

Utilización de los objetos predefinidos en JavaScript

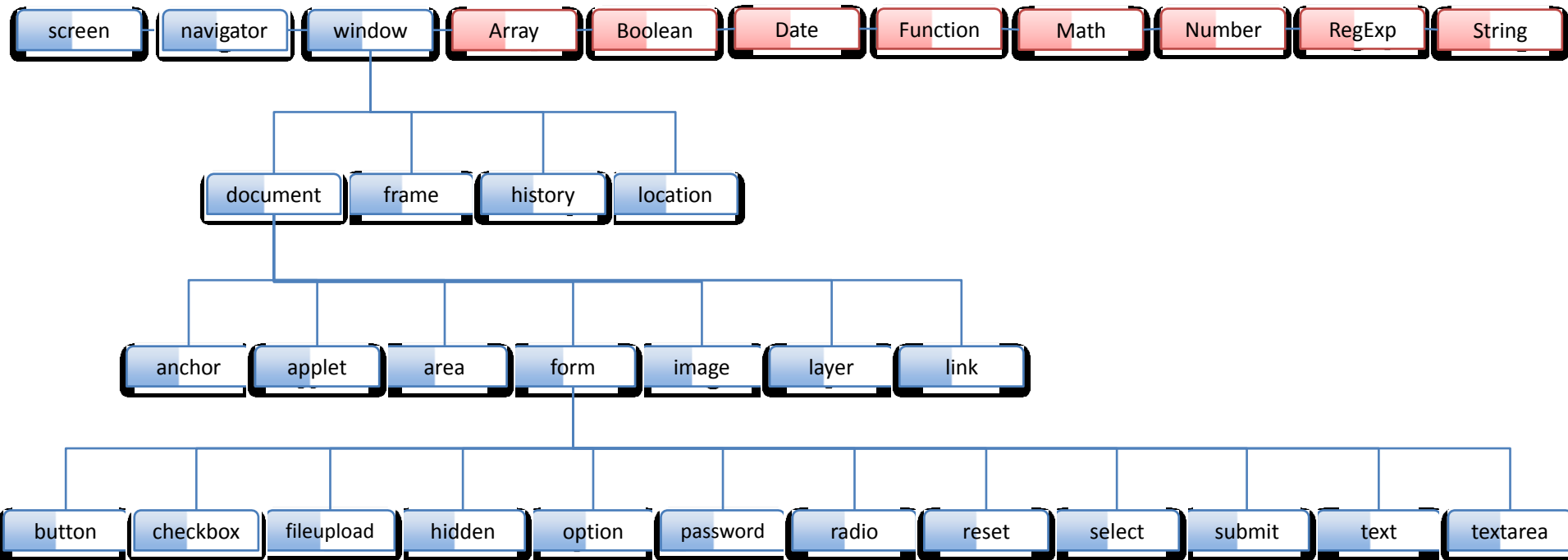
Objetos nativos

Módulo: Desarrollo Web en entornos cliente

CFGs Desarrollo de Aplicaciones Web

Objetos nativos de JavaScript

Los objetos de JavaScript se ordenan de modo jerárquico.



Objetos nativos de JavaScript - Date

- Permite realizar controles relacionados con el tiempo en las aplicaciones web.
 - Permite obtener la fecha y hora actual
 - Puede almacenar fechas
 - Permite realizar cálculo de fechas
 - Convierte fechas a Strings
- Cuenta con una serie de métodos divididos en tres subconjuntos:
 - Métodos de lectura.
 - Métodos de escritura.
 - Métodos de conversión.

Objetos nativos de JavaScript - Date

¿Formas de crear un objeto Date?

Cuatro formas de crearlo

```
var fecha1 = new Date();
```

```
var fecha2 = new Date(949278000000);
```

- el número de milisegundos desde 1/1/1970

```
var fecha3 = new Date("31 January 2000");
```

```
var fecha4 = new Date(2000, 0, 31, 15, 35, 20, 20);
```

- 0 es Enero, 11 es Diciembre

Objetos nativos de JavaScript - Date

Métodos

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

Objetos nativos de JavaScript - Date

Métodos

- `getDate()` Día del mes
- `getDay()` Día de la semana en número
- `getMonth()` El mes en número
- `getFullYear()` El año en formato de 4 dígitos

Objetos nativos de JavaScript - Date

Estableciendo valores con Date

- setDate()
- setMonth()
- setFullYear()

Ejemplos:

```
miFecha.setFullYear(2000);
```

```
miFecha.setDate(27);
```

```
miFecha.setMonth(1);
```

Nota: No hay método `setDay()`, el día se calcula por medio de los otros campos

Objetos nativos de JavaScript - Date

Ejemplo Date

```
var miFecha = new Date("1 Jan 2015");  
miFecha.setDate(32);  
document.write(miFecha);
```

