IPA

1st Semester, 2007-2008

Internet 1

Ch. 13
JavaScript: Objects

attasr@ipa.edu.sa

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

Fig. 12.1 `Math` object methods.

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

Fig. 12.1 `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.

Fig. 12.2 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

12.4.2 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.

Fig. 12.3 `String` object methods.

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.

Fig. 12.3 `String` object methods.

Methods of the `String` Object

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.

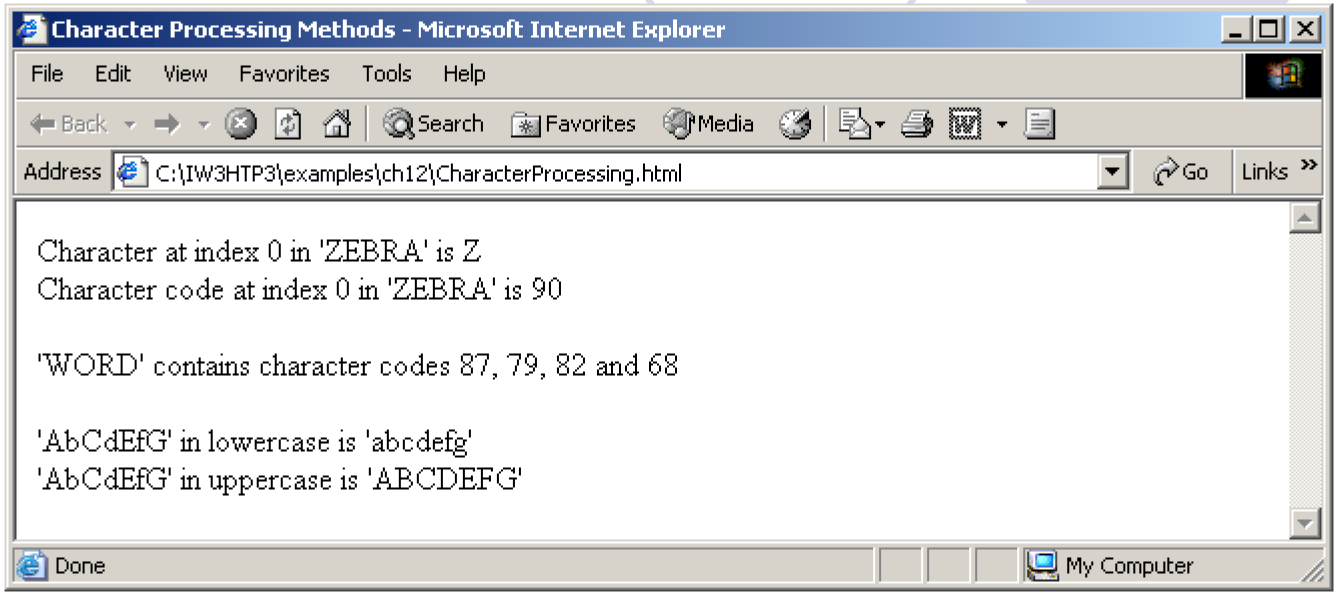
Fig. 12.3 `String` object methods.

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

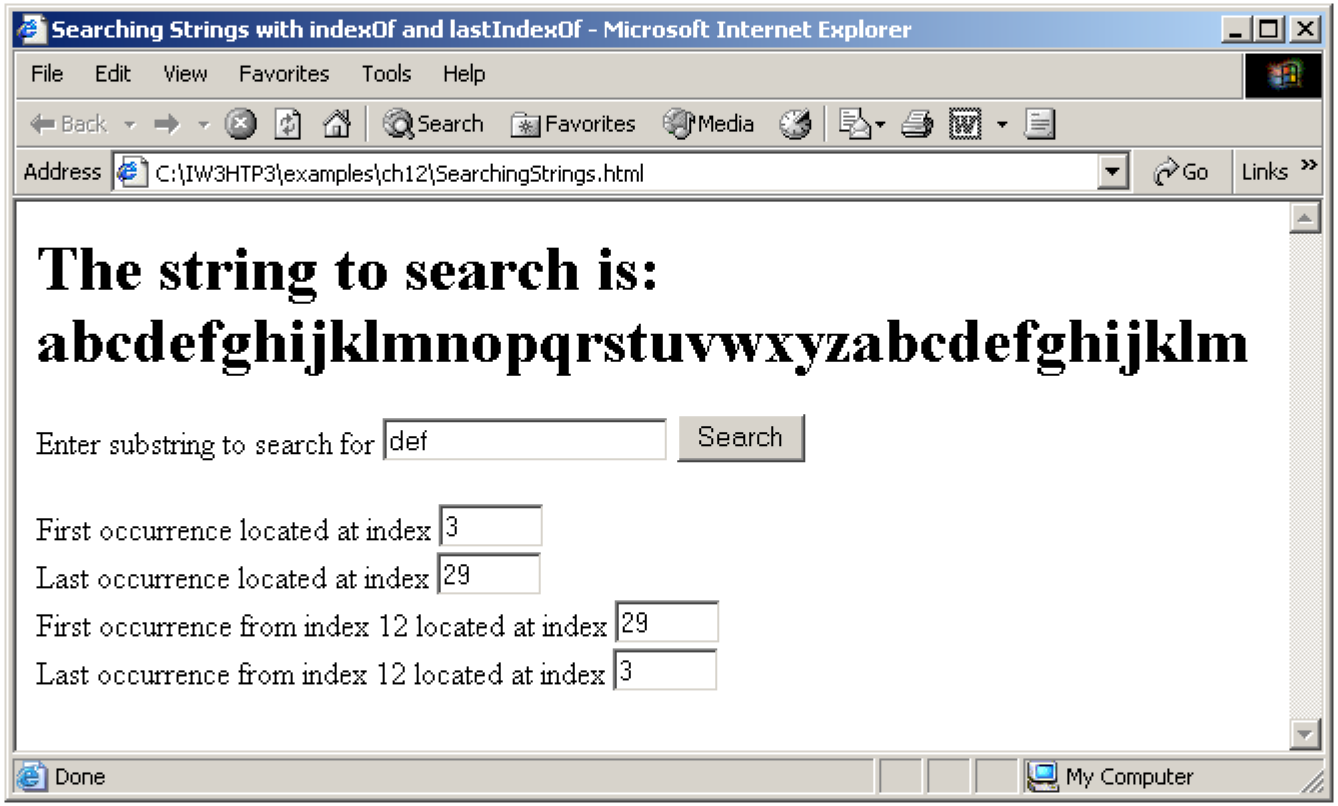
```
Character at index 0 in 'ZEBRA' is Z
Character code at index 0 in 'ZEBRA' is 90

'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>
```

