UNIVERSIDAD NACIONAL DE EDUCACIÓN

Enrique Guzmán y Valle

Alma Mater del Magisterio Nacional

FACULTAD DE CIENCIAS

Escuela Profesional de Matemática e Informática

MONOGRAFÍA

LENGUAJES DE PROGRAMACIÓN JAVASCRIPT

Java y Javascript. Características. Norma de escritura. Variables y operadores lógicos. Mensajes. Ejercicios. Estructuras condicionales. Funciones y objetos. Aplicaciones.

Examen de Suficiencia Profesional Resolución N° 1317-2018-D-FAC

Presentada por:

Roque Erickson ROMAN ARENAZA

Para optar al Título Profesional de Licenciado en Educación especialidad: Informática

Lima, Perú

2019

MONOGRAFÍA

LENGUAJES DE PROGRAMACIÓN JAVASCRIPT

JAVA y Javascript. Características. Norma de escritura. Variables y operadores lógicos. Mensajes. Ejercicios. Estructuras condicionales. Funciones y objetos. Aplicaciones.

Examen de suficiencia profesional Resolución N° 1317-2018-D-FAC

Dr. Richard Santiago QUIVIO CUNO

Presidente

Dr. Juan Carlos HUAMAN HURTADO

Secretario

Dr. Guillermo Pastor MORALES ROMERO

Vocal

Línea de Investigación: Tecnologías y Soportes Educativos

DEDICADO A:

Mis padres: Javier y Ana

A toda mi familia, por todo su apoyo constante y creer siempre en mi persona.

RECONOCIMIENTO

A la Universidad Nacional de Educación Enrique Guzmán y Valle – Facultad de Ciencias, por haberme aceptado ser parte de ella para poder seguir mis estudios en la especialidad de Informática

Tabla de cContenidos

Lista de Figuras

Lista de Tablas

Introducción

Este trabajobusca dar a conocer algunos alcances primordiales sobre la programación.Deforma práctica y comprensible semencionarálos principales aspectos de JavaScript

Para escribir un programa no vale ni el castellano yni ninguno de loslenguajes que habitualmente usa el hombre para comunicarse. Para entendernos con uncomputador se utilizan los lenguajes informáticos. Se trata de un lenguaje de tipo script compacto, basado en objetos y guiados por eventos diseñadosespecíficamente para el desarrollo de aplicaciones cliente -servidor dentro del ámbito de Internet.

Los programas JavaScript van incrustados en los documentos HMTL, y se encargan de realizar acciones en el cliente, como puede ser; pedir datos, confirmaciones, mostrar mensajes, crear animaciones, comprobar campos etc.El programa que va a interpretar los programas JavaScript es el propio navegador, lo que significa que, si el programa no soporta JavaScript, no podremos ejecutar las funciones que programemos. Desde luego, Netscape y Explorer lo soportan, el primero desde la versión 2 y el segundo desde la versión 3.

Ahora que ya conocemos nuestra realidad donde las tecnologías cada día son más importantes y el manejo de estas herramientas son obligatorias, sobre todo los nativos digitales y los nuevos profesionales, creo que el conocimiento de Javascript debe ser prioritario.

Capítulo I: Lenguajes de programación

1.1. Lenguajes de programación

Lenguaje de programación es un conjunto de sintaxis y reglas semánticas que definen los programas del computador. Es una técnica estándar de comunicación para entregarle instrucciones al computador. Un lenguaje le da la capacidad al programador de especificarle al computador, qué tipo de datos actúan y que acciones tomar bajo una variada gama de circunstancias, utilizando un lenguaje relativamente próximo al lenguaje humano. Un programa escrito en un lenguaje de programación necesita pasar por un proceso de compilación, interpretación o intermedio, es decir, ser traducido al lenguaje de máquina para que pueda ser ejecutado por el computador.

Los lenguajes de programación se clasifican en tres grandes categorías: lenguaje máquina, lenguaje de bajo nivel y lenguaje de alto nivel.

Loslenguajes máquinas son aquellos cuyas instrucciones son directamente entendibles por la computadora y no necesitan traducción posterior para que la UCP pueda comprender y ejecutar el programa.La programación en lenguaje maquina es difícil, por ello se necesitan lenguajes que permitan simplificar este proceso.

Figura I

Los lenguajes de bajo nivel (ensambladores) han sido diseñados para ese fin.

Los lenguajes de programación de alto nivel (Fortran, Pascal, Cobol, Java, etc.) son aquellos en los que las instrucciones o sentencias a la computadora son escritas con palabras similares a los lenguajes humanos, en general lenguaje inglés, como es el caso de Java, lo que facilita la escritura y la fácil comprensión por el programador.

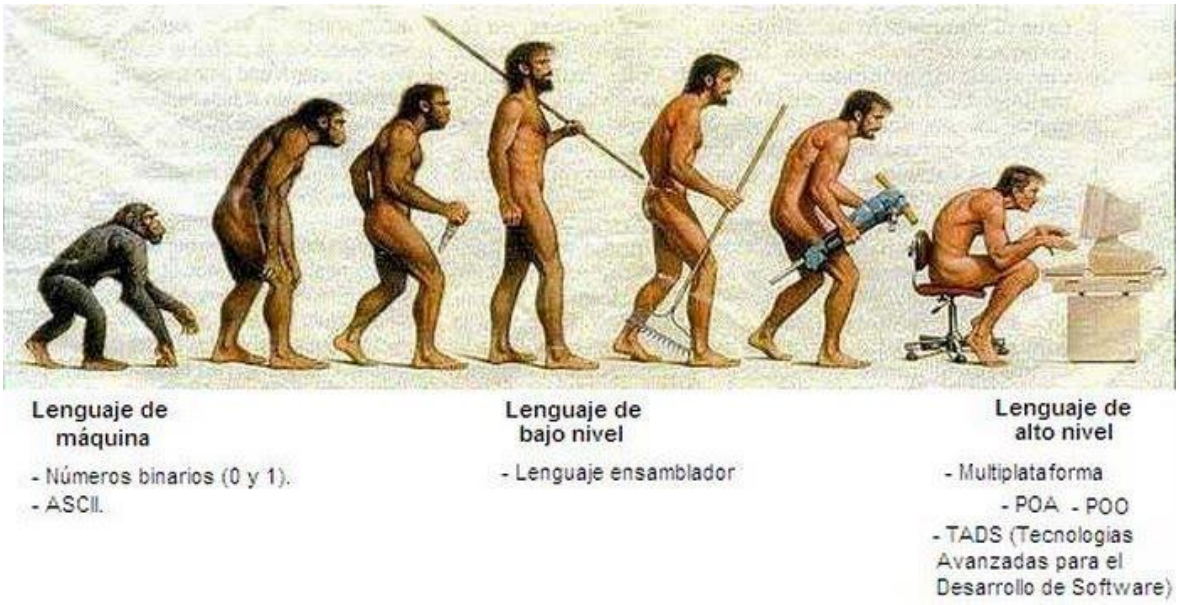


Figura 2

1.2. Historia de los lenguajes de programación

Con la idea de facilitarnos las tareas que debemos de desempeñar los humanos, hemos venido inventado diversas herramientas a lo largo de nuestra historia, que nos permiten tener una mejor calidad de vida.

Los computadores son uno más de los inventos del hombre, aunque debemos decir que las tecnologías para su fabricación y explotación han tenido un desarrollo sorprendente en las últimas décadas. Esta herramienta por sí sola no es capaz de efectuar ninguna tarea, es tan sólo un conjunto de cables y circuitos que necesitan recibir instrucción por parte de los humanos para desempeñar alguna tarea. El problema entonces, se puede fijar en cómo vamos a poder hacer que un conjunto de circuitos desempeñen una determinada tarea y nos entreguen los resultados que nosotros

esperamos, es decir, de qué manera se puede lograr la comunicación entre el hombre y el ordenador.

Así pues, tratando de dar una solución al problema planteado, surgieron los lenguajes de programación, que son como un lenguaje cualquiera, pero simplificado y con ciertas normas, para poder trasmitir nuestros deseos al ordenador.

Por otro lado, como se sabe, un conjunto de circuitos no entendería ningún lenguaje que nosotros conozcamos, por más sencillo que éste parezca. Los circuitos en todo caso, sólo reconocen presencia o ausencia de energía, es decir que debemos hablarle a la máquina en su propio lenguaje (presencia y ausencia de energía, 0 y 1), o nuestro lenguaje deberá de ser traducido a un lenguaje binario cuyo alfabeto es el 0 y el 1, mediante las herramientas desarrolladas para llevar a cabo esta tarea, las cuales reciben el nombre de traductores, y como veremos más adelante, los hay de muchos tipos, dependiendo de características más específicas del lenguaje a traducir y de la manera de llevar a cabo su traducción.

Entonces, para crear un lenguaje de programación, deberemos crear la herramienta que lo traduce, y es justamente de ellas, de las que hablaremos a continuación, para describir como han ido evolucionando en los últimos 50 años.

