DESARROLLO WEB EN ENTORNO CLIENTE

CAPÍTULO 3:

Utilización de los objetos predefinidos de JavaScript

Objetivos

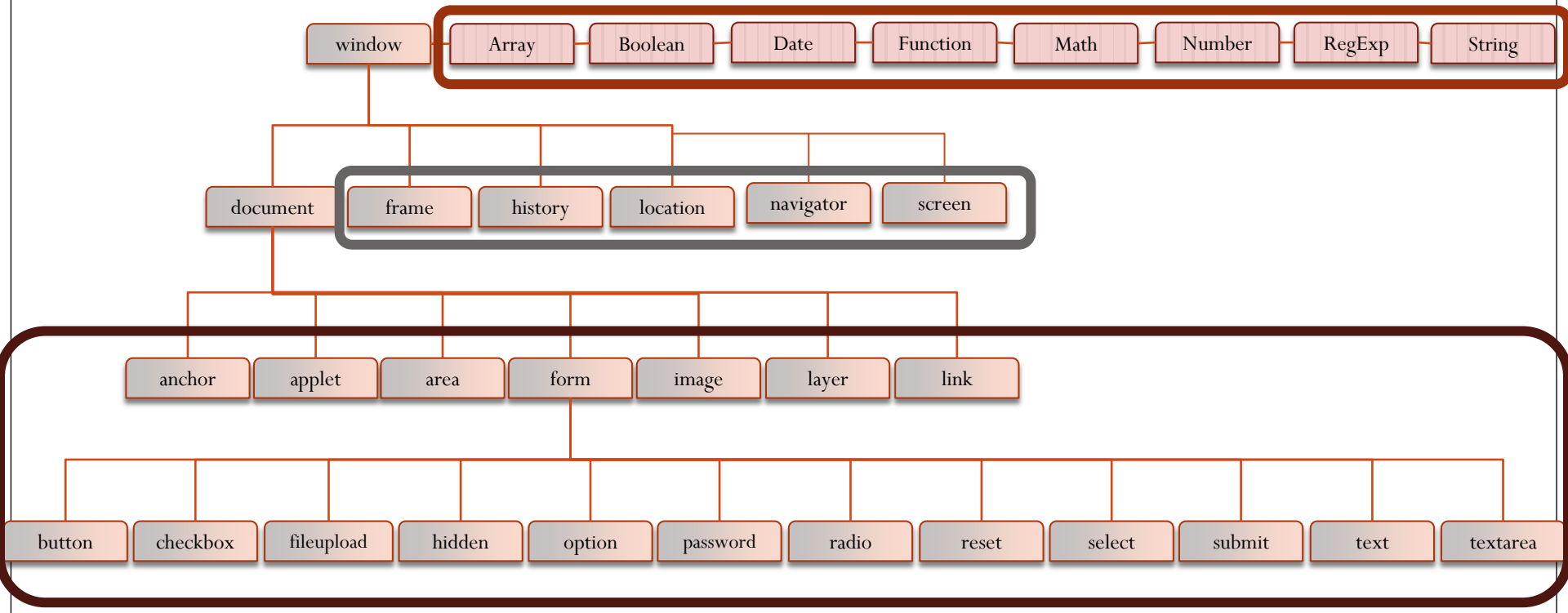
- Conocer cuáles son los principales objetos predefinidos
- Comprender las propiedades y métodos de los principales objetos de JavaScript
- Aprender a generar código HTML desde sentencias JavaScript
- Dominar el uso de los marcos de HTML y aprender a realizar interacciones entre ellos con JavaScript
- Manipular y gestionar la creación y apariencia de las ventanas del navegador, además de la comunicación entre ellas

ÍNDICE

1. OBJETOS NATIVOS DE JAVASCRIPT
2. INTERACCIÓN DE LOS OBJETOS CON EL NAVEGADOR
3. GENERACIÓN DE ELEMENTOS HTML DESDE CÓDIGO JAVASCRIPT
4. APLICACIONES PRÁCTICAS DE LOS MARCOS
5. GESTIÓN DE LAS VENTANAS

