JavaScript: Objects

Introduction

- Use JavaScript to manipulate every element of XHTML document from a script
- Reference for several of JavaScript’s built-in objects
- Demonstrates the capabilities

Thinking About Objects

- Objects
  - Attributes
  - Behaviors
  - Encapsulate data and methods
  - Property of information hiding
  - Details hidden within the objects themselves

Math Object

- Allow the programmer to perform many common mathematical calculations

Math Object

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

Math Object

`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

Math object methods.

Math Object

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.

Properties of the Math object.

`String` Object

- JavaScript’s string and character-processing capabilities
- Appropriate for processing names, addresses, credit card information, etc.

Fundamentals of Characters and Strings

- Characters
  - Fundamental building blocks of JavaScript programs
- String
  - Series of characters treated as a single unit

Methods of the `String` Object

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.

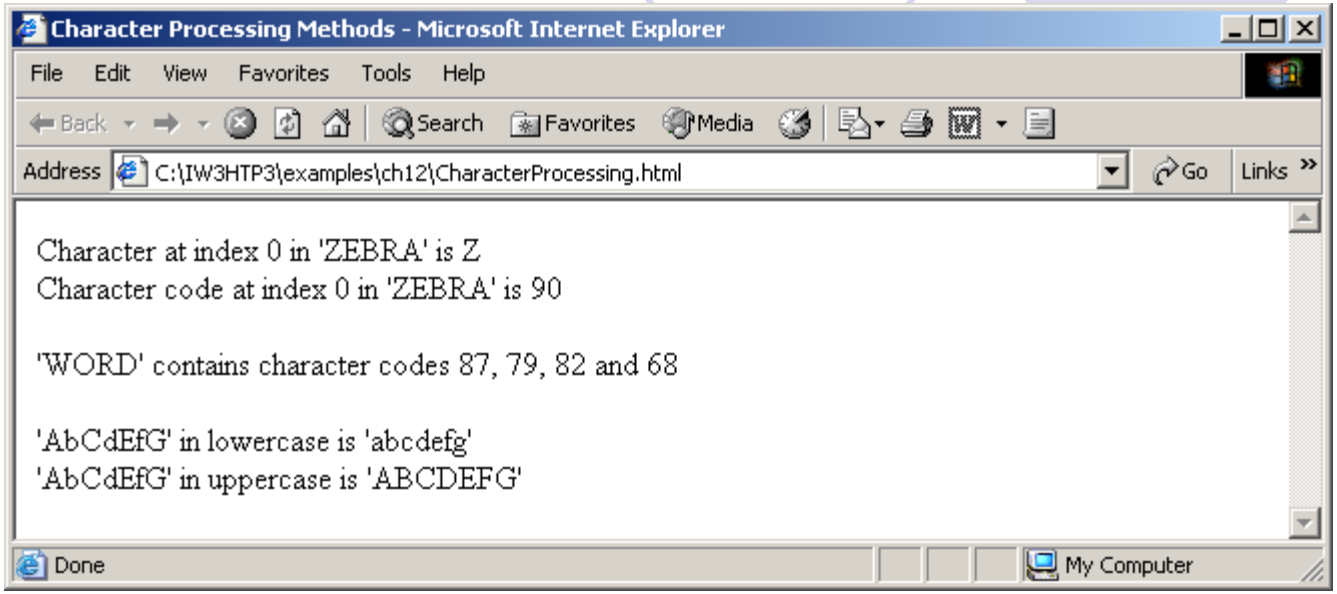
Methods of the `String` Object

`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.

Character Processing Methods

- `charAt`
  - Returns the character at specific position
- `charCodeAt`
  - Returns Unicode value of the character at specific position
- `fromCharCode`
  - Returns string created from series of Unicode values
- `toLowerCase`
  - Returns lowercase version of string
- `toUpperCase`
  - Returns uppercase version of string

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

<!-- Fig. 12.4: CharacterProcessing.html -->
<!-- Character Processing Methods        -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Character Processing Methods</title>

