IES CLAUDIO MOYANO 1919 2019
Junta de Castilla y León
Consejería de Educación
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE
Programa cofinanciado por el Ministerio de Educación, Cultura y Deporte y por el Fondo Social Europeo

DWEC – Javascript Web Cliente.

JavaScript 00 – Lenguaje JavaScript

Introducción

Para el desarrollo de las páginas web lo principal es un fichero HTML que contiene la información a mostrar en el navegador. Posteriormente surgió la posibilidad de "decorar" esa información para mejorar su apariencia, lo que dio lugar al CSS. Y también se pensó en dar dinamismo a las páginas, y apareció el lenguaje JavaScript. Que también lo escribiremos como Javascript o JS.

En un primer momento las 3 funcionalidades estaban mezcladas en el fichero HTML, pero eso complicaba bastante el poder leer esa página a la hora de mantenerla por lo que se pensó en separar los 3 elementos básicos:

HTML: se encarga de estructurar la página y proporciona su información, pero es una información estática

CSS: es lo que da forma a dicha información, permite mejorar su apariencia, permite que se adapte a distintos dispositivos, …

JavaScript: es el que da vida a un sitio web y le permite reaccionar a las acciones del usuario

Por tanto, nuestras aplicaciones tendrán estos 3 elementos y lo recomendable es que estén separados en distintos ficheros:

El HTML lo tendremos habitualmente en un fichero index.html, normalmente en una carpeta llamada public, html (o similar).

El CSS lo tendremos en uno o más ficheros con extensión .css dentro de una carpeta llamada styles, estilos, css (o similar).

EL JS estará en ficheros con extensión .js en un directorio llamado js, scripts, funciones (o similar).

Nota: Existen variedades y complementos del lenguaje JavaScript, como son TypeScript, CoffeScript, Vue, Angular, etc.

Vanilla JavaScript es como se conoce al lenguaje JavaScript cuando se utiliza sin ninguna librería o *framework*. La traducción más castellana sería "JavaScript a pelo".

Las características principales de Javascript son:

- Es un lenguaje interpretado, no compilado.
- Se ejecuta en el lado cliente (en un navegador web), aunque hay implementaciones como NodeJS para el lado servidor.
- Es un lenguaje orientado a objetos (podemos crear e instanciar objetos y usar objetos predefinidos del lenguaje) pero basado en prototipos (por debajo, un objeto es un prototipo y nosotros podemos crear objetos sin instanciarlos, haciendo copias del prototipo).
- Se trata de un lenguaje débilmente tipado, con tipificación dinámica (no se indica el tipo de datos de una variable al declararla e incluso puede cambiarse).

Usaremos JavaScript para:

- Cambiar el contenido de la página
- Cambiar los atributos de un elemento
- Cambiar la apariencia de algo
- Validar datos de formularios
- …

Sin embargo, por razones de seguridad, JavaScript no nos permite hacer cosas como:

- Acceder al sistema de ficheros del cliente
- Capturar datos de un servidor (puede pedirlo y el servidor se los servirá, o no)
- Modificar las preferencias del navegador
- Enviar e-mails de forma invisible o crear ventanas sin que el usuario lo vea
- …

Un poco de historia

JavaScript es una implementación del lenguaje **ECMAScript** (el estándar que define sus características).

El lenguaje surgió en 1997 y todos los navegadores a partir de 2012 soportan al menos la versión **ES5.1** completamente.

En 2015 se lanzó la 6ª versión, inicialmente llamada **ES6** y posteriormente renombrada como **ES2015**, que introdujo importantes mejoras en el lenguaje y que es la versión que usaremos nosotros.

Desde entonces van saliendo nuevas versiones cada año que introducen cambios pequeños. La última es la versión 13, llamada ES2022, que ha sido aprobada en verano de 2022.

Las principales mejoras que introdujo ES2015 son: clases de objetos, let, for..of, Map, Set, Arrow functions, Promesas, spread, destructuring, …

Soporte en los navegadores

Los navegadores no se adaptan inmediatamente a las nuevas versiones de JavaScript, por lo que puede ser un problema usar una versión muy moderna e JS ya que puede haber partes de los programas que no funcionen en los navegadores de muchos usuarios.

