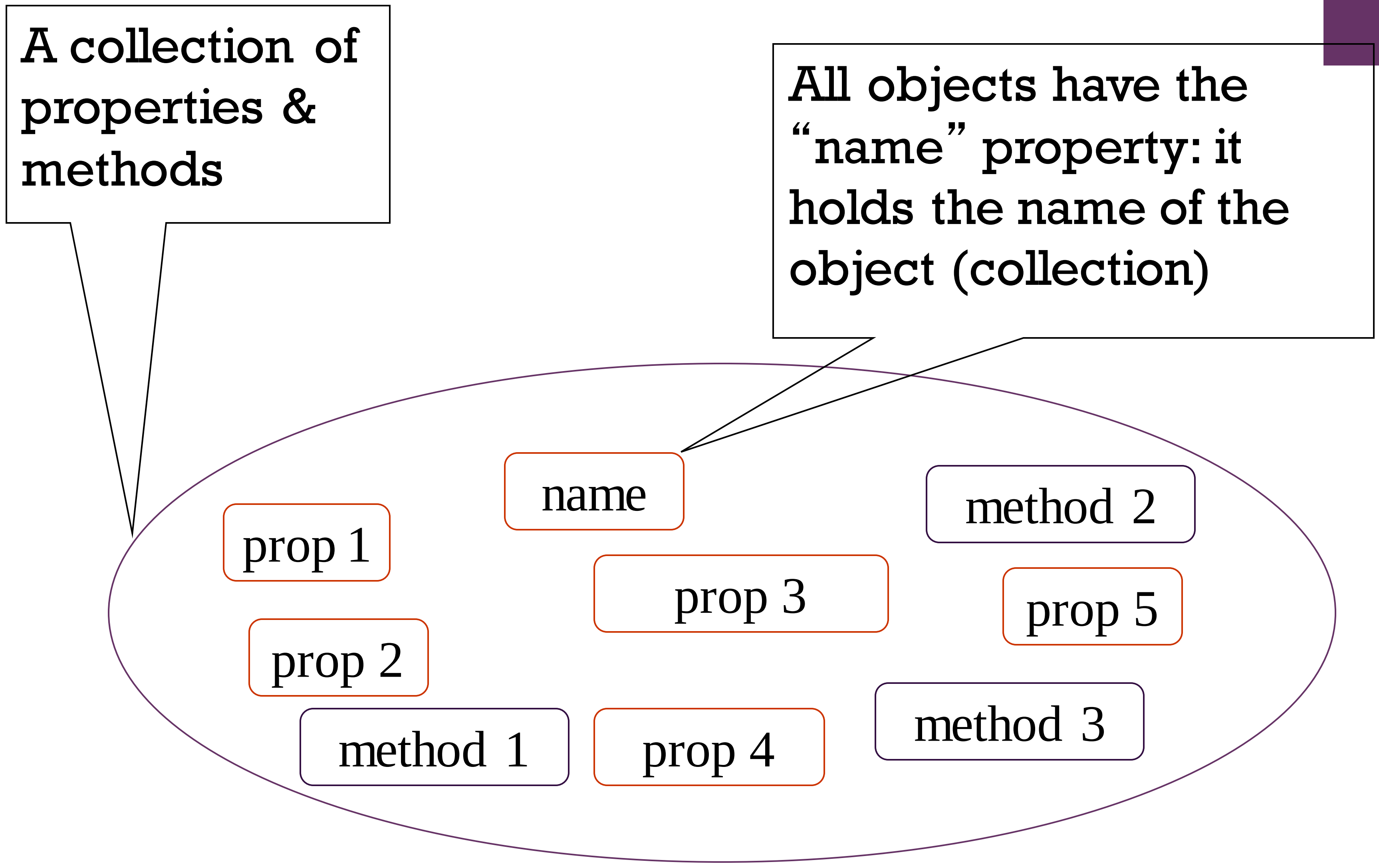
Objects in JavaScript

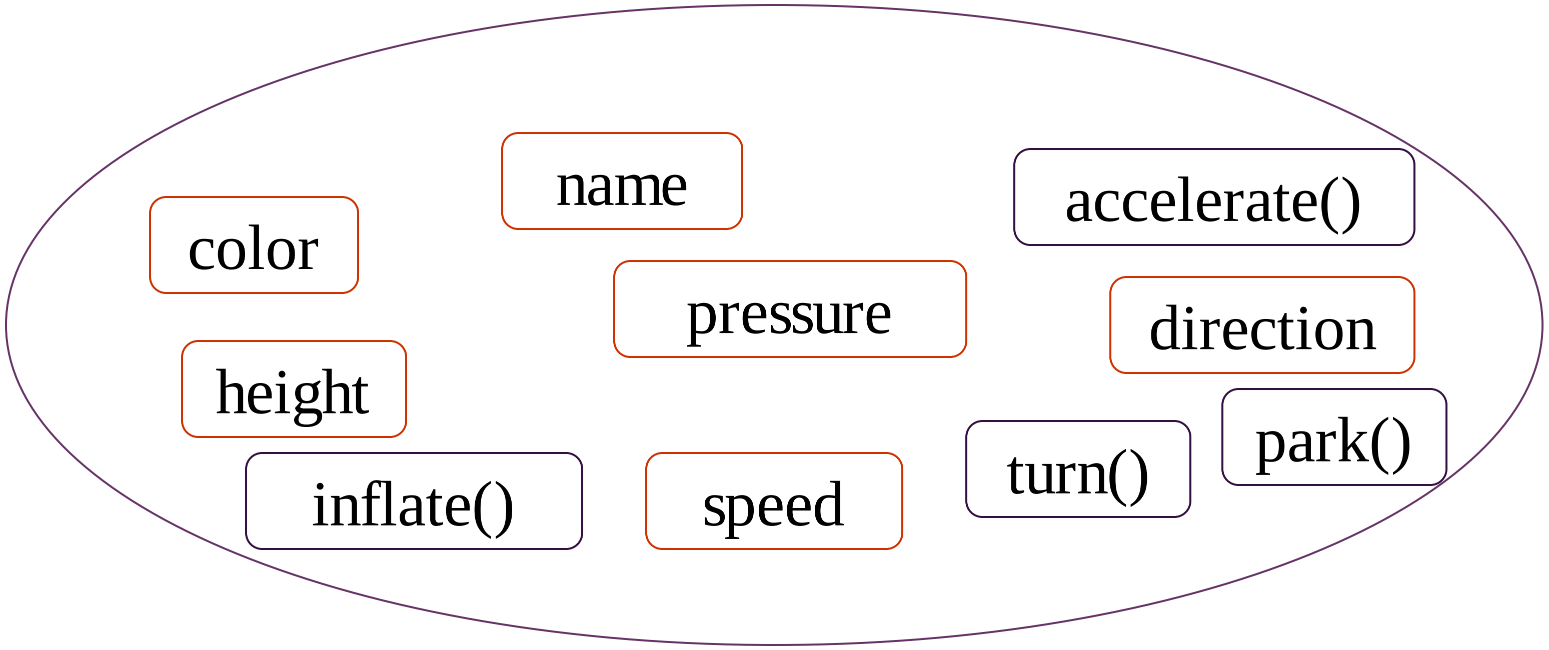
JavaScript: Object Based

- Objects are at the very heart of JavaScript. Objects drive the language, and it is impossible to write useful JavaScript without them.
- To qualify as object-oriented, programming languages must provide support for the following:
  - Data Abstraction
  - Encapsulation
  - Data protection
  - Inheritance
- JavaScript is not an object-oriented language.
- JavaScript does not support data abstraction in the form of Classes, neither is there support for data protection.
- However, JavaScript is defined as an object-based language.

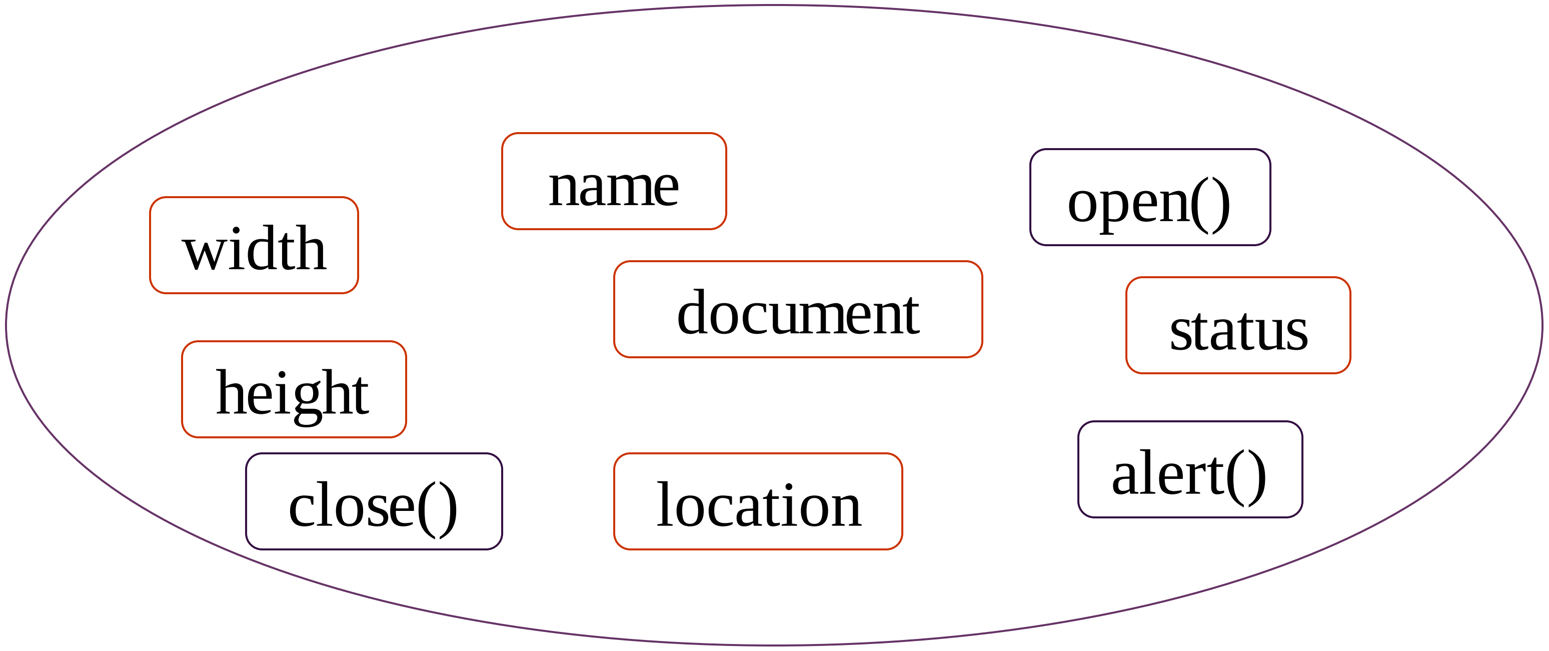
Object: A *named* collection of properties (data, state) & methods (instructions, behavior)


