**TEMA 2: Introducción al lenguaje JavaScript**

**1. ¿Qué es JavaScript?**

Es un lenguaje de programación alto nivel interpretado por el navegador, permitiendo crear páginas webs dinámicas.

Entre sus funciones destacan:

* Validar datos de un formulario: Verificar que los campos están rellenos y en el formato adecuado.
* Detectar el navegador que se está usando: Permitirá optimizar el código según las características del navegador.
* Almacenar información del usuario (Cookies).

**1.1. Características principales**

* Es un lenguaje basado en objetos, aunque con menos restricciones y no de forma completa.
* Es un lenguaje orientado a eventos, debido al tipo de entornos que utiliza.

**1.2. Stack actual de JavaScript**

JavaScript no solo se utiliza en el lado del cliente, también se está usando en servidor, dispositivos multiplataforma o bases de datos. Esto es conocido como Full Stack JS.

Cuando es ejecutado en el servidor, existen motores que interpretan este lenguaje como Spider Monkey y Rhino. El entorno de desarrollo más conocido es Node.js.

**1.3. Qué no es JavaScript**

No es un lenguaje de propósito general, por lo que no tiene control sobre los recursos del equipo. Solo tiene control sobre el HTML en el que está incluido y las ventanas ejecutadas por el navegador. Por motivos de seguridad el sistema de ficheros del cliente está muy restringido.

No es un lenguaje de sustituto de los scripts del servidor, pero su uso combinado con ellos puede permitir la depuración de datos.

No era un lenguaje a orientado a objetos, ya que no utilizaba los conceptos de clase, herencia, polimorfismo u ocultamiento de la información. TypeScript si es orientado a objetos de forma completa.

No es Java. Java es un lenguaje compilado que se ejecuta en una máquina virtual.

**2. El entorno HTML**

Para entender como funciona JavaScript, es necesario conocer HTML.

**2.1 Definición de HTML**

Es un lenguaje de hipertexto que incluye mecanismos para enlazar documentos entre sí.

**2.2 Cómo funciona**

* Se solicitan los recursos a un navegador
* La red redirige la petición hasta el servidor
* El servidor analiza la petición a través de la URL y devuelve la respuesta
* El navegador visualiza la respuesta del servidor.

**2.3 Nuestro primer HTML en local**

Para ejecutar un HTML en local utilizando XAMPP, necesitamos:

* Acceder a la carpeta xampp/htdocs
* Crear una carpeta dentro de la misma.
* Crear un fichero HTML dentro de dicha carpeta.
* Abrir la dirección del fichero en el navegador.

**2.4 Nuestro primer HTML en la nube**

Siguiendo unos pasos similares en c9.io podríamos ejecutar un HTML en la nube.

**2.5 Algo de sintaxis HTML**

Una web escrita en HTML debe tener la siguiente estructura:

* Definición del tipo de fichero con la etiqueta <html>
* Definición del encabezado con la etiqueta <head>
  + Definición del título con la etiqueta <title>
* Definición del cuerpo con la etiqueta <body>

Además, hay que verificar que title y body están dentro de html, y que title está antes que body. También hay que verificar que todas las etiquetas están cerradas.

**3. Comenzando con JavaScript**

Se puede añadir JavaScript en dos lugares:

* En la etiqueta <head>: Cuando se hace referencia a ficheros externos de JavaScript.
* En la etiqueta <body>: Cuando se escribe código para ejecutar en la página. Es recomendable ponerlo al final.

**3.1 Insertar código JavaScript**

Para introducir código JavaScript tenemos que usar la etiqueta <script>

* Si el código está en un fichero externo, se referencia con el atributo src.

<script type="text/javascript" src="js/fichero.js"></script>

* Si es código se va a incluir en el fichero, se introduce dentro de la etiqueta.

**3.2 Sintaxis**

Usa una sintaxis similar a Java o C++, con algunas diferencias:

* JavaScript distingue entre mayúsculas y minúsculas
* JavaScript ignora los espacios y los saltos de línea, aunque ayudan a la legibilidad del código.
* Se utiliza el punto y coma para separar cada instrucción.
* Existen determinadas palabras reservadas que no pueden usarse para variables, funciones o etiquetas.

**3.3 Comentarios**

No son interpretados por el navegador.

Se puede comentar con // y /\* \*/

**3.4 Tipos de datos**

* Números: Solo existe un tipo numérico, representado a través de punto flotante de 64 bits. Es el equivalente al double en Java.
* Cadenas de texto: Pueden representar letras, números, signos o cualquier carácter de Unicode. Debe definirse entre comillas (simples o dobles)
* Booleano: Solo admite true o false.

Secuencias de escape:

* \\ Barra invertida
* \’ Comilla simple
* \” Comilla doble
* \n Salto de línea
* \t Tabulación horizontal
* \v Tabulación vertical
* \f Salto de página
* \r Retorno de carro
* \b Retroceso

**3.5 Variables**

Son las zonas de memoria en las que se almacenan ciertos datos.

* Se declaran mediante la palabra var las globales y let las locales. Se puede declarar más de una variable en una línea.
* La inicialización de una variable puede hacerse:
  + Mediante asignación directa del valor
  + Mediante asignación indirecta a través de cálculos con otras variables
  + Mediante asignación de valores solicitados al usuario.

**3.6 Operadores**

Aritméticos:

* + Suma
* - Resta
* \* Multiplicación
* / División
* % Módulo
* ++ Incremento
* -- Decremento

Lógicos:

* && y
* || ó
* ! Distinto

Asignación:

* = Asigna
* += Suma y asigna
* -= Resta y asigna
* \*= Multiplicación y asigna
* /= División y asigna
* %= Módulo y asigna

Comparación:

* < Menor
* <= Menor o igual
* == Igual
* > Mayor
* >= Mayor o igual
* != Distinto
* === Estrictamente igual
* !== Estrictamente distinto

Condicionales:

* ? Condicional

**3.7 Sentencias de control**

Son aquellas que evalúan condiciones y ejecutan instrucciones según el resultado

* if: Plantea una condición y según su cumplimiento ejecuta la instrucción
* switch: Es un bloque de instrucciones que evalúa el comportamiento en cada distintos casos. La instrucción break es necesaria tras cada caso, y se puede poner una instrucción genérica con default.

**3.8 Sentencias repetitivas**

Ejecutan un mismo bloque de código según el caso:

* while: Lo ejecuta mientras se cumpla una condición.
* do/while: Lo ejecuta hasta que deje de cumplirse una condición.
* for: Lo ejecuta un número determinado de veces.