Eventos (1ª parte)
Formularios
DOM (iniciación)

Eventos en JavaScript

En esta primera entrega utilizaremos los atributos de eventos de las diferentes etiquetas de html. En futuras entrega profundizaremos en la creación de eventos.

Un evento no es más que un suceso para el que puede interesarnos definir una respuesta. Por ejemplo, ante una pulsación del ratón podemos querer avisar de algo.

Los elementos que encontramos en html tiene unos eventos asignados. Hay unos eventos que son genéricos, es decir, que los podemos encontrar en la mayoría de los elementos y hay otros eventos que son exclusivamente de algunos elementos.

Intentar ver todos los eventos y cada uno a que elemento pertenece es una tarea muy larga y bastante repetitiva, por tanto, explicaremos los eventos e iremos diciendo en que elementos los podemos encontrar.

Los eventos son objetos del tipo *event*.

Por ejemplo, el evento onMouseOver es un evento que lo podemos encontrar en todos los elementos. El ratón puede pasar por encima de cualquier elemento de una página web. El evento onFocus, sólo puede producirse en los objetos que puedan coger el foco, por ejemplo un elemento input.

```
<a href="mi_enlace.html" onMouseOver="Hacer_Algo();">Sucederá
algo si se pasa por aquí encima</A>
```

Veamos ahora algunos eventos y en que etiquetas podemos encontrarlo.

La siguiente tabla resume los eventos más importantes definidos por JavaScript:

Evento	Descripción	Elementos que lo soporta
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>

Evento	Descripción	Elementos que lo soporta
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>

Fuente de la tabla http://librosweb.es/javascript/capitulo_6/modelo_basico_de_eventos.html

Veamos ahora como podemos trabajar con los eventos.

<Etiqueta onEvento="CodigoJS;">

donde CodigoJS puede ser una llamada a una función o una secuencia de sentencias javascript.

Este primer modo de utilizar los eventos lo utilizaremos para familiarizarnos con los eventos y poco a poco iremos evolucionando en su uso.

Cuando creamos un manejador para el evento, estamos asignando al correspondiente objeto una propiedad que toma el nombre de dicho manejador, y es esta propiedad la que nos permite acceder al manejador del evento del objeto. Esto mismo es lo que nos va a permitir cambiar el código asociado al evento de un objeto.

Vamos a explicar esto con un ejemplo sencillo.

Al hacer click sobre un parrafo queremos que aparezca una ventana que nos salude.

```
<html>
<head>
<script>
function saludo(){
   alert("Buenos días a todo el mundo");
}
</script>
</head>
<body>
<p onclick="saludo()">buenos días</p>
</body>
</html>
```

Veamos ahora la lista de eventos cuando se producen.

Tipo de evento	Nombre con prefijo on	Descripción
Relacionados con el ratón	onclick	Click sobre un elemento
	ondblclick	Doble click sobre un elemento
	onmousedown	Se pulsa un botón del ratón sobre un elemento
	onmouseenter	El puntero del ratón entra en el área de un elemento
	onmouseleave	El puntero del ratón sale del área de un elemento
	onmousemove	El puntero del ratón se está moviendo sobre el área de un elemento
	onmouseover	El puntero del ratón se sitúa encima del área de un elemento
	onmouseout	El puntero del ratón sale fuera del área del elemento o fuera de uno de sus hijos
	onmouseup	Un botón del ratón se libera estando sobre un elemento
	contextmenu	Se pulsa el botón derecho del ratón (antes de que aparezca el menú contextual)
Relacionados con el teclado	onkeydown	El usuario tiene pulsada una tecla (para elementos de formulario y body)
	onkeypress	El usuario pulsa una tecla (momento justo en que la pulsa) (para elementos de formulario y body)

	onkeyup	El usuario libera una tecla que tenía pulsada (para elementos de formulario y body)
Relacionados con formularios	onfocus	Un elemento del formulario toma el foco
	onblur	Un elemento del formulario pierde el foco
	onchange	Un elemento del formulario cambia
	onselect	El usuario selecciona el texto de un elemento input o textarea
	onsubmit	Se pulsa el botón de envío del formulario (antes del envío)
	onreset	Se pulsa el botón reset del formulario
Relacionados con ventanas o frames	onload	Se ha completado la carga de la ventana
	onunload	El usuario ha cerrado la ventana
	onresize	El usuario ha cambiado el tamaño de la ventana

Existen más eventos. Durante el curso iremos incluyendo los eventos que necesitemos. Esta lista es parte de la que podemos encontrar en http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=842:lista-de-eventos-javascript-on-click-dblclick-mouseover-mouseout-change-submit-keypress-cu01159e&catid=78:tutorial-basico-programador-web-javascript-desde-&Itemid=206

Ahora vamos a incluir un evento desde el código javascript asignando dicho evento a un objeto de la ventana.

