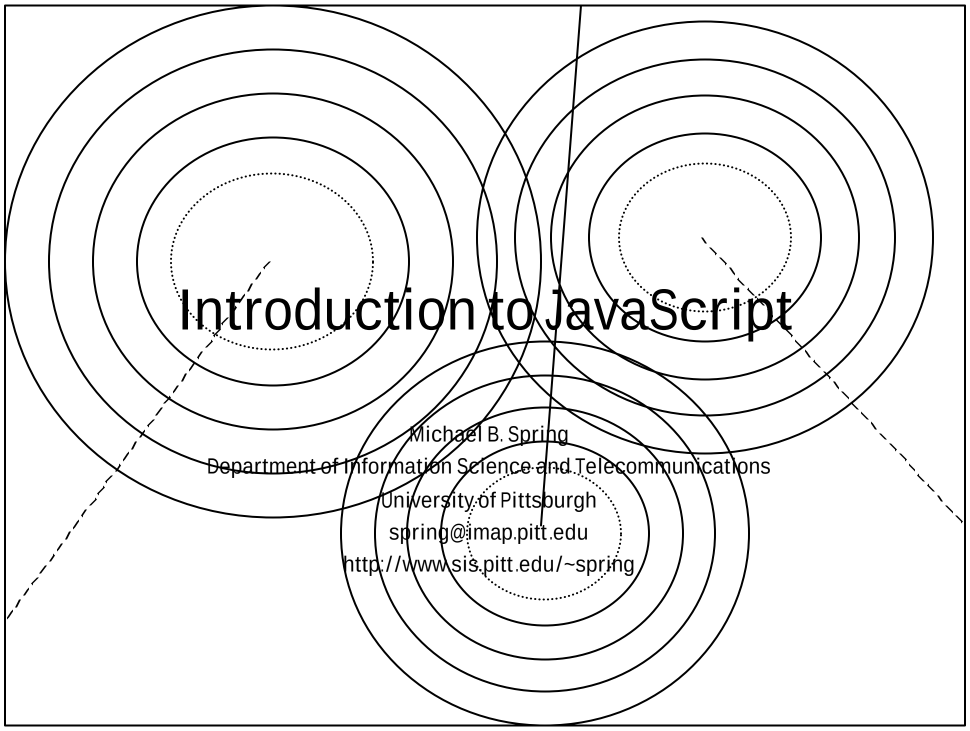
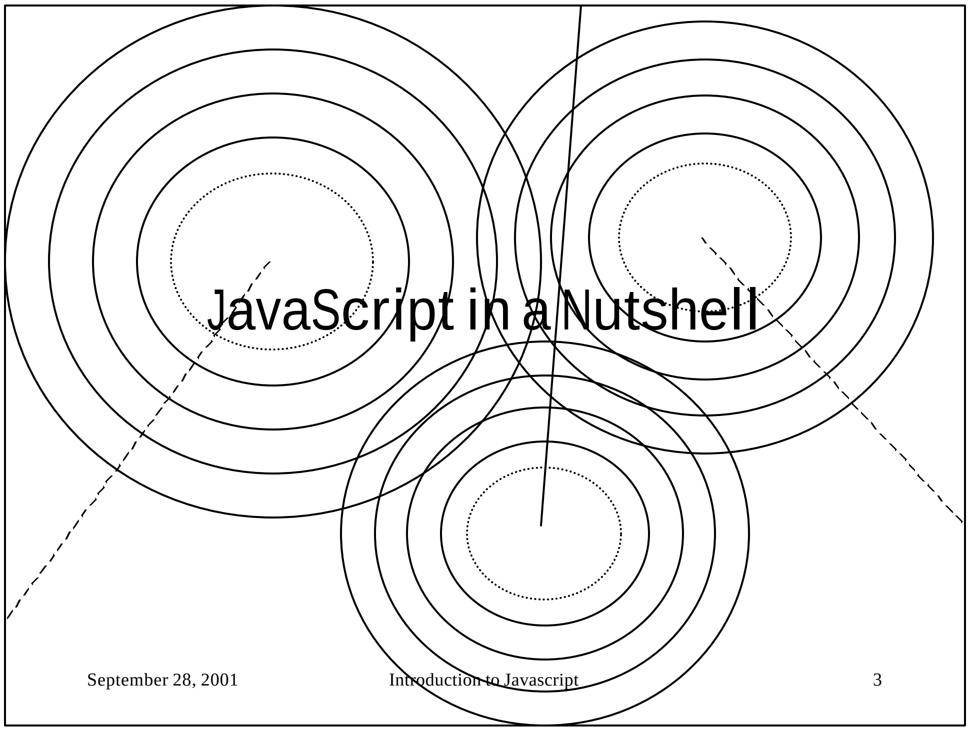


