JavaScript
objetos

JavaScript es un lenguaje POO*

- Presenta objetos que se pueden usar

- Un objeto es un tipo especial de dato que tiene
  - Propiedades
  - Métodos

- Permite crear tus propios objetos y tus propios tipos de variables
- (JavaScript Avanzado, no lo vamos a ver, más complejo que en Java)

Wrappers

Cada tipo de dato primitivo (number, string y boolean) tiene su correspondiente clase definida: son las clases Number, String y Boolean

Un wrapper contiene el mismo valor de tipo de dato primitivo pero además define propiedades y métodos los cuales son usados para manipular los datos.

var s = “Hola Mundo”; //variable primitiva de tipo string

var S = new String (“Hola Mundo”); // Objeto de tipo String

Son equivalentes y con ambos se puede hacer lo mismo, salvo que:

typeof(s) = string
typeof(S) = object

Propiedades

Valores que se asocian con los objetos. Ejemplo con el objeto String:

```
<script type="text/javascript">
        var txt="¡Hola Pepito!";
        document.write(txt.length);
</script>
```

Métodos

Acciones que se ejecutan sobre objetos. Ejemplo con el objeto String:

```
<script type="text/javascript">
        var str="¡Hola Pepito!";
        document.write(str.toUpperCase());
</script>
```

Objeto Date

Para trabajar con fechas y horas.
Se inicializa automáticamente en la declaración, con la fecha y hora actuales.

```
var miFecha=new Date()
```

Ejemplos de uso:

```
miFecha.setFullYear(2010,0,14); //fija la fecha a 14 de Enero de 2010
miFecha.setDate(myDate.getDate()+5); //aumenta la fecha guardada en 5 días
```

```
var miFecha=new Date(); //vamos a hacer una comparación de fechas
miFecha.setFullYear(2010,0,14);
var hoy = new Date();
if (miFecha>hoy)
{
        alert("Hoy es antes del 14 de Enero del 2010");
}
else
{
        alert("Hoy es después del 14 de Enero de 2010");
}
```

Objeto Array

Varias posibilidades para declararlas e inicializarlas:

```
var misCoches=new Array();
misCoches[0]="Saab";
misCoches[1]="Volvo";
misCoches[2]="BMW";


var misCoches=new Array(3);
misCoches[0]="Saab";
misCoches[1]="Volvo";
misCoches[2]="BMW";

var misCoches=new Array("Saab","Volvo","BMW");
```

Ejemplo Array

Presentan numerosos métodos para hacer las operaciones más habituales, por ejemplo, ordenamiento (método sort):

```
<script type="text/javascript">
        var arr = new Array(6);
        arr[0] = "Jani";
        arr[1] = "Hege";
        arr[2] = "Stale";
        arr[3] = "Kai Jim";
        arr[4] = "Borge";
        arr[5] = "Tove";
        document.write(arr + "<br />");
        document.write(arr.sort());
</script>
```

Resultado:

Jani,Hege,Stale,Kai Jim,Borge,Tove
Borge,Hege,Jani,Kai Jim,Stale,Tove

Objeto Boolean

Puede tener 2 valores: false o true

De todas las siguientes maneras se inicializa a false:

```
var myBoolean=new Boolean();
var myBoolean=new Boolean(0);
var myBoolean=new Boolean(null);
var myBoolean=new Boolean("");
var myBoolean=new Boolean(false);
var myBoolean=new Boolean(NaN);
```

En otros casos, se inicializará siempre a true:

```
var myBoolean=new Boolean(true);
var myBoolean=new Boolean("true");
var myBoolean=new Boolean("false");
var myBoolean=new Boolean("Richard");
```

JavaScript

Expresiones regulares

Definición de expresiones regulares

Las expresiones regulares son una forma de describir cadenas de caracteres que sirven para comparaciones y reemplazos complejos.

