Codificación de los elementos del proyecto multimedia

Fundamentos de programación

Codificación

Antes de adentrarnos en el mundo de la codificación y la programación se debe entender el proceso de como una combinación de signos que pueden ser números, letras y símbolos, poseen un valor comprendido sobre un lenguaje posibilitando la formulación y la comprensión de mensajes, algunos ejemplos, como el código Morse, permiten convertir señales intermitentes en números y letras.

Codificación ASCII

ASCII codifican hasta 128 símbolos con diferentes combinaciones, lo que puede generar posibilidades para la inclusión de letras en mayúsculas y minúsculas, signos de puntuación, números y otros caracteres que permiten controlar otros elementos, infortunadamente como es una codificación para idiomas inglés, no puede incluir caracteres como tildes, signos de interrogación, entre otros.

Debido a las limitaciones ofrecidas por ASCII, se dio paso a una combinación de 8 bits, que tampoco contó con el espacio suficiente para la inclusión de los caracteres necesarios para acaparar a todos los usuarios.

Codificación de los elementos del proyecto multimedia

Tabla Unicode

Unicode es una tabla con caracteres numerados con número hexadecimales, por ejemplo una letra M cirílica, denota U+041C; esto significa que este carácter está en la línea 041 y la columna C y así con todos los caracteres que esta tabla representa; el estándar Unicode es internacional, ya que incluye todos los signos de casi todos los idiomas escritos del mundo, jeroglíficos, runas, escritura maya, etc., anotaciones de pesos y medidas, conceptos matemáticos, incluso musicales.

Se debe entender que Unicode no inventa símbolos nuevos, sino que se va actualizando con la información que la sociedad utiliza al día de hoy; por ejemplo con algunos emoticones que han sido utilizados en Japón durante años; hasta la fecha en la última versión 8.0, se codifican aproximadamente 120.000 caracteres.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	€ 20AC	پ 067E	‚ 201A	ƒ 0192	„ 201E	… 2026	† 2020	‡ 2021	ˆ 02C6	‰ 2030	ٹ 0679	‹ 2039	Œ 0152	چ 0686	ژ 0698	ڈ 0688
90	گ 06AF	' 2018	' 2019	" 201C	" 201D	• 2022	– 2013	— 2014	ک 06A9	™ 2122	ڑ 0691	› 203A	œ 0153	ZWNJ 200C	ZWJ 200D	ں 06BA
A0	NBSP 00A0	، 060C	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ھ 06BE	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	؛ 061B	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	؟ 061F
C0	ہ 06C1	ء 0621	آ 0622	أ 0623	ؤ 0624	إ 0625	ئ 0626	ا 0627	ب 0628	ة 0629	ت 062A	ث 062B	ج 062C	ح 062D	خ 062E	د 062F
D0	ذ 0630	ر 0631	ز 0632	س 0633	ش 0634	ص 0635	ض 0636	× 00D7	ط 0637	ظ 0638	ع 0639	غ 063A	ـ 0640	ف 0641	ق 0642	ك 0643
E0	à 00E0	ل 0644	â 00E2	م 0645	ن 0646	ه 0647	و 0648	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ى 0649	ي 064A	î 00EE	ï 00EF
F0	ٍ 064B	ّ 064C	ٌ 064D	ً 064E	ô 00F4	ُ 064F	ْ 0650	÷ 00F7	ù 00F9	ٔ 0652	û 00FB	ü 00FC	LTR 200E	RTL 200F	ے 06D2	

Algoritmos

La creación de algoritmos es un método específico para la creación de un modelo matemático para la solución de un problema en específico.

El diseño de algoritmos es una de las ciencias de la computación para la investigación y resolución de operaciones; básicamente un algoritmo, establece, a modo general, una secuencia de acciones o pasos que pueden solucionar un problema; y su representación, fundamentalmente se basa en diseños de algoritmos en pseudocódigo y/o diagrama de flujo.