Introduction to JavaScript

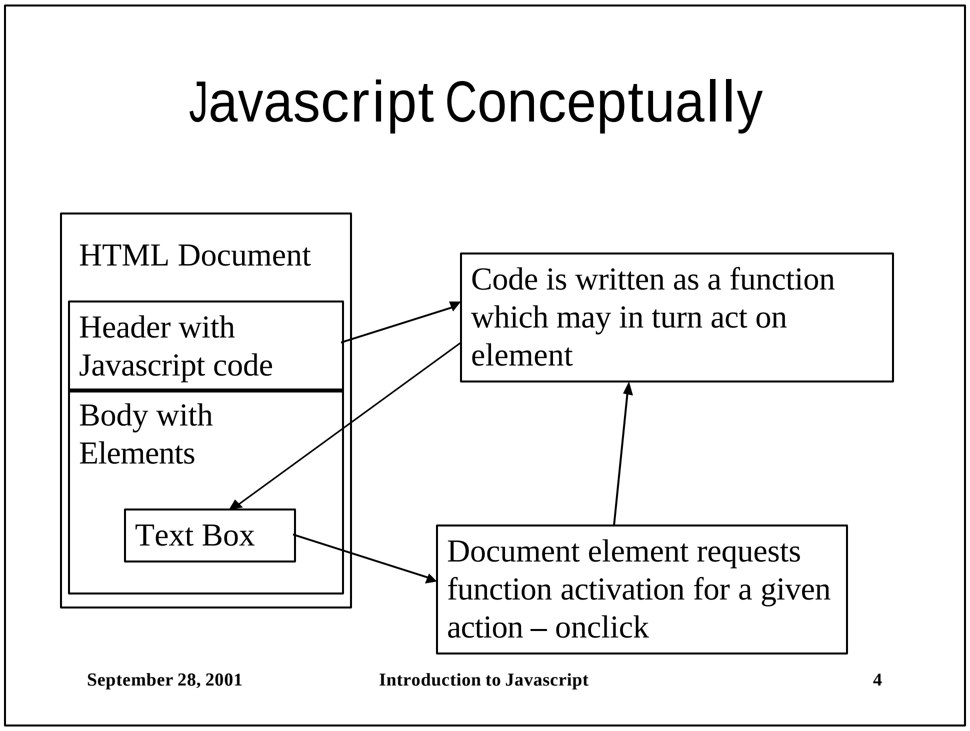
Michael B. Spring
Department of Information Science and Telecommunications
University of Pittsburgh
spring@imap.pitt.edu
http://www.sis.pitt.edu/~spring

Overview

- Javascript in a nutshell
  - Javascript conceptually
  - What it is and isn't
- Basics
  - Data types
  - Expressions and operators
  - Control structures
- Client side program structure
- Javascript objects and events
- Javascript and forms
  - Form Validation
  - Dynamic menus



JavaScript in a Nutshell



Javascript Conceptually

HTML Document

Header with Javascript code

Body with Elements

Text Box

Code is written as a function which may in turn act on element

Document element requests function activation for a given action – onclick

What Javascript Is and Is Not

- JavaScript is
  - an interpreted loosely-typed object-based language
  - event driven, embedded into HTML, and dependent upon a simplified DOM
  - still evolving and is far from platform independent
- JavaScript is not
  - simplified Java -- the two languages have disjoint sets of capabilities
  - simple -- mastery of JavaScript requires advanced programming skills

What JavaScript Can and Can't Do

- JavaScript can:
  - Control document appearance and content
  - Control the browser
  - Interact with the user
  - Read and write client state with cookies
  - Interact with applets
  - Manipulate embedded images
- JavaScript can't:
  - Directly produce graphical dispplays
  - Read or write files
  - Establish network connections
  - Support any kind of multithreading

JavaScript Basics

Syntax Basics

- JavaScript is case-sensitive
- JavaScript ignores whitespace between “tokens”
- Semi-colons are “optional”
- Comments
  - C++ style (i.e. //)
  - C - style (i.e. /* */)
- Identifiers, or “A name used to refer to something else”
  - First character must be a letter or an underscore (_)
- Variables are names associated with a data value.
  - JavaScript is an untyped language (i = 2, sum = ++i)
  - Variable declaration is only required for “local” variables inside a function when variable is also used as a “global” variable (var i; var sum; var i, sum;)

Data Types and Data Type Wrappers

- Primitive Data Types
  - Boolean are true / false values only
  - Functions are code that may be executed multiple times
  - Objects are named pieces of data has a collection of properties
  - Arrays are indexed collection of data values
  - Null indicates “no value”
  - Undefined returned when an variable doesn’t exist
- Data Type Wrappers
  - Each primitive datatype (number, string, etc.) has a corresponding object type defined for it.
  - Object Wrappers contain the same data value but also define properties and methods to manipulate the data values.
  - Wrappers are created as transient objects

Expressions and Operators

- An expression is a phrase that the JavaScript interpreter can evaluate to produce a value.
- There are (generally) three types of operators
  - binary (+, -, *, /, etc.)
  - unary (-3, +62, etc.)
  - ternary (?:)
- A couple useful operators
  - The Conditional (?:)

    greeting = “hello” + ((name != null) ? name : “there”);
  - typeof(i)

    (typeof value == “string”) ? “‘“ + value + “‘“ : value
  - Object Creation Operator (new)

    o = new Object; c = new rectangle(3,5,2,1);
  - The delete operator (sets object value to null)

    Delete o;