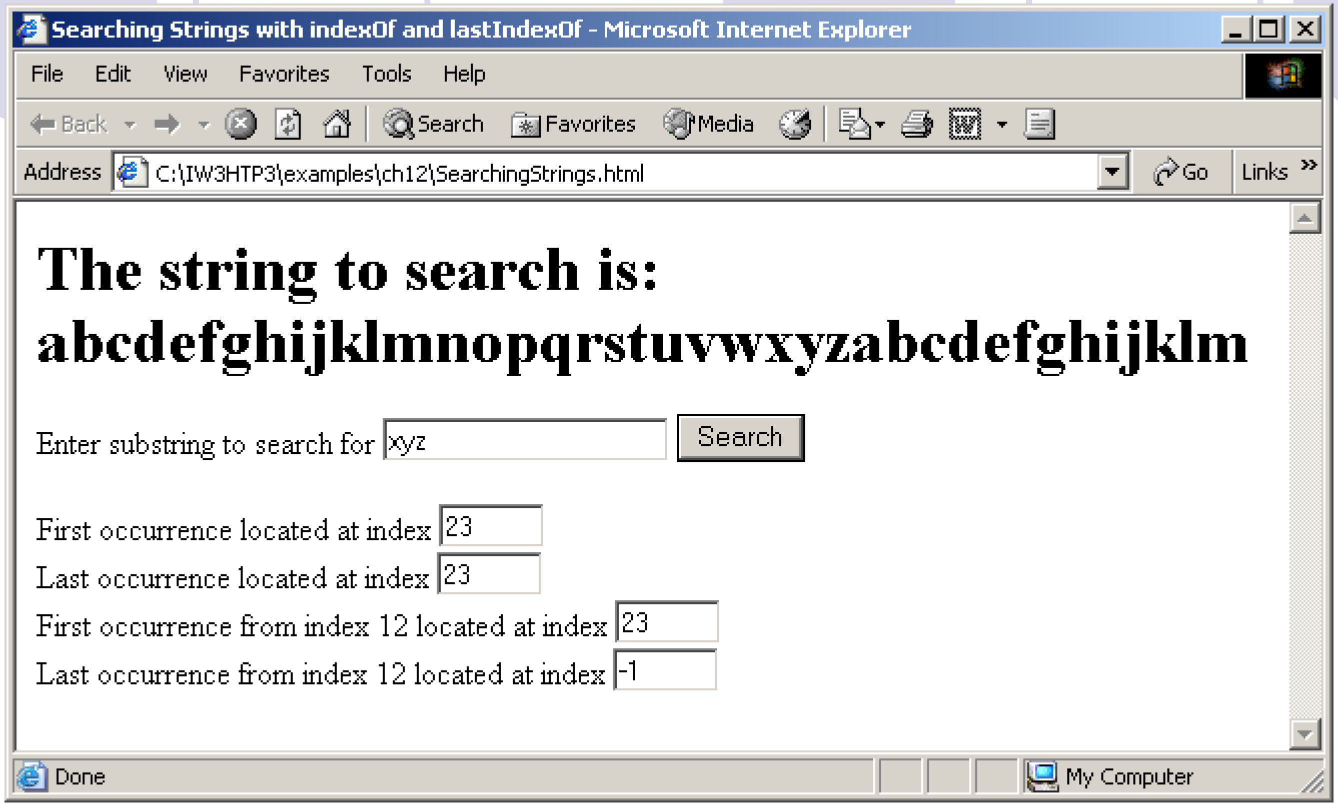
```
        </form>
    </body>
</html>
```


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
Address C:\IW3HTP3\examples\ch12\SearchingStrings.html
Done
My Computer

The string to search is: abcdefghijklmnopqrstuvwxyzabcdefghijklm

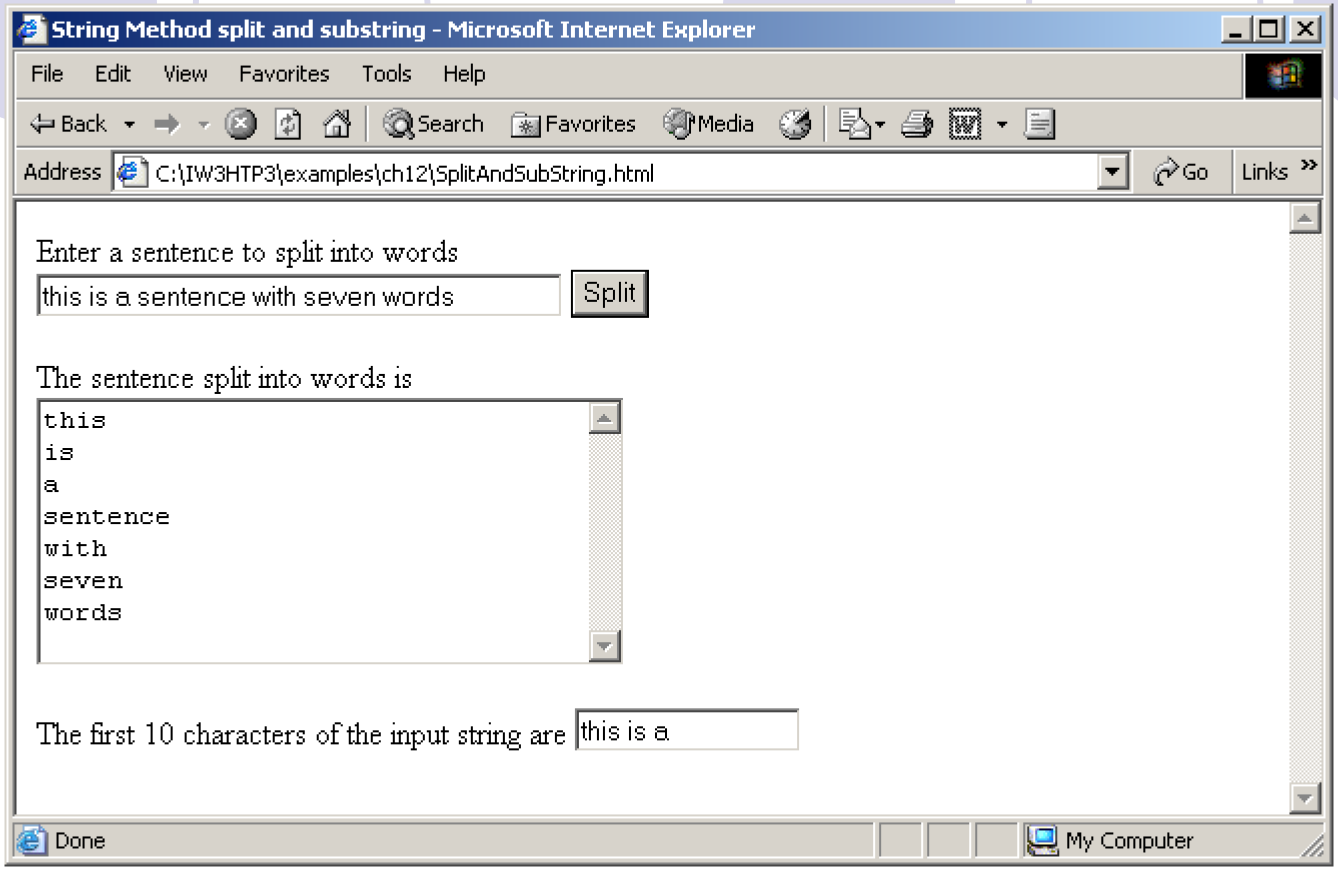
Enter substring to search for xyz Search