Por ejemplo, podemos mostrar un mensaje cuando la página se haya cargado.

```
<html>
<head>
<script>
function aviso(){
  alert("la página se ha cargado completamente");
}
</script>
</head>
<body onload="aviso()">
<p>buenos días</p>
</body>
</html>
```

Ahora lo realizaremos desde el código javascript sin utilizar los atributos de evento de la etiqueta *body.*

```
<html>
<head>
<script>
function aviso(){     alert("la página se ha cargado completamente");}
window.onload=aviso;
</script>
</head>
<body>
<p>buenos días</p>
</body>
</html>
```

Vamos a comentar algunas cuestiones sobre el evento onError. Este evento se da cuando hay un error de sintaxis JavaScript.

Por ejemplo, si intentamos cargar una página que no existe, no se dará este evento.

También tenemos que saber que este evento puede recibir tres parámetros de entrada:

1. Mensaje de error
2. URL. El nombre del fichero que produjo el error
3. Número de línea donde se produce el error.
4. Número de columna donde se produce el error.

Si la función retorna false o nada también se visualiza el error en el depurador del navegador. Si la función retorna true, el error no se visualiza en el depurador del navegador.

```
<html>
<head>
<script>
function iniciar(){
   window.onerror=fallos;
}
function fallos(mensaje,url,linea)
{
  alert("error: " + mensaje + "\n url: " + url + "\n linea: " + linea);
  return true;
}
function abrir()
{
   alert("mensaje de bienvenida");
}
window.onload=iniciar;
</script>
</head>
<body>
<p>Métodos<br/>
<form>
   <input type="button" onclick="abrr()">abrir</button>
   <input type="button" onclick="abrir()">peque</button>
</form>
</body>
</html>
```

También podemos utilizar este evento con las etiquetas img, input type=”image”, object, script, style. Todas estas etiquetas tienen el atributo src. Por tanto, produce el error si no encuentra el fichero externo.

Funciones anónimas

Se denominan funciones anónimas las funciones que no tienen nombre. Explicado así parece que no tenga sentido.

function(){..... definición de función .....}

El problema de crear funciones de este tipo es el no poder hacer referencia a estas funciones.

En principio esto no tiene sentido.

La forma de crearlas y poder acceder a ellas es la siguiente:

```
window.onload=function(){  ..... definición de la función .....}
```

```
<!DOCTYPE html>
<html>
<head>
<script>
window.onload=function(){
   alert(“la página se ha cargado correctamente”);
}
</script>
</head>
<body>
<p>HOLA MUNDO</p>
</body>
</html>
```

En este ejemplo, la ventana de alert se mostrará después de haber visualizado la página.

Formularios

En este tema veremos algunas cosas un poco más prácticas. Vamos a comprobar que los datos introducidos en una formulario son correctos.

Para poder comprobar los datos tenemos que poder acceder a cada uno de ellos y poder comprobar sus valores. Después de comprobar dichos valores podremos enviar o no el formulario al servidor para que pueda tratar la información enviada.

Para empezar tenemos que saber que javascript organiza ciertos elementos como una array de objetos. Uno de estos elementos es el elemento *form*.

Para acceder a los campos podemos hacerlo de diferentes maneras. Para acceder a un campo de un formulario es muy sencillo.

```
<form method="post" action="prueba.php">
  Nombre <input type="text" name="nombre" size="30" maxlength="30" /><br />
  Edad <input type="text" name="edad" size="3" maxlength="3" /> <br />
  <input type="submit" value="enviar" />
  <input type="button" onclick="acceder()" value="Acceso"/>
</form>
```

Para acceder al campo cuyo atributo *name* es igual a “nombre” sólo tenemos que poner:

```
alert(document.forms[0].nombre.value);
```

También podemos acceder y obtener el dato para luego poder trabajar:

```
var js_nombre=document.forms[0].nombre.value;
```

A partir de este momento el contenido del campo “nombre” está en la variable “js_nombre”. Siempre obtendremos datos alfanuméricos.

