PROGRAMACIÓN WEB

JavaScript 2da Parte

PROGRAMACIÓN WEB

PERO ANTES...

- Que vamos a ver en este Curso de Programación Web???
  - Introducción
  - HTML y XHTML
  - CSS (Cascading Style Sheets)
  - Javascript
  - XML
  - Web 1.0

JAVASCRIPT

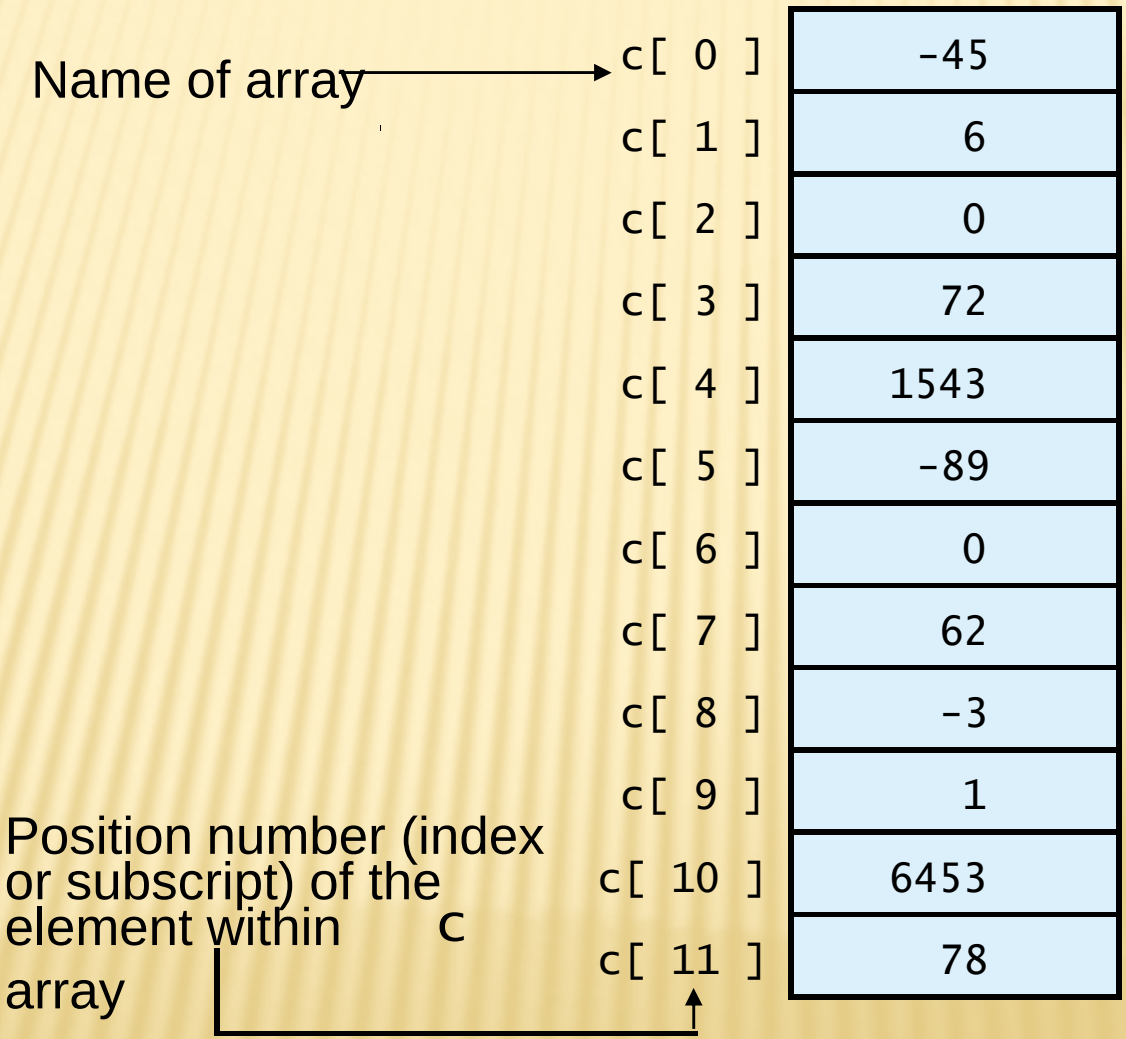
JAVASCRIPT - ARREGLOS

- Arreglos (Arrays)
  - Estructura de datos de items relacionados
    - Tambien llamados collections
  - Dinamico

JAVASCRIPT - ARREGLOS

- Arreglos en Javascript
  - Cada elemento referenciado por un numero
    - Empieza desde el elemento cero
    - Subscript o indice
  - Acceso a un elemento en especifico
    - Nombre del arreglo
    - Corchetes (Brackets)
    - Numero de elementos
  - Los arreglos saben su longitud
    - Propiedad *length*

JAVASCRIPT - ARREGLOS



JAVASCRIPT - PRECEDENCIA

Operators	Associativity	Type
() [] .	left to right	highest
++ -- !	right to left	unary
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
&&	left to right	logical AND
||	left to right	logical OR
?:	right to left	conditional
= += -= *= /= %=	right to left	assignment

Precedence and associativity of the operators discussed so far.

