Índice: Temas 3.3 y 3.4

3.3 El modelo de objetos del Documento

3.3.1 Introducción al DOM

3.3.2 Acceso a los nodos

3.3.3 Manipulación de nodos

3.3.4 Acceso a los atributos

3.4 El modelo de objetos del Navegador

3.4.1 El objeto window

3.4.2 Otros objetos de utilidad

Índice: Temas 3.3 y 3.4

3.3 El modelo de objetos del Documento

3.3.1 Introducción al DOM

3.3.2 Acceso a los nodos

3.3.3 Manipulación de nodos

3.3.4 Acceso a los atributos

3.4 El modelo de objetos del Navegador

3.4.1 El objeto window

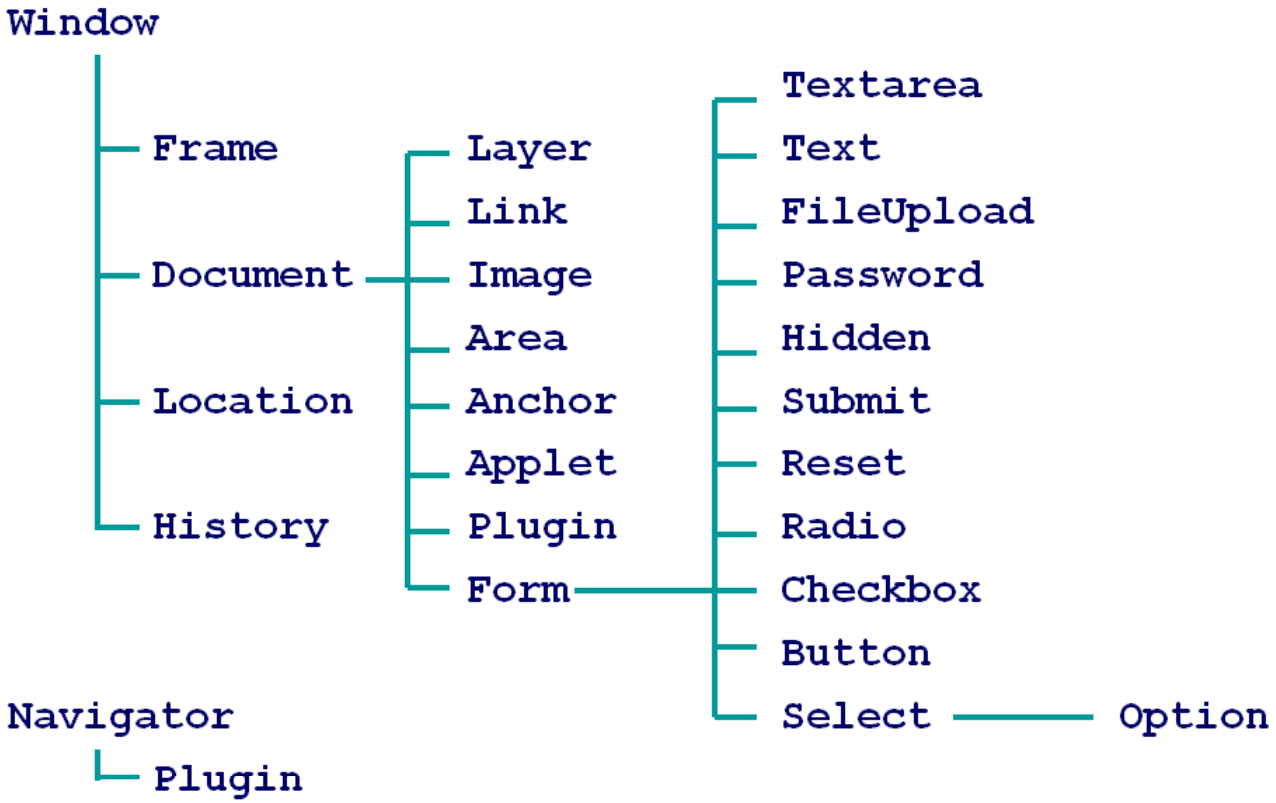
3.4.2 Otros objetos de utilidad

Modelo de Objetos del Documento (1)

- El **Modelo de Objetos del Documento (DOM)** es un interfaz que permite acceder y modificar la estructura y contenido de una página web.
- Especifica cómo se puede acceder desde los lenguajes de script a los distintos elementos (enlaces, imágenes, formularios, etc…) de una página web y cómo se pueden modificar.
- Para lograrlo, se crean una serie de objetos que representan dichos elementos, y que guardan entre ellos unas relaciones de parentesco (jerarquía) que refleja la estructura lógica de una página HTML.
- Una página se presenta en una ventana, que posee un documento, el cual a su vez puede tener una serie de formularios, que pueden contener elementos como botones, cuadros de texto, etc…, a su vez con una serie de propiedades, etc…

Modelo de Objetos del Documento (2)

- A continuación se muestran las clases de objetos del navegador en un diagrama que indica sus relaciones de contenido.



Modelo de Objetos del Documento (3)

- Considérese la siguiente página web sencilla:

```
<!DOCTYPE html>

<html>

    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Página sencilla</title>
    </head>

    <body>
        <p>Esta página es <strong>muy sencilla</strong></p>
    </body>

</html>
```

codig pagina anterior esquema :

el navegador lee y crea este esquema

Modelo de Objetos del Documento (4)

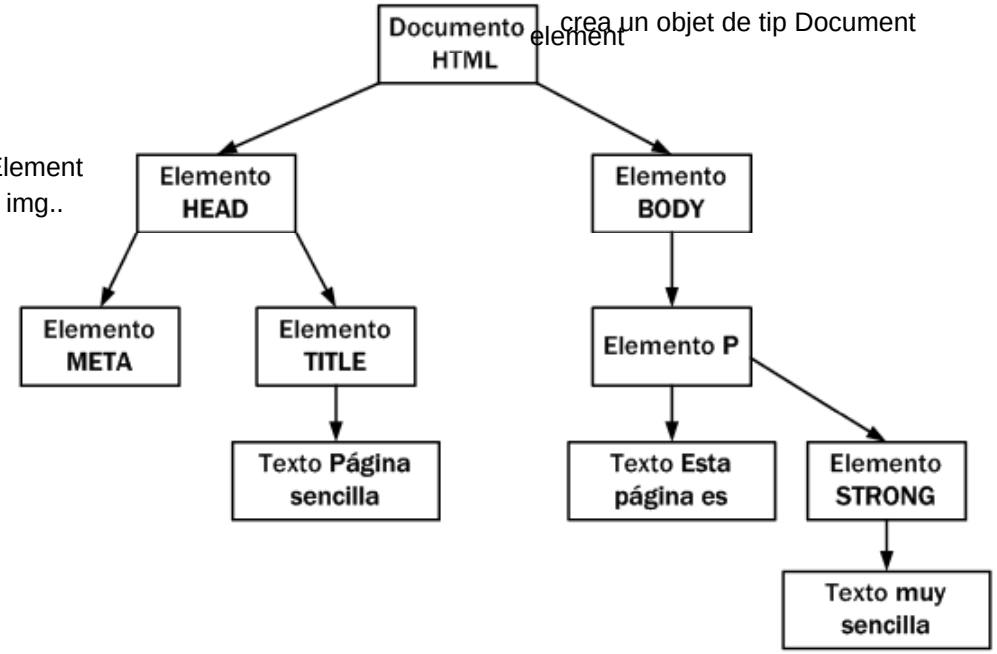
- ✓ Antes de poder utilizar las funciones de DOM, los navegadores convierten automáticamente la página HTML en una estructura de árbol con nodos.
- ✓ Dichas funciones permiten añadir, eliminar, modificar y reemplazar cualquier nodo de cualquier documento de forma sencilla y sin necesidad de recargar la página.