También tenemos que aprender que dentro de cada elemento del array “forms” crea un array “elements”. Y cada elemento que forma parte de este array es un elemento del formulario.

```
alert(document.forms[0].elements[0].value);
```

Visualiza el contenido del campo “nombre” del formulario.

Esta forma de acceder a los elementos de un formulario tiene un pequeño problema. Si añadimos un nuevo formulario o añadimos campos, el orden del array varía y tendríamos que reescribir todo el código javascript.

Un modo de solucionar este problema es asignando un nombre al formulario y acceder mediante este nombre.

```
function acceder()
{
  alert(document.prueba.nombre.value);
}
</script>
</head>
<body>
  <form method="post" action="prueba.php" name="prueba">
    Nombre <input type="text" name="nombre" size="30" maxlength="30" /><br />
    Edad <input type="text" name="edad" size="3" maxlength="3" /> <br />
    <input type="submit" value="enviar" />
    <input type="button" onclick="acceder()" value="Acceso"/>
  </form>
```

Ahora podemos hacer una ejemplo un poco más complejo.

Supongamos que queremos crear una página en la que al pulsar una vez sobre un botón pase una cosa, y que cuando pulsemos otra vez (y las siguientes), pase algo completamente distinto a lo que sucedió la primera vez. Por ejemplo, tenemos un botón que al pulsarlo la primera vez nos

mostraría una ventana preguntándonos el nombre, y que las siguientes veces que lo pulsáramos nos dijera "yo te conozco de algo, ¿verdad? Tú eres XXXXXX".

Hacer esto implica que de alguna manera debemos ser capaces de cambiar el código asociado al evento en el que queremos que se active la acción. ¿Cómo lo cambiamos? No debemos olvidar que el nombre de una función es una referencia a la zona de memoria en la que se encuentra, y si cambiamos al manejador del evento la referencia a una cierta función, estamos cambiando la función que se ejecutará cuando el evento suceda. Para aclarar ideas, vamos a realizar el ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<script >
<!--
var Nombre;
function Pulsacion1Boton() {
  Nombre = prompt("Dime cómo te llamas", "Mi nombre");
  document.Form1.Boton1.onclick = Pulsacion2Boton;
}
function Pulsacion2Boton() {
  alert("Hola, \"" + Nombre + "\", creo que ya nos conocemos :-)");
}
//-->
</script>
</head>

<body>
<form name="Form1">
<linput type=”button” name="Boton1" value="Púlsame" onClick="Pulsacion1Boton();">
</form>
</body>
</html>
```

Lo importante del ejemplo es ver que se puede asignar otra función de forma directa al manejador del evento.

Cuando estamos trabajando con radiobutton o checkbox, hay que poner el mismo nombre a todos los elementos que forman un mismo conjunto y lo que nos devuelve al intentar acceder a dicho nodo es una array, y el número de elementos que lo forman es el número de entradas con el mismo nombre que existen en el formulario.

```
function acceder()
{
  js_aficion=document.prueba.aficion;

  for(a=0;a<js_aficion.length;a++)
  {
    if(js_aficion[a].checked) alert(js_aficion[a].value);
  }

  js_musica=document.prueba.musica;
```

```
  for(a=0;a<js_musica.length;a++)
  {
    if(js_musica[a].checked) alert(js_musica[a].value);;
  }

}
</script>
</head>
<body>
  <form method="post" action="prueba.php" name="prueba">
    Nombre <input type="text" name="nombre" size="30" maxlength="30" /><br />
    Edad <input type="text" name="edad" size="3" maxlength="3" /> <br />
    futbol <input type="radio" name="aficion" value="fut" />
    balonmano <input type="radio" name="aficion" value="bom" />
    tenis <input type="radio" name="aficion" value="ten" /><br />
    clásica <input type="checkbox" name="musica" value="class" />
    rock <input type="checkbox" name="musica" value="rock" />
    romantica <input type="checkbox" name="musica" value="roman" /><br />
    <input type="submit" value="enviar" />
    <input type="button" onclick="acceder()" value="Acceso"/>
  </form>
```

Otro modo de acceder a cada uno de los elementos de un formulario es utilizando las funciones de DOM para acceder de forma directa a cada uno de los nodos. La primera función que vamos a ver es *getElementById("id_nombre")*. "id_nombre" es el atributo "id" de la etiqueta. Ya sabemos que los valores que se asignan a este atributo debe ser único en toda la página web.