JAVASCRIPT - ARREGLOS

- Declarando y asignando arreglos
  - Arreglos en memoria
    - Objetos
    - Operador new
      - Asigna memoria para los objetos
      - Operador de asignación de memoria dinámica

```
var c;
c = new Array( 12 );
```

JAVASCRIPT - ARREGLOS

- Ejemplos usando arreglos
  - Arreglos crecen dinámicamente
    - Asignar mas espacio a medida que mas ítems son añadidos
  - Se debe inicializar los elementos de un arreglo
    - El valor por defecto es *undefined*
    - Lo mas practico es hacerlo con bucles *for*
    - Hacer referencia a elementos no inicializados o elementos fuera de los limites del arreglo es un error

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 11.3: InitArray.html -->
6  <!-- Initializing an Array     -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Initializing an Array</title>
11
12       <script type = "text/javascript">
13          <!--
14          // this function is called when the <body> element's
15          // onload event occurs
16          function initializeArrays()
17          {
18             var n1 = new Array( 5 );    // allo
19             var n2 = new Array();       // allo
20
21             // assign values to each element of Array n1
22             for ( var i = 0; i < n1.length; ++i )
23                n1[ i ] = i;
```

Array n1 has five elements.

Array n2 is an empty array.

The for loop initializes the elements in n1 to their subscript numbers (0 to 4).

```
24
25          // create and initialize five-elements in Array n2
26          for ( i = 0; i < 5; ++i )
27             n2[ i ] = i;
28
29          outputArray( "Array n1 contains", n1 );
30          outputArray( "Array n2 contains", n2 );
31       }
32
33       // output "header" followed by a two-column table
34       // containing subscripts and elements of "theArray"
35       function outputArray( header, theArray )
36       {
37          document.writeln( "<h2>" + header + "</h2>" );
38          docu                                          =" +
39             "
40
41          docu                                          +
42             "align = \"left\">Subscript</th>" +
43             "<th align = \"left\">Value</th></thead><tbody>" );
```

The for loop adds five elements to Array n2 and initialize each element (

4).

Each function displays the contents of its respective Array in an XHTML table.

The second time function ouputArray is called, variable header gets the value of “Array n2 contains” and variable theArray gets the value of n2.

```
44
45             for ( var i = 0; i < theArray.length; i++ )
46                document.writeln( "<tr><td>" + i + "</td><td>" +
47                   theArray[ i ] + "</td></tr>" );
48
49             document.writeln( "</tbody></table>" );
50          }
51          // -->
52       </script>
53
54    </head><body onload = "initializeArrays()"></body>
55 </html>
```

JAVASCRIPT - ARREGLOS

- Es posible declarar e inicializar en un solo paso
  - Especificar lista de valores

```
var n = [ 10, 20, 30, 40, 50 ];
var n = new Array( 10, 20, 30, 40, 50 );
```

  - Tambien es posible inicializar solo unos valores
    - Dejar elementos no inicializados en blanco
    - Elementos inicializados por defecto son undefined

```
var n = [ 10, 20, , 40, 50 ];
```

JAVASCRIPT - ARREGLOS

- Sentencia for...in
  - Realiza una accion por cada elemento del arreglo
  - Itera sobre elementos del arreglo
    - Asigna cada elemento a una variable especifica, uno a la vez
  - Ignora elementos no existentes

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 11.5: SumArray.html     -->
6  <!-- Summing Elements of an Array -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Sum the Elements of an Array</title>
11
12       <script type = "text/javascript">
13          <!--
14          function start()
15          {
16             var theArray = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ];
17             var total1 = 0, total2 = 0;
18
19             for ( var i = 0; i < theArray.length; i++ )
20                total1 += theArray[ i ];
21
22             document.writeln( "Total using subscripts: " + total1 );
23
```

The `for` loop sums the values contained in the 10-element integer array called `theArray`.

```
24            for ( var element in theArray )
25               total2 += theArray[ element ];
26
27            document.writeln( "<br />Total using fo
28               total2 );
29         }
30         // -->
31      </script>
32
33   </head><body onload = "start()"></body>
34 </html>
```

Variable `element` is assigned a subscript in the range of 0 up to, but not including, `theArray.length`.

