


---

## Document Object Model

Roberto Gómez Cárdenas  
rogomez@itesm.mx  
<http://homepage.cem.itesm.mx/rogomez>

Lámina 1 Roberto Gómez C.



---

## ¿Qué es DOM?

- Estándar de la W3C
- Acceso a documentos como XML y HTML
- Dividido en tres niveles
  - Core DOM
  - XML DOM
  - HTML DOM

Lámina 2 Roberto Gómez C.



## Nodos DOM

- DOM transforma los documentos XHTML en un conjunto de elementos denominados nodos, que se encuentran interconectados y que representan los contenidos de las páginas web y las relaciones entre ellos.
- Todo documento HTML cargado en un navegador se convierte en un objeto de tipo Documento.
- Este objeto proporciona acceso a cada uno de sus elementos HTML desde un script.
- El objeto Documento es parte del documento Window y puede ser accedido a través de la propiedad `window.document`.

Lámina 3

Roberto Gómez C.



## Ejemplo

```
<!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>
```

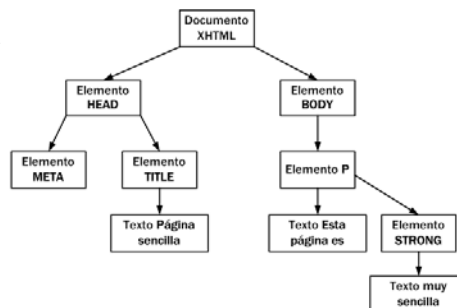



Lámina 4

Roberto Gómez C.



## Tipos de nodos


---

- Especificación completa define 12 tipos de nodo.
- Entre los más usados:

Nodo	Descripción
Document	nodo raíz del que derivan todos los demás nodos del árbol.
Element	representa cada una de las etiquetas XHTML. Se trata del único nodo que puede contener atributos y el único del que pueden derivar otros nodos.
Attr	se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas XHTML, es decir, uno por cada par atributo=valor.
Text	nodo que contiene el texto encerrado por una etiqueta XHTML.
Comment	representa los comentarios incluidos en la página XHTML.

- Otros tipos: DocumentType, CDataSection, DocumentFragment, Entity, EntityReference, ProcessingInstruction y Notation.

Lámina 5
Roberto Gómez C.



## Accediendo a los nodos

---

- El acceso solo se puede hacer cuando el árbol se ha construido completamente, es decir cuando página se cargo completamente.
- Dos opciones
  - A través de sus nodos padre.
  - Acceso directo.
    - `getElementsByTagName()`
    - `getElementsByName()`
    - `getElementById()`

Lámina 6
Roberto Gómez C.



## getElementsByTagName( )

- **getElementsByTagName(nombre-etiqueta)**
  - Obtiene todos los elementos de la página XHTML cuya etiqueta sea igual que el parámetro que se le pasa a la función.

- Ejemplo

```
var parrafos = document.getElementsByTagName("p");
```

- Se obtienen todos los párrafos de una página XHTML.
- El primer párrafo se puede referenciar de la siguiente forma:

```
var primerParrafo = parrafos[0];
```

Lámina 7

Roberto Gómez C.



## getElementsByTagName( )

- Ejemplo

```
var parrafos = document.getElementsByTagName("p");
```

- Para recorrer todos los párrafos de la página con el siguiente código:

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


- La función se puede aplicar de forma recursiva sobre cada uno de los nodos devueltos por la función.

- Ejemplo: Se obtienen todos los enlaces del primer párrafo de la página:

```
var parrafos = document.getElementsByTagName("p");
var primerParrafo = parrafos[0];
var enlaces = primerParrafo.getElementsByTagName("a");
```

Lámina 8

Roberto Gómez C.




## getElementsByTagName()

- `getElementsByTagName()`
  - La función `getElementsByTagName()` es similar a la anterior, pero en este caso se buscan los elementos cuyo atributo `name` sea igual al parámetro proporcionado.
  - Ejemplo

```
var parrafoEspecial = document.getElementsByTagName("especial");

<p name="prueba">...</p>
<p name="especial">...</p>
<p>...</p>
```

Lámina 9 Roberto Gómez C.



