DESARROLLO WEB EN ENTORNO CLIENTE

CAPÍTULO 3:

Utilización de los objetos predefinidos de JavaScript

Juan Manuel Vara Mesa
Marcos López Sanz
David Granada
Emanuel Irrazábal
Jesús Javier Jiménez Hernández
Jenifer Verde Marín

Ra-Ma®

Objetos nativos de JavaScript

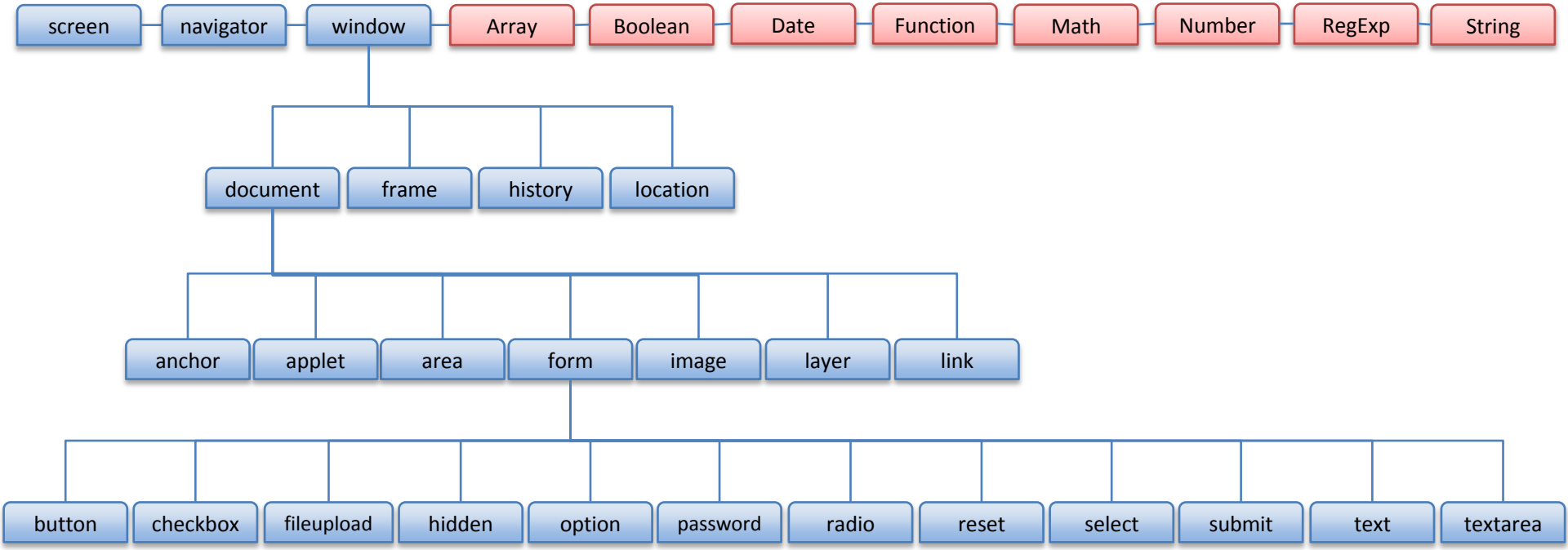
- JavaScript proporciona una serie de objetos definidos nativamente que no dependen del navegador.
- Para crear un objeto se utiliza la palabra clave `new`. Ejemplo:
  - `var mi_objeto = new Object();`

Objetos nativos de JavaScript

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

Objetos nativos de JavaScript

- Los objetos de JavaScript se ordenan de modo jerárquico.



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()

Objetos nativos de JavaScript

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

Objetos nativos de 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()	

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.

Objetos nativos de JavaScript

- El objeto Number – Métodos y propiedades:

Métodos
`toExponential()`
`toFixed()`
`toPrecision()`

Propiedades
`MAX_VALUE`
`MIN_VALUE`
`NaN`
`NEGATIVE_INFINITY`
`POSITIVE_INFINITY`

Objetos nativos de JavaScript

- El objeto String:
  - Permite manipular las cadenas de texto.

