**JavaScript**

**MVC - MODELO VISTA CONTROLADOR**

**MVC**: Se une en tiempo de ejecución.

**Modelo:**

* Es el componente donde los datos son el principal componente.
* Los datos se encuentran en colecciones de datos (base de datos o archivos de texto).
* No tiene conocimiento acerca de las vistas y controladores.

**Vista:**

* Permite que el usuario interactúe con la aplicación.
* La interacción es a través de interfaces de usuario.
* Permite visualizar los datos del modelo.
* Un modelo puede tener muchas vistas, ejemplo: visualizar una colección de datos a través de diferentes diagramas: torta, barras, etc..
* Es notificada cuando el modelo sufre algún cambio (Se actualiza).

**Controlador:**

* Es el encargado de actualizar la vista cuando el modelo realiza algún cambio, y viceversa.
* Actúa como puente entre la vista y el modelo, notificando los cambios que se realizan.
* No se encarga de manipular directamente los datos, ni tampoco mostrarle al usuario una vista de estos, solo es un puente entre la vista y el modelo.

**-Colecciones de datos** (Modelo)

**-JavaScript** (Controlador) – Captura las distintas acciones de la vista notificando al modelo sobre algún cambio que se haya hecho a través de la vista y viceversa. – En ocasiones el controlador puede hacer las veces de vista

**-Documento HTML** (Vista)

**Sintaxis de JavaScript**

**Literal:** Un conjunto de caracteres de valor constante**,** es decir no cambia el valor del literal durante la ejecución del programa.

**Literales numéricos:**

343, 23.4, 556e2 \*\*e: elevados a la segunda potencia.

**Literales de cadena:**

“Maria Perez”, ‘Hola como estas?’

**Literales de expresiones:**

3+5, 4\*3, 8/4+3. Evalúa dos o más expresiones.

**Literales de arreglo:**

[2,34,54,677], [“Hola”, “como”, “estas”]. Almacena datos del mismo tipo de forma secuencial.

**Literales de objeto:**

{codigo:“002”, nombre:“Zapatos”, precio:25.000, descripcion:“Color Rojo”}. Los objetos tienen características, por eso en el ejemplo se habla de un producto con un código, un precio, etc…

**Literales de funciones:**

Function miFuncion(){

Document.write(“Hola mundo!”);

}

La estructura de una función.

**Variables:**

-Almacena valores en memoria

-Se define con la palabra reservada **var**.

Ejemplo: var m;

-No es necesario colocar var

Ejemplo: resultado = 43

-El valor de una variable sin valor es **undefined.**

**Identificador:**

Es el nombre que se le da una variable, debe ser único y no se debe usar en otra variable, puede tener: **Letras**, símbolo **$**, guión de piso **\_**

No debe empezar por un número, ni utilizar palabras reservadas.

Es sensible a las mayúsculas.

**Variables principales:**

1. **Texto o string**

var animal = “oso”;

1. **Número**

var numero = 123;

**Operadores:**

Operadores aritméticos: + - \* /

Operadores de asignación: =

Operaodres de comparación: ==, !=, <, >, <=, >=.

**Sentencias:**

-Son agrupaciones de valores y variables.

-Finalizan con (;)

Ejemplo:

var numero = 123;

-Las sentencias se pueden agrupar en bloques.

Ejemplo:

Function myFuncion(){

var hola = “I love js”;

document.write(“Hola”);

}

-Comienza con identificadores de sentencia (palabras reservadas).

**Sentencias:**

**break** – Permite finalizar las sentencias de tipo switch.

**catch:** - identifica e indica el fin de un grupo de sentencias que están agrupadas.

**continue** – En un ciclo lo interrumple en la sentencia que esta ejecutando y regresa a una nueva iteración.

**do while** – Ciclo que ejecuta un bloque de sentencias que tiene agrupadas y siempre y cuando la condición de continuidad de while sea verdadera.

**for** – Ciclo que ejecuta un bloque de sentencias que termina dependiendo de una condición relacionada con un contador.

