Introducción a JAVASCRIPT

LMSGI - CURSO 23-24

IES SAN VICENTE FERRER - ALGEMESÍ

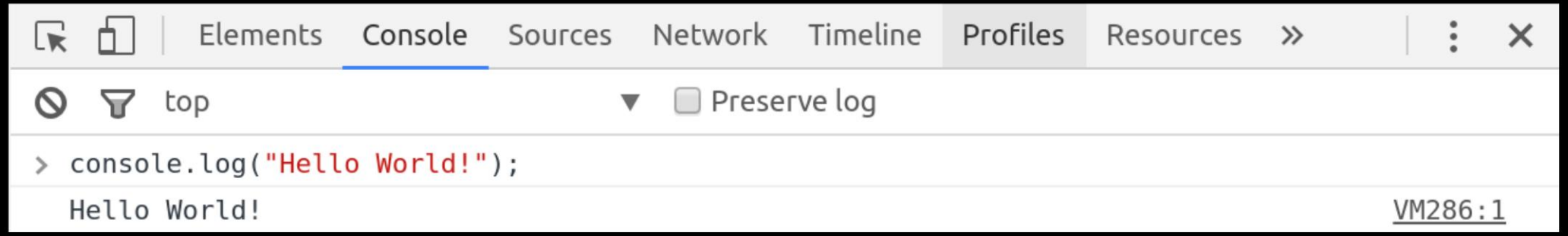
INTRODUCCIÓN

- JavaScript es un lenguaje interpretado, ejecutado por un intérprete normalmente integrado en un navegador web

- Podemos ejecutar código javascript directamente en consola del navegador

CONSUELA DEL NAVEGADOR

- Abre tu navegador Firefox y pulsa F12 para mostrar las herramientas de desarrollador que vienen integradas. Ve a la pestaña de Consola, y desde aquí puedes ejecutar instrucciones en JavaScript y visualizar el resultado de forma inmediata.
- Esta opción es una buena opción para testar pequeñas partes de código



EDITOR DE ESCRITORIO RECOMENDADO

- VISUAL STUDIO CODE

EDITORES WEB

- Fiddle (https://jsfiddle.net/) •

Plunker (https://plnkr.co/)

Integrando JavaScript con HTML

- Para integrar el código JavaScript en nuestro HTML, necesitamos utilizar la etiqueta <script>.
- El sitio recomendado para poner la etiqueta es justo antes de cerrar la etiqueta </html>, para que algunos navegadores puedan cargar y construir el DOM antes de procesar el código JavaScript, de una de otra forma, el navegador bloqueará el renderizado de la página hasta que se haya procesado todo el código JS.

Ubicación del código recomendada

- Dentro de la etiqueta <script> (no recomendado)

```
<!DOCTYPE>
<html>
    <head>
        <title>Ejemplo JS</title>
    </head>
    <body>
        <p>Hola Mundo!</p>
        <script>
            console.log("Hola Mundo!");
        </script>
    </body>
</html>
```

Ubicación del código recomendada

- En un archivo separado

Archivo: ejemplo1.html

```
<!DOCTYPE>
<html>
    <head>
        <title>Ejemplo JS</title>
    </head>
    <body>
        <p>Hola Mundo!</p>
        <script src="ejemplo1.js"></script>
    </body>
</html>
```

Archivo: ejemplo1.js

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

Console.log y console.error

- console.log() se utiliza para escribir por la consola del navegador lo que nosotros le pasamos.
- Puedes utilizar el método console.error() para mostrar los errores.

<noscript>

- La etiqueta <noscript> se utiliza para poner código HTML que será renderizado sólo cuando el navegador no soporte JavaScript o cuando haya sido desactivado.
- Esta etiqueta es muy útil para decirle al usuario que la web necesita tener JavaScript activado para funcionar correctamente, por ejemplo.