Codificación de los elementos del proyecto multimedia

Pseudocódigo

Es la descripción de un algoritmo que asemeja características de programación, pero mezcladas con frases o acciones de un lenguaje más natural, no está regido por algún código y su función principal es encontrar la solución a un algoritmo de la forma más detallada.

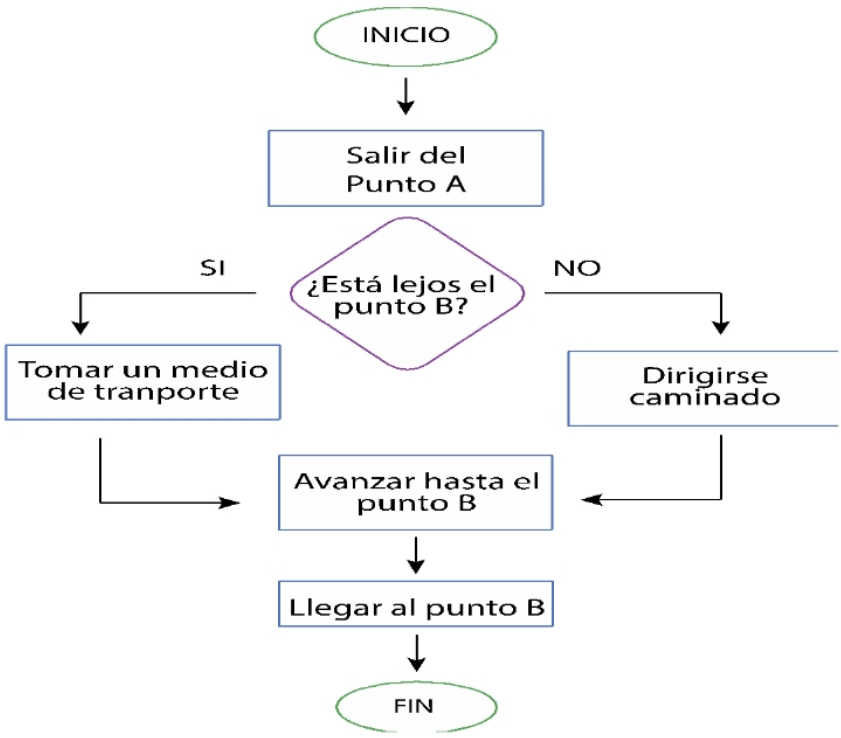
Su representación debe ser sencilla y fácil de manipular, haciendo que el paso de la información facilite la programación.

Por ejemplo, para un problema como averiguar el área de un rectángulo, se indicaría más o menos: El algoritmo calcula el área (a) de un rectángulo cualquiera, si se suministra la longitud de la base y la longitud de la altura.

Diagrama de flujo

Un diagrama es una representación gráfica de una acción en concreto, y cada instrucción está representada por un símbolo que estará conectado por flechas que indican el orden en que se ejecutarán las acciones. Estos diagramas poseen una simbología que se utiliza a modo general y cada uno representa una actividad o acción en el proceso de la solución de un problema.

Al momento de pensar en la creación de algoritmos, debemos tener muy en cuenta que se elaboran de arriba para abajo y de izquierda a derecha, debe iniciar y tener un final y todos los elementos deben ir conectaos por flechas que guiarán los procesos.



Codificación de los elementos del proyecto multimedia

Variables

Una variable es un espacio que permite almacenar información de un determinado tipo de dato que indica de qué tipo será ya que existen tres tipos de variables de datos básicas que son numéricos, de caracteres y lógicos.
Las variables pueden contener muchos tipos de valores, sean textos o números y contienen la cantidad de información que se quiera compactar.

La **variable numérica** se utiliza para almacenar valores numéricos, enteros, reales y pueden realizar operaciones aritméticas como sumas, restas, multiplicaciones, etc.

La **variable tipo carácter** se usa para almacenar caracteres que forman palabras o frases, estas variables vienen encerradas en comillas “ “ con el fin de reconocer y diferenciar.

Las **variables lógicas**, que permiten almacenar alguno de los dos estados lógicos (verdadero o falso).