raíz
document

crea un objet de tip Document

element

crea un objet de tip Element
son etiquetas p head img..



desde js podem acceder aqui para modificar

Tipos de nodos

- ✓ La especificación completa de DOM define 12 tipos de nodos. cuand cargam una pagina
- ✓ A continuación se detallan los tipos más importantes de nodos en los que DOM transforma los documentos HTML:

• **Document**	Nodo raíz del que derivan todos los demás nodos del árbol.
• **DocumentType**	Es el nodo que contiene la representación del DTD empleado en la página (indicado mediante el DOCTYPE).
• **Element**	Representa cada una de las etiquetas HTML. Es el único nodo que puede tener tanto nodos hijos como atributos.
• **Attr**	Representa cada uno de los atributos de las etiquetas HTML, es decir, uno por cada par atributo=valor.
• **Text**	Nodo que contiene el texto encerrado por una etiqueta HTML.
• **Comment**	Representa los comentarios incluidos en la página HTML.

La interfaz NODE (1)

acceder al nodo y modificar que querem eliminar un parrafo...

- ✓ 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.
- ✓ En primer lugar, el objeto **Node** define las siguientes constantes para la identificación de cada uno de los 12 tipos de nodos existentes:

```
Node.ELEMENT_NODE = 1
Node.ATTRIBUTE_NODE = 2
...
Node.COMMENT_NODE = 8
Node.DOCUMENT_NODE = 9
...
Node.NOTATION_NODE = 12
```

La interfaz NODE (2)

- Además de estas constantes, **`Node`** proporciona las siguientes propiedades y métodos:

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

borra el parrafo body.removeChild(p)

La interfaz NODE (3)

`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`

Índice: Temas 3.3 y 3.4

3.3 El modelo de objetos del Documento

3.3.1 Introducción al DOM

3.3.2 Acceso a los nodos

3.3.3 Manipulación de nodos

3.3.4 Acceso a los atributos cambiar color...

3.4 El modelo de objetos del Navegador

3.4.1 El objeto window

3.4.2 Otros objetos de utilidad

Tipos de acceso

cm acceder a los elementos de esquema de las primera pagina de este dcument

document.body.firstChild.lastChild. text para acceder al strong del esquema, en cascada ay que meterse en cada una

- ✓ DOM proporciona dos métodos alternativos para acceder a un nodo específico:
  - **Acceso relativo**: También conocido como acceso a través de nodos padre, consiste en acceder al nodo raíz de la página y después a sus nodos hijos y a los nodos hijos de esos hijos y así sucesivamente hasta el nodo buscado.
  - **Acceso directo**: Cuando se quiere acceder a un nodo específico, es mucho más rápido acceder directamente a ese nodo a tener que llegar hasta él descendiendo a través de todos sus nodos padre.

este debem queda

<p id="1 uno" > dand nombre a atributo
es una funcion geElement by ide alg asi

- ✓ Por último, es importante recordar que el acceso a los nodos, su modificación y su eliminación solamente es posible cuando el árbol DOM ha sido construido completamente, es decir, después de que la página HTML se cargue por completo (evento `onload`).
- ✓ Más adelante se verá cómo asegurar que un código JavaScript solamente se ejecute cuando el navegador ha cargado entera la página HTML.

➢ Acceso relativo (1)

