

DESARROLLO WEB EN ENTORNO CLIENTE TEMA4: MODELO DEL OBJETO DOCUMENTO

Índice

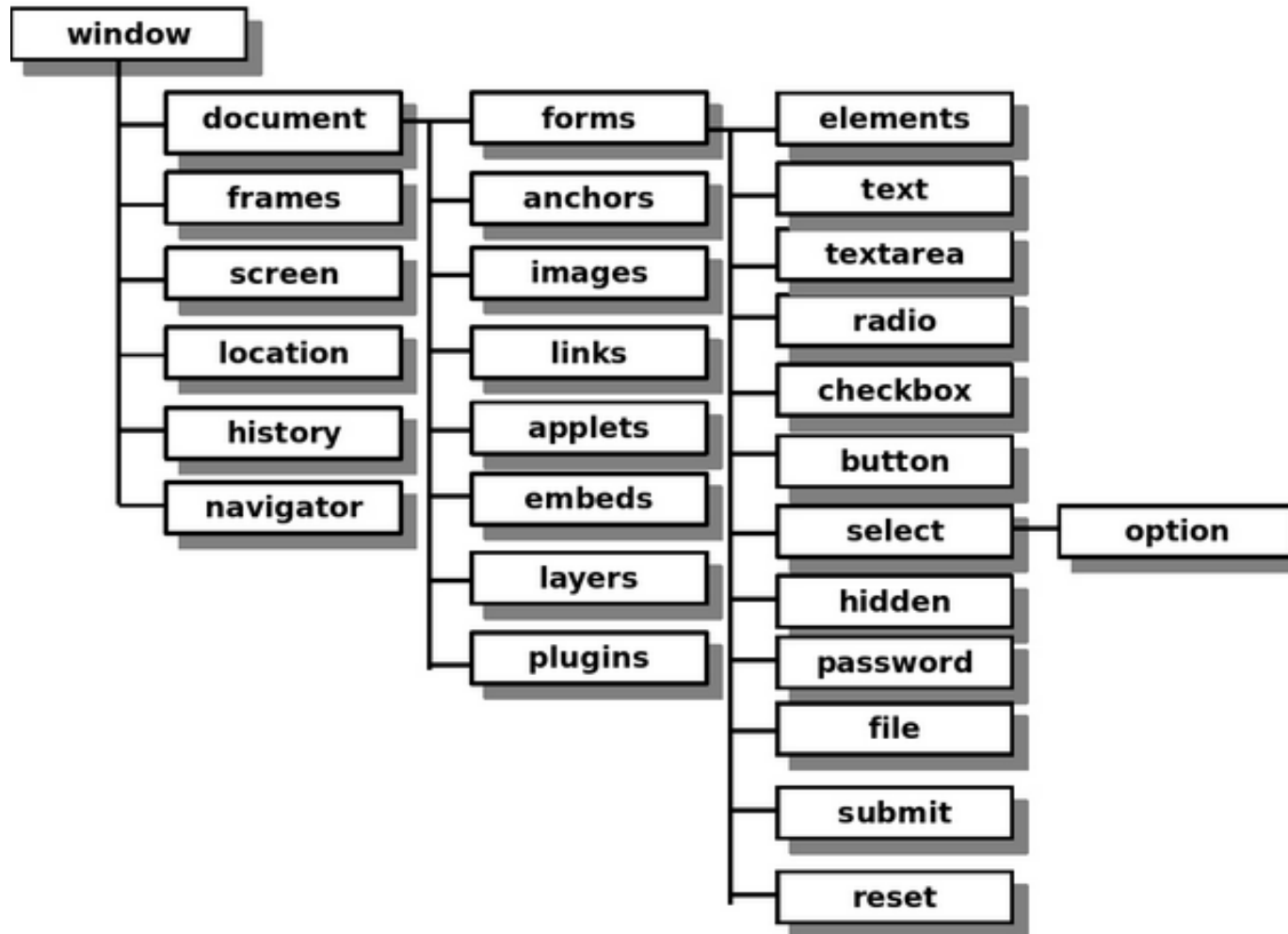
- Interacción de los objetos con el navegador
- Generación de elementos HTML desde código
- Aplicaciones prácticas de los marcos
- Gestión de las ventanas