Introducción

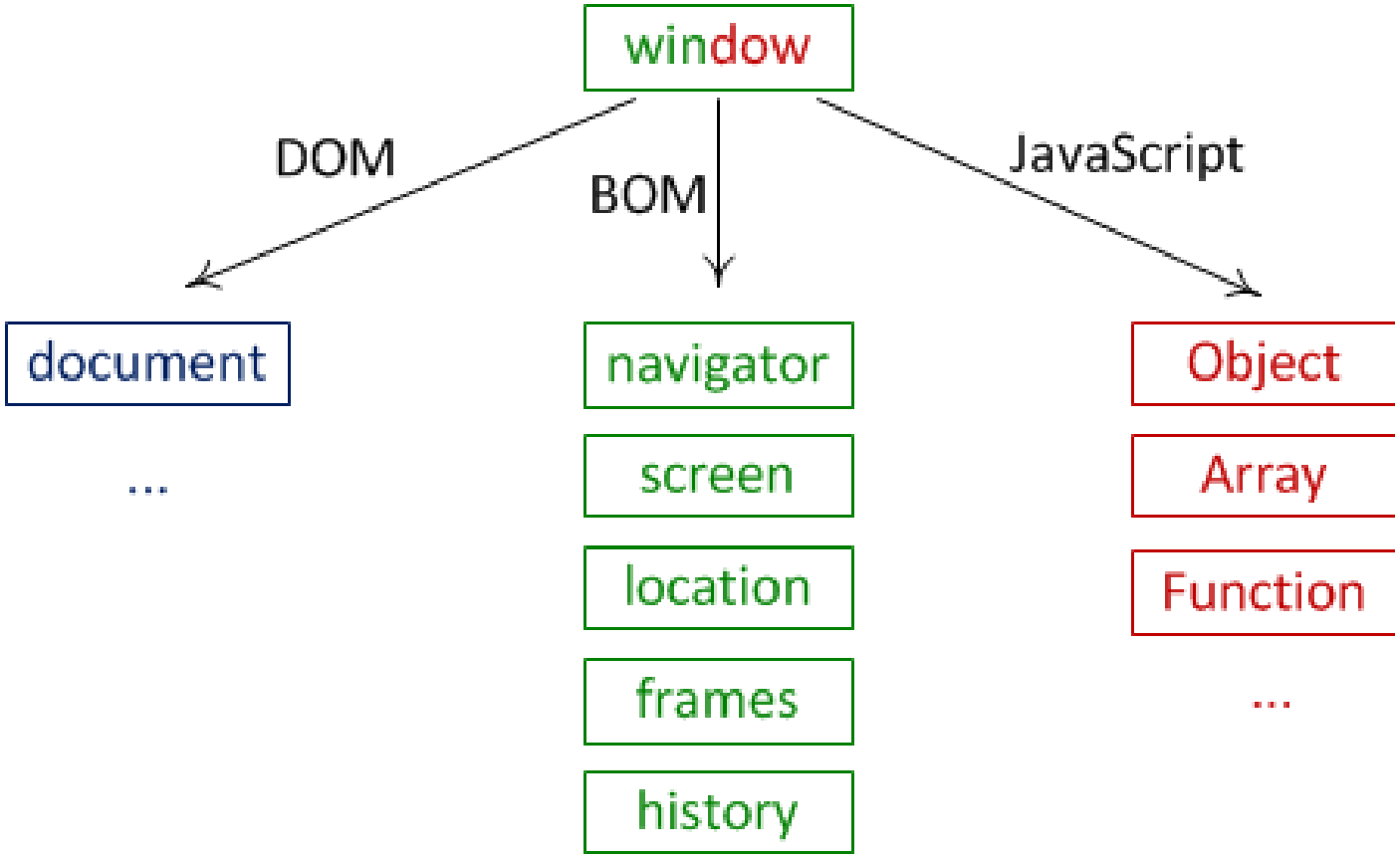
- Los objetos de JavaScript se ordenan de modo jerárquico.



Introducción

- **Objetos del lenguaje:** Object, Boolean, Number, Math, Date, String, Array, RegExp
- **Objetos del navegador:** Window, Navigator, Screen, Location, History
- **Objetos DOM**
  - *Core DOM*: Node, NodeList, NameNodeMap, Document, Element, Attr
  - *HTML DOM*
    - Document, Events, Elements
    - Ancho, Area, Base, Body, Button, Form, Frameset, Image, Input, Link, Meta, Object, Option, Select, Style, Table, Textarea
- **Objetos definidos por el usuario**

Introducción



ÍNDICE

1. **OBJETOS NATIVOS DE JAVASCRIPT**
2. INTERACCIÓN DE LOS OBJETOS CON EL NAVEGADOR
3. GENERACIÓN DE ELEMENTOS HTML DESDE CÓDIGO JAVASCRIPT
4. APLICACIONES PRÁCTICAS DE LOS MARCOS
5. GESTIÓN DE LAS VENTANAS

Objetos nativos de JavaScript

- JavaScript proporciona una serie de objetos definidos nativamente que no dependen del navegador.
- Un objeto se compone de:
  - ***Propiedades*** → características
    - *Constructor*
    - *Prototype* : permite añadir propiedades y métodos al objeto
      Ej: Boolean.prototype.nuevaFuncion= function(){//código}
  - ***Métodos*** → funciones específicas que se pueden realizar con los objetos
    - *toString*(): Devuelve una representación como string del objeto.
    - *valueOf*(): Devuelve el valor primitivo
  - ***Eventos*** → situaciones que pueden llegar a realizarse o no. Cada objeto reconoce una serie de eventos

Objetos nativos de JavaScript

- Para crear un objeto se utiliza la palabra clave `new`. Ejemplo:
  - `var miObjeto = new Object();`

- En JavaScript se accede a las propiedades y a los métodos de los objetos mediante el operador punto (“.”):
  - `miObjeto.nombrePropiedad;`
  - `miObjeto.nombreFunción([parámetros]);`

Objetos nativos de JavaScript

- El objeto **Date**:
  - Permite realizar controles relacionados con el tiempo en las aplicaciones web.
  - Cuenta con una serie de métodos divididos en tres subconjuntos:
    - *Métodos de lectura*.
    - *Métodos de escritura*.
    - *Métodos de conversión*.

