



# Lecture 10 - Java Script

Adding Dynamic Content  
to Web Pages

(May 2019)

# Agenda

- **Generating HTML Dynamically**
- **Monitoring User Events**
- **Basic JavaScript Syntax**
- **Applications**
  - Using JavaScript to customize Web pages
  - Using JavaScript to make pages more dynamic
  - Using JavaScript to validate CGI forms
  - Using JavaScript to manipulate HTTP cookies
  - Using JavaScript to interact with and control frames
  - Controlling applets and calling Java from JavaScript
  - Accessing JavaScript from Java

# Generating HTML Dynamically

- **Idea**

- Script is interpreted as page is loaded, and uses `document.write` or `document.writeln` to insert HTML at the location the script occurs

- **Template**

```
...
```

```
<BODY>
```

```
Regular HTML
```

```
<SCRIPT TYPE="text/javascript">
```

```
<!--
```

```
Build HTML Here
```

```
// -->
```

```
</SCRIPT>
```

```
More Regular HTML
```

```
</BODY>
```

# A Simple Script

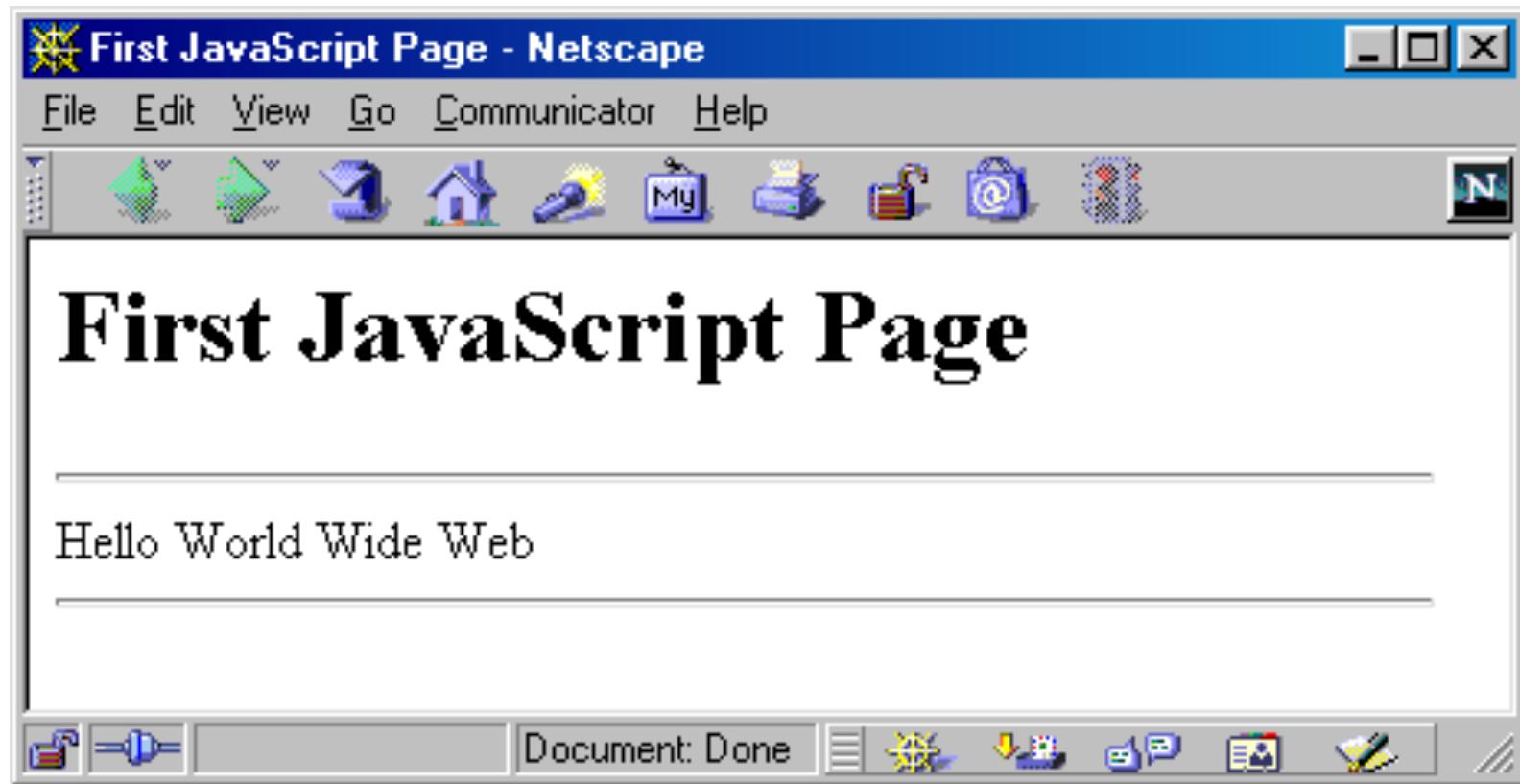
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
    <TITLE>First JavaScript Page</TITLE>
</HEAD>

<BODY>
<H1>First JavaScript Page</H1>

<SCRIPT TYPE="text/javascript">
<!--
document.write("<HR>") ;
document.write("Hello World Wide Web") ;
document.write("<HR>") ;
// -->
</SCRIPT>

</BODY>
</HTML>
```

# Simple Script, Result



# Extracting Document Info with JavaScript, Example

```
<HTML>
<HEAD>
    <TITLE>Extracting Document Info with
        JavaScript</TITLE>
</HEAD>
<BODY BGCOLOR="WHITE">
<H1>Extracting Document Info with JavaScript</H1>
<HR>

<SCRIPT TYPE="text/javascript">
<!--

function referringPage() {
    if (document.referrer.length == 0) {
        return("<I>none</I>") ;
    } else {
        return(document.referrer) ;
    }
}

-->
```

# Extracting Document Info with JavaScript, Example, cont.

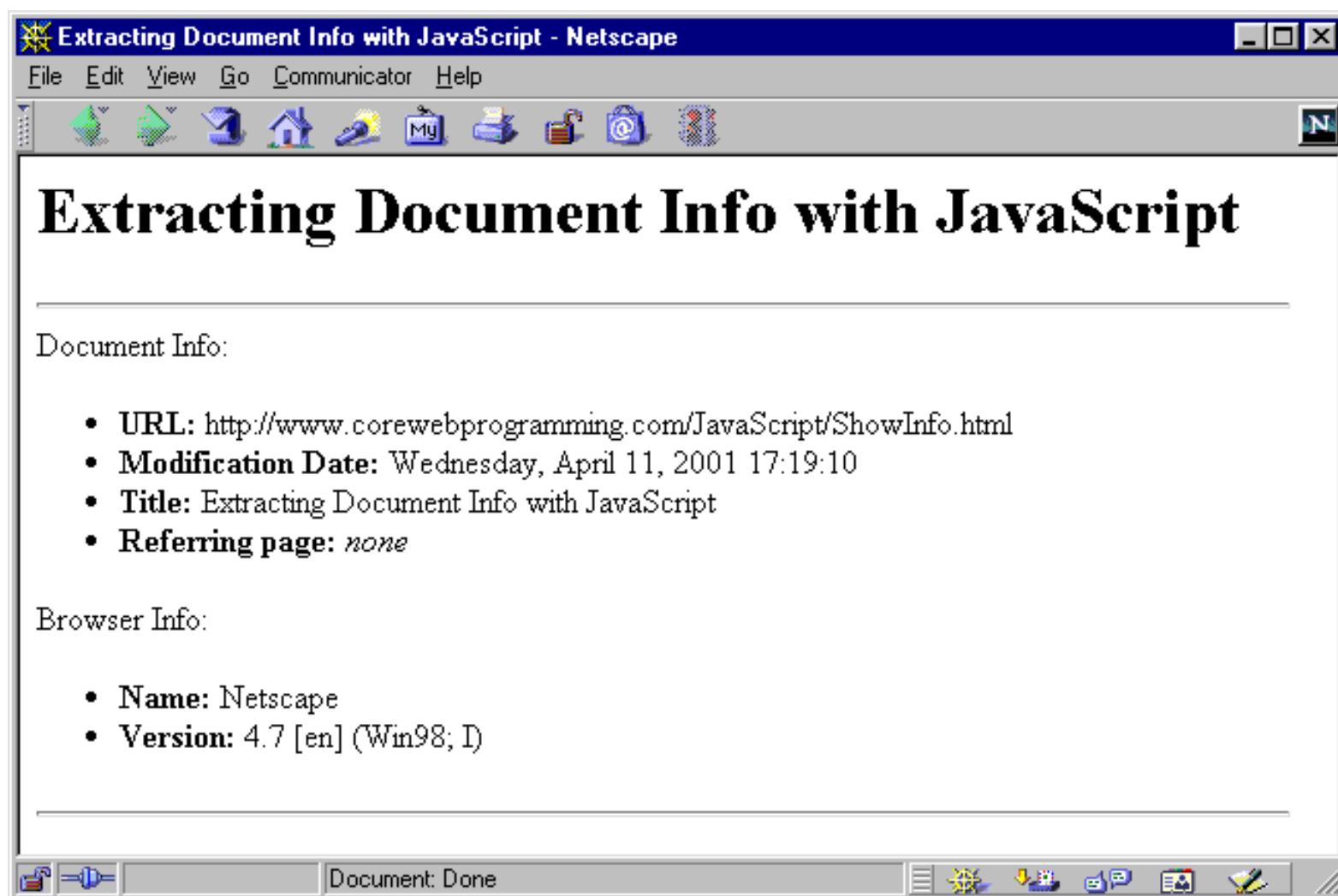
```
...
document.writeln
("Document Info:\n" +
"<UL>\n" +
"  <LI><B>URL:</B> " + document.location + "\n" +
"  <LI><B>Modification Date:</B> " + "\n" +
    document.lastModified + "\n" +
"  <LI><B>Title:</B> " + document.title + "\n" +
"  <LI><B>Referring page:</B> " + referringPage() + "\n" +
"</UL>");

document.writeln
("Browser Info:" + "\n" +
"<UL>" + "\n" +
"  <LI><B>Name:</B> " + navigator.appName + "\n" +
"  <LI><B>Version:</B> " + navigator.appVersion + "\n" +
"</UL>");

// -->
</SCRIPT>

<HR>
</BODY>
</HTML>
```

# Extracting Document Info with JavaScript, Result



# Extracting Document Info with JavaScript, Result



## Document Info:

- URL: <http://www.corewebprogramming.com/JavaScript>ShowInfo.html>
- Modification Date: 04/11/2001 17:19:10
- Title: Extracting Document Info with JavaScript
- Referring page: *none*

## Browser Info:

- Name: Microsoft Internet Explorer
- Version: 4.0 (compatible; MSIE 5.5; Windows 98)

# Multi-Browser Compatibility

## 1. Use Language Attribute