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

Splitting Strings and Obtaining Substrings

- Tokenization
  - The process of breaking a string into tokens
- Tokens
  - Individual words
  - Separated by delimiters

```
<?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. 12.6: SplitAndSubString.html -->
<!-- String Method split and substring -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>String Method split and substring</title>

      <script type = "text/javascript">
         <!--
         function splitButtonPressed()
         {
            var strings = myForm.inputVal.value.split( " " );
            myForm.output.value = strings.join( "\n" );

            myForm.outputSubstring.value =
               myForm.inputVal.value.substring( 0, 10 );
         }
         // -->
      </script>
    </head>

   <body>
      <form name = "myForm" action = "">
         <p>Enter a sentence to split into words<br />
         <input name = "inputVal" type = "text" size = "40" />
         <input name = "splitButton" type = "button" value =
               "Split" onclick = "splitButtonPressed()" /></p>

         <p>The sentence split into words is<br />
         <textarea name = "output" rows = "8" cols = "34">
         </textarea></p>

         <p>The first 10 characters of the input string are
         <input name = "outputSubstring" type = "text"
               size = "15" /></p>
      </form>
   </body>
</html>
```


String Method split and substring - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch12\SplitAndSubString.html
Go Links
Split
Done
My Computer

Enter a sentence to split into words

this is a sentence with seven words

The sentence split into words is

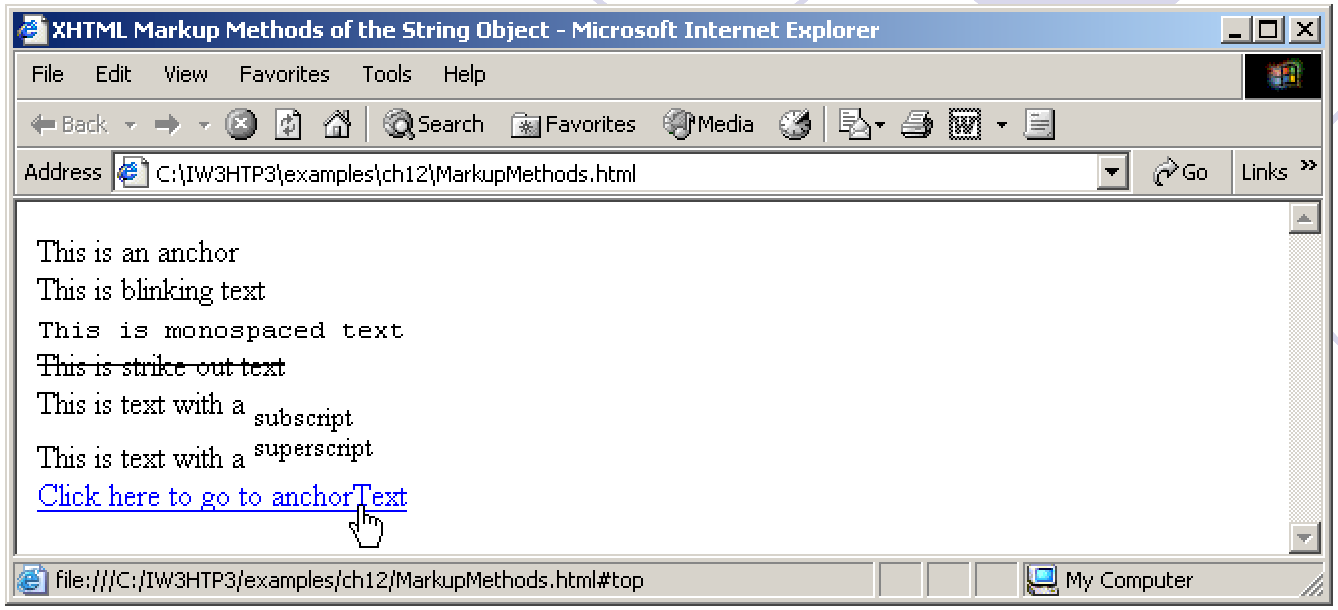
```
this
is
a
sentence
with
seven
words
```

The first 10 characters of the input string are this is a

XHTML Markup Methods

- 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>`

```
<?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. 12.7: MarkupMethods.html             -->
<!-- XHTML markup methods of the String object -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>XHTML Markup Methods of the String Object</title>

      <script type = "text/javascript">
         <!--
         var anchorText = "This is an anchor",
             blinkText = "This is blinking text",
             fixedText = "This is monospaced text",
             linkText = "Click here to go to anchorText",
             strikeText = "This is strike out text",
             subText = "subscript",
             supText = "superscript";

         document.writeln( anchorText.anchor( "top" ) );
         document.writeln( "<br />" + blinkText.blink() );
         document.writeln( "<br />" + fixedText.fixed() );
         document.writeln( "<br />" + strikeText.strike() );
         document.writeln(
            "<br />This is text with a " + subText.sub() );
         document.writeln(
            "<br />This is text with a " + supText.sup() );
         document.writeln(
            "<br />" + linkText.link( "#top" ) );
         // -->
      </script>

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


XHTML Markup Methods of the String Object - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch12\MarkupMethods.html
Go Links
file:///C:/IW3HTP3/examples/ch12/MarkupMethods.html#top
My Computer

This is an anchor
This is blinking text
This is monospaced text
~~This is strike-out text~~
This is text with a $_{\text{subscript}}$
This is text with a $^{\text{superscript}}$
Click here to go to anchorText

`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.

Fig. 12.8 Methods of the `Date` object.

12.5 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.

Fig. 12.8 Methods of the `Date` object.

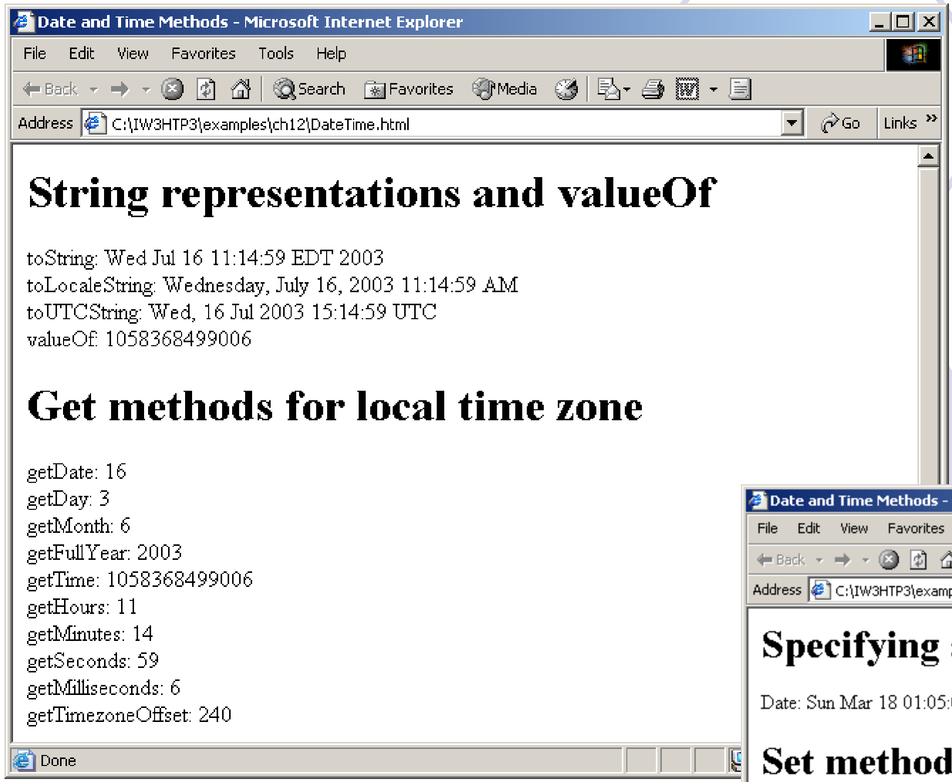
Date Object

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.

Fig. 12.8 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>
```

