Modulo Programación

Clases y Funciones

GUIA DEL ALUMNO

Contenidos

Clases

El lenguaje JavaScript no es un lenguaje orientado a objetos completo, pero permite definir clases con sus atributos y responsabilidades. Finalmente nos permite definir objetos de estas clases.
Pero el otro pilar de la programación orientada a objetos, es decir la herencia, no está implementada en el lenguaje.
Veremos la sintaxis para la declaración de una clase y la posterior definición de objetos de la misma.

Desarrollaremos una clase que represente un cliente de un banco.
La clase cliente tiene como atributos:

nombre
saldo

y las responsabilidades o métodos de la clase son:

Constructor (inicializamos los atributos del objeto)
depositar
extraer

Luego debemos implementar los siguientes métodos (normalmente el constructor se utiliza el caracter mayúscula):

```
function Cliente(nombre,saldo)
{ this.nombre=nombre;
this.saldo=saldo;
this.depositar=depositar;
this.extraer=extraer;
}

function depositar(dinero)
{
this.saldo=this.saldo+dinero;
}

function extraer(dinero)
{
this.saldo=this.saldo-dinero;
}
```

El nombre de la clase coincide con el nombre de la función principal que implementamos (también llamado constructor de la clase):

```
function cliente(nombre,saldo)
{
  this.nombre=nombre;
  this.saldo=saldo;
  this.depositar=depositar;
  this.extraer=extraer;
}
```

A esta función llegan como parámetro los valores con que queremos inicializar los atributos. Con la palabra clave 'this' diferenciamos los atributos de los parámetros (los atributos deben llevar la palabra clave this)

```
this.nombre=nombre;
this.saldo=saldo;
```

También en el constructor inicializamos la referencia a todos los métodos que contendrá la clase (esto es muy importante y necesario para entender porque las otras dos funciones pertenecen a esta clase):

```
this.depositar=depositar;
this.extraer=extraer;
```

Por último, implementamos todos los métodos de la clase:

```
function depositar(dinero)
{
  this.saldo=this.saldo+dinero;
}

function extraer(dinero)
{
  this.saldo=this.saldo-dinero;
}
```

De nuevo recordemos que diferenciamos los atributos de la clase por la palabra clave this.

Ahora veamos el archivo HTML completo donde además definiremos un objeto de la clase planteada:

```
<html>
<head>
<title>Problema</title>

<script type="text/javascript">
 function Cliente(nombre,saldo)
 {
  this.nombre=nombre;
  this.saldo=saldo;
  this.depositar=depositar;
  this.extraer=extraer;
 }

 function depositar(dinero)
 {
  this.saldo=this.saldo+dinero;
 }

 function extraer(dinero)
 {
  this.saldo=this.saldo-dinero;
 }

</script>

</head>
<body>

<script type="text/javascript">
 var cliente1;
 cliente1=new Cliente('diego',1200);
 document.write('Nombre del cliente:'+cliente1.nombre+'<br>');
 document.write('Saldo actual:'+cliente1.saldo+'<br>');
 cliente1.depositar(120);
 document.write('Saldo luego de depositar $120---->'+cliente1.saldo+'<br>');
 cliente1.extraer(1000);
 document.write('Saldo luego de extraer $1000---->'+cliente1.saldo+'<br>');
</script>

</body>
</html>
```

Hemos dividido la declaración de la clase en un bloque Javascript distinto a donde definimos un objeto de la misma, esto no es obligatorio, pero podemos ver que queda más claro.

Para definir un objeto de la clase Cliente tenemos:

```
var cliente1;
cliente1=new Cliente('diego',1200);
```

Luego las llamadas a métodos le antecedemos el nombre del objeto llamado cliente1:

```
document.write('Nombre del cliente:'+cliente1.nombre+'<br>');
document.write('Saldo actual:'+cliente1.saldo+'<br>');
cliente1.depositar(120);
document.write('Saldo luego de depositar $120---->'+cliente1.saldo+'<br>');
cliente1.extraer(1000);
document.write('Saldo luego de extraer $1000---->'+cliente1.saldo+'<br>');
```

Podemos decir que la ventaja que podemos obtener con el planteo de clases es hacer nuestros programas mucho más organizados, entendibles y fundamentalmente, poder reutilizar clases en distintos proyectos.