1. Introducción

- El Modelo del Objeto Documento, más conocido como **DOM**, *permite representar la estructura lógica de una página HTML o documento XML mediante una jerarquía de objetos.*
- Sirve de interfaz a través de la cual se definen reglas de acceso que *permiten manipular el contenido del documento.*
- Toda esta funcionalidad está encapsulada en *objetos que exponen propiedades, métodos y eventos que permiten que estos sean manipulados.*

1. Introducción

4



1. Introducción

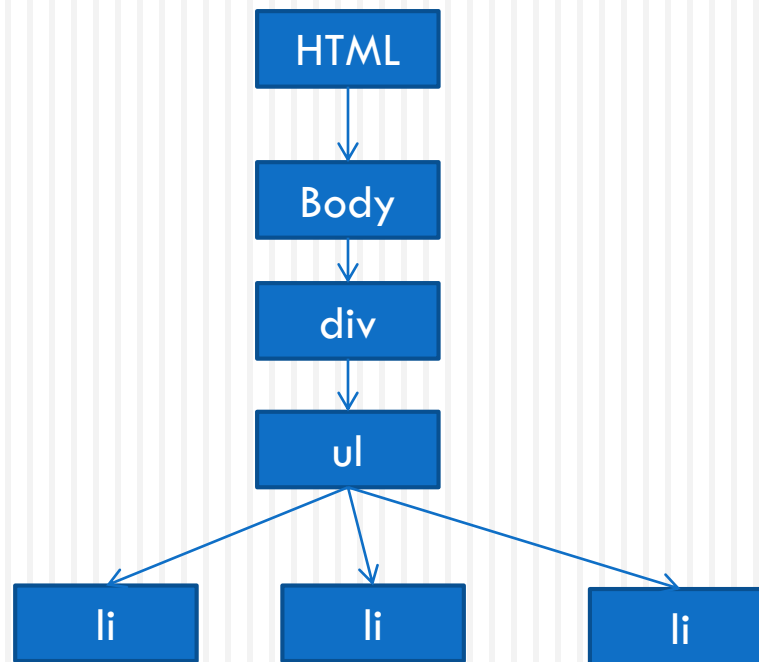
- El navegador, en un segundo plano, se encarga de mantener el documento y el DOM sincronizados, de modo que, *cualquier modificación en el DOM a través de JavaScript se trasladará automáticamente a la presentación del documento.*
- DOM estandarizado a través del W3C.

2. Modelo Objeto del documento

6

- A la hora de interpretar el DOM, éste es representado normalmente en estructura de árbol:

```
<html>
<body>
  <div>
    <ul>
      <li>Propiedades</li>
      <li>Colecciones</li>
      <li>Métodos</li>
    </ul>
  </div>
</body>
</html>
```



3. Interacción de los objetos con el navegador

7

- A continuación, veremos la lista de objetos del DOM que representan las características de cada navegador: Navigator, Screen, Window, Document, History, Location.
- Estos objetos se conocen como **BOM** – Browser Object Model.

3. Interacción de los objetos con el navegador

8

1. Window:

- ❑ Se considera el objeto más importante de JavaScript e implícito, ya que todos los demás objetos están debajo de su jerarquía.
- ❑ Permite gestionar las ventanas del navegador.
- ❑ Es un objeto implícito, con lo cual no es necesario nombrarlo para acceder a los objetos que se encuentran debajo de su nivel de jerarquía.
- ❑ Siempre existirá un objeto windows por pestaña.
- ❑ Si además existen objetos frames insertados dentro del documento cada uno de estos poseerá su propio objeto window.