**function** – Indica que se va a declarar una función.

**if… else** – Indica que solo se va a ejecutar un bloque de sentencias cuando se cumple una condición.

**return** – Permite salir de una función.

**switch** – Evalúa un bloque de sentencias dependiendo de varias opciones.

**throw** – Permite lanzar un error voluntariamente o de manera intencional en nuestro código para determinar el comportamiento del programa.

**try** – Indica el inicio de un bloque de sentencias que están agrupadas para detectar errores.

**var** – Permite declarar una variable.

**while** – Representa un ciclo que ejecuta varias veces un bloque de sentencias y termina cuando la condición de continuidad no se cumple.

**Espacios en blanco y saltos de línea:**

Js permite espacios en blanco y saltos de línea cuando la sentencia es muy larga.

**Comentarios:**

Existen dos tipos:

-De una línea //

-De multiples líneas /\* \*/

**Objetos:**

Son representaciones de objetos de la vida real.

Tiene **Atributos o propiedades** (**características**) y **Metodos** (**acciones**).

-Existen dos formas para acceder a una **PROPIEDAD**:

nombreObjeto.propiedad

nombreObjeto[“propiedad”]

-Para acceder a un **METODO**:

nombreObjeto.nombreMetodo()

hay objetos:

Objetos Boolean

Objetos numericos

Objeto string

Objetos Dates

Objetos Maths y Expresiones regulares

Objetos Arrays

Objetos Funciones

**Existen tres formas para crear un Objeto:**

//1. Lista de pares de valores propiedad/valor

var producto1 = {codigo:"0001", nombre:"Gorra", precio:500}

//Accediendo a las propiedades de los objetos para mostrarla

document.getElementById("c1").innerHTML = producto1.codigo;

//2. Usando la palabra reservada new

var producto2 = new Object();

producto2.codigo = "005";

producto2.nombre = "camiseta";

producto2.precio = 7000;

//Accediendo a las propiedades de los objetos para mostrarla

document.getElementById("c1").innerHTML = producto2.codigo;

//3. Usando una funcion constructor:

function producto3(codigo, nombre, precio){

this.codigo = código; // Despues del igual se usa los nombres del parametro

this.nombre = nombre;

this.precio = precio;

}

var producto3 = new producto3("001", "Gorra", 50);

//Accediendo a las propiedades de los objetos para mostrarla

document.getElementById("c1").innerHTML = producto3.codigo;

**Js cuenta con constructores ya definidos:**

new Object()

Ejemplo: var producto = new Object();

new String()

new Boolean()

new Number()

new Array()

new RegExp()

new Function()

new Date()

**Funciones**

Son conjuntos de sentencias que ejecutan una actividad especifica

Se usa la palabra reservada **function**

Puede recibir o no parámetros

Las sentencias están agrupadas dentro de dos llaves {}

Para devolver un valor se utiliza la palabra reservada **return**

Existen tres formas para llamar una función:

1. Cuando hay un evento generado por el usuario
2. Llamado desde el código, ejemplo:

division(10,2);

function división(p,q){

return p / q

}

1. Cuando se llama así misma, ejemplo

(function(){

alert(“Esta función se invoca así misma”);

})();

Una función se puede asignar a una variable, ejemplo:

var z = function (x,y){return x - y};

var w = z(15,6);

**Tipos de Datos Strings y Numbers**

**Strings:** van dentro de comillas dobles o simples, empiezan desde la posición 0.

**Propiedades:**

Constructor – devuelve la función constructora de la cadena

Length – devuleve la longitud o el numero de elementos de una cadena

Prototype – permite agregar propiedades y métodos a un objeto o una función

**Métodos para las cadenas (String):**

charAt(); concat(); replace(); search(); slide(); toLowerCase(); toUpperCase(); toString(); etc…

**Numbers:** va sin comillas.

**Propiedades:**

Constructor – devuelve la función que crea la propiedad Prototype

Prototype – permite agregar propiedades y métodos a un objeto de tipo numérico en este caso.

MAX\_VALUE: muestra el numero mayor

MIN\_VALUE: muestra el numero menor

NaN: Not a Number, no hay numero, no hay ningún valor.