      <script type = "text/javascript">
         <!--
         var s = "ZEBRA";
         var s2 = "AbCdEfG";

         document.writeln( "<p>Character at index 0 in '" +
            s + "' is " + s.charAt( 0 ) );
         document.writeln( "<br />Character code at index 0 in '"
            + s + "' is " + s.charCodeAt( 0 ) + "</p>" );

         document.writeln( "<p>'" +
            String.fromCharCode( 87, 79, 82, 68 ) +
            "' contains character codes 87, 79, 82 and 68</p>" )
         document.writeln( "<p>'" + s2 + "' in lowercase is '" +
            s2.toLowerCase() + "'" );
         document.writeln( "<br />'" + s2 + "' in uppercase is '"
            + s2.toUpperCase() + "'</p>" );
          // -->
      </script>

   </head><body></body>
</html>
```


Character Processing Methods - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
C:\IW3HTP3\examples\ch12\CharacterProcessing.html
Go
Links
Done
My Computer

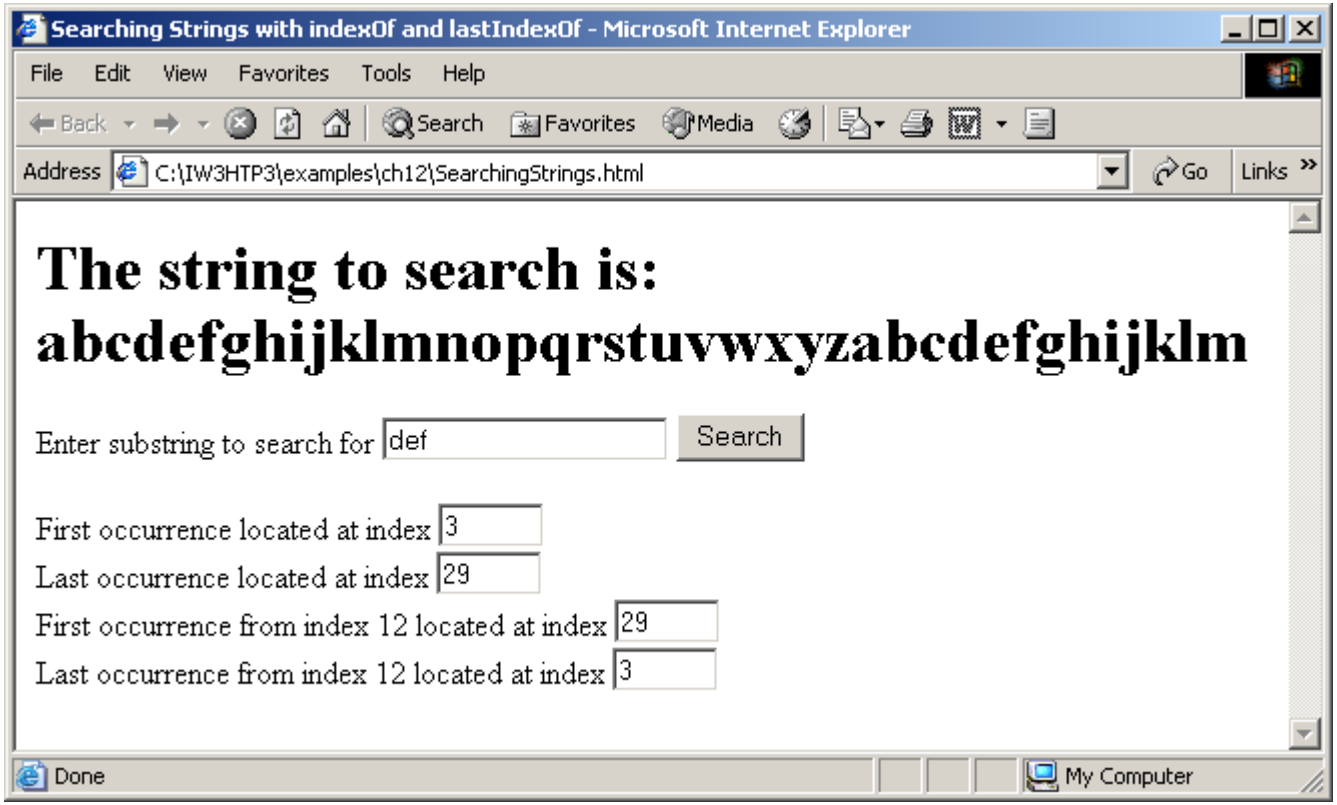
```
Character at index 0 in 'ZEBRA' is Z
Character code at index 0 in 'ZEBRA' is 90