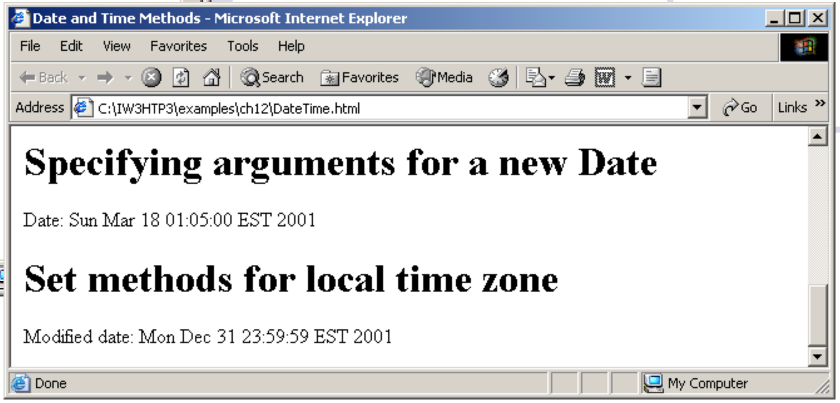


String representations and valueOf

```
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`.

Fig. 12.10 **`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`.

Fig. 12.11 `Number` object methods and properties.

`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.

Fig. 12.12 Important `document` object methods and properties.

`window` Object

- Provides methods for manipulating browser window

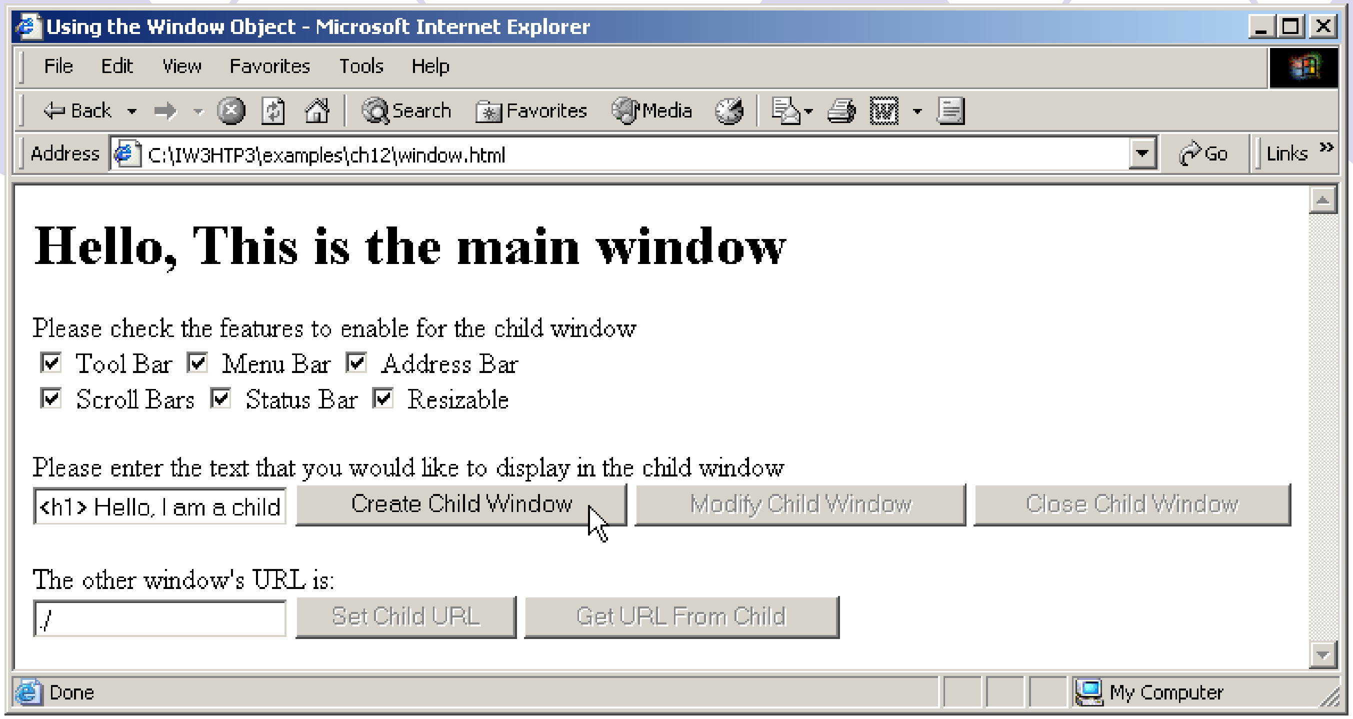
```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- Fig. 12.13: window.html  -->
<!-- Using the Window Object  -->

<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
 <title>Using the Window Object</title>

 <script type = "text/javascript">
    <!--
    var childWindow; // variable to control the child window

    function createChildWindow()
    {
       // these variables all contain either "yes" or "no"
       // to enable or disable a feature in the child window
       var toolBar // specify if toolbar will appear in child window
       var menuBar; // specify if menubar will appear in child window
       var location; // specify if address bar will appear in child window
       var scrollBars; // specify if scrollbars will appear in child window
       var status; // specify if status bar will appear in child window
       var resizable; // specify if the child window will be resizable
```

```
      // determine whether the Tool Bar checkbox is checked
      if ( toolBarCheckBox.checked )
         toolBar = "yes";
      else
         toolBar = "no";

      // determine whether the Menu Bar checkbox is checked
      if ( menuBarCheckBox.checked )
         menuBar = "yes";
      else
         menuBar = "no";

      // determine whether the Address Bar checkbox is checked
      if ( locationCheckBox.checked )
         location = "yes";
      else
         location = "no";

      // determine whether the Scroll Bar checkbox is checked
      if ( scrollBarsCheckBox.checked )
         scrollBars = "yes";
      else
         scrollBars = "no";

      // determine whether the Status Bar checkbox is checked
      if ( statusCheckBox.checked )
         status = "yes";
      else
         status = "no";

      // determine whether the Resizable checkbox is checked
      if ( resizableCheckBox.checked )
         resizable = "yes";
      else
         resizable = "no";

      // display window with selected features
      childWindow = window.open( "", "", "resizable = " + resizable +
         ", toolbar = " + toolBar + ", menubar = " + menuBar +
         ", status = " + status + ", location = " + location +
         ", scrollbars = " + scrollBars );

      // disable buttons
      closeButton.disabled = false;
      modifyButton.disabled = false;
      getURLButton.disabled = false;
      setURLButton.disabled = false;
   } // end function createChildWindow

```

```
// insert text from the textbox into the child window
function modifyChildWindow()
{
   if ( childWindow.closed )
      alert( "You attempted to interact with a closed window" );
   else
      childWindow.document.write( textForChild.value );
} // end function modifyChildWindow

