Ciclo Formativo de Grado Superior de Administración de Sistemas Informáticos en red

Módulo Profesional: LMSGI

Unidad de Trabajo 6.- Lenguaje Javascript - DOM

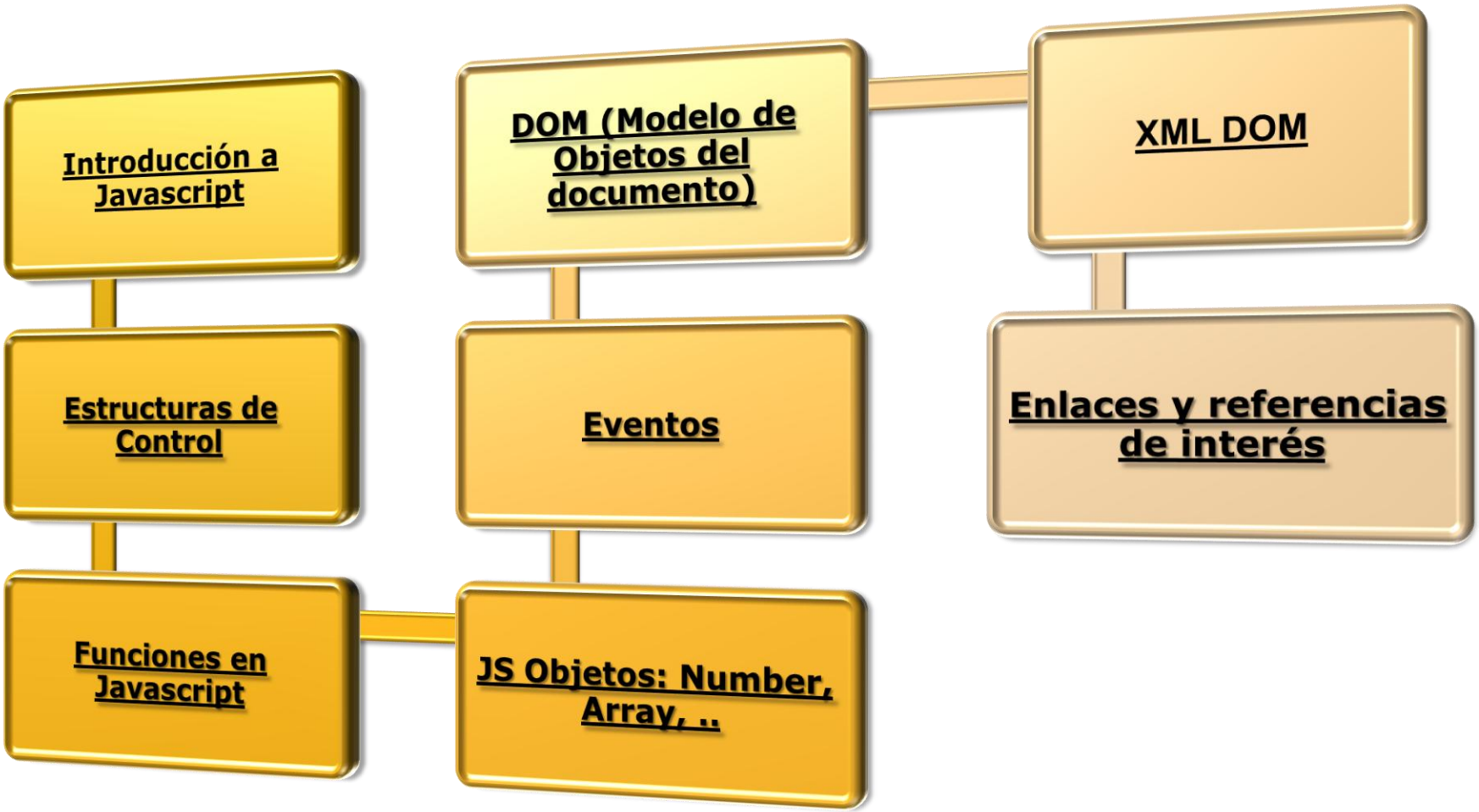
Departamento de Informática y Comunicación
IES San Juan Bosco (Lorca-Murcia)
Profesor: Juan Antonio López Quesada

Abstract/Resumen

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas.

Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

Índice de Contenidos



Introducción a Javascript

- JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas.
- Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.
- Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.
- A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java. Legalmente, JavaScript es una marca registrada de la empresa Sun Microsystems, como se puede ver en http://www.sun.com/suntrademarks/.