```
<script>

function acceder()
{
  var js_nombre=document.getElementById("id_nombre");
  alert(js_nombre.value);
}

</script>
</head>
<body>
  <form method="post" action="prueba.php" name="prueba">
    Nombre <input type="text" name="nombre" id="id_nombre" size="30"
              maxlength="30" /><br />
    Edad <input type="text" name="edad" id="id_edad" size="3" maxlength="3" /> <br />

    <input type="submit" value="enviar" />
    <input type="button" onclick="acceder()" value="Acceso"/>
  </form>
```

Este método es uno de los más utilizados por los programadores web.

Otro método que podemos utilizar es "getElementsByName('nom_name')". Este método nos devuelve un array con todos los elementos de la página web que tenga ese nombre.

```
function acceder()
{
  js_aficion=document.getElementsByName("aficion");

  for(a=0;a<js_aficion.length;a++)
  {
    if(js_aficion[a].checked) alert(js_aficion[a].value);
  }

  js_musica=document.getElementsByName("musica");

  for(a=0;a<js_musica.length;a++)
  {
    if(js_musica[a].checked) alert(js_musica[a].value);;
  }

}

</script>
</head>


<body>
  <form method="post" action="prueba.php" name="prueba">
    Nombre <input type="text" name="nombre" size="30" maxlength="30" /><br />
    Edad <input type="text" name="edad" size="3" maxlength="3" /> <br />
    futbol <input type="radio" name="aficion" value="fut" />
    balonmano <input type="radio" name="aficion" value="bom" />
    tenis <input type="radio" name="aficion" value="ten" /><br />
    clásica <input type="checkbox" name="musica" value="class" />
    rock <input type="checkbox" name="musica" value="rock" />
    romantica <input type="checkbox" name="musica" value="roman" /><br />
    <input type="submit" value="enviar" />
    <input type="button" onclick="acceder()" value="Acceso"/>
  </form>
```

Otro métodos que podemos utilizar para poder acceder o seleccionar los elementos de una página html son:

- *getElementsByClassName()* → Toma todos los elementos que tengan el mismo valor en el atributo *class* de la etiqueta.

- *getElementByTagName()* → Toma todos los elementos que sean de la misma etiqueta.

Hasta ahora hemos visto como acceder hasta uno o varios objetos del navegador y asignarlos a una variable. Dicha variable es un objeto del tipo del objeto al cual hemos accedido, por tanto, sus métodos o propiedades dependerán del objetos que sea.

Como punto de partida hay que tener en cuenta que son propiedades todos los atributos de la etiqueta.

Si deseamos saber el atributo “name” de una etiqueta preguntaremos por *objeto.name*. Hay algún atributo que puede cambiar de nombre, como por ejemplo el atributo *class* es la propiedad *className*.

```
function acceder()
{
  obj_nombre=document.getElementsByName("nombre");

  alert(obj_nombre[0].name + " -- " + obj_nombre[0].className +
        " -- "+ obj_nombre[0].maxLength);
}

</script>
</head>
<body>
  <form method="post" action="prueba.php" name="prueba">
    Nombre <input type="text" name="nombre" size="30" maxlength="30"
           class="pepe" /><br />
    Edad <input type="text" name="edad" size="3" maxlength="3" /> <br />
    <input type="submit" value="enviar" />
    <input type="button" onclick="acceder()" value="Acceso"/>
```

Sólo tenemos que tener en cuenta que al utilizar propiedades con más de una palabra la segunda siempre irá en mayúsculas *obj_nombre[0].className*.

Con los ejercicios iremos viendo las diferentes propiedades y métodos de los objetos. Como puedes comprobar a partir de este punto se abre un gran número de propiedades y métodos que podemos utilizar. Estos varían según el tipo de objetos que estemos tratando.

Validación de formularios

Ahora pasemos a validar el formulario y si todo es correcto realizamos el envío al servidor, de lo contrario no realizamos el envío.

Vamos a estudiar dos formas de controlar en envío del formulario.

Primera forma. Devolviendo un valor *false* al atributo de evento *onsubmit* de la etiqueta *form*. El atributo evento *onsubmit* devuelve *false*. ¿Cómo hacemos esto?

onsubmit=”return false”;

Si queremos que devuelva según la validación

onsubmit=”return funcion()”

y al final de la función *funcion()* colocaremos un *return xxxx* donde *xxxx* será true o false.