// close the child window
function closeChildWindow()
{
   if ( childWindow.closed )
      alert( "You attempted to interact with a closed window" );
   else
      childWindow.close();

   closeButton.disabled = true;
   modifyButton.disabled = true;
   getURLButton.disabled = true;
   setURLButton.disabled = true;
} // end function closeChildWindow

```

```
   // copy the URL of the child window into the parent window’s myChildURL
    function getChildWindowURL()
    {
       if ( childWindow.closed )
          alert( "You attempted to interact with a closed window" );
       else
          myChildURL.value = childWindow.location;
    } // end function getChildWindowURL

    // set the URL of the child window to the URL
    // in the parent window’s myChildURL
    function setChildWindowURL()
    {
       if ( childWindow.closed )
          alert( "You attempted to interact with a closed window" );
       else
          childWindow.location = myChildURL.value;
    } // end function setChildWindowURL
    //-->
</script>

</head>

<body>
<h1>Hello, This is the main window</h1>
```

```
<p>Please check the features to enable for the child window<br/>
   <input id = "toolBarCheckBox" type = "checkbox" value = ""
      checked = "checked" />
      <label>Tool Bar</label>
   <input id = "menuBarCheckBox" type = "checkbox" value = ""
      checked = "checked" />
      <label>Menu Bar</label>
   <input id = "locationCheckBox" type = "checkbox" value = ""
      checked = "checked" />
      <label>Address Bar</label><br/>
   <input id = "scrollBarsCheckBox" type = "checkbox" value = ""
      checked = "checked" />
      <label>Scroll Bars</label>
   <input id = "statusCheckBox" type = "checkbox" value = ""
      checked = "checked" />
      <label>Status Bar</label>
   <input id = "resizableCheckBox" type = "checkbox" value = ""
      checked = "checked" />
      <label>Resizable</label><br/></p>

<p>Please enter the text that you would like to display
   in the child window<br/>
   <input id = "textForChild" type = "text"
      value = "<h1> Hello, I am a child window</h1> <br\>"/>
   <input id = "createButton" type = "button"
      value = "Create Child Window" onclick = "createChildWindow()" />
   <input id= "modifyButton" type = "button" value = "Modify Child Window"
      onclick = "modifyChildWindow()" disabled = "disabled"/>
   <input id = "closeButton" type = "button" value = "Close Child Window"
      onclick = "closeChildWindow()" disabled = "disabled"/></p>

<p>The other window's URL is: <br/>
   <input id = "myChildURL" type = "text" value = "./"/>
   <input id = "setURLButton" type = "button" value = "Set Child URL"
      onclick = "setChildWindowURL()" disabled = "disabled"/>
   <input id = "getURLButton" type = "button" value = "Get URL From Child"
      onclick = "getChildWindowURL()" disabled = "disabled"/></p>

</body>
</html>
```


Using the Window Object - Microsoft Internet Explorer

Address C:\IW3HTP3\examples\ch12\window.html

Hello, This is the main window

Please check the features to enable for the child window

☑ Tool Bar ☑ Menu Bar ☑ Address Bar

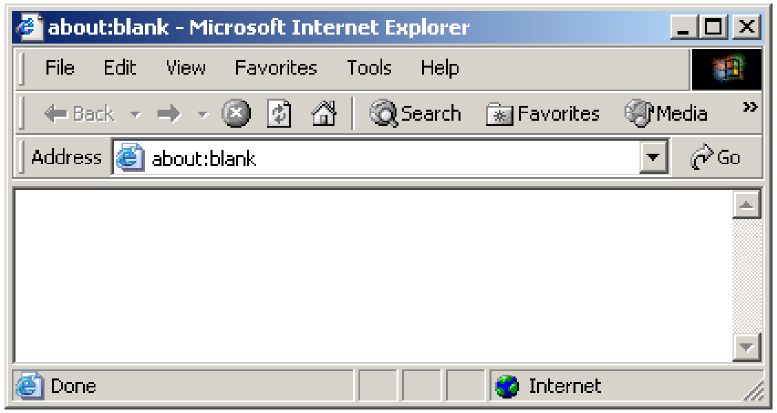
☑ Scroll Bars ☑ Status Bar ☑ Resizable

Please enter the text that you would like to display in the child window

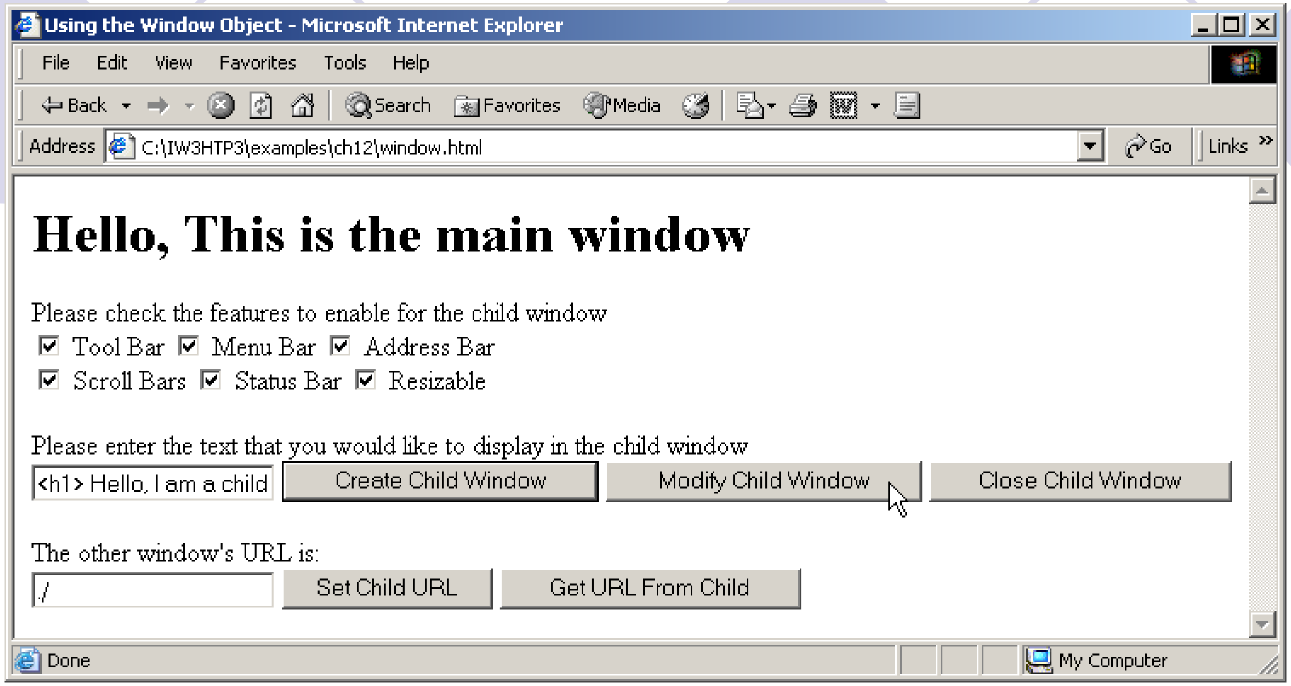
<h1> Hello, I am a child

The other window's URL is:

./


about:blank - Microsoft Internet Explorer

Address about:blank


Using the Window Object - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch12\window.html
Go Links
Done
My Computer

Hello, This is the main window

Please check the features to enable for the child window

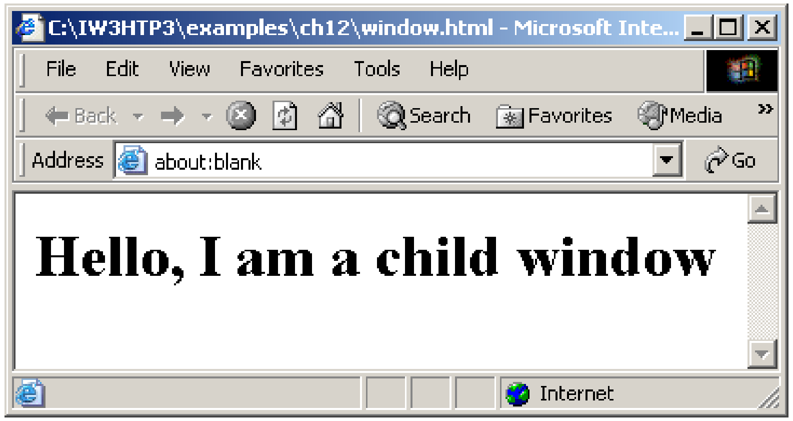
[✓] Tool Bar [✓] Menu Bar [✓] Address Bar
[✓] Scroll Bars [✓] Status Bar [✓] Resizable

Please enter the text that you would like to display in the child window

<h1> Hello, I am a child | Create Child Window | Modify Child Window | Close Child Window

The other window's URL is:

./ | Set Child URL | Get URL From Child


C:\IW3HTP3\examples\ch12\window.html - Microsoft Inte...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address about:blank
Go
Internet

Hello, I am a child window

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.

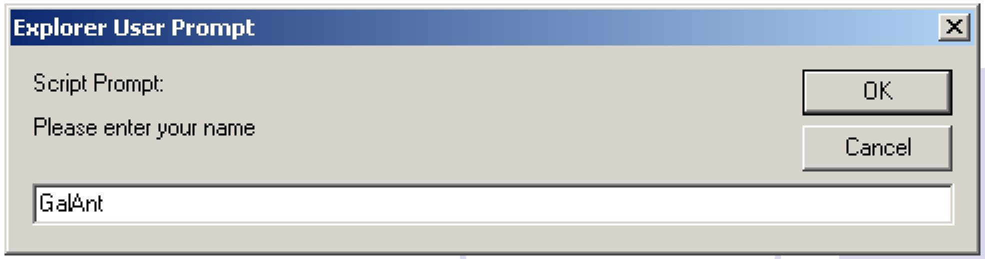
Fig. 12.14 Important `window` object methods and properties.

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>
```