- 1946: KonradZuse, ingeniero Alemán, mientras trabajaba en los Alpes de Bavaria, desarrolló el lenguaje Plankalkul, el cual, fue aplicado entre otras cosas para jugar al ajedrez.
- 1949: Aparece Short Code, que viene a ser el primer lenguaje que fue usado en un dispositivo de cómputo electrónico, aunque se debe decir que se trata de un lenguaje traducido a mano.

- 1951: Grace Hopper, trabajando para Remington Rand, comenzó el trabajo de diseño del primer compilador conocido ampliamente, el A-0, el cual, al ser liberado por la compañía en 1957, lo hizo con el nombre de MATH-MATIC.
- 1952: Alick E. Glennie, durante su tiempo libre en la Universidad de Manchester, concibe un sistema de programación llamado AUTOCODE, que viene a ser un compilador muy rudimentario.
- 1957: aparece FORTRAN (FORmulaTRANslating) sistema traductor de fórmulas matemáticas. Fue desarrollado por un equipo, al frente del cual se encontraba John Backus quien después vendría a contribuir en el desarrollo del compilador para el lenguaje ALGOL (para aplicaciones científico-matemático) y de la notación usada para la especificación sintáctica de los lenguajes, conocida como BNF (Backus Naur Form).

A partir de los años sesenta, empiezan a surgir diferentes lenguajes de programación, atendiendo a diversos enfoques, características y propósitos, que más adelante describiremos. Por lo pronto, puede decirse, que actualmente existen muchos lenguajes de programación y continuamente están apareciendo otros nuevos, que prometen hacer mejor uso de los recursos computacionales y facilitar el trabajo de los programadores.

1.3. Paradigmas de programación

La programación orientada a objetos (POO) se suele conocer como un nuevo paradigma de programación. Otros paradigmas conocidos son: el paradigma de la programación imperativa (con lenguaje tales como Pascal o C), el paradigma de la programación Lógica (PROLOG) y el paradigma de la programación funcional (Lisp). El significado de paradigma (paradigma en latín; paradigma en griego) en su origen

significaba un ejemplo ilustrativo, en particular enunciado modelo que mostraba todas las inflexiones de una palabra. En el libro TheStructure of ScientificRevolutións, el historiador Thomas Kuhn describía un paradigma como un conjunto de teorías, estándar y métodos que juntos representan un medio de organización del conocimiento: es decir, un medio de visualizar el mundo. En este sentido, la programación orientada a objetos es un nuevo paradigma.

La orientación a objetos fuerza a reconsiderar nuestro pensamiento sobre la computación, sobre lo que significa realizar computación y sobre cómo se estructura la información dentro del computador.

No existe ningún estilo de programación idóneo para todas las clases de programación. La orientación a objetos se acopla a la simulación de situaciones del mundo real.

En POO, las entidades centrales son los objetos, que son tipos de datos que encapsulan con el mismo nombre estructuras o datos y las operaciones o algoritmos que manipulan esos datos.

1.4. Algoritmos

Un algoritmoes un conjunto finito de pasos definidos, estructurados en el tiempo y formulados con base en un conjunto finito de reglas no ambiguas, que proveen un procedimiento para dar la solución o indicar la falta de ésta a un problema en un tiempo determinado.

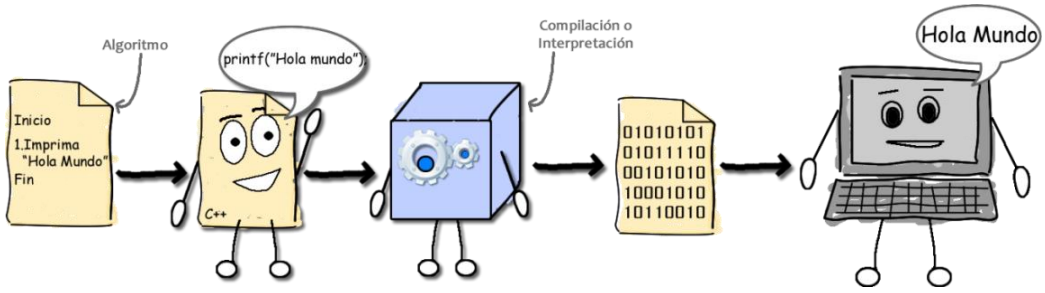


Figura 3

De la definición proporcionada y los cuestionamientos previos a ésta, podemos ver que un *algoritmo*, para ser catalogado como tal, debe exhibir ciertas propiedades:

- **Ser definido**.- Sin ambigüedad, cada paso del algoritmo debe indicar la acción a realizar sin criterios de interpretación.
- **Ser finito**.- Un número específico y numerable de pasos debe componer al algoritmo, el cual deberá finalizar al completarlos.
- **Tener cero o más entradas**.- Los datos son proporcionados a un algoritmo como insumo (o estos son generados de alguna forma) para llevar a cabo las operaciones que comprende.
- **Tener una o más salidas**.- Debe siempre devolver un resultado; de nada sirve un algoritmo que hace algo y nunca sabemos que fue. El devolver un resultado no debe ser considerado como únicamente "verlos" en forma impresa o en pantalla, como ocurre con las computadoras. Existen muchos otros mecanismos susceptibles de programación que no cuentan con una salida de resultados de esta forma. Por *salida de resultados* debe entenderse todo medio o canal por el cual es posible apreciar los efectos de las acciones del algoritmo.
- **Efectividad**.- El tiempo y esfuerzo por cada paso realizado debe ser preciso, no usando nada más ni nada menos que aquello que se requiera para su ejecución y en su ejecución.

Existen ***tres fases*** en la elaboración de un algoritmo para resolver un problema computacional:

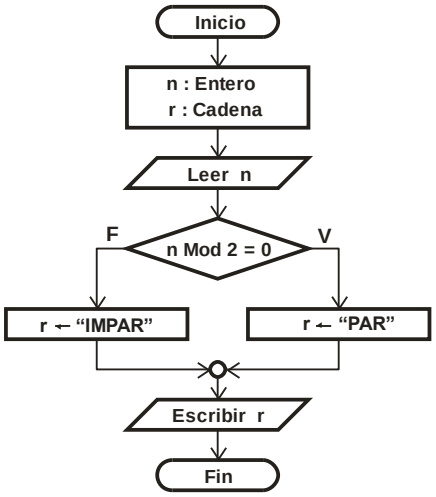
1) **Análisis.**- En esta se determina cuál es exactamente el problema a resolver. Que datos forman la entrada del algoritmo y cuáles deberán obtenerse como salida.
2) **Diseño**.- Elaboración del algoritmo.

3) **Prueba**.- Comprobación del resultado. Se observa si el algoritmo obtiene la salida esperada para todas las entradas.

Para expresar la solución de un problema se pueden usar diferentes herramientas de programación tales como:

- **Diagrama de flujo** (Flow Chart). - Es una representación gráfica que utiliza símbolos normalizados por ANSI, y expresa las sucesivas instrucciones que se deben realizar para resolver el problema.
- **Diagrama N-S** (Nassi-Schneiderman). - Conocido también como el diagrama de Chapin, es como un diagrama de flujo, pero sin flechas y con cajas continuas.
- **Pseudocódigo**. - Permite expresar las instrucciones en un lenguaje común (inglés, español, etc.) para facilitar la escritura como la lectura de la solución de un programa. No existen reglas para escribir pseudocódigos.

Por ejemplo, sea un número entero N, se desea saber si este número es par o impar. A continuación, utilizamos las tres herramientas descritas anteriormente, para resolver el problema.




```
Inicio
  //Variables de
  n : Entero
  r : Cadena
  //Entrada
  Leer n
  //Proceso
  Si n mod 2 = 0 Entonces
  r ← "PAR"
  SiNo
  r ← "IMPAR"
  Fin Si
  //Salida
```

Con diagrama N-S:

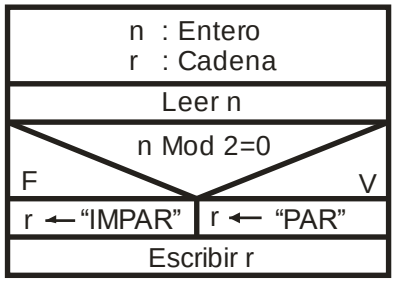


Figura 4

1.5 Introducción a la Programación Orientada a Objetos

Grady Booch, autor del método de diseño orientado a objetos, define la programación orientada a objetos (POO) como:

"*Un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representan un instancia de alguna clase, y cuyas clases son todas miembros de una jerarquía de clases unidas mediante relaciones de herencia".*

Existen tres importantes partes en la definición: La programación orientada a objetos:

1) Utiliza objetos, no algorítmicos, como bloques de construcción lógicos (jerarquía de objetos),

2) Cada objeto es una instancia de una clase.

3) Las clases se relacionan unas con otras por medio de relaciones de herencia.