Definicion de Varias Clases

En JavaScript podemos definir varias clases en un mismo programa.
Vamos a desarrollar un programa que contenga dos clases. Plantearemos una clase Numeroquiniela que representa una persona que elige un número de quiniela y además registra su nombre, la clase tiene por objetivo la carga por el teclado del número deseado.
Por otra parte crearemos una clase Bolillero que sortee un valor aleatorio entre 1 y 10 (que representa el valor extraído del bolillero).
La codificación de las dos clases es:

```
<html>
<head>
<title>Problema</title>
<script type="text/javascript">

 //clase Numeroquiniela **********************************************
 function Numeroquiniela(nombre)
 {
   this.nombre=nombre;
   this.cargarnumero=cargarnumero;
   this.verificarsigano=verificarsigano;
 }

 function cargarnumero()
 {
```

```
  this.numero=prompt("Que número de quiniela quiere?","");
 }

 function verificarsigano(num)
 {
  if (this.numero==num)
    return true;
  else
    return false;
 }

 //clase Bolillero ************************************************
 function Bolillero()
 {
  this.numero=-1;
  this.sortear=sortear;
 }

 function sortear()
 {
  this.numero=parseInt(Math.random()*10)+1;
 }

</script>
</head>
<body>

<script type="text/javascript">
 var numeroquiniela1;
 numeroquiniela1=new Numeroquiniela("juan");
 numeroquiniela1.cargarnumero();
 var numeroquiniela2;
 numeroquiniela2=new Numeroquiniela("ana");
 numeroquiniela2.cargarnumero();
 var bolillero;
 bolillero=new Bolillero();
 bolillero.sortear();
 document.write('Numero sorteado:' + bolillero.numero + '<br>');
 document.write(numeroquiniela1.nombre + ' eligió ' + numeroquiniela1.numero +'<br>');
 document.write(numeroquiniela2.nombre + ' eligió ' + numeroquiniela2.numero +'<br>');
 if (numeroquiniela1.verificarsigano(bolillero.numero))
 {
  document.write(numeroquiniela1.nombre + ' a ganado <br>');
 }
 if (numeroquiniela2.verificarsigano(bolillero.numero))
 {
  document.write(numeroquiniela2.nombre + ' a ganado <br>');
```

```
  }
</script>
</body>
</html>
```

Al constructor de la clase Numeroquiniela llega como parámetro el nombre de la persona que la compra (podíamos cargarlo por teclado al nombre también).
Al número que selecciona lo cargamos por teclado. La clase Numeroquiniela además tiene otra responsabilidad, que es avisarnos si a ganado según el número sorteado.
Por otro lado en la página html definimos dos objetos de la clase Numeroquiniela y uno de la clase Bolillero:

La clase Date

JavaScript dispone de varias clases predefinidas para acceder a muchas de las funciones normales de cualquier lenguaje, como puede ser el manejo de vectores o el de fechas.

Esta clase nos permitirá manejar fechas y horas. Se invoca así:

```
fecha = new Date();//creación de un objeto de la clase Date
fecha = new Date(año, mes, dia);
fecha = new Date(año, mes, dia, hora, minuto, segundo);
```

Si no utilizamos parámetros, el objeto fecha contendrá la fecha y hora actuales, obtenidas del reloj de nuestra computadora. En caso contrario hay que tener en cuenta que los meses comienzan por cero. Así, por ejemplo:

```
navidad06 = new Date(2006, 11, 25)
```

El objeto Date dispone, entre otros, de los siguientes métodos:

getYear()	Obtiene y coloca, respectivamente, el año de la fecha. Éste se devuelve como número de 4 dígitos excepto en el caso en que esté entre 1900 y 1999, en cuyo caso devolverá las dos últimas cifras.
setYear(año)	
getFullYear()	Realizan la misma función que los anteriores, pero sin tanta complicación, ya que siempre devuelven números con todos sus dígitos.
setFullYear(año)	
getMonth()	
setMonth(mes)	
getDate()	

setDate(dia)	
getHours()	
setHours(horas)	
getMinutes()	
setMinutes(minutos)	
getSeconds()	Obtienen y colocan, respectivamente, el mes, día, hora, minuto y segundo de la fecha.
setSeconds(segundos)	
getDay()	Devuelve el día de la semana de la fecha en forma de número que va del 0 (domingo) al 6 (sábado)