Objetos nativos de JavaScript

- El objeto Date – Métodos:

Métodos				
getDate()	getTime()	getUTCMonth()	setMonth()	setUTCMonth()
getDay()	getTimezoneOffset()	getUTCSeconds()	setSeconds()	setUTCSeconds()
getFullYear()	getUTCDate()	parse()	setTime()	toDateString()
getHours()	getUTCDay()	setDate()	setUTCDate()	toLocaleDateString()
getMilliseconds()	getUTCFullYear()	setFullYear()	setUTCFullYear()	toLocaleTimeString()
getMinutes()	getUTCHours()	setHours()	setUTCHours()	toLocaleString()
getMonth()	getUTCMilliseconds()	setMilliseconds()	setUTCMilliseconds()	toTimeString()
getSeconds()	getUTCMinutes()	setMinutes()	setUTCMinutes()	toUTCString()
now()	toJSON()			

Objetos nativos de JavaScript

- El objeto Date:

```
<script type="text/javascript">
   var fActual = new Date();
   var fMaya = new Date(2012,11,21);
   alert("La fecha actual es: " + fActual);
   alert("El calendario Maya termina el: " + fMaya);
   var tRestante = fMaya - fActual;
   alert("Quedan " + tRestante + "milisegundos para que termine el calendario Maya");
</script>
```

Objetos nativos de JavaScript

- El objeto **Math**:
  - Permite realizar operaciones matemáticas complejas en JavaScript.

- El objeto Math – Métodos y propiedades:

Métodos		
abs()	exp()	random()
acos()	floor()	round()
asin()	log()	sin()
atan()	max()	sqrt()
ceil()	min()	tan()
cos()	pow()	sign()

Propiedades
E
LN2
LN10
LOG2E
LOG10E
PI
SQRT1_2
SQRT2

Objetos nativos de JavaScript

- El objeto **Number**:
  - Permite realizar tareas relacionadas con tipos de datos numéricos.
- El objeto Number – Métodos y propiedades:

Propiedades
MAX_VALUE
MIN_VALUE
NaN
NEGATIVE_INFINITY
POSITIVE_INFINITY

Devuelve String

Method	Description
toString()	
toExponential()	`var x = 9.656;` `x.toExponential(2);  // returns 9.66e+0`

toFixed()	`var x = 9.656;`
	`x.toFixed(0);  // returns 10`
	`x.toFixed(2);  // returns 9.66`

toPrecision()	`var x = 9.656;`
	`x.toPrecision();  // returns 9.656`
	`x.toPrecision(2);  // returns 9.7`

valueOf()	Returns a number as a number

Objetos nativos de JavaScript

- El objeto **String**:
  - Permite manipular las cadenas de texto.
- El objeto String – Métodos y propiedades:

Propiedades
Length

Métodos			
anchor()	fixed()	localeCompare(string)	strike()
big()	fontcolor()	match(regexp)	sub()
blink()	fontsize()	replace(searchvalue, newvalue)	substr(start, length)
bold()	fromCharCode(num)	search(searchvalue)	substring(start, end)
charAt(pos)	indexOf(searchvalue ,start)	slice(start,end)	sup()
charCodeAt(pos)	italics()	small()	toLowerCase()
concat(string,str ing,..)	lastIndexOf(searchv alue, start)	split(separator,limit)	toUpperCase() trim()
startsWith(string , length)	endsWith(string, length)	includes(string)	

ÍNDICE

1. OBJETOS NATIVOS DE JAVASCRIPT
2. **INTERACCIÓN DE LOS OBJETOS CON EL NAVEGADOR**
3. GENERACIÓN DE ELEMENTOS HTML DESDE CÓDIGO JAVASCRIPT
4. APLICACIONES PRÁCTICAS DE LOS MARCOS
5. GESTIÓN DE LAS VENTANAS

Interacción de los objetos con el navegador

- Además de los objetos presentados anteriormente, existe otro tipo de objetos que permiten manipular diferentes características del navegador en sí mismo.

Interacción de los objetos con el navegador

- El objeto 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.
  - Su suele utilizar para en base al resultado que ofrezca la información anterior, tomar una decisión sobre el qué código ejecutar

Interacción de los objetos con el navegador

- El objeto Navigator – Métodos y propiedades:

Métodos	Descripción
`javaEnable()`	Especifica si el navegador tiene o no habilitado Java

Propiedades	Descripción
`appCodeName`	Devuelve el nombre del código del navegador.
`appName`	Devuelve el nombre del navegador.
`appVersión`	Devuelve la versión del navegador.
`cookieEnable`	Determina si el navegador tiene las cookies habilitadas.
`platform`	Devuelve en que plataforma esta el navegador compilado
`userAgent`	Cadena que contiene la cabecera completa del agente enviada en una petición HTTP. Contiene la información de las propiedades appCodeName y appVersion.
`language`	Devuelve el idioma del navegador
`product`	Devuelve el motor de renderizado del navegador
`plugins`	Devuelve los plugins instalados en el navegador

Interacción de los objetos con el navegador

- El objeto Navigator – Métodos y propiedades:
  - Ejemplo

```
<script type="text/javascript">
    document.write("Navigator <b>appCodeName</b>: " + navigator.appCodeName + "<br>");
    document.write("Navigator <b>appName</b>: " + navigator.appName + "<br>");
    document.write("Navigator <b>appVersion</b>: " + navigator.appVersion + "<br>");
    document.write("Navigator <b>language</b>: " + navigator.language + "<br>");
    document.write("Navigator <b>platform</b>: " + navigator.platform + "<br>");
    document.write("Navigator <b>product</b>: " + navigator.product + "<br>");
    document.write("Navigator <b>userAgent</b>: " + navigator.userAgent + "<br>");
    document.write("Navigator <b>cookies</b>: " + navigator.cookieEnabled + "<br>");
    document.write("Navigator <b>onLine</b> (solo IE): " + navigator.onLine + "<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>
```