Introducción a Javascript

Incluir JavaScript en el mismo documento XHTML

```
<script type="text/javascript"> alert("Un mensaje de prueba"); </script>
```

Definir JavaScript en un archivo externo

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de código JavaScript en el propio documento</title>
<script type="text/javascript" src="/js/codigo.js"></script>
</head>
```

Incluir JavaScript en los elementos XHTML

```
<p onclick="alert('Un mensaje de prueba')">Un párrafo de texto.</p>
```

Introducción a Javascript

- Algunos navegadores no disponen de soporte completo de JavaScript, otros navegadores permiten bloquearlo parcialmente e incluso algunos usuarios bloquean completamente el uso de JavaScript porque creen que así navegan de forma más segura.
- En estos casos, es habitual que si la página web requiere JavaScript para su correcto funcionamiento, se incluya un mensaje de aviso al usuario indicándole que debería activar JavaScript para disfrutar completamente de la página.
- El lenguaje HTML define la etiqueta <noscript> para mostrar un mensaje al usuario cuando su navegador no puede ejecutar JavaScript. El siguiente código muestra un ejemplo del uso de la etiqueta <noscript>:

```
<body>
<noscript>
<p>Bienvenido a Mi Sitio</p> <p>La página que estás viendo requiere para su funcionamiento el uso de JavaScript. Si lo has deshabilitado intencionadamente, por favor vuelve a activarlo.</p>
</noscript>
</body>
```

Etiqueta noscript.....

Introducción a Javascript

- No se tienen en cuenta los espacios en blanco y las nuevas líneas
- Se distinguen las mayúsculas y minúsculas:
- No se define el tipo de las variables: al crear una variable, no es necesario indicar el tipo de dato que almacenará. De esta forma, una misma variable puede almacenar diferentes tipos de datos durante la ejecución del script.
- No es necesario terminar cada sentencia con el carácter de punto y coma (;)
- Se pueden incluir comentarios.

Introducción a Javascript

Ejemplo de comentario de una sola línea:

```
// a continuación se muestra un mensaje
alert("mensaje de prueba");
```

Los comentarios de una sola línea se definen añadiendo dos barras oblicuas (//) al principio de la línea.

Ejemplo de comentario de varias líneas:

```
/* Los comentarios de varias líneas son muy útiles cuando se necesita incluir bastante
   información en los comentarios */
alert("mensaje de prueba");
```

Los comentarios multilínea se definen encerrando el texto del comentario entre los símbolos /* y */.

Comentarios.....

Introducción a Javascript

```
<!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=iso-8859-1" />
<title>El primer script</title>
<script type="text/javascript">
alert("Hola Mundo!");
</script>
</head>
<body>
<p>Esta página contiene el primer script</p>
</body>
</html>
```

El primer script.....

Introducción a Javascript

alert()

Crea una caja de diálogo con un icono de peligro amarillo, un botón 'Aceptar' y un texto definido por el parámetro enviado a la función.

confirm()

Crea una caja de confirmación con un icono de interrogación, botones Aceptar y Cancelar y un texto definido por el parámetro enviado a la función. Devuelve 1 cuando el usuario abandona el diálogo pulsando Aceptar y 0 si lo hace pulsando Cancelar o el aspa de cerrar.

prompt()

Muestra un diálogo de campo de formulario con botones Aceptar y Cancelar, un texto definido por el primer parámetro enviado a la función y un input de texto con valor predeterminado definido por el segundo parámetro. La función devuelve el valor insertado en el campo de formulario si el usuario pulsa en Aceptar o null si pulsa Cancelar o el aspa de cerrar.

Funciones de Interés.....

Introducción a Javascript

- Una variable es un elemento que se emplea para almacenar y hacer referencia a otro valor. Gracias a las variables es posible crear "programas genéricos", es decir, programas que funcionan siempre igual independientemente de los valores concretos utilizados.

- Las variables en JavaScript se crean mediante la palabra reservada var. De esta forma, el ejemplo anterior se puede realizar en JavaScript de la siguiente manera:

```
var numero_1 = 3;
var numero_2 = 1;
var resultado = numero_1 + numero_2;
```

- La palabra reservada var solamente se debe indicar al definir por primera vez la variable, lo que se denomina declarar una variable.
- Si cuando se declara una variable se le asigna también un valor, se dice que la variable ha sido inicializada. En JavaScript no es obligatorio inicializar las variables, ya que se pueden declarar por una parte y asignarles un valor posteriormente.

Variables.....

Introducción a Javascript

- Una de las características más sorprendentes de JavaSript para los programadores habituados a otros lenguajes de programación es que tampoco es necesario declarar las variables. En otras palabras, se pueden utilizar variables que no se han definido anteriormente mediante la palabra reservada var.

- En cualquier caso, se recomienda declarar todas las variables que se vayan a utilizar.

- El nombre de una variable también se conoce como identificador y debe cumplir las siguientes normas:

  a) *Sólo puede estar formado por letras, números y los símbolos $ (dólar) y _ (guión bajo).*
  b) *El primer carácter no puede ser un número.*

Variables.....

Introducción a Javascript

Numéricas

```
var iva = 16; // variable tipo entero
var total = 234.65; // variable tipo decimal
```

Cadenas de texto

```
var mensaje = "Bienvenido a nuestro sitio web";
var nombreProducto = 'Producto ABC';
var letraSeleccionada = 'c';