'WORD' contains character codes 87, 79, 82 and 68

'AbCdEfG' in lowercase is 'abcdefg'
'AbCdEfG' in uppercase is 'ABCDEFG'
```

Searching Methods

- `indexOf` and `lastIndexOf`
  - Search for a specified substring in a string

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

<!-- Fig. 12.5: SearchingStrings.html -->
<!-- Searching Strings                -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>
         Searching Strings with indexOf and lastIndexOf
      </title>

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

         function buttonPressed()
         {
            searchForm.first.value =
               letters.indexOf( searchForm.inputVal.value );
            searchForm.last.value =
               letters.lastIndexOf( searchForm.inputVal.value );
            searchForm.first12.value =
               letters.indexOf( searchForm.inputVal.value, 12 );

         searchForm.last12.value =
            letters.lastIndexOf(
               searchForm.inputVal.value, 12 );
      }
      // -->
   </script>

</head>
<body>
   <form name = "searchForm" action = "">
      <h1>The string to search is:<br />
          abcdefghijklmnopqrstuvwxyzabcdefghijklm</h1>
      <p>Enter substring to search for
      <input name = "inputVal" type = "text" />
      <input name = "search" type = "button" value = "Search"
             onclick = "buttonPressed()" /><br /></p>

      <p>First occurrence located at index
      <input name = "first" type = "text" size = "5" />
      <br />Last occurrence located at index
      <input name = "last" type = "text" size = "5" />
      <br />First occurrence from index 12 located at index
      <input name = "first12" type = "text" size = "5" />
      <br />Last occurrence from index 12 located at index
      <input name = "last12" type = "text" size = "5" /></p>
```

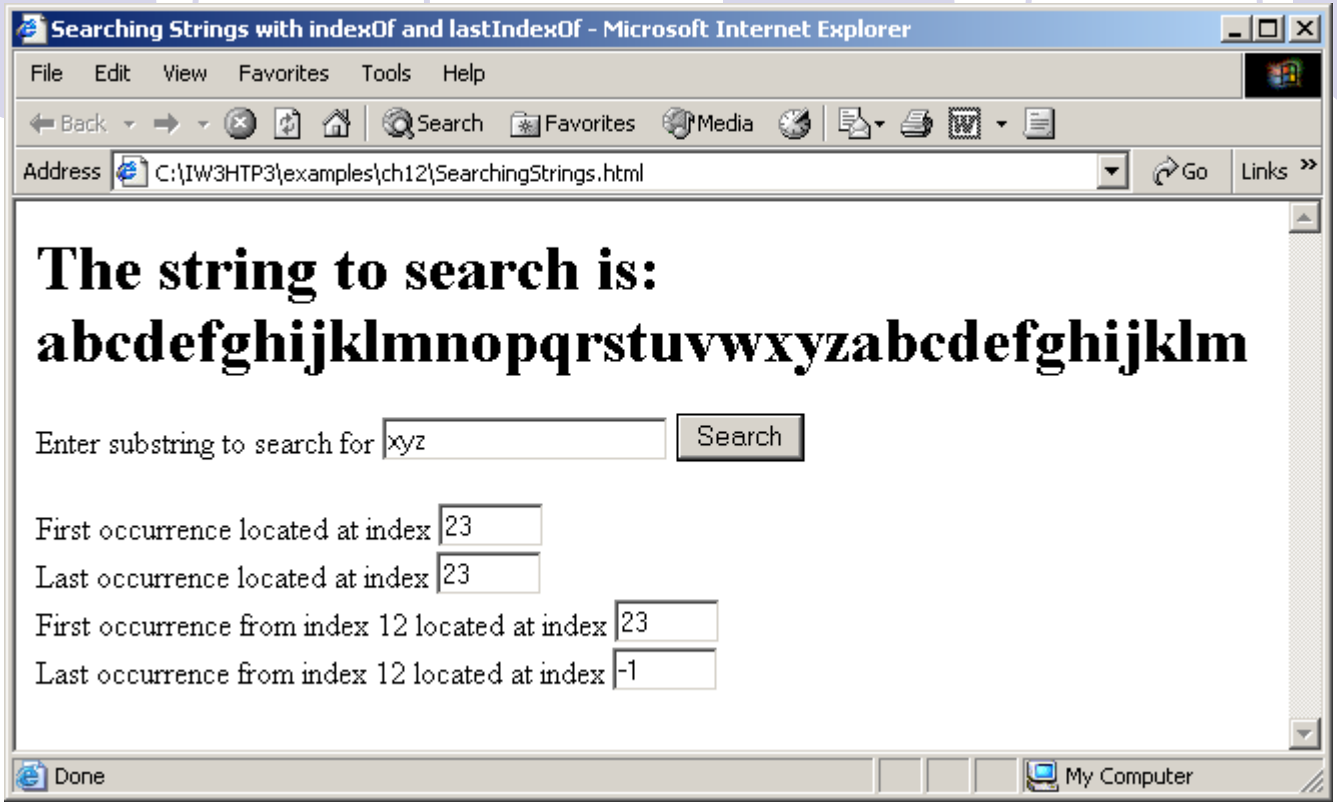
```
        </form>
    </body>
</html>
```