A collection of properties & methods
All objects have the "name" property: it holds the name of the object (collection)
name
prop 1
prop 2
prop 3
prop 4
prop 5
method 1
method 2
method 3

Example: A Bicycle



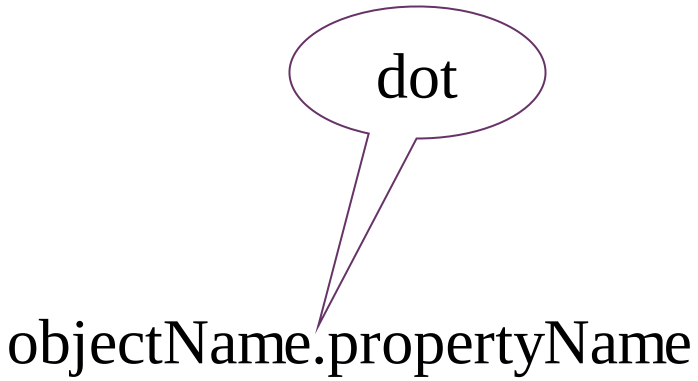
Example: JavaScript's "window" Object



Properties

- Properties are object attributes.
- Objects may have a single or several properties
- A property may have one of the following values:
  - Number, Text, Boolean
  - Array, Functions
  - Objects
- Object properties are defined by using the object's name, a period, and the property name.
  - e.g., background color is expressed by: **document.bgcolor** where **document** is the object, **bgcolor** is the property.
- Example

```
<script type="text/javascript">

let txt="Hello World!";

document.write(txt.length);

</script>
```



Methods

- Methods are functions associated with an object that can be used to manipulate that object

- Example:

window.close() - Here "close()" is a method that has been defined for the "window" object. Its function is to close the "window"

dot

objectName.methodName(argumnets)

Info is passed on to the method through one or more arguments

- Example

```
<script type="text/javascript">
let str="Hello world!";
document.write(str.toUpperCase());
</script>
```

+
The JavaScript Object

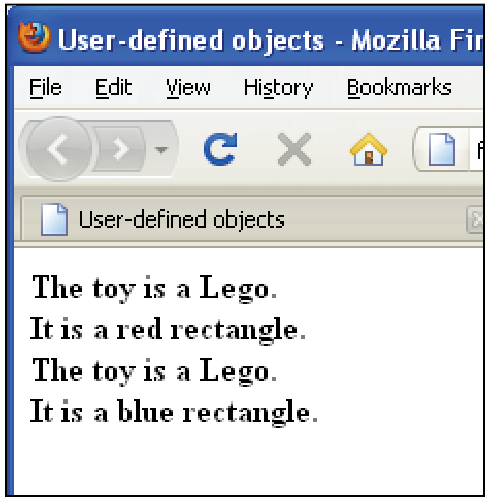
- Objects refers to windows, documents, images, tables, forms, buttons or links, etc.
- There are several types of objects in JavaScript:
  - Native (build in):
    - strictly defined within the language (Number, String, Image, etc.)
    - (additional) JavaScript objects (documents, paragraphs, etc.)
  - Browser objects
    - Objects that contain info *not* about the contents of the display, but the browser itself
    - Examples: history, navigator
  - User-defined object

Example, build in objects

```
<!DOCTYPE html>
<html lang="en”> <head> <title>Chapter 5, Example 1</title></head>
<body>
<script>
let myString = "Welcome to Wrox books. " + "The Wrox website is www.wrox.com. " +
                  "Visit the Wrox website today. Thanks for buying Wrox";
let foundAtPosition = 0;
let wroxCount = 0;
while (foundAtPosition != -1) {
          foundAtPosition = myString.indexOf("Wrox", foundAtPosition);
          if (foundAtPosition != -1) {
             wroxCount++;
             foundAtPosition++;
          }
       }
      document.write("There are " + wroxCount + " occurrences of the word Wrox");
   </script>
</body>
</html>
```

User defined objects

```
<html>
<head><title>User-defined objects</title>
<script type= "text/javascript">
          let toy = new Object(); // Create the object
          toy.name = "Lego"; // Assign properties to the object
          toy.color = "red";
          toy.shape = "rectangle";
          toy.display=printObject; // Function name is assigned as a
                                   // property of the object

          function printObject(){
                    document.write("<b>The toy is a " + toy.name + ".<br>");
                    document.write("It is a " + toy.color + " " + toy.shape + ".<br />");
          }
</script>
</head>
<body> <script type = "text/javascript">
          toy.display(); //Object method is called toy.color="blue";
          toy.display();
</script>
</body>
</html>
```


User-defined objects - Mozilla Fir
File Edit View History Bookmarks
User-defined objects

```
The toy is a Lego.
It is a red rectangle.
The toy is a Lego.
It is a blue rectangle.
```

User-defined "objects" (constructor)

- simply define a function that serves as a constructor
- specify data fields & methods using `this`

- no data hiding: can't protect data or methods

```
// Dave Reed        Die.js        9/20/01
//
// Die class definition
//////////////////////////////////////////////

function Die(sides)
{
   this.numSides = sides;
   this.numRolls = 0;
   this.Roll = Roll;
}

function Roll()
{
    this.numRolls++;
    return Math.floor(Math.random()*this.numSides) + 1;
}
```

define Die function (i.e., constructor)

initialize data fields in the function, preceded with this

similarly, assign method to separately defined function (which uses this to access data)

This reference

- Internally, JavaScript creates an object, and then calls the constructor function.
- Inside the constructor, the variable *this* is initialized to point to this newly created object.
- The *this* keyword is a sort of shorthand reference that keeps track of the current object.
- When a function is used as a constructor, the *this* keyword is used to set the properties for the object that was just created.
- In this way you can create as many objects as you need and JavaScript *this* will refer to the current object.

Creating objects example

```
<html>
<!-- Dave Reed  js19.html  9/20/01 -->

<head>
  <title>Dice page</title>

  <script language="JavaScript"
          src="Die.js">
  </script>
</head>

<body>
  <script language="JavaScript">
    die6 = new Die(6);
    die8 = new Die(8);

    roll6 = -1;    // dummy value to start loop
    roll8 = -2;    // dummy value to start loop
    while (roll6 != roll8) {
      roll6 = die6.Roll();
      roll8 = die8.Roll();

      document.write("6-sided: " + roll6 +
                     "&nbsp;&nbsp;&nbsp;&nbsp;" +
                     "8-sided: " + roll8 + "<br>");
    }

    document.write("<p>Number of rolls: " +
                   die6.numRolls);
  </script>
</body>
</html>
```

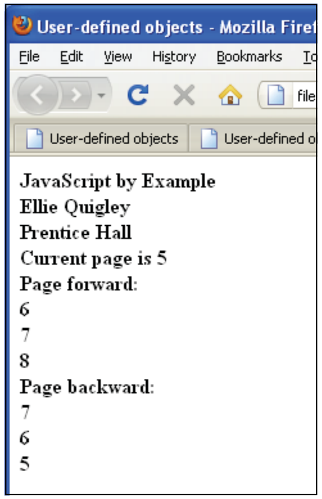
create a Die object using new here, the argument to Die initializes numSides for that particular object

each Die object has its own properties (numSides & numRolls)

Roll(), when called on a particular Die, accesses its numSides property and updates its NumRolls

+ Example

```
<html>
<head><title>User-defined objects</title>  <script type ="text/javascript">
           function Book(title, author, publisher){                    // Receiving  parameters
                     this.pagenumber=0; // Properties
                     this.title  = title;
                     this.author = author;
                     this.publisher  = publisher;
                     this.uppage = pageForward;
                     this.backpage = pageBackward;
                     }
          function pageForward(){ this.pagenumber++;  return this.pagenumber; } // Functions to be used as methods
          function pageBackward(){ this.pagenumber--; return this.pagenumber; }
</script> </head>
<body>
<script type = "text/javascript">
           let myBook = new Book("JavaScript by Example”, "Ellie Quigley", "Prentice Hall" ); // Create new object
          myBook.pagenumber=5; //Assign a page number
          document.write( "<b>"+ myBook.title +  "<br>" + myBook.author + "<br>" + myBook.publisher  +
                      "<br>Current page is " + myBook.pagenumber );
          document.write("<br>Page forward: " );
          for(i=0;i<3;i++){
                     document.write("<br />" + myBook.uppage()); // Move forward a page
                     }
          document.write("<br />Page backward: ");
          for(;i>0; i--){
                     document.write("<br />" + myBook.backpage()); // Move back a page
}
</script> </body>
</html>
```

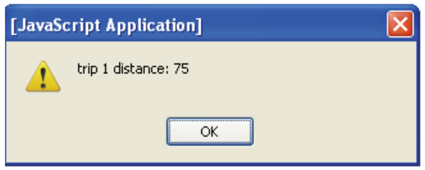

User-defined objects - Mozilla Firef
File Edit View History Bookmarks To
User-defined objects
User-defined o

```
JavaScript by Example
Ellie Quigley
Prentice Hall
Current page is 5
Page forward:
6
7
8
Page backward:
7
6
5
```

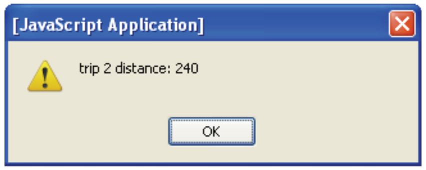
Inline Functions as Methods

- Rather than naming a function outside the class, an inline or anonymous function can be assigned directly to a property within the constructor function.
- Every instance of the class will have a copy of the function code.
- Because it is part of the definition of the constructor, only objects of the correspondent class will have access to the method, thereby encapsulating the method.
- In previous examples the functions that served as methods were defined outside of the constructor, making available to any class.

