Overview of JavaScripts

Jeff Offutt

http://www.cs.gmu.edu/~offutt/

SWE 642
Software Engineering for the World Wide Web

Core Web Programming, Hall, PTR
Web Technologies: A Computer Science Perspective, Jeff Jackson

JavaScript

• I was told recently that more programs are written in JavaScript than any other language
  – I find that scary!
• Advantages:
  – Can modify HTML on the client
  – Fairly easy to write simple functions
• Disadvantages:
  – Weak typing
  – Poor development tools
  – Many scalability problems (maintenance, reliability, readability, security, efficiency, portability, …)
Outline

1. Introduction to JavaScript

2. Introduction to Cascading Style Sheets and Dynamic HTML

1. Introduction to JavaScripts

Some of the basic syntax rules and lots of examples
Java Scripts

• JavaScript is not Java
• JavaScript is a scripting language:
  – Embedded in HTML
  – Interpreted as the page is loaded
  – Can manipulate the HTML page
• Is executed on the client (client side)
  – Serverside JavaScript: Netscape’s Livewire allows JavaScript to be used for form processing
  – We will not discuss this
• No type checking

JavaScript History and Versions

• JavaScript was introduced as part of the Netscape 2.0 browser
• Microsoft soon released its own version called JScript
• ECMA developed a standard language known as ECMAScript
• ECMAScript Edition 3 is widely supported and is what we will call “JavaScript”
JavaScript Characteristics

- JavaScript does not need to be compiled
  - JavaScript is an interpreted language
  - A JS interpreter is software that runs inside a browser that reads and executes JavaScript
- Interpreted vs. compiled languages:
  - Advantage: simplicity
  - Disadvantage: efficiency, maintainability, scalability, reliability

Developing JavaScript Software

- Writing JavaScript code
  - Any text editor (e.g., Vim, Emacs, Notepad)
  - IDE editors (Eclipse)
  - Specialized software (e.g., MS Visual InterDev)
- Executing JavaScript
  - Load into browser (need HTML document)
  - Browser detects syntax and run-time errors
    - Firefox: JavaScript console lists errors (Ctrl-Shift-J)
    - IE6: Exclamation icon and pop-up window (Bottom left)
Developing JavaScript Software

Firefox JavaScript console
Tools → Web Developer → Error Console (Ctrl-Shift-J)

IE 6 / 7 Error Window:
Error indicator; double-clicking icon opens error window

Click to see error messages
Why and Why Not JavaScript?

- What can be done with JavaScript on the client and cannot be done with other techniques on the server side?
  - Monitor user events and take action
  - Some dynamic effects

- What can be done on both client and server, but are better with JavaScript?
  - Build HTML dynamically when page is loaded
  - Interactive web pages
  - Communicate with the server asynchronously (Ajax)

- What are the drawbacks of JavaScript?
  - Platform dependent
  - Can be turned off
  - Very hard to write reliable and maintainable Javascript
  - Performance
  - Security
Where does JavaScript Go?

- Between `<head>` and `</head>` or `<body>` and `</body>`
  - `<SCRIPT Language="JavaScript">...</SCRIPT>`
  - `<SCRIPT Language="JavaScript" Src="com_function.js">...

- Some “script” calls may be embedded in the HTML tags
  - `<SELECT name=country onchange="jmp(url)">`
  - `<A HRef="javascript:new_window();">New Win</A>`

- Comments: `//` single-line and `/* */` multi-line

JavaScript Objects

- JS objects have collections of properties

- A property is something that can be modified:
  - Data properties: primitive values or references to objects
  - Method properties: can be executed
JavaScript Syntax (1)

• Data Types
  – Numeric, String, Array, Boolean, NULL (NaN)
  – Mixing numbers and strings (parseInt, parseFloat)
• Variables
  – Begin with letter or _, no white space or punctuation
  – Case sensitive
  – No fixed types
• Operators (similar to Java)
• Statements (similar to Java: if, for, while, switch…)
  – for in loops:
    var myArray = new Array("A", "B","C");
    for (i in myArray) ⇔ for (i=0; i<3; i++)

JavaScript Syntax (2)

28 Reserved Words
http://en.wikibooks.org/wiki/JavaScript/Reserved_Words

break    do    if    switch    typeof
caose    else    in    this    var
catch    false    instanceof    throw    void
continue    finally    new    true    while
default    for    null    try    with
delete    function    return
JavaScript Syntax (3)

• Much of the syntax is similar to C and Java
• No typing (dynamic typing)
  