NEGATIVE\_INFINITY

POSITIVE\_INFINITY

**Métodos:**

toExponential(x); toFixed(x); toPrecision(x); toString(); valueOf().

**Tipos de Datos Date Object y Arrays**

***Rutas Relativas***

Ejemplo: src="js/piedra.js"

**src=””** Fuente del archivo

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

Javascript puede ir dentro del Head, Body o un archivo externo

JS interno:

<script type="text/javascript">

alert("Hola Mundo!");

</script>

JS Externo:

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

</script>

***Funciones predefinidas o Instrucciones***

**Las funciones predefinidas siempre llevan paréntesis**

alert("Hola mundo"); // muestra un aviso

document.write('Hola Mundo'); // escribe en el documento

isNan(); is not a number, texto que no se puede transformar en número.

prompt(); Trae cosas del usuario, inserta un dato de tipo texto (string)

confirm();

return

function permite declarar una funcion

Number(); Transforma a valor numérico un texto cuando es posible

parsInt(); Transforma a valor numérico un texto cuando es posible

prompt trae cosas del usuario, siempre inserta un dato de tipo texto, así se inserten números. Para solucionar esto se utiliza la Funcion parseInt

**parseInt** convierte cualquier numero que tenga un texto por dentro en un numero total. *parseInt: Pasear como entero*

**var peso;**

peso = prompt("¿Cuál es tu peso en kg?", "70");

alert( parseInt(peso) + 10);

**var peso;**

peso = prompt("¿Cuál es tu peso en kg?", "70");

peso = parseInt(peso);

alert( peso + 10);

*El navegador siempre hace primero lo que encuentre entre paréntesis.*

o se puede utilizar **Number / hace lo mismo que parseInt/**

**var peso;**

peso = prompt("¿Cuál es tu peso en kg?", "70");

peso = Number(peso);

alert( peso + 10);

function parImpar()

{

// esta variable solo existe en este funcion

var numero = **prompt**("Ingresa un número");

// **isNaN**: No es un numero - is not a number, true si el valor es alfanúmerico, false si es númerico.

if ( isNaN(numero))

{

alert("No me engañes, eso no es un número.");

}

else

{

//alert("Es un número");

var modulo = numero%2;

var tipo = (modulo == 0)?"Par":"Impar";

alert("El número " + numero + " es " + tipo)

};

}

**Podemos también crear funciones propias**

***Tipos de datos:***

**Numéricos:** 1,2,3, …

**Strings:** texto, va entre comillas “ ” o ‘ ’

**Booleanos:** Verdadero o Falso - true, false

***Variables***

Espacio en la memoria de un ordenador (RAM) donde se almacenara un valor que podrá cambiar durante la ejecución del programa. Var Es una instrucción nativa del lenguaje.

Sintaxis (Declaración): **var puntuación; var numero1 = 3;**

Han de comenzar por **letra,** símbolo **$** o **\_**

Son case sensitive (Reconoce mayúsculas y minúsculas)

No deben ser palabras reservadas

Se recomienda que sean descriptivos

Se declaran:

var puntuacion;

puntuacion=0

**var puntuacion=0;** *// Simplificado en una misma linea*

var puntuacion, record, jugador;

puntuación = 0;

record = 5000;

jugador = “juan”;

**var puntuación=0, record=5000, jugador=“juan”;** *// Simplificado*

***Concatenar:***

**Pegar cadenas, se utiliza +**

var nombre = "Giovanny";

var apellido = "Calderón";

alert(nombre + " " + apellido**)**

---+

var nombre=prompt("Introduce tu nombre por favor");

var apellido=prompt("Introduce tu apellido por favor"); **document.write("<p>");** //abre una linea nueva en el html

document.write("Bienvenido: " + nombre + " " + apellido); **document.write("</p>");** // cierra la linea en el html

--+

***Condicionales y Operadores en Programación***

### Operadores

### Asignación:

var numero1 = 3;

### Macintosh HD:Users:giovannycalderon:Desktop:Captura de pantalla 2015-05-20 a las 3.12.55 p.m..pngIncremento y Decremento: ++ --

var numero = 5;

++numero;

alert(numero); // numero = 6

var numero = 5;