## getElementById()

- `getElementById()`
  - Devuelve el elemento XHTML 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.
  - Ejemplo

```
var cabecera = document.getElementById("cabecera");

<div id="cabecera">
  <a href="/" id="logo">...</a>
</div>
```

Lámina 10 Roberto Gómez C.



## Creando elementos

- Crear y añadir a la página un nuevo elemento XHTML sencillo consta de cuatro pasos diferentes:
  1. Creación de un nodo de tipo **Element** que represente al elemento.  
`var parrafo = Document.createElement(`
  2. Creación de un nodo de tipo **Text** que represente el contenido del elemento.  
`var contenido = Documento.createTextNode( "Esto es una prueba")`
  3. Añadir el nodo **Text** como nodo hijo del nodo **Element**.  
`parrafo.appendChild( contenido)`
  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.  
`document.body.appendChild(parrafo)`

Lámina 11

Roberto Gómez C.



## Borrando elementos

- 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.
- Ejemplo

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

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

Lámina 12

Roberto Gómez C.



## Acceso directo a los atributos XHTML

- 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.

Lámina 13

Roberto Gómez C.



## Ejemplo


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

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

- Se obtiene el atributo href del enlace mediante `enlace.href`.
- Para obtener el atributo id, se utilizaría `enlace.id`.

Lámina 14


Roberto Gómez C.



## Ejemplo cambio atributo

```
<html>
<head>
<script type="text/javascript">
function cambioReferencia() {
document.getElementById("ref1").href="http://www.securityfocus.com";
}
</script>
</head>
<body>
Acceda a la siguiente pagina:
<br> <br>
<a id="ref1" href="http://homepage.cem.itesm.mx/rogomez">
Referencia pagina </a>
<br> <br> <br>
<input type="button" onclick="cambioReferencia()"
value="Cambiar la referencia" />
</body>
</html>
```

Lámina 15 Roberto Gómez C.



## Propiedades básicas formularios y elementos

- Cuando se carga una página web, el navegador crea automáticamente un arreglo llamado forms y que contiene la referencia a todos los formularios de la página.
- Para acceder al arreglo se utiliza el objeto document:
  - document.forms es el arreglo que contiene todos los formularios de la página.

document.forms[0]

Lámina 16 Roberto Gómez C.





## Elementos del arreglo

- Además del arreglo de formularios, el navegador crea un arreglo llamado `elements` por cada uno de los formularios de la página.
- Cada arreglo `elements` contiene la referencia a todos los elementos (cuadros de texto, botones, listas desplegables, etc.) de ese formulario.

```
document.forms[0].elements[0];
```

- obtiene el primer elemento del primer formulario de la página.

```
document.forms[0].elements[document.forms[0].elements.length-1];
```

- obtiene el último elemento del primer formulario de la página.

Lámina 17

Roberto Gómez C.



## Inconveniente arreglos

- ¿Qué sucede si cambia el diseño de la página y en el código HTML se cambia el orden de los formularios originales o se añaden nuevos formularios?
  - *"el primer formulario de la página"* ahora podría ser otro formulario diferente al que espera la aplicación.
- Una forma de evitar los problemas del método anterior consiste en acceder a los formularios de una página a través de su nombre (atributo `name`) o a través de su atributo `id`.

Lámina 18

Roberto Gómez C.



## Ejemplo acceso vía atributo name

- El objeto document permite acceder directamente a cualquier formulario mediante su atributo name:

```
var formularioPrincipal = document.formulario;  
var formularioSecundario = document.otro_formulario;
```

```
<form name="formulario" >
```

```
...
```

```
</form>
```

```
<form name="otro_formulario" >
```

```
...
```

```
</form>
```

Lámina 19

Roberto Gómez C.



## Accediendo elementos

- Los elementos de los formularios también se pueden acceder directamente mediante su atributo name

```
var formularioPrincipal = document.formulario;  
var primerElemento = document.formulario.elemento;
```


```
<form name="formulario">
```

```
  <input type="text" name="elemento" />
```

```
</form>
```

Lámina 20

Roberto Gómez C.