Un programa puede parecer orientado a objetos, pero si cualquiera de estos elementos no existe, no es un programa orientado a objeto. Específicamente, la programación sin herencia es distinta de la programación orientada a objetos; se denomina programación con tipos abstractos de programación basado en objetos.

Capítulo II: JAVA y Javascript, Historia y evolución

2.1.- El Lenguaje de Programación Java y sus características

Java es un lenguaje de programación que se desarrolló para satisfacer las nuevas necesidades que requería la creación de aplicaciones a finales de los 90. Java es un lenguaje totalmente orientado a objetos. Fue desarrollado por SUN como respuesta a los problemas de programación de C++. Describiremos que es un lenguaje Java, una abreve historia y sus características más significativas.

Java es un lenguaje de programación 100% Orientado a Objetos, cuyas aplicaciones se pueden ejecutar en cualquier plataforma, es decir, sobre cualquier sistema operativo como Windows, Solaris, Linux, etc.

Pueden mencionarse muchas características de Java, sin embargo, para el presente mencionaremos sólo las más significativas para la implementación del mismo: Orientado a Objetos, Robusto, Independiente de plataforma, Multitarea.

Java soporta las tres características propias del paradigma de la programación orientada a objetos: encapsulación, herencia y polimorfismo. Java hace uso de la definición de

entidades formadas por métodos y variables que reciben el nombre de clases, la instancia de alguna clase cualquiera en Java recibe el nombre de objeto.

Las principales características de Java son:

Robustez: Java realiza verificaciones en busca de plataformas tanto en tiempo de compilación como entiempo de ejecución. La comprobación de tipos en java ayuda a detectar errores, antes posibles, en el ciclo de desarrollo. Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria.

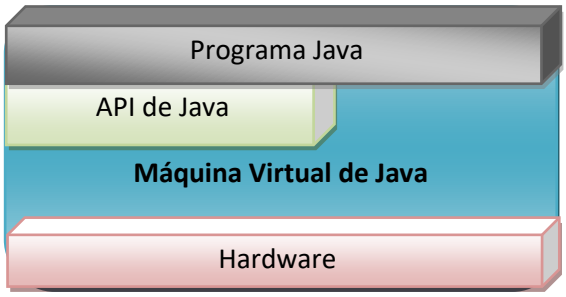
Además, para asegurar el funcionamiento de la aplicación, realiza una verificación de los byte-codes. Que son el resultado de la compilación de un programa Java. Es un código de máquina virtual que es interpretado por el intérprete Java. Es el código maquina directamente entendible por el Hardware, pero ya ha pasado todas las fases del compilador: análisis de instrucciones, orden de operadores, etc., y ya tiene generada la pila de ejecución de órdenes.

Independiente de plataforma: Mientras que en lenguajes de programación como C++ existe la necesidad de recompilar el código fuente cada vez que se cambia de plataforma, Java ofrece la posibilidad de que los archivos que son generados para una aplicación sean independientes de plataforma, es decir, que se compilen una vez y se ejecuten en cualquier plataforma. Esto es posible gracias a que las aplicaciones hechas en Java generan archivos conocidos como bytecodeestos archivos no corresponden a algún procesador o sistema operativo en particular, digamos Intel o Motorola, Unix o Windows, sino que al momento de ser ejecutados un intérprete propio de cada plataforma interpreta el bytecode al correspondiente sistema y procesador en el cual se

está ejecutando. Cada plataforma (Macintosh, Windows, Linux, etc) tiene su propio interprete de Java, pero el archivo bytecode es el mismo para todas las plataformas.

Plataforma de ejecución: Las aplicaciones Java (.java) para poder ejecutarlas deben ser compiladas, dando como resultado un código intermedio denominado bytecode (class). En el siguiente grafico se muestra la plataforma de ejecución de java: Los bytecode son ejecutados por la máquina virtual de Java (JVM), la JVM interpreta los bytecodes y luego los traduce al lenguaje maquina según la plataforma donde se ejecute.

La gran ventaja de máquina virtual de Java es aportar portabilidad al lenguaje de marca que desde SUN se ha creado diferentes máquinas virtuales java para diferentes arquitecturas y así un programa class escrito en Windows puede ser interpretado en un entorno Linux. Tan solo es necesario disponer de dicha máquina virtual para dichos entornos. En la figura siguiente se muestra el esquema de ejecución de Java.



Multitarea

A pesar de que las capacidades de multitarea que pueden ser implementadas en Java dependen en gran parte del sistema operativo en el cual se ejecuten, digamos Windows o Unix, dichas capacidades superan en gran manera a los entornos de flujo

único (single-thread) que ofrecen otros lenguajes de programación. Al ser Multitarea Java permite la ejecución concurrente de varios procesos ligeros o hilos de ejecución.

2.2.- JavaScript y sus orígenes

A los comienzos de Internet, las páginas Web estaban compuestas únicamente de texto y de vínculos hipertextos, limitando así su utilidad a un campo científico y universitario. Por otro lado, las limitaciones técnicas de la época, la velocidad de conexión, no permitían proponer algo mejor. Es así que a mediados de 1990 cuando la necesidad de disponer de páginas Web más amigables y con más servicios, en 1995, BrendanEich, por aquel entonces ingeniero informático de Netscape, tuvo el encargo de desarrollar un nuevo navegador Web, aprovechó la oportunidad para desarrollar un lenguaje de Script, originalmente denominado LiveScript, que debería ser en teoría un complemento del Java (estos dos lenguajes suelen confundirse debido a su denominación casi idéntica a pesar de no tener casi nada en común).Este lenguaje de programación tenía como objetivo desarrollar páginas Web más atractivas y amigables para el usuario, sin tener que usar para él una programación en el lado del servidor.

El cambio de nombre coincidió aproximadamente con el momento en que Netscape agregó soporte para la tecnología Java en su navegador Web Netscape Navigator en la versión 2.0 en diciembre de 1995. La denominación produjo confusión, dando la impresión de que el lenguaje es una prolongación de Java, y se ha

caracterizado por muchos como una estrategia de mercadotecnia de Netscape para obtener prestigio e innovar en lo que eran los nuevos lenguajes de programación Web.

Comienza entonces un periodo de intensa producción del lenguaje de script.Microsoft no tuvo otra salida que desarrollar su propia versión; nació entonces en 1996 Javascript, integrado en su navegador Internet Explorer y cuya última versión actualmente es Jscript.Net. Sin embargo a pesar de que JavaScript sea un lenguaje respetuoso de las instrucciones dadas por el ECMA (EuropeanComputerManufacturersAssociation), organismo encargado internacional de la estandarización de sistemas de información y comunicación. Loseditores de programas (Microsoft de un lado con Internet Explorer y Sun del otro con Firefox) desarrollaron navegadores que interpretaban de manera diferente Java Script; en consecuencia, algunos scripts, podían ejecutarse de manera correcta en el navegador y en otros generarían un error.

A finales de los noventa debido a estos acontecimientos surgieron otros lenguajes como ASP y PHP que se hicieron muy populares, pero fue sobre todo el uso abusivo de los pop-up la razón del desinterés creciente por JavaScript y la desesperación de los usurarios termino por desprestigiar sus ventajas entre los desarrolladores Web; llegando incluso a considerarlo como un sublenguaje.

Posteriormente la llegada de los bloqueadores de pop-up integrados en los navegadores le permitió a JavaScript recuperar el prestigio perdido.Hoy en día, JavaScript ha recuperado su popularidad, efectivamente el surgimiento de nuevas tecnologías Web sobre todo el llamado Web 2.0 devuelve a la programación un lugar primordial.

2.3. Versiones

La primera versión de JavaScript fue exitosa y Netscape Navigator 3.0 ya incorporaba la siguiente versión 1.1 del lenguaje. Al mismo tiempo Microsoft lanzó JScript con su navegador Internet Explorer 3, JScript era una copia de JavaScript al que le cambiaron el nombre para evitar problemas legales por la marca.

Para evitar una guerra de tecnologías,Netscape decidió estandarizar el lenguaje JavaScript, de esta forma en 1997 se envió la especificación de JavaScript1.1 al organismo ECMA-262, en el que se definió por primera vez el lenguaje ECMAScript.

Resumen de la evolución de las versiones de JavaScript

1995. Primera versión **1.1**de JavaScript, todavía con nombres provisionales como Mocha yLiveScript.

1997. Definición de primer estándar JavaScript, a cargo de ECMA Internacional que fue denominado ECMA-262 firstedition también denominado JavaScript **1.2.**

1998. Aparición del segundo estándar JavaScript denominado ECMA-262 secondedition también denominado JavaScript **1.3.**

2000. Aparición de la especificación del estándar JavaScript ECMA-262 thirdedition también denominado JavaScript **1.5.**

2010. Aparición de la especificación del estándar JavaScript ECMA-262 fifthedition también denominado JavaScript **1.8.5.**

2.4.- Diferencias entre Java y JavaScript

Java

Es un lenguaje de programación (como el Pascal, el BASIC o el C y C++) que fue desarrollado por la empresa SUN principalmente para crear aplicaciones para

Internet. El lenguaje Java es completo, es decir nos permite realizar cualquier operación sobre el Computador Java es un lenguaje de programación orientada a objetos, próximo a la forma de pensar humana; es un lenguaje que es compilado por la máquina virtual Java; es un lenguaje multiplataforma funciona en diferentes sistemas operativos que tenga instalada la máquina virtual Java.

