayor, mayor igual	**<, <=, >, >=**	a > b; b <= a;
Distinto	**!=**	let a = 2; a != “2” → false
Estrictamente Distinto	**!==**	Let a = 2; a !== “2” → true
And (y)	**&&**	a && b
Or (o)	**||**	a || b
Negación	**!**	let a = true; !a → false

Operadores extra

Operador	Símbolo	Ejemplo
Suma lo indicado	**+=**	a+=b → a = a + b;
Resta lo indicado	**-=**	a-=b → a = a – b;
Multiplica lo indicado	***=**	a*=b → a = a * b;
Calcula el módulo por lo indicado	**%=**	a%=b → a = a % b;
Incremento unitario	**++**	let a = 2; a++; a → 3
Decremento unitario	**--**	Let a = 2; a--; a → 1;

~~Conceptos básicos de JavaScript~~

~~Sintaxis del lenguaje~~

~~Tipos de datos~~

~~Expresiones y operadores~~

Estructuras de control

Estructuras de control

Estructuras **Selectivas**

if – else
switch

Estructuras **Iterativas**

for
foreach
while

Estructuras de control selectivas

Estructura Selectiva **if – else**

```
if(condicion){
        sentencia verdadera;
} else {
        sentencia falsa;
}
```

Estructura Selectiva **if – else** anidada

```
if(condicion1){
        sentencia verdadera;
} else if(condicion2){
        sentencia verdadera (condición 2)
} else {
        sentencia falsa;
}
```

Ejemplos:

```
<script>

  if(4 < 5){
    console.log("4 es menor que 5");
  } else {
    console.log("error!");
  }

</script>
```

```
<script>

  if(5 < 4){
    console.log("error!");
  } else if(4 < 3){
    console.log("error!");
  } else {
    console.log("3 es el valor mas chico");
  }

</script>
```

Estructuras de control selectivas

Estructura Selectiva **switch**

```
switch(condicion){

    case condicion_1:
            sentencia1;
            sentencia2;
            break;

    case condicion_2:
    case condicion_3:
            sentencia1;
            break;

    default:
            sentencia default;
}
```

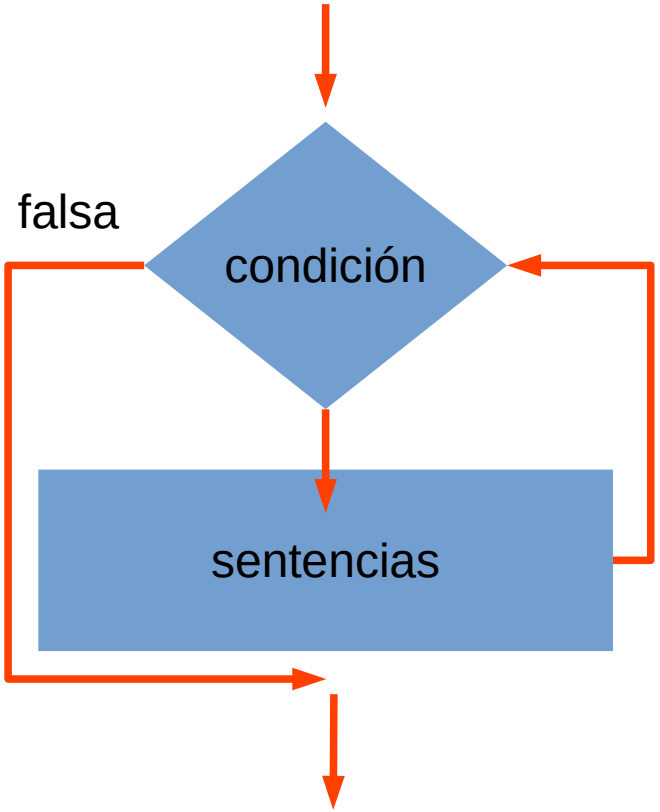
Ejemplo:

```
<script>

  let a = 3;
  switch (a){
    case (1):
        console.log("El valor es 1");
        break;
    case (2):
        console.log("El valor es 2");
        break;
    case (3):
        console.log("El valor es 3");
        break;
    default :
        console.log("No es ninguno de los valores.");
  }

</script>
```

Estructuras de control iterativas

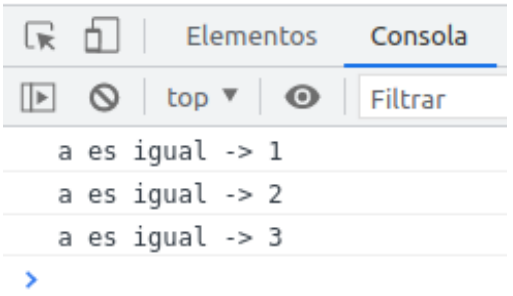
Estructura repetitiva **while**

```
while(condicion){
        sentencia 1;
        sentencia 2;
        sentencia n;
}
```



Ejemplo:

```
<script>

  let a = 1;
  while (a <= 3){
    console.log("a es igual -> " + a);
    a++;
  }

</script>
```



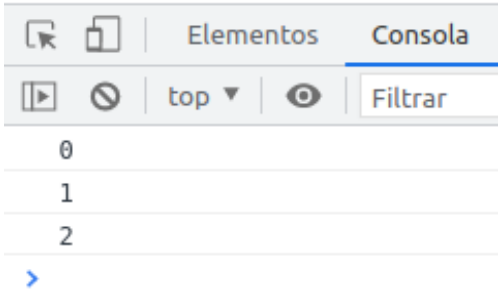
Estructuras de control iterativas

Estructura repetitiva **for**

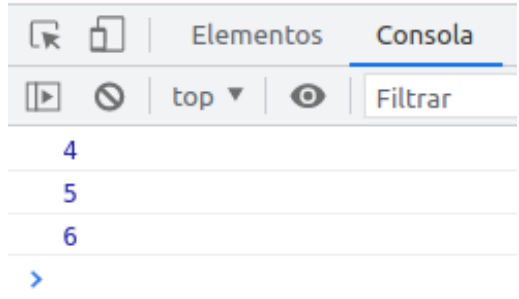
```
for(indice in objeto){
      sentencias;
}
```

Ejemplo:

```
<script>

  let ar = [4,5,6];

  for(let i in ar){
    console.log(i);
  }

</script>
```




```
<script>

  let ar = {
    a:4,
    b:5,
    c:6
  };

  for(let i in ar){
    console.log(ar[i]);
  }

</script>
```



¿Consultas?