3. Interacción de los objetos con el navegador

9

Métodos

alert()	forward()	setInterval()
back()	home()	setTimeout()
blur()	moveTo()	scrollBy()
close()	open()	scrollTo()
confirm()	print()	stop()
find()	prompt()	setInterval()
focus()	resizeTo()	setTimeout()

Propiedades

closed	location	pageYoffset
defaultStatus	locationbar	parent
document	menubar	personalbar
frames	name	scrollbars
history	opener	self
innerHeight	outerHeight	status
innerWidth	outerWidth	toolbar
length	pageXoffset	top

3. Interacción de los objetos con el navegador

10

- `Window.open(URL, name, specs, replace):`
 - ❑ Permite abrir una nueva ventana y cargarla en el documento actual, dentro de un frame por ejemplo.
 - ❑ También se puede abrir otra ventana nueva del navegador y cargar allí el documento deseado.
 - ❑ Devuelve una referencia del objeto creado.

3. Interacción de los objetos con el navegador

11

➤ Ejemplo:

```
<html><head></head><body>
.....
<h1> Ejemplo de Apariencia de una Ventana</h1>
<br><input type="Button" value="Abre una Ventana" onclick="
myWindow1=window.open('', 'Nueva Ventana', 'width=300, height=200');
myWindow1.document.write('<html>');
myWindow1.document.write('<head>');
myWindow1.document.write('<title>Ventana Test</title>');
myWindow1.document.write('</head>');
myWindow1.document.write('<body>');
myWindow1.document.writeln('Se usan las propiedades: ');
myWindow1.document.write('<li>height=200</li> <li>width=300</li>');
myWindow1.document.write('</body>');
myWindow1.document.write('</html>');"/>
</body></html>
.....
```

3. Interacción de los objetos con el navegador

12

- Apariencia de las ventanas:
 - ❑ La ventanas cuentan con propiedades que permiten decidir su tamaño, ubicación o los elementos que contendrá.

Propiedades	
directories	scrollbars
height	status
menubar	toolbar
resizable	width

3. Interacción de los objetos con el navegador

13

- Comunicación entre ventanas:
 - ❑ Desde una ventana se pueden abrir o cerrar nuevas ventanas.
 - ❑ La primera se denomina ventana principal, mientras que las segundas se denominan ventanas secundarias.
 - ❑ Desde la ventana principal se puede acceder a las ventanas secundarias.

3. Interacción de los objetos con el navegador


14

➤ Ejemplo:

```
<html>
<head>
  <meta http-equiv="content-type"
        content="text/html; charset=utf-8">
  <title>Comunicación entre ventanas</title>
  <script type="text/javascript">
    var ventanaSecundaria = window.open("", "ventanaSec", "width=500 , height=500");
  </script>
</head>
<body>
  <h1> Comunicación entre ventanas </h1><br>
  <form name=formulario>
    <input type="text" name="url" size="50" value="http://www.">
    <input type="button" value="Mostrar URL en ventana secundaria"
      onclick="ventanaSecundaria.location = document.formulario.url.value;">
  </form>
</body>
</html>
```


3. Interacción de los objetos con el navegador

15



A.4.1. Abrir una ventana con la dirección www.larioja.com en su navegador. Dicha ventana deberá ser una pestaña diferente (_blank) al documento actual.

```
<script type="text/javascript">  
    window.open("http://www.larioja.com","", "_blank");  
</script>
```



A.4.2. Abre una nueva ventana con la dirección www.jesuitasrioja.org en tu navegador, dicha ventana deberá ser una pestaña diferente al documento actual y podrá ser cerrada desde la ventana principal, para ello utilizar `window.confirm()` y `window.close()`.

3. Interacción de los objetos con el navegador

16



A.4.3. En un documento HTML crea un botón que al presionarlo que al presionarlo abra una nueva ventana (100x100). Crea otro botón para redimensionar dicha ventana mediante *resizeTo(500,500)*.

3. Interacción de los objetos con el navegador

17



A.4.4. Desde el documento principal, abre una ventana vacía en tu navegador. Dicha ventana deberá tener un ancho de 300 píxeles y una altura de 200 píxeles. Después crea un método llamado *desplazarVentana()* que desplace la nueva ventana de forma aleatoria. En dicho método debes imprimir las nuevas coordenadas generadas.