JavaScript

JavaScript, es un lenguaje de programación que se utilizapara crear páginas Web dinámicas, que es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensaje al usuario.

Técnicamente JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlo; es decir se pueden probar en cualquier navegador sin necesidad de procesos intermedios.

JavaScript es un lenguaje que se incorpora dentro de la página Web, formando parte del código HTML.

Capítulo III: Características Básicasde JavaScript

3.1.- Inclusión de JavaScript en la página

Para hacer que un documento HTML incluya instrucciones en JavaScript se debehacer uso de la etiqueta **<SCRIPT>**de esta forma:

<script type=text/JavaScript>
código JavaScript
</script>

Si se quiere especificar qué versión de JavaScript se utiliza, se evitará quelos navegadores que no soportan la versión decodifiquen el JavaScript, se utilice por ejemplo:

<scriptlanguage=”JavaScript1.3”>

3.2.- Uso de un archivo externo

También podemos utilizar el código JavaScript escrito en un archivo separado. Este archivo debe tener la extensión **js.** En el archivo se coloca sólo código en JavaScript. Después ese código se puede invocar desde la página Web con el código:

<script language="Javascript" src="archivo.js">

3.3.-Normas de escritura en Javascript

a. Los comentarios deben empezar con el símbolo

// Este es un comentario de una sola línea.

/* este es un comentario con espacio más largo, puede tener varias líneas. */

Las líneas de código terminan con el signo de punto y coma **(;)**

b. JavaScript distingue entre mayúsculas y minúsculas

c. Las llaves ({y}) permiten agrupar código.

3.4.- Tipos de datos

Los tipos de datos son los distintos valores que puede gestionar un Computador y que un usuario/programador puede manipular mediante lenguaje de programación.

JavaScript reconoce seis tipos de valores diferentes:

Tabla I

Números	Enteros o coma flotante
Booleanos	True o False
Cadenas	Los tipos de datos cadenadeben ir delimitados por comillas simples o dobles
Objetos	Obj =new Object();
Nulos	Null
Indefinidos	Un valor indefinido es el que corresponde a una variable que ha sido creada pero que no le ha sido asignado un valor.

3.5.- Variables

Una variable es un elemento que tiene un determinado nombre y que permite almacenarvalores. En otros lenguajes de programación existen diferentes tipos de variables en función del tipo de datos que pueden guardar: variables para cadenas, para números enteros, para números reales, etc. JavaScript es muy permisivo en este aspecto de manera que una variable puede guardar cualquier tipo de dato y además pueden crearse en cualquier parte del programa.

3.5.1.-Nombre de las variables

Deben empezar con una letra la cual puede ir seguida de números, el signo "_" o más letras. También pueden asignarse valores a cada variable; estosvalores pueden ser:

a. **Cadenas de texto**: "Esto es una prueba", 'prueba' o "esto es una 'prueba' de código". Siempre se encierran entre comillas dobles o simples. Una variable de texto que no tiene contenido, se dice que tiene valor null. La palabra nulles un término reconocido por JavaScript.
b. **Valores numéricos**: 1, -100, 1.6, 2.0E2.
c. **Valores booleanos:** true o false.

3.5.2Caracteres especiales

Los valores de tipo texto van entre comillas y dentro de ellos se pueden colocar caracteres especiales (caracteres que no se pueden ver, como el cambio de línea) los cuales tenemos:

a: Alarma

b: Retroceso (cursor una posición hacia atrás).

f: Nueva página

n: Nueva línea

r: Retorno de carro

t: Tabulador

\\: Signo "\"

3.5.3.-Declaración de una variable

Para declarar una variables no tenemos más que poner la palabra “var” y a continuación la lista de variables separados por comas.No todos los nombres de variables son válidos hay algunas restricciones:

a. Un nombre valido de variable no puede tener espacios.

b. Puede estar formada por números, letras y el carácter subrayado

c. No se puede usar palabras reservadas (if, for,while.)

d. No puede empezar por un número, es decir, el primer carácter del nombre de la variable ha de ser una letra “b”.

Para declarar una variable se puede emplear:

```
varvariable= valor;

variable  =  valor;
```

De tal modo, que realmente en JavaScript no hace falta declarar una variable antes de su uso. Ejemplos:

```
vartestear    = 0;
testeaTexto  = “Mi casa”;
SeleccionarColor = true;
```

JavaScript permite que una variable pueda almacenar distintos tipos de datos en cada parte del código. Es decir, una variable que ahora almacena texto, después puede almacenar números.

Tras declarar la variable, su valor puede cambiar mediante la asignación de un valor:

Testear = 12.3;

O mediante la asignación del resultado de una operación:

Testear = 12 * 3 + varX;

O simplemente:

Variable= valor;

3.6.- Operadores

Los operadores son los elementos que permiten realizar operaciones con los datos del código. En JavaScript existen los siguientes tipos de operadores:

3.6.1.- Operadores aritméticos

Tabla II

Operador	**Significado**
+	Suma
-	Resta
*	Multiplicación
/	Dividir
%	Resto de la división
++	Incremento
--	Decremento

Los operadores aritméticos son binarios (necesitan dos operarios) y realizan sobre sus operando alguna de las operaciones aritméticas conocidas. En concreto tenemos:

Suma +:

Se trata de un operador usado para sumar dos valores numéricos o para concatenar cadenas entre sí o números y cadenas.

```
var var1 = 10, var2= "Buenos", var3 = " días", var4 = 31;
document.write(var1+var4)   /* resultado 41 */
document.write(var2+var3)   /* resultado: Buenos días */
document.write(var1+var3)   /* resultando: 10 días */
```

Resta - :

Operador usado para restar valores numéricos. Puede actuar sobre un único operando numérico cambiándole de signo.

```
var num1 = 10, num2 = 8, res = 0;
res = num1 - num2;              /*res contiene 2 */
res = -res                                  /* ahora res contiene -2*/
```

Producto (*) y cociente (/):

Realizan las operaciones aritméticas de multiplicar y dividir dos valores:

```
var op1 = 50, op2= 4, div, mul;
div = op1/op2  /*div contiene 12.5 */
mul = op1 * op2         /*mul contendrá 200 */
```

Resto %:

También llamado operador módulo calcula el resto de una división.

```
var op1 = 50, op2= 4, resto;
resto = op1 % op2;          /*resto contiene 2 */
```

Incre

Estos .

Si el operador se antepone a la variable la operación de incremento o decremento esprioritario sobre cualquier otra.

```
var op1=5, op2 = 5, res;
res = ++op1;          /*res adquiere el valor 6 y luego op1 el 5*/
res = op2--;     /*res adquiere el valor 5 y luego op2 el 6*/
```

3.6.2.- Operadores lógicos

Los operadores lógicos sirven para componer condiciones más simples por medio de las reglas de la **y, o** y **no** lógicas. Ellos nos permiten expresar condiciones compuestas de las que queremos averiguar su valor de verdad.

Tabla III

Operador	**Significado**
&&	AND (‘y’ lógica)
‖	OR (‘o’ lógica)
!	NOT (‘no’ lógica)

Noiguales!=

Noidénticos!==

Invierten el sentido de las comparaciones iguales == e idénticos === respectivamente.