Las variables se expresan de la misma forma, mediante la palabra VAR y su valor dependerá del tipo de variable, los nombres de las variables se conocen como identificadores que son una secuencia de caracteres acompañados de letras o subrayados, mayúsculas o minúsculas; para asignar una variable, el dato debe usar el operador “ = “.

Variable numérica

```
<script>
    var iva = 16; // variable tipo entero
    var total = 589.45;  // variable tipo decimal
    document.writeln('Iva: '+iva);
    document.writeln('<br />Total: '+total);
</script>
```

Variable tipo carácter

```
cript>
  var nombre = "Carlos Andrés"; //Variable Caracter
  var Direccion= "Cll 24 N 5-18"; //Variable caracter numérico
  var sexo = "M";
  document.writeln('Nombre: '+nombre);
  document.writeln('<br />Direccion: '+Direccion);
  document.writeln('<br />Sexo: '+sexo);
script>
```

Variable tipo carácter

```
<script>
    var clienteRegistrado=false;
    var ivaIncluido=true;
    document.writeln('Cliente registrado: '+clienteRegistrado)
    document.writeln('<br />Iva incluido: '+ivaIncluido);
</script>
```

Codificación de los elementos del proyecto multimedia

Numbers

Son números enteros o decimales, positivos o negativos, se distinguen de otros tipos de números por lo que se indica al declarar la variable.

Strings (cadenas)

Son caracteres alfanuméricos, se denominan variables de texto, ya que se utilizan para almacenar información y se tratarán como texto, además de estar siempre entre comillas.

```
<script>
var numTexto1= "3"; //variable tipo string-texto al asignar el valor entrecomillado
var numTexto2= "5"; //igual tipo que la anterior
document.write("Al usar el operador + con las variables string 3 y 5 obtenemos: ");
document.write(numTexto1 + numTexto2);  //te aparecerá la cadena "35" , al haber sido tratadas como te
xto
var num1 = 3 ;                //variable tipo number-número al asignar el valor sin comillas
var num2 = 5;                 //igual tipo que la anterior
document.write("<br>");
document.write("Al usar el operador + con las variables number 3 y 5 obtenemos: ");
document.write(num1 + num2);  //te aparecerá el resultado de la suma: 8, al haber sido tratado como nú
meros
</script>
```

Booleans

Es una variable que solo puede tener 2 valores, ´true´ y ´false´ y se utiliza para que el programa ejecute una de las 2 vías, si es verdadero, tomará un camino, por el contrario si el resultado es falso, tomará por otro.

```
<script>
    var base;
    var altura;
    var area;
    base = 10;
    altura =5;
    area = (base * altura)/2;
    document.writeln('el area del triangulo es: '+area);
</script>
```

Como se observa se encuentran variables (var) que poseen valores que se suministraron creando otros valores, es decir: la variable var base; tiene un valor de base = 10, sucede los mismo para la variable altura y área, cada una con valores o variables numéricas, asignando información a cada una de las variables que podemos incluir en el script.

Codificación de los elementos del proyecto multimedia

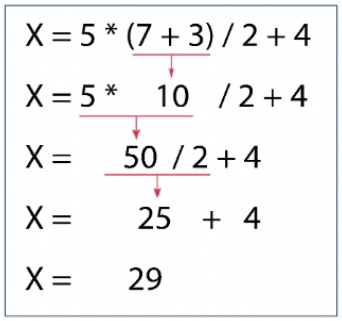
Construcción de fórmulas

Cuando se construyen fórmulas se deben tener en cuenta la jerarquía de las operaciones, ya que estas determinan el orden en el que se resuelven las operaciones aritméticas, este orden permite que una expresión cualquiera se resuelva igual en papel, una calculadora o en el computador.

Operador	Descripción
()	Paréntesis
^	Exponenciación
/ y *	División y multiplicación
+ y -	Suma y resta

Prioridad de las operaciones

Cuando se quiere asignar un orden específico, se deben emplear paréntesis para agrupar de forma correcta y darle prioridad a las operaciones, estas siempre serán las primeras en ejecutarse.



