UD2 Sintaxis de JavaScript

2º CFGS DAW

Desarrollo Web en Entorno Cliente

2019-20

1 – Introducción

Al haber superado el módulo **Programación** de primer curso se tienen los conocimientos de programación mínimos.

En esta unidad se repasarán esos conocimientos aplicados a la sintaxis de JavaScript

Toda la documentación de **JavaScript** se puede consultar en el siguiente enlace:

https://www.w3schools.com/js/default.asp

1 – Introducción

JavaScript es un lenguaje de script interpretado que se ejecuta en el cliente.

De Script: documento con instrucciones en un lenguaje de programación.

Interpretado: se ejecuta traduciendo línea a línea cada instrucción.

1 – Introducción

Para introducir código en **JavaScript** se utiliza la etiqueta **<script>**.

```
<script>
    alert('Mensaje desde JavaScript');
</script>
```

1 – Introducción

El punto y coma!

Al ser interpretado, Javascript, reconoce cada línea como una instrucción independiente. Cuando hay un salto de línea acaba una instrucción.

Esto significa que no es necesario poner el **punto y coma** al final de cada instrucción, pero como buena costumbre al programar, será obligatorio ponerlo.

No pasa nada, estáis acostumbrados ya que en Java lo poníais. XD

1 – Introducción

```
<script type="text/javascript" language="javascript">
    alert('Mensaje desde JavaScript');
</script>
```

Atributos

type: en HTML5 es opcional.

language: está obsoleto y no debe utilizarse aunque si se utiliza no da error, debería usarse **type** en su lugar.

1 – Introducción

Dos opciones:

```
<script type="text/javascript">
    alert(‘Mensaje desde JavaScript’);
</script>
```

```
<script type="text/javascript">
    alert(‘Mensaje desde JavaScript’);
</script>
```

1 – Introducción

Tercera y **mejor** opción:

Es conveniente separar la lógica de negocio de la aplicación web del contenido (html) y del estilo (css). Por ello se recomienda incluir el código JavaScript en archivos propios con extensión **.js** igual que con los archivos con css.

```
<script src="ruta_hasta_un_fichero_con_extensión_js">
</script>
```

Al usar esta opción no puede haber contenido dentro de la etiqueta script.

1 – Introducción

Se puede usar la etiqueta script para introducir tanto código JavaScript como se quiera.

Aunque se puede añadir el código JavaScript en cualquier parte del código HTML lo recomendable es incluirlo dentro de la sección <head>.

2 – Comentarios

En JavaScript existen dos tipos de comentarios.

Comentario de línea: //

Comentario de bloque: /*
*/

Los comentarios no son interpretados por el interprete de JavaScript del navegador.

3 – Salida de datos

Mediante Javascript se puede generar salida en diferentes puntos:

Ventana de alerta:

- window.alert("mensaje");
- alert("mensaje");

Con \n se pone un salto de línea.

Consola del navegador:

- console.log("mensaje");

Dentro del documento HTML:

- document.write("mensaje");
- document.getElementById("body").innerHTML = "mensaje“;

3 – Salida de datos

Mediante Javascript se puede generar salida en diferentes puntos:

Dentro del documento HTML:

```
document.write("mensaje");
```

Con la función anterior se pueden añadir elementos a la página web.

```
document.write("<p>Esto es un párrafo</p>");
document.write("<div>");
document.write("Hola "+ nick +", esto es una etiqueta div.");
document.write("</div>");
```

4 – Entrada de datos

Mediante Javascript se puede capturar datos de entrada:

confirm():

```
var respuesta;
respuesta = confirm("¿Seguro que quiere cancelar?");
alert("Ha pulsado: "+ respuesta);
```

prompt():

```
var provincia;
provincia = promtp("Introduzca la provincia");
alert("Ha introducido: "+ provincia);
```

Nota importante

Las funciones vistas anteriormente:

alert();
console.log();
confirm();
prompt();

Se usan principalmente en del desarrollo de la aplicación web. Sustituyendo esas funcionalidades por otras funciones o instrucciones que otorgan una interacción más integrada y fluida.

Nota importante

Las funciones vistas anteriormente:

alert();
confirm();
prompt();

Paran la ejecución del script hasta que el usuario actúe con la ventana emergente.