Andlógico&&

Este operador se utiliza para concatenar comparaciones, es decir, para comprobar varias condiciones. El resultado sólo será **true** si todas las comparaciones lo son verdaderas.

```
var op1 = 2, op2 = 50, op3 = 25, comp;
comp = (op1 > op2) && (op1 < op3);          /* comp adquiere el valor  false */
```

comp es false porque op1 no es mayor que op2 aunque sea mayor que op3

Or lógico II

Como el anterior, sirve para realizar comparaciones compuestas y sólo devolverá false cuando todas las comparaciones los sean. Es decir basta que una comparación sea true para que devuelva el valor true.

```
var op1 = 2, op2 = 50, op3 = 25, comp;
comp = (op1 > op2) && (op1 < op3);          /*comp adquiere el valor true */
```

comp es true porque op1 es menor que op3, (op1 < op3 es por tanto true)

3.6.3.- Operadores de comparación

Los de comparación son binarios y su resultado es un booleano (un valor de verdad) Nos permiten expresar si una relación de igualdad/desigualdad es cierta o falsa dados los operadores.Entre ellos tenemos:

Operador	Significado
= =	Igual
!=	Distinto
>	Mayor que

Tabla IV

<	Menor que
>=	Mayor e igual que
<=	Menor e igual que

Mayor que >Compara dos valores y devuelve true si el primero es mayor que el segundo. Compara tanto números como cadenas.

```
var hoy = 4; ayer = 10, comp;
comp = hoy > ayer                    /* comp adquiere el valor false*/
```

Menor que <

Compara dos valores y devuelve true si el primero es mayor que el segundo. Compara tanto números como cadenas.

```
var hoy = 4; ayer = 10, comp;
comp = hoy < ayer                    /* comp adquiere el valor true*/
```

Mayor o igual >=

Compara dos valores y devuelve true si el primero es mayor o es igual que el segundo. Compara tanto números como cadenas.

```
var hoy = 4; ayer = 4, comp;
comp = hoy >= ayer                   /* comp adquiere el valor true*/
```

M

C

Compara tanto números como cadenas.

```
var hoy = 4; ayer = 4, comp;
comp = hoy <= ayer                   /* comp adquiere el valor true*/
```

Iguales ==

Compara dos valores y devuelve true si ambos son iguales. Compara tanto números como cadenas.

```
var hoy = 4; ayer = 4, comp;
comp = hoy == ayer                    /* comp adquiere el valor true*/
```

Idénticos ===

Compara dos valores y devuelve true si el primero es mayor o es igual que el segundo. Compara tanto números como cadenas.

```
var hoy = 4; ayer = '4', comp;
comp = hoy == ayer;          /* comp adquiere el valor true*/
comp = hoy === ayer          /* comp adquiere el valor false*/
```

3.6.4.- Operadores varios

En este punto completamos los operadores de que dispone JavaScript.Entre ellos tenemos:

New

Es un operador que sirve para crear nuevas instancias de un objeto vía su constructor.

Delete:

Es un operador

```
var hoy = new Date("12 /30/2018")

var lista = new Array(1,4,7,9,10);
delete(lista,0);
```

Typeof:

Es un operador que nos devuelve una cadena que describe el tipo de dato que corresponde con el objeto (variable, función, etc) que se escribe a continuación los tipos devueltos son **number, string, boolean, object, function y undefined**.

```
hoy = 1.2345;
tipo = typeof(hoy);
```

La variable tipo contendrá **number**.

3.7.- Literales

Son valores que se utilizan para inicializar una variable, es decir para dar los valores que inicialmente se desean que tengan las variables. Su uso es frecuente junto con la declaración de variables. Por ejemplo

```
var Numero =0;      varLogic =True;          var Texto = “”;
```

3.8.- Comentarios

Es recomendable documentar el código JavaScript con comentarios, es hacer un programa más fácil de entender, será útil para una posterior modificación. Si nos acostumbramos a poner comentarios referentes a que hace el programa y como lo hace, no se perderá tiempo en posteriores revisiones.

JavaScript permite dos formas de comentarios:

a) Comentarios de una línea, precedidos por //. Ejemplo:

   //Esto es un comentario de una línea

b) Comentarios demás de una línea, cerrados por la pareja /*/.Ejemplo:

   /*Este comentario tienemás de una línea*/

3.9.- Mensajes

Se trata de ventanas que desde el código se lanzan al usuario para hacer que este reaccione ante una situación o nos informe ante una duda. Realmente todos los mensajes se obtienen a través del objeto Windows.

3.9.1 Alert

Es el mensaje más usado. Saca un mensaje por la pantalla el cual solo deja la posibilidad de aceptarle. Su uso es mostrar información al usuario pero resaltándola de la página. Su sintaxis es:

alert (texto_del_mensaje**);**

3.9.2Prompt

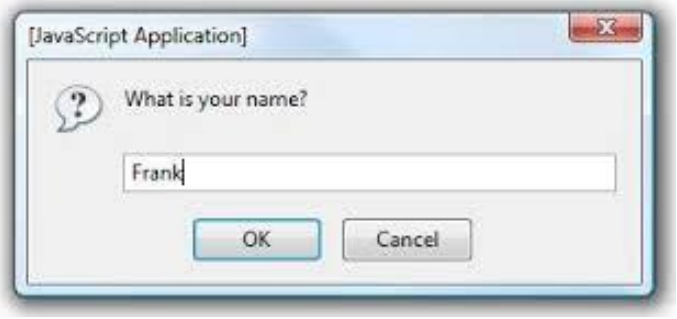
En este caso se trata de una ventana que pide entrar datos al usuario. De modo que esta función devuelve un valor que se puede usar en el código si es asignado a una variable. Su sintaxis es:

prompt(texto_del_mensaje,valor_por_defecto**);**

El segundo parámetro (valor por defecto) no es obligatorio incluirle y permite asignar un valor al cuadro de texto en el que el usuario tendrá que introducir información. Ejemplo de uso de **prompt**:

respuesta=prompt("¿Qué quieres hacer?","comer");

En el ejemplo, el resultado de lo que el usuario responde se almacena en la variable *resultado* y al principio la ventana contendrá el valor ***comer*** en el cuadro de texto destinado al usuario naturalmente, el usuario podrá variar este valor si lo desea.



3.9.3 Confirm

Saca un mensaje de confirmación el cual suele tener dos botones: Aceptar y Cancelar. Sintaxis:

confirm(texto_del_mensaje**)**

La ventana mostrará el texto elegido (normalmente es una pregunta) y el usuario elegirá si desea aceptar o no el contenido. **Confirm**devuelve un valor **true** en el caso de que el usuario acepte el mensaje, y **false** si no lo hace.

Capítulo IV: Programación Javascript: Estructuras y funciones

4.1.- Estructuras de control:

Las estructuras de control son la responsables directas del control de flujo del código. Estas estructuras de control se pueden clasificar en dos grandes grupos: Estructuras Condicionales y Estructuras Repetitivas o también conocidas como bucles.

4.1.1Estructuras Condicionales:

Una estructura condicional es un tipo de estructura que realiza una tarea u otra dependiendo del resultado de evaluar una condición. Entre ellas tenemos:

Condicional if…else

La instrucción **if**realiza lo que se denomina un sí lógico. Su forma es:

```
if (condición) {
..código que se ejecuta si la condición es cierta
}
```

También admite esta otra:

Se **If (condición) {**
..código que se ejecuta si la condición es cierta
} else {
de **...código que se ejecuta si la condición es falsa**
}
admite dentro una

instrucción **if**, colocar otra instrucción **if**. A esto se le llama“anidar” condiciones **if**.

Condicional switch:

Este condicional aparece a partir del JS 1.2.Su forma es:

```
Switch (condición) {
        case caso1 :
                sentencias para caso1;
                break;
        casecasoN :
                sentencias para casoN;
                break;
        default :
        sentencias por defecto;
                break;
        }
```

4.1.2 Estructuras repetitivas o bucles

Entre ellos tenemos:

Bucle while:

Un bucle es una estructura de programación que permite repetir sentencias hasta que se cumpla una determinada condición. Su forma es:

```
while (condicion) {
... sentencias que se ejecutan mientras la condición se
cumpla
}
```

Ejemplo

Bucle for:

```
var x=1;
while(x<11)
{
document.write(x," ");
x++;
}
// Sale 1 2 3 4 5 6 7 8 9 10
```

Su efecto

es muy similar a la anterior estructura. Permite ejecutar una serie de sentencias hasta que se cumpla una determinada condición. Su estructura es:

```
for(valor_inicial; condición; actualización)
{
..sentencias que se ejecutan mientras la condición se cumpla
}
```

Ejemplo:

```
for(x=1;x<11;x++)
{
document.write(x," ");
}
// Sale 1 2 3 4 5 6 7 8 9 10
```

Bucle Do……while

Sintaxis:

```
Do{
    instrucciones a repetir
}while (condición);
```

Bucle for …in

Sintaxis:

```
For (var 'Propiedad' in 'objeto') {
    Instrucciones a repetir
    }
```

ConstrucciónWith.

Su sintaxis es

```
with(objeto) {
 Varias sentencias
 }
```

4.2.- Funciones

Lasfunciones son una serie de instrucciones que realizan una determinada tarea. A lasfunciones se las pone un nombre que luego puede ser utilizado en el código de lapágina.

4.2.1.- Concepto de una Función

Antes de poder usar una función en el código de la página, se la debe definir; es decir,se debe indicar qué operaciones son las que debe hacer la función. La definición de la función es:

```
functionnombredelafunción(argumento1, argumento2,...)
{
instrucciones que debe realizar la función
}
```

El código que está encerrado entre llaves indica lo que realiza la función (por ejemplomostrar un mensaje de ayuda), cada vez que desde el código se llame a la función, éstarealizará sus instrucciones.Por otro lado los argumentos son variables que algunas funciones necesitan pararealizar su tarea.