--numero;

alert(numero); // numero = 4

**Negación**

var visible = true;

alert(!visible); // Muestra "false" y no "true"

var cantidad = 0;

vacio = !cantidad; // vacio = true

cantidad = 2;

vacio = !cantidad; // vacio = false

**And**

&& y su resultado solamente es true si los dos operandos son true

var valor1 = true;

var valor2 = false;

resultado = valor1 && valor2; // resultado = false

valor1 = true;

valor2 = true;

resultado = valor1 && valor2; // resultado = true

**Or**

|| y su resultado es true si alguno de los dos operandos es true

var valor1 = true;

var valor2 = false;

resultado = valor1 || valor2; // resultado = true

valor1 = false;

valor2 = false;

resultado = valor1 || valor2; // resultado = false

**Matemáticos**

Hay priporidad: Multiplcacion y división tiene preferencia

alert(5+6\*3); // resultado es 23 --- 6\*3=18+5=23

alert((5+6)\*3); // resultado es 33 --- los paréntesis ayudan a ordenar la operación

var numero1 = 10;

var numero2 = 5;

resultado = numero1 / numero2; // resultado = 2 --- Division

resultado = 3 + numero1; // resultado = 13 ---- Suma

resultado = numero2 – 4; // resultado = 1 ---- Resta

resultado = numero1 \* numero2; // resultado = 50 --- Multiplicación

resultado5 = numero1 % numero2; // resultado = 0 --- Residuo

**Comparacion**

mayor que (>), menor que (<),

**Relacionales**

Los operadores relacionales definidos por JavaScript son idénticos a los que definen las matemáticas: mayor que (>), menor que (<), mayor o igual (>=), menor o igual (<=), igual que (==) y distinto de (!=) comparar negativo.

var numero1 = 3;

var numero2 = 5;

resultado = numero1 > numero2; // resultado = false

resultado = numero1 < numero2; // resultado = true

numero1 = 5;

numero2 = 5;

resultado = numero1 >= numero2; // resultado = true

resultado = numero1 <= numero2; // resultado = true

resultado = numero1 == numero2; // resultado = true

resultado = numero1 != numero2; // resultado = false

**Igualdad**

El operador == se utiliza para **comparar** el valor de dos variables,

var numero1 = 10;

var numero2 = 5;

// El operador "=" asigna valores

var numero1 = 5;

resultado = numero1 = 3; // numero1 = 3 y resultado = 3

**// El operador "==" compara variables**

var numero1 = 5;

resultado = numero1 == 3; // numero1 = 5 y resultado = false

***Truco avanzado: caracteres secretos***

-Salto de línea \n

con , separo los parámetros. peso = prompt("¿Cuál es tu peso en kg?", "100");

***Manejador de eventos como atributo HTML***

***Manejador de eventos Semántico***

***Manejador de eventos Múltiple***

**src=””** Fuente del archivo

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

pesoEnMarte; **// CamelCase** es una técnica donde se coloca la primera letra en minúscula y cuando se comienza otra palabra la primera letra se coloca en mayúscula.

***strings***

alert("2" + 5 + 8);

258 *// Como el primer dato es un string o texto los demas valores los toma como string*

alert(5 + 8 "2");

132

alert("2" + 5 \* 8);

240 *// porque la multiplicación es mas fuerte que la suma*

**Bloque de instrucciones:**

**if**(opcionUsuario == piedra)  
 {  
        alert("Empate!");  
    }

**Anidar condiciones dentro de condiciones:**

**if**(opcionUsuario == piedra)  
{  
    **if**(opcionMaquina == piedra)  
    {  
        alert("Empate!");  
    }  
    **else** **if**(opcionMaquina == papel)  
    {  
        alert("Perdiste :( ");  
    }  
    **else** **if**(opcionMaquina == tijera)  
    {  
        alert("Ganaste!");  
    }  
}

**Array**

Vectores o Arrays desde Cero, van con corchetes cuadrados [ ]

Hay 2 formas de declararlos

**var articulos= ["zapatillas", "pantalon", "camiseta", "calcetines"];**

var artículos=new Array(“zapatillas”,”pantalon”,”camiseta”, “calcetines”);

Tambien se pueden declarar array vacios

**var artículos=[];**

**var** opciones = ["Piedra", "Papel", "Tijera"];

alert(opciones[1]); //muestra Papel

Donde: “Piedra es igual a 0”

“Papel es igual a 1”

“Tijeras es igual a 2”

**Operador Ternario**

Operador Ternario:

o.sexo = (s==true)?"Macho":"Hembra";

//Variable = (condición)?verdadero:falso, si no se cumple ejecuta lo que esta despues de los : si se cumple ejecuta lo que esta despues de ?

