

DESARROLLO WEB EN ENTORNO CLIENTE TEMA2: SINTAXIS DEL LENGUAJE JAVASCRIPT

Índice

2

- Características de JavaScript
- El lenguaje JavaScript: sintaxis
- Variables
- Operadores
- Estructuras de control

1. Características de JavaScript

3

➤ ¿Qué es JavaScript?

- ❑ Lenguaje de programación interpretado utilizado fundamentalmente para dotar de comportamiento dinámico a las páginas web.
- ❑ Cualquier navegador web actual incorpora un intérprete para código JavaScript.

1. Características de JavaScript

4

- Algunas características:
 - ❑ Su sintaxis se asemeja a la de C++ y Java.
 - ❑ Sus objetos utilizan herencia basada en prototipos.
 - ❑ Es un lenguaje débilmente tipado.
- Una página HTML se ejecuta en orden secuencial: una variable o cualquier otro elemento puese ser invocado solo si está cargado en memoria. *Así que todo lo que se encuentre definido en el encabezamiento, es por tanto, visible para los script que vengan después en el body, etc.*

2. El lenguaje JavaScript: sintaxis

5

➤ Mayúsculas y minúsculas:

- ❑ El lenguaje distingue entre mayúsculas y minúsculas, a diferencia de por ejemplo HTML.
- ❑ No es lo mismo utilizar `alert()` que `Alert()`.

➤ Comentarios en el código:

- ❑ Los comentarios no se interpretan por el navegador.
- ❑ Existen dos formas de insertar comentarios:
 - ❑ Doble barra (`//`) – Se comenta una sola línea de código.
 - ❑ Barra y asterisco (`/*` al inicio y `*/` al final) – Se comentan varias líneas de código.

2. El lenguaje JavaScript: sintaxis

6

```
<html>
<head>
  <meta http-equiv="content-type"
        content="text/html; charset=utf-8">
  <title>Uppercase and Lowercase</title>
  <script type="text/javascript">
    function message(){
      alert("Function message called!")
    }
    function Message(){
      alert("Function Message called!")
    }
  </script>
</head>
<body>
  <h2>There is a difference:</h2>
  Lowercase:<input type="button" onclick="message();" value="pulse"/>
  Uppercase:<input type="button" onclick="Message();" value="pulse"/>
</body>
</html>
```

2. El lenguaje JavaScript: sintaxis

7

➤ El punto y coma:

- ❑ Se suele insertar un signo de punto y coma (;) al final de cada instrucción de JavaScript.
- ❑ Su utilidad es separar y diferenciar cada instrucción.
- ❑ Se puede omitir si cada instrucción se encuentra en una línea independiente (*la omisión del punto y coma no es una buena práctica de programación*).