Están basadas en las expresiones regulares de Perl y se representan por el objeto RegExp (REGular EXPresion).

var patron = /pato/;

var patron = new RegExp(“pato”);

Presenta tres métodos: test(), exec() y compile()

```
var patt1=new RegExp("e");
document.write(patt1.exec("The best things in life are free"));
//como hay una “e” en la cadena, la salida de la sentencia será 'e'.
```

```
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
//como hay una “e” en la cadena, la salida de la sentencia será true
```

Objeto RegExp

Tiene varias propiedades: ignoreCase, lastMatch...

Múltiples modificadores:

^ Indicar coincidencia al principio de la cadena

$ Indicar coincidencia al final de la cadena.

var cadena = /alumn[ao]/; //buscaría tanto alumno como alumna
var cadena = /alumno[0-9]/; //bucaría alumno0, alumno1, … hasta alumno9
var cadena = /alumno[0-9a-zA]/; //Formaría cadenas como alumno5, alumnor o alumnoA

+ indica que lo que tiene a su izquierda puede estar 1 o mas veces.
***** indica que puede estar 0 o mas veces
? indica opcionalidad, es decir, lo que tenemos a la izquierda puede aparecer 0 o 1 vez.
{3} Indicar exactamente el número de veces que puede aparecer (3 en este caso).

\d un dígito. Equivale a [0-9]
\D cualquier caracter que no sea un dígito.
\w Cualquier carácter alfanumérico. Equivalente a [a-zA-Z0-9_].
\W cualquier carácter no alfanumérico
\s espacio
\t tabulador

Algunos ejemplos útiles

Número de teléfono nacional (sin espacios)

Fecha con formato dd/mm/yyyy

Código postal

e-mail

Algunos ejemplos útiles

Número de teléfono nacional (sin espacios)

/^\d{9}$/
/^[0-9]{9}$/
Comienza (^) por una cifra numérica (\d) de la que habrá 9 ocurrencias ({9}) y aquí acabará la cadena ($).

Fecha con formato dd/mm/yyyy
/^\d{2}\/\d{2}\/\d{4}$/

Código postal
^\d{5}$/

e-mail
/^(.+\@.+\..+)$/
Comienza (^) por caracteres cualesquiera que no sean salto de línea (.) de los que habrá al menos una ocurrencia(+).
Después el carácter arroba (\@), seguido de al menos un carácter que no podrá ser el salto de línea (.+), después viene el carácter punto (\.), seguido de al menos un carácter donde ninguno podrá ser el salto de línea (.+), y aquí acabará la cadena ($).

Algunos objetos:

Window	Representa la ventana del navegador.
Navigator	Contiene información sobre el navegador Web
Screen	Contiene información sobre la pantalla del ordenador
History	Contiene información sobre las URLs visitadas
Location	Contiene información sobre la URL actual en el navegador

Ejemplo de uso:

```
<html>
        <head>
                <script type="text/javascript">
                        function printpage()
                        {
                                window.print()
                        }
                </script>
        </head>
        <body>
                <input type="button" value="Print this page" onclick="printpage()" />
        </body>
</html>
```

JavaScript

DOM

HTML DOM

Es un estándar de la W3C.

Es una abreviatura de HTML Document Object Model.

Consiste en un conjunto de objetos para los html y una forma estándar para acceder y manipular las páginas (documentos html).

HTML DOM no depende de ninguna plataforma ni lenguaje de programación determinado. DOM es una utilidad disponible también para otros lenguajes como Java y PHP.

Alguno de los objetos son Document, Image, Body, Form, Frame, Iframe, etc.