Esto puede derivar en que la página web no se haya cargado del todo cuando aparece dicha ventana emergente dando una mala experiencia de usuario.

5 – Variables

JavaScript es un lenguaje **débilmente tipado** por lo que cuando se declara una variable no es necesario indicar el tipo de dato que va a almacenar.

Además se puede cambiar el tipo de dato que almacena durante a ejecución del script.

Hay que tener cuidado con este comportamiento cuando se programa.

5 – Variables

Para usar una variable en JavaScript simplemente se utiliza un identificador, nombre de la variable.

Aunque es mejor utilizar la palabra reservada **var** para declarar una variable antes de usarla.

```
var nombre;
var edad;
```

5 – Variables

Tipos de variables

Como se ha visto no se declaran las variables según su tipo.

Aún así, las variables pueden contener los siguientes tipos de valores:

- Números: enteros (7, -13, 26) o reales/decimales (3.1, 5.0, -2.9).
- Lógicos: true false
- Cadenas o strings: se pueden usar comillas simples o dobles, las cadenas son arrays de caracteres.

5 – Variables

Tipos de variables

```
var edad=23, nueva_edad, incremento;

var nombre="Rosa García";

inc=4;

nueva_edad=edad+inc;

alert(nombre +" dentro de "+ inc +" años tendrá "+ nueva_edad +" años");
```

5 – Variables

Tipos de variables

Aunque JavaScript es débilmente tipado, se pueden usar funciones de conversión para forzar el tipo de dato almacenado.

parseInt()

parseFloat()

variable.toString()

5 – Variables

Tipos de variables

```
var num="100";                  // Es una cadena
var num2="100.13";              // Es una cadena
var num3=11;                    // Es un entero
var n=parseInt(num);            // Almacena un entero. Si hubiera habido
                                //                     parte decimal la truncaría.
var n2=parseFloat(num);         // Almacena un decimal
var n3=num3.toString();         // Almacena una cadena
```

5 – Variables

Tipos de variables

Es importante saber que los datos recibidos con la función **prompt** siempre se almacenan como una cadena.

```
var anyo;
var edad;
anyo = promtp("¿En qué año naciste?");
edad = 2019 – anyo;    // Esto producirá un error
edad = 2019 – parseInt(anyo);
alert("Tienes "+ edad +" años. ");
```

6 – Arrays

Arrays

También conocidos como vectores, son una colección de elementos.
Se puede definir un array de diferentes maneras:

```
var vector = new Array();

var frutas = new Array("Manzana", "Pera", "Melocotón", "Sandía");

var consolas = ["Switch", "PS4", "Xbox"];

var alumno = ["Ana", "Gómez", 23];
```

6 – Arrays

Arrays

El acceso a los elementos de un array se realiza con el índice de la posición:

```
var frutas = new Array("Manzana", "Pera", "Melocotón", "Sandía");

var miFrutaFavorita = frutas[2];

for(var i=0; i<frutas.length; i++)
    alert(frutas[i]);
```

6 – Arrays

Arrays

Los elementos de un array pueden ser a su vez otro array, así se crean los arrays multidimensionales:

```
var consolas = [["Wii", "Switch"], ["PS3", "PS4"], ["Xbox", "Xbox ONE"]];

    alert(consolas[2][0]);
```

