DWEC
TEMA 2

LA ESTRUCTURA DEL LENGUAGE JAVASCRIPT

P E G O
IES Enric Valor

GENERALITAT VALENCIANA
CONSELLERIA D'EDUCACIÓ, CULTURA I ESPORT

Unió Europea
Fons Social Europeu
El FSE inverteix en el teu futur

ÍNDICE

2.1 SINTAXIS DEL LENGUAJE. OPERADORES Y PALABRAS RESERVADAS

La sintaxis de JavaScript es muy parecida a Java y C++. Cualquier programador que sepa programar en Java, PHP u otro lenguaje con sintaxis similar será capaz de comprender la sintaxis de JavaScript de una manera rápida. No obstante, al ser JavaScript un lenguaje de programación del lado del cliente, es importante conocer sus aspectos básicos como pueden ser los eventos, el control de los elementos HTML, etc.

2.1 SINTAXIS DEL LENGUAJE. OPERADORES Y PALABRAS RESERVADAS

Las 10 reglas básicas de la sintaxis del lenguaje

Regla l. Las instrucciones en JavaScript terminan en un punto y coma. Ejemplo:

```
var s = "hola";
```

Regla 2. Uso de decimales en JavaScript. Los números en JavaScript que tengan decimales utilizarán el punto como separador de las unidades con la parte decimal. Ejemplos de números:

```
var x = 4;
var pi = 3.14;
```

Regla 3. Los literales se pueden escribir entre comillas dobles o simples.

```
var s1 = "hola";
var s2 = 'hola';
```

Regla 4. Cuando sea necesario declarar una variable, se utilizará la palabra reservada *var .*

2.1 SINTAXIS DEL LENGUAJE. OPERADORES Y PALABRAS RESERVADAS

Las 10 reglas básicas de la sintaxis del lenguaje

Regla 5. El operador de asignación, al igual que en la mayoría de lenguajes, es el símbolo igual (=).

Regla 6. Se pueden utilizar los siguientes operadores aritméticos: (+ - * /) . Ejemplo:

```
var x = (5*4)/2+1;
```

Regla 7. En las expresiones regulares, también se pueden utlizar variables. Ejemplo:

```
var t = 4;
var x = (5*t)/2+1;
var y;
y = x * 2;
```

Regla 8. Comentarios en JS. Existen dos opciones:

a) // cuando comentamos una línea.

b) /* y */ todo el contenido entre ambas etiquetas.

2.1 SINTAXIS DEL LENGUAJE. OPERADORES Y PALABRAS RESERVADAS

Las 10 reglas básicas de la sintaxis del lenguaje

Regla 9. Los identificadores en JS comienzan por una letra o una barra baja (_) o un símbolo del dólar ($).

Regla 10. JS es sensible a las mayúsculas y minúsculas (case-sensitive). Ejemplo:

```
var nombre = "Julio";
var Nombre = "Ramón";
```

2.1 SINTAXIS DEL LENGUAJE. OPERADORES Y PALABRAS RESERVADAS

Las palabras reservadas de JavaScript

Palabra	Descripción
var	Utilizada para declarar una variable
if ... else	Estructura condifcional.
for	Estructura de repetición.
do ... While	Estructura de repetición.
switch	Serie de sentencias que va ser ejecutadas dependiendo de diferentes circunstancias.
break	Termina un switch o un bucle.
continue	Sale del bucle y se coloca al comienzo de este.
function	Declaración de una variable.
return	Sale de la función.
try ... catch	Utilidad para el manejo de excepciones.

2.1 SINTAXIS DEL LENGUAJE. OPERADORES Y PALABRAS RESERVADAS

Uso de constantes y variables en JavaScript

Las variables son la base de la programación. Una variable es una posición de memoria donde se pueden alojar datos. El nombre de la variable permitirá a JavaScript poder ubicar y localizar dicho espacio cuando interprete los scripts.

Constantes: En JavaScript, a partir de ES6 (ES2015), se puede utilizar la palabra reservada **const**. Por lo tanto, en vez de utilizar:

```
var pi = - 3.141592;
```

Se aconseja emplear:

```
const PI =  3.141592;
```

Dado que pi es una constante (valor invariable). Por convención, se suele **utilizar mayúsculas** cuando se definen constantes.