Inline methods



trip 1 distance: 75



trip 2 distance: 240

```
<html> <head><title>functions</title>
<script type="text/javascript">

function Distance(r, t){ //Constructor function
            this.rate = r;
            this.time = t;
            this.calculate=function() { return r * t; } // anonymous
            }
</script>
</head> <body>
 <script type ="text/javascript">
            let trip1 = new Distance(50, 1.5);
            let trip2 = new Distance(75, 3.2);
            alert("trip 1 distance: "+ trip1.calculate());
            alert("trip 2 distance: "+ trip2.calculate());
</script>
</body>
</html>
```

+
Using object literals

- Object literals enable you to create objects that support many features without directly invoking a function.
  - When a function acts as constructor you have to keep track of the order of arguments that will be passed, and so on.
  - They are similar to hashes in other languages using key/value pairs to represent fields.
  - The fields can be nested.
- The basic syntax for an object literal is:
  - A colon separates the property name from its value.
  - A comma separates each set of name/value pairs from the next set.
  - The comma should be omitted from the last name/value pair.
  - Even with nested key/value pairs, the last key/value pair does not have a comma.
  - The entire object is enclosed in curly braces.

+ Examples, object literals

- property names can be anything but undefined:

  let silly = {42: "hi", true: 3.14, "q": "Q"};

- you can add properties to an object after creating it:

  silly.favoriteMovie = "Fight Club";

  silly["anotherProp"] = 123;

- if you access a non-existent property, it is undefined:

  silly.fooBar

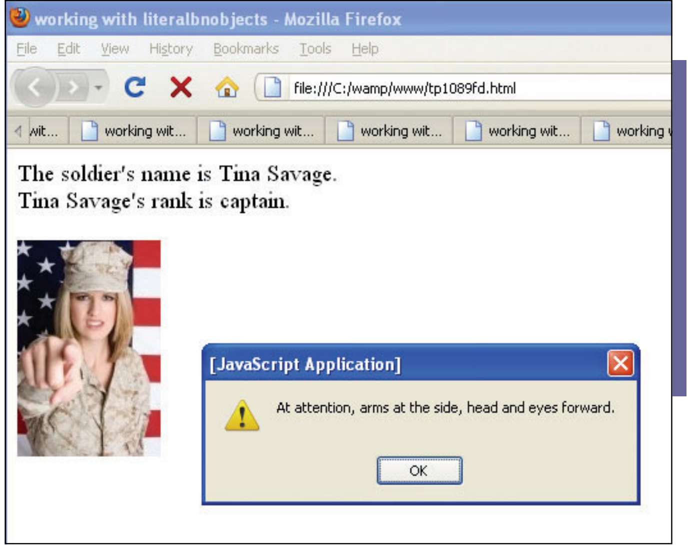
  typeof(silly.fooBar)

  Undefined

- delete removes a property from the object

```
 silly.age= 29;       // if only...
 delete silly.age;     // no one will know!
 typeof(silly.age)
undefined
```

+
Using literals


working with literalbnobjects - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/wamp/www/tp1089fd.html
[JavaScript Application]
OK

The soldier's name is Tina Savage.
Tina Savage's rank is captain.

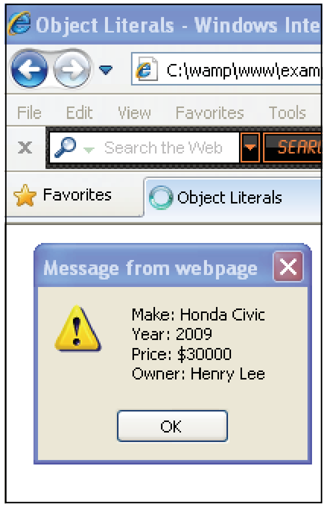
At attention, arms at the side, head and eyes forward.

```
<html>
<head><title>working with literal objects</title>
<script type="text/javascript">
            let soldier = {
                         name: undefined,
                        rank: "captain",
                        picture: "keeweeboy.jpg",
                        fallIn: function() {
                                    alert("At attention, arms at the side, head and eyes forward.");
                                    },
                        fallOut: function() {
                                    alert("Drop out of formation, step back, about face!");}
                                    };
</script>
</head> <body> <big>
<script type="text/javascript">
            soldier.name="Tina Savage"; // Assign value to object property
            document.write("The soldier's name is ",  soldier.name,".<br />");
            document.write(soldier.name+"'s rank is ",   soldier.rank+."<br />");
            document.write("<img src='"+soldier.picture+"'>")
            soldier.fallIn();//call object's methods
            soldier.fallOut();
</script>
</big> </body> </html>
```

+

Using literals

```
<html> <head><title>Object Literals</title>
<script type = "text/javascript">
        let Car = { // Create a Car object
                        make:undefined,
                        year:2006,
                        price:undefined,
                        owner:{   name:"Henry Lee", cell_phone:"222-222-2222",
                                  address:{street:"10 Main", city:"SF", state:"CA"} },
                        dealer: "SF Honda",
                        display: function() {   details="Make:"+Car.make+"\n";
                                                details += "Year:"+Car.year+"\n";
                                                details += "Price: $"+Car.price+"\n";
                                                details += "Owner:"+ Car.owner.name+"\n";
                                                alert(details);}
                        };
</script> </head>  <body>
<script type="text/javascript">
        Car.make="Honda Civic";// Assign value
        Car.year=2009; // Update the year
        Car.price=30000;
        Car.display();
</script> </body>  </html>
```

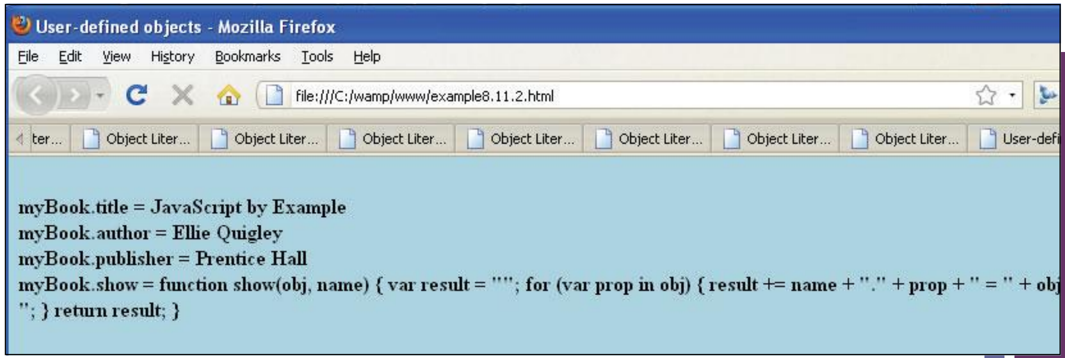


```
Make: Honda Civic
Year: 2009
Price: $30000
Owner: Henry Lee
```

+
for/in loop

- JavaScript provides the *for/in* loop, which can be used to iterate through a list of object properties or array elements.
- The *for/in* loop reads: for each property in an object (or for each element in an array) get the name of each property (element), in turn, and for each of the properties (elements), execute the statements in the block that follows.
- The *for/in* loop is a convenient mechanism for looping through the properties of an object.