Strings

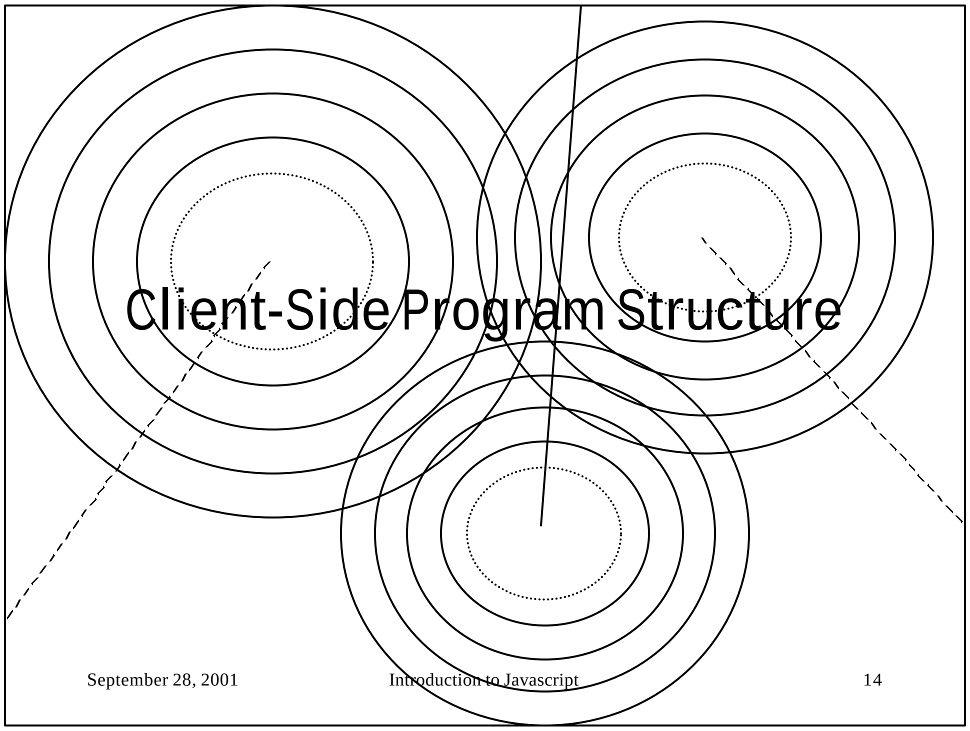
- A series of characters enclosed in double quotes.
- JavaScript has many built-in string operations.
  - concatenation msg = “Hello, “ + “world”;
  - length last_char= s.char(s.length -1);
  - substring sub = s.substring(0,4)
  - indexOf i = s.indexOf(‘a’);
  - charAt i = s.charAt(s.length -1);

Conditional Statements

```
if(name == null)   name = “John Doe”
if((address == null) | | (address == ““))
{
     address = “undefined”;
     alert(“Please provide a mailing address”);
}
if(name == null) name=“John Doe”
else         document.write(name)
```

Loop Statements

```
while(count < 10){
    document.write(count);
    count++; }
for (count=0; count<10; count++)
    document.write(count);
for (prop in MyObject)
    document.write(“name: “ + prop ”; value: “ +
      MyObject[prop], “<br>“);
```



Client-Side Program Structure

Client-Side Program Structure

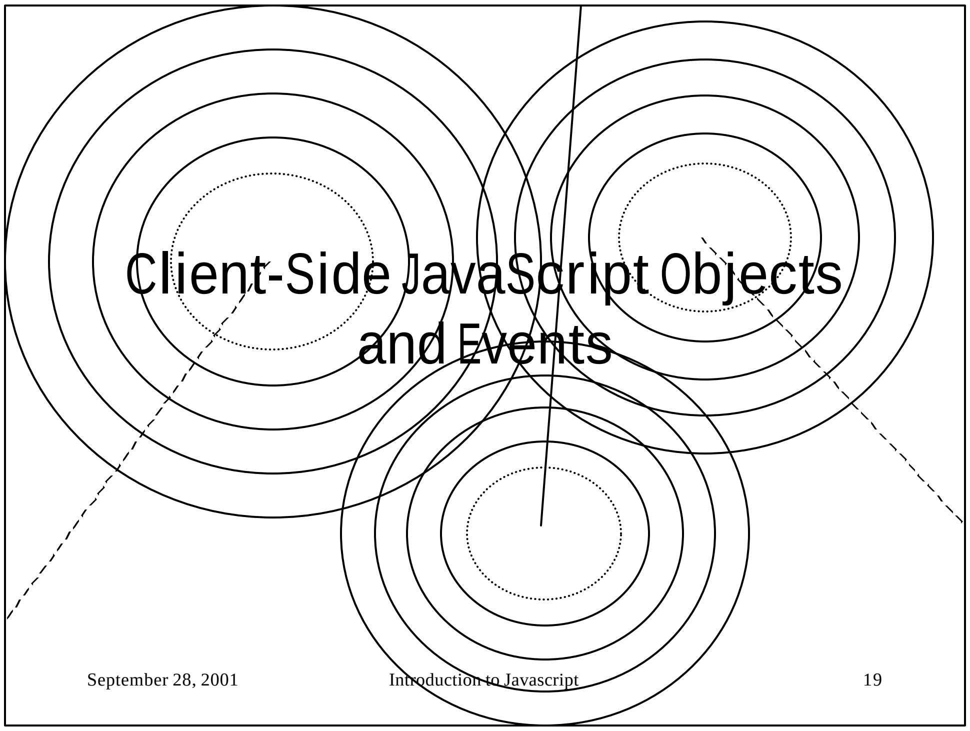
- Techniques for embedding JavaScript code in HTML:
  - code between <SCRIPT> and </SCRIPT> tags.
  - <SCRIPT src=url> to refer to a file of JavaScript.
- A single HTML file may contain more than one pair of (non-overlapping) <SCRIPT> tag pairs
- JavaScript statements between <SCRIPT> tags are executed in the order they appear.
  - functions are an exception
- Different <SCRIPT> pairs on the same page are part of the same JavaScript Program.
  - Context scope is the HTML page, not the script block

```
<HTML>
<HEAD>
<TITLE>Javascript Test File #1</TITLE>
</HEAD>
<BODY>
  <SCRIPT language="JavaScript">
  <!-- this makes the program an html comment
  document.write("<P>This was written by
      javascript</P>");
  // javascript comment to end html comment -->
  </SCRIPT>
  <NOSCRIPT>
  <P>If you see this,
  there is no java scripting on this machine</P>
  </NOSCRIPT>
<P>This para was written by html normally</P>
</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE>Today’s Date</TITLE>
       <SCRIPT LANGUAGE=“JavaScript”>
       // Define functions for later use
       function print_todays_date()
       {
              var d = new Date(); // today’s date and time
              document.write(d.toLocaleString());
       }
       </SCRIPT>
</HEAD>
<BODY>
<HR>The date and time are:<BR><b>
       <SCRIPT LANGUAGE=“JavaScript”>
       // call the function defined above
       print_todays_date();
       </SCRIPT>
</B><HR>
</BODY>
</HTML>
```