Searching Strings with indexOf and lastIndexOf - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites
Media
Address
C:\IW3HTP3\examples\ch12\SearchingStrings.html
Go
Links
Done
My Computer

The string to search is: abcdefghijklmnopqrstuvwxyzabcdefghijklm

Enter substring to search for def Search

First occurrence located at index 3
Last occurrence located at index 29
First occurrence from index 12 located at index 29
Last occurrence from index 12 located at index 3



Searching Strings with indexOf and lastIndexOf - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch12\SearchingStrings.html
Go Links
Done
My Computer

The string to search is: abcdefghijklmnopqrstuvwxyzabcdefghijklm

Enter substring to search for xyz Search

First occurrence located at index 23

Last occurrence located at index 23

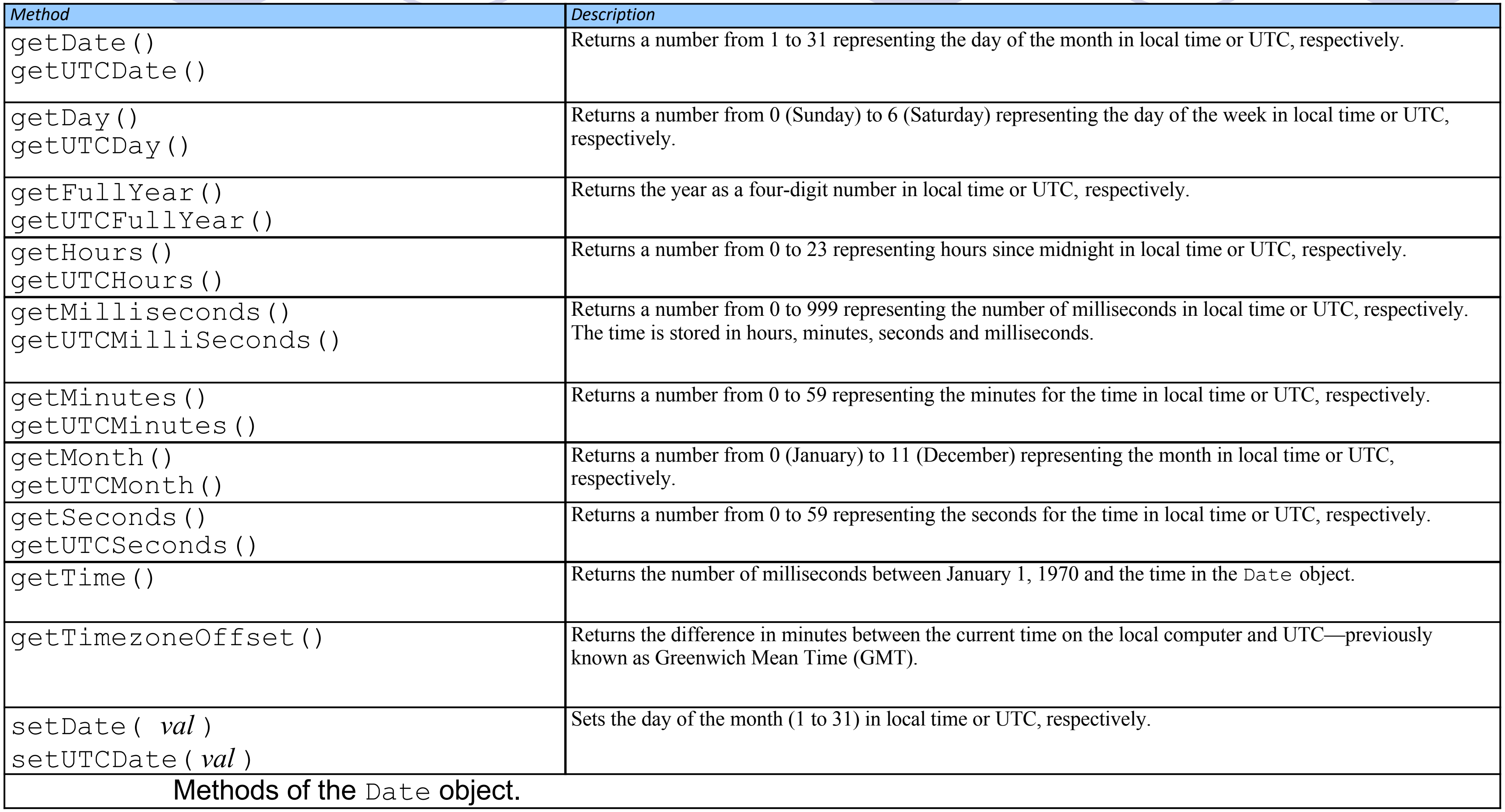
First occurrence from index 12 located at index 23