/* El contenido de texto1 tiene comillas simples, por lo que se encierra con comillas dobles */
var texto1 = "Una frase con 'comillas simples' dentro";
/* El contenido de texto2 tiene comillas dobles, por lo que se encierra con comillas simples */
var texto2 = 'Una frase con "comillas dobles" dentro';
```

Tipos de variables.....

Introducción a Javascript

- No obstante, a veces las cadenas de texto contienen tanto comillas simples como dobles. Además, existen otros caracteres que son difíciles de incluir en una variable de texto (tabulador, ENTER, etc.) Para resolver estos problemas, JavaScript define un mecanismo para incluir de forma sencilla caracteres especiales y problemáticos dentro de una cadena de texto.

Si se quiere incluir...	Se debe incluir...
Una nueva línea	\n
Un tabulador	\t
Una comilla simple	\'
Una comilla doble	\"
Una barra inclinada	\\

Introducción a Javascript

Booleanos

- Las variables de tipo boolean o booleano también se conocen con el nombre de variables de tipo lógico. Aunque para entender realmente su utilidad se debe estudiar la programación avanzada con JavaScript del siguiente capítulo, su funcionamiento básico es muy sencillo.

- Una variable de tipo boolean almacena un tipo especial de valor que solamente puede tomar dos valores: true (verdadero) o false (falso). No se puede utilizar para almacenar números y tampoco permite guardar cadenas de texto.

- Los únicos valores que pueden almacenar estas variables son true y false, por lo que no pueden utilizarse los valores verdadero y falso. A continuación se muestra un par de variables de tipo booleano:

```
var clienteRegistrado = false;
var ivaIncluido = true;
```

Introducción a Javascript

Arrays

- Un array es una colección de variables, que pueden ser todas del mismo tipo o cada una de un tipo diferente.

var dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"];

var nombre_array = [valor1, valor2, ..., valorN];

- Una vez definido un array, es muy sencillo acceder a cada uno de sus elementos. Cada elemento se accede indicando su posición dentro del array. La única complicación, que es responsable de muchos errores cuando se empieza a programar, es que las posiciones de los elementos empiezan a contarse en el 0 y no en el 1:

var diaSeleccionado = dias[0]; // diaSeleccionado = "Lunes"

var otroDia = dias[5]; // otroDia = "Sábado"

Tipos de variables.....

Introducción a Javascript

Operadores Aritméticos

Operator	Description	Example	Result	
+	Addition	x=y+2	x=7	y=5
-	Subtraction	x=y-2	x=3	y=5
*	Multiplication	x=y*2	x=10	y=5
/	Division	x=y/2	x=2.5	y=5
%	Modulus (division remainder)	x=y%2	x=1	y=5
++	Increment	x=++y	x=6	y=6
		x=y++	x=5	y=6
--	Decrement	x=--y	x=4	y=4
		x=y--	x=5	y=4

Operadores de Asignación

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

Operadores.....

Introducción a Javascript

Operador de Concatenación

Operator	Example	Result
=		
	txt1="What a very"; txt2="nice day"; txt3=txt1+txt2;	**text3="What avery nice day"**
	txt1="What a very "; txt2="nice day"; txt3=txt2+txt1;	**text3="nice dayWhat avery"**
	txt1="What a very"; txt2="nice day"; txt3=txt1+" "+txt2;	**text3="What avery nice day"**
	x=5+"5";	**x="55"**

Operadores Relacionales

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

Operadores.....

Introducción a Javascript

Operadores Lógicos

Operator	Description (x=6 and y=3)	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

Tabla de Verdad de los operadores lógicos

vari1	varia2	vari1 && varia	vari1 || vari2	! vari1
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Operadores.....

Ejercicios.

Estructuras de Control

Se emplea para tomar decisiones en función de una condición. Su definición formal es:

```
if(condicion)
{
...
 }
```

Si la condición se cumple (es decir, si su valor es true) se ejecutan todas las instrucciones que se encuentran dentro de {...}. Si la condición no se cumple (es decir, si su valor es false) no se ejecuta ninguna instrucción contenida en {...} y el programa continúa ejecutando el resto de instrucciones del script.

```
if(edad>=10 && edad<=20)
{
   alert("La edad pertenece al intervalo [10,20]");
 }
```

Estructura if…..

Estructuras de Control

```
if(condicion) {
...
} else
{ ...
}
```

Si la condición se cumple (es decir, si su valor es true) se ejecutan todas las instrucciones que se encuentran dentro del if(). Si la condición no se cumple (es decir, si su valor es false) se ejecutan todas las instrucciones contenidas en else { }.

```
if(edad>=10 && edad<=20)
{
    alert("La edad pertenece al intervalo [10,20]");
 } else
{
    alert("La edad está fuera del intervalo [10,20]")
}
```

Estructura if…..

Estructuras de Control

La estructura if...else se puede encadenar para realizar varias comprobaciones seguidas:

```
if(edad < 12)
{
    alert("Todavía eres muy pequeño");
} else if(edad < 19)
{
    alert("Eres un adolescente");
} else if(edad < 35)
{
    alert("Aun sigues siendo joven");
} else
{
    alert("Piensa en cuidarte un poco más");
}
```

No es obligatorio que la combinación de estructuras if...else acabe con la instrucción else, ya que puede terminar con una instrucción de tipo else if().

Estructuras de Control

La estructura for permite realizar este tipo de repeticiones (también llamadas bucles) de una forma muy sencilla.