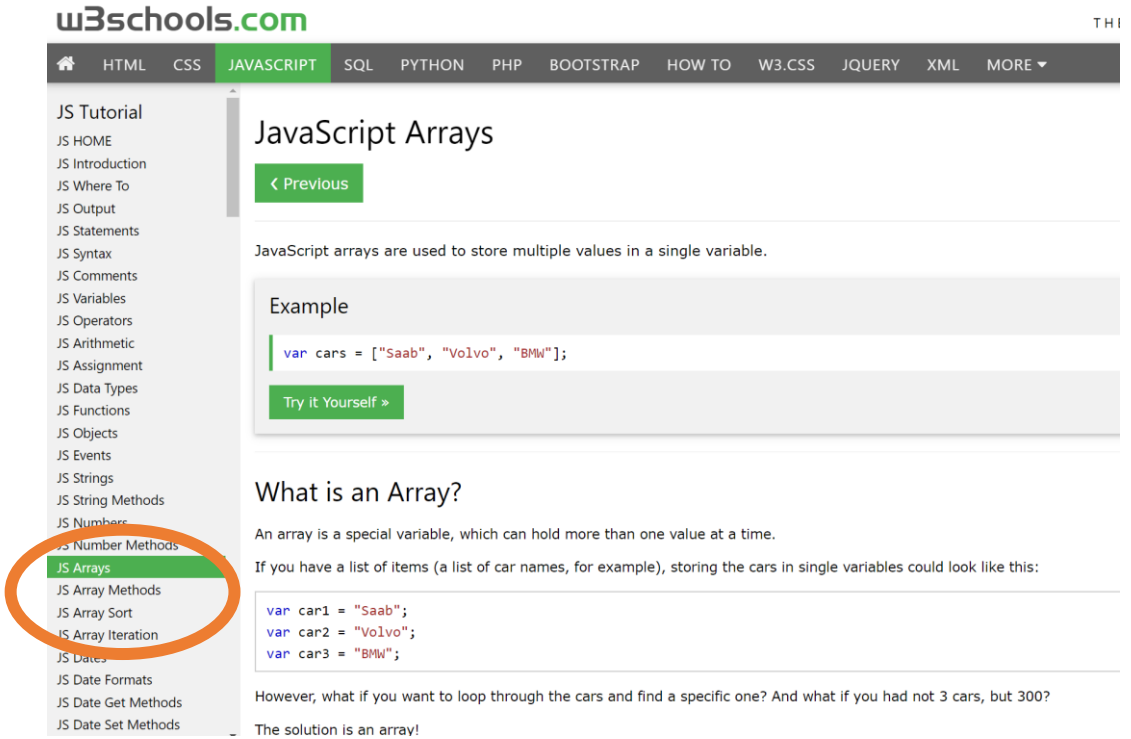
6 – Arrays

Arrays

Sobre los arrays se pueden realizar diferentes acciones mediante funciones.
https://www.w3schools.com/js/js_array_methods.asp

```
miArray.length;
```

Obtiene la longitud del array.



7 – Operadores

Combinando variables, valores y operadores se formulan expresiones más complejas que son parte esencial a la hora de programar.

Existen diferentes tipos de operadores.

Son los mismos que en la mayoría de lenguajes de programación.

7 – Operadores

Asignación	Aritméticos	Comparación	Lógicos
=	+	==	&&
+=	-	!=	||
-=	*	>	!
*=	/	<	
/=	%	>=	
%=	++	<=	
	--		

8 – Estructuras de control

Permiten controlar el flujo de información en los programas.

Igual que pasa con los operadores, son similares a los usados en la mayoría de lenguajes de programación.

A continuación se muestran ejemplos de uso de cada uno de ellos.

8 – Estructuras de control

if

```
if (condición) {
    // instrucciones que se ejecutan si la condición se cumple
}

if (condición) {
    // instrucciones que se ejecutan si la condición se cumple
}
else{
    // instrucciones que se ejecutan si la condición no se cumple
}
```

8 – Estructuras de control

if

```
if (condiciónA) {
    // instrucciones que se ejecutan si la condiciónA se cumple
}
else if (condiciónB) {
    // instrucciones que se ejecutan si la condiciónB se cumple
}
else{
    // instrucciones que se ejecutan si la condición no se cumple
}
```

8 – Estructuras de control

if

```
var dia;
dia=prompt("Introduce el día de la semana ");
if (xdia == "domingo") {
    alert("Hoy es festivo");
}
else
    alert ("Hoy no es domingo, es muy probable que tengas que trabajar");
```

8 – Estructuras de control

if

```
if (condición) {

}
else{
    if (condiciónB) {
        // instrucciones que se ejecutan si la condiciónB se cumple
    }
    else{
        // instrucciones que se ejecutan si la condición no se cumple
    }
}
```

Actividades

Actividad 1

La estructura de control condicional

8 – Estructuras de control

switch

```
switch(expresion) {
    case x:         // bloque de código
                break;
    case y:         // bloque de código
                break;
    default:        // bloque de código
}
```

En case puede ponerse un número, un carácter ("s") o una variable.

Actividades

Actividad 2

La estructura de control condicional múltiple

8 – Estructuras de control

for

```
for (Inicialización del índice; Condición; Modificación en el índice){
    // instrucciones
}
```

```
for (i=0; i<=100; i++) {
    alert(i);
}
```

En case puede ponerse un número, un carácter ("s") o una variable.

