Tema 4. Objetos

- Programación orientada a objetos
- Clases y Objetos
- Funciones constructoras
- *prototype*
- Clases predefinidas
- La clase *Object*
- La clase *Number*
- La clase *Array*
- La clase *String*
- La clase *Date*
- El objeto *Math*

Programación orientada a objetos

- Es un paradigma de programación que define los programas como un conjunto de objetos que se comunican entre sí para realizar tareas.
- Los objetos son entidades que agrupan **estado** (datos) y **comportamiento** (procedimientos o métodos que operan sobre esos datos).
- Características de la OO
  - **Herencia** → Las clases se relacionan mediante la herencia, formando una jerarquía de clases. Los objetos heredan las propiedades y el comportamiento de todas las clases de la jerarquía a la que pertenecen.
  - **Encapsulamiento** → El contenido de la información de un objeto está oculta al mundo exterior. El encapsulamiento protege los datos asociados a un objeto de modificaciones no autorizadas, eliminando efectos secundarios e interacciones.
  - **Polimorfismo** → Capacidad de definir entidades en el programa que puedan tomar más de una forma.

Clases

- Una clase es la descripción de un conjunto de objetos similares.
- La clase define:
  - La interfaz → Protocolo de acceso a los métodos públicos.
  - La estructura → Conjunto de atributos cuyo valor determina el estado del objeto.
  - El comportamiento → La especificación de los métodos públicos y privados.
- La definición de una clase no crea ningún objeto, sino que incorpora un nuevo tipo de datos en el lenguaje OO que se esté utilizando.
- La definición de la clase contendrá las operaciones necesarias para crear(constructor) y destruir(destructor) los objetos de esa clase.
- JavaScript no permite la creación de clases del mismo tipo que otros lenguajes OO como Java o C++. Sin embargo, es posible crear *pseudoclases,* mediante las funciones constructoras y con la propiedad *prototype* de los objetos.

Objetos

- Definición
  - El objeto se crea mediante la palabra reservada *new* seguida del nombre de la clase que se quiere instanciar:
  - Para acceder a los atributos y métodos del objeto, se utiliza el operador '.'
  - *var objeto = new Object();*
- Atributos
  - *objeto.id=1;*
  - *objeto.nom="Mi objeto";*
- Métodos
  - Los métodos se definen asignando funciones al objeto. Podemos crear funciones anónimas para asignarlas a los métodos del objeto.
  - *objeto.showName=function() {*

    alert(this.nom);

    }

Ejemplo 1

```
<html>
<body>
<script type="text/javascript">
libro=new Object();
libro.titulo = "El Quijote";
libro.autor = "Miguel de Cervantes";
libro.tema = "Novela";
libro.mostrar = function(){
alert("El libro " + this.titulo + " de " + this.autor +
   " trata sobre " + this.tema);
}
alert(libro.titulo);
libro.mostrar();
</script>
</body>
</html>
```

Funciones constructoras

- La función constructora define la función para crear un objeto. De modo que esta función define al mismo tiempo:
  - La clase
  - El constructor de la clase
- La función constructora puede definir todos los parámetros que necesita para construir los nuevos objetos y posteriormente utilizar esos parámetros para la inicialización de las propiedades.

```
function Libro(titulo,autor,tema){
    this.titulo = titulo;
    this.autor = autor;
    this.tema = tema;
    this.mostrar = function(){
       alert("El libro " + this.titulo + " de " +
         this.autor + " trata sobre " + this.tema);
    }
}
milibro=new Libro("El Quijote","Cervantes","Novela");
```

prototype

- Al incluir los métodos como funciones dentro de la propia función constructora, hace que para cada objeto creado (instancia de la clase), se crea una copia de cada función definida como método.
- Para resolver este problema, se hace uso de la propiedad *prototype*
  - Es un atributo que tiene todo objeto, puesto que lo hereda de *Object*
  - Es una referencia al prototipo del objeto
  - Todo atributo o método que contenga el prototipo lo heredan los objetos, de manera que no se crea, sino que es compartido por todos los objetos de la clase
  - Se utiliza para definir métodos o variables de clase
- *prototype* también permite modificar las propiedades y métodos de los objetos predefinidos por JavaScript e incluso añadir nuevas propiedades y métodos.