Ejemplo: Mostrar en una página la fecha y la hora actual.

```
<HTML>
<HEAD>

<SCRIPT type="text/javascript">
  function mostrarFechaHora()
  {
    var fecha fecha=new Date();
    document.write('Hoy es ');
    document.write(fecha.getDate()+'/');
    document.write((fecha.getMonth()+1)+'/');
    document.write(fecha.getFullYear());
    document.write('<br>');
    document.write('Es la hora ');
    document.write(fecha.getHours()+':');
    document.write(fecha.getMinutes()+':');
    document.write(fecha.getSeconds());
  }

//Llamada a la función
  mostrarFechaHora();
</SCRIPT>

</HEAD>
<BODY>

</BODY> </HTML>
```

En este problema hemos creado un objeto de la clase Date. Luego llamamos una serie de métodos que nos retornan datos sobre la fecha y hora actual del equipo de computación donde se está ejecutando el navegador.

Es bueno notar que para llamar a los métodos disponemos:
<nombre de objeto>.<nombre de método>(parámetros)

Clase Math

Esta clase es un contenedor que tiene diversas constantes (como Math.E y Math.PI) y los siguientes métodos matemáticos:

Método	Descripción	Expresión de ejemplo	Resultado del ejemplo
Abs	Valor absoluto	Math.abs(-2)	2
sin, cos, tan	Funciones trigonométricas, reciben el argumento en radianes	Math.cos(Math.PI)	-1
asin, acos, atan	Funciones trigonométricas inversas	Math.asin(1)	1.57
exp, log	Exponenciación y logaritmo, base E	Math.log(Math.E)	1
Ceil	Devuelve el entero más pequeño mayor o igual al argumento	Math.ceil(-2.7)	-2
Floor	Devuelve el entero más grande menor o igual al argumento	Math.floor(-2.7)	-3
Round	Devuelve el entero más cercano o igual al argumento	Math.round(-2.7)	-3
min, max	Devuelve el menor (o mayor) de sus dos argumentos	Math.min(2,4)	2
Pow	Exponenciación, siendo el primer argumento la base y el segundo el exponente	Math.pow(2,3)	8
Sqrt	Raíz cuadrada	Math.sqrt(25)	5

Random	Genera un valor aleatorio comprendido entre 0 y 1.	Math.random()	Ej. 0.7345

Ejemplo: Confeccionar un programa que permita cargar un valor comprendido entre 1 y 10. Luego generar un valor aleatorio entre 1 y 10, mostrar un mensaje con el número sorteado e indicar si ganó o perdió:

```
<html>
<head>
</head>
<body>

<script type="text/javascript">
  var selec=prompt('Ingrese un valor entre 1 y 10','');
  selec=parseInt(selec);
  var num=parseInt(Math.random()*10)+1;
  if (num==selec)
    document.write('Ganó el número que se sorteó es el '+ num);
  else
    document.write('Lo siento se sorteó el valor '+num+' y usted eligió el '+selec);
</script>

</body>
</html>
```

Para generar un valor aleatorio comprendido entre 1 y 10 debemos plantear lo siguiente:

```
var num=parseInt(Math.random()*10)+1;
```

Al multiplicar Math.random() por 10, nos genera un valor aleatorio comprendido entre un valor mayor a 0 y menor a 10, luego, con la función parseInt, obtenemos sólo la parte entera. Finalmente sumamos uno. El valor que cargó el operador se encuentra en:

```
var selec=prompt('Ingrese un valor entre 1 y 10','');
```

Con un simple if validamos si coinciden los valores (el generado y el ingresado por teclado)

Clase String

Un string consiste en uno o más caracteres encerrados entre simple o doble comillas.
Concatenación de cadenas (+)
JavaScript permite concatenar cadenas utilizando el operador +.
El siguiente fragmento de código concatena tres cadenas para producir su salida:

```
var final='La entrada tiene ' + contador + ' caracteres.';
```

Dos de las cadenas concatenadas son cadenas literales. La del medio es un entero que automáticamente se convierte a cadena y luego se concatena con las otras.