3. Interacción de los objetos con el navegador

18


- ❑ Ejemplo: *barra de progreso mediante ejecución de método a intervalos regulares.*



```
<script type="text/javascript">
  var idIntervalo=setInterval(intervaloReloj,500);
  function intervaloReloj(){
    var d= new Date();
    var t= d.toLocaleTimeString();
    document.getElementById("reloj").value=t;
    document.getElementById("progreso").style.width=document.getElementById("progreso").clientWidth +15;
    console.log(document.getElementById("progreso").clientWidth);
    console.log(document.getElementById("progreso").style.width);
    document.getElementById("txtProgreso").value=document.getElementById("progreso").style.width;
    if (document.getElementById("progreso").clientWidth>500) {
      document.getElementById("progreso").style.width=0;
    }
  }
}</script>
```



```
<body>
  <h2>Métodos del objeto Window</h2>
  <input type="text" id="reloj" disabled>
  <input type="text" id="txtProgreso" disabled>
  <div id="progreso" style="width:10px;height:30px;background-color:green;margin-top:10px">Progreso</div>
  <button id="btnParar" onclick="window.clearInterval(idIntervalo);">Stop</button>
</body>
```



3. Interacción de los objetos con el navegador

19

- Aplicaciones prácticas de los marcos (frames/iframes): Cuidado!!!
- En la actualidad se desaconseja el uso de marcos por estar en contra de varios principios web:
- Ver: <https://www.eniun.com/marcos-frames-html5/>
 - ❑ Es posible dividir la ventana de una aplicación web en dos o más partes independientes.
 - ❑ Con JavaScript se puede interactuar entre estos sectores independientes.
 - ❑ Dichos sectores se denominan marcos.
 - ❑ Los objetos tipo frame pueden ser accedidos mediante el objeto window.

3. Interacción de los objetos con el navegador

20

- ❑ Algunas páginas web presentan una estructura en la cual una parte permanece fija mientras que otra va cambiando.
- ❑ Por ejemplo la página de la API de Java:

The screenshot shows a web browser displaying the Java Platform Standard Ed. 7 API documentation. The page is titled "Class AbstractButton" and is part of the "javax.swing" package. The left sidebar contains a navigation menu with "All Classes" and "Packages" sections. The main content area shows the class hierarchy, implemented interfaces, and subclasses. The page is structured with a fixed header and a main content area that changes based on the selected class.

Java™ Platform Standard Ed. 7

Overview Package **Class** Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

javax.swing

Class AbstractButton

java.lang.Object
java.awt.Component
java.awt.Container
javax.swing.JComponent
javax.swing.AbstractButton

All Implemented Interfaces:

ImageObserver, ItemSelectable, MenuContainer, Serializable, SwingConstants

Direct Known Subclasses:

JButton, JMenuItem, JToggleButton

public abstract class AbstractButton
extends JComponent
implements ItemSelectable, SwingConstants

Defines common behaviors for buttons and menu items.

Buttons can be configured, and to some degree controlled, by Actions. Using an Action with a button has many benefits beyond directly configuring a button. Refer to Swing Components Supporting Action for more details, and you can find more information in How to Use Actions, a section in *The Java Tutorial*.

For further information see How to Use Buttons, Check Boxes, and Radio Buttons, a section in *The Java Tutorial*.

Warning: Serialized objects of this class will not be compatible with future Swing releases. The current serialization support is appropriate for short term storage or RMI between applications running the same version of Swing. As of 1.4, support for long term storage of all JavaBeans™ has been added to the java.beans package. Please see XMLEncoder.

Nested Class Summary

Nested Classes

3. Interacción de los objetos con el navegador

21

- ❑ Los marcos se definen utilizando HTML mediante estas etiquetas:

`<frameset>`

`<frame>`

- ❑ Atributos de la etiqueta `<frame>`:

Atributos

frameborder

marginheight

marginwidth

name

noresize

scrolling

src

3. Interacción de los objetos con el navegador

22

➤ Ejemplo:

1

```
<html><head><title>Ejemplos de control de marcos</title></head>
  <frameset cols="50%,50%">
    <frame src="Marco1.html" name="Marco1" noresize>
    <frame src="Marco2.html" name="Marco2" noresize>
  </frameset>
  <body></body>
</html>
```