2.1 SINTAXIS DEL LENGUAJE. OPERADORES Y PALABRAS RESERVADAS

Uso de constantes y variables en JavaScript

En JavaScript, se pueden ejecutar las siguientes órdenes**:**

```
var x;
x = 2 * x + 1;
var pi = 3.141592;
var paginaweb = “Myfpschool”;
var pregunta = ‘¿cuantos años tienes?’, respuesta=”veinte”;
paginaweb=”Myfpschool.com”;
paginaweb=”Myfpschool” + “.” + “com”;
```

2.1 SINTAXIS DEL LENGUAJE. OPERADORES Y PALABRAS RESERVADAS

Operadores aritméticos en JavaScript

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

2.1 SINTAXIS DEL LENGUAJE. OPERADORES Y PALABRAS RESERVADAS

Operadores de asignación

Operador	Ejemplo de uso
=	X = y;
+=	x += y; // igual que (x = x + y)
-=	x -= y; // igual que (x = x - y)
*=	x *= y; // igual que (x = x * y)
/=	x /= y; // igual que (x = x / y)
%=	x %= y; // igual que (x = x % y)

Operadores de maneja de string.

```
var = "hola" + " " + "mundo";
```

2.1 SINTAXIS DEL LENGUAJE. OPERADORES Y PALABRAS RESERVADAS

Operadores lógicos y de comparación

Operador	Descripción
==	Igual que
===	Igual valor y tipo
!=	Distinto
!==	Distinto valor o distinto tipo
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
?	Operador ternario

El operador condicional (ternario) es el único operador en JavaScript que tiene tres operandos. Este operador se usa con frecuencia como atajo para la instrucción if.

"La Cuota es de: " + (isMember ? "$2.00" : "$10.00")

2.1 SINTAXIS DEL LENGUAJE. OPERADORES Y PALABRAS RESERVADAS

Operadores de tipo

Los operadores de tipo permiten conocer si un objeto es una instancia de un tipo concreto o bien conocer el tipo de una variable. A continuación, se muestran los operadores de tipo

disponibles en JavaScript:

- **typeOf : Devuelve el tipo de una variable.**
- **instanceOf. Devuelve true si un objeto es una instancia de un tipo de objeto.**

2.2 TIPOS DE DATOS. ASIGNACIONES Y EXPRESIONES

JavaScript es un lenguaje bastante simplificado en lo que a tipos de datos se refiere y, por ello, solamente tiene cinco tipos de datos:

1. String.
2. Number.
3. Boolean.
4. Array.
5. Object.

Ver ejemplos:

```
var edad = 25; // Number
var nombre = "Dimas"; // String
var asignatura = ["lengua", "mate", "cono"]; // Array
var persona = {nombre:"Dimas", apellido:"Moreno"}; // Objeto
```

2.2 TIPOS DE DATOS. ASIGNACIONES Y EXPRESIONES

El valor null

Null para los lenguajes de programación es nada o algo que no existe. Generalmente, cuando se asigna null a una variable, es porque es o será un objeto.

Véase un ejemplo de uso del null:

```
var persona = null;
persona = {nombre:"Dimas", apellido:"Moreno"};
```

Conversión entre tipos de datos:

JavaScript es un lenguaje interpretado, por lo tanto, la conversión entre un tipo de dato y otro es transparente al programador. JavaScript asignará el tipo de datos más acertado a la variable que el programador esté utilizando.

No obstante, existen funciones de conversión como **string()** y **number()** para convertir a cadenas de caracteres y números respectivamente.

2.2 TIPOS DE DATOS. ASIGNACIONES Y EXPRESIONES

Conversión entre tipos de datos:

También existen las funciones:

- **parseInt()**. Que convierte una cadena a un número entero.
- **parseFloat()**. Que convierte una cadena a un número decimal.