JAVASCRIPT - ARREGLOS

- Los arreglos pueden proveer una manera mas corta y limpia para las sentencias *switch*
  - Cada elemento representa un caso (*case*)

JAVASCRIPT - ARREGLOS

- Referencias y Parametros de Referencia
  - Dos maneras de pasar parametros
    - Pasar por valor
      - Pasa una copia del valor original
      - Por defecto para numeros y booleans
      - Variable original no es afectada
    - Pasar por referencia
      - Como los objetos son pasados, como los arreglos
      - Pasar la ubicación en memoria del valor
      - Permite acceso directo al valor original
      - Mejora el desempeño

JAVASCRIPT - ARREGLOS

- Pasando arreglos a funciones
  - El nombre de un arreglo es un argumento
  - No es necesario pasar el tamaño del arreglo
    - Los arreglos conocen su tamaño
  - Se pasa por referencia
    - Elementos individuales se pasan por valor si son números o booleans
- Array.join
  - Crea una cadena con todos los elementos del arreglo
  - Hay que especificar un separador

JAVASCRIPT - ARREGLOS

- Ordenar Arreglos
  - Array.sort
    - Por defecto usa comparaciones de Cadenas para determinar el orden de los elementos del Arreglo
    - Funcion de comparacion opcional
      - Retorna negativo si el primer argumento es menor que el segundo
      - Retorna cero si ambos argumentos son iguales
      - Retorna positivo si el primer argumento es mayor que el segundo

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 11.9: sort.html -->
6  <!-- Sorting an Array     -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Sorting an Array with Array Method sort</title>
11
12       <script type = "text/javascript">
13          <!--
14          function start()
15          {
16             var a = [ 10, 1, 9, 2, 8, 3, 7, 4, 6, 5 ];
17
18             document.writeln( "<h1>Sorting an Array</h1>" );
19             outputArray( "Data items in original order: ", a );
20             a.sort( compareIntegers );  // sort the array
21             outputArray( "Data items in ascending order: ", a );
22          }
```

Method `sort` takes as its optional argument the name of a function that compares two arguments and returns a value of –1, 0 or 1.

```
23
24          // outputs "header" followed by the contents of "theArray"
25          function outputArray( header, theArray )
26          {
27             document.writeln( "<p>" + header +
28                theArray.join( " " ) + "</p>" );
29          }
30
31          // comparison function for use with sort
32          function compareIntegers( value1, value2 )
33          {
34             return parseInt( value1 ) - parseInt( value2 );
35          }
36          // -->
37       </script>
38
39    </head><body onload = "start()"></body>
40 </html>
```

Function `compareIntegers` calculates the difference between the integer values of its arguments.

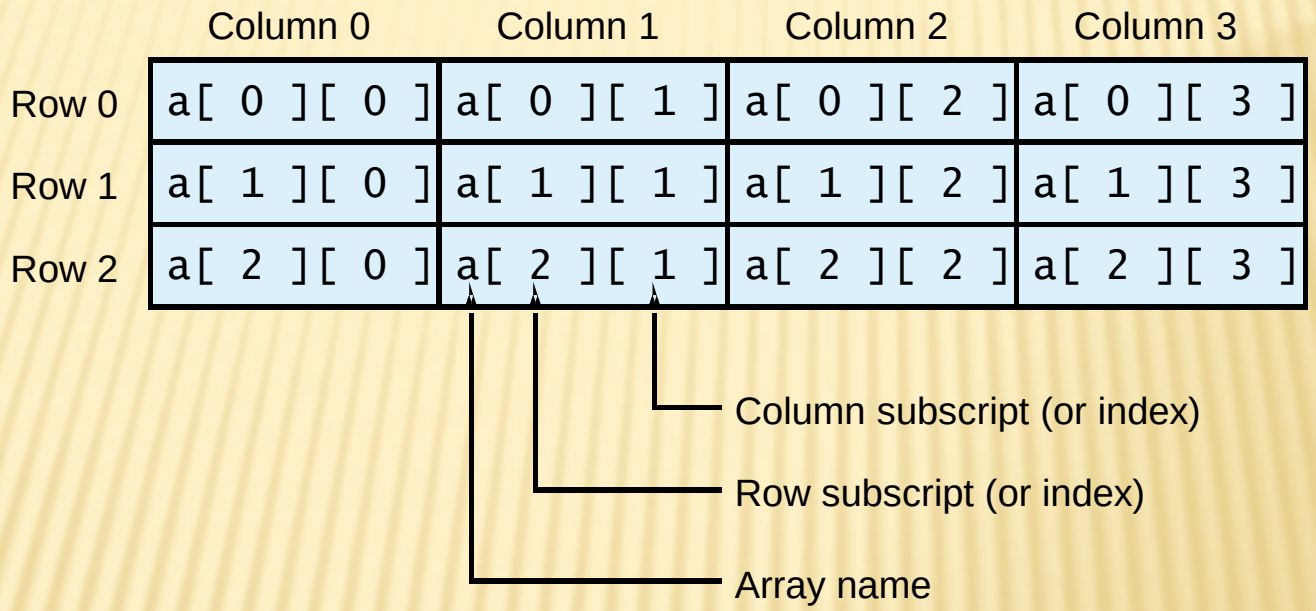
JAVASCRIPT - ARREGLOS

- Búsqueda en Arreglos
  - Búsqueda Lineal
  - Búsqueda Binaria
- Búsqueda Lineal
  - Itera a través de cada elemento hasta que encuentra el buscado (match)
  - Ineficiente
    - El peor caso es que evalué todo el arreglo
- Búsqueda Binaria
  - Requiere que los datos estén ordenados
  - Corta el rango de la búsqueda a la mitad en cada iteración
  - Eficiente
    - Solo busca sobre pequeños grupos de elementos

JAVASCRIPT - ARREGLOS

- Arreglos Multidimensionales
  - Los arreglos de dos dimensiones son equivalentes a las tablas
    - Filas y columnas
      - Se especifica primero la fila, luego la columna
    - Dos subscripts

JAVASCRIPT - ARREGLOS



JAVASCRIPT - ARREGLOS

- Declarando e inicializando arreglos multidimensionales
  - Agrupar por fila en corchetes
  - Tratar como arreglos de arreglos
  - Ej: Creamos el arreglo *b* con una fila de dos elementos y una segunda con tres elementos:

```
var b = [ [ 1, 2 ], [ 3, 4, 5 ] ];
```

JAVASCRIPT - ARREGLOS

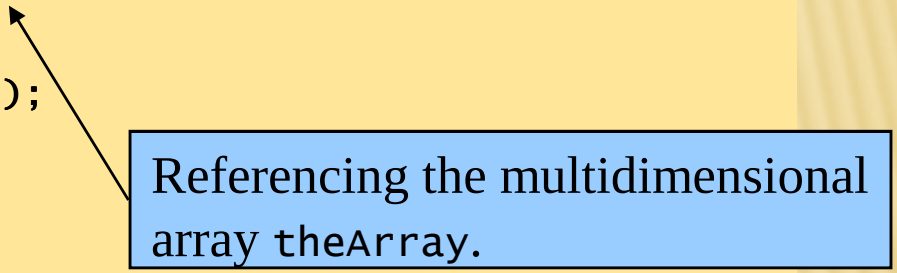
- También es posible usar el operador *new*
  - Crear arreglo b con dos filas, la primera de cinco columnas y la segunda con tres:

```
var b;

b = new Array( 2 );
b[ 0 ] = new Array( 5 );
b[ 1 ] = new Array( 3 );
```

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 11.13: InitArray3.html          -->
6  <!-- Initializing Multidimensional Arrays -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Initializing Multidimensional Arrays</title>
11
12       <script type = "text/javascript">
13          <!--
14          function start()
15          {
16             var array1 = [ [ 1, 2, 3 ],     // first row
17                            [ 4, 5, 6 ] ];   // second row
18             var array2 = [ [ 1, 2 ],        // first row
19                            [ 3 ],           // second row
20                            [ 4, 5, 6 ] ];   // third row
21
22             outputArray( "Values in array1 by row", array1 );
23             outputArray( "Values in array2 by row", array2 );
24          }
```

```
25
26          function outputArray( header, theArray )
27          {
28             document.writeln( "<h2>" + header + "</h2><tt>" );
29
30             for ( var i in theArray ) {
31
32                for ( var j in theArray[ i ] )
33                   document.write( theArray[ i ][ j ] + " " );
34
35                document.writeln( "<br />" );
36             }
37
38             document.writeln( "</tt>" );
39          }
40          // -->
41       </script>
42
43    </head><body onload = "start()"></body>
44 </html>
```



Objetos

JAVASCRIPT

JAVASCRIPT - OBJETOS

- Introduccion
  - Usar Javascript para manipular cada elemento de un documento XHTML desde un script
  - En esta seccion veremos varios de los Objetos de Javascript y demostraremos sus aptitudes

JAVASCRIPT - OBJETOS

- Objetos
  - Atributos
  - Funcionamiento
  - Encapsula datos y metodos
  - Propiedad de ocultar la informacion (interfaces)
  - Detalles de implementación de los objetos están ocultos dentro de los propios objetos

JAVASCRIPT - OBJETOS

- Objeto *Math*
  - Permite al programador realizar varios cálculos matemáticos comunes