## Usando DOM para acceder formularios

---

- Posible acceder a los formularios y a sus elementos usando funciones DOM de acceso directo a los nodos.
- Ejemplo uso `document.getElementById()` para acceder a un formulario y a uno de sus elementos:


```

var formularioPrincipal = document.getElementById("formulario");
var primerElemento = document.getElementById("elemento");

<form name="formulario" id="formulario" >
  <input type="text" name="elemento" id="elemento" />
</form> >

```

Lámina 21
Roberto Gómez C.




## Propiedades

---

- Independientemente del método utilizado para obtener la referencia a un elemento de formulario, cada elemento dispone de las siguientes propiedades útiles para el desarrollo de las aplicaciones:

Propiedad	Descripción
type	indica el tipo de elemento que se trata
form	es una referencia directa al formulario al que pertenece el elemento.
name	obtiene el valor del atributo name de XHTML
value	permite leer y modificar el valor del atributo value de XHTML


Lámina 22
Roberto Gómez C.



## Eventos más usados en formularios

Evento	Descripción
onclick	evento que se produce cuando se pincha con el ratón sobre un elemento.
onchange	evento que se produce cuando el usuario cambia el valor de un elemento de texto
onfocus	evento que se produce cuando el usuario selecciona un elemento del formulario.
onblur	evento complementario de onfocus, ya que se produce cuando el usuario ha <i>deseleccionado</i> un elemento por haber seleccionado otro elemento del formulario

Lámina 23 Roberto Gómez C.




## Obtener valor campos formulario cuadro texto y textarea

- El valor del texto mostrado por estos elementos se obtiene y se establece directamente mediante la propiedad value.

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

Lámina 24 Roberto Gómez C.



## Ejemplo uso

---

```

<html>
<head>
<title> Ejemplo acceso formas en DOM </title>
<script type="text/javascript">
function suma(valor) {
    var x = document.getElementById("x").value;
    var y = document.getElementById("y").value;
    document.write("Los valores son:" + x + " y " + y + "<br>");
    r = Number(x) + Number(y);
    document.write("El valor de la suma es: " + r);
}
</script>
</head>
<body>
Proporcione los siguientes valores: <br><br>
Valor de x <input type="text" id="x" /> <br>
Valor de y <input type="text" id="y" /> <br><br>
<input type="button" value="Enviar datos" onclick="suma()" \>
</body>
</html>

```

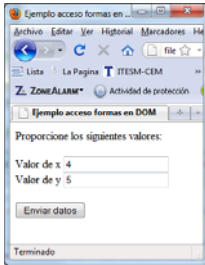
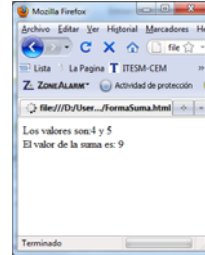




Lámina 25
Roberto Gómez C.



## Radiobutton

---

- Se puede conocer los *radiobuttons* se ha seleccionado.
- La propiedad `checked` devuelve `true` para el *radiobutton* seleccionado y `false` en cualquier otro caso.
- Ejemplo:
 

```

<input type="checkbox" value="conds" name="condiciones" id="condiciones"/>
He leído y acepto las condiciones
<input type="checkbox" value="priv" name="privacidad" id="privacidad"/>
He leído la política de privacidad

```
- Determinando si cada checkbox ha sido seleccionado
 


```

var elemento = document.getElementById("condi");
alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);

elemento = document.getElementById("priv");
alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);

```

Lámina 26
Roberto Gómez C.




## Select

- Las listas desplegables (<select>) son los elementos en los que es más difícil obtener su valor.
- Considerar

```
<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>
```
- Lo que se requiere es obtener el valor del atributo value de la opción (<option>) seleccionada por el usuario .
- Se deben usar varias propiedades.

Lámina 27 Roberto Gómez C.