Interacción de los objetos con el navegador

- El objeto **Navigator** – Métodos y propiedades:
  - Chrome

Navigator **appCodeName**: Mozilla
Navigator **appName**: Netscape
Navigator **appVersion**: 5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36
Navigator **language**: es
Navigator **platform**: Win32
Navigator **product**: Gecko
Navigator **userAgent**: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36
Navigator **cookies**: true
Navigator **onLine** (solo IE): true
El navegador NO está preparado para soportar los Applets de Java

  - Mozilla

Navigator **appCodeName**: Mozilla
Navigator **appName**: Netscape
Navigator **appVersion**: 5.0 (Windows)
Navigator **language**: es-ES
Navigator **platform**: Win32
Navigator **product**: Gecko
Navigator **userAgent**: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0
Navigator **cookies**: true
Navigator **onLine** (solo IE): true
El navegador SÍ está preparado para soportar los Applets de Java

Interacción de los objetos con el navegador

- Muestra las características de los plugins que tiene instalado el navegador.

Interacción de los objetos con el navegador

- El objeto Screen:
  - Corresponde a la pantalla utilizada por el usuario.
  - Todas sus propiedades son solamente de lectura.
- El objeto Screen – Propiedades:

Propiedades	Descripción
`availHeight`	Devuelve la altura de la pantalla (excluyendo la barra de tareas de Windows)
`availWidth`	Devuelve el ancho de la pantalla (excluyendo la barra de tareas de Windows)
`colorDepth`	Devuelve la profundidad de bits de la paleta de colores para la visualización de imágenes
`height`	Devuelve la altura total de la pantalla
`width`	Devuelve el ancho total de la pantalla

Interacción de los objetos con el navegador

- El objeto Screen:

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

Altura total: 768
Altura disponible: 728
Anchura total: 1366
Anchura disponible: 1366
Profundidad de color: 24

Interacción de los objetos con el navegador

- ¿Qué dimensiones tienen las barras de herramientas de tu navegador?

Interacción de los objetos con el navegador

- El objeto **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.
- El objeto **History** – Métodos y propiedades:

Métodos	Descripción
`back()`	Carga la URL anterior que existe en el historial de navegación
`forward()`	Carga la siguiente URL que existe en el historial de navegación
`go()`	Carga la URL específica que le indiquemos

Propiedades	Descripción
`length`	Devuelve la cantidad de URLs que contiene el historial de navegación

Interacción de los objetos con el navegador

- El objeto History:

```
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Document</title>
  </head>
  <body>
    <form>
     <input type="button" value="Atras‟ onClick="history.back();">
     <input type="button" value="Adelante" onClick="history.forward();">
    </form>
  </body>
</html>
```

Interacción de los objetos con el navegador

- El objeto **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.
  - Gracias a este objeto podemos recargar una página, cargar una nueva o reemplazar por otra

```
protocolo://maquina_host[:puerto]/camino_al_recurso
```

Interacción de los objetos con el navegador

- El objeto **Location** – Métodos y propiedades:

Propiedades	Descripción
hash	http://www.example.com/test.htm#part2
host	protocolo://maquina_host[:puerto]/camino_al_recurso
hostname	protocolo://maquina_host:puerto]/camino_al_recurso
href	protocolo://maquina_host[:puerto]/camino_al_recurso
pathname	protocolo://maquina_host[:puerto]/camino_al_recurso
port	protocolo://maquina_host[:puerto]/camino_al_recurso
protocol	protocolo://maquina_host[:puerto]/camino_al_recurso
search	https://www.w3schools.com/submit.htm?email=someone@example.com
origin	protocolo://maquina_host[:puerto]/camino_al_recurso

Interacción de los objetos con el navegador

- El objeto **Location** – Métodos y propiedades:

Métodos	Descripción
`assign(url)`	Este método hace que la ventana cargue y muestre el documento de la URL especificada por parámetro.
`reload(forcedReload)`	Este método realiza una recarga del recurso de la URL actual Parámetro optativo (true → recarga desde el servidor, false → recarga desde la caché)
`replace(url)`	Este método reemplaza el recurso actual por el recurso de la URL pasada por parámetro.

Interacción de los objetos con el navegador

- El objeto Location – Métodos y propiedades:

```
<html lang="en">
  <head>
     <meta charset="UTF-8">
     <title>Document</title>
     <script type="text/javascript">
         document.write("URL completa: " + location.href + "<br>");
         document.write("Pathname: " + location.pathname + "<br>");
         document.write("Protocolo: " + location.protocol + "<br>");
     </script>
   </head>
   <body>
      <form>
         <input type="button" value="Recargar" onclick="location.reload()">
      </form>
   </body>
</html>
```

Interacción de los objetos con el navegador

- ¿Detectas alguna diferencia entre el método assign() y replace()?

Interacción de los objetos con el navegador

- El objeto **Window**:
  - Se considera el objeto más importante de JavaScript.
  - 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.