JAVASCRIPT - OBJETOS

Method	Description	Example
`abs( x )`	absolute value of `x`	`abs( 7.2 )` is 7.2 `abs( 0.0 )` is 0.0 `abs( -5.6 )` is 5.6
`ceil( x )`	rounds `x` to the smallest integer not less than `x`	`ceil( 9.2 )` is 10.0 `ceil( -9.8 )` is -9.0
`cos( x )`	trigonometric cosine of `x` (`x` in radians)	`cos( 0.0 )` is 1.0
`exp( x )`	exponential method e^x	`exp( 1.0 )` is 2.71828 `exp( 2.0 )` is 7.38906
`floor( x )`	rounds `x` to the largest integer not greater than `x`	`floor( 9.2 )` is 9.0 `floor( -9.8 )` is -10.0
`log( x )`	natural logarithm of `x` (base *e*)	`log( 2.718282 )` is 1.0 `log( 7.389056 )` is 2.0
`max( x, y )`	larger value of `x` and `y`	`max( 2.3, 12.7 )` is 12.7 `max( -2.3, -12.7 )` is -2.3

JAVASCRIPT - OBJETOS

`min( x, y )`	smaller value of `x` and `y`	`min( 2.3, 12.7 )` is 2.3 `min( -2.3, -12.7 )` is -12.7
`pow( x, y )`	`x` raised to power `y` (xy)	`pow( 2.0, 7.0 )` is 128.0 `pow( 9.0, .5 )` is 3.0
`round( x )`	rounds `x` to the closest integer	`round( 9.75 )` is 10 `round( 9.25 )` is 9
`sin( x )`	trigonometric sine of `x` (`x` in radians)	`sin( 0.0 )` is 0.0
`sqrt( x )`	square root of `x`	`sqrt( 900.0 )` is 30.0 `sqrt( 9.0 )` is 3.0
`tan( x )`	trigonometric tangent of `x` (`x` in radians)	`tan( 0.0 )` is 0.0

Fig. 12.1 `Math` object methods.

JAVASCRIPT - OBJETOS

Constant	Description	Value
`Math.E`	Base of a natural logarithm (*e*).	Approximately 2.718.
`Math.LN2`	Natural logarithm of 2.	Approximately 0.693.
`Math.LN10`	Natural logarithm of 10.	Approximately 2.302.
`Math.LOG2E`	Base 2 logarithm of *e*.	Approximately 1.442.
`Math.LOG10E`	Base 10 logarithm of *e*.	Approximately 0.434.
`Math.PI`	π—the ratio of a circle's circumference to its diameter.	Approximately 3.141592653589793.
`Math.SQRT1_2`	Square root of 0.5.	Approximately 0.707.
`Math.SQRT2`	Square root of 2.0.	Approximately 1.414.

Fig. 12.2 Properties of the `Math` object.

JAVASCRIPT - OBJETOS

- Objeto String
  - La forma de procesar Cadenas y Caracteres en Javascript
  - Apropiado para procesar nombres, direcciones, informacion de tarjetas de credito, etc
- Caracteres (Characters)
- Cadena (String)
  - Serie de caracteres tratados como una sola unidad

JAVASCRIPT - OBJETOS

Method	Description
`charAt( index )`	Returns a string containing the character at the specified *index*. If there is no character at the *index*, `charAt` returns an empty string. The first character is located at *index* 0.
`charCodeAt( index )`	Returns the Unicode value of the character at the specified *index*. If there is no character at the *index*, `charCodeAt` returns `NaN` (Not a Number).
`concat( string )`	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., `s1.concat( s2 )` is the same as `s1 + s2`).
`fromCharCode( value1, value2, )`	Converts a list of Unicode values into a string containing the corresponding characters.
`indexOf( substring, index )`	Searches for the first occurrence of *substring* starting from position *index* in the string that invokes the method. The method returns the starting index of *substring* in the source string or –1 if *substring* is not found. If the *index* argument is not provided, the method begins searching from index 0 in the source string.
`lastIndexOf( substring, index )`	Searches for the last occurrence of *substring* starting from position *index* and searching toward the beginning of the string that invokes the method. The method returns the starting index of *substring* in the source string or –1 if *substring* is not found. If the *index* argument is not provided, the method begins searching from the end of the source string.