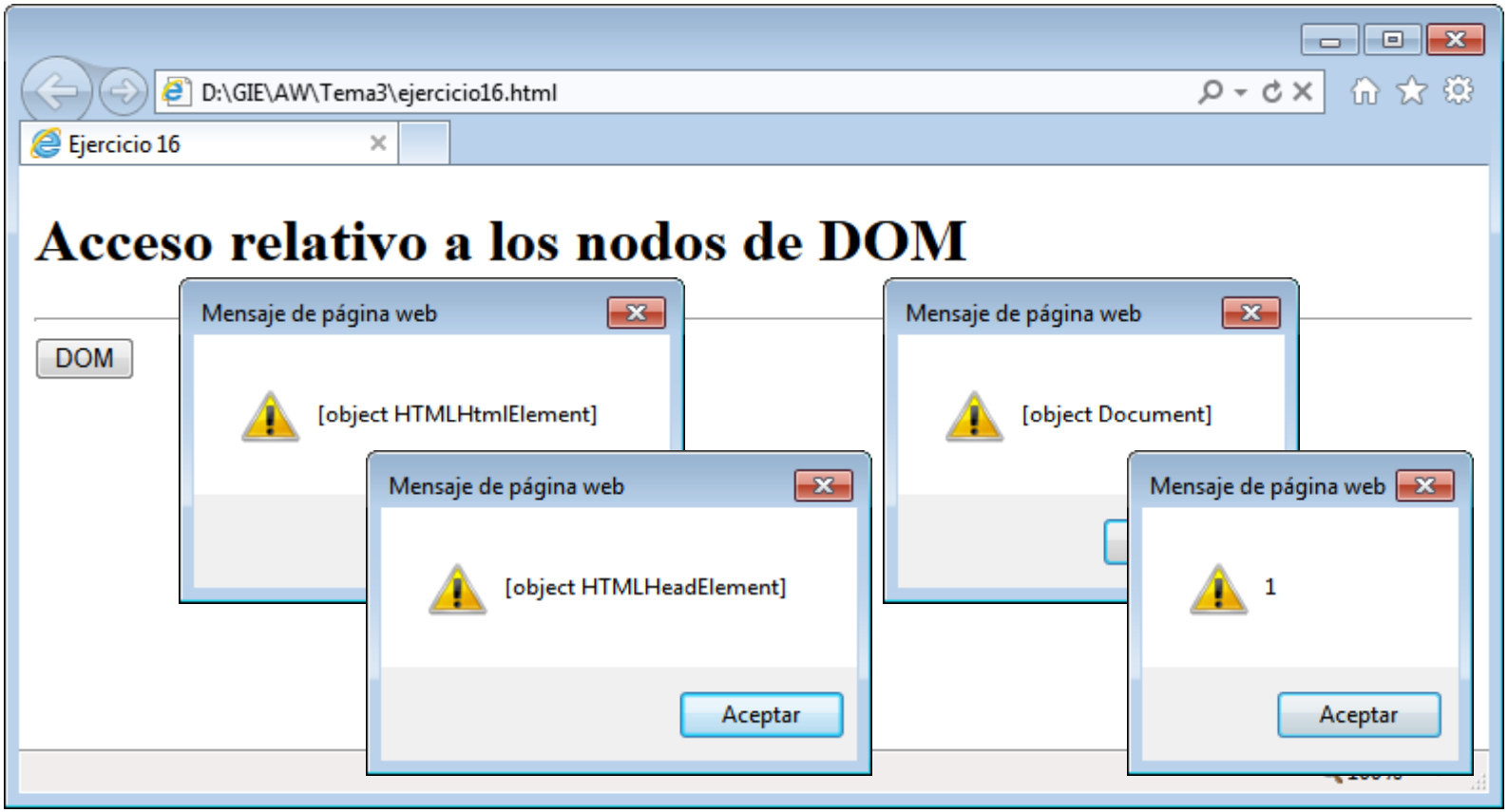
no lo vam a utilizar

✓ La operación básica consiste en obtener el objeto que representa el elemento raíz de la página, para luego utilizar los diferentes métodos de la interfaz **node** para acceder al nodo deseado. Por ejemplo, se pueden hacer cosas del tipo:

```
var objeto_html = document.documentElement;    // html (elemento raiz)

var objeto_head = objeto_html.firstChild;      // head
var objeto_body = objeto_html.lastChild;       // body

var objeto_head = objeto_html.childNodes[0];   // alternativa para obtener head
var objeto_body = objeto_html.childNodes[1];   // alternativa para obtener body

var numeroDescendientes = objeto_html.childNodes.length;      // longitud lista

objeto_head.parentNode;                        // devuelve el objeto_html
objeto_body.parentNode;                        // devuelve el objeto_html
objeto_body.previousSibling;                   // devuelve el objeto_head
objeto_head.nextSibling;                       // devuelve el objeto_body
objeto_head.ownerDocument;                     // devuelve el document

document.nodeType;                             // 9 (tipo del nodo document)
document.documentElement.nodeType;             // 1 (tipo del nodo html)
```

➢ Acceso relativo (2)

✓ Comprobar el acceso a los diferentes nodos de una página mostrando el resultados mediante alertas por pantalla:



Acceso directo (1) este es importante

- ✓ Cuando se trabaja con una página web real, el árbol DOM tiene miles de nodos de todos los tipos. Por este motivo, no es eficiente acceder a un nodo descendiendo a través de todos los ascendentes de ese nodo.
- ✓ Para solucionar este problema, DOM proporciona una serie de métodos para acceder de forma directa a los nodos deseados.
- ✓ **getElementsByTagName("etiqueta")**
  - Dicha función obtiene todos los elementos de la página cuya **etiqueta** coincida con la especificada como argumento de la función:

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

  - El resultado se devuelve en un objeto de tipo **NodeList**, similar a un array de nodos:

```
var primerParrafo = parrafos[0];      var numeroParrafos = parrafos.length;
```

  - Se puede aplicar de forma recursiva sobre cada uno de los nodos devueltos por la función:

```
var enlaces = primerParrafo.getElementsByTagName("a");
```

var imagenes =document.getElementsByTagName("img") arai con tda imagen de una pagina

var imagen1= imagenes[0] ; para ver la primera imagen

< img name = "image2" >

Acceso directo (2)

✓ getElementsByName("name")

- Dicha función obtiene todos los elementos de la página cuyo atributo **name** coincida con el especificado como argumento de la función.

```
<p name="especial">Contenido del párrafo</p>
```

<img name = " imagen2 ">

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

imagen2 le pasam est dnd pne especial

✓ getElementById("id")

- Dicha función obtiene todos el elemento de la página cuya atributo **id** se especifica como argumento de la función:

```
<div id="cabecera">
```

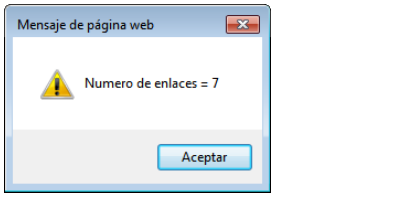
```
var cabecera = document.getElementById("cabecera");
```

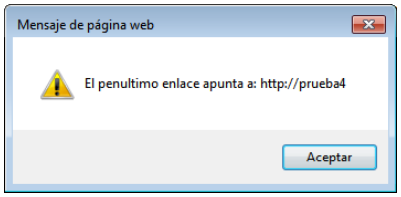
✓ *getAttribute("atributo"): El acceso a los atributos se verá más adelante.*

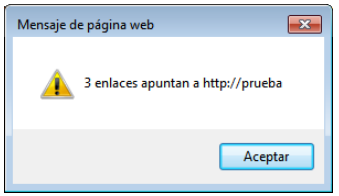
Acceso directo (3)

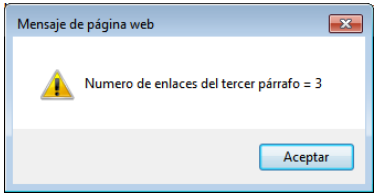
- ✓ Dada la página web *ejercicio17.html*, obtener: 1) el número de enlaces de la página; 2) la dirección a la que enlaza el penúltimo enlace; 3) el número de enlaces que enlazan a http://prueba; 4) el número de enlaces del tercer párrafo.

en js









Índice: Temas 3.3 y 3.4

3.3 El modelo de objetos del Documento

3.3.1 Introducción al DOM

3.3.2 Acceso a los nodos

3.3.3 Manipulación de nodos

3.3.4 Acceso a los atributos

3.4 El modelo de objetos del Navegador

3.4.1 El objeto window

3.4.2 Otros objetos de utilidad

Creación de nodos (1)

en js

- La interfaz **node** proporciona métodos para la crear (**createNode()**) y añadir (**appendChild()**) nuevos nodos al árbol de la página.
- Veámoslo con un ejemplo. Supóngase que se desea añadir de forma dinámica un nuevo párrafo (Párrafo 1) a la siguiente página:

cuand pulsa el usuari

```
<html>
  <head><title>Página sencilla</title></head>
  <body></body>
</html>
```

- Para ello, se deben seguir los siguientes pasos:

1. Crear un nodo de tipo elemento (para el párrafo).
2. Crear un nodo de tipo texto (para el contenido).
3. Asociar el nodo texto al nodo párrafo.
4. Añadir el nodo párrafo al nodo body.

