Introducción a JavaScript parte II

JUAN CARLOS CONDE RAMÍREZ

WEB-TECHNOLOGIES

Objetivos

- Conocer características avanzadas de JavaScript.
- Entender la forma de implementar funciones que retornan un valor.
- Aplicar conceptos de Programación Orientada a Objetos en JavaScript.
- Conocer las clases predefinidas de JavaScript más utilizadas como: `Date`, `Array`, `Math`, `String`, entre otras.

Funciones que retornan un valor I

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

Ejemplo: Función que retorna un valor

```
<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 = prompt(“Ingresa un valor entre 1 y 5”, “”);
                  valor = parseInt(valor);
                  var r = convertirCastellano(valor);
                  document.write(r);
            </script>
      </body>
</html>
```

Funciones que retornan un valor II

- 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:

Funciones que retornan un valor III

```
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`.

Programación Orientada a Objetos I

- Un objeto es una estructura que contiene tanto las variables (llamadas propiedades) como las funciones que manipulan dichas variables (llamadas métodos).

- A partir de esta estructura se ha creado un nuevo modelo de programación (la programación orientada a objetos) que atribuye a los mismos propiedades como herencia o polimorfismo.

- Como veremos, JavaScript simplifica en algo este modelo y hace una programación híbrida entre la programación estructurada y la programación orientada a objetos.

Programación Orientada a Objetos II

- El modelo de la programación orientada a objetos normal y corriente separa los mismos en dos: clases e instancias (objetos).

- Las primeras son entes más abstractos que definen un conjunto determinado de objetos.

- Las segundas son miembros de una clase, poseyendo las mismas propiedades que la clase a la cual pertenecen.

Programación Orientada a Objetos III

- Propiedades y métodos. Para acceder a los métodos y propiedades de un objeto debemos utilizar la siguiente sintaxis:

```
objeto.propiedad
objeto.metodo(parametros)
```

- **Objetos**: Son todas las cosas con identidad propia.
  - Se relacionan entre si.
  - Poseen características (atributos) y tienen responsabilidades (funciones, métodos) que deben cumplir.
  - Son ejemplares (instancias) de una clase y conocen a la clase a la cual pertenecen.

Programación Orientada a Objetos IV

- **Atributos o propiedades**: Son las características, cualidades distintivas de cada objeto. Deben ser mínimos para poder realizar todas las operaciones que requiere la aplicación.

- Ejemplos de objetos del mundo real:

Casa:
atributos: tamaño, precio, número de habitaciones, etc.;
responsabilidades: comodidad, seguridad, etc.
Mesa:
atributos: altura, largo, ancho, etc.;
responsabilidades: contener elementos.
Ventana:
atributos: tamaño, color, etc.;
responsabilidades: abrirse cerrarse etc

Programación Orientada a Objetos V

- **Responsabilidades o Métodos**: Son las responsabilidades que debe cumplir la clase. El objetivo de un método es ejecutar las actividades que tiene encomendada la clase.

- Es un algoritmo (conjunto de operaciones) que se ejecuta en respuesta a un mensaje; respuestas a mensajes para satisfacer peticiones.

- Un método consiste en el nombre de la operación y sus argumentos. El nombre del método identifica una operación que se ejecuta.

Programación Orientada a Objetos VI

- Un método está determinado por la clase del objeto receptor, todos los objetos de una clase usan el mismo método en respuesta a mensajes similares.

- La interpretación de un mensaje (selección del método ejecutado) depende del receptor y puede variar con distintos receptores, es decir, puede variar de una clase a otra.

- **Clases**: Una clase es un molde para objetos que poseen las mismas características (que pueden recibir los mismos mensajes y responden de la misma manera).

Programación Orientada a Objetos VII

- Una clase es una representación de una idea o concepto. Unidad que encapsula códigos y datos para los métodos (operaciones).

- Todos los ejemplares de una clase se comportan de forma similar (invocan el mismo método) en respuesta a mensajes similares.

- La clase a la cual pertenece un objeto determina el comportamiento del objeto.

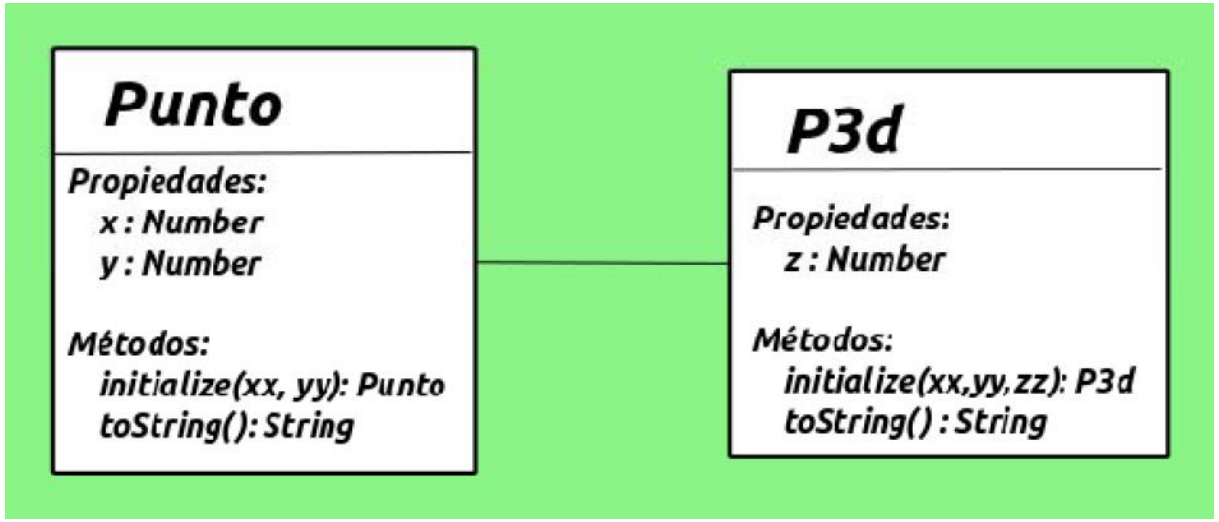
Programación Orientada a Objetos VIII

- Una clase tiene encomendadas actividades que ejecutan los métodos.

- Las clases están definidas por:
  - Atributos (Propiedades),
  - Comportamiento (operaciones o métodos) y
  - Relaciones con otros objetos.

- Una aplicación es un conjunto de objetos de determinadas clases.

Programación Orientada a Objetos IX



Clase `Date`, I

- 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.

Clase `Date`, II

- Así, por ejemplo: `navidad06 = new Date(2006, 11, 25)`

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