Propiedad length
Retorna la cantidad de caracteres de un objeto String.

```
var nom='Juan';
document.write(nom.length); //Resultado 4
```

Métodos

charAt(pos)	Retorna el caracter del índice especificado. Comienzan a numerarse de la posición cero var nombre='juan'; var caracterPrimero=nombre.charAt(0);
substring (posinicial, posfinal)	Retorna un String extraída de otro, desde el caracter 'posinicial' hasta el 'posfinal'-1: cadena3=cadena1.substring(2,5); *En este ejemplo, "cadena3" contendrá los caracteres 2, 3, 4 sin incluir el 5 de cadena1 (Cuidado que comienza en cero).*
indexOf (subCadena)	Devuelve la posición de la subcadena dentro de la cadena, o -1 en caso de no estar. Tener en cuenta que puede retornar 0 si la subcadena coincide desde el primer caracter. var nombre='Rodriguez Pablo';

	var pos=nombre.indexOf('Pablo');
	if (pos!=-1)
	  document.write ('Está el nombre Pablo en la variable nombre');

toUpperCase()	Convierte todos los caracteres del String que invoca el método a mayúsculas: cadena1=cadena1.toUpperCase(); *Luego de esto, cadena1 tiene todos los caracteres convertidos a mayúsculas.*
toLowerCase()	Convierte todos los caracteres del String que invoca el método a minúsculas: cadena1=cadena1.toLowerCase(); *Luego de esto, cadena1 tiene todos los caracteres convertidos a minúsculas.*

Ejemplo: Cargar un string por teclado y luego llamar a los distintos métodos de la clase String y la propiedad length.

```
<html>
<head>
</head>
<body>

<script type="text/javascript">
 var cadena=prompt('Ingrese una cadena:',");
 document.write('La cadena ingresada es:'+cadena);
 document.write('<br>');
 document.write('La cantidad de caracteres son:'+cadena.length);
 document.write('<br>');
 document.write('El primer caracter es:'+cadena.charAt(0));
 document.write('<br>');
 document.write('Los primeros 3 caracteres son:'+cadena.substring(0,3));
 document.write('<br>');
 if (cadena.indexOf('hola')!=-1)
 document.write('Se ingresó la subcadena hola');
 else
   document.write('No se ingresó la subcadena hola');
 document.write('<br>');
 document.write('La cadena convertida a mayúsculas es:'+cadena.toUpperCase());
```

```
  document.write('<br>');
  document.write('La cadena convertida a minúsculas es:'+cadena.toLowerCase());
  document.write('<br>');
</script>

</body>
</html>
```

Funciones

En programación es muy frecuente que un determinado procedimiento de cálculo definido por un grupo de sentencias tenga que repetirse varias veces, ya sea en un mismo programa o en otros programas, lo cual implica que se tenga que escribir tantos grupos de aquellas sentencias como veces aparezca dicho proceso.

La herramienta más potente con que se cuenta para facilitar, reducir y dividir el trabajo en programación, es escribir aquellos grupos de sentencias una sola y única vez bajo la forma de una FUNCION.

Un programa es una cosa compleja de realizar y por lo tanto es importante que esté bien ESTRUCTURADO y también que sea inteligible para las personas. Si un grupo de sentencias realiza una tarea bien definida, entonces puede estar justificado el aislar estas sentencias formando una función, aunque resulte que sólo se le llame o use una vez.

Hasta ahora hemos visto como resolver un problema planteando un único algoritmo.
Con funciones podemos segmentar un programa en varias partes.

Frente a un problema, planteamos un algoritmo, éste puede constar de pequeños algoritmos.

Una función es un conjunto de instrucciones que resuelven una parte del problema y que puede ser utilizado (llamado) desde diferentes partes de un programa.

Consta de un nombre y parámetros. Con el nombre llamamos a la función, es decir, hacemos referencia a la misma. Los parámetros son valores que se envían y son indispensables para la resolución del mismo. La función realizará alguna operación con los parámetros que le enviamos. Podemos cargar una variable, consultarla, modificarla, imprimirla, etc.

Incluso los programas más sencillos tienen la necesidad de fragmentarse. Las funciones son los únicos tipos de subprogramas que acepta JavaScript. Tienen la siguiente estructura:

function <nombre de función>(argumento1, argumento2, ..., argumento n)
{

```
  <código de la función>
}
```

Debemos buscar un nombre de función que nos indique cuál es su objetivo (Si la función recibe un string y lo centra, tal vez deberíamos llamarla centrarTitulo). Veremos que una función puede variar bastante en su estructura, puede tener o no parámetros, retornar un valor, etc.