Es posible también convertir variables de tipo fecha a string con la función **toDateString()**.

```
var fecha = new Date();
var cadena = fecha.toDateString(); // Ahora cadena contendrá la fecha
actual "Sun, 13 Jan 2019".
```

Si lo que se desea es convertir la hora en formato UT C, se puede utilizar alternativamente la función toUTCString().

```
var fecha = new Date();
var cadena = fecha.toUTCString(); // Ahora cadena contendrá la fecha ac-
tual en formato UTC "Sun, 13 Jan 2019 07:45:49 GMT".
```

2.3 INTRODUCCIÓN A LAS FUNCIONES.

Las funciones son uno de los elementos de la base de la programación estructurada. Reutilizar el código es algo básico en cualquier lenguaje de programación porque la regla que se debe seguir es "escribe una vez y utilízala tantas veces como puedas". La variación del resultado de las funciones dependerá de sus argumentos.

```
function suma(a, b) {
    return a + b;
}
```

- Una función puede tener cero, uno o varios parámetros.
- El código de la función estará dentro de las llaves { y }.
- La función devolverá el resultado con la sentencia return.

```
var c = suma(3,3); // c valdrá 6
c = suma(c,3); // ahora c valdrá 9
var text = "El valor de c será ahora: "+c;
```

2.4 INTRODUCCIÓN A LOS OBJETOS.

Desde hace mucho tiempo, la programación estructurada era el único paradigma efectivo y eficiente en programación, pero las cosas cambiaron y nació la programación orientada a objetos, que rompía con lo establecido. No era una evolución, era una nueva filosofía que no tenía nada que ver con lo anterior.

Hoy en día, es prácticamente imposible programar sin utilizar programación orientada a objetos. Esta forma de programar es más cercana a cómo se expresarían las cosas en la vida real que en otros tipos de programación clásicos.

Analistas y programadores piensan y describen realidades y el entorno de una manera distinta para plasmar dichos conceptos en programas en términos de objetos, atributos y métodos.

```
var perro = {
    raza:"Podenco",
    peso:12,
    altura:58,
    color:"negro"
};
```

Para acceder a los atributos se puede hacer de las siguientes maneras:

```
perro.raza;    o    perro["raza"];
```

2.4 INTRODUCCIÓN A LOS OBJETOS.

ACTIVIDAD:

Crea una página web con un script que contenga un objeto de la clase persona, la cual tendrá los siguientes atributos:

- Nombre: Pepe
- Edad: 30
- Altura: 178

Una vez creado el objeto, se deberá mostrar por consola: ″El usuario Pepe tiene 30 años y mide 178 centímetros″

Todo ello utilizando los atributos del objeto.

2.5 VARIABLES Y ÁMBITO DE UTILIZACIÓN.

A continuación, se estudiará la diferencia que existe entre variables globales y locales en JS. Véase ejemplo:

```
var a; // Esta es una variable global
a=10;

function suma() {
    var b = 5;
    return a + b; // la función devolverá 15
}
```

La diferencia entre las variables a y b es que a puede utilizarse en otras funciones que se puedan crear, mientras que b solamente existirá dentro de la función suma.

2.5 VARIABLES Y ÁMBITO DE UTILIZACIÓN.

Las diferencias entre var y let

JS permite declarar las variables como var y let. La diferencia entre ambas palabras reservadas es que, cuando se declara una variable con var, la variable puede utilizarse fuera del bloque donde fue declarada. Véase ejemplo:

```
var a = 10; // x vale 10
{
    let a = 5; // a vale 5
}
// aquí a vale 10
{
    let b=3;
}
// aquí no se puede utilizar b
{
    var c=7;
}
// aquí sí se puede utilizar c y valdrá 7.
```

2.6 EVENTOS.

Los eventos son cualquier suceso que le puede ocurrir a un elemento HTML. Como puede suponer, JavaScript puede darse cuenta de ese evento y reaccionara este ejecutando el código que se haya programado.

Los eventos más comunes son:

- Pulsar un botón.
- Modificar un campo de texto.
- Pulsar una tecla.
- La página ha terminado de cargar-se.
- Hacer clic sobre un elemento HTML.

El formato de programar un evento en JS seria el siguiente:

```
<Elemento_HTML evento='código_JavaScript'>
```

2.6 EVENTOS.

Tipos de eventos en JS

Relacionados en campos de texto

Evento	Ejecutado
onblur	Cuando se abandona un campo de entrada
onchange	Cuando se cambia el contenido de un campo de entrada o cuando un usuario selecciona un valor de una lista desplegable.
onfocus	Cuando obtiene el foco un campo de entrada
onselect	Cuando se selecciona el texto de entrada
onsubmit	Cuando se hace clic en el botón de enviar.
onreset	Cuando se hace clic en el botón de reinicio.
onkeydown onkeypress	Cuando se presiona o mantiene presionada una tecla.
onkeyup	Cuando se deja de presionar una tecla.