```
getYear()
setYear(año)
```

- 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

Clase `Date`, III

`getFullYear()` `setFullYear(año)`

- Realizan la misma función que los anteriores, pero sin tanta complicación, ya que siempre devuelven números con todos sus dígitos.

`getMonth()`
`setMonth(mes)`
`getDate()`
`setDate(dia)`

`getHours()`
`setHours(horas)`
`getMinutes()`
`setMinutes(minutos)`

`getSeconds()`
`setSeconds(segundos)`

- Obtienen y colocan, respectivamente, el mes, día, hora, minuto y segundo de la fecha.

`getDay()`

- Devuelve el día de la semana de la fecha en forma de número que va del 0 (domingo) al 6 (sábado)

Ejemplo: clase `Date`

- 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());
            }
```

Clase `Date`, V

```
            //Llamada a la función
            mostrarFechaHora();
        </SCRIPT>
    </HEAD>
    <BODY>
    </BODY>
</HTML>
```

- En este problema hemos creado un objeto de la clase `Date`.

Clase `Date`, V

- 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 `Array`, I

- Un vector es una estructura de datos que permite almacenar un conjunto de datos.
- Con un único nombre se define un vector y por medio de un subíndice hacemos referencia a cada elemento del mismo (componente).
- Ejemplo: Diseñar un vector para almacenar los cinco sueldos de operarios y luego mostrar el total de gastos en sueldos (cada actividad en una función).

Ejemplo 1: clase `Array`

```
<html>
    <head></head>
    <body>
        <script type="text/javascript">
            function cargar(sueldos)
            {
                var f;
                for(f=0;f<sueldos.length;f++)
                {
                    var v;
                    v=prompt('Ingrese sueldo:','');
                    sueldos[f]=parseInt(v);
                }
            }
            function calcularGastos(sueldos)
            {
                var total=0;
                var f;
```

Ejemplo 1: clase `Array`

```
                for(f=0; f<sueldos.length; f++)
                {
                     total=total+sueldos[f];
                }
                document.write('Listado de sueldos<br>');
                for(f=0;f<sueldos.length;f++)
                {
                     document.write(sueldos[f]+'<br>');
                }
                document.write('Total de gastos en sueldos:'+total);
            }
            var sueldos;
            Sueldos = new Array(5);
            cargar(sueldos);
            calcularGastos(sueldos);
        </script>
    </body>
</html>
```

Clase `Array`, II

- Recordemos que el programa comienza a ejecutarse a partir de las líneas que se encuentran fuera de la funciones:
  - `var sueldos;`
  - `sueldos=new Array(5);`
  - `cargar(sueldos);`
  - `calcularGastos(sueldos);`

- Lo primero, definimos una variable y posteriormente creamos un objeto de la clase `Array`, indicándole que queremos almacenar 5 valores.