```
function acceder()
{
  var salta=false;
  var obj_nombre=document.getElementById("idnombre");
  if(obj_nombre.value.toUpperCase()=="SI")  salta=true;
```

```
    return salta;
  }

  </script>
  </head>
  <body>
    <form method="post" action="prueba.php" name="prueba"
     onsubmit="return acceder()">
     Nombre <input type="text" name="nombre" id="idnombre" size="30" maxlength="30"
                class="pepe" title="Introduce SI para saltar de página" /><br />
     Edad <input type="text" name="edad" size="3" maxlength="3" /> <br />
     <input type="submit" value="enviar" />
    </form>
```

Otro modo de enviar un formulario sin utilizar ninguna etiqueta *input* del tipo *submit* es utilizando el método *submit()* del objeto *form.*

```
  function acceder()
  {
    var salta=false;
    var obj_formu=document.getElementById("idform");
    var obj_nombre=document.getElementById("idnombre");
    if(obj_nombre.value.toUpperCase()=="SI")  salta=true;

    if(salta) obj_formu.submit();
  }

  </script>
  </head>
  <body>
    <form method="post" id="idform" action="prueba.php" name="prueba">
      Nombre <input type="text" name="nombre" id="idnombre" size="30" maxlength="30"
                 class="pepe" title="Introduce SI para saltar de página"/><br />
      Edad <input type="text" name="edad" size="3" maxlength="3" /> <br />

      <input type="button" onclick="acceder()" value="Acceso"/>
    </form>
```

Para darle el foco a un campo utilizamos el método *focus()* del objeto. Si queremos acceder al campo y seleccionar el contenido utilizaremos el método *select()*

DOM

Para W3C DOM (Modelo de objetos del documento) es una interfaz de programación de aplicaciones (API) para documentos HTML y XML. El DOM define la estructura lógica de los documentos y el modo en que se accede y manipula un documento.

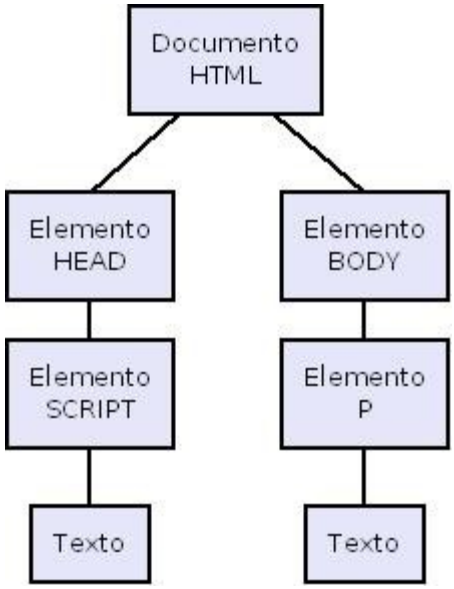
Con el DOM los programadores pueden construir documentos, navegar por su estructura, y añadir, modificar o eliminar elementos y contenido.

Todos los navegadores deben contener y contienen, este interfaz para poder trabajar con documentos HTML y XML.

Al trabajar con DOM nos proporciona un control muy preciso sobre la estructura del documento HTML. DOM crea una estructura de árbol del documento que se va a manipular.

Veamos uno de los ejemplos empleados en el tema