```
<SCRIPT LANGUAGE="JavaScript">
<!--
languageVersion = "1.0";
// -->
</SCRIPT>

<SCRIPT LANGUAGE="JavaScript1.1">
<!--
languageVersion = "1.1";
// -->
</SCRIPT>

...
<SCRIPT LANGUAGE="JavaScript1.5">
<!--
languageVersion = "1.5";
// -->
</SCRIPT>
```

**Note: Don't include that attribute TYPE="text/javascript"**

# Multi-Browser Compatibility, cont.

## 2. Use Vendor/Version Info

- `navigator.appName`
- `navigator.appVersion`

# Monitoring User Events

- **Use Various onXxx Attributes**
  - onClick
  - onLoad
  - onMouseOver
  - onFocus
  - etc.

# User Events, Example

```
<HTML>
<HEAD>
    <TITLE>Simple JavaScript Button</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
function dontClick() {
    alert("I told you not to click!");
}
// --
</SCRIPT>
</HEAD>

<BODY BGCOLOR="WHITE">
<H1>Simple JavaScript Button</H1>

<FORM>
    <INPUT TYPE="BUTTON"
        VALUE="Don't Click Me"
        onClick="dontClick()">
</FORM>
</BODY>
</HTML>
```

# User Events, Result



# JavaScript Syntax: Dynamic Typing

- **Idea**
  - Like Lisp, values are typed, not variables
  - A value is only checked for proper type when it is operated upon
- **Example**

```
var x = 5; // int
x = 5.5; // float
x = "five point five"; // String
```

# JavaScript Syntax: Function Declarations

## 1. Declaration Syntax

- Functions are declared using the **function** reserved word
- The return value is not declared, nor are the types of the arguments
- Examples:

```
function square(x) {  
    return(x * x);  
}  
  
function factorial(n) {  
    if (n <= 0) {  
        return(1);  
    } else {  
        return(n * factorial(n - 1));  
    }  
}
```

# JavaScript Syntax: Function Declarations, cont.

## 2. First Class Functions

- Functions can be passed and assigned to variables
- Example

```
var fun = Math.sin;  
alert("sin(pi/2)=" + fun(Math.PI/2));
```



# JavaScript Syntax: Objects and Classes

## 1. Fields Can Be Added On-the-Fly

- Adding a new property (field) is a simple matter of assigning a value to one
- If the field doesn't already exist when you try to assign to it, JavaScript will create it automatically.
- For instance:

```
var test = new Object();
test.field1 = "Value 1"; // Create field1
                        property
test.field2 = 7; // Create field2 property
```

# JavaScript Syntax: Objects and Classes, cont.

## 2. You Can Use Literal Notation

- You can create objects using a shorthand “literal” notation of the form

```
{ field1:val1, field2:val2, . . . , fieldN:valN }
```

- For example, the following gives equivalent values to object1 and object2

```
var object1 = new Object();
object1.x = 3;
object1.y = 4;
object1.z = 5;

object2 = { x:3, y:4, z:5 };
```

# JavaScript Syntax: Objects and Classes, cont.

## 3. The "for/in" Statement Iterates Over Properties

- JavaScript, unlike Java or C++, has a construct that lets you easily retrieve all of the fields of an object
- The basic format is as follows:

```
for(fieldName in object) {  
    doSomethingWith(fieldName);  
}
```

- Also, given a field name, you can access the field via `object["field"]` as well as via `object.field`

# Field Iteration, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
<HEAD>
<TITLE>For/In Loops</TITLE>

<SCRIPT TYPE="text/javascript">
<!--

function makeObjectTable(name, object) {
    document.writeln("<H2>" + name + "</H2>") ;
    document.writeln("<TABLE BORDER=1>\n" +
                    "  <TR><TH>Field<TH>Value") ;
    for(field in object) {
        document.writeln ("  <TR><TD>" + field +
                         "<TD>" + object[field]) ;
    }
    document.writeln("</TABLE>") ;
}
// -->
</SCRIPT>
```

# Field Iteration, Example

```
...
</HEAD>
<BODY BGCOLOR="WHITE">
<H1>For/In Loops</H1>

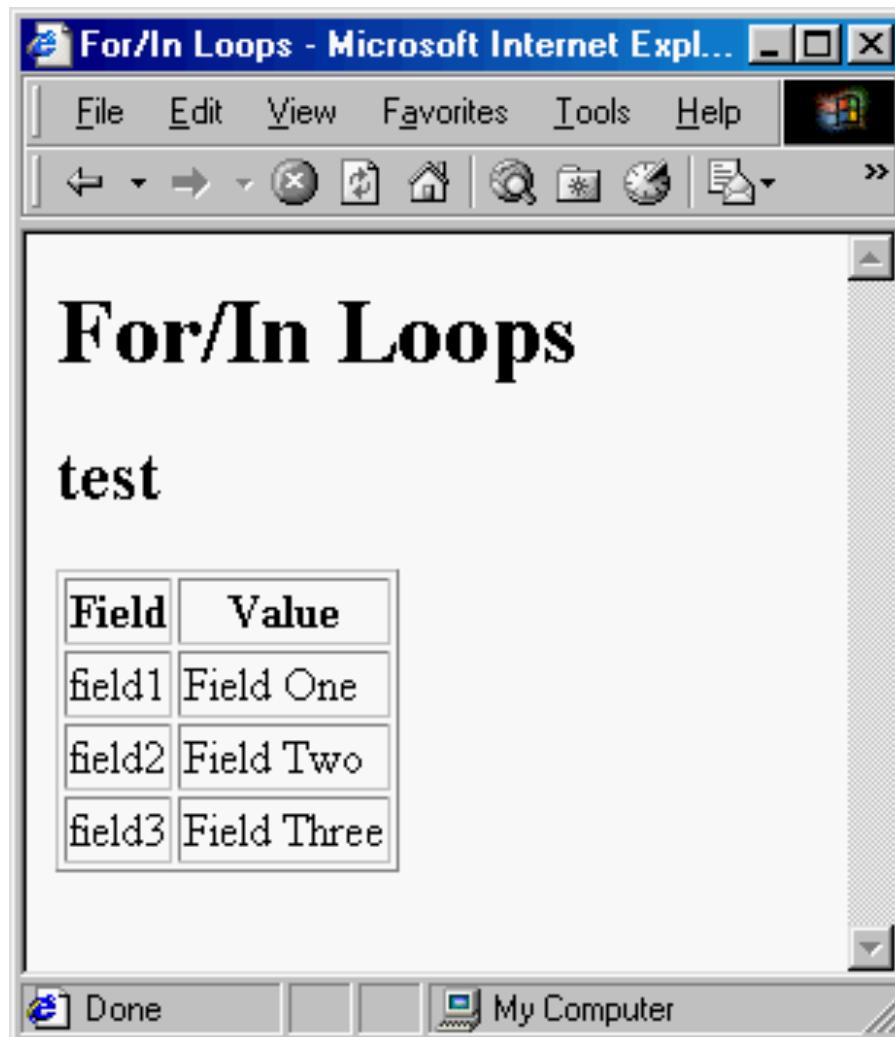
<SCRIPT TYPE="text/javascript">
<!--

var test = new Object();
test.field1 = "Field One";
test.field2 = "Field Two";
test.field3 = "Field Three";
makeObjectTable("test", test);

// -->
</SCRIPT>

</BODY>
</HTML>
```

# Field Iteration, Result



The for/in statement iterates over object properties

# JavaScript Syntax: Objects and Classes, cont.

## 4. A “Constructor” is Just a Function that Assigns to “this”

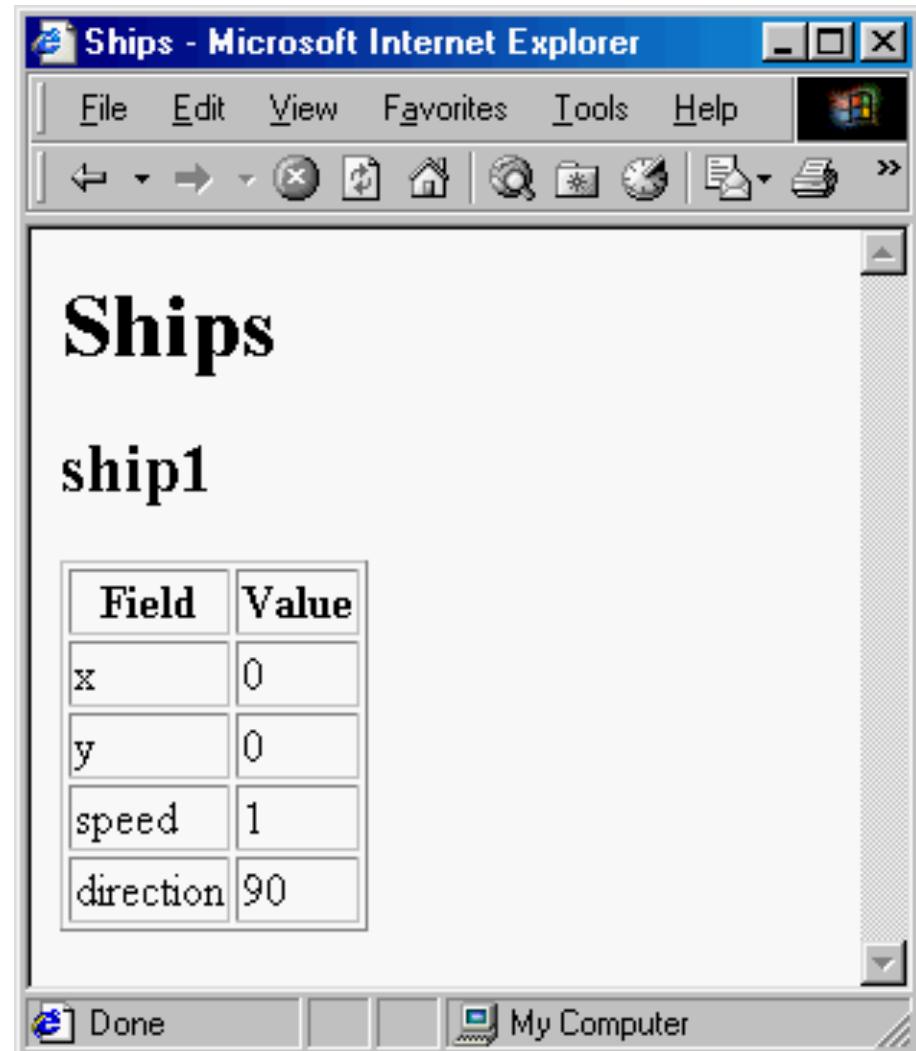
- JavaScript does not have an exact equivalent to Java's class definition
- The closest you get is when you define a function that assigns values to properties in the `this` reference
- Calling this function using `new` binds `this` to a new Object
- For example, following is a simple constructor for a `Ship` class

```
function Ship(x, y, speed, direction) {  
    this.x = x;  
    this.y = y;  
    this.speed = speed;  
    this.direction = direction;
```