8 – Estructuras de control

while

```
while (condición){
    // instrucciones
}
```

```
var i=0;
while (i<=100) {
    alert (i);
    i += 1;     // opciones similares:     i++          i = i + 1;
}
```

8 – Estructuras de control

do – while

```
do {
    // instrucciones
} while(condición);
```

```
do {
    var auxclave = prompt("introduce la clave ","vivaYo");
} while (auxclave != "dracarys")
alert("Has acertado la clave");
```

8 – Estructuras de control

break y **continue**

Se pueden utilizar en los bucles.

La instrucción **break** hace que el bucle deje de ejecutarse.

La instrucción **continue** hace que salte a la siguiente ejecución del bucle.

8 – Estructuras de control

break y **continue**

```
for (i = 0; i < 10; i++) {
    if (i === 3)
        break;
    text += "The number is " + i + "<br>";
}
```

```
for (i = 0; i < 10; i++) {
    if (i === 3)
        continue;
    text += "The number is " + i + "<br>";
}
```

Actividades

Actividad 3

Estructuras de control repetitivas

9 – Funciones

Las funciones y procedimientos aparecieron con la metodología de programación modular.

Permiten extraer partes de código para estructurar los programas y para su reutilización.

En la teoría, la diferencia entre funciones y procedimientos es que las funciones devuelven un valor y los procedimientos no.

En JavaScript tanto funciones como procedimientos se tratan como funciones.

9 – Funciones

La definición de una función consta de las siguientes partes básicas:

- Un identificador (nombre de la función).
- Una lista de parámetros de entrada.
- Las llaves que agrupan las instrucciones incluidas en la función.

Definir una función es simplemente especificar su nombre y definir qué acciones realizará en el momento en que sea invocada, mediante la palabra reservada **function**.

9 – Funciones

Función que no devuelve ningún valor → **procedimiento**

```
function nombrefuncion (parámetro1, parámetro2...){
    // instrucciones
}
```

Función que devuelve un valor → **función**

```
function nombrefuncion (parámetro1, parámetro2...){
    // instrucciones
    return valor;
}
```

9 – Funciones

Las funciones se utilizan mediante una **llamada**:

```
var resultado = suma(23, 74, 18);
```

Para llamar a una función es necesario especificar su nombre e introducir los parámetros que queremos que utilice.

Esta llamada se puede efectuar en una línea de órdenes o bien a la derecha de una sentencia de asignación en el caso de que la función devuelva algún valor debido al uso de la instrucción return.

9 – Funciones

La definición de una función se puede realizar en cualquier lugar del programa.

Se recomienda hacerlo al principio del código o en un fichero "js " que contenga funciones → biblioteca de funciones.

La llamada a una función se realizará cuando sea necesario, es decir, cuando se demande la ejecución de las instrucciones que hay dentro de ella.

9 – Funciones

```
// Definiciones de las funciones
function suma (sumando1,sumando2){
	return sumando1 + sumando2;
}
function profesor (){
	alert ("El profesor es: Álex Torres");
}

var op1=5;op2=25;
var resultado;

resultado=suma(op1,op2);
console.log (op1 +"+"+ op2 +"="+ resultado); // console.log(op1 +"+"+ op2 +"="+ suma(op1, op2));

profesor();
```

10 – Ámbito de las variables

Las variables se pueden utilizar en el ámbito en el que se declaran.

Se deben declarar antes de ser usadas.

El ámbito de una variable se delimita por:

- Las llaves de apertura y cierre que delimitan los bloques de instrucciones.
- El archivo HTML donde se ejecuta el código JavaScript.

10 – Ámbito de las variables

```
var vble global1 = 20;
function prueba(){
    var vble local1 = 10;          //Definición de variable local
    var suma = vble global1 + vble local1;
    alert ("La suma de la vble local (10) y la global (20) es "+ suma);
}

prueba();
alert ("La variable global es "+vble global1);
```

Spoiler

Desde JavaScript se pueden alterar los elementos HTML de una página web y las propiedades de dichos elementos.

Es necesario conocer el valor del atributo **id** del elemento HTML.

Se usa la siguiente función **getElementById()** sobre el documento.

```
document.getElementById('matrix').src = “nuevaFoto.jpg";
```

Actividades

Actividad 4

Funciones