  ```javascript
  var X = 5;
  X = 5.5;
  X = "doh!";
  ```

• Fields can be added at anytime
  
  ```javascript
  var tmp = new Object();
  tmp.f1 = "yes";
  tmp.f2 = 42;
  ```

  – Literal notation: tmp2 = {f1: "yes", f2: 42};

• Iterating over fields:
  
  ```javascript
  for (f in tmp2)
      document.writeln (f + "is" + tmp2[f]);
  ```

JavaScript Syntax (4)

Constructors

```javascript
function Course (name, num, time, enroll)
{
    this.name = name;
    this.num = num;
    this.time = time;
    this.enroll = enroll;
}
```
JavaScript Syntax (5)

- Object Methods:
  ```javascript
  function Print()
  {
    document.writeln("<TABLE><TR><TH>Field<TH>Value</TR>
    for (fname in this)
    {
      document.writeln("<TR><TD>" + fname + "<TD>" + this[fname]);
    }
    document.writeln("</TABLE>");
  }
  ```
  
- Use:
  ```javascript
  var SWE642 = new Course("SWE for WWW", "SWE 642",
    "M 4:10", 36);
  SWE642.Print();
  ```

JavaScript Syntax (6)

- Arrays:
  ```javascript
  var Students = new Array (36);
  Students (0) = "AA";
  for (var s=0; s <= Students.length; s++)
    document.writeln (Students (s));
  ```

- Put functions in `<HEAD>`
  - Easier to read
  - Function can be used in multiple places
  - Functions parsed and available first if page loads slowly

- Use a backslash to insert quotes in strings:
  "You can\'t be too careful of \"s.\"
First Example


• How do we deploy JavaScript components?
• How do we obtain data and how do we export data?
• How do we monitor user events, and how do we take actions?
• Pay attention to those statements:
  - `document.FieldName.Entry.value`
  - `OnMouseOver = "ChangeFieldName ()"`
  - `Onclick = "ChangeFieldName ()"`

Browser Object Model (BOM)

BOM – collection of objects that the browser makes available to us for use with JavaScript

```
window Object

location Object  history Object  document Object  navigator Object  screen Object

DOM

form Object  link Object  ...  image Object
```
Using BOM objects

• window is a global object
• Addressing is done by separating objects, methods, and properties with dots
  - window.alert ("Hello");
  - alert ("Hello");
• There are also alternative ways to access the objects:
  - document.forms[0].elements[0].value
  - document.myform.nameinput.value
  - document.myform["nameinput"].value

More BOM / DOM Examples

• window.open, window.close, confirm, alert
• history.go(-1) history.go(0)
• window.location.replace(""), location.href="";
• window.Navigator.appName navigator.appVersion
  navigator.cookieEnabled, navigator.userAgent
• window.screen.height, screen.width
• window.document
• this.value
JavaScript Objects (1)

- Math object:
  - Methods: sin(), cos(), floor(), max(), max(), ...

- Number object:
  - Data: MIN_VALUE, MAX_VALUE, NaN, POSITIVE_INFINITY, NEGATIVE_INFINITY, PI
  - Methods: isNaN(), toString()
  - if (isNaN(age)) ... -- false if age is a number, true otherwise

JavaScript Objects (2)

Strings:
- Catenate: var str = "Jeff" + " " + "Offutt";
- Length: str.length; — 11
- Finding a character: str.charAt (5); ‘O’, starts at zero!
- Finding a position: str.indexOf ("f"); — 2, first occurrence
- Getting a substring: str.substring (0, 3); — "Jeff"
- Lower case: str.toLowerCase(); — "jeff offutt"
- Upper case: str.toUpperCase(); — "JEFF OFFUTT"
- var pos = str.search (/ff/); — 2, where "ff" starts
- var nameArr = str.split (" "); — ["Jeff", "Offutt"]
JavaScript Uses

- Two common purposes:
  1. Build HTML dynamically when page is loaded
  2. Monitor user events and take action

- Classes of applications
  1. Customizing web pages
  2. Making web pages more dynamic
  3. Validating forms
  4. Manipulating cookies
  5. Interacting with frames
  6. Calling Java programs
  7. Accessing JavaScript from Java

I. Generating HTML Dynamically

- JavaScript will execute when page is loaded, and output appears in the document
  `<SCRIPT Language="JavaScript">
  <!-- Comment so browsers without JS will ignore it. 
  document.write("JavaScript output\n");
  // -- >
  </SCRIPT>

- This is just a demonstration – the line could have been inserted directly into the HTML
I. Generating HTML Dynamically (2)

```javascript
function ReferringPage () {
    if (document.referrer.length == 0)
        return ("<EM>none</EM>");
    else
        return (document.referrer);
}

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

---

I. Generating HTML Dynamically (3)

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

Language:
- Specifies different versions of JavaScript
- Different browsers know different JavaScripts
I. Generating HTML Dynamically (4)