}

# Constructor, Example

```
var ship1 =  
    new Ship(0, 0, 1, 90);  
makeObjectTable("ship1", ship1);
```



# JavaScript Syntax: Objects and Classes, cont.

## 5. Methods Are Function-Valued Properties

- No special syntax for defining methods of objects
- Instead, you simply assign a function to a property

# Class Methods, Example

- Consider a version of the Ship class that includes a move method

```
function degreesToRadians(degrees) {  
    return(degrees * Math.PI / 180.0);  
}  
  
function move() {  
    var angle = degreesToRadians(this.direction);  
    this.x = this.x + this.speed * Math.cos(angle);  
    this.y = this.y + this.speed * Math.sin(angle);  
}  
  
function Ship(x, y, speed, direction) {  
    this.x = x;  
    this.y = y;  
    this.speed = speed;  
    this.direction = direction;  
    this.move = move;  
}
```

# Class Methods, Result

```
var ship1 = new Ship(0, 0, 1, 90);
makeObjectTable("ship1 (originally)", ship1);
ship1.move();
makeObjectTable("ship1 (after move)", ship1);
```

The screenshot shows a Microsoft Internet Explorer window titled "Ships - Microsoft Internet Explorer". The main content area displays the heading "Ships" and a section titled "ship1 (originally)". Below this is a table showing the initial state of the ship's properties:

Field	Value
x	0
y	0
speed	1
direction	90
move	function move() { var angle = degreesToRadians(this.direction); this.x = this.x + this.speed * Math.cos(angle); this.y = this.y + this.speed * Math.sin(angle); }

The screenshot shows a Microsoft Internet Explorer window titled "Ships - Microsoft Internet Explorer". The main content area displays the heading "ship1 (after move)". Below this is a table showing the state of the ship's properties after the move method was called:

Field	Value
x	6.123031769111886e-17
y	1
speed	1
direction	90
move	function move() { var angle = degreesToRadians(this.direction); this.x = this.x + this.speed * Math.cos(angle); this.y = this.y + this.speed * Math.sin(angle); }

# JavaScript Syntax: Objects and Classes, cont.

## 5. Arrays

- For the most part, you can use arrays in JavaScript a lot like Java arrays.
  - Here are a few examples:

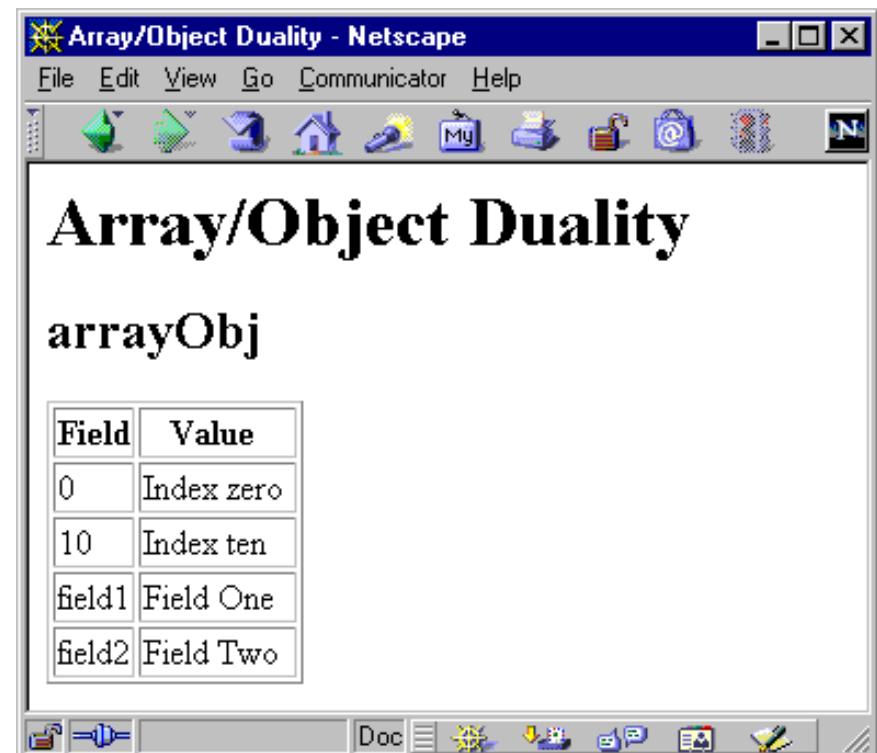
```
var squares = new Array(5);
for(var i=0; i<squares.length; i++) {
    vals[i] = i * i;
}
// Or, in one fell swoop:
var squares = new Array(0, 1, 4, 9, 16);
var array1 = new Array("fee", "fie", "fo",
"fum");
// Literal Array notation for creating an array.
var array2 = [ "fee", "fie", "fo", "fum" ];
```

- Behind the scenes, however, JavaScript simply represents arrays as objects with numbered fields
  - You can access named fields using either `object.field` or `object["field"]`, but numbered fields only via `object[fieldNumber]`

# Array, Example

```
var arrayObj = new Object();
arrayObj[0] = "Index zero";
arrayObj[10] = "Index ten";
arrayObj.field1 = "Field One";
arrayObj["field2"] = "Field Two";

makeObjectTable("arrayObj",
                arrayObj);
```



# Application: Adjusting to the Browser Window Size

- **Netscape 4.0 introduced the `window.innerWidth` and `window.innerHeight` properties**
  - Lets you determine the usable size of the current browser window

# Determining Browser Size, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
    <TITLE>Strawberries</TITLE>

<SCRIPT TYPE="text/javascript">
<!--
function image(url, width, height) {
    return ('<IMG SRC="' + url + '"'
        ' WIDTH=' + width +
        ' HEIGHT=' + height + '>');
}

function strawberry1(width) {
    return(image("Strawberry1.gif", width,
    Math.round(width*1.323)));
}

function strawberry2(width) {
    return(image("Strawberry2.gif", width,
    Math.round(width*1.155)));
}
// -->
</SCRIPT>
```

# Determining Browser Size, Example, cont.

```
...
<SCRIPT TYPE="text/javascript">
<!--
  var imageWidth = window.innerWidth/4;
  var fontSize = Math.min(7,
    Math.round(window.innerWidth/100)) ;

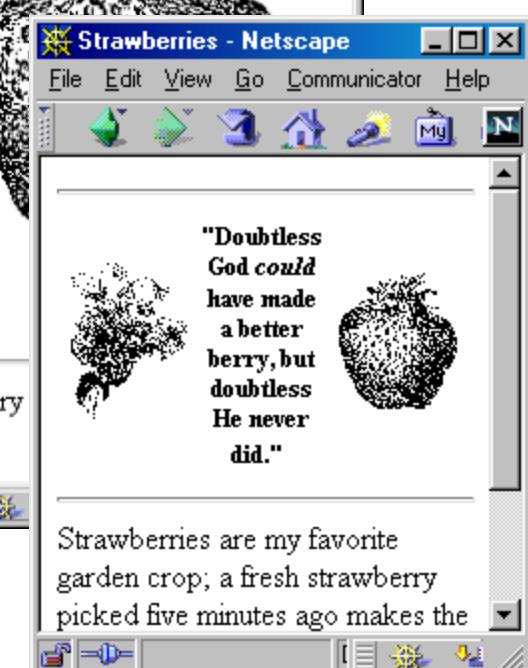
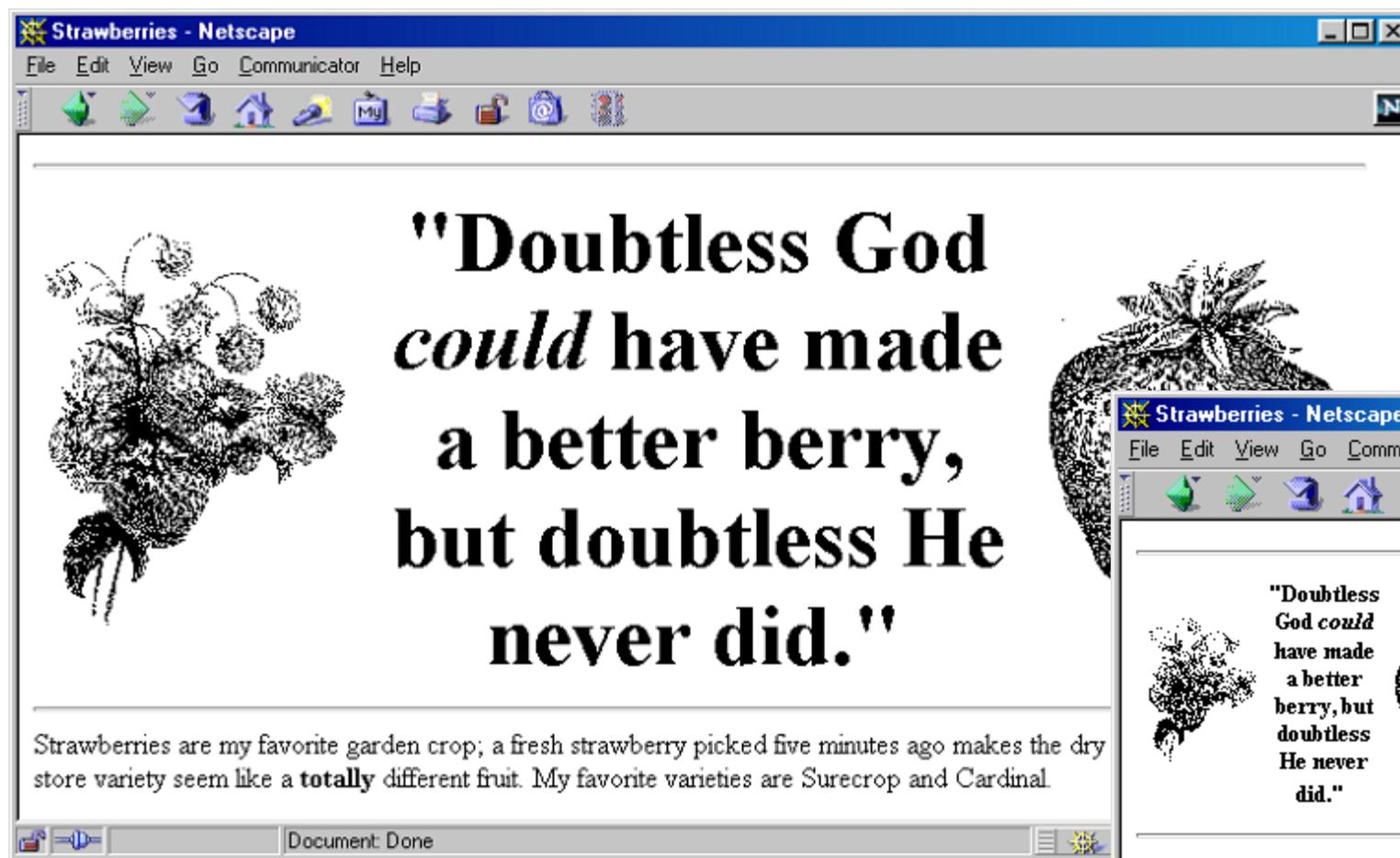
document.writeln
('<TABLE>\n' +
 '  <TR><TD>' + strawberry1(imageWidth) + '\n' +
 '    <TH><FONT SIZE=' + fontSize + '>\n' +
 '      "Doubtless God <I>could</I> have made\n' +
 '      a better berry, but doubtless He\n' +
 '      never did."</FONT>\n' +
 '    <TD>' + strawberry2(imageWidth) + '\n' +
 '  </TR>\n' +
 '</TABLE>');
// -->
</SCRIPT>
<HR>
```

Strawberries are my favorite garden crop; a fresh ...