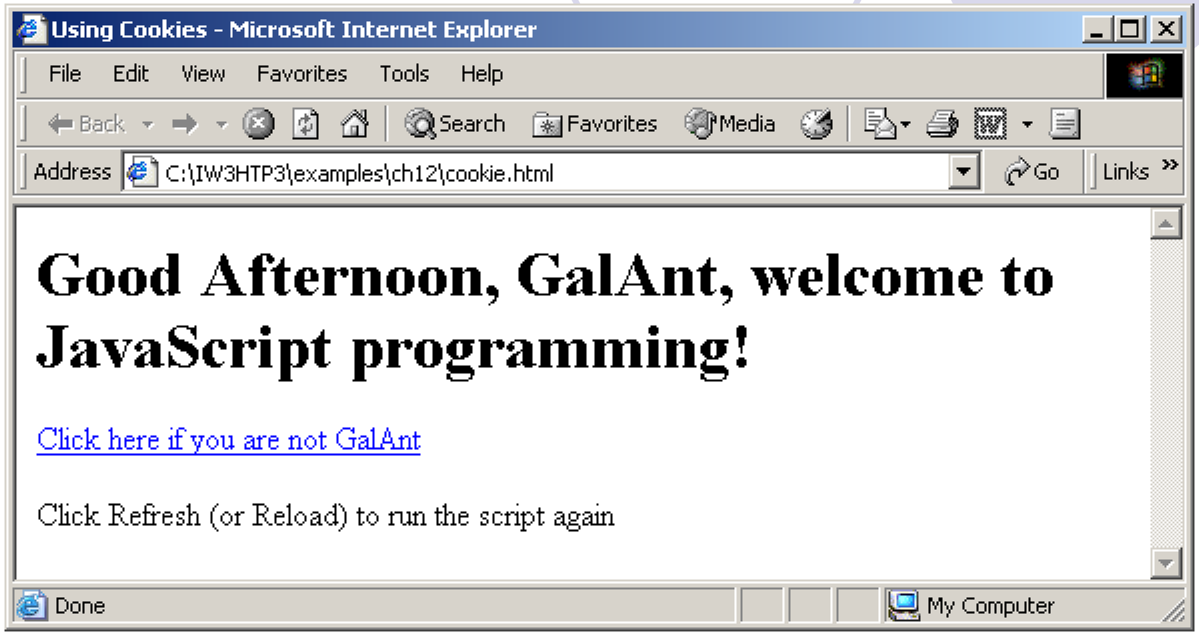


Explorer User Prompt

Script Prompt:

Please enter your name

GalAnt



Using Cookies - Microsoft Internet Explorer

Address 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

Final JavaScript Example

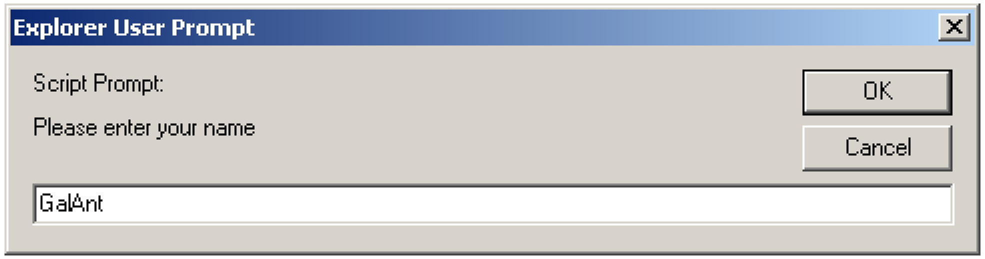
- Combines concepts discussed previously

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

<!-- Fig. 12.16: final.html  -->
<!-- Putting It All Together -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Putting It All Together</title>

      <script type = "text/javascript">
         <!--
         var now = new Date(); // current date and time
         var hour = now.getHours(); // current hour

         // array with names of the images that will be randomly selected
         var pictures =
            [ "CPE", "EPT", "GPP", "GUI", "PERF", "PORT", "SEO" ];

         // array with the quotes that will be randomly selected
         var quotes = [ "Form ever follows function.<br/>" +
            " Louis Henri Sullivan", "E pluribus unum." +
            " (One composed of many.) <br/> Virgil", "Is it a" +
            " world to hide virtues in?<br/> William Shakespeare" ];

         // write the current date and time to the web page
         document.write( "<p>" + now.toLocaleString() + "<br/></p>" );

         // determine whether it is morning
         if ( hour < 12 )
            document.write( "<h2>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( "<h2>Good Afternoon, " );
            else
               document.write( "<h2>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 );
         }