Objetos nativos de JavaScript

- El objeto String – Métodos y propiedades:

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

Propiedades
Lenght

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.

Interacción de los objetos con el navegador

- El objeto Navigator – Métodos y propiedades:

Métodos
`javaEnable()`

Propiedades
`appCodeName`
`appName`
`appVersión`
`cookieEnable`
`platform`
`userAgent`

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.

Interacción de los objetos con el navegador

- El objeto Screen – Propiedades:

Propiedades
availHeight
availWidth
colorDepth
height
pixelDepth
width

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.

Interacción de los objetos con el navegador

- 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

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	
captureEvents()	open()
close()	releaseEvents()
getSelection()	routeEvents()
handleEvent()	write()
home()	writeln()

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

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.

Interacción de los objetos con el navegador

- El objeto History – Métodos y propiedades:

Métodos
`back()`
`forward()`
`go()`

Propiedades
`current`
`length`
`next`
`previous`

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.

Interacción de los objetos con el navegador

- El objeto Location – Métodos y propiedades:

Métodos
`assign()`
`reload()`
`replace()`

Propiedades
`hash`
`host`
`hostname`
`href`
`pathname`
`port`
`protocol`
`search`

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:

```
<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:

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>`.

Aplicaciones prácticas de los marcos

- 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>
  <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>
```

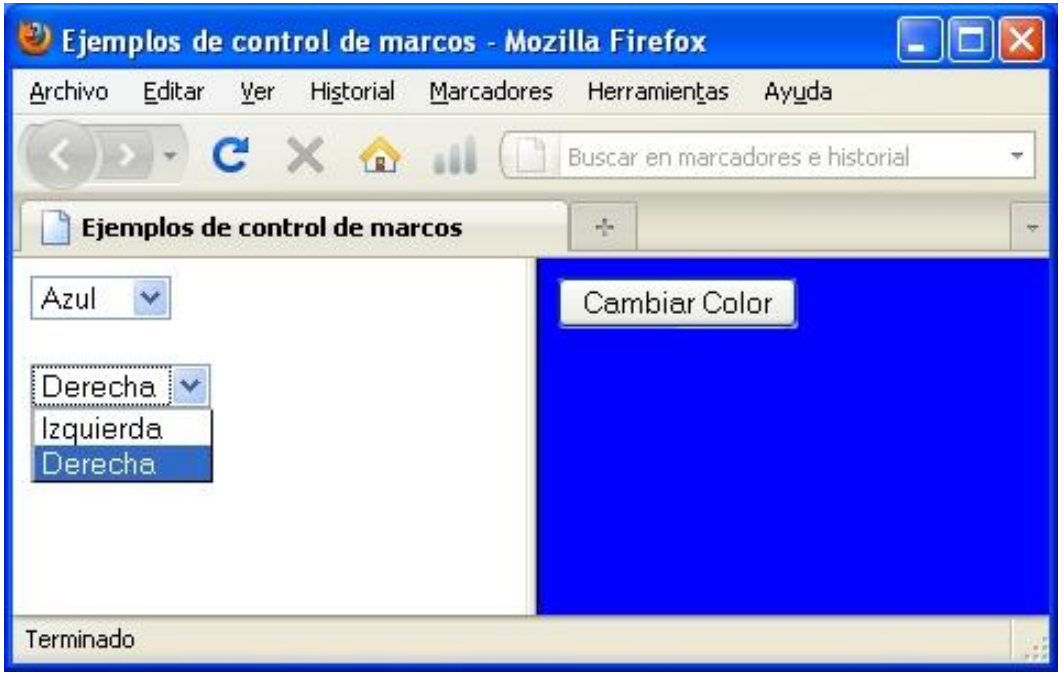
Aplicaciones prácticas de los marcos

- 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.

Gestión de las ventanas

- Abrir y cerrar nuevas ventanas:
  - El 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, widht=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`

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>
```