- Llamamos a la función cargar enviándole el vector.

Clase `Array`, III

- En la función, a través de un ciclo `for` recorremos las distintas componentes del vector y almacenamos valores enteros que ingresamos por teclado.

- Para conocer el tamaño del vector accedemos a la propiedad `length` de la clase `Array`.

- En la segunda función sumamos todas las componentes del vector, imprimimos en la página los valores y el total de gastos.

Clase `Array`, IV

- Otro ejemplo: Diseñar un vector con elementos de tipo `string`.

  a. Almacenar los meses del año.

  b. En otra función solicitar el ingreso de un número entre 1 y 12.

  c. Mostrar a qué mes corresponde y cuántos días tiene dicho mes.

Ejemplo 2: clase `Array`

```
<html>
     <head></head>
     <body>
          <script type="text/javascript">
               function mostrarFecha(meses,dias)
               {
                    var num;
                    num = prompt('Ingrese número de mes:','');
                    num = parseInt(num);
                    document.write('Corresponde al mes:'+meses[num-1]);
                    document.write('<br>');
                    document.write('Tiene '+dias[num-1]+' días');
               }
               var meses = new Array(12);
               meses[0]='Enero';
               meses[1]='Febrero';
               meses[2]='Marzo';
               meses[3]='Abril';
               meses[4]='Mayo';
               meses[5]='Junio';
               meses[6]='Julio';
               meses[7]='Agosto';
               meses[8]='Septiembre';
               meses[9]='Octubre';
```

Ejemplo 2: clase `Array`

```
            meses[10]='Noviembre';
            meses[11]='Diciembre';

            var días = new Array(12);
            dias[0]=31;
            dias[1]=28;
            dias[2]=31;
            dias[3]=30;
            dias[4]=31;
            dias[5]=30;
            dias[6]=31;
            dias[7]=31;
            dias[8]=30;
            dias[9]=31;
            dias[10]=30;
            dias[11]=31;

            mostrarFecha(meses,dias);
        </script>
    </body>
</html>
```

Clase `Array`, V

- En este problema definimos dos vectores, uno para almacenar los meses y otro los días.

- Decimos que se trata de vectores paralelos porque en la componente cero del vector meses almacenamos el `string 'Enero'` y en el vector `dias`, la cantidad de días del mes de enero.

- Es importante notar que cuando imprimimos, disponemos como subíndice el valor ingresado menos 1, esto debido a que normalmente el operador de nuestro programa carga un valor comprendido entre 1 y 12.

- Recuerda que los vectores comienzan a numerarse a partir de la componente cero.

```
document.write('Corresponde al mes:'+meses[num-1]);
```

Clase `Math`, I

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

Método	Descripción	Expresión 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 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 entre 0 y 1.	Math.random()	0.7345

Clase Math, II

- Ejemplo: Diseñar 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('Ingresa 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 eligiste el '+selec);
        </script>
    </body>
</html>
```

Clase `Math`, III

- 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('Ingresa un valor entre 1 y 10','');`

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

Clase `String,I`

- 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.

Clase `String`, II

- **Propiedad `length`**
  - Retorna la cantidad de caracteres de un objeto `String`.

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

- **Método `charAt(pos)`**
  - Retorna el carácter del índice especificado. Comienzan a numerarse de la posición cero.

```
var nombre='juan';
var caracterPrimero=nombre.charAt(0);
substring(posinicial, posfinal)
```

Clase `String`, III

- **Método `substring(posinicial, posfinal)`**
  - Retorna un String extraída de otro, desde el carácter '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).

Clase `String`, IV

- **Método `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 carácter.

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

Clase `String`, V

- **Método `toUpperCase()`**
  - Convierte todos los caracteres del `String` que invoca el método a mayúsculas:

```
cadena1 = cadena1.toUpperCase();
```

- **Método `toLowerCase()`**
  - Convierte todos los caracteres del `String` que invoca el método a minúsculas:

```
cadena1 = cadena1.toLowerCase();
```

Ejemplo: clase `String`

- Ejemplo: Proporcionar 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('Ingresa 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 carácter es:'+cadena.charAt(0));
            document.write('<br>');
            document.write('Los primeros 3 caracteres son:'+cadena.substring(0,3));
            document.write('<br>');
```

Ejemplo: clase `String`

```
            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>
```