En la página de *Kangax* podemos ver la compatibilidad de los diferentes navegadores con las distintas versiones de JavaScript.

También podemos usar *CanIUse* para buscar la compatibilidad de un elemento concreto de JavaScript, así como de HTML5 o CSS3.

Si queremos asegurar la máxima compatibilidad debemos usar la versión ES5 (pero nos perdemos muchas mejoras del lenguaje).

Lo mejor sería usar la ES6 (o posterior) y después *transpilar* nuestro código a la versión ES5. De esto se ocupan los *transpiladores* (**Babel** es el más conocido), por lo que no suponen un esfuerzo extra para el programador.

Si queremos asegurar la máxima compatibilidad debemos usar la versión ES5 (pero nos perdemos muchas mejoras del lenguaje) o mejor, usar la ES6 (o posterior) y después transpilar nuestro código a la version ES5. De esto se ocupan los transpiladores (Babel es el más conocido) por lo que no suponen un esfuerzo extra para el programador.

Herramientas

La consola del navegador

Es la herramienta que más nos va a ayudar a la hora de depurar nuestro código. Abrimos las herramientas para el desarrollador (en Chrome y Firefox pulsando la tecla F12) y vamos a la pestaña Consola:

Allí vemos mensajes del navegador como errores y advertencias que genera el código y, todos los mensajes que pongamos en el código para ayudarnos a depurarlo (usando los comandos **console.log** y **console.error**).

Además, en ella podemos escribir instrucciones JavaScript que se ejecutarán mostrando su resultado. También la usaremos para mostrar el valor de nuestras variables y para probar código que, una vez que funcione correctamente, lo copiaremos a nuestro programa.

Siempre depuraremos los programas desde la consola (pondremos puntos de interrupción, veremos el valor de las variables, …).

Editores

Podemos usar el que más nos guste, desde editores tan simples como NotePad++ hasta complejos IDEs. La mayoría soportan las últimas versiones de la sintaxis de JavaScript (Netbeans, Eclipse, Visual Studio, Sublime Tex, Atom, Kate, Notepad++, …).

Por el momento utilizaremos el editor **Visual Studio Code** por su sencillez y por los *plugins* que incorpora para hacer más cómodo el trabajo del desarrollador. En *Visual Studio Code* instalaremos algunos *plugins* como:

SonarLint: es más que un *linter* y nos informa de todo tipo de errores, pero también del código que no cumple las recomendaciones (incluye gran número de reglas). Marca el código mientras lo escribimos y además podemos ver todas las advertencias en el panel de Problemas (Ctrl+Shift+M)

Otros plugins: según el documento a nuestra disposición “Instalación de Extensiones de Visual Studio Code”

Editores on-line

Son muy útiles porque permiten ver el código y el resultado a la vez. Normalmente tienen varias pestañas o secciones de la página donde poner el código HTML, CSS yJavascript y ver su resultado.

Algunos de los más conocidos son Codesandbox, Fiddle, Plunker, CodePen, …aunque hay muchos más.

Como ejemplo: https://jsfiddle.net/tqobL253/

Que se puede ver cómo queda incrustado en una página web si incrustamos el siguiente código:

```
<script async src="//jsfiddle.net/tqobL253/embed/js,html,css,result/dark/"></script>
```

npm

npm es el gestor de paquetes del framework JavaScript **Node.js** y suele utilizarse en programación *frontend* como gestor de dependencias de la aplicación.

Esto significa que será la herramienta que se encargará de descargar y poner a disposición de nuestra aplicación todas las librerías JavaScript que vayamos a utilizar.

Para instalar *npm* tenemos que instalar *NodeJS*.

Git

Usaremos repositorios git realizar el control de versiones de nuestras aplicaciones.

GitHub

Usaremos una cuenta en GitHub.com, generalmente asociada a otra cuenta de Gmail.com, para llevar el control de las versiones y tener copia de nuestro código en la nube.