Ejemplo 2

```
<html>
<head>
<script type="text/javascript">
function Libro(titulo,autor,tema){
    this.titulo = titulo;
    this.autor = autor;
    this.tema = tema;
}
Libro.prototype.mostrar = function(){
   alert("El libro " + this.titulo + " de " + this.autor + " trata sobre " +
    this.tema);
}
</script></head>
<body>
<script type="text/javascript">
milibro=new Libro("El Quijote","Cervantes","Novela");
alert(milibro.titulo);
milibro.mostrar();
</script>
</body>
</html>
```

Clases predefinidas (I)

- JavaScript dispone de clases predefinidas que permiten definir objetos nativos para manejar distintos tipos de datos.
- Además, define una clase para cada uno de los tipos de datos primitivos, llamadas tipos de referencia. De esta forma, existen objetos de tipo *Boolean* para las variables booleanas, *Number* para las variables numéricas y *String* para las variables de cadenas de texto.
- Las clases *Boolean*, *Number* y *String* almacenan los mismos valores de los tipos de datos primitivos y añaden propiedades y métodos para manipular sus valores.
- La principal diferencia entre los tipos de datos primitivos y sus clases es que los datos primitivos se manipulan por valor y los tipos de referencia se manipulan por referencia.
- Podemos aplicar propiedades y métodos a los tipos primitivos, ya que en ese caso JavaScript convierte el tipo primitivo al tipo de referencia. Esta acción es transparente al programador.
- De modo que existen pocas diferencias entre el uso de tipos primitivos o referencias.
  - typeof
  - Booleanos

Clases predefinidas (II)

Clase	Descripción
Object	Es la clase base de la que heredan las demás clases
Function	Es la clase de las funciones. En JavaScript toda función es un objeto de esta clase
Boolean	Clase del tipo *Boolean*
Number	Clase del tipo *Number*
Array	Clase para la manipulación de vectores
String	Clase del tipo *String*, para la manipulación de cadenas
Date	Manipulación de fechas
Math	Propiedades y funciones matemáticas. No es una clase, sino un objeto.

La clase Object

- Es la clase base de la cual heredan todos los objetos.

Atributos	Descripción
constructor	Referencia (puntero) a la función creadora del objeto
prototype	Referencia al prototipo del objeto

Métodos	Descripción
hasOwnProperty(propiedad)	Indica si existe la propiedad para el objeto
toString()	Devuelve una representación en cadena del objeto
valueOf()	Devuelve el valor primitivo para el objeto

La clase Number

- Es el tipo de referencia del tipo primitivo numérico.
- Además de las propiedades y métodos heredados de la clase *Object*, posee los indicados en la tabla.
- Al tipo primitivo también se le pueden aplicar los métodos como si fuera un objeto.

Métodos	Descripción
toFixed(ndec)	Fija el número de decimales mostrados
toExponential(ndec)	Devuelve una representación del número en formato exponencial con ndec decimales
toPrecision(ndec)	Devuelve la representación más adecuada del número (decimal o exponencial)

Ejemplo 3

```
<script type="text/javascript">
var n1=new Number(100.25);
var n2=new Number(0.00000000012345);
document.write("<p>toString() --> ",n1.toString(),"</p>");
document.write("<p>valueOf() --> ",n1.valueOf(),"</p>");
document.write("<p>toFixed(4) --> ",n1.toFixed(4),"</p>");
document.write("<p>toExponential(2) --> ",n1.toExponential(2),"</p>");
document.write("<p>toPrecision(2) --> ",n1.toPrecision(),"</p>");
document.write("<p>toPrecision(2) --> ",n2.toPrecision(),"</p>");
var bol=new Boolean(false);
alert(bol&&true);
alert(bol.valueOf()&&true);
</script>
```

La clase Array

- Es la clase para definir variables (objetos) de tipo vector. Los vectores en JavaScript pueden ser heterogéneos, formados por valores de distintos tipos de datos.
- Los vectores pueden aumentar o disminuir su tamaño en tiempo de ejecución.
- Pueden comportarse como pilas o colas, a través de los métodos definidos a tal efecto.
- Definición
  - *var v=new Array()*
  - *var v=new Array(10)*
  - *var v=new Array(5,"vector",3.25,false);*
  - *var v[5,"vector",3.25,false}*
- Asignación i acceso
  - *v[i]=valor; x=v[i];*
- Atributos
  - *length*. Esta propiedad nos dice en cada momento la longitud del array, es decir, cuántos elementos tiene.

Métodos de Array

Métodos	Descripción
join(separador)	Une los elementos de un array en una cadena, separando cada elemento por el separador especificado.
reverse()	Invierte el orden de los elementos del array
sort()	Ordena los elementos del array
concat(elems)	Devuelve un vector con los elementos elems añadidos al final del mismo
slice(ini [,fin])	Devuelve un vector con los elementos desde la posición ini a fin o al final si se omite el 2º argumento
push()	Añade uno o más elemetos al final
pop()	Elimina y devuelve el último elemento del vector
shift()	Elimina y devuelve el primer elemento del vector
unshift()	Añade un elemento al principio del vector