Creación de nodos (2)

- El código correspondiente sería:

```
1. var p = document.createElement("p");
2. var texto = document.createTextNode("Párrafo 1");
3. p.appendChild(texto);
4. document.body.appendChild(p);
```

cream el nod p en la estructura de esquema

este texto es un nodo que detiene el parrafo 1 que está entre parente

- Y el resultado:

```
<html>
<head><title>Página sencilla</title></head>
<body>
    <p>Párrafo 1</p>
</body>
</html>
```

Creación de nodos (3)

- ✓ Además, se puede añadir un segundo párrafo:

```
1. var p2 = document.createElement("p");
2. var texto = document.createTextNode("Párrafo 2");
3. p2.appendChild(texto);
```

- ✓ Detrás del primero:

```
4. document.body.appendChild(p2);
```

- ✓ O delante:

```
4. var p = document.getElementsByTagName("p")[0];
5. document.body.insertBefore(p2,p);
```

Reemplazo de nodos (1)

- La interfaz `node` proporciona el método `replaceChild()` para reemplazar cualquier nodo existente originalmente en la página o creado dinámicamente.
- Supongamos que queremos reemplazar el nodo anteriormente añadido (primer párrafo de la página) por otro.
- Para ello, se deben seguir los siguientes pasos:

1. Crear un nodo de tipo elemento (para el párrafo).
2. Crear un nodo de tipo texto (para el contenido).
3. Asociar el nodo texto al nodo párrafo.
4. Acceder al nodo que se pretende reemplazar.
5. Ejecutar desde su nodo padre el reemplazo del nodo antiguo por el nuevo.

Reemplazo de nodos (2)

- El código correspondiente sería:

```
1. var pnew = document.createElement("p");
2. var texto = document.createTextNode("Párrafo 2");
3. pnew.appendChild(texto);
4. var pold = document.getElementsByTagName("p")[0];
5. pold.parentNode.replaceChild(pnew,pold);
```

esta particularizad a esa pagina l pdem quitar

- Y el resultado:

```
<html>
<head><title>Página sencilla</title></head>
<body>
    <p>Párrafo 2</p>
</body>
</html>
```

Eliminación de nodos

- La interfaz **`node`** proporciona el método **`removeChild()`** para eliminar cualquier nodo existente originalmente en la página o creado dinámicamente.
- Supongamos que queremos eliminar el nodo anteriormente añadido (primer párrafo de la página). Para ello, se deben seguir los siguientes pasos:
  1. Acceder al elemento a eliminar.
  2. Eliminarlo desde su elemento padre.
- El código correspondiente sería:

```
1. var p = document.getElementsByTagName("p")[0];
2. document.body.removeChild(p);
```

- O, en el caso de que no se conozca su padre:

```
2. p.parentNode.removeChild(p);
```

Ejercicio de manipulación de nodos

- Programar las funciones necesarias para crear nuevos párrafos con el texto "Párrafo 1", crear párrafos delante del primero con el texto"Párrafo 2", reemplazar el primero (por "Párrafo 3") y eliminar el primero.

Índice: Temas 3.3 y 3.4

3.3 El modelo de objetos del Documento

3.3.1 Introducción al DOM

3.3.2 Acceso a los nodos

3.3.3 Manipulación de nodos

3.3.4 Acceso a los atributos

3.4 El modelo de objetos del Navegador

3.4.1 El objeto window

3.4.2 Otros objetos de utilidad

Acceso a los atributos

- ✓ Además del tipo de etiqueta HTML y su contenido de texto, DOM permite el acceso directo a todos los **atributos** de cada etiqueta.
- ✓ Los atributos 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 HTML detrás del nombre del nodo.
- ✓ Ejemplo:

var parafo= document.getElementbyID("p1")
paraf.size=" 20 px" mdifica el tama

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

var a = document.getElementsByTagName("a");
alert(a.href);                          // muestra http://www.paginaweb.com
a.href = www.otrapaginaweb.com          // modifica el enlace
```

Acceso a las propiedades CSS (1)

- El acceso a las **propiedades CSS** no es tan directo y sencillo como el acceso a los atributos HTML, dado que los estilos CSS se pueden aplicar de varias formas diferentes sobre un mismo elemento HTML.
- El acceso a las propiedades CSS establecidas mediante el atributo **`style`** se realiza a través de la propiedad **`style`** del nodo que representa a ese elemento.
- Ejemplo:

```
<p id="parrafo1" style="color:#C00">Párrafo</a>

var p = document.getElementById("parrafo1");
var color = p.style.color;
```

Aunque el funcionamiento es homogéneo entre distintos navegadores, los resultados no son exactamente iguales: Firefox y Safari muestran el valor rgb(204, 0, 0) , Internet Explorer muestra el valor #c00, etc…

Acceso a las propiedades CSS (2)

- Si el nombre de una propiedad CSS es compuesto, se accede a su valor modificando ligeramente su nombre.
- La transformación del nombre de las propiedades CSS compuestas consiste en eliminar todos los guiones medios (-) y escribir en mayúscula la letra siguiente a cada guión medio.
- Ejemplos:
  - `font-weight` se transforma en `fontWeight`
  - `border-top-style` se transforma en `borderTopStyle`
- Como la palabra `class` está reservada por JavaScript, no es posible utilizarla para acceder al atributo `class` del elemento HTML. En su lugar, DOM utiliza el nombre `className` para acceder al atributo `class` de HTML.

Acceso a las propiedades CSS (3)

- Para obtener el valor de las propiedades CSS independientemente de cómo se hayan aplicado, es necesario utilizar otras propiedades de JavaScript.

```
<p id="parrafo1">Párrafo</p>          // Código HTML
#parrafo1 { color: #008000; }         // Regla CSS
```

- Si se utiliza un navegador de la familia Internet Explorer, se hace uso de la propiedad **currentStyle**:

```
var p = document.getElementById("parrafo1");
var color = p.currentStyle['color'];
```

- Si se utiliza cualquier otro navegador, se puede emplear la función **getComputedStyle()**:

```
var p = document.getElementById("parrafo1");
var color = document.defaultView.getComputedStyle(p,'').
                                    getPropertyValue('color');
```

Acceso a las propiedades CSS (4)

- La propiedad `currentStyle` requiere el nombre de las propiedades CSS según el formato de JavaScript (sin guiones medios), mientras que la función `getPropertyValue()` exige el uso del nombre original de la propiedad CSS.
- Por este motivo, la creación de aplicaciones compatibles con todos los navegadores se puede complicar en exceso.
- Existen funciones compatibles con todos los navegadores, como la función `getStyle()` creada por el programador Robert Nyman y publicada en su blog personal:

  http://www.robertnyman.com/2006/04/24/get-the-rendered-style-of-an-element/