Execution of JavaScript Programs

- Scripts
  - in order of appearance as part of the browsers HTML parsing process.
- Functions
  - execute when called
  - Are frequently used as event handlers which allow for asynchronous execution
  - can be defined to manipulate elements that are not yet defined



Client-Side JavaScript Objects and Events

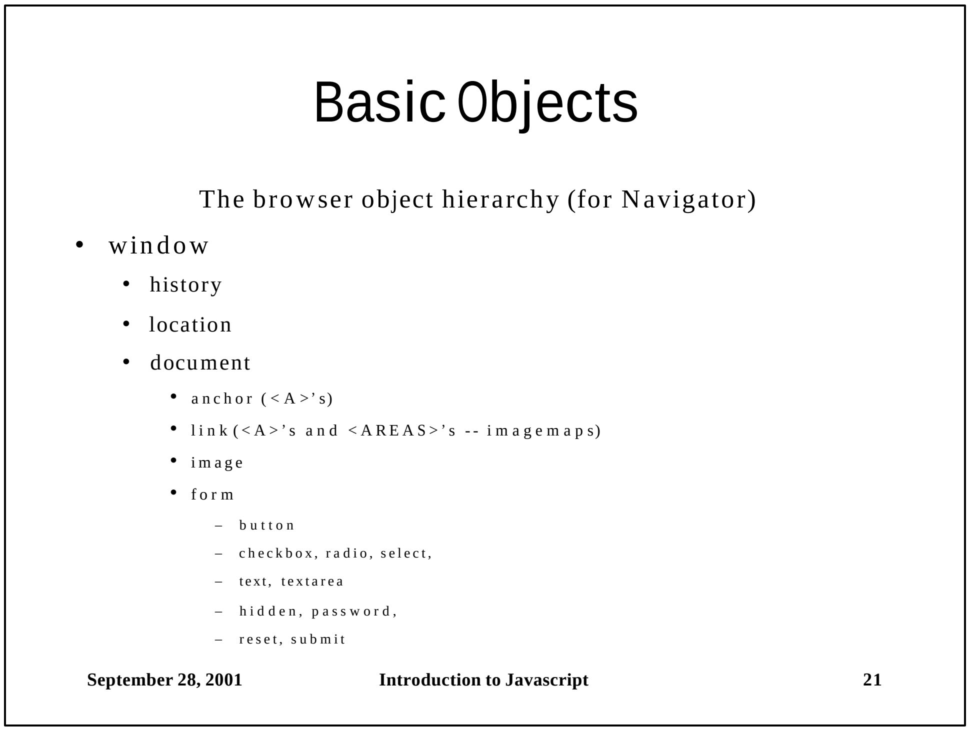
JavaScript and Events

- Events occur when a user interacts with the HTML file (which defines the "user-interface")
- JavaScript extends HTML with the events:
  - onClick, onFocus, onBlur, onChange, onMouseOver
- Event Handlers are normally written as functions

```
<input type text name ="t0"
Value ="" onChange ="validate(this)">
```

- They can be written as direct attribute changes

```
<input type ="text" name ="t1"
Value = "" onChange ="this.value='not so fast'">
```

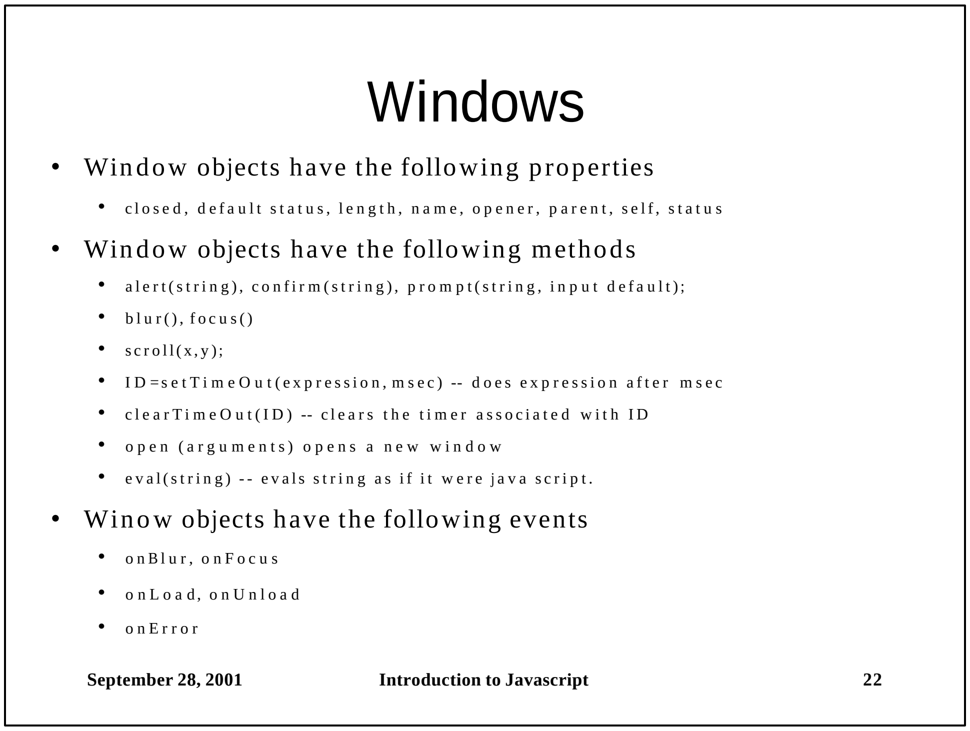


Basic Objects

The browser object hierarchy (for Navigator)