2

```
<html><body>
  <form name="form1">
    <select name="color">
      <option value="green">Verde
      <option value="blue">Azul
    </select><br><br>
    <select name="marcos">
      <option value="0">Izquierda
      <option value="1">Derecha
    </select>
  </form>
</body></html>
```

3

```
<html><body><form>
  <input type="Button" value="Cambiar Color" onclick="
    campoColor = parent.Marco1.document.form1.color
    if(campoColor.selectedIndex==0){
      colorin = 'green';
    }
    else{
      colorin = 'blue';
    }
    campoFrame = parent.Marco1.document.form1.marcos
    if(campoFrame.selectedIndex==0){
      window.parent.Marco1.document.bgColor = colorin
    }else{
      window.parent.Marco2.document.bgColor = colorin
    }">
</form></body></html>
```

3. Interacción de los objetos con el navegador

23



A.4.5. El ejemplo anterior no está hecho para HTML5. Cámbialo para usar *iframes* en lugar de *frames*. Debe tener la misma funcionalidad.

3. Interacción de los objetos con el navegador

24

2. Location :

- ❑ Corresponde a la URL de la página web en uso.
- ❑ Su principal función es la de consultar las diferentes partes que forman una URL como por ejemplo:

- El dominio.
- El protocolo.
- El puerto.

Métodos

`assign()`

`reload()`

`replace()`

Propiedades

`hash`

`host`

`hostname`

`href`

`pathname`

`port`

`protocol`

`search`

3. Interacción de los objetos con el navegador

25


```
<script type="text/javascript">
    document.write("<br>URL completa: "+ location.href);
    document.write("<br>Pathname: "+ location.pathname);
    document.write("<br>Protocolo: "+ location.protocol);

</script>
<form>
    <input type="button" value="Recargar" onClick="location.reload();">
</form>
```

3. Interacción de los objetos con el navegador

26

CUIDADO !!!



A.4.6. Crea un documento en el que se incluya una caja de texto, destinada a introducir nuevas direcciones URL, y un *iframe* en donde se cargará el contenido de la nueva URL.

Cross Origin
framing



3. Interacción de los objetos con el navegador

27

3. History:

- ❑ Almacena las referencias de las páginas web visitadas.
- ❑ Las referencias se guardan en una lista utilizada principalmente para desplazarse entre dichas páginas web.
- ❑ No es posible acceder a los nombres de las URL, ya que es información privada, pero sí al índice para moverse a través del mismo.

3. Interacción de los objetos con el navegador

28

Métodos

`back()`

`forward()`

`go()`

Propiedades

`current`

`length`

`next`

`previous`

3. Interacción de los objetos con el navegador

29

```
<form>
  <input type="button" value="Atrás" onClick="history.back();">
  <input type="button" value="Adelante" onClick="history.forward();">
</form>
```

3. Interacción de los objetos con el navegador

30

4. Navigator:

- ❑ Permite identificar las características de la plataforma sobre la cual se ejecuta la aplicación web. Ejemplo:

- Tipo de navegador.
- Versión del navegador.
- Sistema operativo.

Métodos

`javaEnable()`

Propiedades

`appName`

`appVersion`

`cookieEnable`

`platform`

`userAgent`

3. Interacción de los objetos con el navegador

31

```
<script type="text/javascript">
  document.write("Navegador: " + navigator.appName + "</br>");
  document.write("Versión: " + navigator.appVersion + "</br>");
  if (navigator.javaEnabled()) {
    document.write("El navegador SÍ está preparado para soportar los Applets de Java");
  }else{
    document.write("El navegador NO está preparado para soportar los Applets de Java");
  }
</script>
```

3. Interacción de los objetos con el navegador

32

5. Screen:

- ❑ Corresponde a la pantalla utilizada por el usuario.
- ❑ Todas sus propiedades son solamente de lectura.