32 no es un día válido en Enero, de manera que calcula 32 días desde el 1 de Enero, por tanto la fecha será el 1 de Febrero

```
> var miFecha = new Date("1 Jan 2015");  
< undefined  
> miFecha.setDate(32);  
< 1422748800000  
> console.log(miFecha)  
Sun Feb 01 2015 00:00:00 GMT+0000 (Hora estándar GMT)
```


Objetos nativos de JavaScript - Date

Cálculo de fechas

```
var fechaActual = new Date();  
var diaActual = fechaActual.getDate();  
fechaActual.setDate(diaActual + 28);
```

Añade 28 días a la fecha

Objetos nativos de JavaScript - Date

Ejercicio

Crea un programa que llame a una función que muestre la fecha en el siguiente formato...

"Hoy es el día 27 de Noviembre del año 2014"

Objetos nativos de JavaScript - Math

Permite realizar operaciones matemáticas complejas

Método	Descripción	Ejemplo
<code>abs(x)</code>	Valor absoluto de x	<code>abs(7.2)</code> es 7.2 <code>abs(0.0)</code> es 0.0 <code>abs(-5.6)</code> es 5.6
<code>ceil(x)</code>	redondea x al entero no menor que x	<code>ceil(9.2)</code> es 10.0 <code>ceil(-9.8)</code> es -9.0
<code>cos(x)</code>	Coseno de x (x en radianes)	<code>cos(0.0)</code> es 1.0
<code>exp(x)</code>	exponencial e^x	<code>exp(1.0)</code> es 2.71828 <code>exp(2.0)</code> es 7.38906
<code>floor(x)</code>	redondea x al entero superior no más grande que x	<code>floor(9.2)</code> es 9.0 <code>floor(-9.8)</code> es -10.0
<code>log(x)</code>	Logaritmo neperiano de x (base e)	<code>log(2.718282)</code> es 1.0 <code>log(7.389056)</code> es 2.0
<code>max(x, y)</code>	Mayor valor entre x e y	<code>max(2.3, 12.7)</code> es 12.7 <code>max(-2.3, -12.7)</code> es -2.3

Propiedades

E

LN2

LN10

LOG2E

LOG10E

PI

SQRT1_2

SQRT2

Objetos nativos de JavaScript - Math

<code>min(x, y)</code>	Valor menor entre x e y	<code>min(2.3, 12.7)</code> es 2.3 <code>min(-2.3, -12.7)</code> es -12.7
<code>pow(x, y)</code>	Elevar x a y	<code>pow(2.0, 7.0)</code> es 128.0 <code>pow(9.0, .5)</code> es 3.0
<code>round(x)</code>	Redondea x al entero más cercano	<code>round(9.75)</code> es 10 <code>round(9.25)</code> es 9
<code>sin(x)</code>	Seno de x (x en radianes)	<code>sin(0.0)</code> es 0.0
<code>sqrt(x)</code>	Raíz cuadrada de x	<code>sqrt(900.0)</code> es 30.0 <code>sqrt(9.0)</code> es 3.0
<code>tan(x)</code>	Tangente de x (x en radianes)	<code>tan(0.0)</code> es 0.0

Objetos nativos de JavaScript - Number

Permite realizar tareas relacionadas con tipos de datos numéricos.

Métodos
<code>toExponential()</code>
<code>toFixed()</code>
<code>toPrecision()</code>

Propiedades
<code>MAX_VALUE</code>
<code>MIN_VALUE</code>
<code>NaN</code>
<code>NEGATIVE_INFINITY</code>
<code>POSITIVE_INFINITY</code>

Objetos nativos de JavaScript - String

Permite manipular las cadenas de texto.

Métodos

<code>anchor()</code>	<code>fixed()</code>	<code>link()</code>	<code>strike()</code>
<code>big()</code>	<code>fontcolor()</code>	<code>match()</code>	<code>sub()</code>
<code>blink()</code>	<code>fontsize()</code>	<code>replace()</code>	<code>substr()</code>
<code>bold()</code>	<code>fromCharCode()</code>	<code>search()</code>	<code>substring()</code>
<code>charAt()</code>	<code>indexOf()</code>	<code>slice()</code>	<code>sup()</code>
<code>charCodeAt()</code>	<code>italics()</code>	<code>small()</code>	<code>toLowerCase()</code>
<code>concat()</code>	<code>lastIndexOf()</code>	<code>split()</code>	<code>toUpperCase()</code>