Ejemplo 4

```
<script type="text/javascript">
  var j=new Array(2), i=new Array(1,"hola",3);
  var b=new Array("palabra","letra","amor","color","cariño");
  var c=new Array(false);
  j[0]=new Array(3);  j[1]=new Array();
  j[0][0]=0; j[0][1]=1; j[0][2]=2; j[1][0]=3; j[1][1]=4; j[1][2]=5;
  document.write(c);
  document.write("<p>"+i+"<\p>");
  document.write("<p>j[0][0]="+j[0][0]+"; j[0][1]="+j[0][1]+ ";
    j[0][2]="+j[0][2]+"<br>");
  document.write("j[1][0]="+j[1][0]+"; j[1][1]="+j[1][1]+"; j[1][2]="+j[1][2]+"</p>");
  document.write("<p>i[0]="+i[0]+"; i[1]="+i[1]+"; i[2]="+i[2]+"</p>");
  document.write("<p>toString: "+b.toString()+"</p>");
  document.write("<p>Antes de ordenar: "+b.join(', ')+"</p>");
  document.write("<p>Ordenados: "+b.sort()+"</p>");
  document.write("<p>Ordenados en orden inverso: "+b.reverse()+"</p>");
  b[b.length]="fin";
  alert(b);
</script>
```

La clase String

- Los objetos de la clase *String* son cadenas de texto
- Puede utilizarse el tipo primitivo o el tipo de referencia, ya que JavaScript los convierte según sea necesario de manera automática.
- Definición
  - var s=new String("Hola amigos"); //tipo referencia
  - var s="Hola Amigos"; //tipo primitivo
- Atributos
  - length → Valor numérico que nos indica la longitud en caracteres de la cadena dada.

Métodos de String (I)

Métodos	Descripción
anchor(nombre)	Crea un ancla, asignando al atributo name el valor de nombre.
big()	Muestra la cadena de caracteres con una fuente grande.
blink()	Muestra la cadena de texto con un efecto intermitente
charAt(indice)	Devuelve el carácter situado en la posición especificada por *indice*
fixed()	Muestra la cadena de caracteres con una fuente no proporcional.
fontcolor(color)	Cambia el color con el que se muestra la cadena.
fontsize(tamaño)	Rango de 1 a 7
italics()	Muestra la cadena en cursiva
indexOf(cad,indice)	Devuelve la posición de la primera ocurrencia de *cad* dentro de la cadena actual, a partir de *indice* (o desde el principio).

Métodos de String (II)

Métodos	Descripción
lastIndexOf(cad,indice)	Como antes pero hacia atrás.
link(URL)	Convierte la cadena en un vínculo asignando al atributo *href* el valor de URL.
small()	Muestra la cadena con una fuente pequeña.
split(separador)	Parte la cadena en un array de caracteres.
strike()	Muestra la cadena de caracteres tachada.
sub()	Muestra la cadena con formato de subíndice.
substring(pos1,pos2)	Devuelve la subcadena que comienza en la posición *pos1* + 1 y que finaliza en la posición *pos2*.
sup()	Muestra la cadena con formato de superíndice.
toLowerCase()	Devuelve la cadena en minúsculas.
toUpperCase()	Devuelve la cadena en minúsculas.

Ejemplo 5 (I)

```
var cad = "Hello World";
with(document){
  write("La cadena es: "+cad+"<br>");
  write("Longitud de la cadena: "+cad.length+"<br>");
  write("Haciendola ancla: "+cad.anchor("b")+"<br>");
  write("En grande: "+cad.big()+"<br>");
  write("Parpadea: "+cad.blink()+"<br>");
  write("Caracter 3 es: "+cad.charAt(3)+"<br>");
  write("Fuente FIXED: "+cad.fixed()+"<br>");
  write("De color: "+cad.fontcolor("#FF0000")+"<br>");
  write("De color: "+cad.fontcolor("salmon")+"<br>");
  write("Tamaño 7: "+cad.fontsize(7)+"<br>");
  write("<I>orl</I> esta en la posicion: "+cad.indexOf("orl"));
  write("<br>En cursiva: "+cad.italics()+"<br>");
  write("La primera <I>l</I> esta, empezando a contar por detras,");
```

Ejemplo 5 (II)

```
  write(" en la posicion: "+cad.lastIndexOf("l")+"<br>");
  write("Haciendola enlace:
   "+cad.link("http://www.ifpsmislata.com")+"<br>");
  write("En pequeño: "+cad.small()+"<br>");
  write("Tachada: "+cad.strike()+"<br>");
  write("Subindice: "+cad.sub()+"<br>");
  write("Superindice: "+cad.sup()+"<br>");
  write("Minusculas: "+cad.toLowerCase()+"<br>");
  write("Mayusculas: "+cad.toUpperCase()+"<br>");
  write("Subcadena entre los caracteres 3 y 10: ");
  write(cad.substring(2,10)+"<br>");
  write("Subcadenas resultantes de separar por las <b>o:</b><br>");
  var ja = cad.split("o");
  for(i=0;i<ja.length;i++)
   write(ja[i]+"<br>");
}
```