+
# example


User-defined objects - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/wamp/www/example8.11.2.html
ter... Object Liter... Object Liter... Object Liter... Object Liter... Object Liter... Object Liter... Object Liter... User-defi

```
myBook.title = JavaScript by Example
myBook.author = Ellie Quigley
myBook.publisher = Prentice Hall
myBook.show = function show(obj, name) { var result = ""; for (var prop in obj) { result += name + "." + prop + " = " + obj
"; } return result; }
```

```
<html>
<head><title>User-defined objects</title>
<script type = "text/javascript">
        function book(title, author, publisher){
                this.title = title;
                this.author = author;
                this.publisher = publisher;
                this.show=show; // Define a method for the object
                }
        function show(obj, name){       // Function to show the object's properties
                var result = "";
                for (let prop in obj){
                        result += name + "." + prop + " = " + obj[prop] + "<br />";
                        }
                return result;
                }
</script> </head> <body bgcolor="lightblue">
<script type="text/javascript">
myBook = new book("JavaScript by Example", "Ellie", "Prentice Hall");
document.write("<br /><b>" + myBook.show(myBook, "myBook"));
</script> </body> </html>
```

Modern JavaScript samples and concepts

+ Examples (basic)

```
let user = new Object(); // "object constructor" syntax
let user = {};  // "object literal" syntax
```

```
let user = {
  name: "John",
  age: 30
};
```

```
user.noSuchProperty === undefined
```

```
"key" in object
```

```
for(key in object) {
  // executes the body for each
 key among object properties
}
```

```
let user = { name: "John" };

let admin = user; // copy the reference
```

```
let user = {
  name: "John",
  sizes: {
    height: 182,
    width: 50
  }
};

let clone = Object.assign({}, user);
```

Example (methods with *this*)

```
let user = {
  name: "John",
  age: 30
};

user.sayHi = function() {
  alert("Hello!");
};

user.sayHi(); // Hello!
```

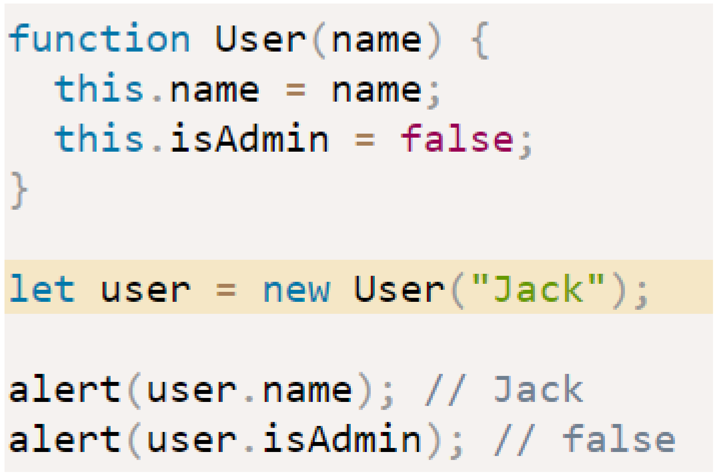
```
let user = {
  sayHi() { // same as "sayHi: function()"
    alert("Hello");
  }
};
```

```
let user = {
  name: "John",
  age: 30,

  sayHi() {
    alert(this.name);
  }

};
```

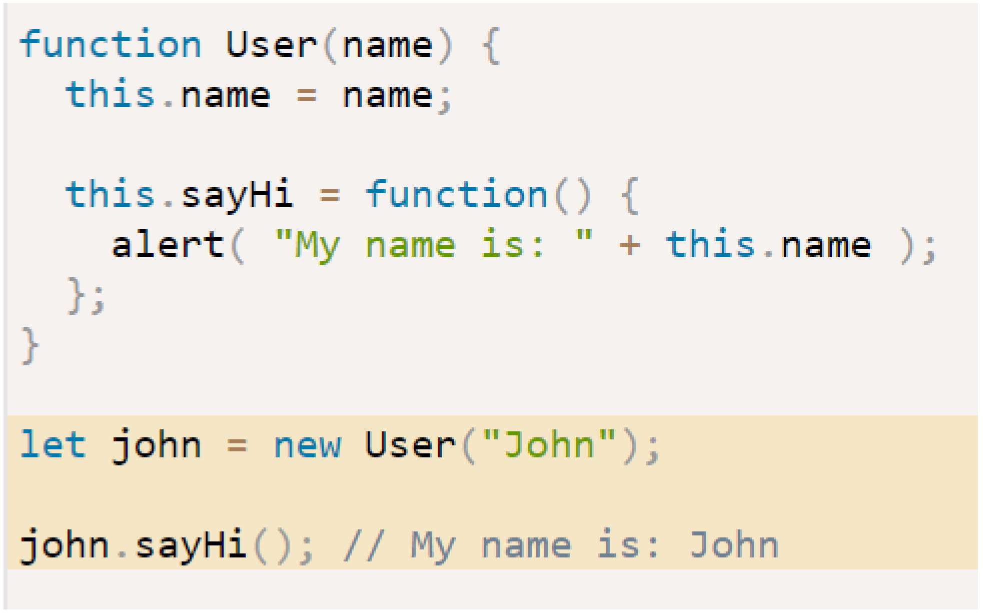
```
let user = {
  sayHi: function() {
    alert("Hello");
  }
};
```

+

Examples (*new* operator)



```
function User(name) {
  this.name = name;
  this.isAdmin = false;
}

let user = new User("Jack");

alert(user.name); // Jack
alert(user.isAdmin); // false
```



```
function User(name) {
  this.name = name;

  this.sayHi = function() {
    alert( "My name is: " + this.name );
  };
}

let john = new User("John");

john.sayHi(); // My name is: John
```

Property getters and setters

- Getters and setters
  - *Accessor properties* - they are essentially functions that work on getting and setting a value, but look like regular properties to an external code.

```
let user = {
  name: "John",
  surname: "Smith",

  get fullName() {
    return `${this.name} ${this.surname}`;
  },

  set fullName(value) {
    [this.name, this.surname] = value.split(" ");
  }
};
```

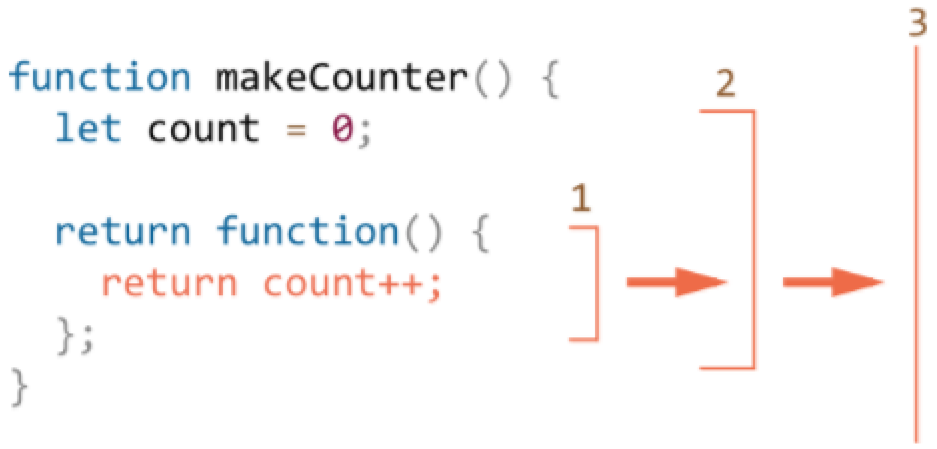
```
// set fullName is executed with the given value.
user.fullName = "Alice Cooper";

alert(user.name); // Alice
alert(user.surname); // Cooper
```

Closure

- In JavaScript, every running function, code block, and the script as a whole have an associated object known as the Lexical Environment
- The Lexical Environment object consists of two parts:
  - Environment Record – an object that has all local variables as its properties (and some other information like the value of this)
  - A reference to the outer lexical environment, usually the one associated with the code lexically right outside of it (outside of the current curly brackets)
- When code wants to access a variable – it is first searched for in the inner Lexical Environment, then in the outer one, then the more outer one and so on until the end of the chain.

+ Closure (2)

```
function makeCounter() {
  let count = 0;

  return function() {
    return count++; // has access to the outer counter
  };
}

let counter = makeCounter();

alert( counter() ); // 0
alert( counter() ); // 1
alert( counter() ); // 2
```



```
function makeCounter() {
  let count = 0;

  return function() {
    return count++;
  };
}
```

Prototypal inheritance

- In programming, we often want to take something and extend it

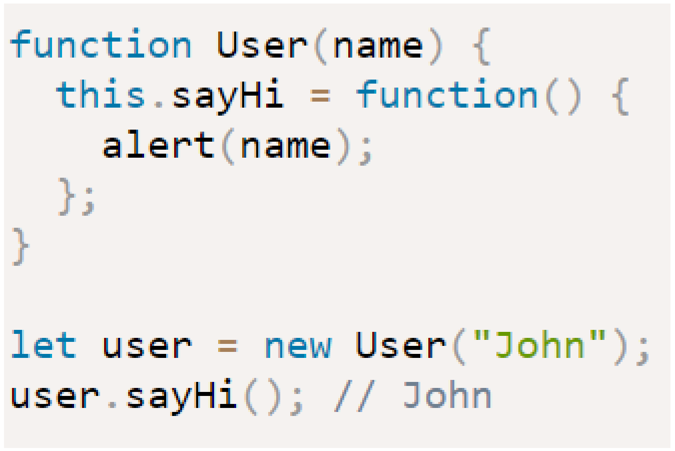
```
let animal = {
  eats: true,
  walk() {
    alert("Animal walk");
  }
};
```

```
let rabbit = {
  jumps: true,
  __proto__: animal
};
```

```
rabbit.walk();
```

Class patterns

- From earlier



```
function User(name) {
  this.sayHi = function() {
    alert(name);
  };
}

let user = new User("John");
user.sayHi(); // John
```

functional class pattern

+ Class patterns (2)

factory class pattern

```javascript
function User(name, birthday) {
  // only visible from other methods inside User
  function calcAge() {
    return new Date().getFullYear() - birthday.getFullYear();
  }

  return {
    sayHi() {
      alert(`${name}, age:${calcAge()}`);
    }
  };
}

let user = User("John", new Date(2000, 0, 1));
user.sayHi(); // John, age:17
```

prototype class pattern

```javascript
function User(name, birthday) {
  this._name = name;
  this._birthday = birthday;
}

User.prototype._calcAge = function() {
  return new Date().getFullYear() - this._birthday.getFullYear();
};

User.prototype.sayHi = function() {
  alert(`${this._name}, age:${this._calcAge()}`);
};

let user = new User("John", new Date(2000, 0, 1));
user.sayHi(); // John, age:17
```

Classes

- The "**class**" construct allows to define prototype-based classes with a clean, nice-looking syntax.

```
function User(name) {
  this.name = name;
}

User.prototype.sayHi = function() {
  alert(this.name);
}

let user = new User("John");
user.sayHi();
```