4.2.2 Sintaxis de una Función

Para usar (invocar) una función en el script, basta con poner su nombre seguido delos

```
function error() {
document.write("<b>Ocurrió un error</B><BR>");
}
```

paréntesis.Ejemplo:

4.2.3 Funciones propias del lenguaje

Las funciones que se van a describir son funciones que no están ligadas a los objetos del navegador, sino que forman parte del lenguaje. Estasfunciones nos van a permitir convertir cadenas a enteros o a reales, evaluar una expresión. Las seis funciones que trataremos serán:

Función parseInt (cadena base)

Esta función devuelve un entero como resultado de convertir la cadena que se le pasa como primer argumento. Opcionalmente podemos indicarle la base en la que queremos que nos devuelva este entero. Si no especificamos base, nos devuelve el resultado en base diez.

Si la cadena no se pude convertir a un entero, la función devuelve el mayor entero que haya podido construir a partir de la cadena. Si el primer carácter de la cadena no es un número, la función devuelveNaN(Not a Number).

alert(parseInt(“110011”,2));//Sale 51

```
a = parseInt("1234");
```

En 'a' tenemos almacenado el valor 1234, y podremos hacer cosas como a++, obteniendo 1235 ;)

Produce el mismo efecto que hacer

```
a = parseInt("1234",10);
```

Puesto que, como hemos dicho, la base decimal es la base por defecto.

```
a = parseInt("a1234");
```

En 'a' tenemos almacenado el valor NaN.

Función parseFloat(cadena)

Esta función convierte el texto (que debe tener cifras numéricas) a formato de número con decimales. Si la cadena no se puede convertir a un real, la función devuelve el mayor real que haya podido construir a partir de ella.

a = parseFloat("1.234");

En 'a' tenemos almacenado el valor 1.234.

a = parseFloat("1.234e-1");

En 'a' tenemos almacenado el valor 0.1234.

a = parseFloat("1.2e-1er45");

En 'a' tenemos almacenado el valor 0.12.

Función escape (cadena)

Muestra el código ASCII de los símbolos del texto. Cada número en el resultado va precedido del símbolo **%** y el código ASCII sale en forma Hexadecimal.

escape('$');

devuelve %24

escape("a B c_1_%á");

devuelve a%20B%20c%20_1_%25%E

Función unescape (códigos)

Hace justo lo inverso del anterior. Devuelve los códigos que representan los códigos ASCII en forma de texto que se le pasa como parámetro.

unescape('%¿qué saldraaqui?');

obtenemos la cadena vacía ", si ponemos

unescape('%a¿quesaldraaqui?');

obtenemos ' que saldraaqui?', si ponemos

unescape('%a¿ques%aldraaqui?');

FuncionisNaN(Expresion)

Devuelve true si la expresión tiene un contenido no numérico, en caso contrario (i.e., el argumento pasado es numérico) devuelve ‘false’. Ejemplo:

isnNaN (“Hola Mundo”);
devuelve ‘true’,
isnNaN (“091”);
devuelve ‘false’,

Funcioneval(expesion)

Esta función devuelve el resultado de evaluar la expresión pasada como parámetro.Ejemplo:

eval (“2+3”)
Nos devuelve 5,
eval (2+3)
También nos devuelve 5,

4.3.- Objetos y Eventos de JavaScript

Son una de las bases fundamentales de JavaScript. Un objeto es unaagrupación de variables, que en ese caso se llaman propiedades, y de funciones, las cuales se llaman métodos. Las propiedades definen las características de los objetos y los métodos las operaciones que podemos hacer con él. JavaScript posee muchos objetos predefinidos y también permite crear nuestros propios objetos.

Métodos

Los métodos son funciones asociadas a los objetos. Su uso es:

objeto.metodo()

Donde el método además puede poseer parámetros.

Tipos de Objetos del lenguaje

Objeto del lenguaje: <u>string:</u>

El objeto stringsirve para manejar cadenas de texto. Cada vez que creamos una variable de cadena, en realidad estamos creando una variable de tipo string. Por lo tanto no será necesaria su declaración

Propiedades

a. **length**: devuelve la longitud de la cadena.

b. **prototype**: permite agregar métodos y propiedades al objeto

Métodos De String:

a. **anchor(*nombre*).** Crea un marcador en el texto.

b. **big().** Muestra la cadena de caracteres con una fuente grande.

c. **blink().** Muestra el texto de modo intermitente.

d. **bold().**Muestra el texto en negrita.

e. **charAt(*n*).** Muestra el carácter situado en la posición ***n*** de la cadena.

f. **fixed().** Muestra la cadena en fuente no proporcional.

g. **fontcolor(*color*).** Determina el color del texto.

h. **fontsize(*n*).** Muestra el texto en el tamaño ***n***, donde ***n*** es un número del 1 al 7

i. **indexOf(*cadenaInterna,inicio)***. Devuelve la posición de la cadena interna en el texto, teniendo en cuenta que el primer carácter es el número 0. El primer parámetro es el texto que se busca; el segundo es opcional e indica desde qué posición del texto comenzamos a buscar. Si no se encuentra la cadena interna, se devuelve el valor.

j. **italics().** Muestra el texto en cursiva.

k. **lastIndexOf(*cadenaInterna,inicio*).** Idéntico a **indexOf**sólo que en este caso cuenta la última vez que aparece la cadena (en lugar de la primera vez como hace **indexOf**).

l. **link(*URL*).** Crea un hipervínculo en la cadena de texto, el parámetro ***URL***indica el destino del vínculo.

m. **small().** El texto se muestra con fuente pequeña.

n. **strike().** Subraya el texto.

o. **sub().** El texto va en subíndice.

p. **substring(*x,y).*** Muestra el fragmento de texto que va desde la posición **x** a la posición **y**.

q. **sup().**Superindice

r. **toLowerCase().** Convierte la cadena a minúsculas.

s. **toUpperCase().**Convierte la cadena a mayúsculas.

Objeto del Lenguaje: ARRAY

Los arrays (matrices) son elementos indispensables en la programación de ordenadores. Un array es un conjunto de datos agrupados. Para acceder a cada elemento

individual delarray se usa un número de índice, el primer elemento tendrá el índice 0.

En el caso de los arrays de JavaScript, su uso es muy eficaz y más libre que en los lenguajes formales (como Pascal por ejemplo).

Para crear un array se hace:

nombrearray = **new Array()**

Esto crea un array de tamaño indeterminado.

Para rellenar los valores del array:
nombrearray[0] = valor;
nombrearray[1] = valor;
...
nombrearray[n] = valor;

Tras asignar valores el array se hace más grande. También se puede especificar sutamaño al crearle:

nombreArray= new Array(tamaño)

O incluso asignar valores en la propia creación del array.

Ejemplo:

Además un array puede tener distintos tipos de datos:

equipos= new Array(“Real Madrid”, “F. C. Barcelona”);

miComputador = new Array(“HP”, “Pentium III”, 64);

Y cada elemento de un array puede ser otro array:

```
elemento = new Array(8);
elemento[3] = new Array(5);
```

Propiedades

a) Length

Como su nombre indica esta propiedad nos devuelve la longitud del array, es decir, el número de elementos que puede almacenar. Su uso es muy simple:

```
var lista = new Array(50);
tamagno = lista.length; /*tamagno almacenaría el valor 50 */
```

b) Prototype

Esta es una propiedad muy potente en el sentido que nos permite agregar al objeto Array las propiedades y métodos que queramos.

```
Array.protoype.descriptor = null;
dias = new Array ('lunes', 'Martes', 'Miercoles', 'Jueves',
'Viernes');
dias.descriptor = "Dias laborables de la semana";
```

Métodos:

a) Concat(*array*).-Agrupa dos arrays. Disponible desde la versión 1.2. Ejemplo:

```
a = new Array(12, 3, 5);
b = new Array(“Hola”, “Adios”);
c = a.concat(b);
//c es el array (12, 3, 5, “Hola”, “Adios”)
```

b) **Join().**Saca una cadena de texto que contiene todos los elementos del array:

```
a = new Array("Rojo", "Azul", "Verde");
b = a.join();
//b contiene "Rojo,Azul,Verde"
```

c) **Reverse().**Invierte el orden del los elementos de un array. El primero pasa a ser el último y viceversa.

d) **Sort().**Obtiene la matriz de manera ordenada.

Objeto del lenguaje: MATH

Es el objeto que usa JavaScript para dotar al lenguaje de funciones matemática avanzadas y de constantes predefinidas, como el número PI.

Propiedades de math

a. **E.** Devuelve la constante de Euler o número **e. (**Base de los logaritmos naturales "neperiano")
b. **LN2.** Devuelve el logaritmo neperiano de 2.
c. **LN10.** Devuelve el logaritmo neperiano de 10.
d. **LOG2E.** Logaritmo en base 2 de **e**.
e. **LOG10E.** Logaritmo en base 10 de **e.**
f. **PI.** Devuelve el número PI.
g. **SQRT1_2.** Raíz cuadrada de 0,5.
h. **SQRT2.** Raíz cuadrada de 2.