- window
  - history
  - location
  - document
    - anchor (<A>'s)
    - link(<A>'s and <AREAS>'s -- imagemaps)
    - image
    - form
      - – button
      - – checkbox, radio, select,
      - – text, textarea
      - – hidden, password,
      - – reset, submit

September 28, 2001 **Introduction to Javascript** **21**



Windows

- Window objects have the following properties
  - closed, default status, length, name, opener, parent, self, status
- Window objects have the following methods
  - alert(string), confirm(string), prompt(string, input default);
  - blur(), focus()
  - scroll(x,y);
  - ID=setTimeOut(expression, msec) -- does expression after msec
  - clearTimeOut(ID) -- clears the timer associated with ID
  - open (arguments) opens a new window
  - eval(string) -- evals string as if it were java script.
- Winow objects have the following events
  - onBlur, onFocus
  - onLoad, onUnload
  - onError

September 28, 2001 **Introduction to Javascript** **22**

Location and History

- Location
  - The location object has the following properties
    - href, protocol, host, hostname, port, path, hash, search,
  - The Location object only has one method
    - assign(string) changes the href
- History
  - The history object has the following properties
    - current, length, previous, next
  - The history object has the following methods
    - back(), forward(), go(num), and go(string)

Documents

- The Document Object has the following properties
  - alinkColor,linkColor,vlinkColor
  - bgColor, fgColor
  - cookie, domain, lastModified, referrer, title, URL
- The Document object has the following methods
  - close()
  - eval(string)
  - open() opens document for writing
  - write and writeln

Component Arrays

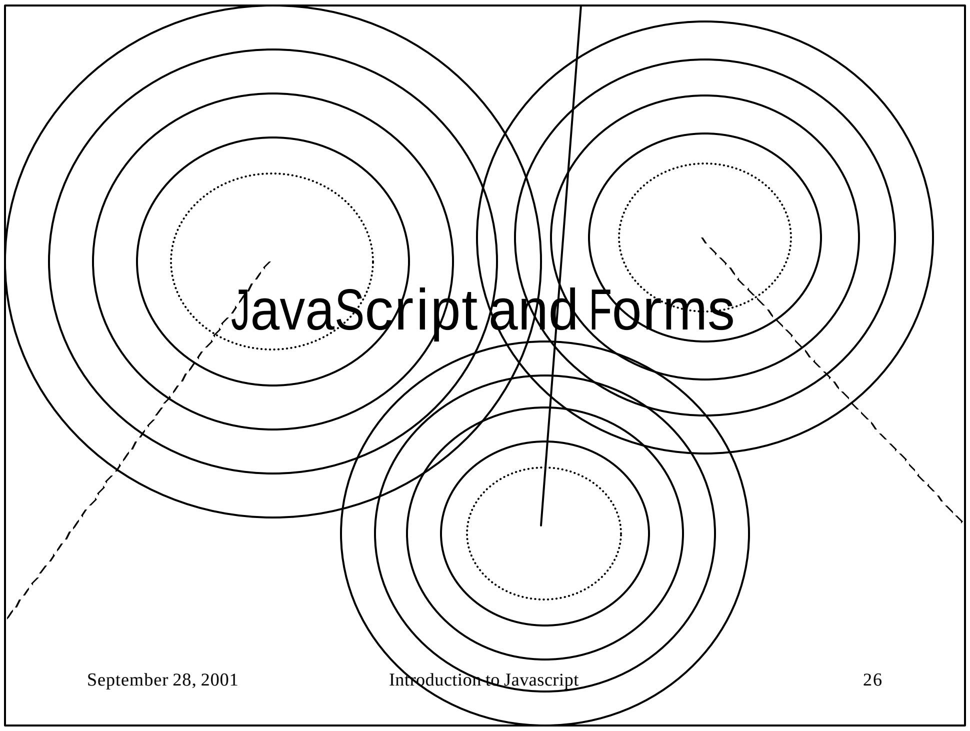
- Some of the power of the document comes from its component arrays
- The arrays can be accessed by number or by associative name
- The following arrays are defined for documents

• anchors	arguments
• elements	forms
• frames	history
• images	links
• embeds	applets
• mimeTypes options	
• plugins	

- events for links, area, and anchor object
  - onClick, onMouseOver, onMouseOut

September 28, 2001 **Introduction to Javascript** **25**

JavaScript and Forms



September 28, 2001 Introduction to Javascript 26

JavaScript and Forms

- In the CGI model a form and its input data are “submitted” - sent to the server - all at once.
- In JavaScript the emphasis is on event handling.
  - While forms have events such as “onSubmit” and “onReset”, a “submit” button is not necessary in JavaScript.
  - The submit function may be performed by any button.
  - In addition elements of a form can respond to events such as:
    - onClick
    - onFocus
    - onBlur
    - onChange

Forms

- Forms have the following properties
  - Name
  - Method
  - Action
  - Enctype
  - Target
- In addition, JavaScript sees
  - Elements
  - Length

Form Elements

- (Almost) all form elements define event handlers.
  - onClick() onChange are the most important.
- !! On Unix, event handlers only work for text entry elements !!
- All elements have a type property
- When user input is passed to the web server it is in the form of name=value pairs.
  - Name property is optional (sort of)
  - Specified default value is over written by user input.

    <INPUT NAME="textfield1" VALUE="value1">
- Button values indicate the text displayed on the button.
- Checkbox and Radio button values the value is the string submitted to the server when a box or button is checked.