Propiedades

availHeight

availWidth

colorDepth

height

pixelDepth

width

3. Interacción de los objetos con el navegador

33

```
<script type="text/javascript">
  document.write("<br>Altura total: "+ screen.height);
  document.write("<br>Altura disponible: "+ screen.availHeight);
  document.write("<br>Anchura total: "+ screen.width);
  document.write("<br>Anchura disponible: "+ screen.availWidth);
  document.write("<br>Profundidad de color: "+ screen.colorDepth);
  |
</script>
```

Ver ejemplos en w3schools del BOM:

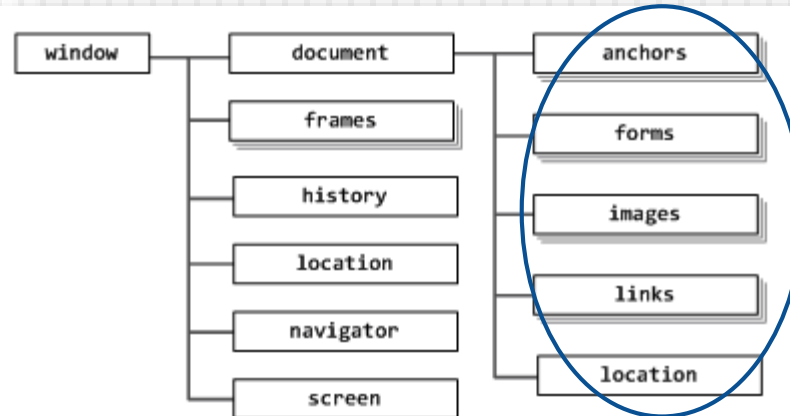
https://www.w3schools.com/js/js_window.asp

3. Interacción de los objetos con el navegador

34

6. Document:

- ❑ Se refiere a los documentos que se cargan en la ventana del navegador.
- ❑ Permite manipular las propiedades y el contenido de los principales elementos de las páginas web.
- ❑ Cuenta con una serie de sub-objetos como los vínculos, puntos de anclaje, imágenes o formularios.



3. Interacción de los objetos con el navegador

35



A.4.7. Hacer ejercicios propuestos en Moodle.

3. Interacción de los objetos con el navegador

36

Ver propiedades y métodos de document:

https://www.w3schools.com/js/js_htmlDOM_document.asp

3. Interacción de los objetos con el navegador

37

```
<html>
<head>
  <meta http-equiv="content-type"
        content="text/html; charset=utf-8">
  <title>DOM</title>
</head>
<body>
  <form>
    <h2>Noticias disponibles</h2>
    <ul>
      <li>Propiedades</li>
      <ul>
        <li>Title</li>
        <li>referrer</li>
        <li>domain</li>
        <li>body</li>
        <li>cookie</li>
      </ul>
      <li>Colecciones</li>
      <ul>
        <li>imagenes</li>
        <li>anchors</li>
        <li>applets</li>
        <li>forms</li>
        <li>links</li>
      </ul>
    </ul>
    <hr>
    <a href="#home"><Home</a><br/>
    <a href="#siguiente"><Siguiente</a><br/>
  </form>
</body>
</html>
```



4. Acceso DOM

38

- En este apartado veremos como se consulta y se interactúa entre nuestro código JavaScript y el DOM.
- Existen diferentes posibilidades a la hora de acceder a un elemento de forma individual o a un conjunto de elementos. Estos mecanismos se clasifican en:
 - ❑ Propiedad
 - ❑ Colección
 - ❑ Recorrido de nodos
 - ❑ Métodos de búsqueda

4. Acceso DOM

39

□ Propiedad

- Existen elementos que permiten ser accesibles directamente a través de las propiedades del objeto document.
- Estos elementos se caracterizan por ser únicos dentro del documento.
- Ej.: *var cabecera= document.head;*

□ Colección

- Usado para elementos que no tienen porqué ser únicos, el objeto document nos ofrece propiedades que devuelven una colección ordenada de esos elementos (forms, links, images, etc.)

4. Acceso DOM

40

❑ Recorrido de nodos

- Se puede explorar el árbol de nodos.
- Para ellos se usan propiedades como, `childNodes`, `firstChild`, `lastChild`, `previousSibling`, `nextSibling` o `parentNode`.
- Se suelen usar en combinación con `nodeType` y `nodeName`.
- También con métodos como `hasChildNodes()`, que indica si el nodo contiene a su vez nodos anidados.