JAVASCRIPT - OBJETOS

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

JAVASCRIPT - OBJETOS

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

JAVASCRIPT - OBJETOS

- Métodos para procesar caracteres
  - *charAt*
    - Retorna el caracter en la posición especificada
  - *charCodeAt*
    - Retorna un valor Unicode del caracter en la posición especificada
  - *fromCharCode*
    - Retorna una cadena creada de una serie de valores Unicode
  - *toLowerCase*
    - Retorna la cadena en minuscula
  - *toUpperCase*
    - Retorna la cadena en mayuscula

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 12.4: CharacterProcessing.html -->
6  <!-- Character Processing Methods        -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Character Processing Methods</title>
11
12       <script type = "text/javascript">
13          <!--
14          var s = "ZEBRA";
15          var s2 = "AbCdEfG";
16
17          document.writeln( "<p>Character at index 0 in '" +
18             s + "' is " + s.charAt( 0 ) );
19          document.writeln( "<br />Character code at index 0 in '"
20             + s + "' is " + s.charCodeAt( 0 ) + "</p>" );
21
22          document.writeln( "<p>'" +
23             String.fromCharCode( 87, 79, 82, 68 ) +
24             "' contains character codes 87, 79, 82 and 68</p>" )
25
```

```
26          document.writeln( "<p>'" + s2 + "' in lowercase is '" +
27             s2.toLowerCase() + "'" );
28          document.writeln( "<br />'" + s2 + "' in uppercase is '"
29             + s2.toUpperCase() + "'</p>" );
30          // -->
31       </script>
32
33    </head><body></body>
34 </html>
```

JAVASCRIPT - OBJETOS

- Métodos de búsqueda
  - `indexOf` y `lastIndexOf`
    - Buscar una subcadena especifica en una cadena

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 12.5: SearchingStrings.html -->
6  <!-- Searching Strings                -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>
11          Searching Strings with indexOf and lastIndexOf
12       </title>
13
14       <script type = "text/javascript">
15          <!--
16          var letters = "abcdefghijklmnopqrstuvwxyzabcdefghijklm";
17
18          function buttonPressed()
19          {
20             searchForm.first.value =
21                letters.indexOf( searchForm.inputVal.value );
22             searchForm.last.value =
23                letters.lastIndexOf( searchForm.inputVal.value );
24             searchForm.first12.value =
25                letters.indexOf( searchForm.inputVal.value, 12 );
```

```
26            searchForm.last12.value =
27               letters.lastIndexOf(
28                  searchForm.inputVal.value, 12 );
29         }
30         // -->
31      </script>
32
33   </head>
34   <body>
35      <form name = "searchForm" action = "">
36         <h1>The string to search is:<br />
37             abcdefghijklmnopqrstuvwxyzabcdefghijklm</h1>
38         <p>Enter substring to search for
39         <input name = "inputVal" type = "text" />
40         <input name = "search" type = "button" value = "Search"
41                onclick = "buttonPressed()" /><br /></p>
42
43         <p>First occurrence located at index
44         <input name = "first" type = "text" size = "5" />
45         <br />Last occurrence located at index
46         <input name = "last" type = "text" size = "5" />
47         <br />First occurrence from index 12 located at index
48         <input name = "first12" type = "text" size = "5" />
49         <br />Last occurrence from index 12 located at index
50         <input name = "last12" type = "text" size = "5" /></p>
```

```
51         </form>
52     </body>
53 </html>
```

JAVASCRIPT - OBJETOS

- Elementos XHTML
  - Anchor
    - `<a name = “top”>` Anchor `</a>`
  - Blink
    - `<blink>` blinking text `</blink>`
  - Fixed
    - `<tt>` monospaced text `</tt>`
  - Strike
    - `<strike>` strike out text `</strike>`
  - Subscript
    - `<sub>` subscript `</sub>`
  - Superscript
    - `<sup>` superscript `</sup>`

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 12.7: MarkupMethods.html              -->
6  <!-- XHTML markup methods of the String object -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>XHTML Markup Methods of the String Object</title>
11
12       <script type = "text/javascript">
13          <!--
14          var anchorText = "This is an anchor",
15              blinkText = "This is blinking text",
16              fixedText = "This is monospaced text",
17              linkText = "Click here to go to anchorText",
18              strikeText = "This is strike out text",
19              subText = "subscript",
20              supText = "superscript";
21
22          document.writeln( anchorText.anchor( "top" ) );
23          document.writeln( "<br />" + blinkText.blink() );
24          document.writeln( "<br />" + fixedText.fixed() );
25          document.writeln( "<br />" + strikeText.strike() );
```