The Form Object

- Represents a single HTML form
- All forms are found in the forms[] array.
  - property of the Document object
  - document.forms[0] is the first form on a page.
  - document.forms[document.forms.length] is the last.
- All elements of a form are found in the elements[] array
  - contains JavaScript Objects representing the various input elements of a form.
  - document.forms[2].elements[3].value refers to the value of the fourth element of the third form on a page

A Note About Names

- The name attribute of the <FORM> tag can be useful in referring to form elements.

```
<FORM NAME="questions">
   …<INPUT TYPE="Text" NAME="zipcode"
</FORM>
```

- This allows:

```
document.questions // as opposed to document.forms[0]
document.questions.zipcode //document.forms[0].elements[6]
```

- Checkbox and Radio Button set values are stored in a property array.

```
document.questionnaire.favorite[0]        // first value
document.questionnaire.favorite[1]        // second value
```

Client-Side Form Validation

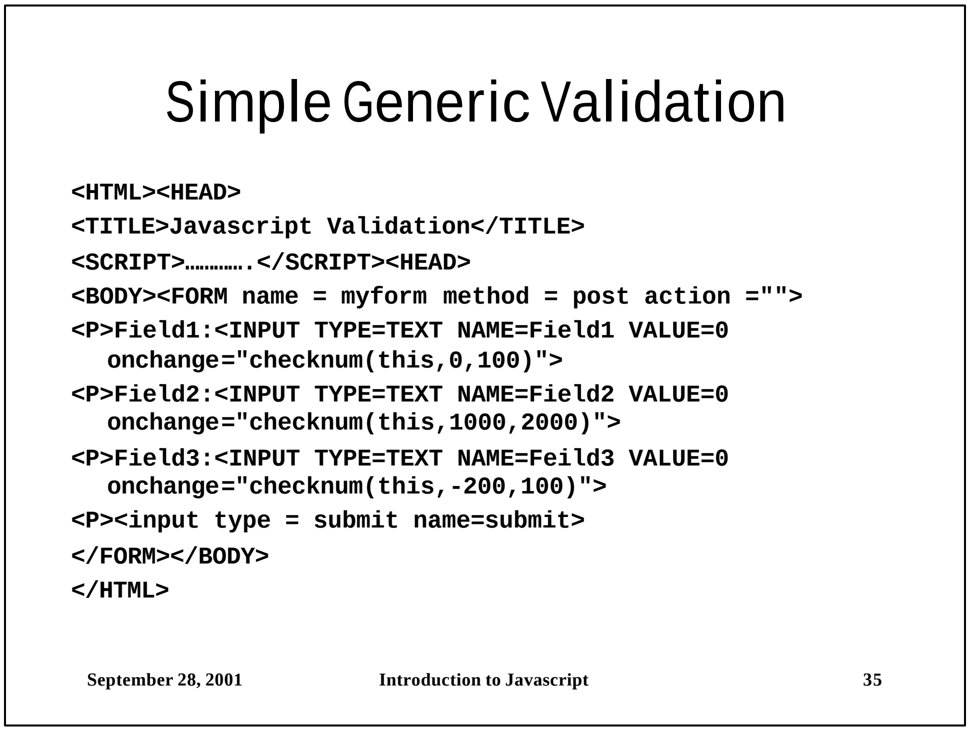
- Checking a form for appropriate content can dramatically reduce traffic to the server.
- onSubmit();
  - Event Handler of the form object.
  - Can notify the user when a form contains missing or invalid input values.
  - relies heavily on the type property of form elements.
  - validation function should return false if form contains input errors.
  - Store and report specific input errors.
  - Cannot handle all checking. (username already taken, etc.)

Simple Validation

```
<HTML><HEAD>
<TITLE>Javascript Validation</TITLE>
<SCRIPT>………….</SCRIPT><HEAD>
<BODY><FORM name = myform method = post action ="">
<P>Field1:<INPUT TYPE=TEXT NAME=PHONE VALUE=0
  onchange="checkphone()">
<P>Field2:<INPUT TYPE=TEXT NAME=NAME VALUE=0
  onchange="checkname()">
<P>Field3:<INPUT TYPE=TEXT NAME=Feild3 VALUE=0
  onchange="checknum(this,-200,100)">
<P><input type = submit name=submit>
</FORM></BODY>
</HTML>
```

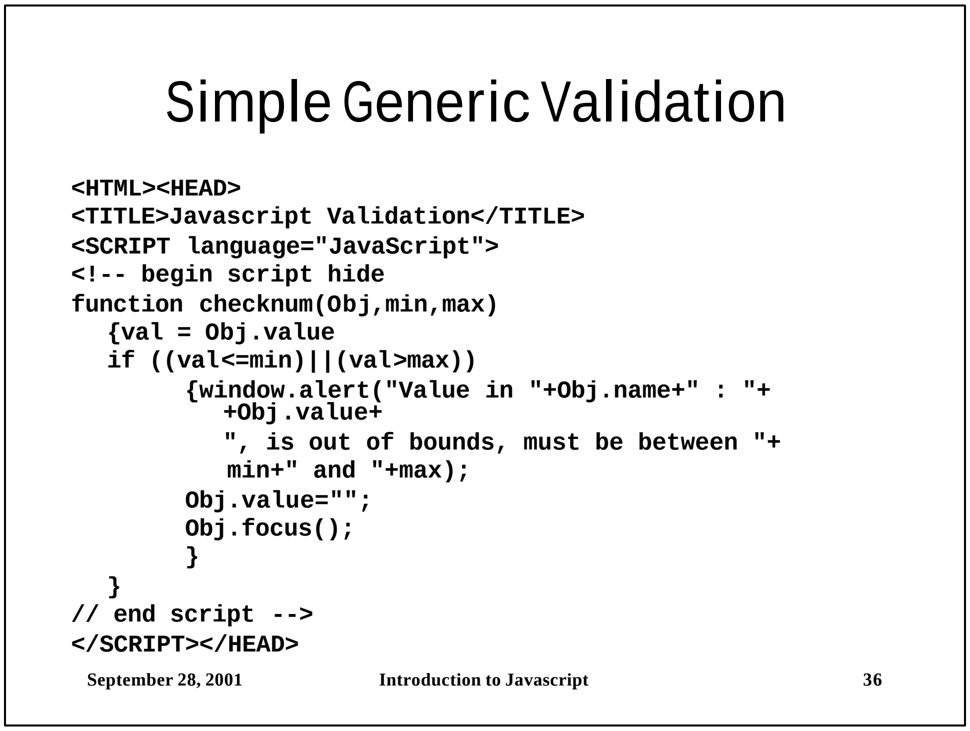
Simple Validation

```
<HTML><HEAD>
<TITLE>Javascript Validation</TITLE>
<SCRIPT language="JavaScript">
<!-- begin script hide
function checkphone()
  {chkstr=document.myform.PHONE.value
  for (i = 0; i < chkstr.length; i++) {
        ch = chkstr.substring(i, i+1);
        // CHECK EACH CHARACTER
        if ((ch >= "0" && ch <= "9"){
        {window.alert(" Phone number is digits only ");
        Obj.value="";
        Obj.focus(); break;}
        }
  }
// end script -->
</SCRIPT></HEAD>
```