4. Acceso DOM

41

❏ Métodos de búsqueda

- Los métodos de búsqueda ofrecidos por el objeto `document` permiten usar como criterio de búsqueda tanto atributos del elemento como selectores CSS.
- Si empleamos los atributos: **`getElementById()`**, **`getElementsByName()`**, **`getElementsByTagName()`** y **`getElementsByClassName()`**.
- En caso de usar como criterio de búsqueda selectores CSS, se ofrecen los métodos: **`querySelector(cadenaSelector)`** y **`querySelectorAll()`**

4. Acceso DOM

42

1

```
<html>
<head>
  <meta http-equiv="content-type"
        content="text/html; charset=utf-8">
  <title>Consulta DOM</title>
  <script type="text/javascript">
    function verAtributos(){
      //alert(document.images[0].src); // Colecciones
      alert(document.getElementById('imgIzquierda').src); // métodos de búsqueda
    }

  </script>
</head>
<body>
  
  <br/>
  
  <br/>
  <hr>
  <button type="button" onclick="verAtributos();">Ver src de la primera imagen</button>
</body>
</html>
```

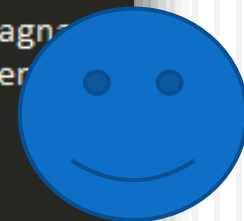


4. Acceso DOM

43

2

```
body>
<div id="elementosTag">
  <h3>Modificación de tamaño texto</h3>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptat
    laudantium aperiam delectus accusamus, impedit quidem velit sequi
    magni aspernatur perferendis ipsam minus, soluta.
  </p>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Magna
    fugit, totam ducimus, quod blanditiis deleniti facilis sapier
    nulla expedita aut ea sequi!
  </p>
</div>
<div>
  <span>Seleccionar color:</span>
  <select id="cboSize">
    <option id="10">Small</option>
    <option id="12">Medium</option>
    <option id="14">Large</option>
  </select>
  <button id="actualizar" onclick="modificarSize();">Cambiar</button>
</div>
</body>
```



4. Acceso DOM

44

```
<script type="text/javascript">
  function modificarSize(){
    var parrafos= document.getElementsByTagName('p')
    for (var i = 0; i < parrafos.length; i++) {
      parrafos[i].style.fontSize=document.getElementById("cboSize").value;
    }
  }
</script>
```

5. Modificar el DOM

45

- Es importante tener en cuenta que el DOM debe estar cargado antes de poder interactuar con él.

- Para ello:

1. Evento *onload*:

```
window.onload=function(){  
    alert("Documento cargado");  
}
```

2. Evento *readystatechange*:

```
document.addEventListener("readystatechange",cargaDOM,false);  
function cargaDOM(evento){  
    if (document.readyState=='complete') {  
        alert("Documento cargado");  
    };  
}
```

3. Interacción de los objetos con el navegador

46



A.4.8. Qué hace el siguiente código?

```
document.addEventListener("readystatechange", cargaDOM, false);  
function cargaDOM(evento){  
    if (document.readyState=='complete') {  
        alert("Documento cargado");  
    }else{  
        alert("cargando..." + document.readyState);  
    }  
}
```

5. Modificar el DOM

47

- Hasta ahora hemos cambiado atributos de nodos, pero podemos ir más allá y modificar el contenido de los nodos o incluso añadir nodos o quitarlos.
 1. Mediante código HTML
 2. Usar objetos de tipo elemento

5. Modificar el DOM

48

1. Mediante código HTML

❑ Contenido estático:

```
<script type="text/javascript">
  var SO = navigator.platform;
  document.write("<h1>Documento abierto con: " + SO + "</h1>");
</script>
```

```
<script type="text/javascript">
  var texto = prompt("Ingresa un título para la nueva ventana: ");
  var ventanaNueva = window.open();
  ventanaNueva.document.write("<h1>" + texto + "</h1>");
</script>
```


5. Modificar el DOM

49

- No sólo creación de texto, también es posible crear y manipular todo tipo de objetos:

```
<script type="text/javascript">
  document.write("<form name=\"cambiacolor\">");
  document.write("<b>Selecciona un color para el fondo de página:</b><br>");
  document.write("<select name=\"color\">");
  document.write("<option value=\"red\">Rojo</option>");
  document.write("<option value=\"blue\">Azul</option>");
  document.write("<option value=\"yellow\">Amarillo</option>");
  document.write("<option value=\"green\">Verde</option>");
  document.write("</select>");
  document.write("<input type=\"button\" value=\"Modifica el color\" onclick=\"document.bgColor=document.
    cambiacolor.color.value\">");
  document.write("</form>");
</script>
```

5. Modificar el DOM

50



A.4.9. Del ejemplo anterior, modifícalo para agregar dos colores más al conjunto de posibles colores del fondo de la página.