// A diferencia de un if else el operador ternario solo permite ejecutar una instrucción

**Metodos y propiedades:**

**.lenght** muestra los elementos que contiene un array

Artículos.length

**.push** agrega elementos al final del array

**.unshift** agrega elementos al comienzo del array

**.pop** elimina elementos al final del array

**.shift** elimina elementos al comienzo del array

**.createElement** crea un elemento HTML con sus atributos. Ejemplo:

var imagen = document.createElement("img");

imagen.src = o.foto;

imagen.alt = o.nombre;

imagen.title = o.nombre;

// document tiene un objeto para poder acceder al body y uno para acceder al head

document.body.appendChild(imagen);

\*\*\*\*\*

var articulos= ["zapatillas", "pantalon", "camiseta", "calcetines"];

// Acceder al ultimo elemento del array

document.write("<p>El último elementos es <strong>" + articulos[articulos.length-1] + "</strong></p>");

----- El último elementos es calcetines

\*\*\*\*\*\* -----

**Generar un numero aleatorio:**

//Genera un número aleatorio entre un rango de enteros  
**function** aleatorio(minimo, maximo)  
{  
    **var** numero = Math.floor( Math.random() \* (maximo - minimo + 1) + minimo );  
    **return** numero;  
}

**var** opcionMaquina = aleatorio(0,2);

Math es un objeto

Es el paquete de opciones matemáticas que tiene JS por dentro.

Math.floor

Redondea al mínimo los números decimales

Ejemplo: 2.7 se redondea a 2

Math.ceil \*\*techo\*\*\*

Pasa al siguiente numero los números decimales

Ejemplo: 2.7 se redondea a 3

Math.sqrt

Raiz cuadrada

Math.random

Genera un numero random entre 0 y 1 random= 0 – 1 ejemplo: 0.1, 0.3, 0.9

*Random \* (máximo – mínimo + 1) + min*

Suponiendo que quiero un numero entre 6 y 8

*8 – 6 + 1*

*3*

suponiendo que random genere 0.9

*0.9 \* 3*

*2.7 + 6*

*8.7*

*redondeado hacia abajo da 8*

\*\*\*\*\*-----

Buscar en google Javascript Math MDN(Mozilla developer network)

Tarea: piedra papel tijera lagarto spot

Hacer la interfaz con html en vez de prompt y alert

\*\*\*\*\*-----

**Clase 2**

Para escribir en el HTML se utiliza **document.write**

**var** edad = 28; document.write("Tu edad <br /> es " + edad );

**ln** se utiliza para separa un texto del otro:

document.writeln("escribiendo en HTML");

document.writeln("basico");

escribiendo en HTML basico

Para poder hacer un salto delinea se debería hacer desde el HTML, se pueden agregar etiquetas HTML.

document.writeln("escribiendo en HTML <br/>");

document.writeln("basico");

escribiendo en HTML

basico

var edad = 28;

document.writeln("Tu edad es " + edad);

**MATRICES o VECTORES [ ]**

Son Arrays, colección de información unidas por un mismo concepto.

Un Vector tiene una dimensión

Una Matriz tiene 2 dimensiones.

Menú:

Producto / Ventas / Contacto

0 1 2

*var menu = ["Productos", "Ventas", "Contacto"];*

*document.write(menu[2]);*

En Js para ubicar un punto: [2][0]

Se elije primero X luego Y.