Métodos del objeto math:

Tabla V

abs: Valor absoluto	**cos**: coseno	**pow**: Potencia de
acos: Arco coseno	**exp**: Exponencial	**random**: Número al azar
asin: Arco seno	**floor**: Redondeo inferior	**round**: Redondear
atan: Arco tangente	**log**: Logaritmo natural	**sin**: Seno
atan2: Arco tangente	**max**: máximo	**sqrt**: Raíz cuadrada
ceil: Redondeo superior	**min**: Mínimo	**Tan**: Tangente

Objeto del lenguaje: DATE

El objeto Date contiene un valor que representa fecha y hora de un instante dado. Para crear una instancia de este objeto usamos alguna de las siguientes sintaxis:

```
var fecha= new Date()
var fecha= new date(número)
var fecha= new date(cadena)
var fecha=
new date(año, mes, día[, hora[, minutos[, seg[,ms]]]])
```

forma

la varia en

fecha la fecha dada por el argumento como el número de milisegundos transcurridos desde la media noche del 1 de enero de 1970. El tercer tipo se usa cuando la fecha se pasa en forma de cadena. Por último la fecha puede crearse pasándole como argumento los números de año, mes, día, hora y opcionalmente, hora, minuto, segundo y milisegundo.

Los años posteriores a 1970 puede escribirse con dos dígitos, pero es aconsejable usar siempre cuatro dígitos por aquello de los efectos 2000.

```
var hoy = new date() /*fecha del día en hoy */
varevento = new Date("December 30 2018");
varotro = new Date("30 Dec2018");
varotro = new Date("10/12/2018"); //Oct, 12, 2018
varinstante = new Date(2018, 11, 10, 20,00);
```

Estas son tres posibles formas de declarar objetos de tipo fecha. Las dos últimas almacenan el mismo día, pero en la última además se guarda la hora.
Donde se usen cadenas para indicar una fecha podemos añadir al final las siglas GMT (o UTC) para indicar que la hora se refiere a hora del meridiano Greenwich, si no se toma como hora local, o sea, según la zona horaria configurada en el Computador donde se ejecute el script.

Métodos de los objetos date()

a. **getDate().** Devuelve el día del mes.
b. **getDay().** Devuelve el día de la semana en forma de número.
c. **getFullYear().**Devuelve el año, pero en forma de 4 números. Con esto se asegura la compatibilidad con el año 2000. Está función se añadió al JavaScript 1.3, por lo que ciertos navegadores no podrán usarla.
d. **getHours().** Devuelve la hora.
e. **getMinutes().** Devuelve los minutos.
f. **getMonth().** Devuelve el mes (con números del 0 al 11).
g. **getSeconds().** Devuelve los segundos.
h. **getTime().** Devuelve el número de milisegundos de la fecha, empezando a contar desde 1970.
i. **getTimezoneOffset().** Devuelve la diferencia en minutos entre la zona horariaactual y la hora solar (GMT).
j. **getYear().** Devuelve el año.
k. **setDate(*valor*).** Establece el día del mes.
l. **setFullYear(*valor*).** Establece el año (con cuatro cifras).
m. **setHours(*valor*).** Establece la hora.
n. **setMinutes(*valor*).** Establece los minutos.
o. **setMonth(*valor*).** Establece el mes (con un número del 1 al 11).
p. **setSeconds(*valor*).** Establece los segundos.
q. **setTime(*valor*).** Establece la fecha con el número de milisegundos desde el 1 de Enero de 1970
r. **setYear(*valor*).** Establece el año.
s. **toLocaleString().** Devuelve la fecha en formato

Objeto del lenguaje: Number

Es el objeto destinado al manejo de datos y constantes numéricas. Realmente no es habitual crear objetos de este tipo por cuanto JavaScript los crea automáticamente cuando es necesario. No obstante la sintaxis para su creación es la habitual para cualquier objeto:

minúmero = new Number(valorinicial)

El valor inicial es optativo, si no se usa el objeto se crea con valor null

Métodos

Los heredados del objeto **Object**

Propiedades

Además de las heredadas del objeto **Object**Number posee las siguientes propiedades:

a. **MAX_VALUE**

Indica el valor máximo utilizable por JavaScript, actualmente 1.79E+308.

b. MIN_VALUE

Indica el valor mínimo utilizable por JavaScript, actualmente 2.22E-308.

c. NaN

Una constante usada para indicar que una expresión ha devuelto un valor no numérico. NaN no puede compararse usando los operadores lógicos habituales, para ver si un valor es iguala NaN se debe usar la función incorporada **isNaN**

d. NEGATIVE_INFINITY

Una constante para indicar infinito positivo, es decir, un valor superior al MAX_VALUE

e. POSITIVE_INFINITY

Una constante para indicar infinito negativo, es decir, un valor superior al MAX_VALUE con signo negativo.

Aplicación didáctica

Sesión de clase

I.-Datos informativos:

CURSO : Computación

TEMA : Lenguaje de programación Javascript

GRADO Y SECCION: 5TO "B"

DOCENTE : Roque Román Arenaza

DURACION : 60 Minutos

Capacidad fundamental	Pensamiento critico
Aprendizajes esperados	• Define el lenguaje de programación JavaScript • Realiza aplicaciones
Actitud	• La responsabilidad en la aplicación de las normas de seguridad en las practicas

II.- Tema transversal: Educación para la cultura productiva y emprendedora.

III.- Estrategia metodológica

Momentos	Actividad	Método/Técnica	Tiempo
Inicio	-Despertar el interés hablando de la historia del lenguaje de programación.	expositivo/ narrativo	10 minutos
Proceso	-Se procede a definir las características principales del lenguaje de programación, así como las estructuras del lenguaje java. -Se demuestra mediante una aplicación interactiva del lenguaje JavaScript.	Inductivo/ deductivo Explicativo/ demostrativo	30 minutos
Salida	-Ahora los alumnos hacen una aplicación como practica calificada.	Evaluación/practica demostrativa	20 minutos

IV.- Evaluación

Capacidad de área: Ejecución de procesos

Indicadores	**Técnicas**	**Instrumentos**
Define el lenguaje de programación JavaScript. Realiza una aplicación práctica en el lenguaje java	Evaluación en la ejecución	Practica calificada
Actitudes y valores	**Responsabilidad**	**Instrumentos**
Aplica permanentemente las normas de seguridad al realizar las prácticas. Muestra empeño en la ejecución de las actividades de aprendizajes	Evaluación de actitudes	Lista de cotejo

V.- Recursos

- Computadora
- Instalador de programa java creator.
- USB

VI.- Referencia:

Deitel, Harvey. Deitel Paul (2004): *Como programar en JavaScrpt.* Quinta Edición, Pearson educación

Joyanes Aguilar Luis (1998), *Programación Orientada a Objetos*, 1ra. Edición, Mc Graw Hill.

Guía de laboratorio

Aplicaciones en Javascript

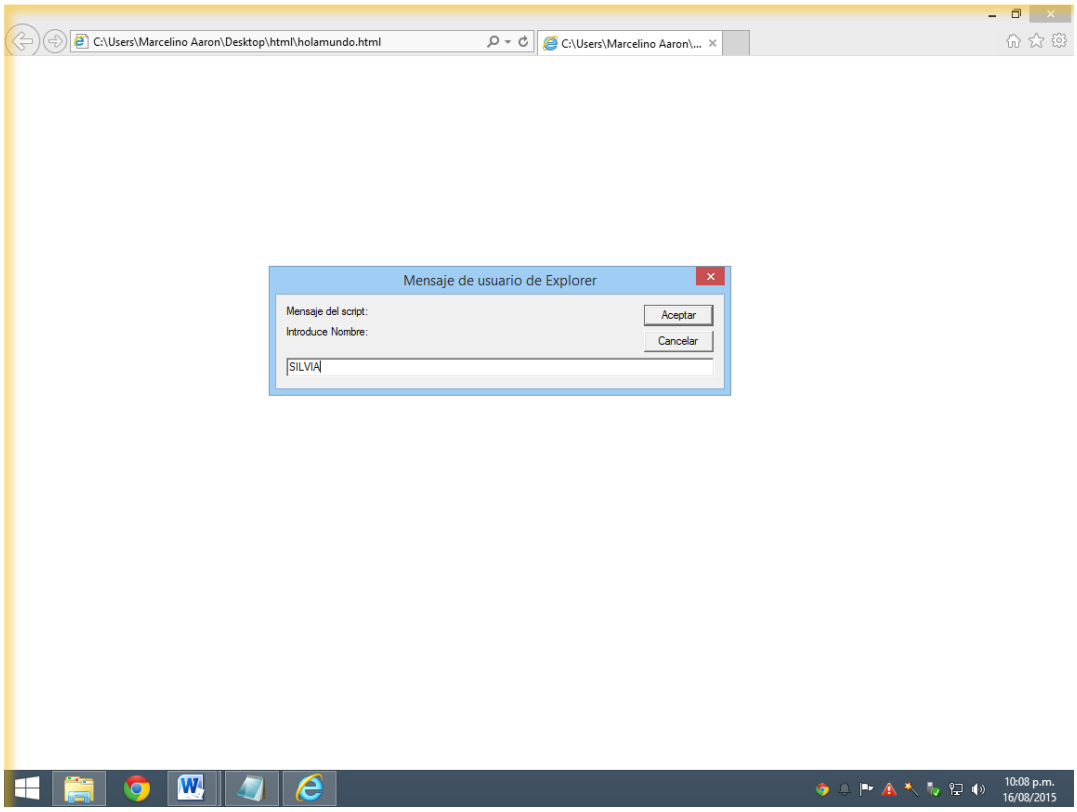
Paso 1: Buscamos el navegador donde se pueda ejecutar el programa