```
</BODY>
```

```
</HTML>
```

# Determining Browser Size, Results



# Application: Using JavaScript to Make Pages Dynamic

- **Modifying Images Dynamically**
  - The document.images property contains an array of Image objects corresponding to each IMG element in the current document
  - To display a new image, simply set the SRC property of an existing image to a string representing a different image file

# Modifying Images, Example

- The following function changes the first image in a document

```
function changeImage() {  
    document.images[0].src = "images/new-image.gif";  
}
```

- Referring to images by name is easier:

```
<IMG SRC="cool-image.jpg" NAME="cool"  
WIDTH=75 HEIGHT=25>
```

```
function improveImage() {  
    document.images["cool"].src = "way-cool.jpg";  
}
```

# Modifying Images: A Clickable Image Button, Example

```
<SCRIPT TYPE="text/javascript">
<!--
imageFiles = new Array("images/Button1-Up.gif",
                      "images/Button1-Down.gif",
                      "images/Button2-Up.gif",
                      "images/Button2-Down.gif");
imageObjects = new Array(imageFiles.length);
for(var i=0; i<imageFiles.length; i++) {
  imageObjects[i] = new Image(150, 25);
  imageObjects[i].src = imageFiles[i];
}

function setImage(name, image) {
  document.images[name].src = image;
}
```

# Modifying Images: A Clickable Image Button, Example

```
function clickButton(name, grayImage) {  
    var origImage = document.images[name].src;  
    setImage(name, grayImage);  
    var resetString =  
        "setImage('" + name + "' , '" + origImage + "');  
    setTimeout(resetString, 100);  
}  
// -->  
</SCRIPT>  
  
</HEAD>  
...  
<A HREF="location1.html"  
    onClick="clickButton('Button1', 'images/Button1-  
    Down.gif')">  
<IMG SRC="images/Button1-Up.gif" NAME="Button1"  
    WIDTH=150 HEIGHT=25></A>  
  
<A HREF="location2.html"  
    onClick="clickButton('Button2', 'images/Button2-  
    Down.gif')">  
<IMG SRC="images/Button2-Up.gif" NAME="Button2"  
    WIDTH=150 HEIGHT=25></A>
```

# Highlighting Images Under the Mouse, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0  
Transitional//EN">  
<HTML>  
<HEAD>  
  <TITLE>High Peaks Navigation Bar</TITLE>  
  
<SCRIPT TYPE="text/javascript">  
<!--  
  
// Given "Foo", returns "images/Foo.gif".  
function regularImageFile(imageName) {  
  return("images/" + imageName + ".gif");  
}  
  
// Given "Bar", returns "images/Bar-Negative.gif".  
function negativeImageFile(imageName) {  
  return("images/" + imageName + "-Negative.gif");  
}
```

# Highlighting Images Under the Mouse, Example, cont.

```
// Cache image at specified index. E.g., given index 0,  
// take imageNames[0] to get "Home". Then preload  
// images/Home.gif and images/Home-Negative.gif.  
  
function cacheImages(index) {  
    regularImageObjects[index] = new Image(150, 25);  
    regularImageObjects[index].src =  
        regularImageFile(imageNames[index]);  
    negativeImageObjects[index] = new Image(150, 25);  
    negativeImageObjects[index].src =  
        negativeImageFile(imageNames[index]);  
}  
  
imageNames = new Array("Home", "Tibet", "Nepal",  
                      "Austria", "Switzerland");  
regularImageObjects = new Array(imageNames.length);  
negativeImageObjects = new Array(imageNames.length);  
  
// Put images in cache for fast highlighting.  
for(var i=0; i<imageNames.length; i++) {  
    cacheImages(i);  
}
```

# Highlighting Images Under the Mouse, Example, cont.

```
...
function highlight(imageName) {
    document.images[imageName].src =
        negativeImageFile(imageName);
}

function unHighlight(imageName) {
    document.images[imageName].src =
        regularImageFile(imageName);
}
// -->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="WHITE">
<TABLE BORDER=0 WIDTH=150 BGCOLOR="WHITE"
       CELLPADDING=0 CELLSPACING=0>
<TR><TD><A HREF="Tibet.html"
              TARGET="Main"
              onMouseOver="highlight('Tibet')"
              onMouseOut="unHighlight('Tibet')">
              <IMG SRC="images/Tibet.gif"
                   NAME="Tibet"
                   WIDTH=150 HEIGHT=25 BORDER=0>
            </A>
```

# Highlighting Images Under the Mouse, Result

High Peaks Travel Inc. - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites History

**Home**  
**TIBET**  
NEPAL  
AUSTRIA  
SWITZERLAND

**High Peaks Travel Inc.**

Tired of the same old vacations in Cleveland? Tour the high peaks with **High Peaks Travel!**

We have package deals for beginner, experienced, and expert climbers, discount priced (\*) for the budget-conscious traveller.

HPT is currently arranging trips to the following exciting locations:

[Tibet](#)  
[Nepal](#)  
[Austria](#)  
[Switzerland](#)

Sign up today!

(\*) No ropes or safety equipment provided on discount tours.

My Computer

High Peaks Travel Inc. - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites History

**Home**  
**TIBET**  
NEPAL  
AUSTRIA  
SWITZERLAND

**High Peaks Travel Inc.**

Tired of the same old vacations in Cleveland? Tour the high peaks with **High Peaks Travel!**

We have package deals for beginner, experienced, and expert climbers, discount priced (\*) for the budget-conscious traveller.

HPT is currently arranging trips to the following exciting locations:

[Tibet](#)  
[Nepal](#)  
[Austria](#)  
[Switzerland](#)

Sign up today!

(\*) No ropes or safety equipment provided on discount tours.

file:///C:/CWP2/Chapter24/highpeaks/Tibet.html

My Computer

# Making Pages Dynamic: Moving Layers

- Netscape 4 introduced “layers” – regions that can overlap and be positioned arbitrarily
- JavaScript 1.2 lets you access layers via the `document.layers` array, each element of which is a `Layer` object with properties corresponding to the attributes of the `LAYER` element
- A named layer can be accessed via `document.layers["layer name"]` rather than by using an index, or simply by using `document.layerName`

# Moving Layers, Example

- Descriptive overlays slowly “drift” to final spot when button clicked

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Camps on K-3</TITLE>

<SCRIPT TYPE="text/javascript">
<!--
function hideCamps() {
  // Netscape 4 document model.
  document.layers["baseCamp"].visibility = "hidden";
  document.layers["highCamp"].visibility = "hidden";
  // Or document.baseCamp.visibility = "hidden";
}

function moveBaseCamp() {
  baseCamp.moveBy(1, 3);
  if (baseCamp.pageX < 130) {
    setTimeout("moveBaseCamp()", 10);
  }
}
```

# Moving Layers, Example, cont.

```
function showBaseCamp () {
    hideCamps () ;
    baseCamp = document.layers["baseCamp"] ;
    baseCamp.moveToAbsolute(0, 20) ;
    baseCamp.visibility = "show" ;
    moveBaseCamp () ;
}
function moveHighCamp () {
    highCamp.moveBy(2, 1) ;
    if (highCamp.pageX < 110) {
        setTimeout("moveHighCamp()", 10) ;
    }
}