Propiedades

`length`

Objetos nativos de JavaScript - String

Métodos

Method	Description
<code>charAt(<i>index</i>)</code>	Returns a string containing the character at the specified <i>index</i> . If there is no character at the <i>index</i> , <code>charAt</code> returns an empty string. The first character is located at <i>index</i> 0.
<code>charCodeAt(<i>index</i>)</code>	Returns the Unicode value of the character at the specified <i>index</i> . If there is no character at the <i>index</i> , <code>charCodeAt</code> returns NaN (Not a Number).
<code>concat(<i>string</i>)</code>	Concatenates its argument to the end of the string that invokes the method. The string invoking this method is not modified; instead a new String is returned. This method is the same as adding two strings with the string concatenation operator + (e.g., <code>s1.concat(s2)</code> is the same as <code>s1 + s2</code>).
<code>fromCharCode(<i>value1</i>, <i>value2</i>,)</code>	Converts a list of Unicode values into a string containing the corresponding characters.
<code>indexOf(<i>substring</i>, <i>index</i>)</code>	Searches for the first occurrence of <i>substring</i> starting from position <i>index</i> in the string that invokes the method. The method returns the starting index of <i>substring</i> in the source string or -1 if <i>substring</i> is not found. If the <i>index</i> argument is not provided, the method begins searching from index 0 in the source string.
<code>lastIndexOf(<i>substring</i>, <i>index</i>)</code>	Searches for the last occurrence of <i>substring</i> starting from position <i>index</i> and searching toward the beginning of the string that invokes the method. The method returns the starting index of <i>substring</i> in the source string or -1 if <i>substring</i> is not found. If the <i>index</i> argument is not provided, the method begins searching from the end of the source string.

Objetos nativos de JavaScript - String

Métodos

<code>slice(<i>start</i>, <i>end</i>)</code>	Returns a string containing the portion of the string from index <i>start</i> through index <i>end</i> . If the <i>end</i> index is not specified, the method returns a string from the <i>start</i> index to the end of the source string. A negative <i>end</i> index specifies an offset from the end of the string starting from a position one past the end of the last character (so <i>-1</i> indicates the last character position in the string).
<code>split(<i>string</i>)</code>	Splits the source string into an array of strings (tokens) where its <i>string</i> argument specifies the delimiter (i.e., the characters that indicate the end of each token in the source string).
<code>substr(<i>start</i>, <i>length</i>)</code>	Returns a string containing <i>length</i> characters starting from index <i>start</i> in the source string. If <i>length</i> is not specified, a string containing characters from <i>start</i> to the end of the source string is returned.
<code>substring(<i>start</i>, <i>end</i>)</code>	Returns a string containing the characters from index <i>start</i> up to but not including index <i>end</i> in the source string.
<code>toLowerCase()</code>	Returns a string in which all uppercase letters are converted to lowercase letters. Non-letter characters are not changed.
<code>toUpperCase()</code>	Returns a string in which all lowercase letters are converted to uppercase letters. Non-letter characters are not changed.
<code>toString()</code>	Returns the same string as the source string.
<code>valueOf()</code>	Returns the same string as the source string.

Objetos nativos de JavaScript - String

Métodos

<i>Methods that generate XHTML tags</i>	
<code>anchor(<i>name</i>)</code>	Wraps the source string in an anchor element (<code><a></code>) with <i>name</i> as the anchor name.
<code>blink()</code>	Wraps the source string in a <code><blink></blink></code> element.
<code>fixed()</code>	Wraps the source string in a <code><tt></tt></code> element.
<code>link(<i>url</i>)</code>	Wraps the source string in an anchor element (<code><a></code>) with <i>url</i> as the hyperlink location.
<code>strike()</code>	Wraps the source string in a <code><strike></strike></code> element.
<code>sub()</code>	Wraps the source string in a <code><sub></sub></code> element.
<code>sup()</code>	Wraps the source string in a <code><sup></sup></code> element.

```
> a="Google"
< "Google"

> a.link("www.google.es");
< "<a href='www.google.es'>Google</a>"

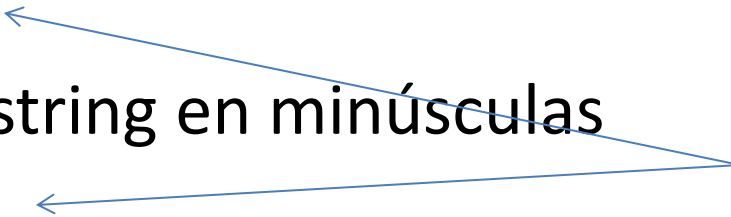
> a.sub();
< "<sub>Google</sub>"

> a.sup();
< "<sup>Google</sup>"
```

Objetos nativos de JavaScript - String

Métodos

- **charAt**
 - Devuelve el caracter de una posición
- **charCodeAt**
 - Devuelve el valor Unicode del caracter que ocupa la posición pasada
- **fromCharCode**
 - Devuelve el caracter a partir de su valor Unicode
- **toLowerCase**
 - Devuelve el string en minúsculas
- **toUpperCase**
 - Devuelve el string en mayúsculas