1. **Prioridad de paréntesis.**
2. **Prioridad división y multiplicación.**
3. **Prioridad de operación, solución de izquierda a derecha.**

Arrays (matrices)

Puede contener múltiples valores, su estructura va dentro de { } y sus valores van separados por comas, sus valores pueden ser numéricos, textos o ambos, y sus elementos van en posiciones numeradas que van de 0 en adelante.

```
<script>
var granPoblacion = ["Madrid", "Barcelona", "Valencia", "Sevilla"];       //declaramos el array
document.write("granPoblacion[2]: " + granPoblacion[2] + "<br>");         //imprimimos el valor '2'
granPoblacion[2] = "Málaga";                                              // cambiamos el valor '2'
document.write("Cambiamos Valencia por: " + granPoblacion[2] +"<br>");    //imprimimos el nuevo valor
'2'
granPoblacion[4] = "Zaragoza";                                          //añadimos el valor Zaragoza al arra
y
document.write("Hemos añadido al array: " + granPoblacion[4]);            //imprimimos el nuevo valor
</script>
```

También se pueden crear arrays vacíos, para luego agregar elementos como en el siguiente ejemplo:

```
var ciudades = [];
ciudades[0] = "Cuenca";
ciudades[1] = "Gijón";
ciudades[2] = "Granada"
```

Un array es un tipo de objeto con diversos métodos que permite manipular valores almacenados. La estructura para llamarla es: **nombreArray.metodo ( )**

Método PoP: ***nombreArray.pop ( )***
Este método elimina el último elemento del array, para el ejemplo anterior usaríamos un script: ciudades.pop ( ) y eliminaría la última ciudad.

Método Push: ***nombreArray.push ( )***
Este método añade nuevos valores al final del array, para el ejemplo anterior usaríamos un script: ciudades.push ("Lugo", "Castilla", "Oviedo") sumando valores al array.

Método Shift ( ) y unshift ( ): ***nombreArray.shift ( ) y nombreArray.unshift ()***
Esto dos método trabajan al principio de la lista del array, para el primer ejemplo, eliminaría el primer valor del array ***ciudades.shift ( )***, para el segundo array, su tarea es añadir elementos al principio del array, ***ciudades.unshift*** ("León", "Ciudad Real").

Método Splice: ***nombreArray.splice ( )***
Este método permite insertar o eliminar elementos en cualquier parte del array, eliminando de la siguiente forma:
ciudades.splice (2,3):
ciudades.splice (1,0 "Cadiz", "Victoria", "Tarragona")

Método IndexOf ():
Nos da la ubicación que ocupa un array, buscando solo la primera coincidencia en caso de estar repetido.
ciudades.indexOf ("Oviedo")

Funciones

Son elementos que empaquetan y aíslan una parte del código para que realice una tarea específica, podemos decir que son un paquete de instrucciones que ejecutan una tarea determinada; en este ejemplo vemos una función llamada Saludo.

Codificación de los elementos del proyecto multimedia

Funciones

Son elementos que empaquetan y aíslan una parte del código para que realice una tarea específica, podemos decir que son un paquete de instrucciones que ejecutan una tarea determinada; en este ejemplo vemos una función llamada Saludo.

```
<script>

//de                                  e caso la llamamos saludo()

function saludo() {
   document.write("Hola, este es el resultado de la función saludo");
}

//llamamos a la función saludo() para que ejecute sus instrucciones

saludo();

</script>
```

Declaración de la función

En este ejemplo se hace o se declara la función (function) y se le da un nombre a dicha función, en este caso (saludo) y los paréntesis que identifican la función; además está el componente que se encuentra entre paréntesis, son el bloque de las instrucciones que incluirán la información que se llevará a cabo:

```
function saludo() {.....}
```

1. La palabra clave 'function'
2. El nombre que queremos darle a dicha función, en este caso: saludo
3. Unos paréntesis que añadimos al nombre para identificarla como función: saludo()
4. Un bloque de instrucciones que queda encerrado entre llaves{ } En este caso solo se incluye (*un document.write*), pero podrían ser varias que se explorarán en un futuro. Las instrucciones deben llevar al final el correspondiente punto y coma (;) con el que termina la instrucción.