Last occurrence from index 12 located at index -1

`Date` Object

- Provides methods for date and time manipulations

Date Object



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.

Date Object

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.

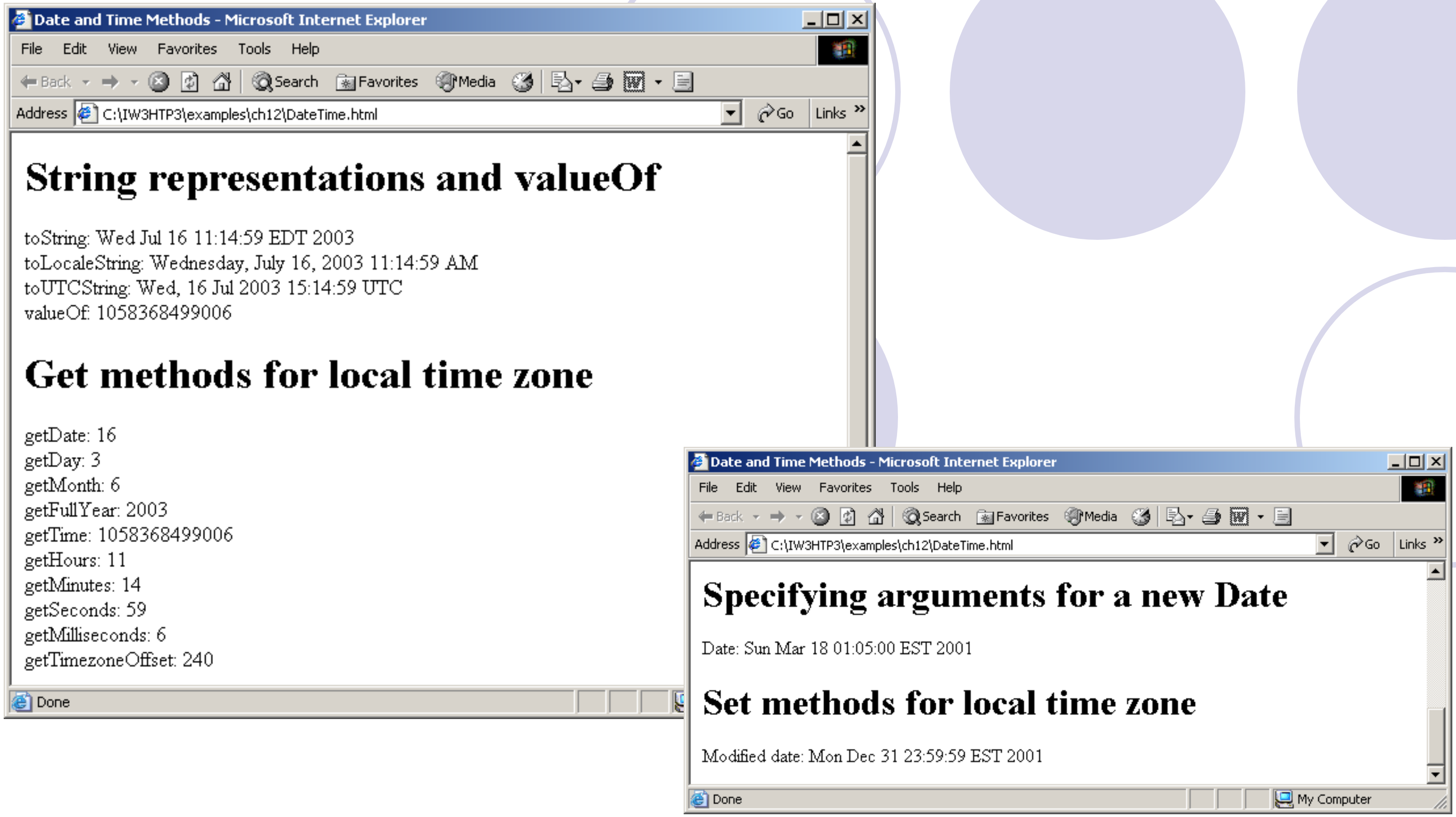
Methods of the `Date` object.