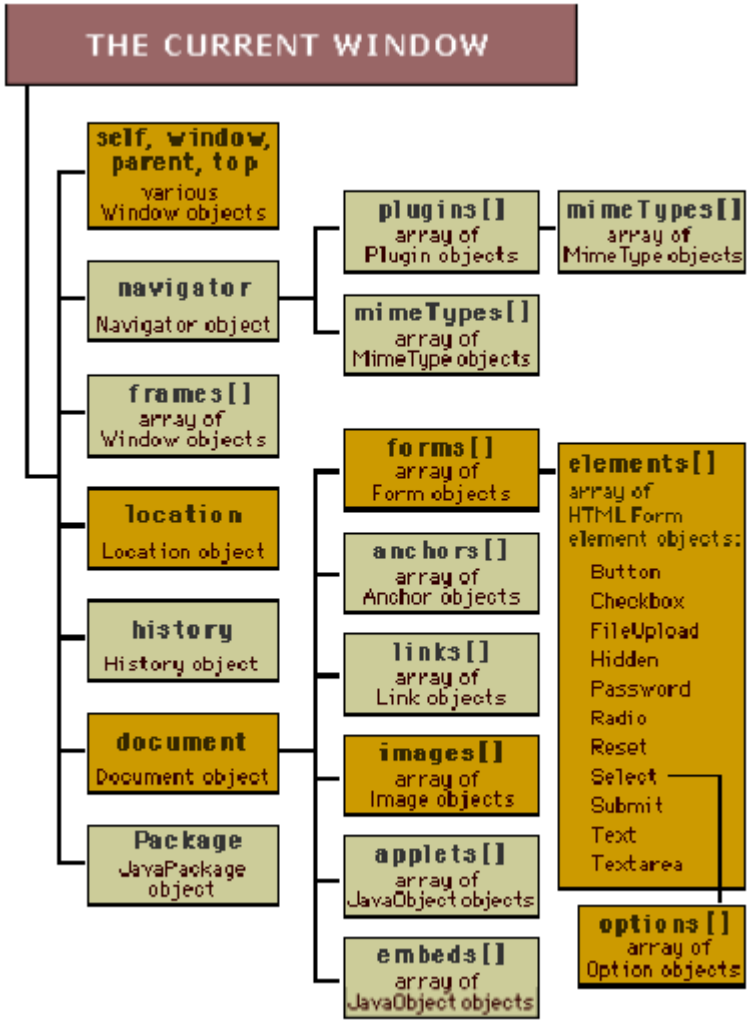
```
window.document.cookie
```

=

```
document.cookie
```

Interacción de los objetos con el navegador

- El objeto **Window**:



Interacción de los objetos con el navegador

- El objeto **Window** – Métodos y propiedades:

Métodos		
alert()	clearInterval()	setinterval()
getComputedStyle()	clearTimeOut()	setTimeOut()
blur()	moveTo()	scrollBy()
close()	moveBy()	scrollTo()
confirm()	open()	stop()
find()	prompt()	resizeBy()
focus()	resizeTo()	getSelection()
matchMedia()	print()	

Propiedades		
closed	location	pageYoffset
defaultStatus	screenLeft / screenX	parent
document	screenTop / screenY	screen
frames	name	scrollbars
history	opener	self
innerHeight	outerHeight	status
innerWidth	outerWidth	toolbar
length	pageXoffset	top
localStorage	sessionsStorage	

Interacción de los objetos con el navegador

- Correcta utilización para setTimeOut y setInterval

```
setTimeOut("unaFuncion()", 500);

//se debe cambiar por:
setTimeOut(unaFuncion, 500);

//y si necesita parámetros:
setTimeOut("unaFuncion(10, 20)", 500);

// se puede cambiar por:
setTimeOut(function (){
   unaFuncion(10, 20);
}, 500);
```

Interacción de los objetos con el navegador

- El objeto **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.