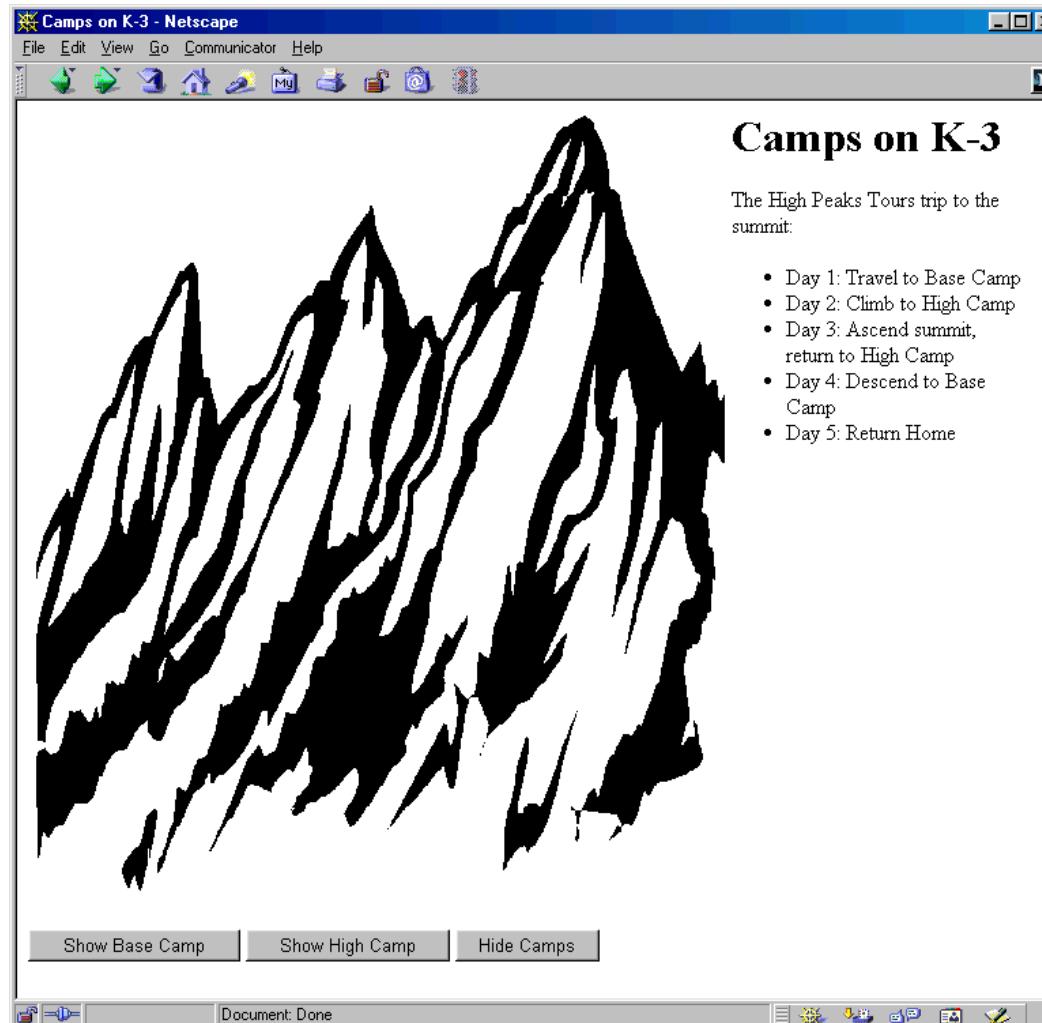
function showHighCamp () {
    hideCamps () ;
    highCamp = document.layers["highCamp"] ;
    highCamp.moveToAbsolute(0, 65) ;
    highCamp.visibility = "show" ;
    moveHighCamp () ;
}
// -->
</SCRIPT>
```

# Moving Layers, Example, cont.

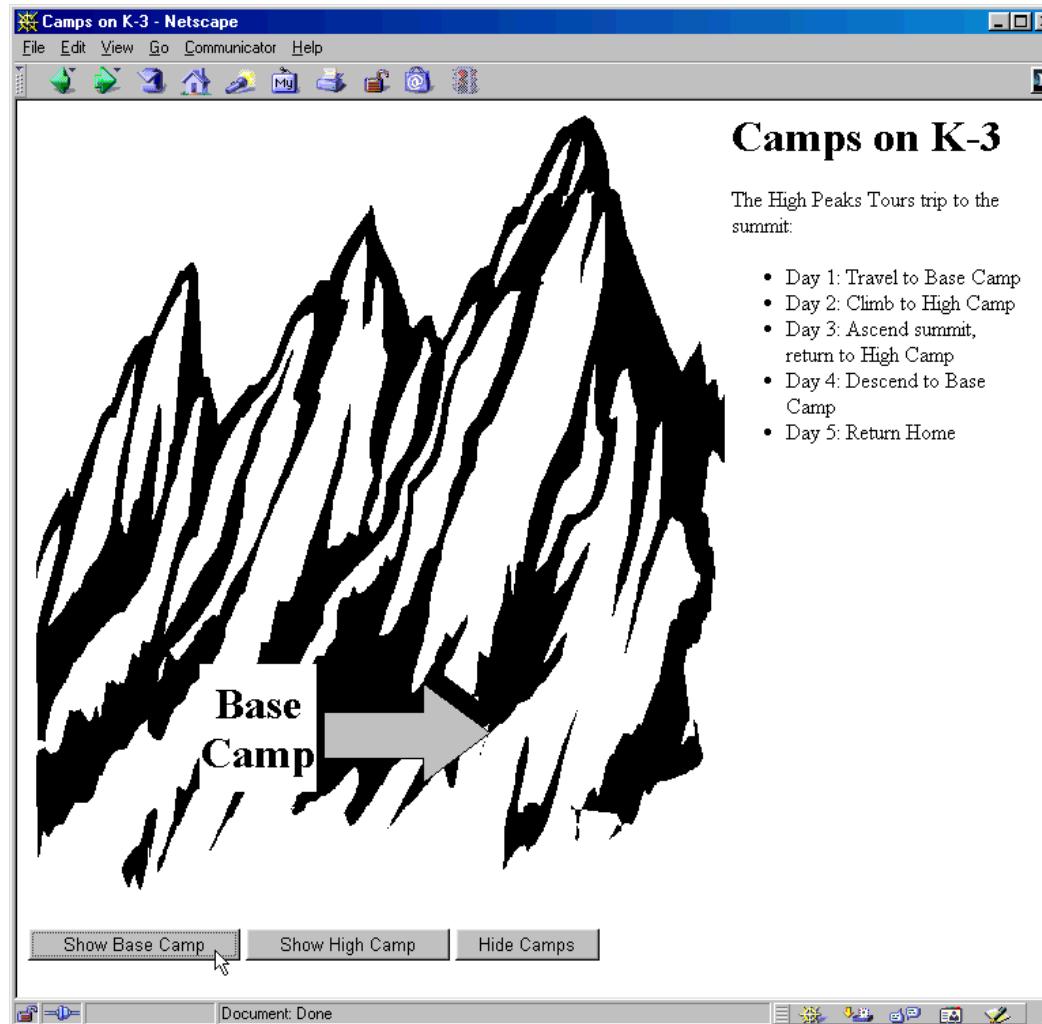
```
<LAYER ID="highCamp" PAGEX=50 PAGEY=100 VISIBILITY="hidden">
  <TABLE>
    <TR><TH BGCOLOR="WHITE" WIDTH=50>
      <FONT SIZE="+2">High Camp</FONT>
      <TD><IMG SRC="images/Arrow-Right.gif">
    </TABLE>
  </LAYER>
<LAYER ID="baseCamp" PAGEX=50 PAGEY=100 VISIBILITY="hidden">
  <TABLE>
    <TR><TH BGCOLOR="WHITE" WIDTH=50>
      <FONT SIZE="+2">Base Camp</FONT>
      <TD><IMG SRC="images/Arrow-Right.gif">
    </TABLE>
  </LAYER>

<FORM>
  <INPUT TYPE="Button" VALUE="Show Base Camp"
         onClick="showBaseCamp () ">
  <INPUT TYPE="Button" VALUE="Show High Camp"
         onClick="showHighCamp () ">
  <INPUT TYPE="Button" VALUE="Hide Camps"
         onClick="hideCamps () ">
</FORM>
```

# Moving Layers, Result



# Moving Layers, Result



# Application: Using JavaScript to Validate CGI Forms

## 1. Accessing Forms

- The `document.forms` property contains an array of `Form` entries contained in the document
- As usual in JavaScript, named entries can be accessed via name instead of by number, plus named forms are automatically inserted as properties in the `document` object, so any of the following formats would be legal to access forms

```
var firstForm = document.forms[0];
// Assumes <FORM NAME="orders" ...>
var orderForm = document.forms["orders"];
// Assumes <FORM NAME="register" ...>
var registrationForm = document.register;
```

# Application: Using JavaScript to Validate CGI Forms, cont.

## 2. Accessing Elements within Forms

- The Form object contains an elements property that holds an array of Element objects
- You can retrieve form elements by number, by name from the array, or via the property name:

```
var firstElement = firstForm.elements[0];
// Assumes <INPUT ... NAME="quantity">
var quantityField =
    orderForm.elements["quantity"];
// Assumes <INPUT ... NAME="submitSchedule">
var submitButton = register.submitSchedule;
```

# Checking Form Values Individually, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
    <TITLE>On-Line Training</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
...
// When the user changes and leaves textfield, check
// that a valid choice was entered. If not, alert
// user, clear field, and set focus back there.

function checkLanguage() {
    // or document.forms["langForm"].elements["langField"]
    var field = document.langForm.langField;
    var lang = field.value;
    var prefix = lang.substring(0, 4).toUpperCase();
    if (prefix != "JAVA") {
        alert("Sorry, '" + lang + "' is not valid.\n" +
              "Please try again.");
        field.value = ""; // Erase old value
        field.focus(); // Give keyboard focus
    }
}
```

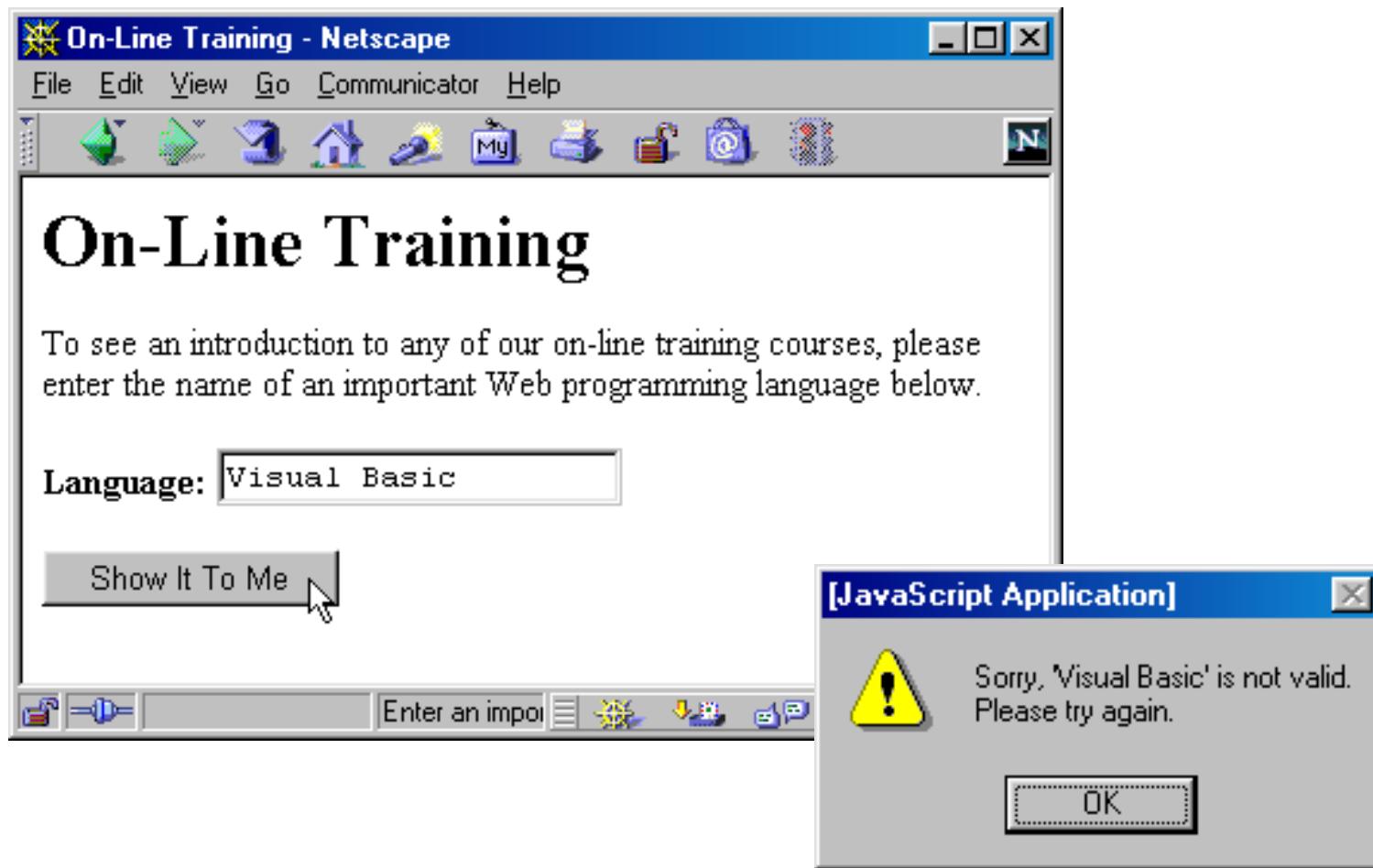
# Checking Form Values Individually, Example, cont.

```
// -->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="WHITE">
<H1>On-Line Training</H1>

<FORM ACTION="cgi-bin/registerLanguage" NAME="langForm">
To see an introduction to any of our on-line training
courses, please enter the name of an important Web
programming language below.
<P>
<B>Language:</B>
<INPUT TYPE="TEXT" NAME="langField"
       onFocus="describeLanguage()"
       onBlur="clearStatus()"
       onChange="checkLanguage()">
<P>
<INPUT TYPE="SUBMIT" VALUE="Show It To Me">
</FORM>

</BODY>
</HTML>
```