## Obteniendo

- Se deben usar las siguientes propiedades.
  - Options: arreglo que contiene la referencia a todas las opciones de la lista. La primera opción de la lista se puede obtener mediante  
`document.getElementById("id_de_la_lista").options[0].`
  - selectedIndex: guarda el índice de la opción seleccionada. El índice hace referencia al array options creado automáticamente por el navegador para cada lista.

Lámina 28 Roberto Gómez C.



## Ejemplo

```
// 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];

// 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);
```

Lámina 29

Roberto Gómez C.



## Otra opción

- Se pueden abreviar todos los pasos necesarios en una única instrucción.

```
var lista = document.getElementById("opciones");


// Obtener el valor de la opción seleccionada
var valorSeleccionado = lista.options[lista.selectedIndex].value;

// Obtener el texto que muestra la opción seleccionada
var valorSeleccionado = lista.options[lista.selectedIndex].text;
```

- Nota: no confundir el valor de la propiedad selectedIndex con el valor correspondiente a la propiedad value de la opción seleccionada

Lámina 30

Roberto Gómez C.




## Evitar envío duplicado

---

- Un usuario puede pulsar dos veces seguidas sobre el botón "Enviar".
  - Si la conexión del usuario es demasiado lenta o la respuesta del servidor se hace esperar, el formulario original sigue mostrándose en el navegador y por ese motivo, el usuario tiene la tentación de volver a pinchar sobre el botón de "Enviar".
- Una buena práctica en el diseño de aplicaciones web suele ser la de deshabilitar el botón de envío después de la primera pulsación

Lámina 31 Roberto Gómez C.



## Código evitar envío duplicados

---

```


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

```

- En primer lugar, se deshabilita el botón mediante la instrucción `this.disabled = true;`
  - Es la única instrucción necesaria si sólo se quiere deshabilitar un botón.
- A continuación, se cambia el mensaje que muestra el botón. Del original "Enviar" se pasa al más adecuado "Enviando..."
- Por último, se envía el formulario mediante la función `submit()` en la siguiente instrucción: `this.form.submit()`

Lámina 32 Roberto Gómez C.






## Ejemplo uso

```
<html>
<head>
</head>
<body>
Proporcione los siguientes valores:
<br><br>
Valor de x <input type="text" id="x" /> <br>
Valor de y <input type="text" id="y" /> <br>
<br> <br>
<input type="button" value="Enviar Datos" onclick="this.disabled=true;
this.value='Datos Enviados'; />
</html>
```


Lámina 33 Roberto Gómez C.



## Validación

- Antes de enviar un formulario al servidor, se recomienda validar mediante JavaScript los datos insertados por el usuario.
- Si el usuario ha cometido algún error al rellenar el formulario, se le puede notificar de forma instantánea, sin necesidad de esperar la respuesta del servidor.
- Normalmente, la validación de un formulario consiste en llamar a una función de validación cuando el usuario pulsa sobre el botón de envío del formulario.

Lámina 34 Roberto Gómez C.




## Código básico

---

- Para incorporar la validación a un formulario:
 

```
<form action=" " method=" " id=" " name=" "
                                onsubmit="return validacion() ">
...
</form>
```
- Se manda llamar la función validacion() que es la que llevará a cabo la validación.
  - Si el evento onsubmit devuelve el valor true, el formulario se envía como lo haría normalmente.
  - Si el evento onsubmit devuelve el valor false, el formulario no se envía.

Lámina 35
Roberto Gómez C.




## Esquema función validación

---

```
function validacion() {
  if (condicion que debe cumplir el primer campo del formulario) {
    // Si no se cumple la condicion...
    alert('[ERROR] El campo debe tener un valor de...');
    return false;
  }
  else if (condicion que debe cumplir el segundo campo del formulario) {
    // Si no se cumple la condicion...
    alert('[ERROR] El campo debe tener un valor de...');
    return false;
  }
  ...
  else if (condicion que debe cumplir el último campo del formulario) {
    // Si no se cumple la condicion...
    alert('[ERROR] El campo debe tener un valor de...');
    return false;
  }

  // Si el script ha llegado a este punto, todas las condiciones
  // se han cumplido, por lo que se devuelve el valor true
  return true;
}
```

Lámina 36
Roberto Gómez C.