La clase Date

- La clase *Date* permite representar y manipular valores relacionados con fechas.
- La fecha y hora se almacena como el número de milisegundos que han transcurrido desde el 1 de Enero de 1970 a las o horas. (Inicio de la era Unix)
- El constructor permite establecer sólo una fecha o la fecha y hora a la vez. El formato es (año, mes, día, hora, minuto, segundo).
- Los meses se indican mediante un valor numérico que empieza en el 0 (Enero) y termina en el 11 (Diciembre). Los días del mes se cuentan de forma natural empezando en 1.
- Definición
  - *var data = new Date() //fecha actual*
  - *var data = new Date(año, mes, día, horas, minutos, segundos);*
  - *var data = new Date(año, mes, día)*
  - *var data = new Date(1000000)* //1000000 milisegundos después del inicio la era Unix

Métodos de Date (I)

Métodos	Descripción
getDate()	Devuelve el día del mes actual (1 – 31).
getDay()	Devuelve el día de la semana actual (0 – 6).
getHours()	Devuelve la hora del día actual (0 – 23).
getMinutes()	Devuelve los minutos de la hora actual (0 – 59).
getMonth()	Devuelve el mes del año actual (0 – 11).
getSeconds()	Devuelve los segundos del minuto actual (0 – 59).
getTime()	Devuelve el tiempo transcurrido en milisegundos desde el 1 de enero de 1970 hasta el momento actual.
getFullYear()	Devuelve el año actual con cuatro dígitos

Métodos de Date (II)

Métodos	Descripción
setDate(día_mes)	Pone el día del mes actual.
setDay(día_semana)	Pone el día de la semana actual.
setHours(horas)	Pone la hora del día actual.
setMinutes(minutos)	Pone los minutos de la hora actual.
setMonth(mes)	Pone el mes del año actual.
setSeconds(segundos)	Pone los segundos del minuto actual.
setTime(milisegundos)	Pone la fecha que dista los milisegundos que le pasemos desde el 1 de enero de 1970.
setFullYear(año)	Pone el año actual con cuatro dígitos.
toGMTString()	Devuelve una cadena que usa las convenciones de Internet con la zona horaria GMT.

Ejemplo 6

```
<html>
<head>
<title>Exemple 6</title>
</head>
<body>
<script type="text/javascript">
<!--
data=new Date();
document.write("<h2 align='center'>",data.getDate()," - ",
   data.getMonth()+1," - ",data.getFullYear());
//-->
</script>
</body>
</html>
```

El objeto Math

- Es un objeto instanciado que contiene constantes y funciones matemáticas.
- No se define, ya que está predefinido como objeto
- Atributos
  - *E* → Número 'e', base de los logaritmos naturales (neperianos).
  - *LN2* → Logaritmo neperiano de 2.
  - *LN10* → Logaritmo neperiano de 10.
  - *LOG2E* → Logaritmo en base 2 de e.
  - *LOG10E* → Logaritmo en base 10 de e.
  - *PI* → Número PI.
  - *SQRT1_2* → Raíz cuadrada de 1/2.
  - SQRT2 → Raíz cuadrada de 2.

Métodos de Math

Métodos	Descripción
abs(numero).	valor absoluto.
acos | *asin*| *atan(numero)*	Arcocoseno | arcoseno | arcotangente
atan2(x,y)	Devuelve el ángulo formado por el vector de coordenadas (x,y) con respecto al eje OX.
ceil(numero)	Devuelve el entero redondeado por exceso.
cos | *sin* | *tan(numero)*	Coseno | seno | tangente
exp | *log(numero).*	Exponencial | logaritmo
floor(numero)	Devuelve el entero redondeado por defecto.
max | *min(x,y)*	Máximo | Mínimo de x e y.
pow(base,exp)	Potencia
random().	Devuelve un número aleatorio entre 0 y 1.
round(numero)	Redondea al entero más cercano.
sqrt(numero)	Raíz cuadrada.

Math (II)

```
<html>
<head>
<title>Exemple 7</title>
</head>
<body>
<script type="text/javascript">
n1=parseInt(prompt("Introduce un número",""));
n2=parseInt(prompt("Introduce otro número",""));
alert("Máximo: "+Math.max(n1,n2));
alert("Mínimo: "+Math.min(n1,n2));
document.write("<p>",n1," elevado a ",n2,":
   ",Math.pow(n1,n2)+"</p>");
document.write("<p>El seno de PI/4 es: "+Math.sin(Math.PI/4)+"</p>");
</script>
</body>
</html>
```