No modifican el string original

The diagram consists of a blue rounded rectangle containing the text 'No modifican el string original'. Two blue arrows originate from the right side of this rectangle. One arrow points to the 'toLowerCase' method in the list above, and the other points to the 'toUpperCase' method in the list above.

```

<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Métodos para caracteres -->

<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
    <title>Métodos para caracteres</title>
    <script type = "text/javascript">
<!--
    var s = "BAIFO";
    var s2 = "GgAaBbEeTtAa";

    document.writeln("<p>El caracter de la posición 0 de '" + s + "' es " + s.charAt(0));
    document.writeln("<br />El código del caracter de la posición 0 en '" + s + "' es " +
        s.charCodeAt(0) + "</p>" );

    document.writeln("<p>'"+ String.fromCharCode(65, 108, 105, 115, 73, 111, 83) +
        "' son los caracteres correspondientes a los códigos 65, 108, 105, 115,
        73, 111 y 83</p>" );
    document.writeln("<p>'"+ s2 + "' en minúsculas es '" + s2.toLowerCase() + "'");
    document.writeln("<br />'"+ s2 + "' en mayúsculas es '" + s2.toUpperCase() + "'</p>" );
// -->
    </script>
</head>
<body>
</body>
</html>

```

El caracter de la posición 0 de 'BAIFO' es B

El código del caracter de la posición 0 en 'BAIFO' es 66

'AlisIoS' son los caracteres correspondientes a los códigos 65, 108, 105, 115, 73, 111 y 83

'GgAaBbEeTtAa' en minúsculas es 'ggaabbeettaa'

'GgAaBbEeTtAa' en mayúsculas es 'GGAABBEETTAA'

Objetos nativos de JavaScript - String

Métodos

- `indexOf` y `lastIndexOf`: Busca una subcadena concreta dentro de un string

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Buscando cadenas con indexOf y lastIndexOf</title>

    <script type="text/javascript">
    <!--
    var letters="abcdefghijklmnopqrstuvwxyzabcdefghijklm";

    function buttonPressed() {
        buscar.first.value=letters.indexOf(buscar.inputVal.value);
        buscar.last.value=letters.lastIndexOf(buscar.inputVal.value);
        buscar.first12.value=letters.indexOf(buscar.inputVal.value, 12);
        buscar.last12.value=letters.lastIndexOf(buscar.inputVal.value, 12);
    }
    // -->
    </script>
</head>
<body>
    <form name="buscar" action="">
    <h1>Cadena en la que se va a buscar es:<br />abcdefghijklmnopqrstuvwxyzabcdefghijklm</h1>
    <p>
        Introduce la subcadena a buscar <input name="inputVal" type="text" />
        <input name="search" type="button" value="Buscar"onclick="buttonPressed()" /><br/>
    </p>
    <p>
        La primera apar. es en la pos. <input name="first" type="text" size="5" /><br/>
        La última apar. es en la pos.<input name="last" type="text" size="5" /><br/>
        Primera apar. a partir desde la pos. 12 en el índice <input name="first12" type="text"
size="5" /><br/>
        Última apar. a partir desde la pos. 12 en el índice <input name="last12" type="text"
size="5" />
    </p>
    </form>
</body>

```

Objetos nativos de JavaScript - String

La cadena en la que se va a buscar es:

abcdefghijklmnopqrstuvwxyzabcdefghijklm

Introduce la subcadena a buscar

La primera aparición es en la posición

La última aparición es en la posición

Primera aparición a partir desde la posición 12 en el índice

Última aparición a partir desde la posición 12 en el índice

La cadena en la que se va a buscar es:

abcdefghijklmnopqrstuvwxyzabcdefghijklm

Introduce la subcadena a buscar

La primera aparición es en la posición

La última aparición es en la posición

Primera aparición a partir desde la posición 12 en el índice

Última aparición a partir desde la posición 12 en el índice

Objetos nativos de JavaScript - String

Obtener Subcadenas

Example

Split a string into an array of substrings:

```
var str = "How are you doing today?";  
var res = str.split(" ");
```

The result of *res* will be an array with the values:

```
How,are,you,doing,today?
```

Syntax

```
string.split(separator,limit)
```

Example

Omit the separator parameter:

```
var str = "How are you doing today?";  
var res = str.split();
```

The result of *res* will be an array with only one value:

```
How are you doing today?
```

Example

Use the limit parameter:

```
var str = "How are you doing today?";  
var res = str.split(" ",3);
```

The result of *res* will be an array with only 3 values:

```
How,are,you
```

Example

Separate each charater, including white-space:

```
var str = "How are you doing today?";  
var res = str.split("");
```

The result of *res* will be an array with the values:

```
H,o,w, ,a,r,e, ,y,o,u, ,d,o,i,n,g, ,t,o,d,a,y,?
```

Example

Use a letter as a separator:

```
var str = "How are you doing today?";  
var res = str.split("o");
```

The result of *res* will be an array with the values:

```
H,w are y,u d,ing t,day?
```

Objetos nativos de JavaScript - Navegador

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

Browser BOM

Window

Navigator

Screen

History

Location

Objetos nativos de JavaScript - 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.
- El objeto **Navigator** – Métodos y propiedades:

Métodos

`javaEnable()`

Propiedades

`appCodeName`

`appName`

`appVersión`

`cookieEnable`

`platform`

`userAgent`

```
◀ window.navigator.cookieEnabled
▶ true
◀ window.navigator.platform
▶ "Win32"
◀ window.navigator.plugins
▶ PluginArray { 0: Plugin, 1: Plugin, 2: Plugin, 3: Plugin, 4: Plugin, 5: Plugin, 6: Plugin, 7: Plugin, 8: Plugin, 9: Plugin, 23 más... }
```

Objetos nativos de JavaScript - Navegador



window.navigator.userAgent

`window.navigator.userAgent`

"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36"



◀ `window.navigator.userAgent`

▶ "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:32.0) Gecko/20100101 Firefox/32.0"



>>> `window.navigator.userAgent`

"Opera/9.80 (Windows NT 6.1; WOW64) Presto/2.12.388 Version/12.17"

```
var sBrowser, sUsrAg = navigator.userAgent;
if (sUsrAg.indexOf("Edge") > -1) {
    sBrowser = "Edge"; }
else if (sUsrAg.indexOf("Chrome") > -1) {
    sBrowser = "Google Chrome"; }
else if (sUsrAg.indexOf("Safari") > -1) {
    sBrowser = "Apple Safari"; }
else if (sUsrAg.indexOf("Opera") > -1) {
    sBrowser = "Opera"; }
else if (sUsrAg.indexOf("Firefox") > -1) {
    sBrowser = "Mozilla Firefox"; }
else if ((sUsrAg.indexOf("MSIE") > -1) ||
(sUsrAg.indexOf("Trident") > -1)) {
    sBrowser = "Microsoft Internet Explorer";
}
alert("Tu navegador es: " + sBrowser);
```

Example #1: Browser
detect and return a string

Objetos nativos de JavaScript - Navegador

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

Propiedades
<code>availHeight</code>
<code>availWidth</code>
<code>colorDepth</code>
<code>height</code>
<code>pixelDepth</code>
<code>width</code>

La diferencia entre la *height* y *availHeight* es que *height* es la resolución total, mientras *availHeight* resta menús del sistema operativo, como la barra de tareas de Windows. Lo mismo es el caso para la *width* y *availWidth*.

Objetos nativos de JavaScript - 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 jerarquía.

■ El objeto **window** – Métodos y propiedades:

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

Objetos nativos de JavaScript - Navegador

■ El objeto **window** – Métodos y propiedades:

Example

Open "www.w3schools.com" in a new browser window:

```
window.open("http://www.w3schools.com");
```

Try it yourself »

Syntax

```
window.open(URL, name, specs, replace)
```

myRef = window.open(about.html, 'mywin', 'left=20,top=20,width=500,height=500,toolbar=1,resizable=0');

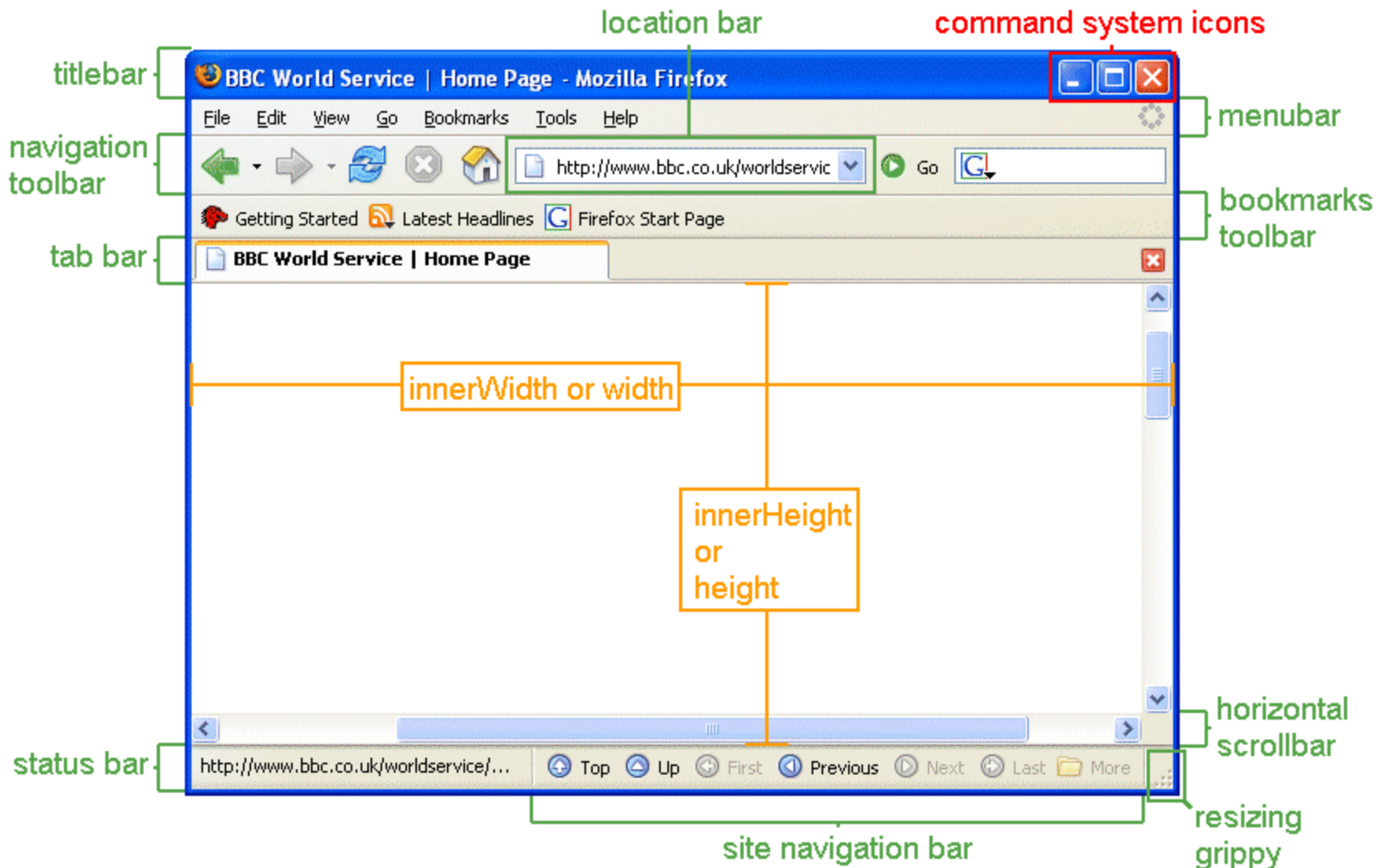
myRef = window.close()

Cierra la venta

window.moveTo(50, 50) – moverá la ventana del navegador a la posición de la ventana x= 50 e y = 50

window.moveBy(5, -5) – moverá la ventana 5px a la derecha y 5px hacia arriba desde la posición actual

Objetos nativos de JavaScript - Navegador



Objetos nativos de JavaScript - 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.

Objetos nativos de JavaScript - Navegador

- El objeto Document – Métodos y propiedades:

Métodos	
<code>captureEvents()</code>	<code>open()</code>
<code>close()</code>	<code>releaseEvents()</code>
<code>getSelection()</code>	<code>routeEvents()</code>
<code>handleEvent()</code>	<code>write()</code>
<code>home()</code>	<code>writeln()</code>

Propiedades		
<code>alinkColor</code>	<code>fgColor</code>	<code>plugins</code>
<code>anchors</code>	<code>forms</code>	<code>referrer</code>
<code>applets</code>	<code>images</code>	<code>title</code>
<code>bgColor</code>	<code>lastModified</code>	<code>URL</code>
<code>cookie</code>	<code>layers</code>	<code>vlinkColor</code>
<code>domain</code>	<code>linkColor</code>	
<code>embeds</code>	<code>links</code>	

Objetos nativos de JavaScript - 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.

Objetos nativos de JavaScript - Navegador

■ El objeto **History** – Métodos y propiedades:

Métodos	Propiedades
<code>back()</code>	<code>length</code>
<code>forward()</code>	
<code>go()</code>	

Para obtener el número de URLs en la lista del historial

```
window.history.length
```

```
11
```

No se puede ver la URL actual. Por motivos de seguridad lo siguiente no funciona:

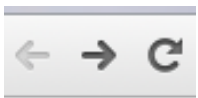


```
window.history[0]
```

```
undefined
```



`history.back()`



`history.forward()`

`history.back() = history.go(-1)`



`history.go(0)`

Objetos nativos de JavaScript - 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.

Objetos nativos de JavaScript - Navegador

- El objeto Location – Métodos y propiedades:

Métodos
<code>assign()</code>
<code>reload()</code>
<code>replace()</code>

Propiedades
<code>hash</code>
<code>host</code>
<code>hostname</code>
<code>href</code>
<code>pathname</code>
<code>port</code>
<code>protocol</code>
<code>search</code>