# Checking Form Values Individually, Results



# Checking Values When Form is Submitted, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
    <TITLE>Camp Registration</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
function isInt(string) {
    var val = parseInt(string);
    return(val > 0);
}

function checkRegistration() {
    var ageField = document.registerForm.ageField;
    if (!isInt(ageField.value)) {
        alert("Age must be an integer.");
        return(false);
    }
    ...
    // Format looks OK. Submit form.
    return(true);
}
// -->
</SCRIPT>
```

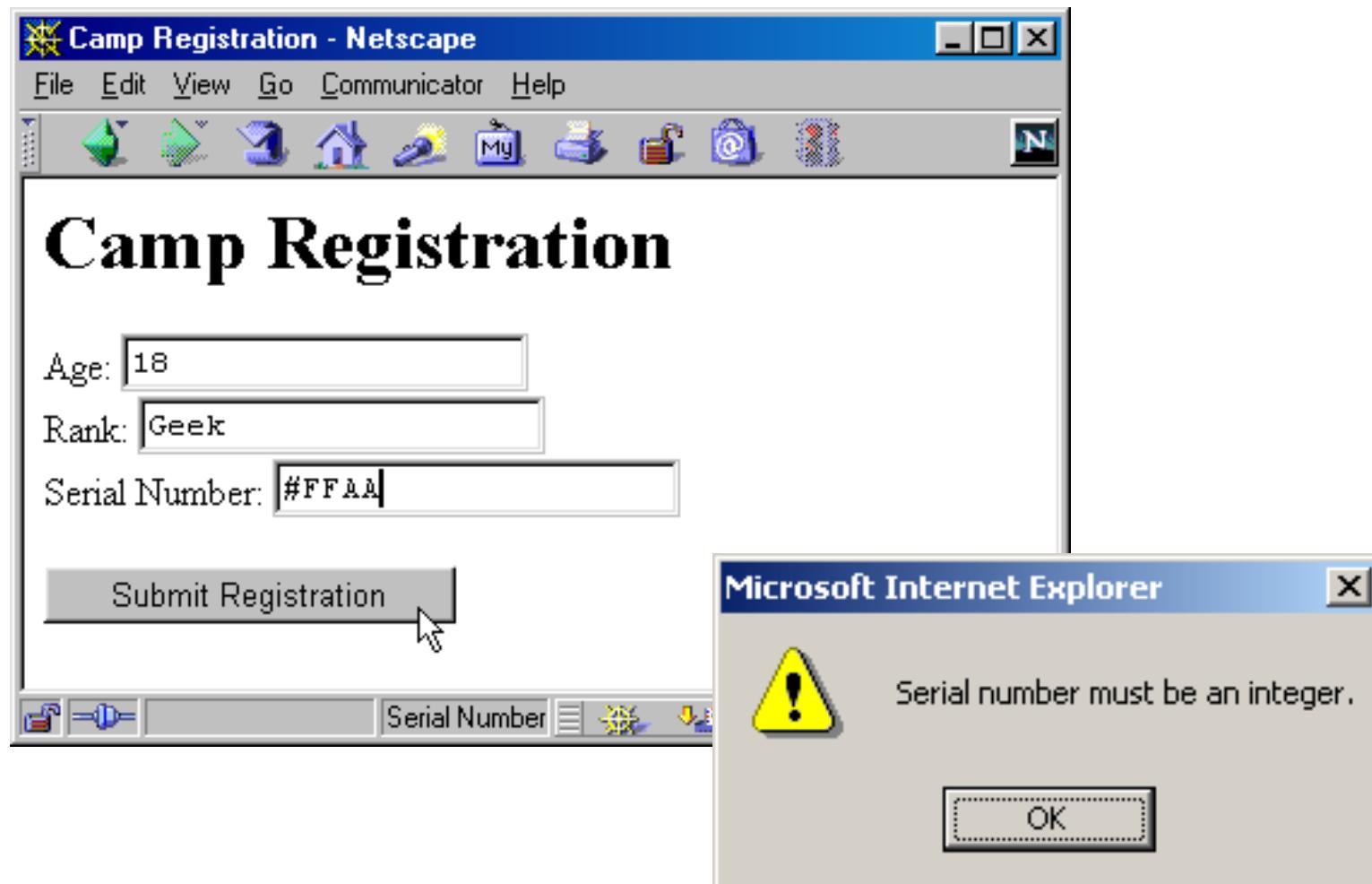
# Checking Values When Form is Submitted, Example, cont.

```
<BODY BGCOLOR="WHITE">
<H1>Camp Registration</H1>

<FORM ACTION="cgi-bin/register"
      NAME="registerForm"
      onSubmit="return (checkRegistration ())">
Age: <INPUT TYPE="TEXT" NAME="ageField"
        onFocus="promptAge () "
        onBlur="clearStatus () ">
<BR>
Rank: <INPUT TYPE="TEXT" NAME="rankField"
        onFocus="promptRank () "
        onBlur="clearStatus () ">
<BR>
Serial Number: <INPUT TYPE="TEXT" NAME="serialField"
                onFocus="promptSerial () "
                onBlur="clearStatus () ">
<P>
<INPUT TYPE="SUBMIT" VALUE="Submit Registration">
</FORM>

</BODY>
</HTML>
```

# Checking Values When Form is Submitted, Results



# Application: Using JavaScript to Store and Examine Cookies

## 1. Using document.cookies

- Set it (one cookie at a time) to store values

```
document.cookie = "name1=val1";  
document.cookie = "name2=val2; expires=" +  
someDate;  
document.cookie = "name3=val3; path=/;  
domain=test.com";
```

- Read it (all cookies in a single string) to access values

# Application: Using JavaScript to Store and Examine Cookies

## 2. Parsing Cookies

```
function cookieVal(cookieName, cookieString) {  
    var startLoc =  
        cookieString.indexOf(cookieName);  
    if (startLoc == -1) {  
        return(""); // No such cookie  
    }  
    var sepLoc = cookieString.indexOf("=",  
        startLoc);  
    var endLoc = cookieString.indexOf(";",  
        startLoc);  
    if (endLoc == -1) { // Last one has no ";"  
        endLoc = cookieString.length;  
    }  
    return(cookieString.substring(sepLoc+1,  
        endLoc));
```

# Exercise

- **Using JavaScript, write a page to solve equation (ex: quadratic equation) in which it can get input from user, validate input and display result.**

# Cookie, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
    <TITLE>Widgets "R" Us</TITLE>
<SCRIPT TYPE="text/javascript">
<!--

function storeCookies() {
    var expires = "; expires=Monday, 01-Dec-01 23:59:59 GMT";
    var first = document.widgetForm.firstField.value;
    var last = document.widgetForm.lastField.value;
    var account = document.widgetForm.accountField.value;
    document.cookie = "first=" + first + expires;
    document.cookie = "last=" + last + expires;
    document.cookie = "account=" + account + expires;
}

// Store cookies and give user confirmation.
function registerAccount() {
    storeCookies();
    alert("Registration Successful.");
}
```

# Cookie, Example, cont.

```
function cookieVal(cookieName, cookieString) {  
    var startLoc = cookieString.indexOf(cookieName);  
    if (startLoc == -1) {  
        return(""); // No such cookie  
    }  
    var sepLoc = cookieString.indexOf("=", startLoc);  
    var endLoc = cookieString.indexOf(";", startLoc);  
    if (endLoc == -1) { // Last one has no ";"  
        endLoc = cookieString.length;  
    }  
    return(cookieString.substring(sepLoc+1, endLoc));  
}  
  
function presetValues() {  
    var firstField = document.widgetForm.firstField;  
    var lastField = document.widgetForm.lastField;  
    var accountField = document.widgetForm.accountField;  
    var cookies = document.cookie;  
    firstField.value = cookieVal("first", cookies);  
    lastField.value = cookieVal("last", cookies);  
    accountField.value = cookieVal("account", cookies);  
}  
// -->  
</SCRIPT>
```

# Cookie, Example, cont.

```
</HEAD>
<BODY BGCOLOR="WHITE" onLoad="presetValues() ">

<H1>Widgets "R" Us</H1>

<FORM ACTION="servlet/cwp.Widgets"
      NAME="widgetForm"
      onSubmit="storeCookies() ">
First Name: <INPUT TYPE="TEXT" NAME="firstField">
<BR>
Last Name: <INPUT TYPE="TEXT" NAME="lastField">
<BR>
Account Number: <INPUT TYPE="TEXT" NAME="accountField">
<BR>
Widget Name: <INPUT TYPE="TEXT" NAME="widgetField">
<BR>
<INPUT TYPE="BUTTON" VALUE="Register Account"
       onClick="registerAccount() ">
<INPUT TYPE="SUBMIT" VALUE="Submit Order">

</FORM>
</BODY>
</HTML>
```