```
var parrafo = document.getElementById("parrafo");
var color = getStyle(parrafo, 'color');
```

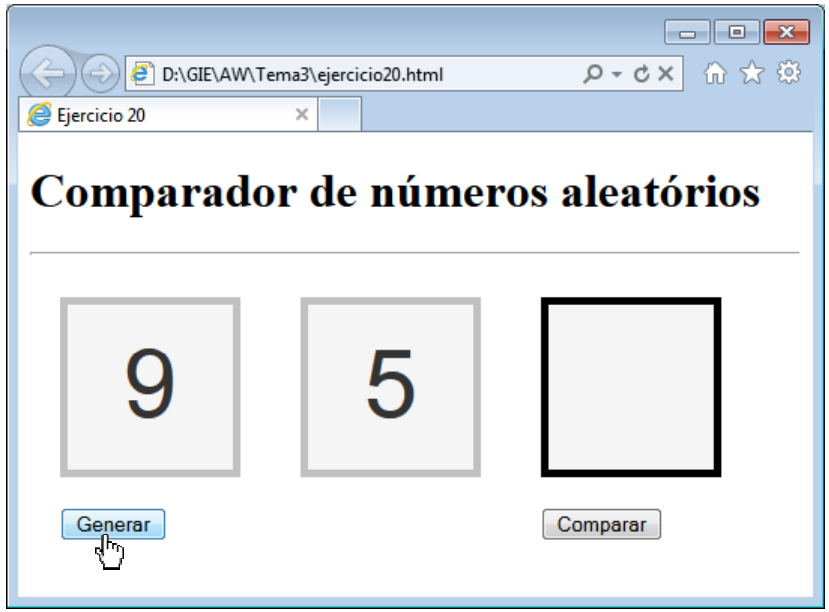
Ejercicio sobre acceso a los atributos

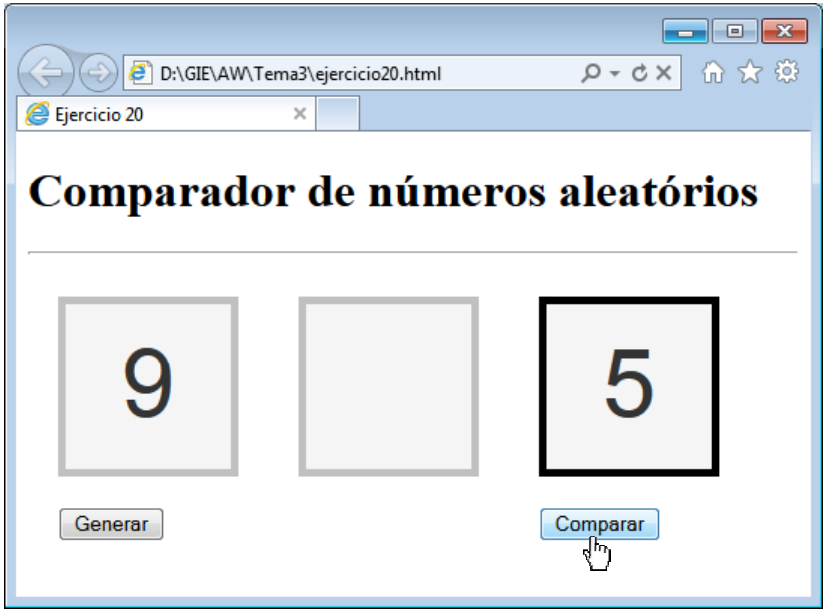
- Dada la página web *ejercicio19.html*, programar las funciones capaces de mostrar y cambiar el color y tamaño de letra de los dos primeros párrafos, y cambiar el estilo de la pseudo-clase `a:hover` del enlace del tercero a verde.

➢ Generador-comparador de números aleatorios

✓ Diseñar una web con la que generar de forma aleatoria dos números entre 0 y 9 a través de un botón “generar”, y que obtenga el menor de ellos a través de un botón “comparar”.

exame





Índice: Temas 3.3 y 3.4

3.3 El modelo de objetos del Documento

3.3.1 Introducción al DOM

3.3.2 Acceso a los nodos

3.3.3 Manipulación de nodos

3.3.4 Acceso a los atributos

3.4 El modelo de objetos del Navegador

3.4.1 El objeto window

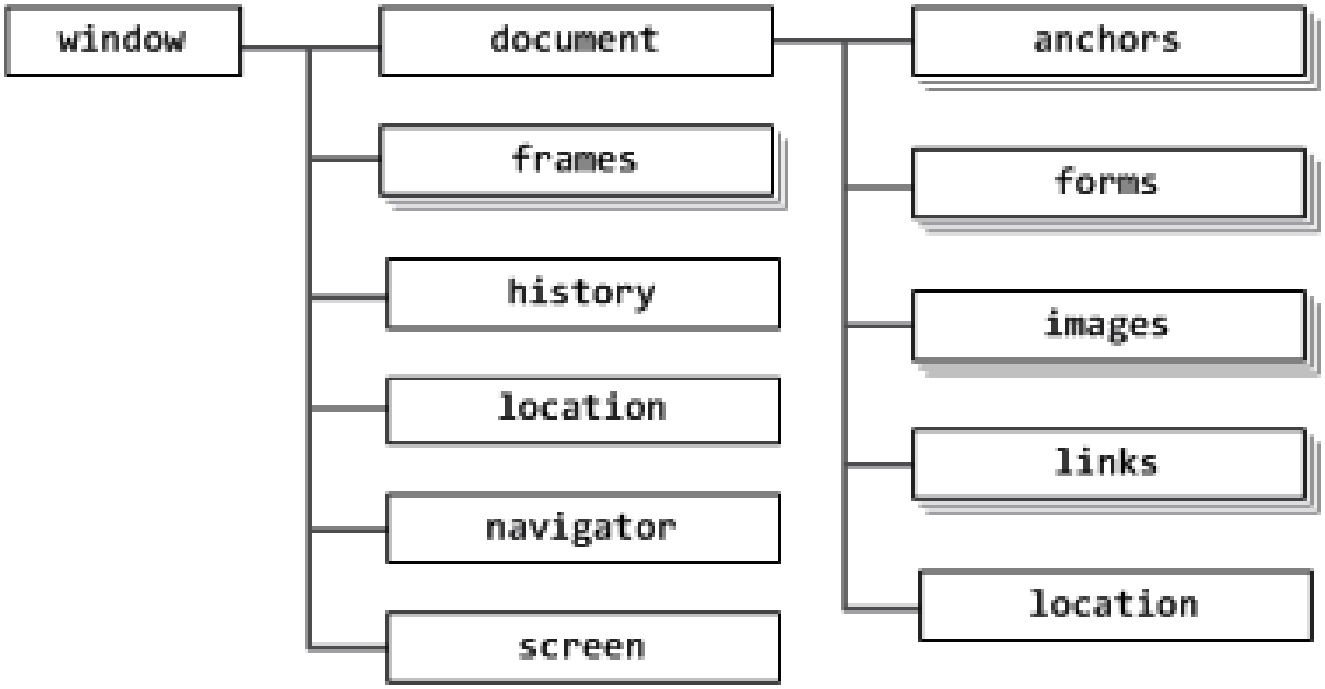
3.4.2 Otros objetos de utilidad

Modelo de Objetos del Navegador (1)

- El **Modelo de Objetos del Navegador (BOM)** es un interfaz a través de la cual es posible:
  - Crear, mover, redimensionar y cerrar ventanas de navegador.
  - Modificar el texto que se muestra en la barra de estado.
  - Obtener información sobre el propio navegador.
  - Acceder a propiedades de la página.
  - Gestión de cookies.
  - Y realizar muchas otras manipulaciones no relacionadas con el contenido de la página HTML, como:
    - Utilización de temporizadores-
    - Etc…

Modelo de Objetos del Navegador (2)

- A continuación se muestran las clases de objetos del navegador en un diagrama que indica sus relaciones de contenido.



El objeto Window

- Se trata del objeto de más alto nivel, y representa la ventana completa del navegador. Tiene métodos para:
  - Mostrar mensajes al usuario
  - Mover, redimensionar y manipular la ventana actual del navegador.
  - Utilizar temporizadores.