Incluir JavaScript en una página web

El código JavaScript va entre etiquetas *<script>*. Puede ponerse en el *<head>* o en el *<body>*. Funciona como cualquier otra etiqueta.

El navegador la interpreta cuando llega a ella. Va leyendo la etiqueta y ejecutando el fichero línea a línea.

Opción 1: en HEAD

```
<!DOCTYPE html>
<head>
    <meta charset="UTF-8">
    <script>
        function saludar() {
            alert("Hola majos/as");
        }
        saludar();
    </script>
</head>
<body>
    <h1>Bienvenidos a DWEC</h1>
</body>
</html>
```

Opción 2: en BODY (delante de otros elementos):

```
<!DOCTYPE html>
<head>
```

```
    <meta charset="UTF-8">
</head>
<body>
    <script>
        function saludar() {
            alert("Hola majos/as");
        }
        saludar();
    </script>
    <h1>Bienvenidos a DWEC</h1>
</body>
</html>
```

Opción 3: en fichero externo de extensión js (en HEAD)

```
<!DOCTYPE html>
<head>
    <meta charset="UTF-8">
    <script src="js/funciones.js"></script>
</head>
<body>
    <h1>Bienvenidos a DWEC</h1>

    <h1>hasta otra</h1>
</body>
</html>
```

Opción 4 (**Recomendada**): en fichero externo de extensión js (en BODY)

```
<!DOCTYPE html>
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="css/estilos.css">
</head>
<body>
    <h1>Bienvenidos a DWEC</h1>

    <h1>hasta otra</h1>
    <script src="js/funciones.js"></script>
</body>
</html>
```

Lo mejor, en cuanto a rendimiento, es ponerla al final del *<body>* para que no se detenga el renderizado de la página mientras se descarga y se ejecuta el código.

También podemos ponerlo en el <head> pero usando los atributos **async** y/o **defer** (en Internet encontraréis mucha información sobre esta cuestión, por ejemplo aquí).

Es posible poner el código directamente entre la etiqueta *<script>* y su etiqueta de finalización, pero lo correcto es que esté en un fichero externo (con extensión **.js**) que cargamos mediante el atributo *src* de la etiqueta. Así conseguimos que la página HTML cargue más rápido que si lo ponemos al final del BODY o usamos *async*.

Además, es preferible no mezclar HTML y JS en el mismo fichero, lo que mejora la legibilidad del código y facilita su mantenimiento.

```
<script src="./scripts/main.js"></script>
```

Mostrar información

JavaScript permite mostrar al usuario ventanas modales para pedirle o mostrarle información. Las funciones que lo hacen son:

- `window.alert(`*`mensaje`*`)`: Muestra en una ventana modal *mensaje* con un botón de *Aceptar* para cerra la ventana.
- `window.confirm(`*`mensaje`*`)`: Muestra en una ventana modal *mensaje* con botones de *Aceptar* y *Cancelar*. La función devuelve **true** o **false** en función del botón pulsado por el usuario.
- `window.prompt(mensaje [, valor predeterminado])`: Muestra en una ventana modal *mensaje* y debajo tiene un campo donde el usuario puede escribir, junto con botones de *Aceptar* y *Cancelar*. La función devuelve el valor introducido por el usuario como texto (es decir que si introduce 54 lo que se obtiene es "54") o **false** si el usuario pulsa *Cancelar*.

También se pueden escribir las funciones sin *window.* (es decir `alert('Hola')` en vez de `window.alert('Hola')`) ya que en JavaScript todos los métodos y propiedades de los que no se indica de qué objeto son se ejecutan en el objeto *window*.

Si queremos mostrar una información para depurar nuestro código no utilizaremos *alert(mensaje)* si no `console.log(mensaje)` o `console.error(mensaje)`. Estas funciones muestran la información en la consola del navegador. La diferencia es que *console.error* la muestra como si fuera un error de JavaScript.

Por último, indicar que podremos usar el método *write* del objeto *document* para enviar información (en lenguaje HTML) al documento HTML donde ponemos este código. Document.window(código_html). Ejemplos: document.write("Hola Daw2") o document.write(' Hola Daw2 ')