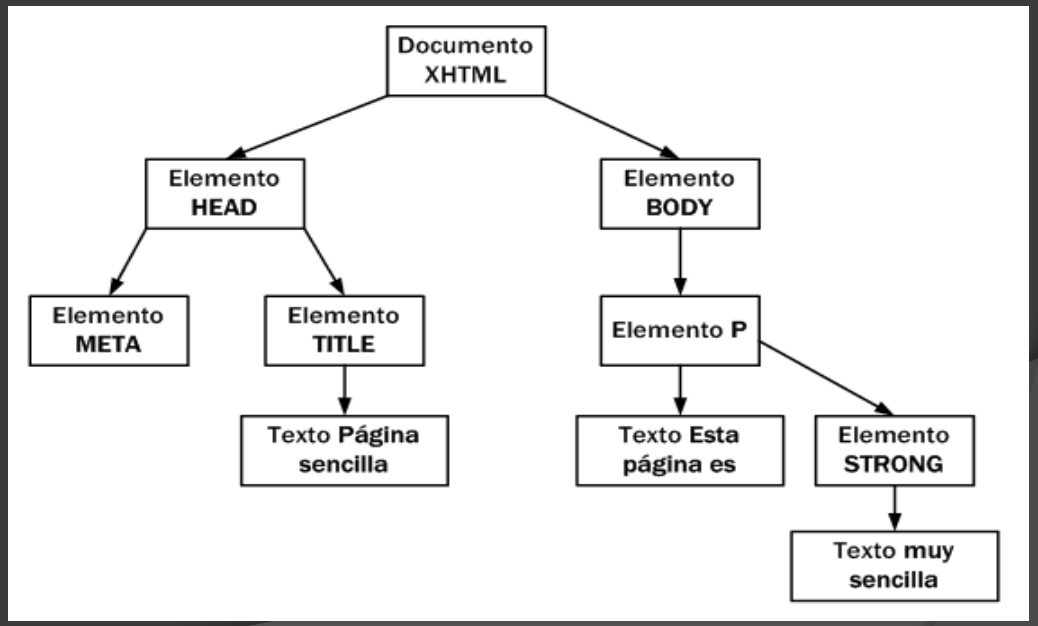
Cada objeto puede tener una colección de objetos asociados, unas propiedades y unos métodos.

Ejemplo: Objeto Document

```
<html>
        <body>
                <img border="0" src="imagen1.gif" width="48" height="48">
                <br />
                <img border="0" src="imagen2.gif" width="107" height="98">
                <br /><br />
                <script type="text/javascript">
                        document.write("Este documento contiene: ")
                        document.write(document.images.length + " imágenes.")
                </script>
</body>
</html>
```

Arbol de nodos

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
                <title>Página sencilla</title>
        </head>
        <body>
                <p>Esta página es <strong>muy sencilla</strong></p>
        </body>
</html>
```



Acceso a los nodos

- Acceso directo
- A través de sus nodos padre

```
//buscaremos todos los parrafos a partir del elemento document
var parrafos = document.getElementsByTagName("p");

var primerParrafo = parrafos[0];

for(var i=0; i<parrafos.length; i++) {
  var parrafo = parrafos[i];
}

var parrafoEspecial = document.getElementsByName("especial");
var cabecera = document.getElementById("cabecera");
```

Creación de nodos

```
// Crear nodo de tipo Element
var parrafo = document.createElement("p");
// Crear nodo de tipo Text
var contenido = document.createTextNode("Hola Mundo!");
// Añadir el nodo Text como hijo del nodo Element
parrafo.appendChild(contenido);
// Añadir el nodo Element como hijo de la pagina
document.body.appendChild(parrafo);
```

Eliminación de nodos

```
var parrafo = document.getElementById("provisional");
parrafo.parentNode.removeChild(parrafo);

<p id="provisional">...</p>
```

Acceso directo atributos HTML

```
var enlace = document.getElementById("enlace");
alert(enlace.href); // muestra http://www...com

<a id="enlace" href="http://www...com">Enlace</a>
```

Para obtener las propiedades CSS, mediante el atributo style:

```
var imagen = document.getElementById("imagen");
alert(imagen.style.margin);