Ejemplo: Mostrar un mensaje que se repita 3 veces en la página con el siguiente texto:

'Cuidado'
'Ingrese su documento correctamente'

'Cuidado'
'Ingrese su documento correctamente'

'Cuidado'
'Ingrese su documento correctamente'

La solución sin emplear funciones es:

```
<html>
<head>
</head>
<body>

<script type="text/javascript">
  document.write("Cuidado<br>");
  document.write("Ingrese su documento correctamente<br>");
  document.write("Cuidado<br>");
  document.write("Ingrese su documento correctamente<br>");
  document.write("Cuidado<br>");
  document.write("Ingrese su documento correctamente<br>");
</script>

</body>
</html>
```

Empleando una función:

```
<html>
<head>
</head>
<body>

<script type="text/javascript">
  function mostrarMensaje()
```

```
  {
   document.write("Cuidado<br>");
   document.write("Ingrese su documento correctamente<br>");
  }

  mostrarMensaje();
  mostrarMensaje();
  mostrarMensaje();
</script>

</body>
</html>
```

Recordemos que JavaScript es sensible a mayúsculas y minúsculas. Si fijamos como nombre a la función mostrarTitulo (es decir la segunda palabra con mayúscula) debemos respetar este nombre cuando la llamemos a dicha función.

Es importante notar que para que una función se ejecute debemos llamarla desde fuera por su nombre (en este ejemplo: mostrarMensaje()).

Cada vez que se llama una función se ejecutan todas las líneas contenidas en la misma.
Si no se llama a la función, las instrucciones de la misma nunca se ejecutarán.

A una función la podemos llamar tantas veces como necesitemos. Las funciones nos ahorran escribir código que se repite con frecuencia y permite que nuestro programa sea más entendible.

Funciones con Parametros

Explicaremos con un ejemplo, una función que tiene datos de entrada.

Ejemplo: Confeccionar una función que reciba dos números y muestre en la página los valores comprendidos entre ellos de uno en uno. Cargar por teclado esos dos valores.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">

 function mostrarComprendidos(x1,x2)
 {
  var inicio;
  for(inicio=x1;inicio<=x2;inicio++)
  {
   document.write(inicio+' ');
  }
 }

 var valor1,valor2;
 valor1=prompt('Ingrese valor inferior:','');
 valor1=parseInt(valor1);
 valor2=prompt('Ingrese valor superior:','');
 valor2=parseInt(valor2);
 mostrarComprendidos(valor1,valor2);

</script>
</body>
</html>
```

El programa de JavaScript empieza a ejecutarse donde definimos las variables valor1 y valor2 y no donde se define la función. Luego de cargar los dos valores por teclado se llama a la función mostrarComprendidos y le enviamos las variables valor1 y valor2. Los parámetros x1 y x2 reciben los contenidos de las variables valor1 y valor2.

Es importante notar que a la función la podemos llamar la cantidad de veces que la necesitemos. Los nombres de los parámetros, en este caso se llaman x1 y x2, no necesariamente se deben llamar igual que las variables que le pasamos cuando la llamamos a la función, en este caso le pasamos los valores valor1 y valor2.

Funciones que retornan un valor

Son comunes los casos donde una función, luego de hacer un proceso, retorne un valor.
Ejemplo 1: Confeccionar una función que reciba un valor entero comprendido entre 1 y 5. Luego retornar en castellano el valor recibido.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">

 function convertirCastellano(x)
 {
  if (x==1)
  return "uno";
  else
    if (x==2)
    return "dos";
    else
      if (x==3)
      return "tres";
      else
        if (x==4)
          return "cuatro";
        else
          if (x==5)
            return "cinco";
          else
            return "valor incorrecto";
 }

 var valor;
 valor=prompt("Ingrese un valor entre 1 y 5","");
 valor=parseInt(valor); var r;
 r=convertirCastellano(valor);
 document.write(r);

</script>
</body>
</html>
```

Podemos ver que el valor retornado por una función lo indicamos por medio de la palabra clave return. Cuando se llama a la función, debemos asignar el nombre de la función a una

variable, ya que la misma retorna un valor.

Una función puede tener varios parámetros, pero sólo puede retornar un único valor.
La estructura condicional if de este ejemplo puede ser remplazada por la instrucción switch, la función queda codificada de la siguiente manera:

```
function convertirCastellano(x)
{
 switch (x)
 {
   case 1:return "uno";
   case 2:return "dos";
   case 3:return "tres";
   case 4:return "cuatro";
   case 5:return "cinco";
   default:return "valor incorrecto";
 }
}
```

Esta es una forma más elegante que una serie de if anidados. La instrucción switch analiza el contenido de la variable x con respecto al valor de cada caso. En la situación de ser igual, ejecuta el bloque seguido de los 2 puntos hasta que encuentra la instrucción return o break.