Llamar la función

1. Se hace con la instrucción saludo ()
2. En este ejemplo observamos que la función tiene unos paréntesis vacíos; en este espacio es posible insertar valores de funciones dependiendo claro está la función establecida.

Codificación de los elementos del proyecto multimedia

Funciones con parámetros

Vamos a construir un ejemplo de una función con parámetros establecidos, lo primero que se debe plantear en el script es el problema que se pretende solucionar, en este ejemplo vamos a calcular el volumen de una esfera, es decir:

$$Volumen = 4/3 * \pi * R3$$

Ejemplo de una función

1. Se solicitará al usuario por medio del Script que introduzca el valor que desea conocer.
2. Internamente el script, solucionará el problema matemático y solucionará gracias a la función y los parámetros la respuesta.
3. El valor de "PI" lo cambiaremos por el valor más cercano, aunque existen funciones que calculan potencias, y cientos de funciones más de las que hablaremos más adelante.

```
var radio = prompt("Introduce la longitud del radio de la esfera: ");
var volumen = 4/3 * 3.14 * radio*radio*radio;
document.write("El volumen de la esfera es: " + volumen);
```

4. Se establecerán los parámetros que calcularán la función y se denominará volEsfera ().

```
function volEsfera(x) {
   var volumen = 4/3 * 3.14 * x*x*x;      //cálculo del volumen
   return volumen;                        //valor que devuelve la función
}
```

5. Una vez establecida la función se puede definir y luego llamarla de la siguiente forma:

```
<script>

function volEsfera(x) {
 var volumen = 4/3 * 3.14 * x*x*x;
 return volumen;
}

/* Aqui y/o antes de la función, podria ir más código, si el script realiza más cosas.
En el momento en que quisiéramos utilizar la función, la llamaríamos con: volEsfera(argumento) */

var radio = prompt("Introduce la longitud del radio de la esfera: ");
document.write("El volumen es: " + volEsfera(radio));

</script>
```

En este ejemplo no se declara una función vacía (); por el contrario la declaramos como function volEsfera (x), donde “x” o cualquier valor que se quisiera dar, le llamamos parámetro de función y al valor que le pasamos a la variable radio, la llamamos argumento.

Como hemos visto en los ejemplos, una función puede:

a) No tener parámetros, como en el ejemplo; saludo ()
b) Tener un parámetro; como en el ejemplo; function volEsfera(x)
c) Tener varios parámetros; como veremos en el siguiente ejemplo.

```
<script>
//introducimos un precio y descuento a un producto
var precio = prompt("Introduzca Precio producto (en euros): ");
var descuento = prompt("Introduzca descuento (en euros): ");

//definimos la función calculoPVP, que tiene como parámetros los valores de precio y descuento
function calculoPVP(precio, descuento) {
   var IVA = 1.21;
   var PVP = (precio - descuento) * IVA; //en la variable PVP hemos almacenado el PVP calculado
   return PVP;  //la función devuelve el valor asignado a la variable PVP
}

document.write("Precio= " + calculoPVP(precio,descuento) +  " €");
</script>
```

En este script, tenemos una función de dos parámetros (precio y descuento) que calcula el PVP y aplica, además, el IVA.

Return: devolución de un resultado

En el ejemplo saludo () , la función ejecuta un `document.write´, que es una función que muestra en pantalla una frase o un “documento escrito” como resultado.

Pero si se requiere que este valor se utilice en el resto del programa, deberemos volver el valor por la función, utilizando la palabra ´return´. Es decir que para este ejemplo, la función nos devuelve el valor de la variable ´volumen´ y el valor de la variable ´PVP´.

Variables locales y globales

Las variables que se definen dentro de las funciones, siempre se declaran con su palabra clave ´var´ y se denominan variables locales, ya que no pueden ser accedidas desde el exterior de la función, para estos ejemplos, serían las variables ´volumen´, ´IVA´ y ´PVP´.