$\longleftrightarrow$

```
class User {

  constructor(name) {
    this.name = name;
  }

  sayHi() {
    alert(this.name);
  }

}

let user = new User("John");
user.sayHi();
```

Class inheritance

```
class Animal {

  constructor(name) {
    this.speed = 0;
    this.name = name;
  }

  run(speed) {
    this.speed += speed;
    alert(`${this.name} runs with speed ${this.speed}.`);
  }

  stop() {
    this.speed = 0;
    alert(`${this.name} stopped.`);
  }

}
```

```
// Inherit from Animal
class Rabbit extends Animal {
  hide() {
    alert(`${this.name} hides!`);
  }
}
```

```
let rabbit = new Rabbit("White Rabbit");

rabbit.run(5); // White Rabbit runs with speed 5.
rabbit.hide(); // White Rabbit hides!
```

Some built-in classes

Date class

- the Date class can be used to access the date and time
  - constructors include:
    - today = new Date(); // sets to current date & time
    - newYear = new Date(2001,0,1); //sets to Jan 1, 2001 12:00AM
  - methods include:
    - newYear.getYear() can access individual components of a date
    - newYear.getMonth()
    - newYear.getDay()
    - newYear.getHours()
    - newYear.getMinutes()
    - newYear.getSeconds()
    - newYear.getMilliseconds()
    - newYear.toString() can convert date to printable String

+
Date example

```
<html>
<!-- Dave Reed  js14.html  9/20/01 -->

<head>
  <title>Time page</title>
</head>

<body>
  Time when page was loaded:
  <script language="JavaScript">
    now = new Date();

    document.write(now.toString() + "<br><br>");

    document.write(now.getHours() + ":" +
                   now.getMinutes() + ":" +
                   now.getSeconds());
</script>
</body>

</html>
```

Here, set to the current date and time using the default constructor

toString displays the full date using month and day names

using the get methods, can display desired components of the date/time

```
<html>
<!-- Dave Reed  js15.html  9/20/01 -->

<head>
  <title>Time page</title>
</head>

<body>
  Time when page was loaded:
  <script language="JavaScript">
    now = new Date();

    time = "AM";
    hours = now.getHours();

    if (hours > 12) {
        hours -= 12;
        time = "PM"
    }
    else if (hours == 0) {
        hours = 12;
    }

    document.write(hours + ":" +
                   now.getMinutes() + ":" +
                   now.getSeconds() + " " +
                   time);
</script>
</body>

</html>
```

Date example (cont.)

suppose we don't like military time

instead of	0:15:22
we want	12:15:22 AM

we must perform the conversions

- need a variable for "AM" or "PM"
- need to adjust hours past noon
- need to handle 12 AM special

Another example

```
<html>
<!-- Dave Reed  js16.html  9/20/01 -->

<head>
  <title>Time page</title>
</head>

<body>
  Time in the new millenium:
  <script language="JavaScript">
    now = new Date();
    newYear = new Date(2001,0,1);

    secs = Math.round((now-newYear)/1000);

    days = Math.floor(secs / 86400);
    secs -= days*86400;
    hours = Math.floor(secs / 3600);
    secs -= hours*3600;
    minutes = Math.floor(secs / 60);
    secs -= minutes*60

    document.write(days + " days, " +
                   hours + " hours, " +
                   minutes + " minutes, and " +
                   secs + " seconds.");
</script>
</body>

</html>
```

you can add and subtract Dates:
the result is a number of milliseconds

here, determine the number of seconds since New Year's day

divide into number of days, hours, minutes and seconds

possible improvements?

+

document object

- Browsers allow you to access information about an HTML document using the document object *(Note: not a class!)*

```
<html>
<!-- Dave Reed  js17.html  9/20/01 -->

<head>
  <title>Documentation page</title>
</head>

<body>
  <table width="100%">
    <tr>
      <td><small><i>
        <script language="JavaScript">
            document.write(document.URL);
        </script>
      </i></small></td>
      <td align="right"><small><I>
        <script language="JavaScript">
            document.write(document.lastModified);
        </script>
      </i></small></td>
    </tr>
  </table>
</body>
</html>
```

document.write(…)
  method that displays text in the page

document.URL
  property that gives the location of the HTML document

document.lastModified
  property that gives the date & time the HTML document was saved

+

navigator object

- can access information about the browser being used to access the Web page using the navigator object *(Again: not a class!)*

navigator.userAgent read-only property returns the user agent string for the current browser, e.g.,

"'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36'"

navigator.appVersion property gives the browser version, e.g.,

"5.0 (Windows NT 10.0; Win64; x64)"

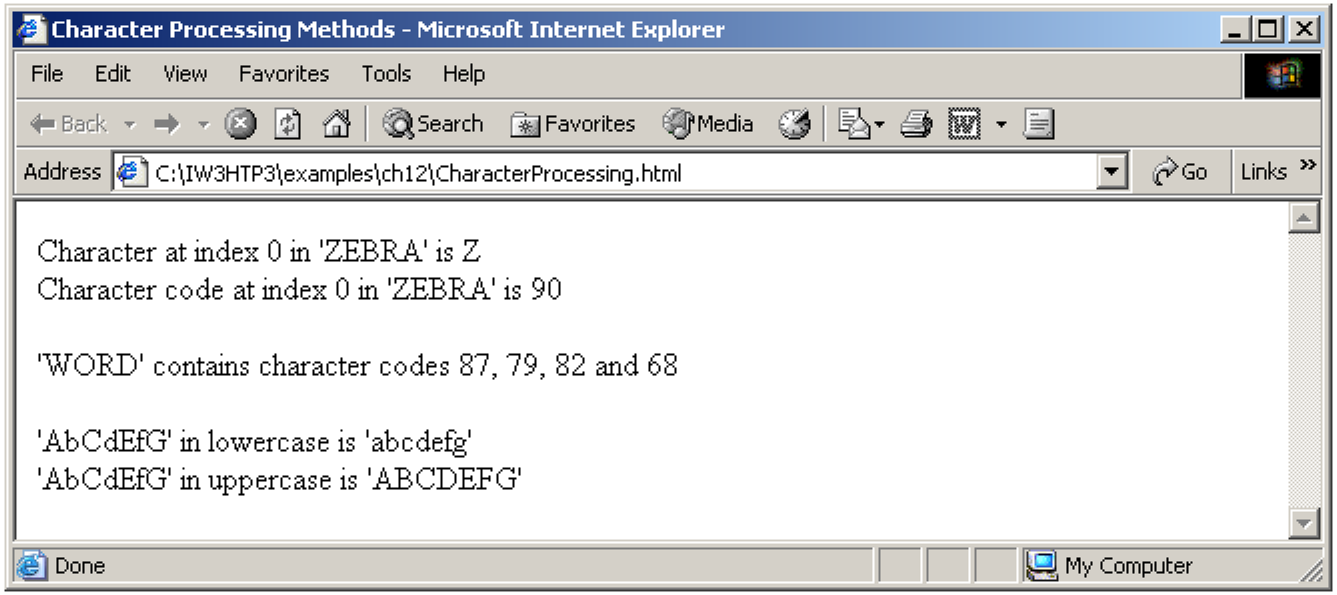
navigator.plugins property gives an array of all of the installed plug-ins

+ String Object

- JavaScript' s string and character-processing capabilities
- Appropriate for processing names, addresses, credit card information, etc.
- Characters
  - Fundamental building blocks of JavaScript programs
- String
  - Series of characters treated as a single unit

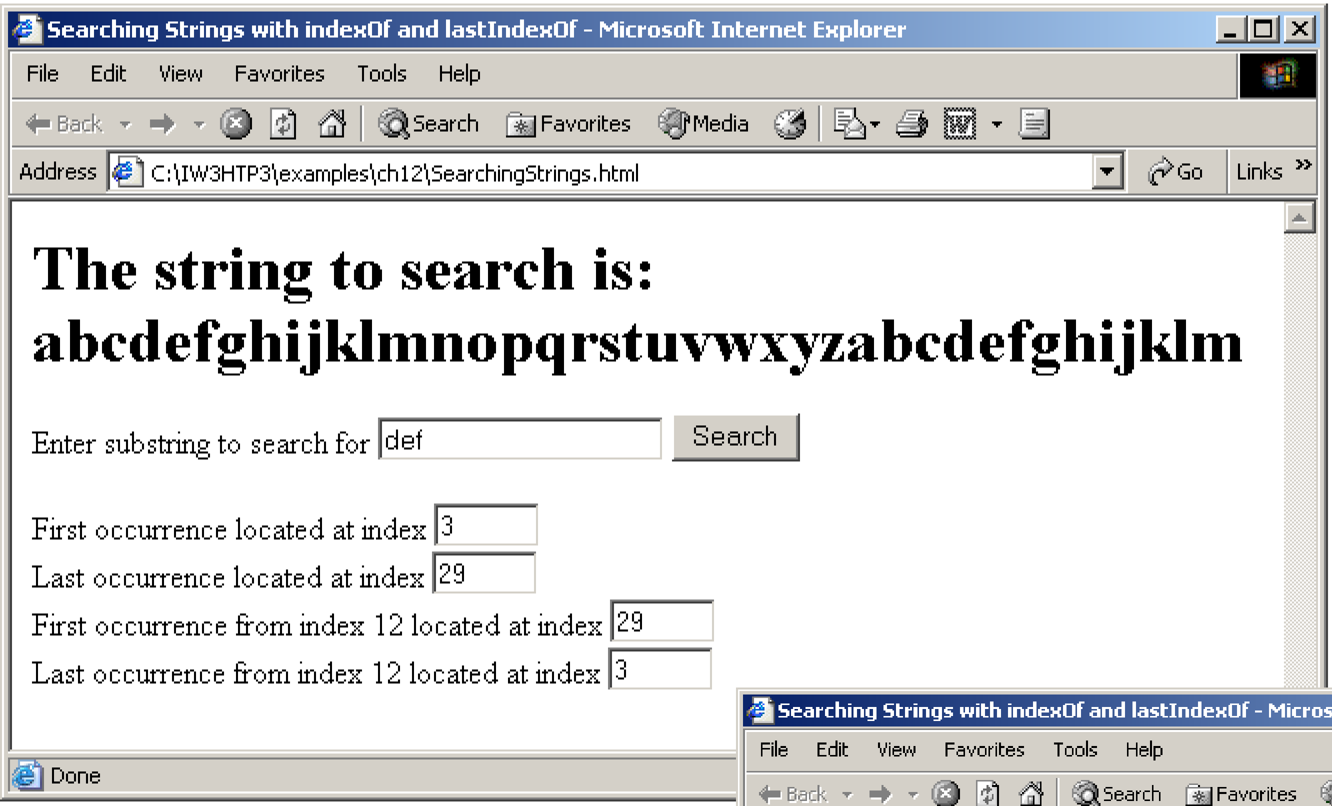
CharacterProcessing
.html

```
<?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'
```