```javascript
<script language="Javascript">
if (top.frames.length != 0) // Keep page from loading in a frame
{
    top.location = document.URL;
}

function setFocus() // Set the mouse focus in a form
{
    document.gcdform.LHS.focus();
}

var lastModificationDate = new Date(document.lastModified)
document.write ('<span class="quote">Last update: ' +
(lastModificationDate.getMonth()+1) + '/' +
lastModificationDate.getDate() + '/2010' + '</span>');
</script>

II. Monitoring User Events

- Clicking on a button — onClick
- Clicking on a link — onSelect
- Exiting a page — onUnload
- Moving onto or off of a link — OnMouseOver, OnMouseOut
- Input focus on a FORM element — onFocus
- Submitting form — onSubmit
- Image loading error — onAbort
II. Monitoring User Events (2)

```html
<HEAD>
<SCRIPT>
<! --
    function DontClick ()
    {
        alert ("I told you not to click!");
    }
    // -->
</SCRIPT>
</HEAD>

<FORM>
    <INPUT Type="button" Value="Don't click me"
           onClick="DontClick()">
</FORM>
```

Events and Event Handling

Examples


- Event handlers as Attributes
  `<A HRef="next.html" Name="next" OnClick="verify();">next</A>`

- Event handlers as Properties
  `document.links[0].onclick = verify();`
  Note: 1. `document.links ["next"]` won’t work.
  2. Verify is the name of the event handling function.

- Obtain values within the page
- Change values within the page
- Add or remove objects
Validating Forms

• Make sure that each data value is “valid”
  – The value conforms to specific constraints
  – Not necessarily correct
• Validate each input field (text, radio, and etc)
• Submitting form data
  – `<FORM onSubmit="return submitIt (this)"); Method="POST"
    ACTION="...">`

    if submitIt (this) returns false, the data will not be passed to the server
  – “this” refers to the current form

JavaScript Uses

Classes of applications

1. Customizing web pages
2. Making web pages more dynamic
3. Validating forms
4. Manipulating cookies
5. Interacting with frames
6. Calling Java programs
7. Accessing JavaScript from Java
1. Customizing Web Pages

- Compatibility is a major problem with web browsers
- Professional web sites go to a lot of trouble to check and adjust the web pages
- Old browsers do not support BGColor on tables:

```javascript
if (navigator.appVersion.substring(0,1) >= 3) {
  var HeadCellColor = "black";
  var HeadFontColor = "white";
} else {
  var HeadCellColor = "white";
  var HeadFontColor = "black";
}
dw (<TABLE><TR BGColor=" + HeadCellColor + ">
  <TH><FONT Color=" + HeadFontColor + ">

Note: “dw” is a shorthand for “document.writeln” – just for use in the slides.
```

1. Customizing Web Pages (2)

Adjusting images to fit window size

```javascript
function Image (url, w, h)
{
  return (<IMG Src=" + url + " Width=" + w + " Height=" + h + ">

function Apple (w)
{
  return (Image ("apple.gif", w, math.round (w*1.323)));
}
function Apple2 (w)
{
  return (Image ("apple.gif", w, math.round (w*1.155)));
}
```
1. Customizing Web Pages (3)

Adjusting images to fit window size ... cont ...

if navigator.appVersion.substring (0, 1) >= 4)
{
    var imageWidth = Window.innerWidth / 4;
    var fontSize = Math.min (7, math.round (Window.innerWidth / 100));
} else {
    var imageWidth = 200;
    var fontSize = 5;
}
dw ("<TABLE>
" +
    "<TR><TD>" + Apple (imageWidth) + "<BR>
    "<TH><FONT Size=" + fontSize + "">" +
    "Apples are great during the break!" +
    "</FONT>" +
    "<TABLE>");

1. Customizing Web Pages

• Checking for plug-ins
• Firefox and Netscape versions 3 and above have a navigator.plugins array (IE does not)

    if (navigator.plugins ["cosmo player 1.0"])
        dw ("<EMBED Src="CoolWorld.vrml" ... />");
    else
        dw ("This example requires VRML.");
2. Making Web Pages Dynamic

• We can change images on mouse movements or time
• document.images – an array of IMG objects

```html
<IMG Src="Cool.jpg" Name="cool">

function MakeCooler ()
{
    document.images["cool"].src = "cooler.jpg";
}
```

2. Making Web Pages Dynamic (2)

Changing on click :

