GENERALITAT VALENCIANA
Conselleria d'Educació, Investigació, Cultura i Esport

ies SERPIS | departament informàtica

Unidad 1 - Introducción

DIN - 2º DAM

Beatriz Fuster Ochando

Unidad 1 - Introducción

1. Introducción a JavaScript
    - 1.1. Características principales
    - 1.2. Lenguajes compilados e interpretados
    - 1.3. Ventajas e inconvenientes

2. Primeros pasos Javascript
    - 2.1. Integración de Javascript con código HTML
    - 2.2. Ficheros externos
    - 2.3. Ejecutar Javascript directo
    - 2.4. Herramientas y entornos de desarrollo

1.1. Características principales

❖ **JavaScript (no deriva de Java):**

- Lenguaje de guiones (script, en inglés)
- Implementado originalmente por NCC (Netscape Communications Corporation) → derivado en el estándar **ECMAScript**
- Es conocido por su uso en páginas web, pero también permite realizar otras tareas de programación y administración no enfocadas al entorno web.

1.2. Lenguajes compilados vs interpretados

- **Lenguajes interpretados:** se ejecutan mediante un intérprete (SpiderMonkey, Rhino o Chakra), que procesa las órdenes del programa, una a una. Ej: JavaScript, Python, etc.
  (-) Unas 10 veces más lentos que los lenguajes compilados
  (+) Más sencillos de depurar y más fácilmente portables.

- **Lenguajes compilados:** requieren un compilador, que convierte las instrucciones programadas en código máquina (código binario entendible por el procesador). Ej: C/C++, Java, etc.
  (+) Ejecución más eficiente
  (-) Requieren mayor espacio en memoria

>> Javascript es un lenguaje interpretado.

1.3. Ventajas y desventajas

- **Ventajas:**
  - **Curva de aprendizaje accesible**: Sintaxis relativamente sencilla.
  - **Amplio ecosistema** de frameworks y librerías
  - Un solo **código base** puede desplegarse en múltiples plataformas (write once, run anywhere).

- **Desventajas:**
  - **Seguridad:** Muy expuesto a vulnerabilidades si no se controla bien.
  - **Consumo de memoria**: Frameworks como Electron generan aplicaciones pesadas porque básicamente incluyen un navegador dentro de cada app.

2.1. Integración de Javascript con código HTML

- El código JavaScript siempre irá entre las etiquetas <script> i </script>
- Estas etiquetas pueden situarse:
- En la sección <head> </head>
- En la sección <body> </body> del document HTML

> *El código JavaScript al final de la etiqueta <body> </body> mejora la velocidad de carga de la página.*

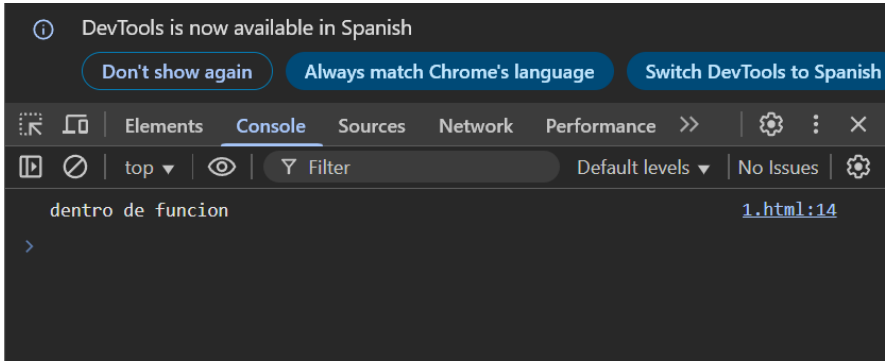
- Los scripts de JavaScript también pueden almacenarse en un archivo externo (extensión.js) y hacer una llamada desde el código principal → aprovechamiento de código

2.1. Integración de Javascript con código HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <h1>Mi primer ejemplo</h1>
    <p id="parrafo">Texto original del párrafo</p>
    <button type="button" onclick="Saludamos()">Clícame</button>
    <script>
      function Saludamos() {
        document.getElementById('parrafo').innerHTML = 'Hola,
bienvenidos';
      }
    </script>
  </body>
</html>
```

2.1. Integración de Javascript con código HTML

Muy importante utilizar la consola para ver errores y comprobar por qué falla usando console.log



2.2. Ficheros externos

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <title>Mi primer ejemplo</title>
  </head>
  <body>
    <h1>A Web Page</h1>
    <p id="demo">texto original</p>
    <button type="button" onclick="myFunction()">Cambia</button>
    <script src="funcionExterna.js"></script>
  </body>
</html>
```

jsexterno.html | funcionExterna.js

```js
function myFunction() {
  document.getElementById('demo').innerHTML = ‘Texto cambiado';
}
```

2.3. Ejecutar javascript directo

Si queremos ejecutar código javascript directamente podemos usar node en la terminal de visual Studio code:

>> node nombre.js

```
var z;
console.log(typeof(z)); //undefined
z=""
console.log(typeof(z)); //string
z=null;
console.log(typeof(z)); //object
```

```
PS C:\Users\bea_v\Documents\SERPIS_24_25\dwec\ejercicios\1. introJS> node 3.datos.js
undefined
string
object
```

2.4. Herramientas y entornos de desarrollo

- Para programar con JavaScript, necesitamos un editor de texto + un entorno de desarrollo (IDE).
- Los navegadores web más actuales ya incorporan un IDE (Integrated Development Environment)
- Del editor esperamos:
  - Selección del lenguaje JavaScript
  - Coloración sintáctica
  - Autocompletado
  - Herramientas avanzadas de búsqueda y reemplazo
  - Instalación de plugins con funcionalidades extra
  - Ejemplos: Sublime, Notepad++(Windows), Notepadqq(Linux), Brackets
- Otras opciones:
  - Un entorno de JavaScript online (codepen.io) → https://codepen.io/
  - Instalar un IDE, como Eclipse, NetBeans ó **Visual Studio Code.**

4. Herramientas y entornos de desarrollo

- Utilizaremos Visual Studio Code:
  - En lliurex viene instalado por defecto.
  - En windows hay que instalarlo: https://code.visualstudio.com/download