Simple Generic Validation

```
<HTML><HEAD>
<TITLE>Javascript Validation</TITLE>
<SCRIPT>………….</SCRIPT><HEAD>
<BODY><FORM name = myform method = post action ="">
<P>Field1:<INPUT TYPE=TEXT NAME=Field1 VALUE=0
   onchange="checknum(this,0,100)">
<P>Field2:<INPUT TYPE=TEXT NAME=Field2 VALUE=0
   onchange="checknum(this,1000,2000)">
<P>Field3:<INPUT TYPE=TEXT NAME=Feild3 VALUE=0
   onchange="checknum(this,-200,100)">
<P><input type = submit name=submit>
</FORM></BODY>
</HTML>
```



Simple Generic Validation

```
<HTML><HEAD>
<TITLE>Javascript Validation</TITLE>
<SCRIPT language="JavaScript">
<!-- begin script hide
function checknum(Obj,min,max)
   {val = Obj.value
   if ((val<=min)||(val>max))
         {window.alert("Value in "+Obj.name+" : "+
            +Obj.value+
            ", is out of bounds, must be between "+
            min+" and "+max);
         Obj.value="";
         Obj.focus();
         }
   }
// end script -->
</SCRIPT></HEAD>
```

Dates in Forms

```
<FORM>
<SCRIPT LANGUAGE="JAVASCRIPT">
function getDate(){
   now = new Date
   var d= now.toLocaleString();
   document.write(d);
   document.write("<INPUT NAME=\"DATE\" TYPE=\"hidden\"
        VALUE=\"" + d + "\"");
   }

getDate();
</SCRIPT>
</FORM>
```

One of Many Compatibility Issues

- !! Internet Explorer does not allow objects to be assigned as input VALUES !!
  - This won’t work:

    ```
    today = new Date();
    document.myform.date.value = today;
    ```
  - But this will:

    ```
    today = new Date();
    document.myform.date.value = ““ + today;
    ```

Prefilling Entries

```
// This function formats a date as mm/dd/yy
function formatDate(dateVar)
{
  newDate = dateVar.toLocaleString();
  newDate = newDate.substring(0,
       newDate.indexOf(“ “));
  return newDate();
}
// Prefill payment date with current date
today = new Date();
document.MyForm.PayDate.value = formatDate(today)
```

Generic Validator

```
<SCRIPT LANGUAGE=“JavaScript1.1”>
function isblank(s){
  for(var i=0; i<s.length; i++){
   var c = s.charAt(i);
   if((c != ‘ ‘) && (c != ‘\n’) && (C != ‘\t’))
       return false;
  }
  return true;
}
function verify(f){
   var msg;
   var empty_fields;
   var errors = ““;
for(var i=0; i < f.length; i++){
  var e = f.elements[i];
  if(((e.type==“text”)||(e.type==“textarea”)) &&
   !e.optional){
   if((e.value==null) || (e.value==““) || isblank(e.value)){
   empty_fields += “\n      “ + e.name;
       continue;
}
```

Generic Validator continued …

```
if (e.numeric || (e.min != null) || (e.max != null)) {
  var v = parseFloat(e.value);
  if (isNaN(v) ||
   ((e.min != null) && (v < e.min))||
   ((e.max != null) && (v > e.max))) {
     errors += "- The field " + e.name + " must be a
   number";
   if (e.min != null)
    errors += " that is greater than " + e.min;
   if (e.max != null) && (e.min != null)
    errors += " and is less than " + e.max;
   else if (e.max != null)
    errors += " that is less than " + e.max;
   errors += ".\n";
   }
     }
   }
}
```

September 28, 2001 Introduction to Javascript 41

Generic Validator still continued

```
if (!empty_fields && !errors) return true;
msg = "________________________________________\n\n";
msg +="The form was not submitted because of";
msg +=" the following error(s).\n";
msg +="Please correct them and resubmit.\n\n";
msg = "________________________________________\n\n";
if(empty_fields){
   msg += " - The following required fields are empty:";
        + empty_fields + "\n";
   if(errors) msg += "\n";
}
msg += errors;
alert(msg);
return false;
}
</SCRIPT>
```

September 28, 2001 Introduction to Javascript 42

The Select and Option Objects

- The select element has no VALUE property.
- The option element does not specify the displayed text but the value submitted to the web server.
  - contained in options[] array.
    - document.forms[0].elements[3].option[6]
- The Option() constructor.
  - In Navigator 3.0 supports dynamic generation of options at run-time.
  - This is in theory only.
  - Very buggy.
  - Can create very nice dynamic menus.