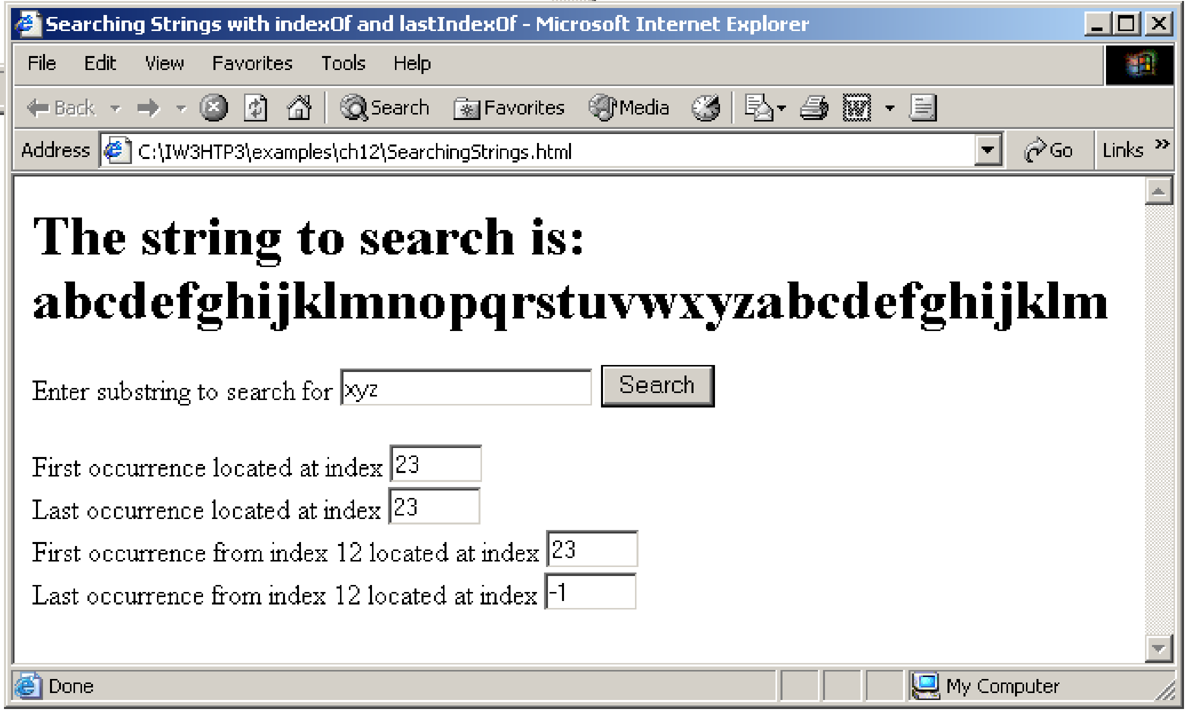
SearchingStrings
.html


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
Search
Done

The string to search is: abcdefghijklmnopqrstuvwxyzabcdefghijklm

Enter substring to search for def

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
Search
Done
My Computer

The string to search is: abcdefghijklmnopqrstuvwxyzabcdefghijklm

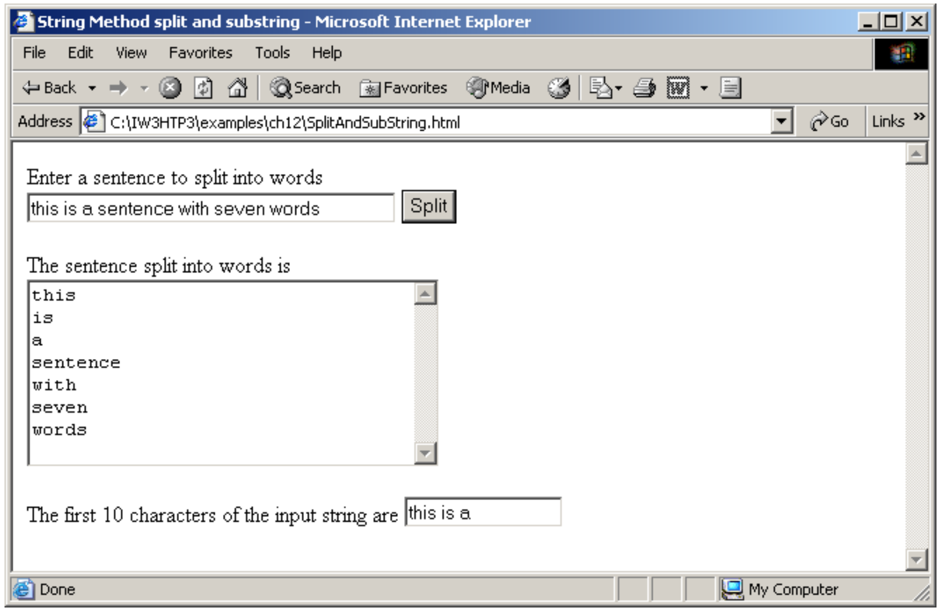
Enter substring to search for xyz

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

```
<?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>
```

Splitting Strings and Obtaining Substrings

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


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

SplitAndSubString
.html

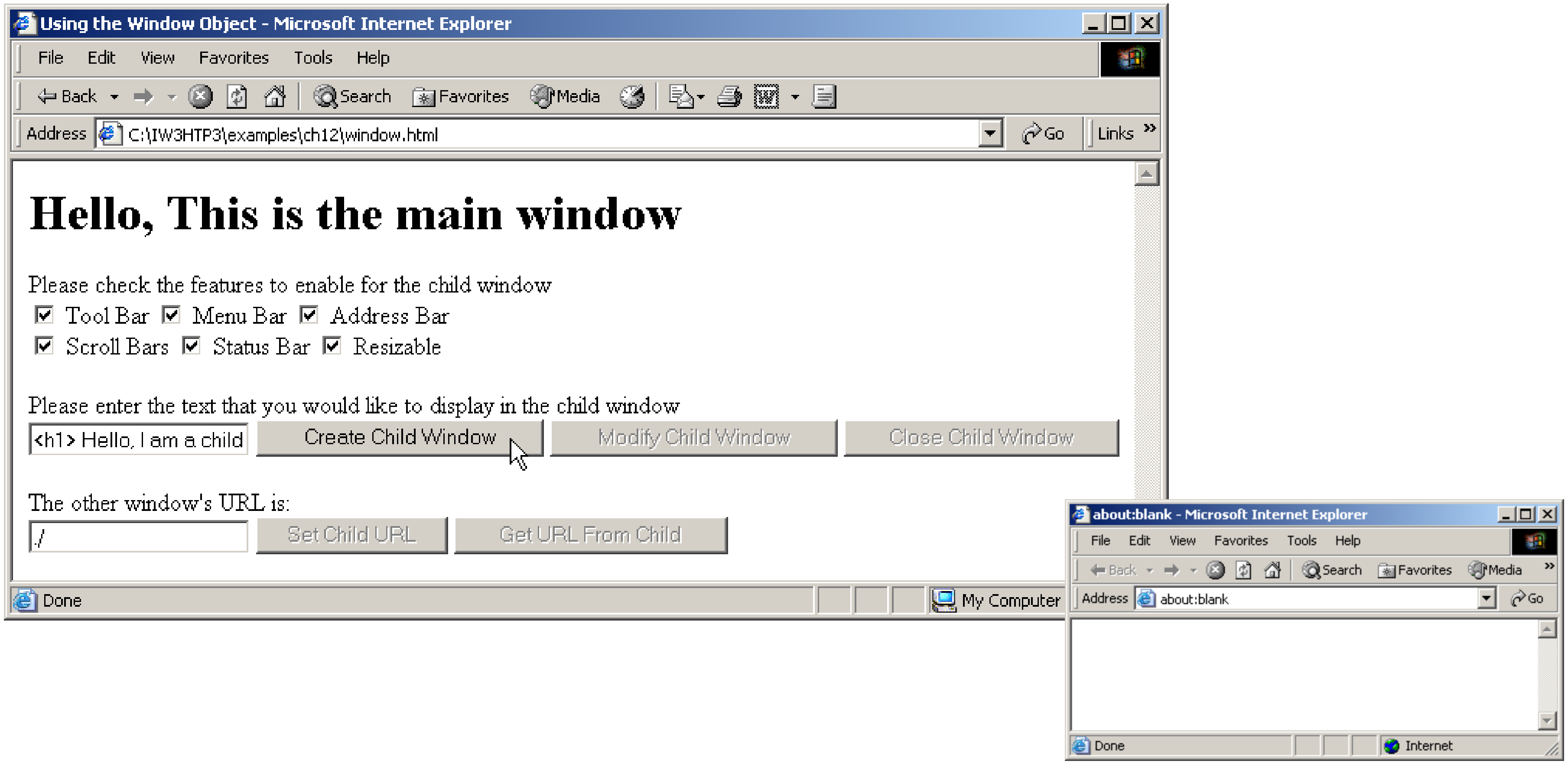
```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- 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>