2. El lenguaje JavaScript: sintaxis

8

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script>
    function message(){
      var price=300;
      var quantity=3;
      message1="data";
      alert("Price"+price+"Quantity"+quantity+ "Message"+ message1);
    }
  </script>
</head>
<body>
  <input type="button" onclick="message();" value="pulse"/>
</body>
</html>
```

file://

Price300Quantity3Messagedata

Aceptar

2. El lenguaje JavaScript: sintaxis

9

Message1 es vble global, por eso se conoce su valor por consola.

Price es vble local de la función, por eso no se conoce su valor.

pulse



The screenshot shows a web browser's developer console. At the top, there is a toolbar with icons for the Inspector, Console, Debugger, Red, and Editor de estilos. Below the toolbar, there is a search bar labeled 'Filtrar salida'. The console output shows the following sequence of events:

- A message box with the text 'pulse' is displayed.
- The console shows the command `>> message1` followed by the output `<- "data"`.
- The console shows the command `>> price`.
- An error message is displayed: `Uncaught ReferenceError: price is not defined` at `<anonymous> debugger eval code:1`. A link `[Saber más]` is provided below the error message.
- The console ends with a prompt `>>`.

2. El lenguaje JavaScript: sintaxis

10

➤ Palabras reservadas:

- ❑ Algunas palabras no se pueden utilizar para definir nombres de variables, funciones o etiquetas:

abstract	arguments	boolean	break	byte
case	catch	char	class*	const
continue	debugger	default	delete	do
double	else	enum*	eval	export*
extends*	false	final	finally	float
for	function	goto	if	implements
import*	in	instanceof	int	interface
let	long	native	new	null
package	private	protected	public	return
short	static	super*	switch	synchronized
this	throw	throws	transient	true
try	typeof	var	void	volatile
while	with	yield		

2. El lenguaje JavaScript: sintaxis

11

➤ Tipos de datos:

- ❑ Los tipos de datos especifican qué tipo de valor se guardará en una determinada variable.
- ❑ Los *tres tipos de datos primitivos de JavaScript* son:
 - ❑ Números
 - ❑ Cadenas de texto
 - ❑ Valores booleanos

2. El lenguaje JavaScript: sintaxis

12

1. Números:

- ❑ En JavaScript existe sólo un tipo de dato numérico.
- ❑ Todos los números se representan a través del formato de punto flotante de 64 bits.

2. Cadenas de texto:

- ❑ El tipo de datos para representar cadenas de texto se llama *string*.
- ❑ Se pueden representar letras, dígitos, signos de puntuación o cualquier otro carácter de Unicode.
- ❑ La cadena de caracteres se debe definir entre comillas dobles o comillas simples.
- ❑ Para concatenar cadenas de caracteres: +.

2. El lenguaje JavaScript: sintaxis

13

❑ Cadenas de texto - Secuencias de escape:

Secuencia de escape	Resultado
\\	Barra invertida
\'	Comilla simple
\"	Comillas dobles
\n	Salto de línea
\t	Tabulación horizontal
\v	Tabulación vertical
\f	Salto de página
\r	Retorno de carro
\b	Retroceso

2. El lenguaje JavaScript: sintaxis

14

A.2.1. Crear un fichero HTML que muestre el siguiente texto en una ventana emergente:

I'm = I am

I don't = I do not

A.2.2. Crear un fichero HTML cuyo body contenga un script donde:

- Declara una variable llamada max**Value**, cuyo valor será igual a Number.MAX_VALUE.
- Declara una variable llamada min**Value**, cuyo valor será igual a Number.MIN_VALUE.
- Muestra el valor de ambas variables mediante una ventana emergente.
- Finalmente, en otra ventana emergente muestra el valor resultante de realizar el siguiente cálculo: $\text{maxValue} * 2$

2. El lenguaje JavaScript: sintaxis

15

A.2.3. Crear un fichero HTML y desde el body del mismo invoca al siguiente método declarado en la cabecera del fichero:

```
<script type="text/javascript">
  var a;
  var b=null;
  var c= "aa";
  var d=15;
  var e=true;
  function verTipos(){
    alert(typeof(a));
    alert(typeof(b));
    alert(typeof(c));
    alert(typeof(d));
    alert(typeof(e));
  }
</script>
```

2. El lenguaje JavaScript: sintaxis

16

3. Valores booleanos:

- ❑ También conocido como valor lógico.
- ❑ Sólo admite dos valores: true o false.
- ❑ Es muy útil a la hora de evaluar expresiones lógicas o verificar condiciones.

3. Variables

17

- Se pueden definir como zonas de memoria las cuales se identifican con un nombre y en las cuales se almacenan ciertos datos.
- Dos características del lenguaje:
 - ❑ Declaración de variables opcional
 - ❑ Lenguaje de tipado débil

3. Variables

18

➤ Declaración de variables:

- ❑ Se declaran mediante la palabra clave **var** seguida por el nombre que se quiera dar a la variable:

var mi_variable;

- ❑ Es posible declarar más de una variable en una sola línea:

var mi_variable1, mi_variable2;

- ❑ No se define ningún tipo en el momento de declarar la variable.
- ❑ Una misma variable puede contener tipos de datos diferentes.

3. Variables

19

➤ Declaración de constantes:

- ❑ Palabra reservada `const`, cuya presencia es obligatoria:

```
const iva=21;
```

3. Variables

20

➤ Inicialización de variables:

- ❑ Se puede asignar un valor a una variable de tres formas:
 - ❑ Asignación directa de un valor concreto.
 - ❑ Asignación indirecta a través de un cálculo en el que se implican a otras variables o constantes.
 - ❑ Asignación a través de la solicitud del valor al usuario del programa.

3. Variables

21



A.2.4. Crear un fichero HTML y copia el siguiente código en el cuerpo de la página:

```
<script type="text/javascript">  
  var primer_saludo = "hola";  
  var segundo_saludo = primer_saludo;  
  primer_saludo = "hello";  
  alert(segundo_saludo);  
</script>
```

3. Variables

22

➤ Ámbito de ejecución:

- ❑ Existen dos ámbitos (zona donde una variable puede ser utilizada):

- ❑ **Global:** Utilizada en cualquier parte del código.
Variables globales son aquellas que se definen fuera de cualquier función, o en las que **NO se utiliza el comando var.**

- ❑ **Local:** Cualquier variable definida dentro de una función, es de tipo local si lleva var.

3. Variables

23

➤ Ámbito de ejecución:

- ❑ Existen dos ámbitos (zona donde una variable puede ser utilizada):

- ❑ Variables en javascript:

1. Fuera de toda función, aunque lleve var será **global**

2. Dentro de una función, si lleva var es local y si no lo lleva es global.

3. Variables



A.2.5. Crear un fichero HTML y copia el siguiente código en el cuerpo de la página:

24

```
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <meta http-equiv="X-UA-Compatible" content="ie=edge">
8   <title>Document</title>
9   <script>
10    // Variables en javascript:
11    //     1. Fuera de toda función, aunque lleve var será global
12    //     2. Dentro de una función, si lleva var es local y si no lo lleva es global.
13    if(true)
14    {
15      var true_global = 1; //Esta vble es considerada global, porque está fuera de toda función, aunque lleve var.
16    }
17    if (false){
18      var false_global = 1;
19    }
20    var sum=function(){
21      var local = 1;           //Dentro de una función, si lleva var es local
22      is_it_local = 3;         //Dentro de una función, si no lleva var es global
23      console.log(true_global);
24      return true_global + local + is_it_local; //true_global es considerada global,
25      // por estar declarada fuera de toda función aunque esté dentro de una instrucción.
26    }
27
28    alert("True global"+ true_global); //1
29    alert("false global"+ false_global); //undefined
30    //alert("local" + local); //ReferenceError: local is not defined
31    // alert("is_it_local" + is_it_local); //ReferenceError: is_it_local is not defined
32    alert(sum());
33    document.write(sum());
34  </script>
35 </head>
36 <body>
37
38 </body>
39 </html>
```


4. Operadores

25

- JavaScript utiliza principalmente cinco tipo de operadores:
 - ❑ Aritméticos.
 - ❑ Lógicos.
 - ❑ De asignación.
 - ❑ De comparación.
 - ❑ Condicionales.

4. Operadores

26

➤ Operadores aritméticos:

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

```
var anyo=2015;
var bisiesto= anyo%4;

var variable1=1;
var variable2= ++variable1;
variable2= variable1++;
alert(variable1);
```

4. Operadores

27

➤ Operadores lógicos:

Operador	Nombre
& &	Y
	O
!	No

➤ Operadores de asignación:

Operador	Nombre
+=	Suma y asigna
-=	Resta y asigna
*=	Multiplica y asigna
/=	Divide y asigna
%=	Módulo y asigna

```
var deudas=1500;  
deudas-=300;  
alert(deudas);  
|
```

4. Operadores

28



A.2.6. Crear un fichero HTML y copia el siguiente código en el cuerpo de la página:

```
<script type="text/javascript">
  var operando_1 = eval(prompt("Introduce el primer valor lógico (true o false):", true));
  var operando_2 = eval(prompt("Introduce el segundo valor lógico (true o false):", true));
  var resultado_logico = operando_1 && operando_2;
  alert("Resultado: " + resultado_logico);
</script>
```

Contesta:

- ¿Para qué vale la función `eval()`?
- ¿Para qué se introduce `true` como segundo parámetro de `prompt()`?

4. Operadores

29

➤ Operadores de comparación:

Operador	Nombre
<	Menor que
<=	Menor o igual que
==	Igual
>	Mayor que
>=	Mayor o igual que
!=	Diferente
===	Estrictamente igual
!==	Estrictamente diferente

4. Operadores

30

➤ Operadores condicionales:

- ❑ Permite indicar al navegador que ejecute una acción en concreto después de evaluar una expresión.

Operador	Nombre
?:	Condicional

4. Operadores

31



A.2.7. Crear un fichero HTML y copia el siguiente código en el cuerpo de la página:

```
<script type="text/javascript">
  var dividendo = prompt("Introduce el dividendo: ");
  var divisor = prompt("Introduce el divisor: ");
  var resultado;
  divisor != 0 ? resultado = dividendo/divisor :
    alert("No es posible la división por cero");
    alert("El resultado es: " + resultado);
</script>
```

5. Estructuras de control

33

➤ “if”:

```
if(expresión){  
    //instrucciones  
}
```

```
if(expresión){  
    //instrucciones_si_true  
}else {  
    //instrucciones_si_false  
}
```

```
if(expresión_1){  
    //instrucciones_1  
}else if (expresión_2){  
    //instrucciones_2  
}else{  
    //instrucciones_3  
}
```


5. Estructuras de control

34



A.2.8. La raíz cuadrada de -1 es i (números imaginarios). Crea un documento HTML donde se pida al usuario que introduzca el resultado de realizar la raíz cuadrada de -1 . En caso de que el valor introduzca coincida con lo esperado, mostrar un mensaje que diga *“¡Muy bien! Conoces perfectamente los números imaginarios”*. En caso contrario: *“¡Fatal! Mírate un poco los número imaginarios...”*.

5. Estructuras de control

35



A.2.9. Mostrar un mensaje de si el número introducido en una caja de texto es par o impar. Para ello implementa en la misma página un método llamado *verParImpar()* que lo calcule.

5. Estructuras de control

36

➤ “switch”:

```
switch (expresión){  
    case valor1:  
        //instrucciones a ejecutar si expresión = valor1  
        break;  
    case valor2:  
        //instrucciones a ejecutar si expresión = valor2  
        break;  
    case valor3:  
        //instrucciones a ejecutar si expresión = valor3  
        break  
    default:  
        /*instrucciones a ejecutar si expresión es diferente a  
        los valores anteriores*/  
}
```

5. Estructuras de control

37

➤ “while”:

```
while (expresión){  
    //instrucciones  
}  
  
do{  
    //instrucciones  
} while (expresión)
```

➤ “for”:

```
for(valor_inicial_variable; expresión_condicional; incremento_o_decremento_de_la_variable){  
    //cuerpo_del_bucle  
}
```



Hay otra forma de utilizar los bucles de tipo for, llamada “for-in”. Encuentra y entiende la diferencia con respecto al habitual.

5. Estructuras de control

38



A.2.10. Crear un fichero HTML y copia el siguiente código en el cuerpo de la página:

```
<script type="text/javascript">
  var countdown = prompt("Introduce un número para iniciar la cuenta atrás: ");
  while (countdown>0){
    alert(countdown+ "... ");
    countdown++;
  }
</script>
```

Ahora haz otra versión sustituyendo el bucle *while* por un *for*.

5. Estructuras de control

39

A.2.11. Crear un fichero HTML y copia el siguiente *script*. Añade en el *body* tantas etiquetas “p” como desees y mediante un botón invoca el método. Depúralo.

```
function verBucle(){  
    var etiquetas=document.getElementsByTagName('p');  
    var contador=0;  
    while(contador < etiquetas.length){  
        console.log(etiquetas[contador].innerHTML);  
        contador++;  
    }  
}
```

5. Estructuras de control

40

- `Array.forEach`: El objeto `Array` dispone del método `forEach`, que le permite recorrer todos los elementos del array y ejecutar una función en cada iteración.
 - La especificación del método es:
`Array.forEach(miFuncion);`
 - La de la función debe ser: `function miFuncion(elemento, posicion, array){}`