<img id="imagen" style="margin:0; border:0;" src="logo.png" />
```

Si el nombre de una propiedad CSS es compuesto, se accede a su valor modificando ligeramente su nombre:
font-weight se transforma en fontWeight
line-height se transforma en lineHeight
...

Ejemplo:

```
<html>
	<head>
		<script type="text/javascript">
			function getValue()
			{
				var x=document.getElementById("Titulo");
				alert(x.innerHTML);
			}
		</script>
	</head>
	<body>
		<h1 id="Titulo" onclick="getValue()">Este es el título</h1>
		<p>Pulsa en el título para que te salga una alert box con su valor.</p>
	</body>
</html>
```

Ejemplo: Crear un reloj por pantalla

```
<html>
	<head>
		<script type="text/javascript">
			function startTime()
			{
				var today=new Date();
				var h=today.getHours();
				var m=today.getMinutes();
				var s=today.getSeconds();
				m=checkTime(m);// add a zero in front of numbers<10
				s=checkTime(s);// add a zero in front of numbers<10
				document.getElementById('txt').innerHTML=h+":"+m+":"+s;
				t=setTimeout('startTime()',500);
			}
			function checkTime(i)
			{
				if (i<10)
				{
					i="0" + i;
				}
				return i;
			}
		</script>
	</head>
	<body onload="startTime()">
		<p> prueba </p>
		<div id="txt"></div>
	</body>
</html>
```

JavaScript
Eventos

Modelo básico de eventos

Cada elemento o etiqueta XHTML define su propia lista de posibles eventos que se le pueden asignar.

Un mismo tipo de evento puede estar definido para varios elementos XHTML diferentes.

Un mismo elemento XHTML puede tener asociados varios eventos diferentes.

Ejemplos:

```
<input type="button" value="Pinchame y verás" onclick="alert('Gracias por pinchar');" />
```

JavaScript permite asignar una función a cada uno de los eventos. De esta forma, cuando se produce cualquier evento, JavaScript ejecuta su función asociada.
Este tipo de funciones se denominan "**event handlers**" en inglés y suelen traducirse por "**manejadores de eventos**":

```
function muestraMensaje() {
  alert('Gracias por pinchar');
}
<input type="button" value="Pinchame y verás" onclick="muestraMensaje()" />
```

Tipos de eventos

El nombre de cada evento se construye mediante el prefijo on, seguido del nombre en inglés de la acción asociada al evento.

Los eventos más utilizados en las aplicaciones web tradicionales son onload para esperar a que se cargue la página por completo,

los eventos onclick, onmouseover, onmouseout para controlar el ratón y onsubmit para controlar el envío de los formularios.

Las acciones típicas que realiza un usuario en una página web pueden dar lugar a una sucesión de eventos.
Al pulsar por ejemplo sobre un botón de tipo <input type="submit"> se desencadenan los eventos onmousedown, onclick, onmouseup y onsubmit de forma consecutiva.

Existen varias formas diferentes de indicar los manejadores de eventos (event handlers)

- Manejadores como atributos de los elementos XHTML.
- Manejadores como funciones JavaScript externas.
- Manejadores "semánticos".

Uso de la variable this

Para referirse al elemento XHTML que ha provocado el evento:

```
<div id="contenidos" style="width:150px; height:60px; border:thin solid silver">
  Sección de contenidos...
</div>
```

Modificación de bordes al pasar el ratón por encima:
Sin this:

```
<div id="contenidos" style="width:150px; height:60px; border:thin solid silver"
onmouseover="document.getElementById('contenidos').style.borderColor='black';“
onmouseout="document.getElementById('contenidos').style.borderColor='silver';">
  Sección de contenidos...
</div>
```

Con this:

```
<div id="contenidos" style="width:150px; height:60px; border:thin solid silver"
onmouseover="this.style.borderColor='black';"
onmouseout="this.style.borderColor='silver';">
  Sección de contenidos...
</div>
```

Ejemplo uso de this

Al llamar a una función, debemos pasar this como parámetro.
Varios case responden igual debido a las diferentes respuestas de los navegadores.