```

SplitAndSubString
.html

```
   <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>
```

12.8 window Object

- Provides methods for manipulating browser window



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

window.html

```
<?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
```

window.html

```

      // 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

```

window.html

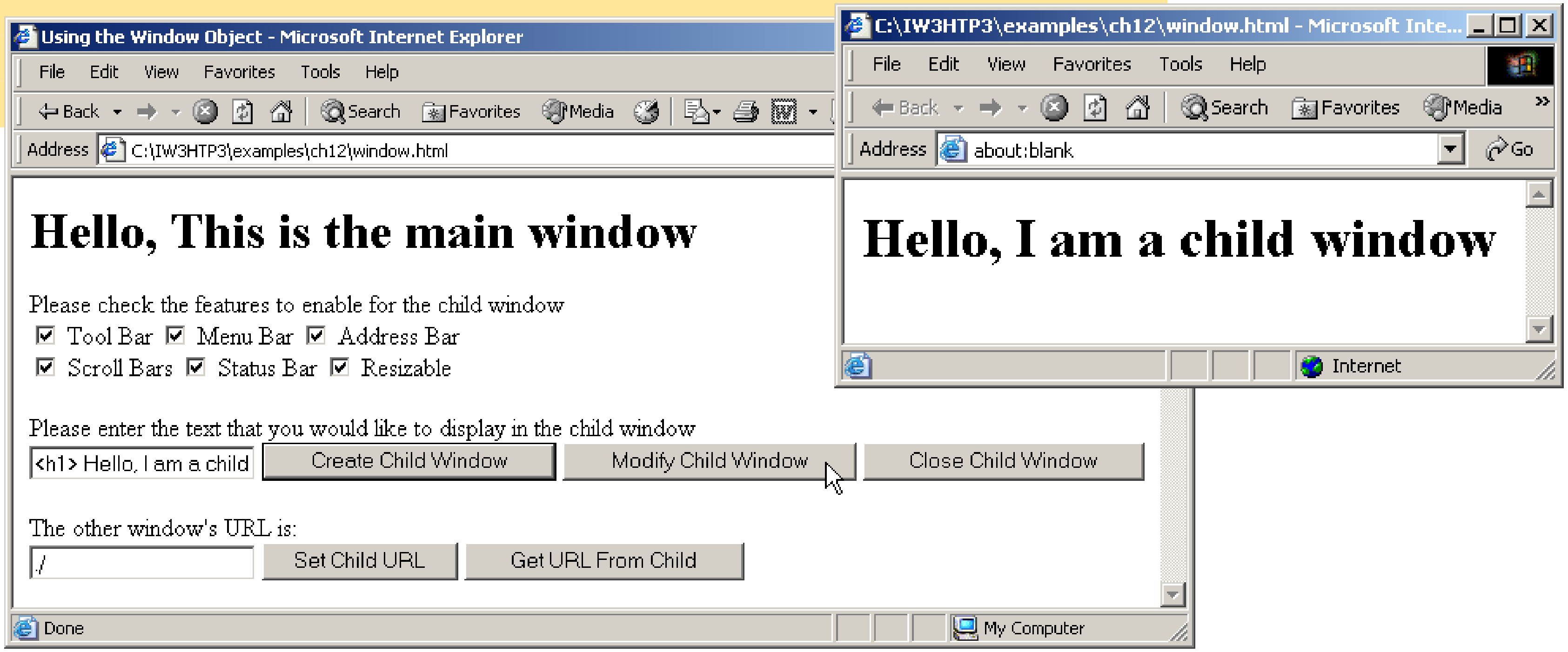
```
   // copy the URL of the child window into the parent window’s myChildURL
   function getChildWindowURL()
   {
      if ( childWindow.closed )
         alert( "You attempted to interact with a closed window" );
      else
         myChildURL.value = childWindow.location;
   } // end function getChildWindowURL

   // 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>
```

window.html

```
<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\>"/>
```

window.html

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

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

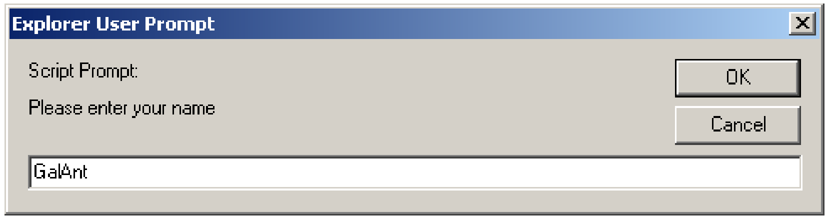
The other window's URL is:

./ | Set Child URL | Get URL From Child

Hello, I am a child window

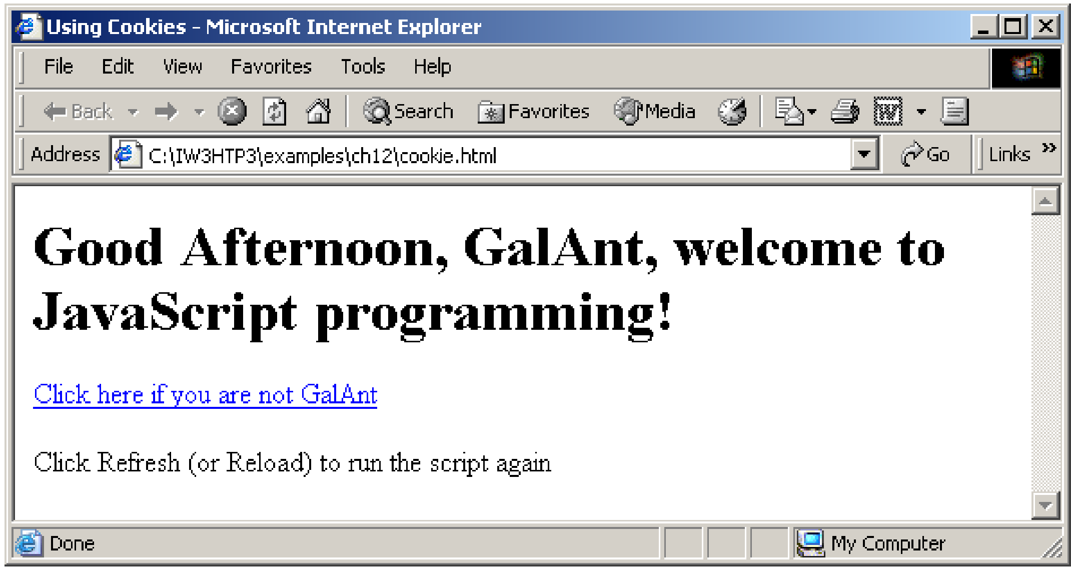
Using Cookies

- 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


Explorer User Prompt

Script Prompt:
Please enter your name

GalAnt


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

Good Afternoon, GalAnt, welcome to JavaScript programming!

Click here if you are not GalAnt

Click Refresh (or Reload) to run the script again cookie.html

```
<?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

```

cookie.html

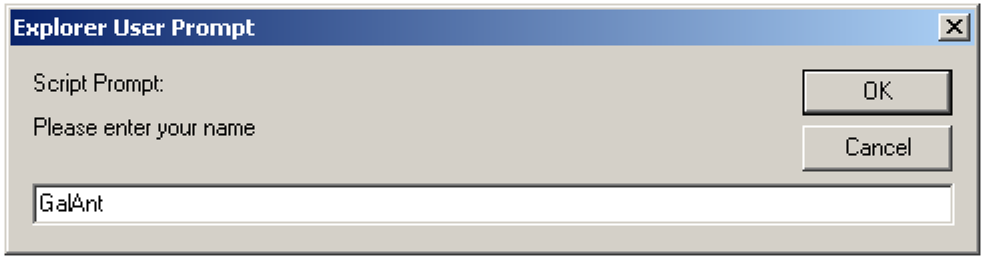
```
            // determine whether it is afternoon or evening
            if ( hour < 6 )
               document.write( "<h1>Good Afternoon, " );
            else
               document.write( "<h1>Good Evening, " );
         }

         // 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>
```