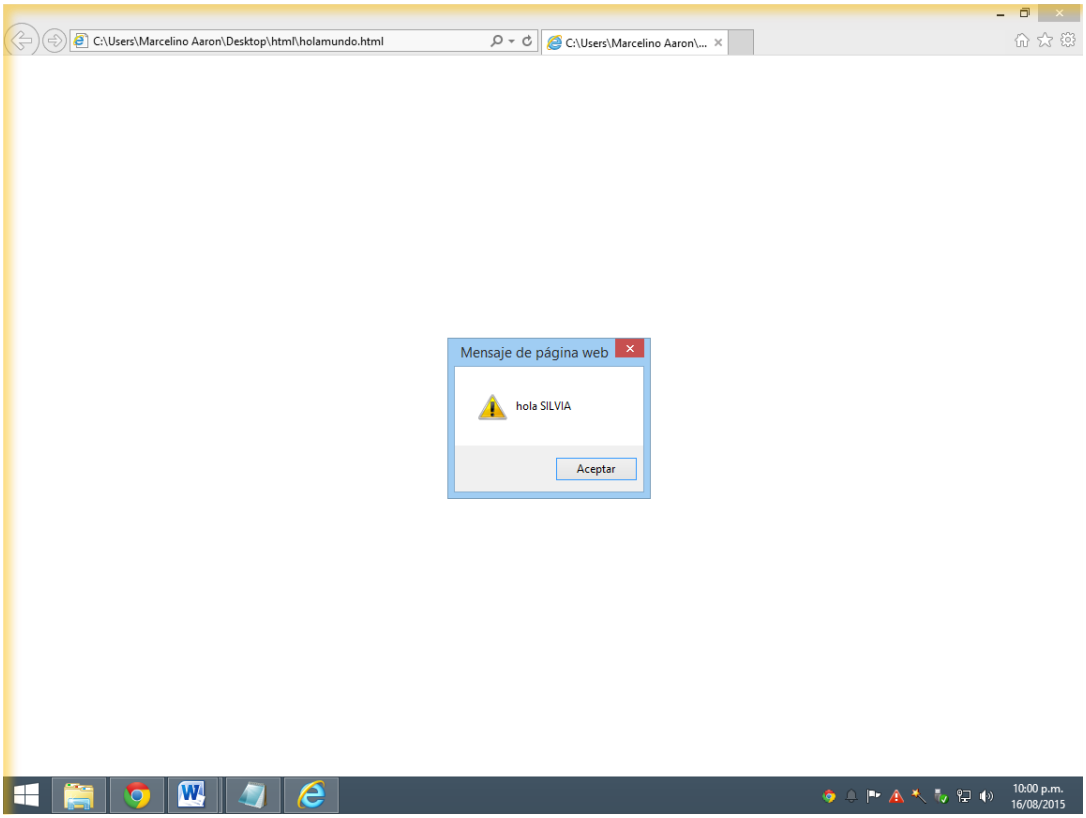
Paso 2: Elegimos JavaScript file y le asignamos un nombre pero ese nombre cualquiera debe de empezar por mayúscula ya que java reconoce las mayúsculas y minúsculas en este caso le puse de nombre **Ejemplo**y aceptamos.

Paso 3: Digitamos las estructuras del programa como indica en el siguiente texto:

Ejemplo:

```
<html><head>
<script>
functionvalor()
{
var nombre;
nombre=prompt("Introduce Nombre:","");
alert("hola "+nombre);
}
</script>
</head>
<body onload=valor();>
</body></html>
```


C:\Users\Marcelino Aaron\Desktop\html\holamundo.html
C:\Users\Marcelino Aaron\...
Mensaje de usuario de Explorer
Mensaje del script:
Introduce Nombre:
Aceptar
Cancelar
SILVIA
10:08 p.m.
16/08/2015


C:\Users\Marcelino Aaron\Desktop\html\holamundo.html
C:\Users\Marcelino Aaron\...
Mensaje de página web
hola SILVIA
Aceptar
10:00 p.m.
16/08/2015

Nota: Para finalizar una frase siempre se pone punto y coma además no olvidar de las llaves al inicio y al final como indica la figura.

Paso 4: Compilamos para verificar que no existe ningún error de sintaxis.

Paso 5: Le damos run o ejecutar y nos aparece la aplicación final como indica el grafico.

Síntesis

Se puede decir que JavaScript es un lenguaje que puede ser utilizado por profesionales y para quienes se inician en el desarrollo y diseño de sitios Web. No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos.

Muchos confunden el Javascript con el Java pero ambos lenguajes son diferentes y tienes sus características singulares. Javascript es un lenguaje interpretado, basado en prototipos, mientras que Java es un lenguaje más orientado a objetos.

Es necesario resaltar que hay dos tipos de JavaScript: por un lado está el que se ejecuta en el cliente, este es el Javascript propiamente dicho, aunque técnicamente se denomina Navigator JavaScript. Pero también existe un Javascript que se ejecuta en el servidor, es más reciente y se denomina LiveWireJavascript.

La estandarización de Javascript comenzó en conjunto con ECMA en noviembre de 1996. Es adoptado este estándar en Junio de 1997 y luego también por la "Internacional OrganizationforStandardization" (ISO). El DOM por sus siglas en inglés "Modelo de Objetos del Documento" fue diseñado para evitar incompatibilidades.

El código JavaScriptse encuentra dentro de las etiquetas <body></body> de nuestras páginas Web. Por lo general se insertan entre:

<script></script>.

También pueden estar ubicados en ficheros externos usando:

<script type="text/javascript" src="micodigo.js"></script>

Su sintaxis es similar a la usada en Java y C, al ser un lenguaje del lado del cliente este es interpretado por el navegador, no se necesita tener instalado ningún Framework.

Javascriptsoporta la mayoría de los navegadores como Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros.

Tener en cuenta que aunque Javascript sea soportado en gran cantidad de navegadores nuestros usuarios pueden elegir la opción de Activar/Desactivar el Javascript en los mismos.

Apreciación crítica y sugerencias

Sabemos que hoy en día, en nuestro país se apuesta poco o nada por la investigación, es por ello que espero que esta monografía motive a seguir investigando. El mundo de la informática, específicamente de la programación es tan complejo que lo que hemos desarrollado se vuelve tan pequeño con lo que queda por conocer.

Podemos observar que los costos de las piezas de hardware han disminuido de manera espectacular, al punto de que las computadoras personales se han convertido en artículos domésticos. Por desgracia, los costos para el desarrollo de programas se incrementan de manera constante conforme los programadores desarrollan aplicaciones más complejas y poderosas; así el desarrollo del software está ayudando a las empresas a controlar e incluso a reducir costos (programación estructurada, mejoramiento paso a paso, uso de funciones, programación basada en objetos, programación orientada a objetos, diseño orientado a objetos y programación genérica).

Asimismo, recomiendo que se integren cursos de programación en la malla curricular de los estudiantes de los últimos años de secundaria, que les permita acceder a estos conocimientos y prepare convenientemente para utilizarlos en su desarrollo intelectual.

Referencias

Brena, R. (1997). *Lenguajes Formales y Autómatas***.** Centro de Inteligencia Artificial, Instituto Tecnológico y de Estudios Superiores de Monterrey. Monterrey - México

Brookshear, J. Glenn (1993). *Teoría de la Computación, Lenguajes formales, Autómatas y Complejidad.* Estados Unidos*:* Addison Wesley Iberoamericana.

Deitel, H. y Deitel, P. (2012)*COMO PROGRAMAR EN JavaScritp***.** México:Editorial PEARSON Educación.

Joyanes A. (2011). *PROGRAMACION EN JavaSript: Metodología, Algoritmos y estructura de Datos.*España:Editorial Mc Graw Hill.

Monreal, J. (2007).*EL MUNDO DE LA COMPUTACION TOMO 3 Y 4***.** Barcelona-España: Editorial OCEANO

PáginasWEB

- http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C
- http://es.wikipedia.org/wiki/C%2B%2B
- http://www.mailxmail.com/curso-holistica-informatica/informatica-aplicaciones-computacionales