```
function resalta(elemento) {
  switch(elemento.style.borderColor) {
    case 'silver':
    case 'silver silver silver silver':
    case '#c0c0c0':
      elemento.style.borderColor = 'black';
      break;
    case 'black':
    case 'black black black black':
    case '#000000':
      elemento.style.borderColor = 'silver';
      break;
  }
}

<div style="width:150px; height:60px; border:thin solid silver"
onmouseover="resalta(this)" onmouseout="resalta(this)">
  Sección de contenidos...
</div>
```

Separación de HTML y JavaScript

Al igual que se intentan separar los contenidos (código html) y el diseño (css), se recomienda separar los contenidos, de su comportamiento (JavaScript).

Ejemplo:

```
<input id="pinchable" type="button" value="Pinchame y verás"
onclick="alert('Gracias por pinchar');" />
```

Se puede transformar en:

```
// Función externa
function muestraMensaje() {
  alert('Gracias por pinchar');
}
// Asignar la función externa al elemento, sin parentesis!!!!!
// se ejecuta una vez que esté cargado todo el DOM (evento onload)
//Si se añaden los paréntesis después del nombre de la función, en realidad se
 está //ejecutando la función y guardando el valor devuelto por la función en la propiedad
//onclick de elemento.
window.onload = function() {
            document.getElementById("pinchable").onclick = muestraMensaje;
}
// Elemento XHTML
<input id="pinchable" type="button" value="Pinchame y verás" />
```

Técnica de los manejadores semánticos

1. Asignar un identificador único al elemento XHTML mediante el atributo id.
2. Crear una función de JavaScript encargada de manejar el evento.
3. Asignar la función externa al evento correspondiente en el elemento deseado.

Dentro de las funciones externas asignadas sí que se puede utilizar la variable this para referirse al elemento que provoca el evento.

Inconveniente: la página se debe cargar completamente antes de que se puedan utilizar las funciones DOM que asignan los manejadores a los elementos XHTML.
¡Para ello debemos utilizar el evento onload!

```
function resalta() {
  // Código JavaScript
}

window.onload = function() {
  var formulario = document.getElementById("formulario");
  var camposInput = formulario.getElementsByTagName("input");

for(var i=0; i<camposInput.length; i++) {
    if(camposInput[i].type == "text") {
      camposInput[i].onclick = resalta;
    }
  }
}
```

JavaScript
Formularios

DOM y acceso al formulario

Mediante el array forms del document y su array de elementos:
document.forms[0]; //primer formulario de la página web
document.forms[0].elements[0]; //elemento 0 del formulario
document.forms[0].elements[document.forms[0].elements.length-1]; //último elemento

Inconvenientes:
- Sintaxis a veces poco concisa
- Alteración del orden de los formularios en un entorno tan cambiante como la Web.

Mejor mediante name o id:

```
var formularioPrincipal = document.formulario;
var primerElemento = document.formulario.elemento;
 <form name="formulario">
        <input type="text" name="elemento" />
</form>

var formularioPrincipal = document.getElementById("formulario");
var primerElemento = document.getElementById("elemento");
<form name="formulario" id="formulario" >
        <input type="text" name="elemento" id="elemento" />
</form>
```

Propiedades y eventos de los formularios

type: Indica el tipo de elemento del que se trata:
text, button, checkbox, select-one, select-multiple, textarea

value: obtiene el texto que se muestra en un botón o en las cajas de texto (text o textarea)

name: se obtiene el valor del atributo name.

form: para referirse al formulario de un elemento:
document.getElementById("id_del_elemento").form

Eventos más habituales:

onclick

onchange: al cambiar el valor de un elemento y perder el foco

onfocus – onblur: cuando el elemento tiene o pierde el foco.

Obtención de datos del formulario:

Cuadros de texto:

```
<input type="text" id="texto" />
var valor = document.getElementById("texto").value;
<textarea id="parrafo"></textarea>
var valor = document.getElementById("parrafo").value;
```

Radiobutton:

```
<input type="radio" value="si" name="pregunta" id="pregunta_si"/> SI
<input type="radio" value="no" name="pregunta" id="pregunta_no"/> NO
<input type="radio" value="nsnc" name="pregunta" id="pregunta_nsnc"/> NS/NC
var elementos = document.getElementsByName("pregunta");
 for(var i=0; i<elementos.length; i++) {
  alert(" Elemento: " + elementos[i].value + "\n Seleccionado: " + elementos[i].checked);
}
```

Checkbox

```
<input type="checkbox" value="condiciones" name="condiciones" id="condiciones"/> He leído y acepto las condiciones
<input type="checkbox" value="privacidad" name="privacidad" id="privacidad"/> He leído la política de privacidad
var elemento = document.getElementById("condiciones");
alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);
elemento = document.getElementById("privacidad");
alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);
```

Obtención de datos del formulario:

Listas select:

```
<select id="opciones" name="opciones">
        <option value="1">Primer valor</option>
        <option value="2">Segundo valor</option>
        <option value="3">Tercer valor</option>
        <option value="4">Cuarto valor</option>
</select>

// Obtener la referencia a la lista
var lista = document.getElementById("opciones");

// Obtener el índice de la opción que se ha seleccionado
var indiceSeleccionado = lista.selectedIndex;
// Con el índice y el array "options", obtener la opción seleccionada
var opcionSeleccionada = lista.options[indiceSeleccionado]; //se podría añadir aquí el .text o .value!!!

// Obtener el valor y el texto de la opción seleccionada
var textoSeleccionado = opcionSeleccionada.text;
var valorSeleccionado = opcionSeleccionada.value;

alert("Opción seleccionada: " + textoSeleccionado + "\n Valor de la opción: " + valorSeleccionado);
```

Otras operaciones:

Asignar el foco en el formulario:
Comprobamos que haya formularios y asignamos el foco al primer elemento que no sea de tipo hidden.

```
if(document.forms.length > 0) {
  for(var i=0; i < document.forms[0].elements.length; i++) {
    var campo = document.forms[0].elements[i];
    if(campo.type != "hidden") {
      campo.focus();
      break;
    }
  }
}
```

Evitar envío duplicado del formulario:
Mediante un botón de tipo <input type="button" />, ya que el código JavaScript mostrado no funciona correctamente con un botón de tipo <input type="submit" />.

```
<form id="formulario" action="#">
  ...
  <input type="button" value="Enviar" onclick="this.disabled=true; this.value='Enviando...';
this.form.submit()" />
</form>
```

Otras operaciones

Limitar el número de caracteres de un textarea

Algunos eventos (como onkeypress, onclick y onsubmit) se puede evitar su comportamiento normal si se devuelve el valor false.

textarea no tiene la propiedad maxlength

```
function limita(maximoCaracteres) {
  var elemento = document.getElementById("texto");
  if(elemento.value.length >= maximoCaracteres ) {
    return false;
  }
  else {
    return true;
  }
}
```

```
<textarea id="texto" onkeypress="return limita(100);"></textarea>
```

Otras operaciones

Validar que los elementos de un checkbox estén seleccionados:

```
formulario = document.getElementById("formulario");
for(var i=0; i<formulario.elements.length; i++) {
  var elemento = formulario.elements[i];
  if(elemento.type == "checkbox") {
    if(!elemento.checked) {
      return false;
    }
  }
}
```

Comprobar que se haya seleccionado al menos una opción de un radiobutton:

```
opciones = document.getElementsByName("opciones");

var seleccionado = false;
for(var i=0; i<opciones.length; i++) {
  if(opciones[i].checked) {
    seleccionado = true;
    break;
  }
}

if(!seleccionado) {
  return false;
}
```