Final JavaScript Example

- Combines concepts discussed previously

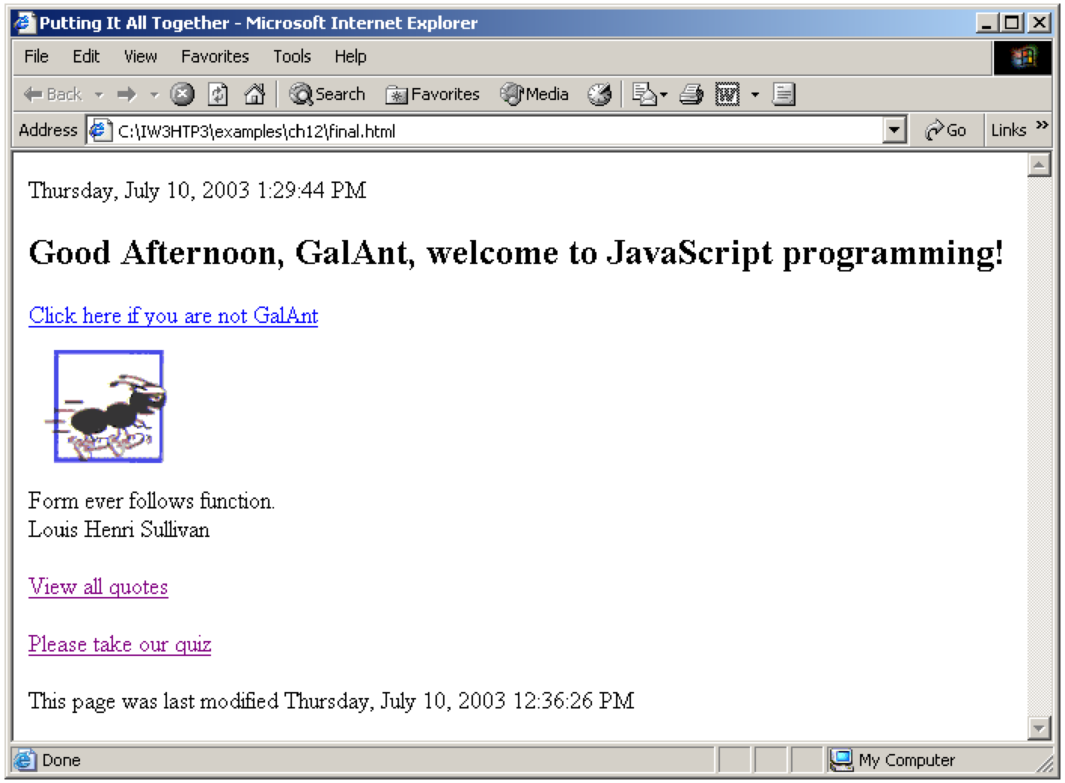
```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- 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!

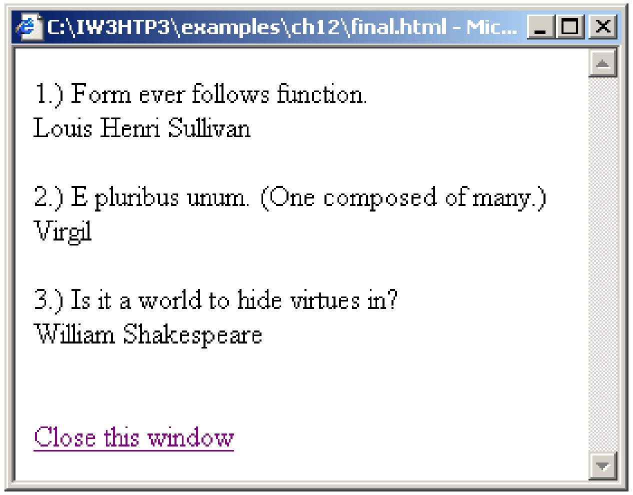
Click here if you are not GalAnt

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

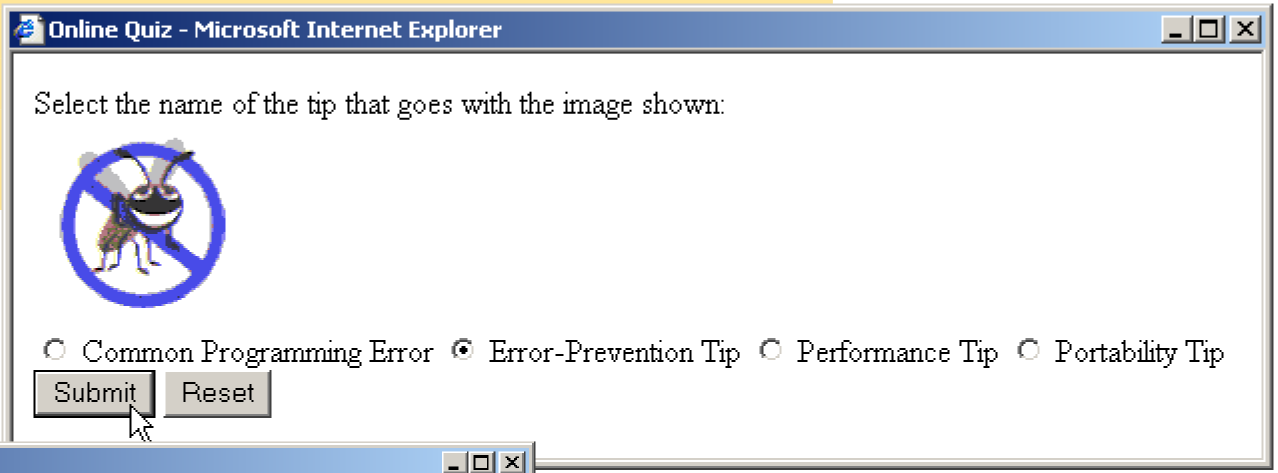
2.) E pluribus unum. (One composed of many.)
Virgil

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

Close this window

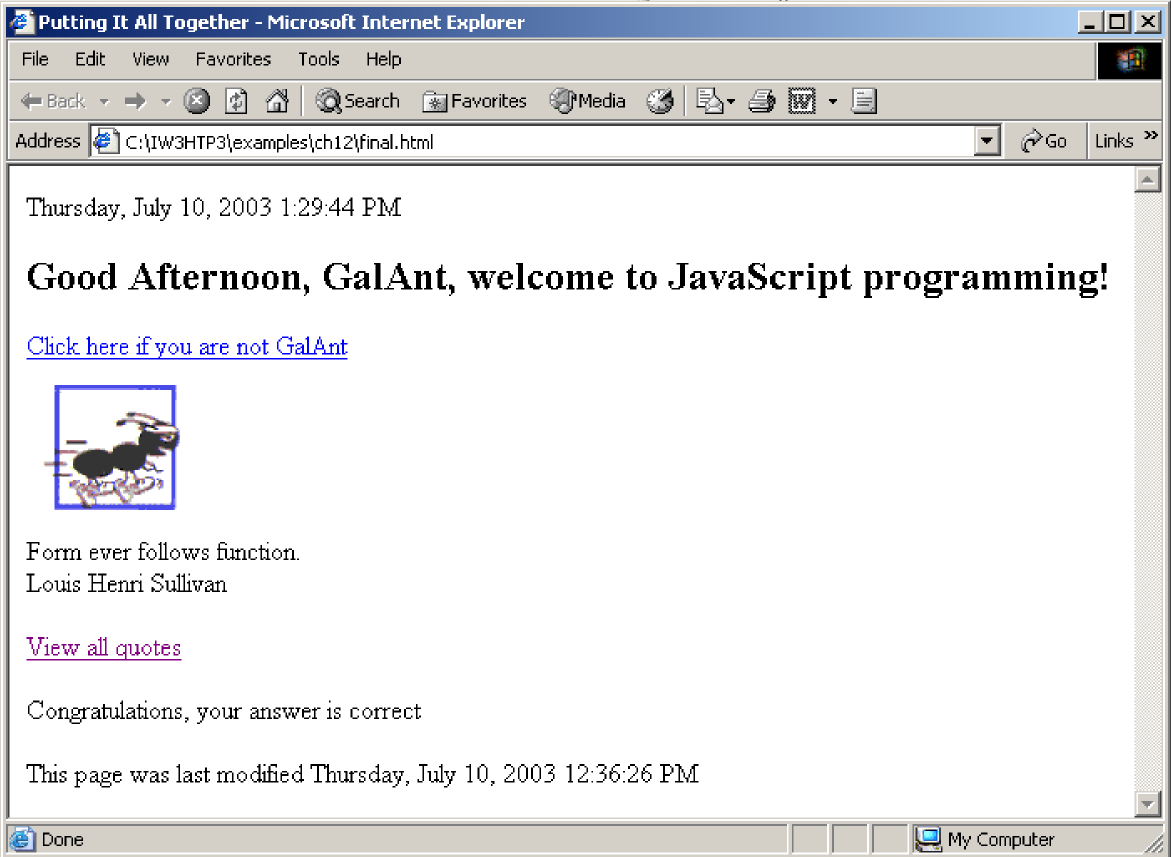
```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- 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
Go
Links
Done
My Computer

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

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

Storing data in the browser

LocalStorage, sessionStorage

- Web storage objects localStorage and sessionStorage allow to save key/value pairs in the browser.
- What's interesting about them is that the data survives a page refresh (for sessionStorage) and even a full browser restart (for localStorage). We'll see that very soon.
  - Unlike cookies, web storage objects are not sent to server with each request. Because of that, we can store much more. Most modern browsers allow at least 5 megabytes of data (or more) and have settings to configure that.
  - Also unlike cookies, the server can't manipulate storage objects via HTTP headers. Everything's done in JavaScript.
  - The storage is bound to the origin (domain/protocol/port triplet). That is, different protocols or subdomains infer different storage objects, they can't access data from each other.

Both storage objects provide the same methods and properties

- setItem(key, value) – store key/value pair.
- getItem(key) – get the value by key.
- removeItem(key) – remove the key with its value.
- clear() – delete everything.
- key(index) – get the key on a given position.
- length – the number of stored items.

localStorage

- Shared between all tabs and windows from the same origin.
- The data does not expire. It remains after the browser restart and even OS reboot.

```
localStorage.setItem('test', 1);
```

```
alert( localStorage.getItem('test') ); // 1
```

If they were any other type, like a number, or an object, they would get converted to a string automatically

Object-like access

```
// set key
localStorage.test = 2;

// get key
alert( localStorage.test ); // 2

// remove key
delete localStorage.test;
```

+
sessionStorage

- The sessionStorage object is used much less often than localStorage.
- Properties and methods are the same, but it's much more limited:
- The sessionStorage exists only within the current browser tab.
  - Another tab with the same page will have a different storage.
  - But it is shared between iframes in the same tab (assuming they come from the same origin).
- The data survives page refresh, but not closing/opening the tab.