Date Object

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.

Methods of the `Date` object.

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

<!-- Fig. 12.9: DateTime.html -->
<!-- Date and Time Methods    -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Date and Time Methods</title>

      <script type = "text/javascript">
         <!--
         var current = new Date();

         document.writeln(
            "<h1>String representations and valueOf</h1>" );
         document.writeln( "toString: " + current.toString() +
            "<br />toLocaleString: " + current.toLocaleString() +
            "<br />toUTCString: " + current.toUTCString() +
            "<br />valueOf: " + current.valueOf() );

         document.writeln(
            "<h1>Get methods for local time zone</h1>" );
         document.writeln( "getDate: " + current.getDate() +
            "<br />getDay: " + current.getDay() +
            "<br />getMonth: " + current.getMonth() +
            "<br />getFullYear: " + current.getFullYear() +
            "<br />getTime: " + current.getTime() +
            "<br />getHours: " + current.getHours() +
            "<br />getMinutes: " + current.getMinutes() +
            "<br />getSeconds: " + current.getSeconds() +
            "<br />getMilliseconds: " +
            current.getMilliseconds() +
            "<br />getTimezoneOffset: " +
            current.getTimezoneOffset() );

         document.writeln(
            "<h1>Specifying arguments for a new Date</h1>" );
         var anotherDate = new Date( 2001, 2, 18, 1, 5, 0, 0 );
         document.writeln( "Date: " + anotherDate );

         document.writeln(
            "<h1>Set methods for local time zone</h1>" );
         anotherDate.setDate( 31 );
         anotherDate.setMonth( 11 );
         anotherDate.setFullYear( 2001 );
         anotherDate.setHours( 23 );
         anotherDate.setMinutes( 59 );
         anotherDate.setSeconds( 59 );
         document.writeln( "Modified date: " + anotherDate );
         // -->
      </script>

   </head><body></body>
</html>
```



Date and Time Methods - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch12\DateTime.html
Go Links
Done
My Computer

String representations and valueOf

```
toString: Wed Jul 16 11:14:59 EDT 2003
toLocaleString: Wednesday, July 16, 2003 11:14:59 AM
toUTCString: Wed, 16 Jul 2003 15:14:59 UTC
valueOf: 1058368499006
```

Get methods for local time zone

```
getDate: 16
getDay: 3
getMonth: 6
getFullYear: 2003
getTime: 1058368499006
getHours: 11
getMinutes: 14
getSeconds: 59
getMilliseconds: 6
getTimezoneOffset: 240
```

Specifying arguments for a new Date