```
<!DOCTYPE html>
<html>
<head>
<script>
window.onload=function(){
   alert("la página se ha cargado correctamente");
}
</script>
</head>
<body>
<p>HOLA MUNDO</p>
</body>
</html>
```



Cada elemento se denomina nodo. Existen diferentes tipos de nodos. Los nodos que a nosotros nos puede interesar son los siguientes:

- Document: es el nodo raíz de todos los documentos HTML y XML. Todos los demás nodos derivan de él.
- DocumentType: es el nodo que contiene la representación del DTD empleado en la página (indicado mediante el DOCTYPE).
- Element: representa el contenido definido por un par de etiquetas de apertura y cierre (<etiqueta>...</etiqueta>) o de una etiqueta abreviada que se abre y se cierra a la vez (<etiqueta/>). Es el único nodo que puede tener tanto nodos hijos como atributos.
- Attr: representa el par nombre-de-atributo/valor.
- Text: almacena el contenido del texto que se encuentra entre una etiqueta de apertura y una de cierre. También almacena el contenido de una sección de tipo CDATA.
- CDataSection: es el nodo que representa una sección de tipo <![CDATA[ ]]>.
- Comment: representa un comentario de XML.

Como hemos comentado DOM crea un árbol de la página web y cada elemento es un nodo y los nodos son de un tipo determinado. Aquí tienes la lista de tipos de nodos. Están definidas como constantes.

- Node.ELEMENT_NODE = 1
- Node.ATTRIBUTE_NODE = 2
- Node.TEXT_NODE = 3
- Node.CDATA_SECTION_NODE = 4
- Node.ENTITY_REFERENCE_NODE = 5
- Node.ENTITY_NODE = 6
- Node.PROCESSING_INSTRUCTION_NODE = 7
- Node.COMMENT_NODE = 8
- Node.DOCUMENT_NODE = 9
- Node.DOCUMENT_TYPE_NODE = 10
- Node.DOCUMENT_FRAGMENT_NODE = 11
- Node.NOTATION_NODE = 12

Ahora podemos ver las propiedades y métodos de Node

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 nodoanteriorNodo dentro de la lista childNodes

Para realizar estos ejercicios tenemos que saber diferentes propiedades y métodos de los nodos de tipo Document y Element.

Del DOM Document ya hemos visto en los apunte *getElementById(), getElementsByName(), getElementsByTagName(), getElementsByClassName()*.

DOM Element tenemos que ver dos propiedades:

- style → Propiedad que nos permite asignar u obtener el atributo style de un elemento. Esta propiedad a su vez tiene propiedades que son las propiedades de CSS. Por tanto, para poder acceder a una propiedad de CSS de una etiqueta hay que utilizar la siguiente sintaxis:

  variable = document.getElementById("id").style.*propiedad*;
  document.getElementById("id").style.*propiedad* = *"valor"*;

  Hay que tener en cuenta la siguiente lista para poder acceder a las propiedades de CSS.

Propiedad CSS	Sintaxis JavaScript	Propiedad CSS	Sintaxis JavaScript
background	background	background-attachment	backgroundAttachment
background-color	backgroundColor	background-image	backgroundImage
background-position	backgroundPosition	background-repeat	backgroundRepeat
border	border	border-bottom	borderBottom
border-bottom-color	borderBottomColor	border-bottom-style	borderBottomStyle
border-bottom-width	borderBottomWidth	border-color	borderColor
border-left	borderLeft	border-left-color	borderLeftColor
border-left-style	borderLeftStyle	border-left-width	borderLeftWidth
border-right	borderRight	border-right-color	borderRightColor
border-right-style	borderRightStyle	border-right-width	borderRightWidth
border-style	borderStyle	border-top	borderTop
border-top-color	borderTopColor	border-top-style	borderTopStyle
border-top-width	borderTopWidth	border-width	borderWidth
clear	clear	clip	clip
color	color	cursor	cursor
display	display	filter	filter

Propiedad CSS	Sintaxis JavaScript	Propiedad CSS	Sintaxis JavaScript
float	cssFloat	font	font
font-family	fontFamily	font-size	fontSize
font-variant	fontVariant	font-weight	fontWeight
height	height	left	left
letter-spacing	letterSpacing	line-height	lineHeight
list-style	listStyle	list-style-image	listStyleImage
list-style-position	listStylePosition	list-style-type	listStyleType
margin	margin	margin-bottom	marginBottom
margin-left	marginLeft	margin-right	marginRight
margin-top	marginTop	overflow	overflow
padding	padding	padding-bottom	paddingBottom
padding-left	paddingLeft	padding-right	paddingRight
padding-top	paddingTop	page-break-after	pageBreakAfter
page-break-before	pageBreakBefore	position	position
text-align	textAlign	text-decoration	textDecoration
text-indent	textIndent	text-transform	textTransform
top	top	vertical-align	verticalAlign
visibility	visibility	width	width
z-index	zIndex		

http://www.aprenderaprogramar.es/index.php?option=com_content&view=article&id=807:cambiar-css-con-javascript-lista-o-tabla-de-equivalencias-de-propiedades-css-js-camelcase-cu01129e&catid=78:tutorial-basico-programador-web-javascript-desde-&Itemid=206

- value → Si estamos trabajando con de un formulario nos devuelve el valor del campo.
- className → Propiedad que nos permite asignar y obtener una clase definida en la hoja de estilos.

  document.getElementById("id").className = "clase";
  variable = document.getElementById("id").className;