5. Modificar el DOM

51

1. Mediante código HTML

□ Contenido dinámico:

■ Propiedades

- innerHTML: código HTML incluido en el elemento actual a excepción de las etiquetas que lo rodean.
- outerHTML: código HTML incluido en el elemento actual incluyendo de las etiquetas que lo rodean.

■ Métodos

- insertAdjacentHTML(posicion,codigo)
 - posicion: beforeBegin, afterBegin, beforeEnd, afterEnd

5. Modificar el DOM

52

```
<html>
<head>
  <meta http-equiv="content-type"
        content="text/html; charset=utf-8">
  <title>Modificar DOM</title>
  <script type="text/javascript">
    var total=0;
    function addNuevoItem(){
      total++;
      document.getElementsByTagName('ul')[0].innerHTML+="
```

5. Modificar el DOM

53



A.4.10. Modifica el ejemplo anterior usando los identificadores id en donde sea preciso, y copia únicamente en la nueva lista el primer y último elemento de la lista original.

5. Modificar el DOM

54

2. Usar objetos de tipo elemento

- ❑ `document.createElement(etiqueta)`: crea el nodo pero no lo inserta en el DOM. Por lo que no será visible hasta que se ubique.
- ❑ `appendChild(nodo)`: inserta el *nodo* en el DOM como hijo menor del elemento actual.
- ❑ `insertBefore(nodo1,nodo2)`: inserta el *nodo1* como hermano inmediatamente mayor de *nodo2* dentro del elemento actual.
- ❑ `removeChild(nuevo, viejo)`: elimina el nodo del DOM y lo devuelve como una referencia.
- ❑ `replaceChild(nuevo,viejo)`: sustituye el nodo viejo por el nuevo dentro del elemento actual.
- ❑ `cloneNode(incluirDescendientes)`: devuelve un elemento que es la copia exacta del actual. Si *incluirDescendientes=true*, duplica el elemento y todos sus descendientes; si es *false*, no incluye los descendientes.

5. Modificar el DOM

55

```
<script type="text/javascript">
  function insertar(){
    var codigoFila="<td>"+document.getElementById('txtCaja').value+"</td>";
    codigoFila+="<td><input type='button' onclick='copiarEncima(this);' value='Copiar encima'/></td>"
    ;
    codigoFila+="<td><input type='button' onclick='borrar(this);' value='Borrar'/></td>";
    var fila=document.createElement('tr');
    fila.innerHTML=codigoFila;
    document.getElementById('tabla').appendChild(fila);

  }
  function copiarEncima(boton){
    var fila= boton.parentNode.parentNode;
    document.getElementById('tabla').insertBefore(fila.cloneNode(true),fila);
  }
  function borrar(boton){
    var fila= boton.parentNode.parentNode;
    document.getElementById('tabla').removeChild(fila);
  }
</script>
```

```
<body>
  <span>Texto</span>
  <input type="text" id="txtCaja">
  <input type="button" onclick="insertar();" value="Insertar texto nuevo">
  <table id="tabla" style="border:1px dashed green;"></table>
</body>
```

5. Modificar el DOM

56



A.4.11. Partiendo del ejemplo anterior:

- Crea la funcionalidad de reemplazar el texto del primer elemento de la tabla por el texto de la caja.
- Luego mejora dicha funcionalidad para que permita modificar cualquier elemento de la tabla identificado por la fila que ocupa. Utiliza para ello una nueva caja de texto.



A.4.12. Siguiendo con el ejercicio anterior cambia el primer y último elemento de la tabla anterior.