Se debe tener presente que al momento de escribir código, las variables no pueden entrar en conflicto con otras variables del mismo nombre que se encuentren a lo largo del script.

Por otra parte, las variables globales, son aquellas que se definen fuera de las funciones o dentro de las pero sin la palabra clave ´var´ que tiene efecto en todo el script; se dice que una variable, puede tener ámbitos (scope) locales o globales.

Codificación de los elementos del proyecto multimedia

Algunos beneficios

Las funciones permiten mejorar las estructuras de los script, generando y facilitando la construcción de los programas; en los ejemplos anteriores las instrucciones no se encuentran repetidas por todo el script, haciendo que sea más sencillo al momento de modificar algún dato y no tener que modificarlos uno por uno.

Las funciones aíslan y agrupan o empaquetan las variables e instrucciones que realizan tareas específicas y solo se ejecutan si son llamadas desde el código principal y tras finalizar sus instrucciones, devuelven un resultado a la parte donde esta lo invocó. Estos valores internos no entran en conflicto con otros códigos del resto del programa.

Condicionales y ciclos

Una condicional como su nombre lo indica, es una condición para distinguir y entender entre una opción y la otra, el proceso más claro es tomar un camino dependiendo SI esto sucede o NO. Para crear condicionales, lo primero que se necesita conocer son los operadores lógicos.

Símbolo	Significa	Ejemplo	Significado del ejemplo
==	"Igual"	"If X==Y"	" Si X es igual a Y"
>	"Mayor que"	"If X>Y"	" Si X es mayor que Y"
<	"Menor que"	"If X<Y"	" Si X es menor que Y"
!	"Si es distinto"	"If X!=Y"	" Si X es distinto o igual a Y"
&&	"Y" conjunción copulativa	"If (x==y) && (x==z)"	" Si X es igual a Y y X es igual a Z"
ll	"O" conjunción adversativa	"If (x==y) ll (x==z)"	" Si X es igual Y o X es igual a Z"

IF and ELSE
If, desencadena los condicionales en el código, ¿qué pasa si?; por el contrario, ELSE, expresa en el caso contrario; ejemplo: Si (IF) sacas un número par, toma a la derecha, pero, (ELSE) si sacas un número impar toma a la derecha.

SWITCH
Es una estructura que está ligada a una decisión, su sintaxis se basa en la palabra switch and case, y plantean la información donde el código se dispara, con unas variantes para ese valor, y se delimitan con la palabra break.

BUCLE FOR
El ciclo más usado ya que gracias a este se pueden accionar todas las demás condicionales, pero para tener buenas prácticas, es necesario usar las condicionales específicas.
La sintaxis del script es la siguiente:

for(condicional inicial; condicional de parada; ritmo de iteración)

La **condición inicial** debe empezar en 0, dándole un inicio al bucle;
for(x=0; condicional de parada; ritmo de iteración);

La condición de parada es el valor cuando se decide que el bucle pare, para este ejemplo pondremos un valor de 8; es decir que el bucle se detendrá al llegar al valor 8;
for(x=0; x<8; ritmo de iteración);

El **ritmo de iteración**, es el ritmo en el que se consume o se reproduce el bucle, lo más común al momento de expresarlo es X++, es decir se hará un bucle un sola vez;
for(x=0; x<8; x++).

BUCLE WHILE
Este bucle es muy parecido al bucle for la gran diferencia a la estructura del script, es que solo se debe pasar la condición de parada While y la gran diferencia con el bucle for es que no necesita una condición de iteración.

Programación orientada a objetos

Se pueden definir los objetos como modelos con el que representamos conceptos o elementos, utilizadas para representar variables o datos y a su vez son características propias de esos objetos o elementos; a estas variables las denominamos propiedades o atributos.

Pensemos que para un ejemplo como celular, podríamos pensar en marcas, modelos, pixeles en la cámara, procesador, etc. Todo dependerá del contexto y la información que necesitemos.