Interacción de los objetos con el navegador

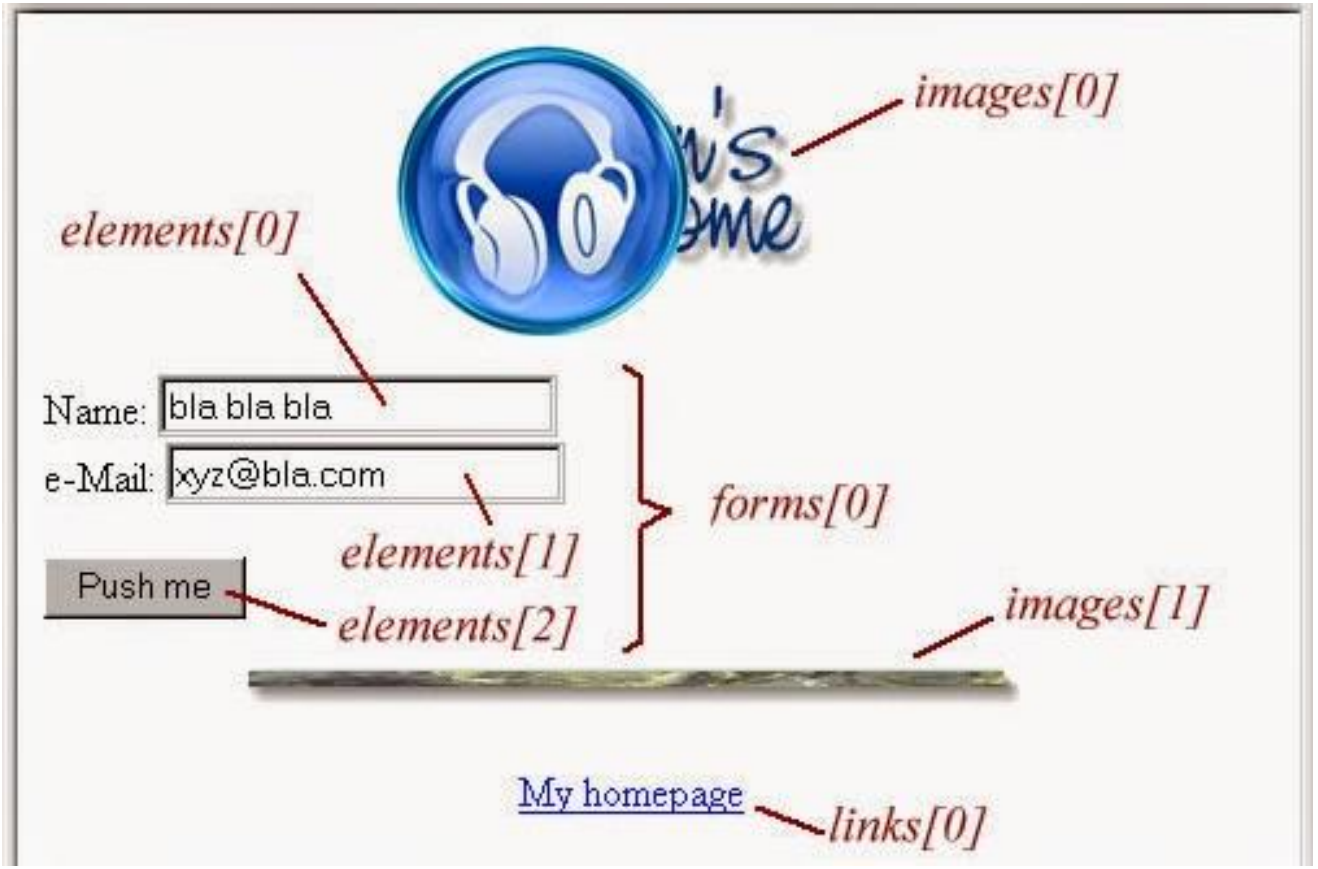
- El objeto **Document** – Métodos y propiedades:

Métodos	
addEventListener()	adoptNode()
createAttribute()	createComment()
createElement()	createTextNode()
getElementById()	getElementsByName()
getElementsByTagName()	hasFocus()
imporNode()	normalize()
open()	querySelector()
querySelectorAll()	removeEventListener()
renameNode()	write() / writeln()

Propiedades		
alinkColor	fgColor	plugins
anchors	forms	referrer
applets	images	title
bgColor	lastModified	URL
cookie	layers	vlinkColor
domain	linkColor	activeElement
embeds	links	baseURI
body	characterSet	doctype
documentElement	documentMode	documentURI
domain	head	Scripts

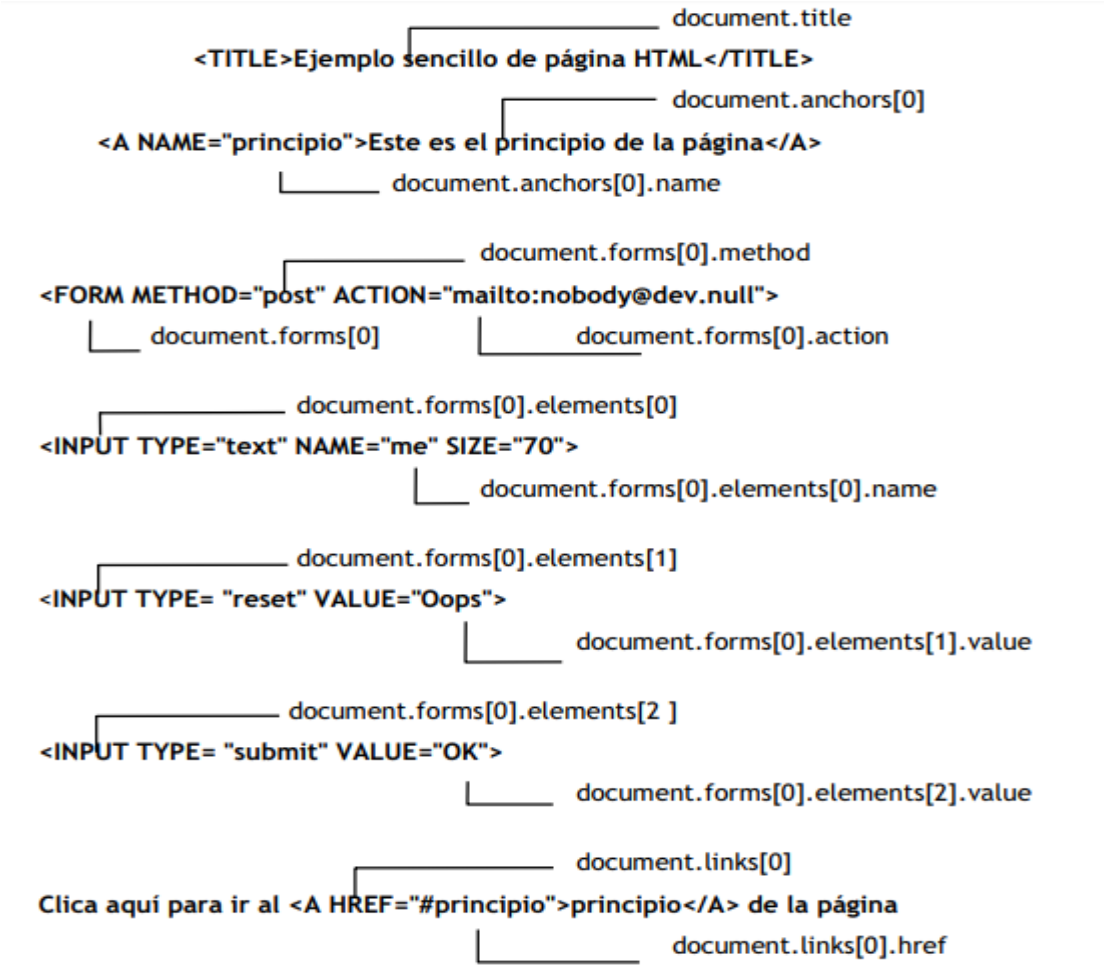
Interacción de los objetos con el navegador