```
for (var i in location) { console.log(i+'='+location[i]);}  
replace=function () { [native code] }  
assign=function () { [native code] }  
ancestorOrigins=[object DOMStringList]  
origin=http://www.w3schools.com  
hash=  
search=  
pathname=/jsref/obj_history.asp  
port=  
hostname=www.w3schools.com  
host=www.w3schools.com  
protocol=http:  
href=http://www.w3schools.com/jsref/obj_history.asp  
reload=function reload() { [native code] }  
undefined
```

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

- 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>
```

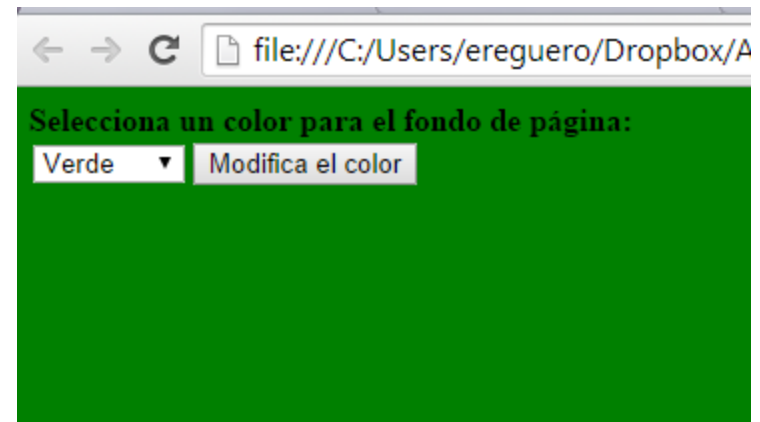
Generación de elementos HTML desde código