Matriz Multidimensional

var dofa = [["Fuerza", "Oportunidad"], ["Debilidades", "Amenazas"]];

document.write(dofa[0][1]);

Oportunidad

**Funciones - function**

La Funciones solo pueden tener textos o números y deben comenzar con una letra. Se invocan con ()

Declarar una function

**function** explosion()  
{  
    alert("BOOM!!");  
    document.write("<h1>BOOM! Elegiste un area minada :(</h1>");  
}

Invocar una Function

ganaste();

**Objetos del Navegador**

Html tenemos objetos en Document Object Model:

document.write(“Hola mamá!);

write es una función del objeto Document

objeto.funcion o variable(“Parametro”);

Math

Los 3 principales Objetos de Javascript son:

navigator, window, document

**Objetos en JS**

Los Objetos se declara con la primera letra en mayúscula y se declaran con un bloque de código. Termina en **,** y no ;

Al declarar el objeto el bloque termina en **;**

var Pokemon =

{

nombre: "Picachu",

tipo: "Eléctrico"

};

**function** Pokemon(n,v,a)   
{  
    **this**.grito = "Pika!";  
    **this**.nombre = n;  
    **this**.vida = v;  
    **this**.ataque = a;  
    **this**.gritar = **function** ()  
    {  
        alert(**this**.grito);  
    }  
}  
  
**function** inicio()  
{  
    **var** rattata = **new** Pokemon("Rattata", 40, 2);  
    rattata.vida = rattata.vida - 13;  
    nombrePokemon.textContent = rattata.nombre;  
}

//Variables de asignación por valor

//Variables de asignación por referencia

Se pueden crear Objetos dentro de Objetos.

return

this: muchas variables diferentes con diferentes valores pero con la misma estructura

new

**Depuracion de Objetos**

Se puede hacer con *console.log*

*O se escribe directamente el nombre*

***Objeto que contiene un metodo***

*Metodo= función Es una función metida dentro de una clase*

*Atributo = Variable Cuando la variable es parte de un Objeto se llama Atributo*

\*\*\*\*\*-----

*Tarea*

*Llenar datos de prueba Pokemon*

**Ciclos**

**While, do, for**

*Son formas donde se escribe código que se repita de manera constante.*

***While*** *--- mientras – si se cumple la condición ocurre esto, hasta que se deje de cumplir la condición*

***While*** *--- mientras – si se cumple la condición ocurre esto, hasta que se deje de cumplir la condición*

***Elemento o variabe iterador:*** *Es la variable que hace cada ciclo, cada vez que un ciclo ocurre se llama iteración.*

**document.getElementById** Se llama al html sobre el cual se quiere trabajar

innerHTML = “Hola papá!” cambia el contenido html original por el nuevo en js

c.innerHTML += " Hola papá!"; conserva el contenido html original y le agrega el nuevo desde js

innerText

**function** inicio()  
{  
    c = document.getElementById("caja");

    c.innerHTML = "<strong>" + calculo + "</strong>";

}

<body>

**<div** **id**="caja"**>**Nada**</div>**

**<script** **type**="text/javascript"**>**  
    inicio();  
    **</script>**

</body>

Caja de texto

**<input** **type**="text" **id**="alejandra" **/>**  
    **<input** **type**="button" **id**="botoncito" **value**="Tocame~" **/>**

**CANVAS**

Los canvas por defecto son blancos

La esquina superior izquierda es 0,0

Es como un plano cartesiano, pero No tiene puntos negativos.

Canvas es toda el área

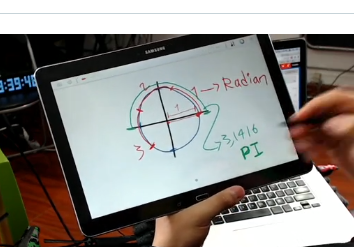
Para dibujar en canvas se dibuja sobre el eje de coordenadas = contexto

Para hacer círculos se utiliza pi

*La media vuelta de un cirulo ilustrado en Radianes es Pi 3.14159265*

*Pi \* 2 es una vuelta entera del circulo.*

Radian: es el radio pegado sobre la curva



Para verificar que funcione:

*Html*

<script>

inicio();

</script>

*Js:*

alert("fuera");

function inicio()

{

alert("dentro");

}

/\*\*\*\*