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

function ClickButton (name, GrayImg)
{
    var origImg = document.images[name].src; // save it.
    SetImg (name, GrayImg);
    var resetstring = "SetImg (" + name + ","  + origImg + ")";
    set Timeout (resetString, 100); // Runs string after 100 milliseconds
```
2. Making Web Pages Dynamic (3)

1. Attach ClickButton() to a button
2. Cache the images

```html
<A Href= "location1.html" onClick="ClickButton ("button1", "images/Button1-down.gif")">
  <IMG Src="images/Button2-Up.gif" Name="Button1">
</A>
```

3. Force a pre-load to save time:

```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 Images (150, 25);
   ImageObjects[i].src = ImageFiles[i];
}
```

---

**JavaScript Tips**

- Build your script one piece at a time
- Store to files and load with HTML files
- Do not change variable types
- Do not use same names for global and local variables
- Remember, JavaScript is case sensitive
- Viewing JavaScript errors
  - Firefox : Tools → Error Console (Ctrl-Shift-J)
  - Microsoft IE : ?
- Debuggers
  - Microsoft IE
    - http://msdn.microsoft.com/scripting
  - Firefox Javascript debugger – Venkman
  - Another Firefox debugger – Firebug
    - http://getfirebug.com/
JavaScript Examples

Most of the previous examples are fully worked out here:

http://www.cs.gmu.edu/~offutt/classes/642/Examples/JS/

II. Cascading Style Sheets and Dynamic HTML

This is pretty straightforward …

   technically …

   the complexity is in imagining what can be done …
Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) provide a simple mechanism for adding style to Web documents

- Fonts
- Colors
- Spacing
- Etc. …

Cascading Style Sheets (CSS)

• Separate the content of the HTML page from its presentation
  - Ex. 1: We might want all paragraph tags, <p>, to have the color dark blue. What if we want to change them all to pink?
  - Ex. 2: What if we want all the links to change text color when the mouse moves over them?

• Appearance control
  - Control the text in ways that are out of reach of HTML tags
  - Special tags, such as <DIV> and <SPAN>, allow more flexible appearance control
How to apply CSS (1)
Embedded style – STYLE elements inside the HEAD element

Class Selectors
Class selector – use to declare a specific style
- Generic class selector
  .footnote { font:70%} (IE)
  <p class="footnote">The is a footnote</p>

- Associate with a specific HTML element
  p.footnote { font:70%} (IE)
  <P Class="footnote">The is a footnote</P>
  <FONT Class="footnote">This a footnote that won’t work</FONT>
Contextual Selector

• Context selector addresses specific occurrences of an element
  – A string of individual selectors separated by white space
  – A search pattern, where only the last element in the pattern is addressed providing it matches the specific context

P.intro B {color:red}

<P Class="intro">The is the <B>most important</B> part. </P>

• Parent-child Selector
  P ~ B {color: red}

Attribute Selector

• A [title] {text-decoration:underline}
• A [class="name"] {text-decoration:underline}
• A [title ="attribute value"] {text-decoration:underline}
• A [title ~= "attribute"] {text-decoration:underline}
How to Apply CSS (2)

- External style sheet
  
  ```html
  <HEAD>
  <LINK Rel=STYLESHEET HRef="style.css" Type="text/css">
  </HEAD>
  ```

- Inline style – through style attribute
  
  ```html
  <P Style="cursor:move; color:blue; font-family: verdana; font-size:12pt;">CSS</P>
  ```

CSS and JavaScript

Dynamic styles

```html
<P onmouseover="style.color='blue';style.fontSize=10;"
    onmouseout="style.color='red';style.fontSize=20;">Dynamic Style</P>
```

Note:

1. `myP.style.color` – means the color property of style object of `myP` element. (Do not forget style)
2. For properties with “-“, style objects have corresponding properties with “-“ removed and the first letter capitalized.
CSS and JavaScript

attachEvent

<SCRIPT Language=javascript>
function go() {
    for (e=0;e<document.all.length;e++)
        if (document.all[e].tagName==’P’) {
            document.all[e]. onmouseover = scred;
            document.all[e]. onmouseout = scblack; } }
function scred() {   event.srcElement.style.color='red';  }
function scblack() {  event.srcElement.style.color='black';  }
</SCRIPT>

<BODY onload = "go();">
DIV/SPAN Tags

- DIV – block-level tag – allows you to create areas on the page.
- DIV tags are commonly used to create elements on the page that will be manipulated independently of the rest of the page

```html
<style>
  div.footnote { font-family:times; font-style:italic}
</style>

<div class="footnote">It's a DIV</div>

- SPAN – inline-tag – used to apply a certain type of style sheet to an area of text or graphics without any line breaks

Summary Client-side Software

- JavaScript
- BOM / DOM
- CSS