Dynamic Menu Generation

```
<FORM> <SELECT NAME="MainCat"
   onChange="BuildSubCatMenu((this.options[selectedIndex]).va
   lue, SubCat, SubCatOptions);">
<OPTION VALUE="0">Please Select a Subject </OPTION>
<OPTION VALUE="1">Art </OPTION>
<OPTION VALUE="2">English </OPTION>
<OPTION VALUE="3">Foriegn Languages </OPTION>
<OPTION VALUE="4">Health & Physical Education </OPTION>
<OPTION VALUE="5">Mathematics </OPTION>
<OPTION VALUE="6">Life Sciences </OPTION>
<OPTION VALUE="7">Physical Sciences </OPTION>
<OPTION VALUE="8">Social Studies </OPTION>
<OPTION VALUE="9">Technology </OPTION>
<OPTION VALUE="10">Vocational Education </OPTION>
<OPTION VALUE="11">Special Education </OPTION>
</SELECT>
Topic: <SELECT NAME="SubCat">
<OPTION VALUE="-1" SELECTED>Please Select Main
   Subject</OPTION>
</SELECT> </FORM>
```

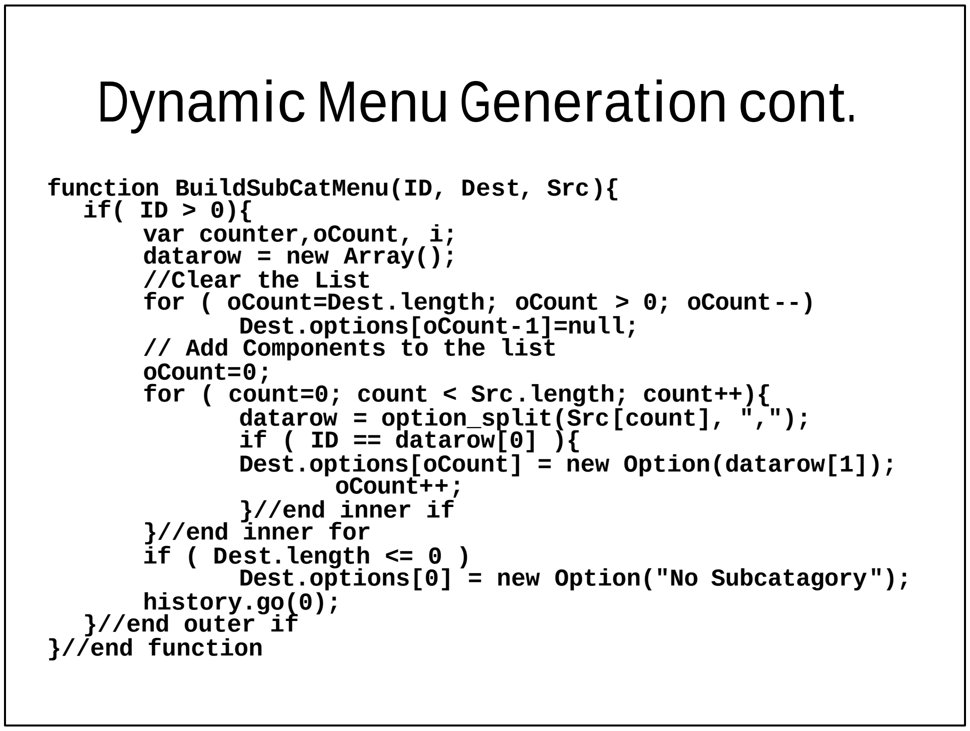
Dynamic Menu Generation cont.

```
<SCRIPT LANGUAGE="JAVASCRIPT">
SubCatOptions = new Array();
SubCatOptions[0] = "1,Appreciation";
SubCatOptions[1] = "1,History";
SubCatOptions[2] = "1,Film/TV";
SubCatOptions[3] = "1,Foundations";
SubCatOptions[4] = "1,General Art";
SubCatOptions[5] = "1,Performing Arts (Music, Theater,
   Dance)";
//English
SubCatOptions[6] ="2,Basic Writing";
SubCatOptions[7] ="2,Creative Writing";
. . .
//  Special Education
SubCatOptions[81] ="11,Hearing";
SubCatOptions[82] ="11,Mentally & Physically Disabled";
SubCatOptions[83] ="11,Severe";
SubCatOptions[84] ="11,Vision";
</SCRIPT>
```

Dynamic Menu Generation cont.

```
<SCRIPT LANGUAGE="JAVASCRIPT">
function option_split(src, delimiter)
{
   count=0
   words=new Array();

   while(src.indexOf(delimiter) > -1) {
        words[count]=src.substring(0,src.indexOf(
   delimiter ));
        count++;
        words[count]=src.substring(src.indexOf( delimiter
   )+1);
        count++;
        src=src.substring(src.indexOf( delimiter )+1);
   }
   return words;
}
</SCRIPT>
```



Dynamic Menu Generation cont.

```
function BuildSubCatMenu(ID, Dest, Src){
   if( ID > 0){
        var counter,oCount, i;
        datarow = new Array();
        //Clear the List
        for ( oCount=Dest.length; oCount > 0; oCount--)
                Dest.options[oCount-1]=null;
        // Add Components to the list
        oCount=0;
        for ( count=0; count < Src.length; count++){
                datarow = option_split(Src[count], ",");
                if ( ID == datarow[0] ){
                Dest.options[oCount] = new Option(datarow[1]);
                        oCount++;
                }//end inner if
        }//end inner for
        if ( Dest.length <= 0 )
                Dest.options[0] = new Option("No Subcatagory");
        history.go(0);
   }//end outer if
}//end function
```