         // write the greeting to the page
         document.writeln(
            name + ", welcome to JavaScript programming!</h2>" );

         // write the link for deleting the cookie to the page
         document.writeln( "<a href = \" JavaScript:wrongPerson() \" > " +
            "Click here if you are not " + name + "</a><br/>" );

```

```
      // write the random image to the page
      document.write ( "<img src = \"" +
         pictures[ Math.floor( Math.random() * 7 ) ] +
         ".gif\" width= \" 105 \" height= \" 100 \" /> <br/>" );

      // write the random quote to the page
      document.write ( quotes[ Math.floor( Math.random() * 3 ) ] );

      // create a window with all the quotes in it
      function allQuotes()
      {
         // create the child window for the quotes
         quoteWindow = window.open( "", "", "resizable=yes, toolbar" +
            "=no, menubar=no, status=no, location=no," +
            " scrollBars=yes" );
         quoteWindow.document.write( "<p>" )

         // loop through all quotes and write them in the new window
         for ( var i = 0; i < quotes.length; i++ )
            quoteWindow.document.write( ( i + 1 ) + ".) " +
               quotes[ i ] + "<br/><br/>");

         // write a close link to the new window
         quoteWindow.document.write( "</p><br/><a href = \" " +
            "JavaScript:window.close()\">" +
            " Close this window </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();
      }

      // open a new window with the quiz2.html file in it
      function openQuiz()
      {
         window.open( "quiz2.html", "", "resizable = yes, " +
            "toolbar = no, menubar = no, status = no, " +
            "location = no, scrollBars = no");
      }
   // -->
```

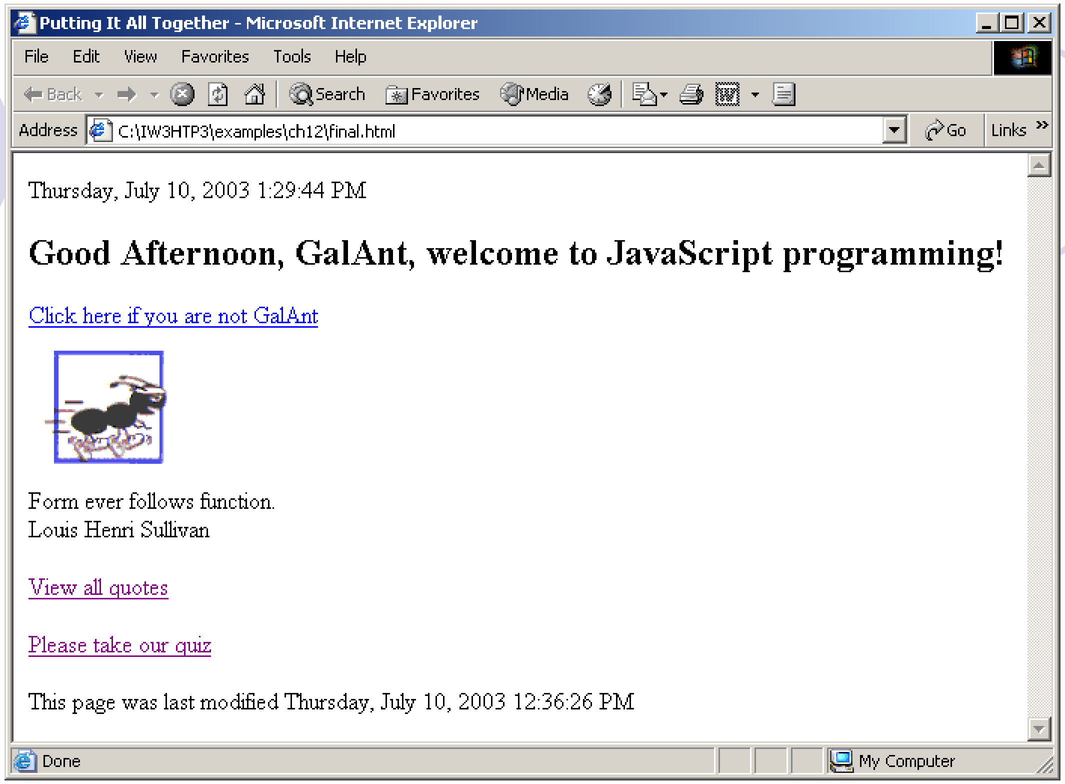
```
      </script>

   </head>

   <body>
      <p><a href = "JavaScript:allQuotes()">View all quotes</a></p>

      <p id = "quizSpot">
         <a href = "JavaScript:openQuiz()">Please take our quiz</a></p>

      <script type = "text/javascript">
         // variable that gets the last midification date and time
         var modDate = new Date( document.lastModified );

         // write the last modified date and time to the page
         document.write ( "This page was last modified " +
            modDate.toLocaleString() );
      </script>

   </body>
</html>
```


Explorer User Prompt

Script Prompt:

Please enter your name

GalAnt


Putting It All Together - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch12\final.html
Go Links
Done
My Computer

Thursday, July 10, 2003 1:29:44 PM

Good Afternoon, GalAnt, welcome to JavaScript programming!

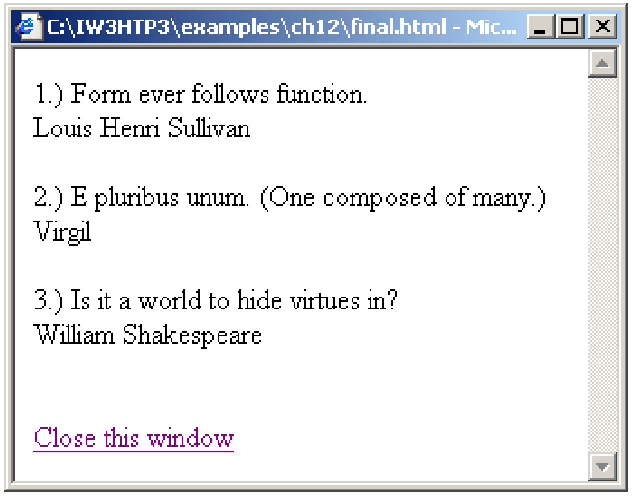
Click here if you are not GalAnt

Form ever follows function.
Louis Henri Sullivan

View all quotes

Please take our quiz

This page was last modified Thursday, July 10, 2003 12:36:26 PM



1.) Form ever follows function.
Louis Henri Sullivan

2.) E pluribus unum. (One composed of many.)
Virgil

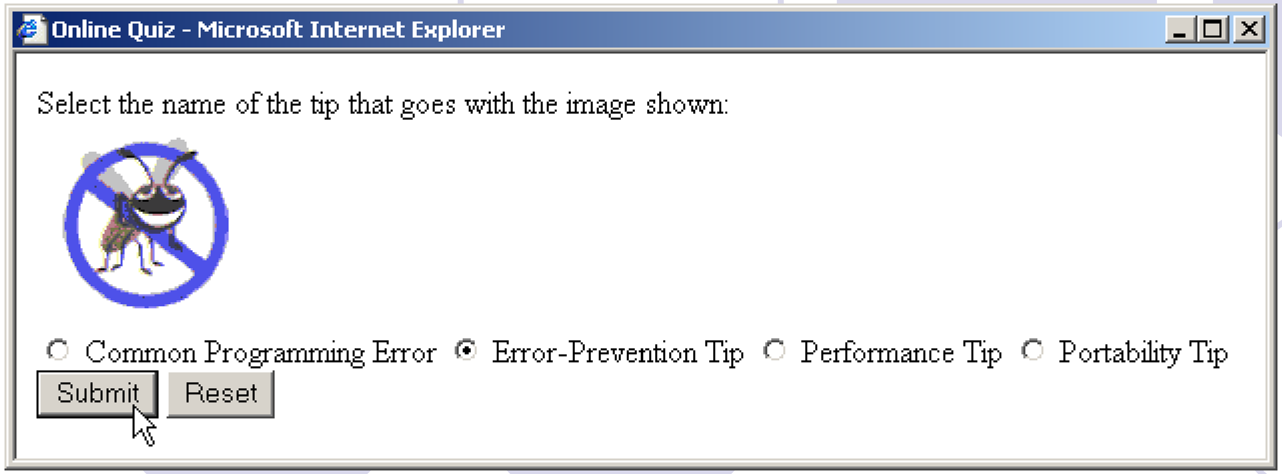
3.) Is it a world to hide virtues in?
William Shakespeare

Close this window

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- Fig. 12.14: quiz2.html -->
<!-- Online Quiz            -->

<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
 <title>Online Quiz</title>

 <script type = "text/JavaScript">
    <!--
    function checkAnswers()
    {
       // determine whether the answer is correct
       if ( myQuiz.radiobutton[ 1 ].checked )
          window.opener.quizSpot.innerText =
             "Congratulations, your answer is correct";
       else // if the answer is incorrect
          window.opener.quizSpot.innerHTML = "Your answer is incorrect." +
             " Please try again <br /> <a href= \" JavaScript:openQuiz()" +
             " \" > Please take our quiz</a>";

       window.opener.focus();
```