```
Date: Sun Mar 18 01:05:00 EST 2001
```

Set methods for local time zone

```
Modified date: Mon Dec 31 23:59:59 EST 2001
```

Boolean and Number Objects

- Object wrappers for boolean `true`/`false` values and numbers

Boolean and Number Objects

Method	Description
`toString()`	Returns the string “true” if the value of the `Boolean` object is true; otherwise, returns the string “false.”
`valueOf()`	Returns the value `true` if the `Boolean` object is `true`; otherwise, returns `false`.

Boolean object methods.

Boolean and Number Objects

Method or Property	Description
`toString( radix )`	Returns the string representation of the number. The optional *radix* argument (a number from 2 to 36) specifies the number's base. For example, radix 2 results in the binary representation of the number, 8 results in the octal representation, 10 results in the decimal representation and 16 results in the hexadecimal representation. See Appendix E, Number Systems for a review of the binary, octal, decimal and hexadecimal number systems.
`valueOf()`	Returns the numeric value.
`Number.MAX_VALUE`	This property represents the largest value that can be stored in a JavaScript program—approximately 1.79E+308
`Number.MIN_VALUE`	This property represents the smallest value that can be stored in a JavaScript program—approximately 2.22E–308
`Number.NaN`	This property represents *not a number*—a value returned from an arithmetic expression that does not result in a number (e.g., the expression `parseInt( "hello" )` cannot convert the string `"hello"` into a number, so `parseInt` would return `Number.NaN`. To determine whether a value is `NaN`, test the result with function `isNaN`, which returns true if the value is `NaN`; otherwise, it returns false.
`Number.NEGATIVE_INFINITY`	This property represents a value less than `-Number.MAX_VALUE`.
`Number.POSITIVE_INFINITY`	This property represents a value greater than `Number.MAX_VALUE`.

`document` Object

- Manipulate document that is currently visible in the browser window

`document` Object

Method or Property	Description
`write(` *string* `)`	Writes the string to the XHTML document as XHTML code.
`writeln(` *string* `)`	Writes the string to the XHTML document as XHTML code and adds a newline character at the end.
`document.cookie`	This property is a string containing the values of all the cookies stored on the user’s computer for the current document. See Section 12.9, Using Cookies.
`document.lastModified`	This property is the date and time that this document was last modified.

Important `document` object methods and properties.

`window` Object

Method or Property	Description
`open(` *url*`,` *name*`,` *options* `)`	Creates a new window with the URL of the window set to *url*, the name set to *name*, and the visible features set by the string passed in as *option*.
`prompt(` *prompt*`,` *default* `)`	Displays a dialog box asking the user for input. The text of the dialog is *prompt*, and the default value is set to *default*.
`close()`	Closes the current window and deletes its object from memory.
`window.focus()`	This method gives focus to the window (i.e., puts the window in the foreground, on top of any other open browser windows).
`window.document`	This property contains the `document` object representing the document currently inside the window.
`window.closed`	This property contains a boolean value that is set to true if the window is closed, and false if it is not.
`window.opener`	This property contains the `window` object of the window that opened the current window, if such a window exists.

Important window object methods and properties.

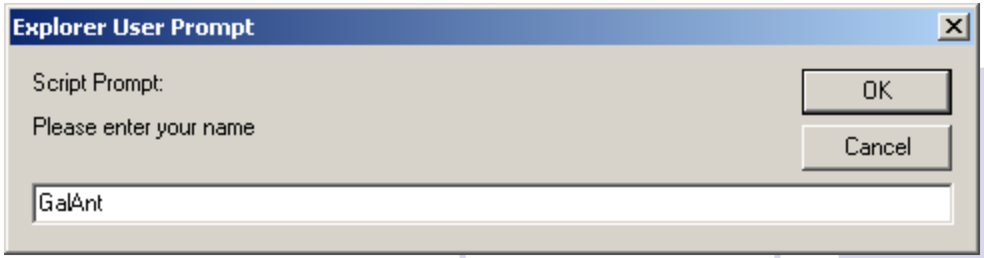
Using Cookies

- Cookie
  - Data stored on user's computer to maintain information about client during and between browser sessions
  - Can be accessed through `cookie` property
  - Set expiration date through `expires` property
  - Use `escape` function to convert non-alphanumeric characters to hexadecimal escape sequences
  - `unescape` function converts hexadecimal escape sequences back to English characters

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- Fig. 12.15: cookie.html -->
<!-- Using Cookies           -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Using Cookies</title>
      <script type = "text/javascript">
         <!--
         var now = new Date(); // current date and time
         var hour = now.getHours(); // current hour (0-23)
         var name;

         if ( hour < 12 ) // determine whether it is morning
            document.write( "<h1>Good Morning, " );
         else
         {
            hour = hour - 12; // convert from 24 hour clock to PM time


      // determine whether it is afternoon or evening
      if ( hour < 6 )
         document.write( "<h1>Good Afternoon, " );
      else
         document.write( "<h1>Good Evening, " );
   }

   // determine whether there is a cookie
   if ( document.cookie )
   {
      // convert escape characters in the cookie string to their
      // english notation
      var myCookie = unescape( document.cookie );

      // split the cookie into tokens using = as delimiter
      var cookieTokens = myCookie.split( "=" );

      // set name to the part of the cookie that follows the = sign
      name = cookieTokens[ 1 ];
   }
   else
   {
      // if there was no cookie then ask the user to input a name
      name = window.prompt( "Please enter your name", "GalAnt" );
```

```
         // escape special characters in the name string
         // and add name to the cookie
         document.cookie = "name=" + escape( name );
      }

      document.writeln(
         name + ", welcome to JavaScript programming! </h1>" );
      document.writeln( "<a href= \" JavaScript:wrongPerson() \" > " +
         "Click here if you are not " + name + "</a>" );

      // reset the document's cookie if wrong person
      function wrongPerson()
      {
         // reset the cookie
         document.cookie= "name=null;" +
            " expires=Thu, 01-Jan-95 00:00:01 GMT";

         // after removing the cookie reload the page to get a new name
         location.reload();
      }

      // -->
   </script>
</head>