```
for(inicializacion; condicion; actualizacion)
{
 ...
 }
```

La idea del funcionamiento de un bucle for es la siguiente: *"mientras la condición indicada se siga cumpliendo, repite la ejecución de las instrucciones definidas dentro del for. Además, después de cada repetición, actualiza el valor de las variables que se utilizan en la condición"*.

La "inicialización" es la zona en la que se establece los valores iniciales de las variables que controlan la repetición.

La "condición" es el único elemento que decide si continua o se detiene la repetición.

La "actualización" es el nuevo valor que se asigna después de cada repetición a las variables que controlan la repetición.

Estructura for.....

Estructuras de Control

Una estructura de control derivada de for es la estructura for...in. Su definición exacta implica el uso de objetos, aunque ahora solamente se va a presentar la estructura for...in adaptada a su uso en arrays. Su definición formal adaptada a los arrays es:

```
for(indice in array) {
...
 }
```

```
var dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"];
for(i in dias) {
alert(dias[i]);
}
```

La variable que se indica como indice es la que se puede utilizar dentro del bucle for...in para acceder a los elementos del array. De esta forma, en la primera repetición del bucle la variable i vale 0 y en la última vale 6.

Estructura for...in.....

Estructuras de Control

La estructura while permite crear bucles que se ejecutan ninguna o más veces, dependiendo de la condición indicada. Su definición formal es:

```
while(condicion)
{ ...
}
```

El funcionamiento del bucle while se resume en: "mientras se cumpla la condición indicada, se *repite indefinidamente las instrucciones incluidas dentro del bucle*".

Si la condición no se cumple ni siquiera la primera vez, el bucle no se ejecuta. Si la condición se cumple, se ejecutan las instrucciones una vez y se vuelve a comprobar la condición. Si se sigue cumpliendo la condición, se vuelve a ejecutar el bucle y así se continúa hasta que la condición no se cumpla.

Estructuras de Control

El bucle de tipo do...while es muy similar al bucle while, salvo que en este caso siempre se ejecutan las instrucciones del bucle al menos la primera vez. Su definición formal es:

```
do
{
...
} while(condicion);
```

De esta forma, como la condición se comprueba después de cada repetición, la primera vez siempre se ejecutan las instrucciones del bucle. Es importante no olvidar que después del while() se debe añadir el carácter ; (al contrario de lo que sucede con el bucle while simple).

Estructura do...while.....

Estructuras de Control

La estructura if...else se puede utilizar para realizar comprobaciones múltiples y tomar decisiones complejas. Sin embargo, si todas las condiciones dependen siempre de la misma variable, el código JavaScript resultante es demasiado redundante:

```
if(numero == 5)
 { ...
} else if(numero == 8)
{ ...
} else if(numero == 20)
{ ...
} else
{ ...
}
```