Además podemos definir métodos, que a vienen a trabajar como una serie de instrucciones que cambian los valores asignados a estas propiedades.

JavaScript

Es un lenguaje que se basa en los objetos, de hecho, todos los elementos podrían ser tratados como objetos, incluyendo datos tipo ´string´ ´arrays´ y ´number´ a los que se le pueden aplicar diferentes acciones, este lenguaje JavaScript incluye una serie de objetos que vienen predefinidos y nos ayudan a incorporar diferentes acciones, a estos objetos se les denomina ´objetos built-in´.

Como podemos analizar, los métodos son las mismas funciones, pero en ellos acoplamos instrucciones a las que se llaman y aplican sobre objetos ya definidos, especificando así estos objetos con las notaciones: **objeto.metodo ()**

Codificación de los elementos del proyecto multimedia

Ejemplos sobre cómo funcionan los objetos

```
1. Math.sin(55);
2. Numerol.toFixed(5);
3. texto1.search("Digital Learning");
```

En estos ejemplos identificamos los siguientes elementos.

OBJETOS	1.	Math	MÉTODOS	1.	sin	ARGUMENTOS	1.	55
	2.	Numerol		2.	toFised		2.	5
	3.	Texto1		3.	search		3.	"Digital Learning"

Y respectivamente realizarán las operaciones.

1. Calcula el seno de 55°.
2. Limita a cinco decimales el valor de un número.
3. Busca la cadena "Digital Learning" en un texto.

Es importante reconocer las diferencias entre los objetos creados y los objetos declarados que se pueden encontrar en un script y, para diferenciarlos, es preciso entender:

Notación literal: se utilizan llaves {} para encerrar la declaración de las propiedades y métodos en los objetos.

```
var asignatura = {                          //definimos objeto asignatura
   nombre: "Historia",                      //definimos sus propiedades (notar el uso de ':' y ',')
   docente: "Irene Castillo Moragas",
   horasAnuales: 60,
   horasRestantes: function (horasImpartidas) {     //definimos método 'horasRestantes'
      return this.horasAnuales - horasImpartidas;
   }
};
```

Podemos observar para el anterior ejemplo, la siguiente información:

1. Los valores de las propiedades se asignan con dos puntos (:) y no con (=).
2. Las propiedades se separan por comas (;) como en los valores de los arrays y no con (;).
3. Se definió una función a un objeto, al que se llamó ´horas restantes´, como está dentro del objeto lo llamaremos método.
4. La palabra this, reemplaza el nombre del objeto, aunque podría haberse referido ´asignatura.horasAnuales´.

Codificación de los elementos del proyecto multimedia

Construcción

Construiremos un objeto vacío ´new Object ()´; al que luego se le podrá asociar propiedades con valores correspondientes, a través de diferentes notaciones, emplearemos lo que se denominan ´funciones anónimas´ y se pueden ejecutar como si fueran una expresión.

```
var asignatura = new Object();        //creamos el objeto 'asignatura'

//ahora le añadimos las propiedades: nombre, docente, horaAnuales y horas Impartidas
    asignatura.nombre = "Historia";
    asignatura.docente = "Irene Castillo Moragas";
    asignatura.horasAnuales = 60;

//definimos un método denominado 'horas restantes' para el objeto 'asignatura', utilizando una función anónima
    asignatura.horasRestantes = function () {
        return this.horasAnuales - this.horasImpartidas;
    };
```

Programación orientada a eventos

Es una opción en la programación que permite determinar los flujos de control de un programa, es decir, separar las etapas o las fases de los eventos de un código de un programa, utilizando eventos que llamarán una función o un método.

Generalmente y grosso modo, se crea un bucle principal que permite activar acciones individuales, enfocando qué ejecutar y en qué orden. Se debe tener en cuenta y comprender muy bien estos tres elementos que nos ayudarán a comprender esta programación.

Eventos
Son las acciones sobre el programa, ayudan a producir o disparar esos eventos sobre un componente y estas conllevan a la seguida de acciones programadas por el usuario.

Ejemplo: Un botón.