<noscript>

- Ejemplo:

```
<!DOCTYPE>
<html>
    <head>
        <title>Ejemplo JS</title>
    </head>
    <body>
        <p>Hola Mundo!</p>
        <noscript>
            <h1>JavaScript no está activado. Por favor, actívelo o la aplicación
web no funcionará correctamente.</h1>
        </script>
    </body>
</html>
```

Variables

- Puedes declarar una variable usando la palabra reservada let (también puedes
- usar var pero no se recomienda desde la versión ES2015). El nombre de la variable deberá ser con el formato CamelCase, es decir, la primera letra en minúsculas, también puede empezar por subrayar (_nombre) o por dólar ($nombre).
- En JS las variables no tienen un tipo de dato explícito. El tipo puede cambiar internamente dependiendo de cuál sea el valor que se le asigne. Esto significa que puedes asignar un string, y posteriormente un número.

Variables

```
let v1 = "Hola Mundo!";
console.log(typeof v1); // Imprime -> string

v1 = 123;
console.log(typeof v1); // Imprime -> number
```

Variables

- ¿Qué pasa si declaramos una variable pero no le asignamos un valor? Hasta que no se le asigne un valor, tendrá un tipo especial conocido como undefined.
- Nota: Este valor es distinto de null (el cual se considera un valor).

```
let v1;
console.log(typeof v1); // Imprime -> undefined
if (v1 === undefined) { // (!v1) or (typeof v1 === "undefined") también funciona
  console.log("Has olvidado darle valor a v1");
}
```

Constantas

- Cuando a lo largo de una función o bloque, una variable no va a cambiar de valor, o cuando queremos definir un valor global inmutable (por ejemplo el número pi), se recomienda declararla como constante con la palabra reservada const
- En el caso de las constantes globales se recomienda utilizar mayúsculas.

```
'use strict';
const MY_CONST=10;
MY_CONST=200; → Uncaught TypeError: Assignment to constant variable.
```

Funciones

- En JavaScript, declaramos funciones usando la palabra reservada function antes del nombre de la función. Los argumentos que le pasaremos a la función van dentro del paréntesis después del nombre de la función (recuerda que no hay tipos de variable en

JS). • Una vez definida la función, entre claves declaramos el cuerpo de ésta. El nombre de las funciones (como el de las variables) debe escribirse en formato CamelCase con la primera letra en minúsculas.

```
function sayHello(name) {
    console.log("Hello " + name);
}

sayHello("Tom"); // Imprime "Hello Tom"
```

Funciones: Retorno de valores

- Podemos usar la palabra reservada return para devolver un valor en una función. Si intentamos obtener un valor de una función que no devuelve nada, obtendremos undefined.

```
function totalPrice(priceUnit, units) {
   return priceUnit * units;
}

let total = totalPrice(5.95, 6);
console.log(total);  // Imprime 35.7
```

Estructuras condicionales: IF

- La estructura if se comporta como en la mayoría de los lenguajes de programación.
- Lo que hace es evaluar una condición lógica, devolviendo un booleano como resultado y si es cierta, ejecuta el código que se encuentra dentro del blog if
- De forma optativa podemos añadir el blog else if, y un blog else.

```
let price = 65;

if(price < 50) {
    console.log("Esto es barato!");
} else if (price < 100) {
    console.log("Esto no es barato...");
} else {
    console.log("Esto es caro!");
}
```

switch

```
let userType = 1;

switch(userType) {
    case 1:
    case 2: // Tipos 1 y 2 entran aquí
        console.log("Puedes acceder a esta zona");
        break;
    case 3:
        console.log("No tienes permisos para acceder aquí");
        break;
    default: // Ninguno de los anteriores
        console.error("Tipo de usuario erróneo!");
}
```

BÚCULO WHILE

```
let value = 1;

while (value <= 5) { // Imprime 1 2 3 4 5
    console.log(value++);
}
```

BUCLE DO WHILE

```
let value = 1;

do { // Imprime 1 2 3 4 5
    console.log(value++);
} while (value <= 5);
```

BÚCULO FOR

```
let limit = 5;

for (let i = 1; i <= limit; i++) { // Imprime 1 2 3 4 5
    console.log(i);
}
```

Podemos utilizar break y continue

```
let limit = 5;

for (let i = 1, j = limit; i <= limit && j > 0; i++, j--) {
    console.log(i + " - " + j);
}
/* Imprime
1 - 5
2 - 4
3 - 3
4 - 2
5 - 1
*/
```