```

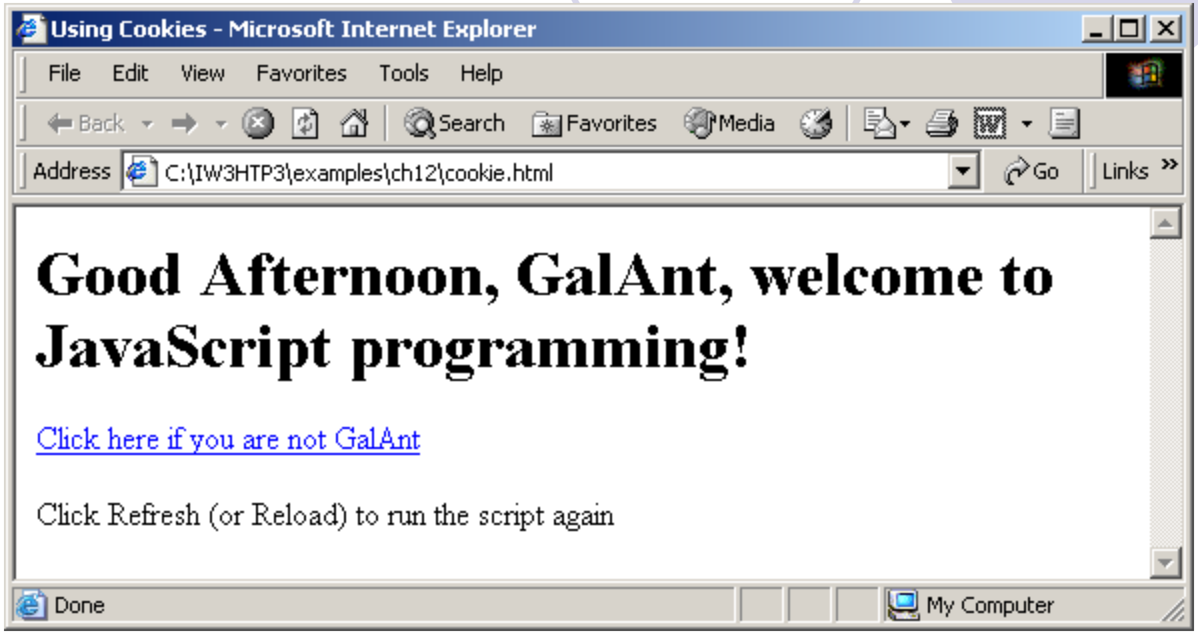
```
   <body>
      <p>Click Refresh (or Reload) to run the script again</p>
   </body>
</html>
```



Script Prompt:

Please enter your name

GalAnt



C:\IW3HTP3\examples\ch12\cookie.html

Good Afternoon, GalAnt, welcome to JavaScript programming!

Click here if you are not GalAnt

Click Refresh (or Reload) to run the script again