JAVASCRIPT

• Lenguaje Interpretado

• Multiplataforma

• Débilmente Tipado y Dinámico

• Orientado a Objetos y Eventos

• Imperativo • Basado en prototipos

• Creado por Brendan Eich, programador de Netscape v2.0 en 1995. ->Livescript ->javascript

• Microsoft lanzó Jscript para Internet Explorer 3.0

Funciones

• isNaN(variable);

//devuelve true si no lo es

• parseInt(variable);

//convierte en número entero

• parseFloat(variable);

//convierte en número flotante

• String(variable);

//convierte en cadena de texto

toFixed ();

Devuelve el número original con tantos decimales como los indicados por el parámetro dígitos y realiza los redondeos necesarios.

length

Calcula la longitud de una cadena de texto (el número de caracteres que la forman).

toUpperCase()

Transforma el texto en mayúscula

toLowerCase()

Transforma el texto en minúscula

charAt()

Obtiene el caracter que en la posición indicada

substring(inicio, final)

Toma una porción del texto

indexOf(caracter)

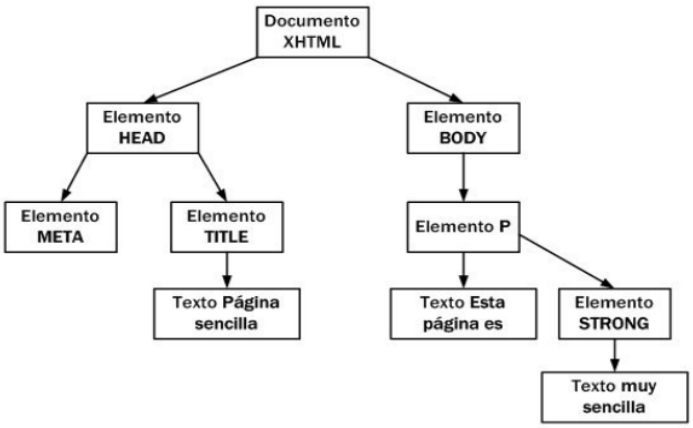
(Calcula la posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si el carácter se incluye varias veces dentro de la cadena de texto, se devuelve su primera posición empezando a buscar desde la izquierda. Si la cadena no contiene el carácter, la función devuelve el valor -1)

lastIndexOf(caracter)

(calcula la última posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si la cadena no contiene el carácter, la función devuelve el valor -1. La posición devuelta se calcula empezando a contar desde el principio de la palabra)

DOM

```
<!DOCTYPE html>
<html>
<head>
     <meta charset="UTF-8">
     <title>Página sencilla</title>
</head>
<body>
     <p>Esta página es <strong>muy sencilla</strong></p>
</body>
</html>
```



DOM define 12 tipos de nodos, aunque las páginas HTML habituales se pueden manipular manejando solamente cuatro o cinco tipos de nodos:

1. **Document** Nodo raíz del que derivan todos los demás nodos del árbol.

2. **Element** Representa cada una de las etiquetas HTML. Se trata del único nodo que puede contener atributos y el único del que pueden derivar otros nodos.

3. **Attr** Se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas HTML, es decir, uno por cada par atributo=valor.

4. **Text** Nodo que contiene el texto encerrado por una etiqueta HTML.

5. **Comment** Representa los comentarios incluidos en la página HTML

Acceso directo a los nodos

1) **getElementsByTagName(nombreEtiqueta)** La función obtiene todos los elementos de la página HTML cuya etiqueta sea igual que el parámetro que se le pasa a la función.

2) **getElementsByName()** Se buscan los elementos cuyo atributo "name" sea igual al parámetro proporcionado

3) **getElementById()** Devuelve el elemento HTML cuyo atributo id coincide con el parámetro indicado en la función. Como el atributo id debe ser único para cada elemento de una misma página, la función devuelve únicamente el nodo deseado.

Creación de elementos HTML simples

Por este motivo, crear y añadir a la página un nuevo elemento HTML sencillo consta de cuatro pasos diferentes:

1. Creación de un nodo de tipo Element que represente al elemento.

2. Creación de un nodo de tipo Text que represente el contenido del elemento.

3. Añadir el nodo Text como nodo hijo del nodo Element.

4. Añadir el nodo Element a la página, en forma de nodo hijo del nodo correspondiente al lugar en el que se quiere insertar el elemento.

El proceso de creación de nuevos nodos puede llegar a ser tedioso, ya que implica la utilización de tres funciones DOM:

1. createElement(etiqueta): crea un nodo de tipo Element que representa al elemento HTML cuya etiqueta se pasa como parámetro.

2. createTextNode(contenido): crea un nodo de tipo Text que almacena el contenido textual de los elementos HTML.

3. nodoPadre.appendChild(nodoHijo): añade un nodo como hijo de otro nodo. Se debe utilizar al menos dos veces con los nodos habituales: en primer lugar se añade el nodo Text como hijo del nodo Element y a continuación se añade el nodo Element como hijo de algún nodo de la página.

```
// 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);
```

Eventos de Teclado

EVENTO	DESCRIPCIÓN	ELEMENTOS QUE PUEDEN TENER ESTE EVENTO
keyup	dejar de teclear	TODOS
keypress	presionar una tecla	TODOS
keydown	presionar una tecla sin soltar	TODOS

Eventos de Formularios

EVENTO	DESCRIPCIÓN	ELEMENTOS QUE PUEDEN TENER ESTE EVENTO
focus	seleccionar un elemento	`<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`, `<body>`
blur	deseleccionar un elemento	`<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`, `<body>`
submit	enviar un formulario	`<form>`
reset	al resetear un formulario	`<form>`
change	deseleccionar un elemento que fue modificado	`<input>`, `<select>`, `<textarea>`

Eventos de pagina

EVENTO	DESCRIPCIÓN	ELEMENTOS QUE PUEDEN TENER ESTE EVENTO
load	Al cargar la página	`<body>`
unload	Al abandonar la página	`<body>`
resize	al achicar o agrandar la ventana del navegador	`<body>`

Eventos de Mouse

EVENTO	DESCRIPCIÓN	ELEMENTOS QUE PUEDEN TENER ESTE EVENTO
click	al hacer click	TODOS
dblclick	al hacer doble click	TODOS
mouseover	al pasar el mouse	TODOS
mousedown	mientras tengo presionado el mouse	TODOS
mouseup	cuando suelto el mouse	TODOS
mousemove	cuando muevo el mouse	TODOS

Local Storage

Guardar

localStorage.setItem(“nombreVariable”, valor);

Obtener

localStorage.getItem(“nombreVariable”);

Eliminar

localStorage.removeItem(“nombreVariable”);

LocalStorage, almacena datos de tipo string.
Si queremos almacenar un json, tenemos que convertirlo a string al guardarlo en localStorage y parsearlo para recuperarlo como JSON

```
localStorage.setItem("productos",
JSON.stringify(productos));

const productosObtenidos =
JSON.parse(localStorage.getItem("productos"));
```