```
26          document.writeln(
27             "<br />This is text with a " + subText.sub() );
28          document.writeln(
29             "<br />This is text with a " + supText.sup() );
30          document.writeln(
31             "<br />" + linkText.link( "#top" ) );
32          // -->
33       </script>
34
35    </head><body></body>
36 </html>
```

JAVASCRIPT - OBJETOS

- Objeto Date
  - Provee métodos para manipular la fecha y el tiempo

JAVASCRIPT - OBJETOS

Method	Description
`getDate()` `getUTCDate()`	Returns a number from 1 to 31 representing the day of the month in local time or UTC, respectively.
`getDay()` `getUTCDay()`	Returns a number from 0 (Sunday) to 6 (Saturday) representing the day of the week in local time or UTC, respectively.
`getFullYear()` `getUTCFullYear()`	Returns the year as a four-digit number in local time or UTC, respectively.
`getHours()` `getUTCHours()`	Returns a number from 0 to 23 representing hours since midnight in local time or UTC, respectively.
`getMilliseconds()` `getUTCMilliSeconds()`	Returns a number from 0 to 999 representing the number of milliseconds in local time or UTC, respectively. The time is stored in hours, minutes, seconds and milliseconds.
`getMinutes()` `getUTCMinutes()`	Returns a number from 0 to 59 representing the minutes for the time in local time or UTC, respectively.
`getMonth()` `getUTCMonth()`	Returns a number from 0 (January) to 11 (December) representing the month in local time or UTC, respectively.
`getSeconds()` `getUTCSeconds()`	Returns a number from 0 to 59 representing the seconds for the time in local time or UTC, respectively.
`getTime()`	Returns the number of milliseconds between January 1, 1970 and the time in the `Date` object.
`getTimezoneOffset()`	Returns the difference in minutes between the current time on the local computer and UTC—previously known as Greenwich Mean Time (GMT).
`setDate(` *val* `)` `setUTCDate(` *val* `)`	Sets the day of the month (1 to 31) in local time or UTC, respectively.

Methods of the `Date` object.

JAVASCRIPT - OBJETOS

Method	Description
`setFullYear( y, m, d )` `setUTCFullYear( y, m, d )`	Sets the year in local time or UTC, respectively. The second and third arguments representing the month and the date are optional. If an optional argument is not specified, the current value in the `Date` object is used.
`setHours( h, m, s, ms )` `setUTCHours( h, m, s, ms )`	Sets the hour in local time or UTC, respectively. The second, third and fourth arguments representing the minutes, seconds and milliseconds are optional. If an optional argument is not specified, the current value in the `Date` object is used.
`setMilliSeconds( ms )` `setUTCMilliseconds( ms )`	Sets the number of milliseconds in local time or UTC, respectively.
`setMinutes( m, s, ms )` `setUTCMinutes( m, s, ms )`	Sets the minute in local time or UTC, respectively. The second and third arguments representing the seconds and milliseconds are optional. If an optional argument is not specified, the current value in the `Date` object is used.
`setMonth( m, d )` `setUTCMonth( m, d )`	Sets the month in local time or UTC, respectively. The second argument representing the date is optional. If the optional argument is not specified, the current date value in the `Date` object is used.
