JavaScript

JavaScript es un lenguaje no compilado que funciona gracias a un interprete de código.
Con JavaScript podemos dar funcionalidad a nuestra página.
Se emplea tanto en entorno servidor (backend) como en entorno cliente (frontend)
Se utiliza para añadir características interactivas a nuestra página web.

framework
Es un conjunto de herramientas basadas en un lenguaje de programación compuesta por bibliotecas de código y patrones de diseño avanzado que sirven como base para el desarrollo de diversas aplicaciones.

Librerías
Sirven de soporte a nuestra página para incluir alguna dependencia externa.

Colocación en HTML
- En un fichero independiente (con extensión js)
  - <script src="script.js"></script> (Colocar en el head)
  - <script src="script.js" async></script> (carga asíncrona más rápida)
    - async: descarga el HTML y ejecuta el JavaScript al mismo tiempo.
    - defer: descarga el HTML y luego ejecuta el javascript
- Entre etiquetas <script></script>
  - Final dela etiqueta </body> (Evita afectar a la velocidad de carga de la página)
- Dentro de una etiqueta (script en linea, no es recomendable)

noscript
Se emplea para avisar al usuario que la página web necesita usar javascript para funcionar correctamente e incluye un mensaje de aviso.

```
<body>
        <noscript>
        <p>Esta web no funciona sin javascript<p>
        </noscript>
</body>
```

Conjunto de reglas
- El interprete de JS ignora espacios en blanco sobrantes.
- Es case sensitive.
- No se definen los tipos de variables.
- No obliga, pero es aconsejable acabar las sentencias con " ; "
- Se recomienda incluir comentarios.

Sintaxis en JS

- Las variables se declaran con let
- Los objetos se encierran entre llaves parejas con el formato propiedad: valor,
- El valor de una propiedad de un objeto se obtiene con objeto.propiedad
- Para definir una función se recomienda usar constantes con const

Declaraciones

Ventana alert:

- Usada para mostrar un mensaje después de realizar alguna acción.
- Tiene la propiedad modal, se abre en primer plano y no permite manipular nada de lo que se encuentre en segundo plano.
  - <button type="button" onclick="alert('Esto es un cuadro de alerta')">Probar alerta</button>

Ventana confirm:

- Función parecida a alerta, muestra un cuadro de dialogo de confirmación.
- Ofrece dos opciones, aceptar y cancelar.
  - Devuelve true o false.

Consola:

- Se ejecuta desde cualquier navegador pulsando f12
- Muestra el código JS para comprobar que funciona.
  - console.log("Texto a escribir");

ELEMENTOS DE CÓDIGO

Comentarios

Se emplean para hacer anotaciones en el código.

- // → Para hacer comentarios en una línea.
- /* Para diversas líneas*/

Variables

Son contenedores para almacenar valores. Se declara la variable con la palabra *let* o *var,* a continuación un símbolo "=" y el valor que le queremos dar.

```
let x = 5;
let y = 3;
let z = x + y;
```

Tipos:

- Globales: Se definen para un entorno global.
- Locales: Definidas para un entorno concreto.

Declaraciones

- **var**: Usada para declaraciones generales y de alcance.
  - Se hereda por referencia, por lo que puede usarse como variable global
- **let**: Usada en variables generales.
  - Solo usada en el bloque que se ha declarado.
- **const**: Uso para datos que no pueden ser sobrescritos.

Definiciones:

- camelCase: Para variables que hacen referencia a varias palabras.
- lowercase: Para variables que referencian a una sola palabra.
- Las constantes se escriben en mayúsculas, separadas por un "_" si son dos o más.

Tipos de operaciones:

- Aritméticas:
  - +, -, *, /, %
- De comparación:
  - <, >, <=, >=, ==, !=
  - Comparación estricta (===) compara valor y tipo del dato.
- Lógicas
  - AND (&&)
  - OR (||)
  - NOT (!)

TIPO DE DATOS

Numbers:

- Positivos, negativos o decimales. Usan operadores aritméticos.

Strings: Cadenas de texto, se pueden concatenar con el operador (+)

- Se encierran entre comillas dobles(" "), o simples(' ').
- Se pueden concatenar con una variable dinámica:
  - Usar como comillas el acento grave (` `) y la variable debe de encontrarse en el formato *${variable}*; No necesita el operador (+)
- Con el carácter " \ " se puede a la hora de escribir un texto:
  - \n → Hacer un sato de línea
  - \t → Hacer un tabulado
  - Escapar los caracteres (' " y \ ).

Booleans: true / false

- Emplean operadores de comparación.

null y undefined:

- undefined: Ausencia de valor.
- null: Tiene un valor que es null.

Arrays: Objeto similar a una lista que posee métodos para efectuar operaciones de recorrido y mutación de esta.

var matriz = [valor, valor, "valor"];

- Tanto la longitud como el tipo de elementos son variables.
- Métodos:
  - .push(*valor*) Añade un valor al final de la matriz.
  - .shift() Remueve el primer valor de la matriz y lo devuelve para procesarlo.
  - .pop() Remueve el último valor de la matriz y lo devuelve para procesarlo.