Cuando el cursor se posiciona sobre el botón, ejecuta una acción.
Cuando el cursor hace un clic largo, ejecuta otra acción.
Cuando el curso hace clic, ejecuta otra acción.

Codificación de los elementos del proyecto multimedia

Propiedad
Es una asignación que describe un componente y dependiendo de su propiedad tiene la capacidad de ejecutar y controlar, estas propiedades, controlan la aplicación, por ejemplo la alineación, la forma de visualización, el tamaño del texto, etc.

Métodos
Función que se llama desde el programa, en su mayoría vienen preprogramados y ayudan a simplificar procesos típicos, y permiten que el programador tenga herramientas que a su vez permiten limpiar el código y hacerlo más sencillo de entender.

Dada su facilidad al momento de trabajar con eventos, el procesamiento es más rápido y permite que la interactividad sea mayor, siendo esta una de sus principales ventajas, pensando siempre en mejorar la experiencia del usuario, ayudando a generar experiencias propias, sin restricciones y con variedad de opciones.

Método para el tratamiento de números

Como ya hemos visto JavaScript nos permite tratar los datos como objetos y existe un objeto que contiene una información que aplica cálculos y diferentes funciones matemáticas, etc.

Math es una calculadora científica con un gran número de decimales y JS nos permite nos permite disponer de toda esa información y poder editarla a nuestro favor.

toFixed: fija el número de decimales que tendrá el número, redondeándolo.
variableNumerica.toFixed(x) , siendo X la cantidad de numerales que se quieren redondear.

toPrecision: fija el número de dígitos que contiene el número, haciendo su redondeo.
variableNumerica.toPrecision(x)

toExponential: devuelve el número en notación exponencial, siendo X el número de decimales determinado.
variableNumerica.toExponential(x)

```
<script>
var numeroEjemplo = 32.73412;  //recordar que los decimales se separan con punto ( . )
var numero0 = numeroEjemplo.toFixed(0);
var numero1 = numeroEjemplo.toFixed(1);
var numero2 = numeroEjemplo.toPrecision(2);
var numero3 = numeroEjemplo.toExponential(3);

document.write("Al aplicar toFixed(0) a " + numeroEjemplo + " obtenemos: " + numero0 +  "<br>");
document.write("Al aplicar toFixed(1) a " + numeroEjemplo + " obtenemos: " + numero1 + "<br>");
document.write("El aplicar toPrecision(2) a " + numeroEjemplo + " obtenemos: " + numero2 + "<br>");
document.write("Al aplicar toExponential(3) a " + numeroEjemplo + " obtenemos: " + numero3);

</script>
```

Codificación de los elementos del proyecto multimedia

Como hemos ya entendido a lo largo de este temario, sabemos diferenciar una variable tipo texto (string) y otro de tipo numérico (number), lo que nos facilita al momento de realizar el script y no confundirnos, dado que los tratos son diferentes; en este momento conviene saber que existen unas funciones generales, que nos permiten realizar dichas conversiones.

parseInt (“String”)
Convierte la cadena o el texto de números a enteros, si el número tiene algún decimal, se elimina y no se redondea, se ubica el número entero que se presente.

```
var numEntero = parseInt (“65.876”) ; // el valor de ´numEntero´ es (65)
```

parseFloat (“String”)
Convierte la cadena o el texto conservando la parte decimal.

```
var numDecimal = parseFloat (“2.753”) ; // el valor de ´numDecimal´ es (2.753)
```

Number (“String”)
Convierte la cadena de texto, tal cual lo encuentre, tenga o no tenga decimal.

```
var num1 = Number (“9.73”) ; // el valor de ´num1´ es (9.73)
var num2 = Number (“245”) ; // el valor de ´num2´ es (245)
```

variableNumerica.toString ( )
Esta variable por su parte, convierte cualquier cadena de número a su cadena de texto equivalente.

```
var numEjemplo = 7564;
var numEjComoTexto = numEjemplo.toStaing ();
```

La variable ´numEjComoTexto´, toma el valor de “7564”