# Cookie, Example, Result

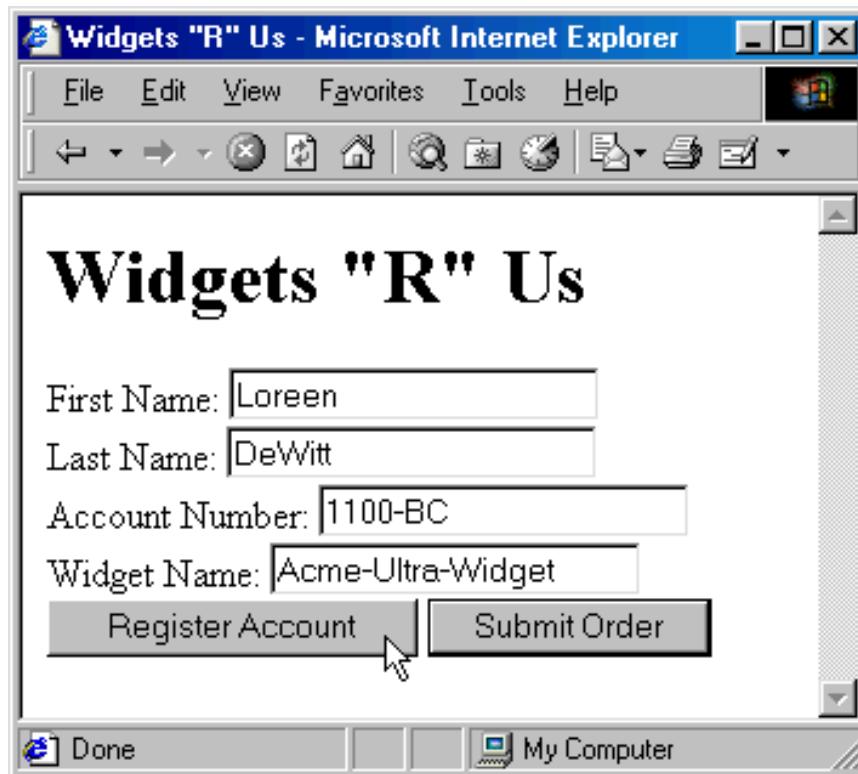
Widgets "R" Us - Microsoft Internet Explorer

File Edit View Favorites Tools Help

First Name: Loreen  
Last Name: DeWitt  
Account Number: 1100-BC  
Widget Name: Acme-Ultra-Widget

Register Account      Submit Order

Done My Computer



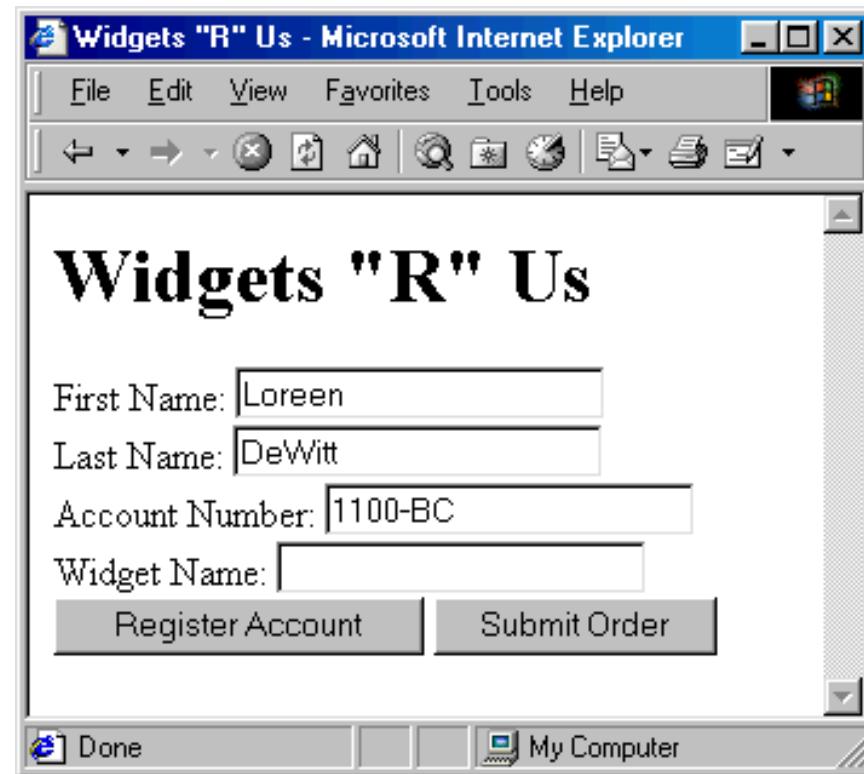
Widgets "R" Us - Microsoft Internet Explorer

File Edit View Favorites Tools Help

First Name: Loreen  
Last Name: DeWitt  
Account Number: 1100-BC  
Widget Name:

Register Account      Submit Order

Done My Computer



# Application: Using JavaScript to Interact with Frames

- **Idea**
  - The default `Window` object contains a `frames` property holding an array of frames (other `Window` objects) contained by the current window or frame.
    - It also has `parent` and `top` properties referring to the directly enclosing frame or window and the top-level window, respectively.
    - All of the properties of `Window` can be applied to any of these entries.

# Displaying a URL in a Particular Frame, Example

- ShowURL.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Frameset//EN">

<HTML>
<HEAD>
    <TITLE>Show a URL</TITLE>
</HEAD>

<FRAMESET ROWS="150, *">
    <FRAME SRC="GetURL.html" NAME="inputFrame">
    <FRAME SRC="DisplayURL.html" NAME="displayFrame">
</FRAMESET>

</HTML>
```

# Displaying a URL in a Particular Frame, Example, cont.

- GetURL.html

```
<HTML>
<HEAD>
    <TITLE>Choose a URL</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
function showURL() {
    var url = document.urlForm.urlField.value;
    // or parent.frames["displayFrame"].location = url;
    parent.displayFrame.location = url;
}

function preloadUrl() {
    if (navigator.appName == "Netscape") {
        document.urlForm.urlField.value =
            "http://home.netscape.com/";
    } else {
        document.urlForm.urlField.value =
            "http://www.microsoft.com/";
    }
}
...

```

# Displaying a URL in a Particular Frame, Example, cont.

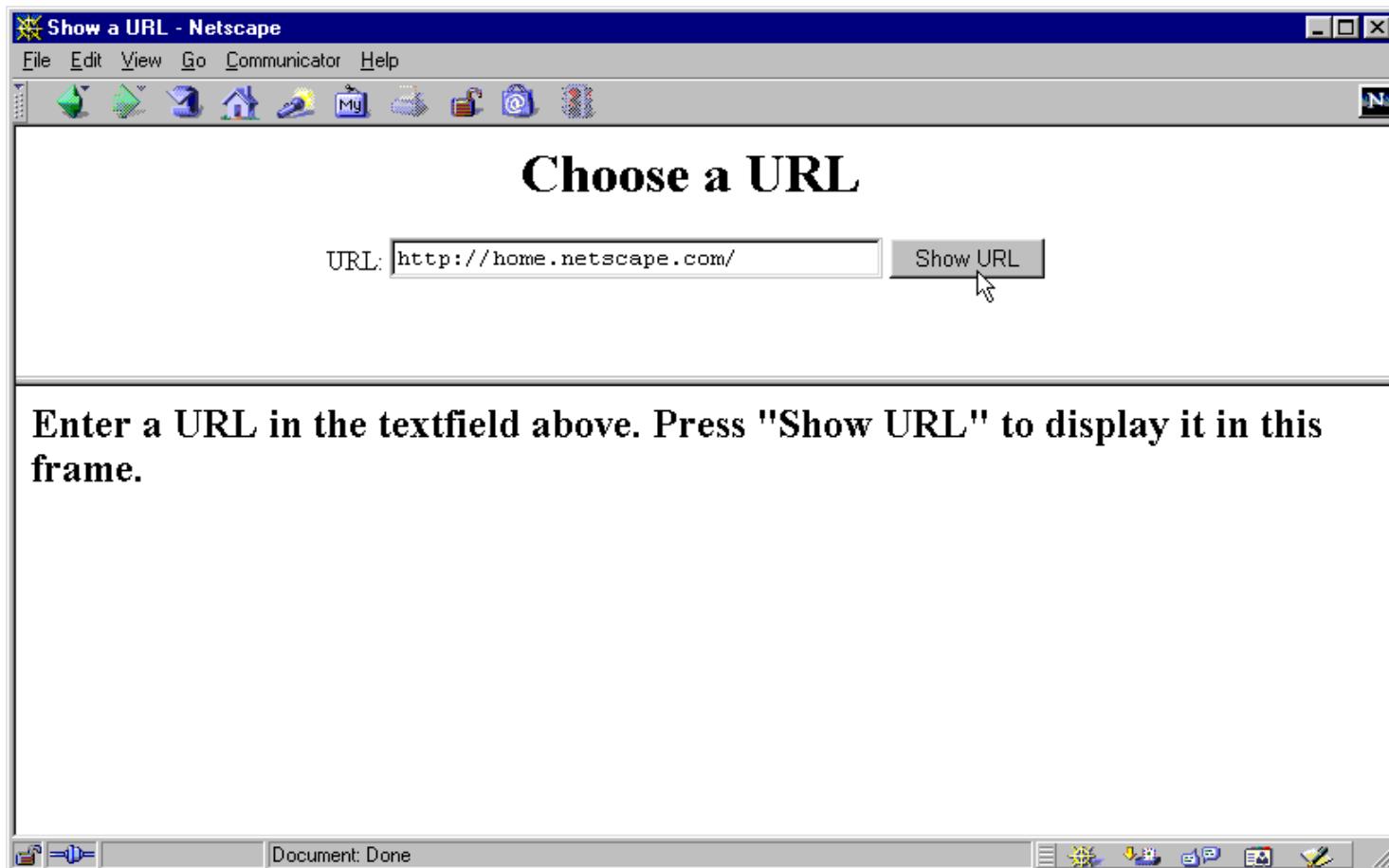
- GetURL.html, cont.