2.6 EVENTOS.

Tipos de eventos en JS

Eventos relativos al ratón

Evento	Ejecutado
onclick	Cuando se hace un clic en un botón
ondblclick	Cuando se hace doble clic en un texto
onmousedown onmouseup	Al presionar o soltar un botón del ratón.
onmousemove	Al mover el puntero del ratón sobre una imagen
onmouseout	Al mover el puntero del ratón fuera de una imagen.
onmouseover	Al mover el ratón sobre una imagen.

2.6 EVENTOS.

Tipos de eventos en JS

Eventos de carga

Evento	Ejecutado
onload	Cuando se carga una imagen o una página.
onerror	Cuando se produce un error al cargar una imagen.
onunload	Cuando el navegador cierra el documento.
onresize	Cuando se cambia el tamaño de la ventana del navegador.

Todos los eventos del DOM:

https://www.w3schools.com/jsref/dom_obj_event.asp

2.6 EVENTOS.

Los listener

Otra forma de añadir los eventos a los elementos u objetos HTML del DOM (listener)

```
<!DOCTYPE html>
<html>
<body>

<p id="myFPtext">texto a modificar</p>
<button id="myFPboton">Dale fuerte!</button>

<script>

function dale(){
var x = document.getElementById("myFPtext");
x.innerHTML = "hola";
}

document.getElementById("myFPboton").addEventListener("click", dale,
false);

</script>

</body>
</html>
```

Los parámetros de la función addEventListener() son los siguientes:

1. El evento (sin el "on").
2. La fución por ejecutar.
3. useCapture. Se utilizará si el usuario quiere iniciar la captura para todos los eventos de ese tipo se lancen en el mismo listener. Por defecto en false.

2.7 OBJETOS STRING O CADENA DE CARACTERES.

Uso de comillas simples o dobles:

```
var web = “myfpschool.com”;
var web = ‘myfpschool.com’;
```

En el caso que se quiera utilizar comillas dentro de un string utilizaremos la barra invertida.

```
var web = “la mejor \”web\” de tecnología”;
```

También es posible incluir tabulación (\t) y saltos de línea (\n) dentro de cadenas.

```
alert("hola \tme llamo \nCarlos");
```

2.8 NÚMEROS.

Convertir un número en un string con el método toString()

```
var num = 999;
num.toString(); // devolverá 999 pero como cadena de caracteres
(888).toString(); // devuelve 888 que lo toma del literal 888
(888 + 111).toString(); // realiza la operación y devuelve 999 pero como
cadena de caracteres
```

Ajuste de decimales, número de decimales de un número

```
var num = 9.99;
num.toFixed(0); // devolverá 10. 0 lugares decimales. Realiza redondeo.
num.toFixed(1); // devolverá 10.0 . 1 lugar decimal. Realiza redondeo.
num.toFixed(2); // devolverá 9.99. 0 lugares decimales.
num.toFixed(3); // devolverá 9.990. 0 lugares decimales.
```

Convertir cualquier variable en un número

```
Number(true); // devuelve 1
var num = false;
Number(num); // devuelve 0
```

2.9 FECHAS

La fechas en JS se pueden visualizar como un número o un string.

```
<p id="fecha"></p>
<script>
document.getElementById("fecha").innerHTML = Date();
</script>
```

El resultado de visualizar la fecha actual con el script anterior será algo similar a:

Mon Jun 27 2019 10:45:12 GMT+0200 (CEST)

2.9 FECHAS

Creación de objetos de tipo fecha

En JS existen 4 formas de inicializar un objeto de tipo fecha:

```
new Date()
new Date(milisegundos)
new Date(fechaString)
new Date(año, mes, día, horas, minutos, segundos, milisegundos)
```

Formatos de tipo fecha.

```
var v = new Date("08/16/2016"); // Formato corto
var v = new Date("Ago 16 2016"); // Formato largo
var v = new Date("Fri Ago 16 2019 12:15:24 GMT+0100 (W. Europe Standard
Time)"); // Formato completo
var v = new Date("2019-08-16"); // Formato ISO
```

2.9 FECHAS

Métodos del tipo fecha

Con los métodos del objeto fecha podremos obtener y modificar los datos del objeto fecha. Los mas comunes son:

- getDate(). Obtiene el día del mes (1-31).
- getDay(). Obtiene el día de la semana en formato numérico (0-6).
- getFullYear(). Obtiene el año (2021)
- getHours().
- getMiliseconds().
- getMinutes().
- getMouth().
- getSeconds().
- getTime(). Obtiene la hora en milisegundos desde el 1 **de enero de 1970.**

En el siguiente link podemos obtener todos los posibles métodos del objeto Date

https://www.w3schools.com/jsref/jsref_obj_date.asp

2.10 ARRAYS

Los arrays o vectores son una serie de elementos u objetos continuos, a los cuales pueden accederse utilizando un índice:

```
<p id=»aqui»></p>
<script>
var dias = [«lunes», «martes», «miércoles»,»jueves», «viernes», «sábado»,
«domingo»];
document.getElementById(«aqui»).innerHTML = dias;
</script>
```

También se puede crear de la siguiente forma:

```
var dias = new Array («lunes», «martes», «miércoles»,»jueves», «viernes»,
«sábado», «domingo»);
```

2.10 ARRAYS

Operaciones con arrays

- Conocer su longitud: `longitud = dias.length;`
- Ordenar alfabéticamente: `dias = dias.sort();`
- Añadir un nuevo elemento: `dias = dias.push("juernes");`
- Extraer el último elemento: `ultimo = dias.pop();`
- Eliminar el primer elemento: `dias.shift();`
- Eliminiar un elemento concreto: `delete dias[2];`

2.10 ARRAYS

Arrays multidimensionales

Los arrays multidimensionales consisten en tener en una posición de un array a otro array de elementos. Pueden ser bidimensionales (llamadas tablas), tridimensionales, etc.

Ver ejemplo:

```
var dias = [["lunes", "martes", "miércoles","jueves", "viernes"], ["sá-
bado", "domingo"]];
console.log(dias[0][2]);//mostrará por consola miércoles
console.log(dias[1][1]);//mostrará por consola domingo
console.log(dias[1]);//mostrará por consola sábado,domingo
```

2.11 SENTENCIAS CONDICIONALES

- La sentencia IF:

```
if (new Date().getHours() < 18) {
    saludo = “Buenos días”;
}
```

- La sentencia ELSE:

```
if (new Date().getHours() < 18) {
    saludo = “Buenos días”;
}else{
    saludo = “Buenas tardes”;
}
```

- La sentencia ELSE IF:

```
if (new Date().getHours() < 10) {
    saludo = “Al que madruga Dios le ayuda”;
}else if (new Date().getHours() < 18) {
    saludo = “Buenos días”;
}else {
    saludo = “Buenas tardes”;
}
```

2.11 SENTENCIAS CONDICIONALES

- La sentencia SWITCH:

```
var = new Date().getDay();
switch (var) {
    case 0:
    dia = "Domingo";
    break;
    case 1:
    dia = "Lunes";
    break;
    case 2:
    dia = "Martes";
    break;
    case 3:
    dia = "Miércoles";
    break;
    case 4:
    dia = "Jueves";
    break;
    case 5:
    dia = "Viernes";
    break;
    case 6:
    dia = "Sábado";
    default:
    dia = " - ";
}
```

2.12 BUCLES

- La sentencia o bucle FOR:

```
for (i = 0; i <= 10; i++) {
    text += "Número: " + i + " ";
}
```

- Alternativa del bucle FOR:

```
var persona = {nombre:"Dimas", apellidos:"Moreno", edad:25};
var texto = "";
var x;
for (x in persona) {
    texto += persona[x];
}
```

2.12 BUCLES

- La sentencia o bucle WHILE:

```
i = 0;
while (i <= 10) {
    text += "Número " + i;
    i++;
}
```

2.12 BUCLES

- Recorrer un array con forEach():

```
var txt = «»;
var txt2 = «»;
var  dias  =  ['lunes','martes','miércoles','jueves','viernes','sabado',
'domingo'];
dias.forEach(muestra);
document.getElementById(«aqui»).innerHTML = txt;

function muestra(valor, index, array) {
    txt = txt + valor + «  «;
}
```

Otro ejemplo:

```
const array1 = ['a', 'b', 'c'];

array1.forEach(element => console.log(element));

// expected output: "a"
// expected output: "b"
// expected output: "c"
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach?v=example