  - Abrir y cerrar nuevas ventanas de navegador.
- Algunas propiedades del objeto **`window`**:

`status`	Barra de estado	**`window.status`**
`name`	Nombre de la ventana	**`window.name`**

- Dado que se trata del elemento de mayor nivel, todas sus propiedades y métodos pueden ser accedidos sin necesidad de colocar su nombre delante del punto.

Mensajes de JavaScript (1)

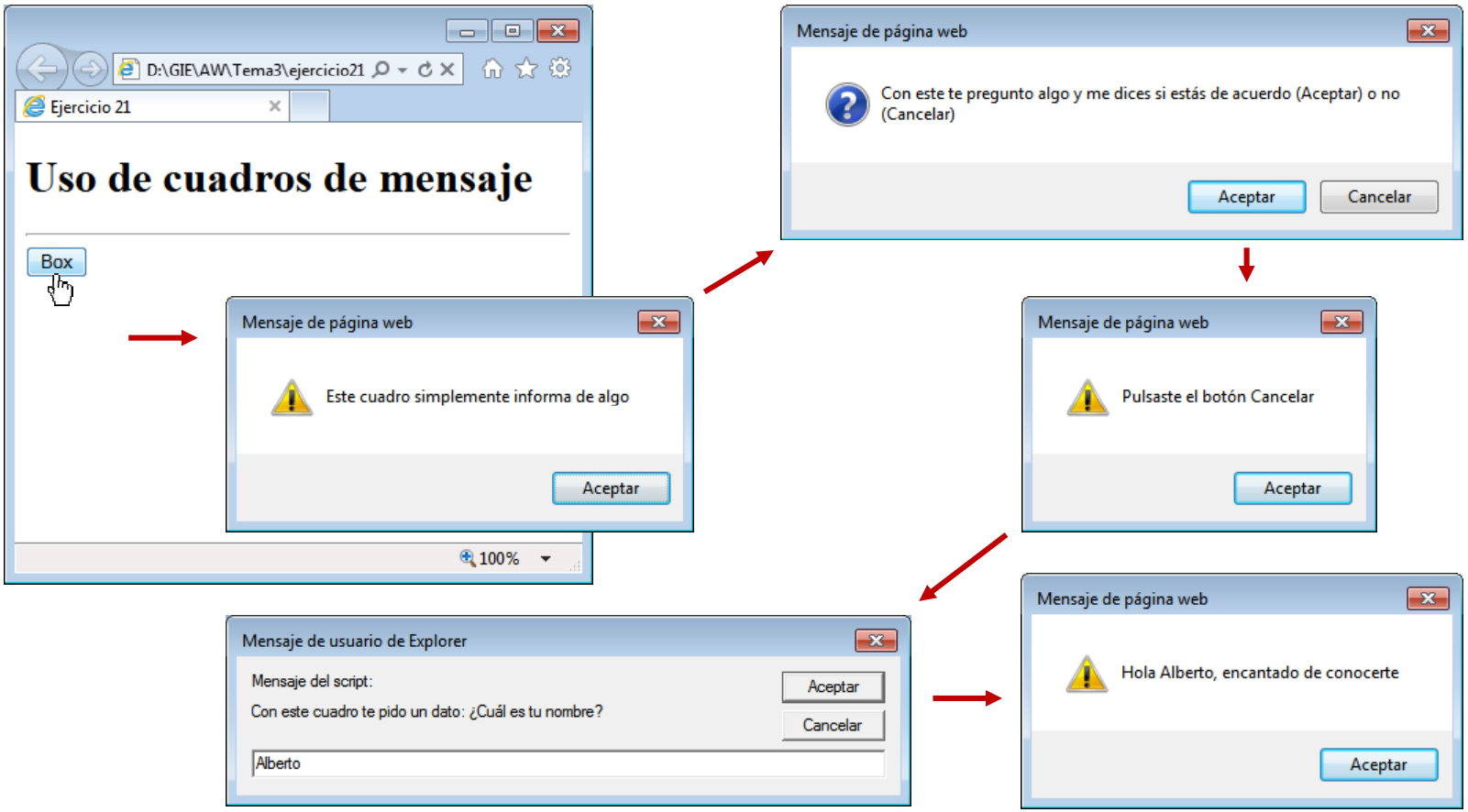
- JavaScript tiene tres tipos de cuadros de mensaje con los que interaccionar con el usuario mostrando o pidiendo algún tipo de información mediante diferentes métodos del objeto **`window`**:

`alert(mensaje)`	Muestra un cuadro de mensaje. `window.alert("Hola")`
`confirm(mensaje)`	Muestra un cuadro de confirmación. Devuelve `true` si el usuario pulsa el botón Aceptar y `false` si pulsa Cancelar. `window.confirm("¿Terminar?")`

`prompt(mensaje,inicial)`	Para leer un dato. Muestra un cuadro con el mensaje, un cuadro de texto (con valor inicial) y botones Aceptar y Cancelar. Devuelve el valor introducido.
	`window.prompt("Introduce tu edad","20")`
	`window.prompt("Introduce tu edad","")`

Mensajes de JavaScript (2)

- ✓ Utilizar los métodos anteriores para solicitar y mostrar información mediante cuadros de texto:



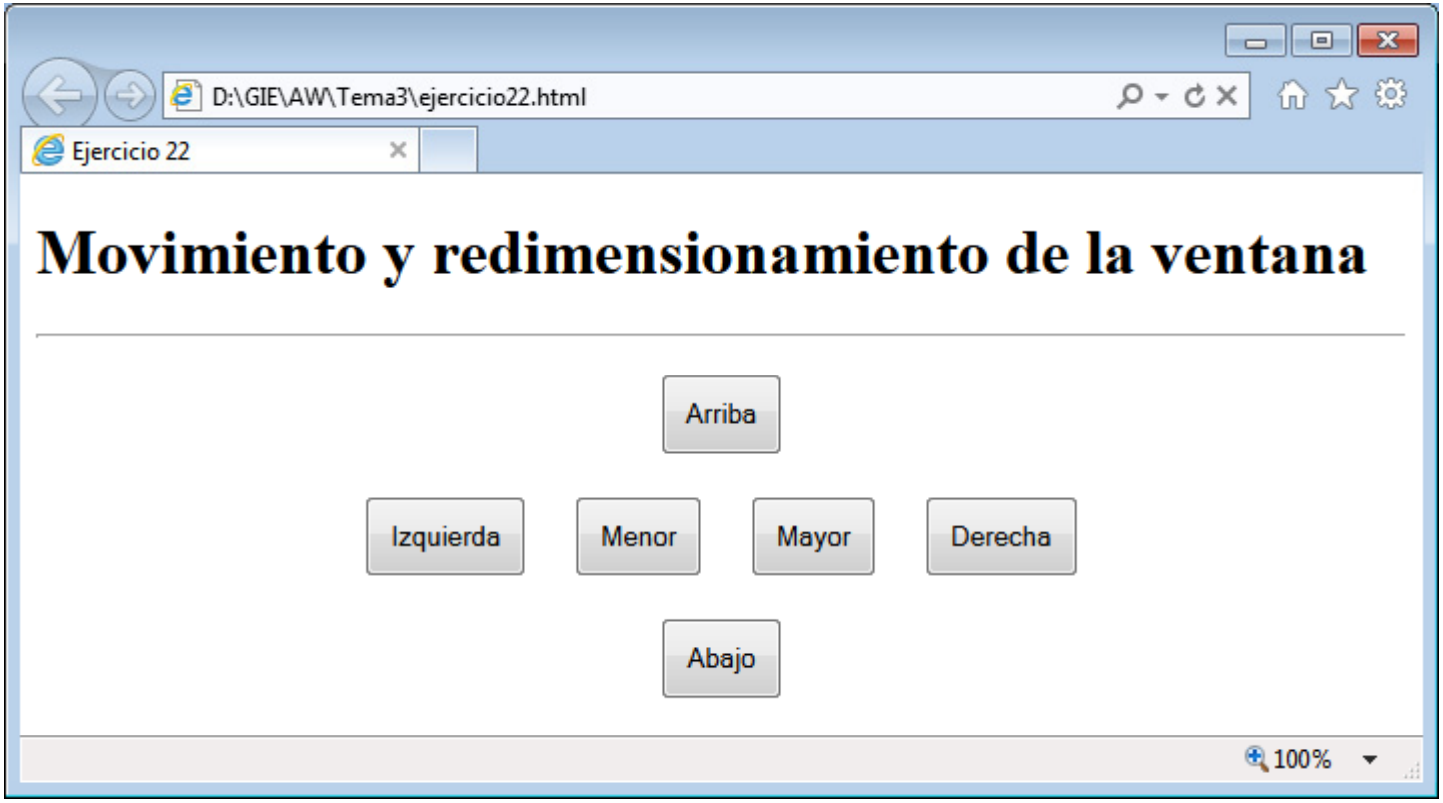
Manejo de la ventana en JavaScript (1)

- ✓ BOM define cuatro métodos para manipular el tamaño y la posición de la ventana:

`moveBy(x,y)`	Desplaza la posición de la ventana x píxeles hacia la derecha e y píxeles hacia abajo. Se permiten desplazamientos negativos (izquierda y arriba). `window.moveBy(10,10)`
`moveTo(x,y)`	Desplaza la esquina superior izquierda a la posición (x, y). Se permiten desplazamientos negativos. `window.moveTo(200,100)`
`resizeBy(x,y)`	Redimensiona la ventana aumentando su anchura y altura en x e y respectivamente. Valores negativos reducen. `window.resizeBy(50,-50)`
`resizeTo(ancho,alto)`	Establece el tamaño de la ventana. `window.resizeTo(300,200)`