## Validando un campo de texto obligatorio

---


- Se comprueba que el valor introducido sea válido, que el número de caracteres introducido sea mayor que cero y que no se hayan introducido sólo espacios en blanco.

```

valor = document.getElementById("campo").value;
if ( valor == null || valor.length == 0 || /\s+$/i.test(valor) ) {
    return false;
}

```

Lámina 37 Roberto Gómez C.



## Validando un campo de texto con valores numérico

---

- Validar que se introdujo un valor numérico en un cuadro de texto.

```

valor = document.getElementById("campo").value;
if( isNaN(valor) ) {
    return false;
}

```


- Ejemplo resultados función isNaN()

```

isNaN(3);           // false
isNaN("3");        // false
isNaN(3.3545);     // false
isNaN(32323.345);  // false
isNaN(+23.2);      // false
isNaN("-23.2");    // false
isNaN("23a");      // true
isNaN("23.43.54"); // true

```

Lámina 38 Roberto Gómez C.



## Ejemplo validación rangos de números

---

```

<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript">
function DataCheck()
{
  var x = document.getElementById("x").value;
  var y = document.getElementById("y").value;


  if ( isNaN(x) || isNaN(y) ) {
    alert("Los datos deben ser numericos");
    return false;
  }
  if ( (Number(x) < 0) || ( Number(x) > 100 ) ) {
    alert("El valor de x es invalido");
    return false;
  }
  if ( (Number(y) < 0) || ( Number(y) > 100 ) ) {
    alert("El valor de y es invalido");
    return true;
  }
  alert("Datos Validos");
  return true;
}
</SCRIPT>
</HEAD>
  
```

```

<BODY>
<FORM NAME="forma"  onSubmit="return DataCheck()" >
  Valor de x (entre 1 y 100): <INPUT ID="x" SIZE=3>
  <BR> <BR>
  Valor de y (entre 1 y 100): <INPUT ID="y" SIZE=3>
  <BR> <BR> <BR>
  <INPUT TYPE=SUBMIT VALUE="Submit">
</FORM>
</BODY>
</HTML>
  
```

Roberto Gómez C.

Lámina 39



## Validar que se seleccionó una opción de una lista

---

- Validar que el usuario seleccionó un elemento de una lista despegable.

```


indice = document.getElementById("opciones").selectedIndex;
if( indice == null || indice == 0 ) {
  return false;
}
  
```

```

<select id="opciones" name="opciones">
  <option value="">- Selecciona un valor -</option>
  <option value="1">Primer valor</option>
  <option value="2">Segundo valor</option>
  <option value="3">Tercer valor</option>
</select>
  
```

Roberto Gómez C.

Lámina 40



## Validar que un checkbox se seleccionó

---

- Validar que se seleccionó un elemento de tipo checkbox.
 


```

valor = document.getElementById("campo").value;
if( isNaN(valor) ) {
    return false;
}
      
```
- Si es necesario comprobar que todos los *checkbox* del formulario han sido 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;
        }
    }
}
      
```

Lámina 41
Roberto Gómez C.



## Validar que se seleccionó un radiobutton

---

- En general, la comprobación que se realiza es que el usuario haya seleccionado algún *radiobutton* de los que forman un determinado grupo.
 


```

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;
}
      
```

Lámina 42
Roberto Gómez C.



---

## Document Object Model


Roberto Gómez Cárdenas

rogomez@itesm.mx

<http://homepage.cem.itesm.mx/rogomez>

Lámina 43

Roberto Gómez C.



---

Lámina 44

Roberto Gómez C.



- [http://www.librosweb.es/javascript/capitulo5/ejercicios\\_sobre\\_dom.html](http://www.librosweb.es/javascript/capitulo5/ejercicios_sobre_dom.html)
- [http://www.w3schools.com/html/dom/dom\\_nodes\\_access.asp](http://www.w3schools.com/html/dom/dom_nodes_access.asp)
- <http://www.learn-javascript-tutorial.com/JavaScriptBasics.cfm>