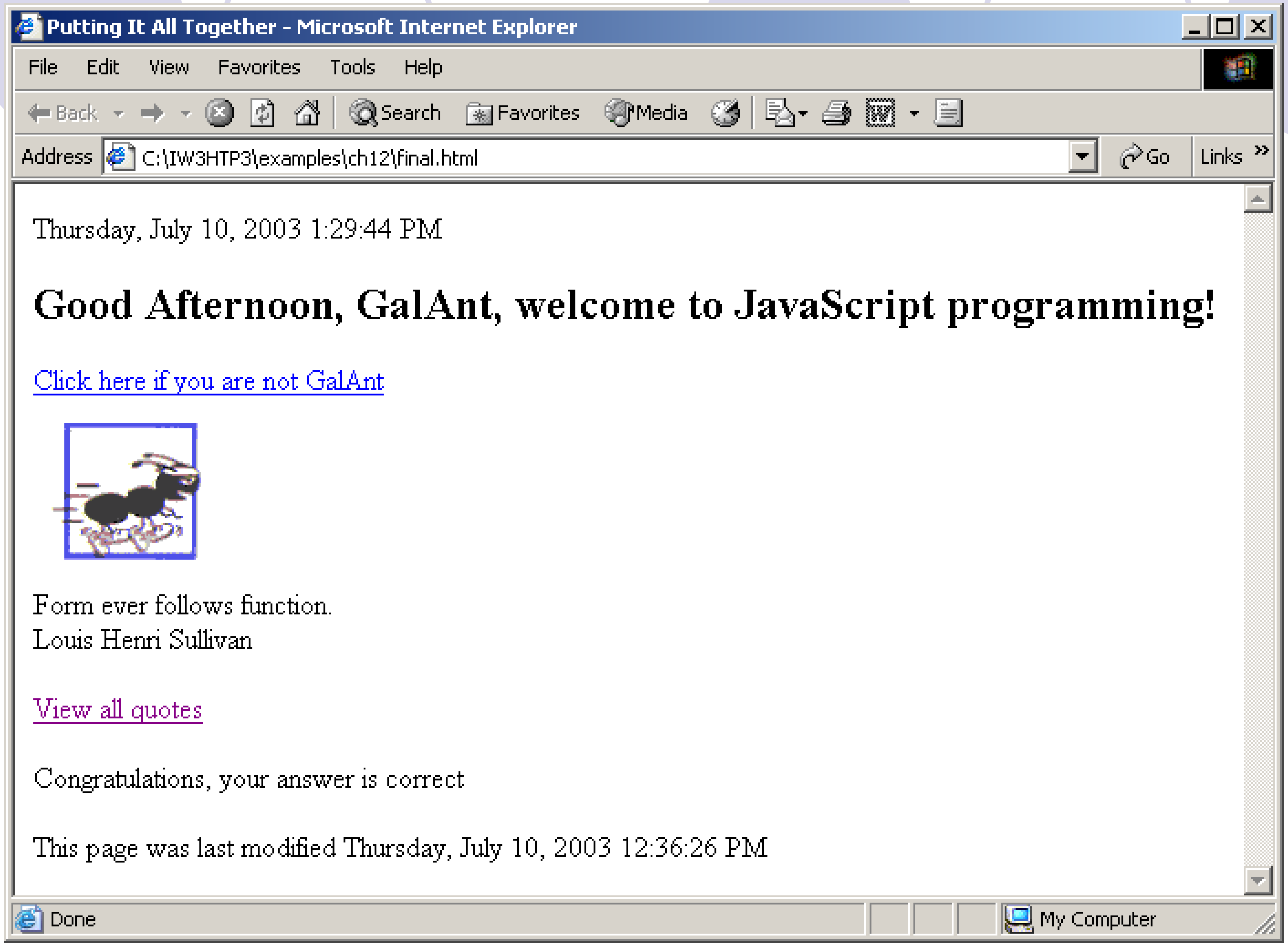
```
      window.close();
   } // end checkAnswers function
   //-->
</script>

</head>

<body>
   <form id = "myQuiz" action = "JavaScript:checkAnswers()">
      <p>Select the name of the tip that goes with the image shown:<br />
         <img src = "EPT.gif" width = "108" height = "100"
            alt = "mystery tip"/>
         <br />

         <input type = "radio" name = "radiobutton" value = "CPE" />
         <label>Common Programming Error</label>

         <input type = "radio" name = "radiobutton" value = "EPT" />
         <label>Error-Prevention Tip</label>

         <input type = "radio" name = "radiobutton" value = "PERF" />
         <label>Performance Tip</label>

         <input type = "radio" name = "radiobutton" value = "PORT" />
         <label>Portability Tip</label><br />
```

```

            <input type = "submit" name = "Submit" value = "Submit" />
            <input type = "reset" name = "reset" value = "Reset" />
         </p>
      </form>
   </body>
</html>
```



Select the name of the tip that goes with the image shown:

Common Programming Error [✓ Error-Prevention Tip] Performance Tip Portability Tip


Putting It All Together - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch12\final.html
Go Links
Done
My Computer

Thursday, July 10, 2003 1:29:44 PM

Good Afternoon, GalAnt, welcome to JavaScript programming!

Click here if you are not GalAnt

Form ever follows function.
Louis Henri Sullivan

View all quotes

Congratulations, your answer is correct

This page was last modified Thursday, July 10, 2003 12:36:26 PM

Assignment 10

- 1) Exercise # 12.7.

Due Date for A # 10:

- Next Monday before your lecture.