Session: Computer Theory
Topic: Javascript

Daniel Chang
<script> Element

<script Language="JavaScript"> [code] </script>

- Typically within <head> (can be within <body>)
- Allows inclusion of programming language in web page
- The only language we will use is Javascript

Javascript Code

- Program code is enclosed within <script> element
- Must enclose with HTML comment marks in a specific format

<script Language="JavaScript">
    <!--
    [code]

    // -->
</script>
Javascript Functions

• Basic unit of Javascript (like a single shell script)

```javascript
function FunctionName([parameter1] [, parameter2]+) {
}
```

• Start with "function" keyword, then a name for your function
• Parentheses can contain zero or more input parameter names (like shell Positional Parameters)
• Braces mark start and end of function
Javascript Commands (Basic)

Functions

- Javascript provides some predefined functions
- "alert()" raises a pop-up window

```javascript
alert('Text in popup window');
```

Objects

```javascript
objectname.function([parameter1] [, parameter2]+);
```

- Javascript provides objects that represent various "things"
- Functions related to the object can be executed with a dot (.) between the object and function
- All commands must end with a semi-colon (;)
- Comments begin with double-slash (//)

"document" object

- Represents a web page
- `open()` - creates a new document (web page)
- `write()` - adds lines to the document
- `close()` - stops writing to document and displays it

Example

```javascript
document.open();  // create new web page
document.write("<html>");
document.write("<h1>Hello</h1>");
document.write("</html>");
document.close();
```
Executing Functions

- Can link to Anchor tag
  
  `<a href="JavaScript:FuncName()">run function</a>`

- Can attach to an "Event"
  - Lots of elements have lots of events

  `<body onload="FuncName()"> [stuff] </body>`
Forms

- `<form>` element can contain other elements specifying interactive "widgets" on form
- "name" - specifies name for form so that elements are accessible from scripts
- "action" - URL of program that will process form

`<input>` Element

- "type" - specifies type of input widget
  - text - single text line field
  - checkbox - individual checkbox
  - radio - button within a group, where only one button in group can be "on" at any one time. Radio input elements with same "name" attribute are considered within the same group
  - hidden - invisible object used to store information
  - button - individual clickable button
  - submit - button which sends form data
  - reset - button which clears all form items
- "src" - URL of an inline image
- "name" - name of element
- "value" - default for text type, submitted value for checkbox, radio and buttons
- "checked" - no assigned value, indicates radio or checkbox has been selected
- "size" - number of allowed characters in a text type
<select> Element
  • Dropdown list of <option> elements
  • <option> Element
    o "name", "value"

<textarea> Element
  • Multiple line text box (enclosed text is default contents)
  • "name", "rows", "cols"
## Widget Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>change</td>
<td>Value is changed</td>
<td>&quot;text&quot;, &lt;textarea&gt;, &lt;select&gt;</td>
</tr>
<tr>
<td>click</td>
<td>Element or link clicked</td>
<td>any button, links, &quot;checkbox&quot;</td>
</tr>
<tr>
<td>load</td>
<td>Page loaded into browser</td>
<td>&lt;body&gt;</td>
</tr>
<tr>
<td>mouseOver</td>
<td>Mouse passes over element</td>
<td>links</td>
</tr>
<tr>
<td>select</td>
<td>Element input field selected</td>
<td>&quot;text&quot;, &lt;textarea&gt;</td>
</tr>
<tr>
<td>submit</td>
<td>Form info is submitted</td>
<td>&quot;submit&quot;</td>
</tr>
<tr>
<td>unload</td>
<td>Page replaced or quit</td>
<td>&lt;body&gt;</td>
</tr>
</tbody>
</table>
Widgets as Objects

document.formname.widgetname

• formname - the "name" attribute of the <form>
• widgetname - the "name" attribute of any element within the <form>

Example
• HTML

    <form name="form1">
    <input type="checkbox" name="check1" value="god">Game of Death
    </form>

• Javascript reference

    document.form1.check1
Object Members

- Widget Objects contain variables for almost any attribute of the element
- "name", "value", "checked"

Examples

```javascript
document.form1.checkbox1.value
document.form1.checkbox1.checked
```
Javascript Variables
• Use "var" to declare variables to hold values

```javascript
var something;

something = "hello";
alert(something);

something = 4;
something = "a sentence\n";
```

• Object Members can also be treated as variables

```javascript
var Result;

Result = document.form1.text1.value;
Result = Result + 1;
document.form1.text2.value = Result;
```
Arrays

- A single "variable" can be made to hold multiple values
- Use "new" to declare arrays, along with the "Array()" function

```javascript
var sodasArray;
sodasArray = new Array(3);

sodasArray[0] = "Coke";
sodasArray[1] = "Cherry Coke";
sodasArray[2] = "Dr. Pepper";
```

- "Index" is the number used to reference an item in the array
- Indexes start from 0 and go up to one less than the total number of items in the array
Unification of HTML and Javascript

Javascript
- Create functions within <script> element
- Access Widget values using object references

HTML
- Create <form> containing Widget elements
- Attach Javascript functions to Widget events
Programming Structures

```javascript
with (objectname)
{
    [member references]
}
```

- "objectname" - any leading portion of an object reference
- Code within "with" structure can refer to object members directly, without repeating object name
- Example

```javascript
var Result;

with (document.form1.check1)
{
    Result = "Object: " + name;
    Result = Result + "Value: " + value;
    Result += "\n";
}
```

Adds to variable (just like line above)
Conditional Structures

If Statement

```java
if (expression)
{
    [then do these commands]
}
else
{
    [otherwise do these commands]
}
```

- If "expression" is equal to "true", then first commands will be executed
- Optional "else" section is executed if "expression" is "false"
- Example

```java
if (document.form1.checkbox1.checked)
{
    outtext = "checkbox1 was checked ";
    outtext += document.form1.checkbox1.value;
    outtext += "\n";
}
```
While Statement
   while (expression)
   {
      [commands]
   }

- If "expression" is equal to "true", then "commands" will be executed
- "Commands" will continue to be executed until "expression" is false

Example
   var count = 0;
   while (count < 3)
   {
      alert('Current count is: ' + count);
      count = count + 1;
   }
For Loop Statement

```
for (initialization; expression; increment) {
    [commands]
}
```

- Used to iterate commands a set number of times
- "Initialization" will be executed first, and only once
- If "expression" is true, commands will be executed
- At end of commands, "increment" will be executed
- Then "expression" test is repeated

Example

```javascript
var count;
for (count=0; count < 3; count++) {
    if (document.form1.radio1[count].checked) {
        outtext = "radio1 group item " + count;
        outtext += "checked\n";
    }
}
```
String and Numeric Values

- By default any value from an HTML widget is a string
- So a "1" in a "text" input element is treated as the character, not the numeral
- In order to perform mathematical operations HTML widget values must be converted from strings to actual number variables

"parseInt" Function

```javascript
parseInt(string, base);
```

- The parseInt() function converts a string into a numeric value, suitable for mathematic operations
- "string" is a string containing numeric characters (typically a variable)
- "base" is the base to use in interpreting the string
- The parseInt() function "returns" the numeric value, so it must be "assigned" to a variable to retain the value

Examples
```javascript
var someString;
var numericValue;

someString = "1";
numericValue = parseInt(someString, 10);

// this results in someString = "11"
someString = someString + someString;

// this results in numericValue = 2
numericValue = numericValue + numericValue;
```