```
switch(variable)
{
case valor_1: ...
                break;
case valor_2: ...
                break;
..
case valor_n: ...
                break;
[default: ...
            break;]
 }
```

Estructura switch.....

Ejercicios

Leer dos enteros y visualizar el mayor. Solución

Leer un número y mostrar si dicho número es o no es par.

Leer un número y mostrar su tabla de multiplicar. Solución

Leer una secuencia de 30 números y mostrar la suma y el producto de ellos..

Leer dos números y realizar el producto median sumas. Solución

Lee una secuencia de números y determina cual es el mayor de ellos.

Calcular la media de una secuencia de números proporcionado por el usuario. Solución

Leer una secuencia se números y mostrar cuáles de ellos es el mayor y el menor.

Leer una secuencia de n números almacenarlos en un vector y mostrar la posición donde se encuentra el mayor valor leído. Solución

Dado dos vectores A y B de n elementos cada uno, obtener un vector C donde la posición i se almacene la suma de A[i]+B[i]. Solución

Dado una secuencia de número leídos y almacenados en un vector A mostrar dichos números en orden.

Dado un vector de secuencias de caracteres mostrar la longitud de cada una de ellas.

Dado un DNI proporcionado por el usuario determinar si es correcto o no. Solución

Dado una secuencia de textos proporcionados por el usuario visualizar: la longitud, en mayúsculas, en minúsculas, cada uno de ellas. Solución

Funciones en Javascript

- Cuando una serie de instrucciones se repiten una y otra vez, se complica demasiado el código fuente de la aplicación, ya que:

  El código de la aplicación es mucho más largo porque muchas instrucciones están repetidas.

  Si se quiere modificar alguna de las instrucciones repetidas, se deben hacer tantas modificaciones como veces se haya escrito esa instrucción, lo que se convierte en un trabajo muy pesado y muy propenso a cometer errores.

- Las funciones son la solución a todos estos problemas, tanto en JavaScript como en el resto de lenguajes de programación. Una función es un conjunto de instrucciones que se agrupan para realizar una tarea concreta y que se pueden reutilizar fácilmente.

Funciones en Javascript

Las funciones en JavaScript se definen mediante la palabra reservada function, seguida del nombre de la función. Su definición formal es la siguiente:

function nombre_funcion()
{
...
}

El nombre de la función se utiliza para llamar a esa función cuando sea necesario. El concepto es el mismo que con las variables, a las que se les asigna un nombre único para poder utilizarlas dentro del código. Después del nombre de la función, se incluyen dos paréntesis cuyo significado se detalla más adelante. Por último, los símbolos { y } se utilizan para encerrar todas las instrucciones que pertenecen a la función

Argumentos y valores de retorno:

// Definición de **variables locales**
f***unction*** nombre_funcion(*[p1, p2, …pn]*) {
// Definición de **variables locales**
...
*[**return** valor_retorno;]*
}

Ejercicios

Función que calcule el factorial de un valor. Solución

Función que determine si un DNI es correcto o no. Solución

Función que determine si un E-mail si cumple el formato.

Función que dado un vector de temperaturas devuelva un vector con aquellas temperaturas < 0º grados. Solución

Función que dado un vector y un entero determine si se encuentra o no.

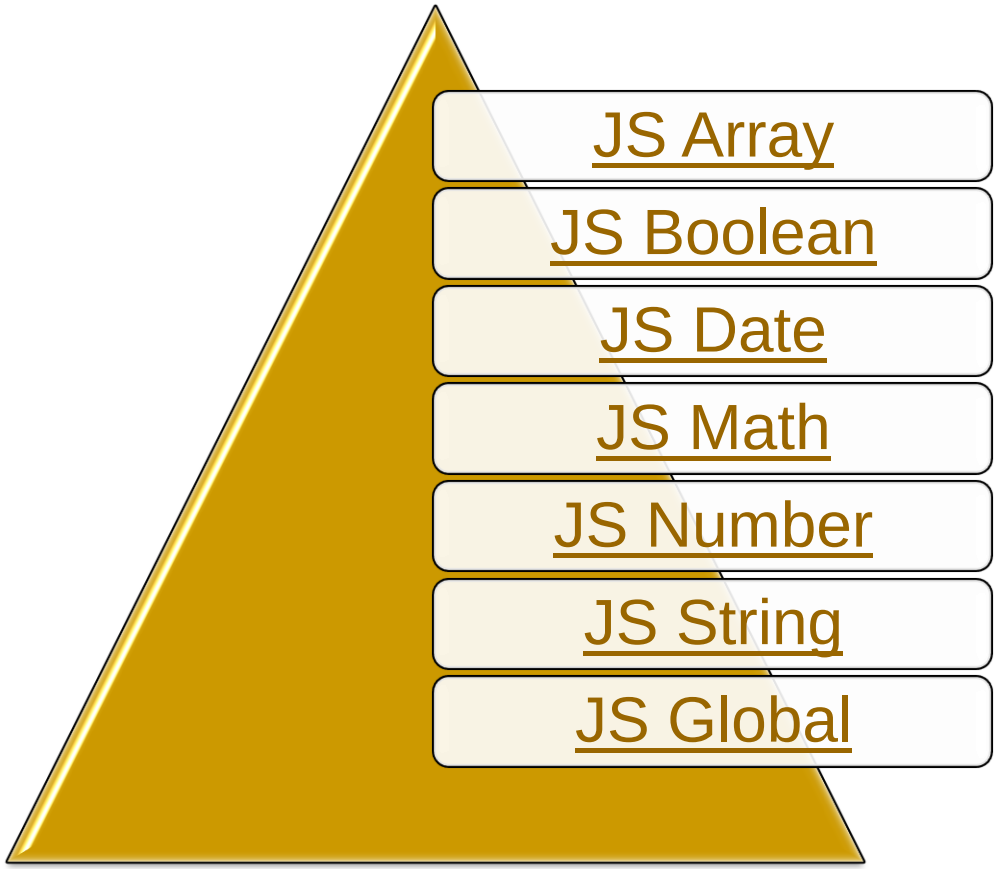
Función que dado un vector de palabras y una palabra determine si se encuentra o no. Solución

Función que dado un entero devuelva todos sus divisores.

Leer una secuencia de n números almacenarlos en un vector y mostrar la posición donde se encuentra el mayor valor leído.

Dado dos vectores A y B de n elementos cada uno, obtener un vector C donde la posición i se almacene la suma de A[i]+B[i]. Solución

JS Objetos: Number, Array, ..



http://www.w3schools.com/jsref/

Eventos

Manejadores de eventos como atributos XHTML

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

 Puedes pinchar sobre este elemento o simplemente pasar el ratón por encima

<body onload="alert('La página se ha cargado completamente');"> Solución

Manejadores de eventos y variable 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... Solución

 Sección de contenidos... Solución

Evento	Descripción	Elementos para los que está definido
onblur	Deseleccionar el elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
onchange	Deseleccionar un elemento que se ha modificado	<input>, <select>, <textarea>
onclick	Pinchar y soltar el ratón	Todos los elementos
ondblclick	Pinchar dos veces seguidas con el ratón	Todos los elementos
onfocus	Seleccionar un elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
onkeydown	Pulsar una tecla (sin soltar)	Elementos de formulario y <body>
onkeypress	Pulsar una tecla	Elementos de formulario y <body>
onkeyup	Soltar una tecla pulsada	Elementos de formulario y <body>
onload	La página se ha cargado completamente	<body>
onmousedown	Pulsar (sin soltar) un botón del ratón	Todos los elementos
onmousemove	Mover el ratón	Todos los elementos
onmouseout	El ratón "sale" del elemento (pasa por encima de otro elemento)	Todos los elementos
onmouseover	El ratón "entra" en el elemento (pasa por encima del elemento)	Todos los elementos
onmouseup	Soltar el botón que estaba pulsado en el ratón	Todos los elementos
onreset	Inicializar el formulario (borrar todos sus datos)	<form>
onresize	Se ha modificado el tamaño de la ventana del navegador	<body>
onselect	Seleccionar un texto	<input>, <textarea>
onsubmit	Enviar el formulario	<form>
onunload	Se abandona la página (por ejemplo al cerrar el navegador)	<body>

Eventos

Manejadores de eventos como funciones externas

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

Solución