- El objeto **Document** – Acceso a los elementos:



Interacción de los objetos con el navegador

- El objeto **Document** – Acceso a los elementos:



ÍNDICE

1. OBJETOS NATIVOS DE JAVASCRIPT
2. INTERACCIÓN DE LOS OBJETOS CON EL NAVEGADOR
3. **GENERACIÓN DE ELEMENTOS HTML DESDE CÓDIGO JAVASCRIPT**
4. APLICACIONES PRÁCTICAS DE LOS MARCOS
5. GESTIÓN DE LAS VENTANAS

Generación de elementos HTML desde código

- Uno de los principales objetivos de JavaScript es convertir un documento HTML estático en una aplicación web dinámica.

- Por ejemplo, es posible ejecutar instrucciones que crean nuevas ventanas con contenido propio, en lugar de mostrar dicho contenido en la ventana activa.

Generación de elementos HTML desde código

- Con JavaScript es posible manipular los objetos que representan el contenido de una página web con el fin de crear documentos dinámicos.
- Por ejemplo, es posible definir el título de una página web basándose en el SO utilizado:

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

Generación de elementos HTML desde código

- Si utilizamos document.write() después de que se haya cargado el documento borra todo su contenido.

```
<!DOCTYPE html>
<html>
<body>
  <h2>My First Web Page</h2>
  <p>My first paragraph.</p>
  <script>
  document.write(5 + 6);
  </script>
</body>
</html>
```

My First Web Page

My first paragraph.

11

```
<!DOCTYPE html>
<html>
<body>
  <h2>My First Web Page</h2>
  <p>My first paragraph.</p>
  <button type="button" onclick="document.write(5 + 6)">Try it</button>
</body>
</html>
```

My First Web Page

My first paragraph.

Try it

11

Generación de elementos HTML desde código

- Otro ejemplo es crear documentos en ventanas emergentes:

```
<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>
```

Generación de elementos HTML desde código

- La generación de código HTML a partir de JavaScript no se limita sólo a la creación de texto como en los ejemplos anteriores. 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>
```

Generación de elementos HTML desde código

- A partir del script anterior se obtiene la siguiente página web dinámica:

Generación de elementos HTML desde código

- Otra opción es utilizar la propiedad **innerHTML**

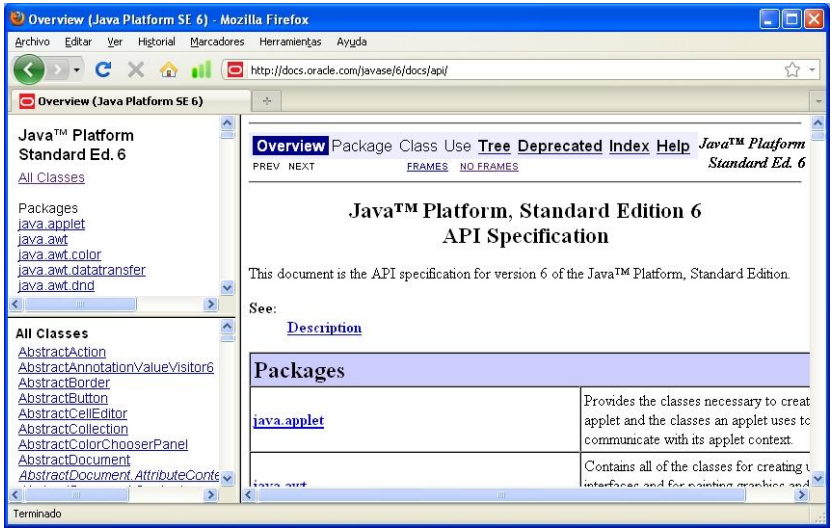
```
<script type="text/javascript">
    document.getElementById("p1").innerHTML = "Nuevo contenido";
</script>

<script type="text/javascript">
    document.getElementById("ul").innerHTML = "<li>Item 1</li>";
</script>
```

Aplicaciones prácticas de los marcos

- 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.

Aplicaciones prácticas de los marcos

- 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:



Aplicaciones prácticas de los marcos

- Los marcos se definen utilizando HTML mediante estas etiquetas:
  - `<frameset>`.
  - `<frame>`.

- Atributos de la etiqueta <frame>:

Atributos
frameborder
marginheight
marginwidth
name
noresize
scrolling
src

Aplicaciones prácticas de los marcos

- JavaScript permite manipular los marcos mediante las propiedades `frames`, `parent` y `top` del objeto `window`.
- Por ejemplo, se define un documento HTML con dos marcos:

```
<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>
</html>
```

Aplicaciones prácticas de los marcos

- El primer marco (`Marco1`) contiene la página `Marco1.html`:

```
<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>
```

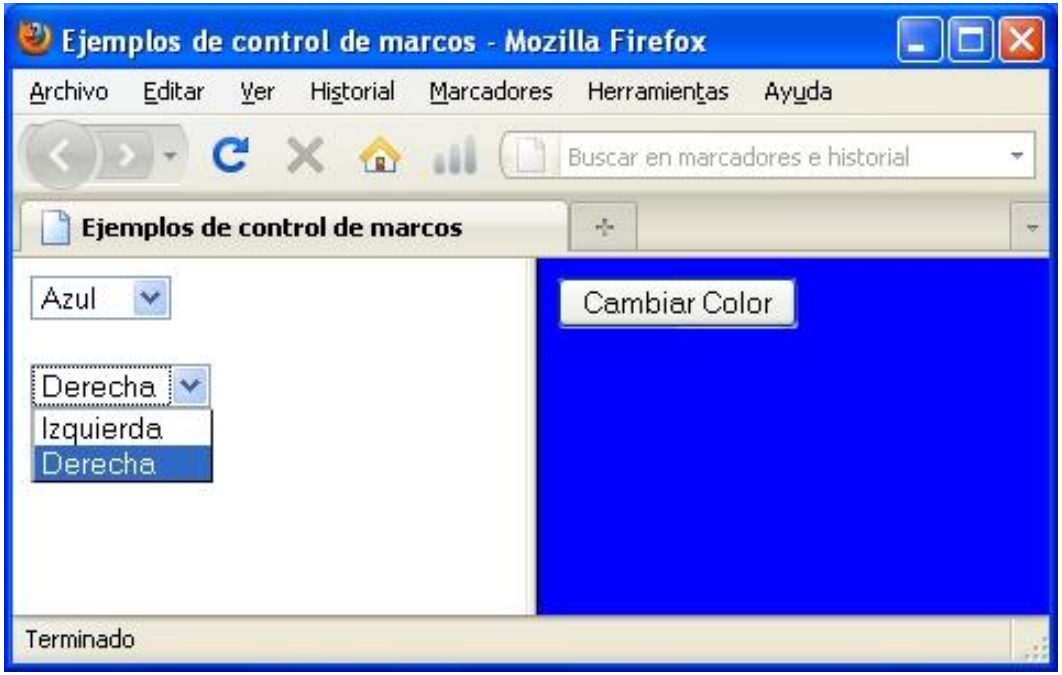
Aplicaciones prácticas de los marcos

- El segundo marco (`Marco2`) contiene la página `Marco2.html`:

```
<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>
```

Aplicaciones prácticas de los marcos

- El resultado se puede ver en esta imagen:



ÍNDICE

1. OBJETOS NATIVOS DE JAVASCRIPT
2. INTERACCIÓN DE LOS OBJETOS CON EL NAVEGADOR
3. GENERACIÓN DE ELEMENTOS HTML DESDE CÓDIGO JAVASCRIPT
4. APLICACIONES PRÁCTICAS DE LOS MARCOS
5. **GESTIÓN DE LAS VENTANAS**

Gestión de las ventanas

- JavaScript permite gestionar diferentes aspectos relacionados con las ventanas como por ejemplo abrir nuevas ventanas al presionar un botón.
- Cada una de estas ventanas tiene un tamaño, posición y estilo diferente.
- Estas ventanas emergentes suelen tener un contenido dinámico.

Gestión de las ventanas

- **Abrir y cerrar nuevas ventanas:**
  - Es una operación muy común en las páginas web.
  - En algunas ocasiones se abren sin que el usuario haga nada.
  - HTML permite abrir nuevas ventanas pero no permite ningún control posterior sobre ellas.

Gestión de las ventanas

- **Abrir y cerrar nuevas ventanas:**
  - Con JavaScript es posible abrir una ventana vacía mediante el método `open()`:
    - `nuevaVentana = window.open();`
  - De este modo la variable llamada `nuevaVentana` contendrá una referencia a la ventana creada.

Gestión de las ventanas

- **Abrir y cerrar nuevas ventanas:**
  - El método `open()` cuenta con cuatro parámetros y todos son opcionales:
    - URL.
    - Nombre de la ventana.
    - Colección de atributos que definen la apariencia de la ventana.
    - True (URL reemplaza al documento actual), false (lo añade)
  - Ejemplo:

```
nuevaVentana = window.open("http://www.misitioWeb.com/ads",
"Publicidad", "height=100, width=100");
```

Gestión de las ventanas

- Un ejemplo completo:

```
<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>
```

Gestión de las ventanas

- Para cerrar una ventana se puede invocar el método `close()`:

```
myWindow1.document.write('<input type=button
value=Cerrar onClick=window.close()>');
```

Gestión de las ventanas

- **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
top	left

Gestión de las ventanas

- **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.

Gestión de las ventanas

- **Comunicación entre ventanas:**
  - En el siguiente ejemplo se muestra cómo acceder a una ventana secundaria:

```
<html>
<head></head>
<body>
  <script>
   var ventanaSecundaria = window.open("", "ventanaSec”, "width=500,
   height=500");
  </script>
  <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>
```