Condicionales

if, else if, else:

- Si dentro del paréntesis del if la condición es verdadera, se ejecuta el código dentro de las llaves.
- De no ser así, se comprueba que la condición del **else if** es verdadera, se ejecuta el código dentro de las llaves del else if.
- En caso contrario, si no se cumple ninguna condición else if, se ejecuta el código después del **else** y se continua con el programa.

switch:

- Conjunto de condiciones (case) que si coinciden con ese valor del switch, ejecutan esa parte del código.
- Si el valor de switch no coincide con ninguna condición se usa el default.

Bucles

Son estructuras empleadas para repetir sentencias un determinado número de veces.

- **for**: Estructura repetitiva empleada para recorrer arrays.
- **do while**: Repite al menos una vez una sentencia.
- **while**: Puede repetir o no una sentencia

OBJETOS

Son variables capaces de contener a otras variables (denominadas propiedades) y puede contener funciones (denominadas métodos).

```
Var miObjeto = {
    nombre: "Benito",
    edad: 73
};
```

- Mediante la "notación de punto":
  - Se puede acceder al las propiedades del objeto.

```
alert(miObjeto.nombre);
```

  - Se puede modificar los valores de las propiedades del objeto.

```
miObjeto.edad = 69
```

Constructor

Es una función especial para crear objetos.

- Puede o no llevar parámetros
- Empleamos *this*. Para hacer referencia a los atributos de la clase.

```
function miObjeto ( valorf1, valorf2){
    this.valor1 = valorf1;
    this.valor2 = valorf2;
};
```

- Empleando el operador **new** podemos crear objetos

```
const cosaAlgo = new miObjeto(valor1, valor2);
```

Funciones:

- Son “subprogramas” que pueden ser llamados por código externo a la función.
- Poseen sus propias declaraciones.
- Pueden recibir y (siempre) devolver valores.
- Las variables que se definen dentro de la función son variables locales
- Pueden usar parámetros, tanto para recibir valores como para devolverlos.

```
function suma(a, b){
    console.log(a +b);
};
```

Operadores ternarios:
Dependiendo de si la condición es true o false muestran el bloque 1 o el 2

```
console.log(1>2
    ? ”El primero es mayor” // Bloque 1
    : “El segundo es mayor”); // Bloque 2
```

DOM (Modelo de objeto de documentos) - Árbol de nodos
Permite a los programadores acceder y manipular las páginas web como si fueran documentos XML

- Toda página HTML parte de un nodo raíz llamado “Document”
- Cada etiqueta HTML se transforma en un nodo de tipo “Elemento”
- Esta conversión se realiza de forma jerárquica.

Para acceder a los elementos empleamos el método document seguido de:

- **.getElementsByTagName**(“valor”):
  - Obtiene todos los elementos de la página que tengan el nombre igual al parámetro que se pasa a la función y los almacena en una array.
- **.getElementsById**(“identificador”):
  - Accede a un nodo que tenga la id especificada en la función.
- **.getElementsByName**(“valor”):
  - Obtiene todos los elementos con el atributo “name” que coincida con “valor”
- **.getElementsByClassName**(“valor”):
  - Accede a todos los elementos con el atributo class pasado como “valor”

Objeto window
Objeto del cual cuelgan los demás como si fueran propiedades suyas. Crea un objeto de este tipo para cada ventana para lanzar una página web.

Temporizadores:

- **.setTimeOut**(función, milisegundos):
  - Se ejecuta la función en primer lugar y luego el tiempo indicado.
- **.setInterval**(función, milisegundos): Llama a la función durante el tiempo indicado.
  - Se cancela con **.clearInterval**

outerHTML: Retorna el contenido del HTML en formato texto.

Invocar funciones sencillas con href
Mediante un enlace se pueden incluir sentencias de JS :

- Mensaje

Nuevos métodos de selección

- **.querySelectorAll**(): Devuelve todos los elementos indicados en el paréntesis en forma de vector en el mismo orden del la página.
  - Empleando una " ," podemos apuntar a diversos elementos al mismo tiempo.

Eventos
Son acontecimientos que suceden cuando ocurre cierto tipo de situación. Estos eventos están asociados a un elemento dentro de la ventana de navegación. Ejemplos: Seleccionar, hacer click, pasar el ratón por un elemento…

- **.addEventListener**("*evento", función)* : Produce un evento. Eventos:
  - **moseup**: Cuando se suelta el botón del ratón pulsado.
  - **mousemove**: cuando el puntero se encuentra encima del elemento.
  - **contextmenu**: cuando se pulsa el botón derecho.
  - **change**: Se produce cuando un cuadro de texto (<input> ó <textarea>) pierde el foco y su contenido ha variado. También con un elemento <select> cuando varía su valor.
  - **submit / reset**: Cuando se pulsa un botón de este tipo.
  - **resize**: cuando se redimensiona la ventana del navegador.
  - **scroll**: Cuando se utiliza una barra de desplazamiento.
  - **focus**: Cuando un elemento obtiene el foco.
  - **blur**: Cuando un elemento pierde el foco.