```
<BODY BGCOLOR="WHITE" onLoad="preloadUrl () ">
<H1 ALIGN="CENTER">Choose a URL</H1>

<CENTER>
<FORM NAME="urlForm">
URL: <INPUT TYPE="TEXT" NAME="urlField" SIZE=35>
<INPUT TYPE="BUTTON" VALUE="Show URL"
      onClick="showURL () ">
</FORM>
</CENTER>

</BODY>
</HTML>
```

# Displaying a URL in a Particular Frame, Result



# Displaying a URL in a Particular Frame, Result, cont.



# Giving a Frame the Input Focus, Example

- If JavaScript is manipulating the frames, the fix is easy: just add a call to focus in showUrl:

```
function showURL() {  
    var url = document.urlForm.urlField.value;  
    parent.displayFrame.location = url;  
    // Give frame the input focus  
    parent.displayFrame.focus();  
}
```

- Fixing the problem in regular HTML documents is a bit more tedious
  - Requires adding onClick handlers that call focus to each and every occurrence of A and AREA that includes a TARGET, and a similar onSubmit handler to each FORM that uses TARGET

# Application: Accessing Java from JavaScript

## 1. Idea

- Netscape 3.0 introduced a package called LiveConnect that allows JavaScript to talk to Java and vice versa
- Applications:
  - Calling Java methods directly.
    - In particular, this section shows how to print debugging messages to the Java console
  - Using applets to perform operations for JavaScript
    - In particular, this section shows how a hidden applet can be used to obtain the client hostname, information not otherwise available to JavaScript
  - Controlling applets from JavaScript
    - In particular, this section shows how LiveConnect allows user actions in the HTML part of the page to trigger actions in

# Application: Accessing Java from JavaScript

- **Calling Java Methods Directly**
  - JavaScript can access Java variables and methods simply by using the fully qualified name. For instance:

```
java.lang.System.out.println("Hello  
Console");
```
  - Limitations:
    - Can't perform operations forbidden to applets
    - No try/catch, so can't call methods that throw exceptions
    - Cannot write methods or create subclasses

# Controlling Applets from JavaScript, Example

- MoldSimulation.html, cont.

```
<BODY BGCOLOR="#C0C0C0">
<H1>Mold Propagation Simulation</H1>

<APPLET CODE="RandomCircles.class" WIDTH=100 HEIGHT=75>
</APPLET>
<P>
<APPLET CODE="RandomCircles.class" WIDTH=300 HEIGHT=75>
</APPLET>
<P>
<APPLET CODE="RandomCircles.class" WIDTH=500 HEIGHT=75>
</APPLET>

<FORM>
<INPUT TYPE="BUTTON" VALUE="Start Simulations"
       onClick="startCircles()">
<INPUT TYPE="BUTTON" VALUE="Stop Simulations"
       onClick="stopCircles()">
</FORM>

</BODY>
</HTML>
```

# Controlling Applets from JavaScript, Example

- **MoldSimulation.html**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
    <TITLE>Mold Propagation Simulation</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
function startCircles() {
    for(var i=0; i<document.applets.length; i++) {
        document.applets[i].startCircles();
    }
}

function stopCircles() {
    for(var i=0; i<document.applets.length; i++) {
        document.applets[i].stopCircles();
    }
}
// -->
</SCRIPT>
</HEAD>
```

# Controlling Applets from JavaScript, Example

- **RandomCircles.java**

```
public class RandomCircles extends Applet
    implements Runnable {
    private boolean drawCircles = false;

    public void startCircles() {
        Thread t = new Thread(this);
        t.start();
    }

    public void run() {
        Color[] colors = { Color.lightGray, Color.gray,
                           Color.darkGray, Color.black };
        int colorIndex = 0;
        int x, y;
        int width = getSize().width;
        int height = getSize().height;

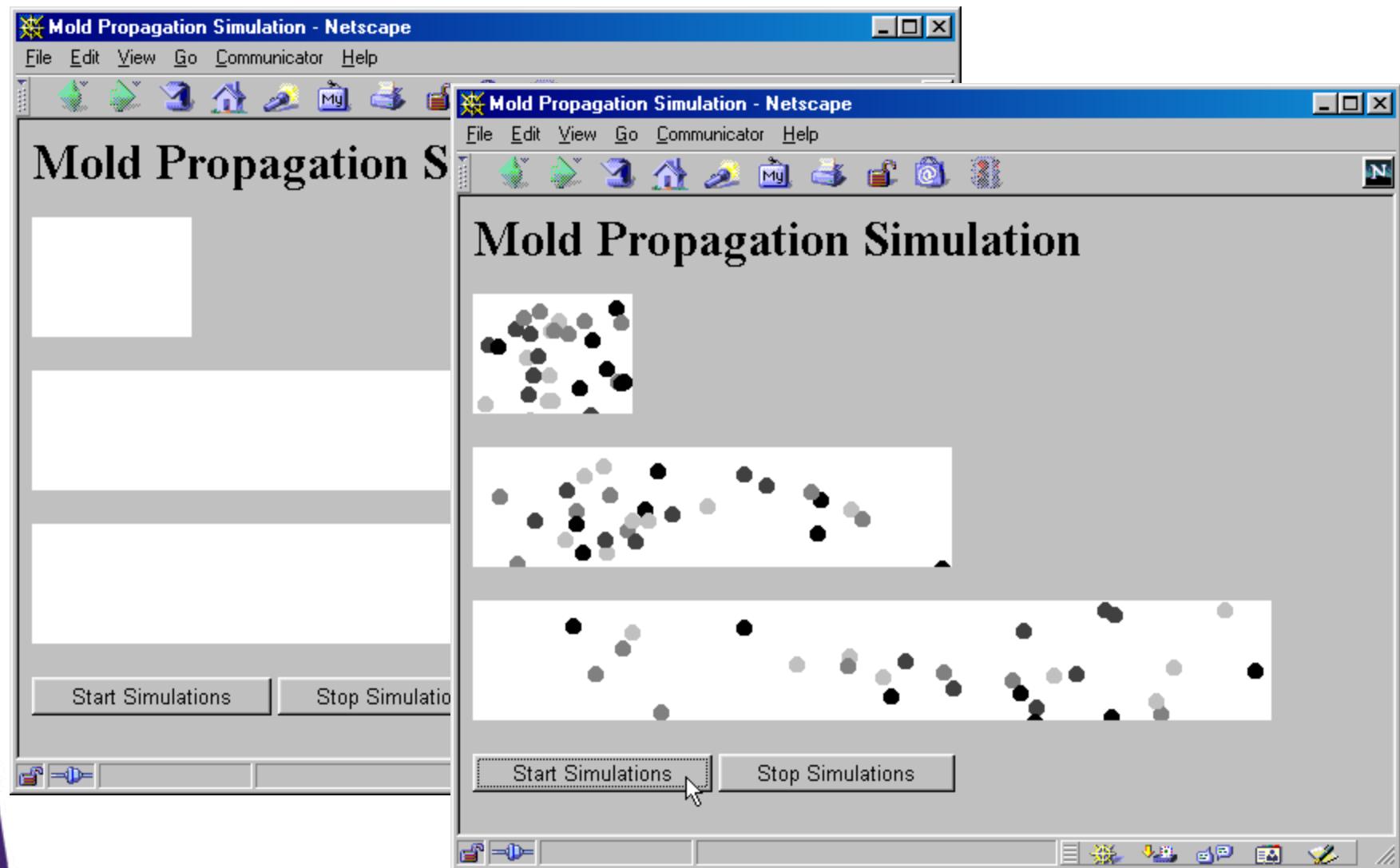
        Graphics g = getGraphics();
        drawCircles = true;
        ...
    }
}
```

# Controlling Applets from JavaScript, Example

- RandomCircles.java, cont.

```
while(drawCircles) {  
    x = (int)Math.round(width * Math.random());  
    y = (int)Math.round(height * Math.random());  
    g.setColor(colors[colorIndex]);  
    colorIndex = (colorIndex + 1) % colors.length;  
    g.fillOval(x, y, 10, 10);  
    pause(0.1);  
}  
}  
  
public void stopCircles() {  
    drawCircles = false;  
}  
  
private void pause(double seconds) {  
    try {  
        Thread.sleep((int)(Math.round(seconds * 1000.0)));  
    } catch(InterruptedException ie) {}  
}  
}
```

# Controlling Applets from JavaScript, Results



# Accessing JavaScript from Java

- **Steps**

1. Obtain and install the JSObject class
  - Installed with Netscape 4 (`javar40.jar`)
  - JDK 1.4 includes JSObject in `jaws.jar`
    - See Chapter 24 in  
[http://java.sun.com/j2se/1.4.1/docs/guide/plugin/developer\\_guide/contents.html](http://java.sun.com/j2se/1.4.1/docs/guide/plugin/developer_guide/contents.html)

2. Import it in your applet

```
import netscape.javascript.JSObject
```

3. From the applet, obtain a JavaScript reference to the current window

```
JSObject window = JSObject.getWindow(this);
```

# Accessing JavaScript from Java, cont.

- **Steps, cont.**

4. Read the JavaScript properties of interest

- Use `getMember` to access properties of the `JSObject`

```
JSObject someForm =  
  
    (JSObject) document.getMember("someFormName");
```

5. Set the JavaScript properties of interest

- Use `setMember` to set properties of the `JSObject`

```
document.setMember("bgColor", "red");
```

6. Call the JavaScript methods of interest

```
String[] message = { "An alert message" };  
window.call("alert", message);  
window.eval("alert('An alert message')");
```

7. Give the applet permission to access its Web page

```
<APPLET CODE=... WIDTH=... HEIGHT=...  
MAYSCRIPT>
```

# Matching Applet Background with Web Page, Example

- MatchColor.java

```
import java.applet.Applet;
import java.awt.*;
import netscape.javascript.JSObject;

public class MatchColor extends Applet {
    public void init() {
        JSObject window = JSObject.getWindow(this);
        JSObject document =
            (JSObject)window.getMember("document");
        // E.g., "#ff0000" for red
        String pageColor =
            (String)document.getMember("bgColor");
        // E.g., parseInt("ff0000", 16) --> 16711680
        int bgColor =
            Integer.parseInt(pageColor.substring(1, 7),
16);
        setBackground(new Color(bgColor));
    }
}
```

# Matching Applet Background with Web Page, Example, cont.

- MatchColor.html

```
<HTML>
<HEAD>
    <TITLE>MatchColor</TITLE>
</HEAD>
<BODY BGCOLOR="RED">
<H1>MatchColor</H1>
<APPLET CODE="MatchColor.class"
        WIDTH=300 HEIGHT=300 MAYSCRIPT>
</APPLET>
</BODY>
</HTML>
```

# Applet That Controls HTML Form Values, Example

- See on-line example for `Everest.html`



# Summary

- **JavaScript permits you to:**
  - Customize Web pages based on the situation
  - Make pages more dynamic
  - Validate HTML form input
  - Manipulate cookies
  - Control frames
  - Integrate Java and JavaScript
- **Refer chapter 9: Vietnamese textbook**