Manejo de la ventana en JavaScript (2)

- ✓ Desarrollar una página web que contenga seis botones, dos para hacer la ventana un poco mayor o menor y cuatro más para desplazar la ventana un poco hacia cada dirección.



Temporización en JavaScript (1)

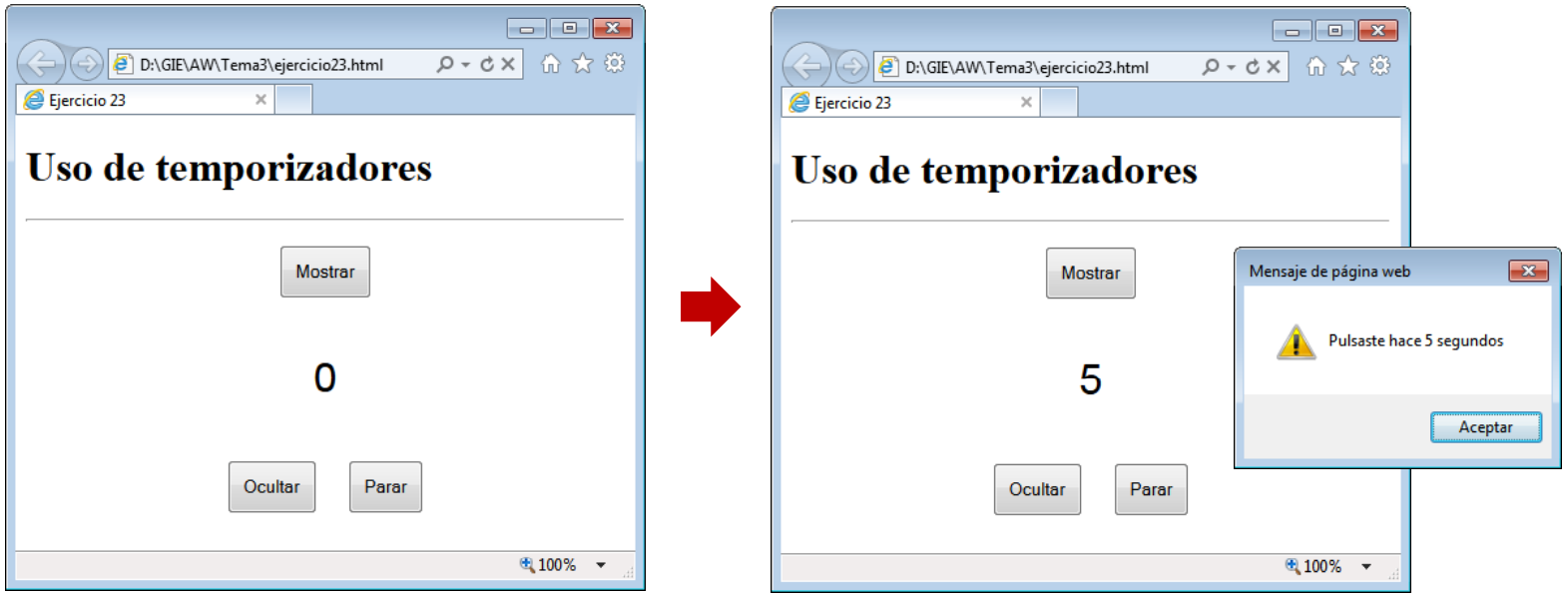
- JavaScript proporciona diferentes métodos para controlar la ejecución de código en intervalos específicos de tiempo:

`setTimeout(código,x)`	Evalúa el código después de x milisegundos. Devuelve un identificador para su cancelación. `id=window.setTimeout(funcion,1000)`
`clearTimeout(id)`	Cancela la tarea programada establecida con setTimeout() identificada con id. `window.clearTimeout(id)`
`setInterval(código,x)`	Evalúa el código cada x milisegundos. Devuelve un identificador para su cancelación. `id=window.setInterval(funcion,1000)`
`clearInterval(id)`	Cancela la tarea periódica establecida con setInterval() identificada con id. `window.clearInterval(id)`

➢ Temporización en JavaScript (2)

✓ Desarrollar una página que contenga tres botones y un segundero:

- El botón Mostrar lanza un mensaje que no aparecerá hasta transcurridos 5 segundos desde la pulsación. La aparición de dicho mensaje puede interrumpirse si se pulsa el botón Ocultar antes de que transcurran los 5 segundos. Hay un segundero que muestra el tiempo transcurrido, y éste se puede parar en cualquier momento mediante el botón parar.



➢ Temporización en JavaScript (3)

✓ Desarrollar una página que contenga dos botones, uno para hacer que la ventana se balancee lateralmente, y otro para parar dicho balanceo:

- El balanceo consiste en mover la ventana 100 píxeles hacia la derecha de uno en uno cada 10 milisegundos, repetir el mismo movimiento hacia la izquierda, etc…

Ventanas emergentes (1)

- JavaScript proporciona el método **open()** del objeto **window** para abrir documentos en nuevas ventanas con unas determinadas características:

```
open(url,windowName,windowFeatures)
```

url	URL de la página a abrir en la ventana.
windowName	Nombre de la ventana.
windowFeatures	Características de la ventana.

- La cadena de características contiene, separadas por comas, asignaciones de valores a las características (**característica=valor**) para las que el valor por defecto no resulte adecuado.

- La ventana se cierra con el método **close()**:

```
ventana=window.open("http://www.ucm.es","VentanaUCM","status=1");
ventana.close();
```

Ventanas emergentes (2)

- Características más usuales:

`width`	Anchura en píxeles.
`height`	Altura en píxeles.
`left`	Desplazamiento desde el marco izquierdo (coordenada x).
`top`	Desplazamiento desde el marco superior (coordenada y).
`location`	Barra de direcciones (valores posibles: `yes/1` o `no/0`).
`menubar`	Barra de menús (valores posibles: `yes/1` o `no/0`).
`directories`	Barra de vínculos (valores posibles: `yes/1` o `no/0`).
`toolbar`	Barra de herramientas (valores posibles: `yes/1` o `no/0`).
`scrollbars`	Barras de desplazamiento (`yes/1`, `no/0` o `auto`).
`status`	Barra de estado (valores posibles: `yes/1` o `no/0`).
`resizable`	Redimensionable (`yes/1` o `no/0`).

➢ Ventanas emergentes (3)

✓ Desarrollar una página web que cree una nueva ventana en donde se muestre la página principal de la Complutense y sólo contenga barras de desplazamiento.

Índice: Temas 3.3 y 3.4

3.3 El modelo de objetos del Documento

3.3.1 Introducción al DOM

3.3.2 Acceso a los nodos

3.3.3 Manipulación de nodos

3.3.4 Acceso a los atributos

3.4 El modelo de objetos del Navegador

3.4.1 El objeto window

3.4.2 Otros objetos de utilidad

Los objetos history y location (1)

- El objeto **history** contiene la lista de las URL de las páginas visitadas. Su propiedad **length** contiene el número de entradas. Algunos de sus métodos son:
  - **back()** Carga la página anterior.
  - **forward()** Carga la página siguiente.
  - **go(pos)** Carga la página en la posición indicada dentro de la lista de historial.
- El objeto **location** proporciona información de la URL de la página actual. Se puede modificar mediante el método **assign()**. Algunas propiedades son:
  - **href** URL completo.
  - **hostname** Nombre del dominio.
  - **pathname** Ruta al recurso.
  - **port** Puerto de acceso (normalmente 80 pero no se especifica).
  - **protocol** Protocolo (normalmente http:).

Los objetos history y location (2)

- ✓ Probar las propiedades y métodos de los objetos history y location subiendo el ejercicio 26 al servidor.
  - Navegar a alguna web desde esta para almacenar algo en el historial y así probar los métodos y propiedades del objeto history.
  - El botón location lanza una función que muestra todas las propiedades del objeto location mediante alertas. El botón AW-GIE carga la web de la asignatura modificando la URL actual mediante el método assign().

Los objetos navigator y screen (1)

- El objeto **navigator** contiene información sobre el navegador:
  - **appName** Nombre del navegador.
  - **appVersion** Versión del navegador.
  - **cookieEnabled** Si están habilitadas las cookies.
  - **platform** Plataforma (win, linux, mac, etc…)
  - **plugins** Array con los plugins soportados.
  - **systemLanguage** Lenguaje del sistema.

- El objeto **screen** contiene información sobre el monitor:
  - **availWidth** Anchura disponible en el monitor.
  - **availHeight** Altura disponible en el monitor.

Los objetos navigator y screen (2)

- ✓ Probar las propiedades y métodos de los objetos **`navigator`** y **`screen`** mostrando la siguiente información en un párrafo vacío.
  - Para mostrar la información correspondiente a cada botón, se puede generar una cadena de texto con la misma e incluirla como contenido del párrafo, accesible mediante la propiedad **`innerHTML`** del mismo.