- 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:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
  <title>Cambio de color</title>
</head>
<body>
  <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>
</body>
</html>
```


Generación de elementos HTML desde código

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

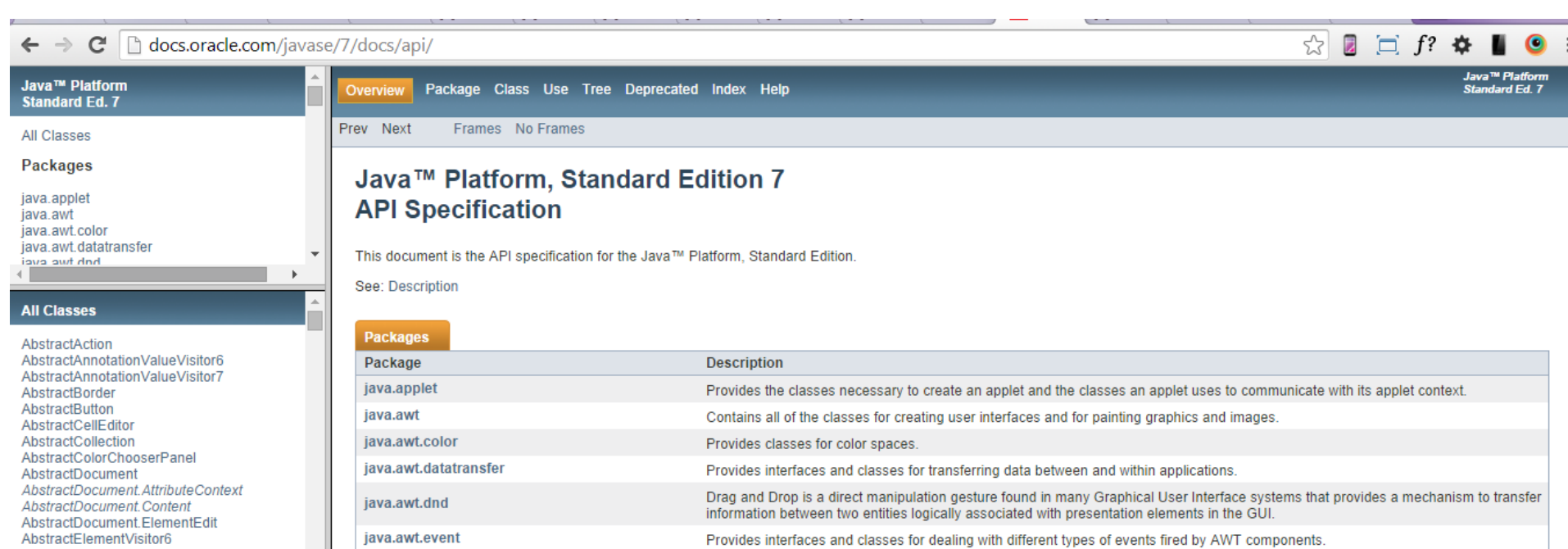


Aplicaciones prácticas de los marcos

- 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:



The screenshot displays the Java Platform, Standard Edition 7 API Specification website. The browser address bar shows the URL `docs.oracle.com/javase/7/docs/api/`. The page has a fixed left sidebar and a main content area.

Left Sidebar:

- Java™ Platform Standard Ed. 7**
- All Classes**
- Packages**
 - `java.applet`
 - `java.awt`
 - `java.awt.color`
 - `java.awt.datatransfer`
 - `java.awt.dnd`
- All Classes**
 - `AbstractAction`
 - `AbstractAnnotationValueVisitor6`
 - `AbstractAnnotationValueVisitor7`
 - `AbstractBorder`
 - `AbstractButton`
 - `AbstractCellEditor`
 - `AbstractCollection`
 - `AbstractColorChooserPanel`
 - `AbstractDocument`
 - `AbstractDocument.AttributeContext`
 - `AbstractDocument.Content`
 - `AbstractDocument.ElementEdit`
 - `AbstractElementVisitor6`
 - `AbstractElementVisitor7`

Main Content Area:

- Overview** | Package | Class | Use | Tree | Deprecated | Index | Help
- Prev | Next | Frames | No Frames
- Java™ Platform, Standard Edition 7 API Specification**
- This document is the API specification for the Java™ Platform, Standard Edition.
- See: Description
- Packages**

Package	Description
<code>java.applet</code>	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
<code>java.awt</code>	Contains all of the classes for creating user interfaces and for painting graphics and images.
<code>java.awt.color</code>	Provides classes for color spaces.
<code>java.awt.datatransfer</code>	Provides interfaces and classes for transferring data between and within applications.
<code>java.awt.dnd</code>	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
<code>java.awt.event</code>	Provides interfaces and classes for dealing with different types of events fired by AWT components.

Aplicaciones prácticas de los marcos

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

Atributos
<code>frameborder</code>
<code>marginheight</code>
<code>marginwidth</code>
<code>name</code>
<code>noresize</code>
<code>scrolling</code>
<code>src</code>

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>  
  <body></body>  
</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

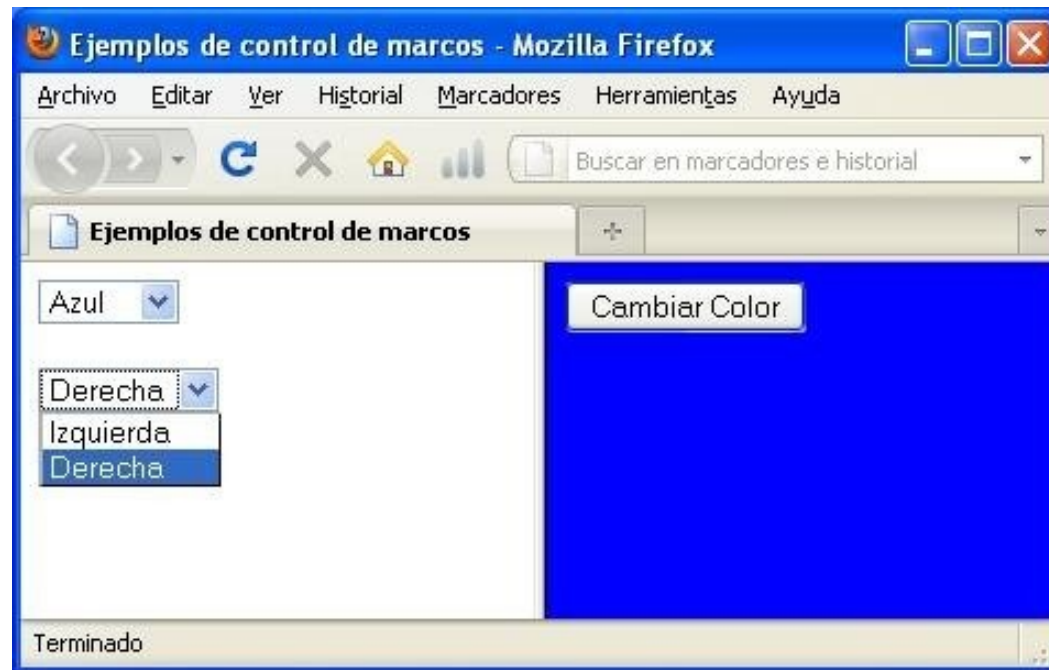
- El segundo marco (Marco2) contiene la página

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:



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 algo.
 - 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.
 - Con JavaScript es posible abrir una ventana mediante el método `open()`:
 - `nuevaVentana = window.open();`
- método `open()` cuenta con tres parámetros:
- URL.
 - Nombre de la ventana.
 - Colección de atributos que definen la apariencia de la ventana.

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:
 - Las ventanas cuentan con propiedades que permiten decidir su tamaño, ubicación o los elementos que contendrá.

Propiedades	
directories	scrollbars
height	status
menubar	toolbar
resizable	width

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>
  <center><h1> Comunicaci&oacute;n 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></center></body></html>
```