`setSeconds( s, ms )` `setUTCSeconds( s, ms )`	Sets the second in local time or UTC, respectively. The second argument representing the milliseconds is optional. If this argument is not specified, the current millisecond value in the `Date` object is used.

Fig. 12.8 Methods of the `Date` object.

JAVASCRIPT - OBJETOS

Method	Description
`setTime( ms )`	Sets the time based on its argument—the number of elapsed milliseconds since January 1, 1970.
`toLocaleString()`	Returns a string representation of the date and time in a form specific to the computer's locale. For example, September 13, 2001 at 3:42:22 PM is represented as *09/13/01 15:47:22* in the United States and *13/09/01 15:47:22* in Europe.
`toUTCString()`	Returns a string representation of the date and time in the form: *19 Sep 2001 15:47:22 UTC*
`toString()`	Returns a string representation of the date and time in a form specific to the locale of the computer (*Mon Sep 19 15:47:22 EDT 2001* in the United States).
`valueOf()`	The time in number of milliseconds since midnight, January 1, 1970.

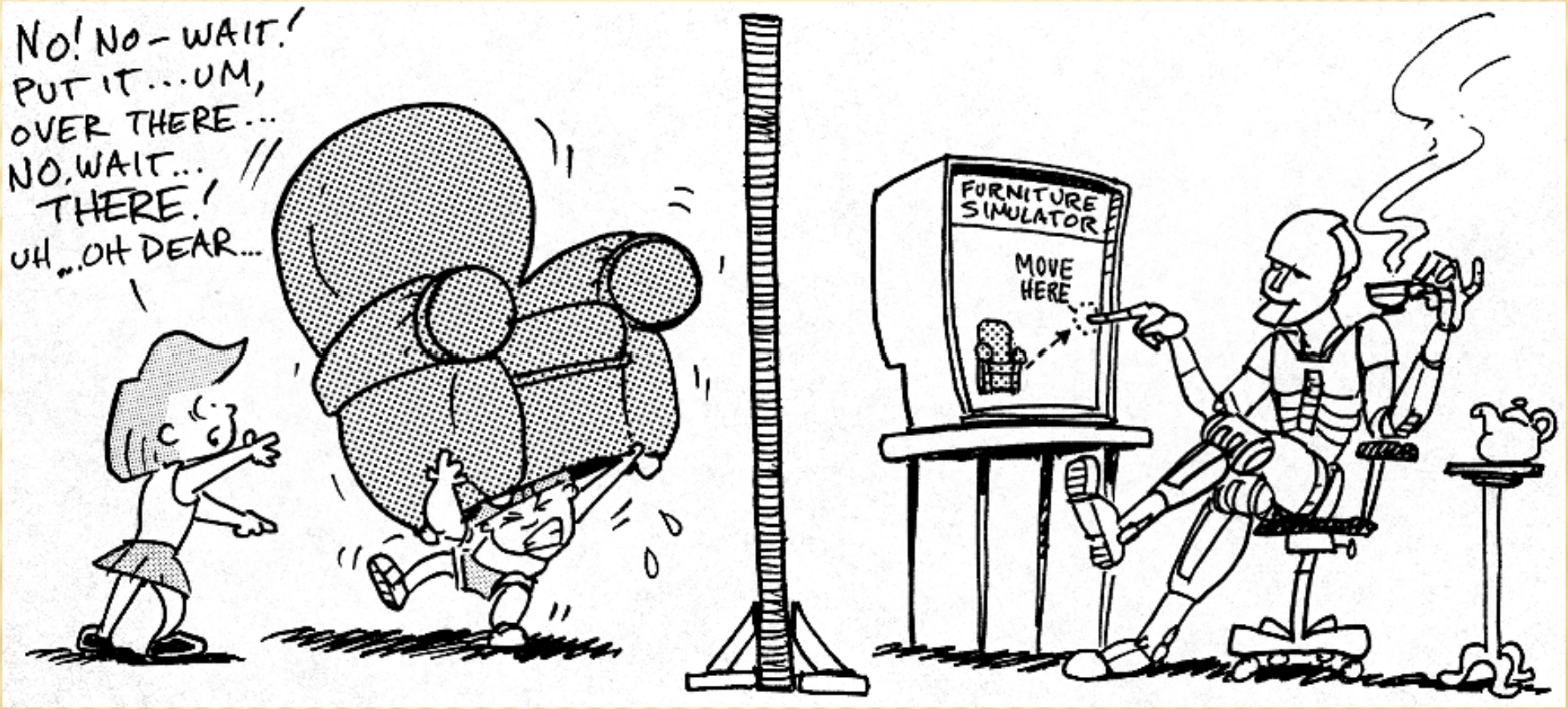
Fig. 12.8 Methods of the `Date` object.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 12.9: DateTime.html -->
6  <!-- Date and Time Methods    -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Date and Time Methods</title>
11
12       <script type = "text/javascript">
13          <!--
14          var current = new Date();
15
16          document.writeln(
17             "<h1>String representations and valueOf</h1>" );
18          document.writeln( "toString: " + current.toString() +
19             "<br />toLocaleString: " + current.toLocaleString() +
20             "<br />toUTCString: " + current.toUTCString() +
21             "<br />valueOf: " + current.valueOf() );
22
23          document.writeln(
24             "<h1>Get methods for local time zone</h1>" );
```

```
25          document.writeln( "getDate: " + current.getDate() +
26             "<br />getDay: " + current.getDay() +
27             "<br />getMonth: " + current.getMonth() +
28             "<br />getFullYear: " + current.getFullYear() +
29             "<br />getTime: " + current.getTime() +
30             "<br />getHours: " + current.getHours() +
31             "<br />getMinutes: " + current.getMinutes() +
32             "<br />getSeconds: " + current.getSeconds() +
33             "<br />getMilliseconds: " +
34             current.getMilliseconds() +
35             "<br />getTimezoneOffset: " +
36             current.getTimezoneOffset() );
37
38          document.writeln(
39             "<h1>Specifying arguments for a new Date</h1>" );
40          var anotherDate = new Date( 2001, 2, 18, 1, 5, 0, 0 );
41          document.writeln( "Date: " + anotherDate );
42
43          document.writeln(
44             "<h1>Set methods for local time zone</h1>" );
45          anotherDate.setDate( 31 );
46          anotherDate.setMonth( 11 );
47          anotherDate.setFullYear( 2001 );
48          anotherDate.setHours( 23 );
49          anotherDate.setMinutes( 59 );
```

```
50          anotherDate.setSeconds( 59 );
51          document.writeln( "Modified date: " + anotherDate );
52          // -->
53       </script>
54
55    </head><body></body>
56 </html>
```

PRACTICA


NO! NO - WAIT.
PUT IT ... UM,
OVER THERE ...
NO, WAIT ...
THERE!
UH .. OH DEAR ...
FURNITURE SIMULATOR
MOVE HERE