```
function resalta(elemento) {
        switch(elemento.style.borderColor)
        {
           case ‘blue‘:  elemento.style.borderColor = 'black'; break;
           case 'black': elemento.style.borderColor = 'blue'; break;
        }
}
<div style="width:150px; height:60px; border:thin solid blue" onmouseover="resalta(this)"
     onmouseout="resalta(this)"> Sección de contenidos... </div>
```

Solución

DOM (Modelo de Objetos del documento)

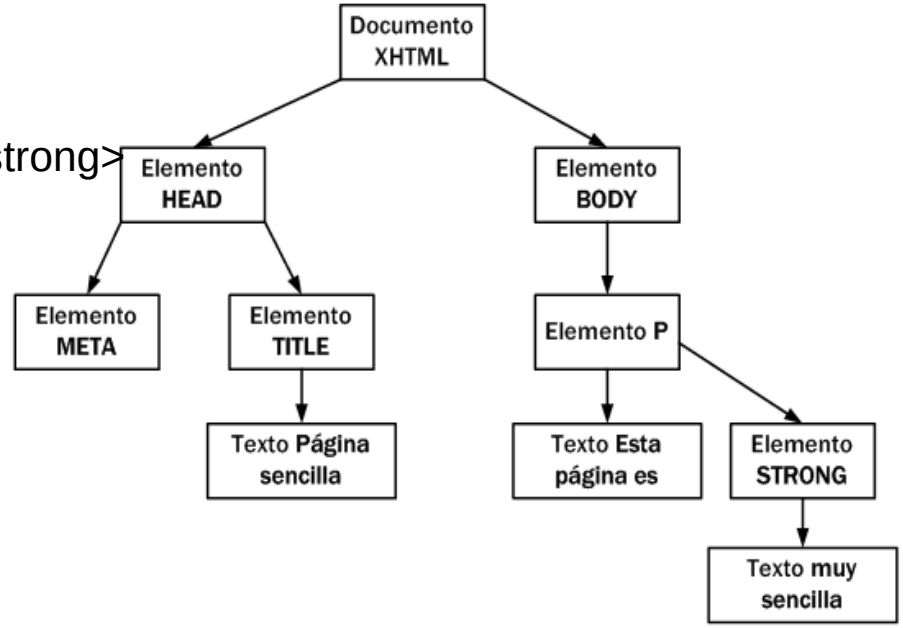
- Cuando se definió el lenguaje XML, surgió la necesidad de procesar y manipular el contenido de los archivos XML mediante los lenguajes de programación tradicionales. XML es un lenguaje sencillo de escribir pero complejo para procesar y manipular de forma eficiente. Por este motivo, surgieron algunas técnicas entre las que se encuentra DOM.

- DOM o Document Object Model es un conjunto de utilidades específicamente diseñadas para manipular documentos XML. Por extensión, DOM también se puede utilizar para manipular documentos XHTML y HTML. Técnicamente, DOM es una API de funciones que se pueden utilizar para manipular las páginas XHTML de forma rápida y eficiente.

DOM (Modelo de Objetos del documento)

Si se considera la siguiente página HTML sencilla:

```
<!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=iso-8859-1" />
<title>Página sencilla</title>
</head>
<body>
<p>Esta página es <strong>muy sencilla</strong>
</p>
</body>
</html>
```



DOM (Modelo de Objetos del documento)

Una vez que DOM ha creado de forma automática el árbol completo de nodos de la página, ya es posible utilizar sus funciones para obtener información sobre los nodos o manipular su contenido. JavaScript crea el objeto Node para definir las propiedades y métodos necesarios para procesar y manipular los documentos.

Propiedad/Método	Valor devuelto	Descripción
nodeName	String	El nombre del nodo (no está definido para algunos tipos de nodo)
nodeValue	String	El valor del nodo (no está definido para algunos tipos de nodo)
nodeType	Number	Una de las 12 constantes definidas anteriormente
ownerDocument	Document	Referencia del documento al que pertenece el nodo
firstChild	Node	Referencia del primer nodo de la lista childNodes
lastChild	Node	Referencia del último nodo de la lista childNodes
childNodes	NodeList	Lista de todos los nodos hijo del nodo actual
previousSibling	Node	Referencia del nodo hermano anterior o null si este nodo es el primer hermano
nextSibling	Node	Referencia del nodo hermano siguiente o null si este nodo es el último hermano
hasChildNodes()	Bolean	Devuelve true si el nodo actual tiene uno o más nodos hijo
attributes	NamedNodeMap	Se emplea con nodos de tipo Element. Contiene objetos de tipo Attr que definen todos los atributos del elemento
appendChild(nodo)	Node	Añade un nuevo nodo al final de la lista childNodes
removeChild(nodo)	Node	Elimina un nodo de la lista childNodes
replaceChild(nuevoNodo, anteriorNodo)	Node	Reemplaza el nodo anteriorNodo por el nodo nuevoNodo
insertBefore(nuevoNodo, anteriorNodo)	Node	Inserta el nodo nuevoNodo antes que la posición del nodo anteriorNodo dentro de la lista childNodes

La interfaz Node.....

DOM (Modelo de Objetos del documento)

getElementsByTagName()
getElementsByName() *http://www.w3schools.com/jsref/met_doc_getelementsbyname.asp*
getElementById()

```
var parrafos = document.getElementsByTagName("p");
var primer_parrafo = document.getElementsByTagName("p")[0];
var numero_parrafos=document.getElementsByTagName("p").length;
var ultimo_parrafo = document.getElementsByTagName("p")[numero_parrafos-1];

var la_capa = document.getElementById(“capa_1");
var nodos = document.getElementById(“capa_1").childNodes;
var primer_nodo = document.getElementById(“capa_1").childNodes[0];
var primer_nodo = document.getElementById(“capa_1").firstChild;
var ultimo_nodo = document.getElementById(“capa_1"). lastChild ;

var capa= document.getElementById(“capa_1");
var primer_parrafo=capa. getElementsByTagName("p")[0]; Solución
```

Acceso directo a los nodos…..

DOM (Modelo de Objetos del documento)

Una vez que se ha accedido a un nodo, el siguiente paso natural consiste en acceder y/o modificar sus atributos y propiedades. Mediante DOM, **es posible acceder de forma sencilla a todos los atributos XHTML y todas las propiedades CSS de cualquier elemento de la página.**

Los atributos XHTML de los elementos de la página se transforman automáticamente en propiedades de los nodos. Para acceder a su valor, simplemente se indica el nombre del atributo XHTML detrás del nombre del nodo.

```
var enlace = document.getElementById("enlace");
alert(enlace.href);
<a id="enlace" href="http://www...com">Enlace</a>
```

Las propiedades CSS no son tan fáciles de obtener como los atributos XHTML. Para obtener el valor de cualquier propiedad CSS del nodo, se debe utilizar el atributo style.

```
var imagen = document.getElementById("imagen");
alert(imagen.style.margin);
<img id="imagen" style="margin:0; border:0;" src="logo.png" />
```

La propiedad border-top-style se accede en DOM mediante el nombre borderTopStyle.

Acceso directo a los atributos…..

DOM (Modelo de Objetos del documento)

```
// 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);
```

Crear un nodo I.....

DOM (Modelo de Objetos del documento)

```
var capa = document.createElement(“div");
var parrafo = document.createElement("p");
var contenido = document.createTextNode("Hola Mundo!");

parrafo.appendChild(contenido);
capa.appendChild(parrafo);

capa.id=“capa_hija”;
capa.style.border=“1px solid blue”;
capa.style.height=“100px”;
capa.style.width=“100px”;

var capa_padre=document.getElementById(“capaX”);
capa_padre.appendChild(capa);
```

Solución

Crear un nodo II.....

DOM (Modelo de Objetos del documento)

```
var capa = document.createElement(“div");
var parrafo = document.createElement("p");
var contenido = document.createTextNode("Hola Mundo!");

parrafo.appendChild(contenido);
capa.appendChild(parrafo);

capa.id=“capa_hija”;
capa.style.border=“1px solid blue”;
capa.style.height=“100px”;
capa.style.width=“100px”;

var capa_padre=document.getElementById(“capaY”);
var segundo_parrafo=capa_padre.getElementsByTagName(“p”)[1];

segundo_parrafo.parentNode.insertBefore(capa, segundo_parrafo);
```

Solución

Crear un nodo III.....

DOM (Modelo de Objetos del documento)

En este caso, solamente es necesario utilizar la función removeChild():

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

... Solución

La función removeChild() requiere como parámetro el nodo que se va a eliminar. Además, esta función debe ser invocada desde el elemento padre de ese nodo que se quiere eliminar. La forma más segura y rápida de acceder al nodo padre de un elemento es mediante la propiedad nodoHijo.parentNode.

Así, para eliminar un nodo de una página XHTML se invoca a la función removeChild() desde el valor parentNode del nodo que se quiere eliminar. Cuando se elimina un nodo, también se eliminan automáticamente todos los nodos hijos que tenga, por lo que no es necesario borrar manualmente cada nodo hijo.

DOM (Modelo de Objetos del documento)

Utilizando las funciones DOM de JavaScript, se crea el nuevo párrafo que se va a mostrar en la página, se obtiene la referencia al nodo original y se emplea la función replaceChild() para intercambiar un nodo por otro:

```
var nuevoP = document.createElement("p");
var texto = document.createTextNode("Este párrafo se ha creado dinámicamente y sustituye al parrafo original");
nuevoP.appendChild(texto);


var capa= document.getElementById(“capaZ”);
var anteriorP =capa.getElementsByTagName("p")[0];

anteriorP.parentNode.replaceChild(nuevoP, anteriorP);
```

Solución

Reemplazar nodos ..

Ejercicios (DOM/Eventos)

Su puesto un con border definido. Define un mecanismo de forma que el usuario pueda mover 10px a la izquierda, arriba, abajo o la derecha. Solución

Tras 5 segundos aparecerá un con imagen de fondo en una position:absolute. *setTimeout(), setInterval()*

Elabora un en el que aparezca un reloj. Solución

Implementa el cambio de idioma del contenido de una capa. *Utiliza <span lang””>* Solución

Dado un XHMTL en el que tenemos un <form> de registro de un usuario definir las comprobaciones que se consideren oportunas respecto a los values incorporados por el usuario.

<form name=”reg” action=”prueba.php” onsubmit=”return valida(this);”> Solución

Cuando el ratón pase por encima de una imagen que sea sustituida por otra y que al salir el cursos del área la imagen anterior se recupere. Solución

Cambiar el href de todos los <a> de la página.

Eliminar todos los <p> de una determinada capa . Solución

Dado un XHTML en el que tenemos un formulario de subida de ficheros del que inicialmente solo se ha habilitado para subir uno. Añade los elementos necesarios que permitan que el usuario pueda añadir tantos como considere oportunos. Solución

Ejercicios (DOM/Eventos)

Cuando se onload() la pagina se crear un el cual se mueva en vertical a intervalos de 2 segundos. *setTimeout(),setInterval()*

Página web con un <form>para realizar un registro de un usuario. Se le añadirá un enlace que permita antes de realizar el <input type="submit"../> verificar la validez de los datos introducidos. Aquel <input> valor incorrecto se mostrará un <p> explicativo. Solución

Un xhtml de forma que cuando se ejecute el evento onload() del <body> un imagen cambie su src en base a un valor aleatorio. Utiliza la función random y un vector de nombres de ficheros.

Un formulario con un <input>, de tal manera que cada vez que pulse una tecla verifique si es un valor dígito y si no lo es se eliminará del value. Utiliza el evento *onkeyup="comprobar(this)"* y la función *charCodeAt()* del JS String Solución

Elabora un menú de forma que cuando el ratón pase por las opciones principales se haga visible el submenú asociado.

Implementa el juego adivina.

*var a=new Number(Math.random()*100);*

a.toFixed(); Solución

Insertar en una capa un <img> al principio o al final. Las propiedades y atributos se obtienen del usuario. Solución

Ejercicios (DOM/Eventos)

Elabora un formulario donde se incorpore un calendario y la aceptación de las políticas de privacidad. Solución

Elabora un sistema de forma que el menú oculte y visualice el contenido en una capa centrada a la página html. Solución

Fotoslider. Realiza una página que muestre un proyector de fotografías para la venta de un piso. Tendremos una carpeta con n imágenes. Necesitaremos una variable global que determine qué imagen es la que se está visualizando. Se definirá 3 funciones:ver(), previous() y first(). Cada 5 segundo se actualizará la imagen con la siguiente a la visualizada. Se visualizarán todas las imágenes en pequeño y se podrán onclick visualizarla. Solución

XML DOM

Para extraer la información que contiene un documento XML, se podría escribir código para analizar el contenido del archivo XML, pues no deja de ser un archivo de texto, tal y como lo podríamos hacer con HTML. Sin embargo, esta solución no es muy aconsejable y desaprovecharía una de las ventajas de XML: el ser una forma estructurada de representar datos.

La mejor forma de recuperar información de archivos XML es utilizar un parser de XML, que sea compatible con el modelo de objeto de documento (DOM) de XML. DOM define un conjunto estándar de comandos que los parsers devuelven para facilitar el acceso al contenido de los documentos XML desde sus programas. Un analizador de XML compatible con DOM toma los datos de un documento XML y los expone mediante un conjunto de objetos que se pueden programar.

DOM para XML es un modelo de objetos estándar (propuesto por el W3C) que muestra el contenido de un documento XML. La Especificación del Modelo de Objeto de documento (DOM) del W3C define actualmente lo que debería mostrar un DOM como propiedades, métodos y eventos.

http://www.w3schools.com/dom/default.asp

Enlaces y referencias de interés

- http://www.librosweb.es/
- http://www.w3schools.com/jsref/default.asp
- http://www.w3schools.com/js/default.asp