JavaScript

Introduction

Outline

- background history and origins
  - strengths and weaknesses
  - JavaScript in the browser
  - JavaScript platforms
- JavaScript and HTML
- syntax
  - data types
- statements
  - if/then
  - iteration
- Date objects
- Array
- functions
- regular expressions

JavaScript origins

- Netscape and Sun Microsystems developed a scripting language named JavaScript in 1995 to add new functionality within web pages
- originally designed for the Netscape Navigator browser
- Brendan Eich, now CTO of Mozilla, developed JavaScript
  - originally named Mocha, then LiveScript
  - Netscape changed the name to JavaScript purely for marketing due to growing Java popularity

JavaScript origins

- JavaScript is not Java, a complex programming language designed for diverse computing purposes
- All major browsers support JavaScript, now one of the most popular web scripting languages
- JavaScript was originally designed with a simpler Java-like syntax just for browsers
- many language syntax similarities with Java and C
- JavaScript is an implementation of the ECMAScript language standard (ECMA International organization – European Computer Manufacturers Association)

JavaScript origins

- prior to JavaScript, web pages used server-side programs (CGI) to handle user interaction with forms, buttons, and menus – where an internet connection to web server must be maintained
- with client-side JavaScript the actions of the user are handled by the browser not the web server – means more information in the web page to download from server but overall faster user experience

JavaScript origins

- browsers equipped with a JavaScript engine interpret then execute the JavaScript code as required
- early JavaScript uses include handling user events like mouse click, hover over a button, and verify data entry in a form
- JavaScript is officially managed by Mozilla Foundation, Javascript web-api's are maintained by W3C
- "JavaScript" is a trademark of Oracle Corporation

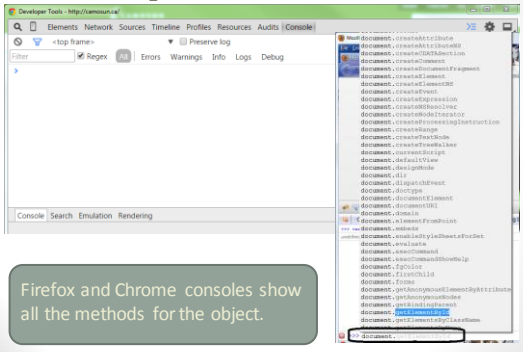
JavaScript - strengths

- quick development
  - no special creation software required
  - fast test and modify cycle
  - many free resources and frameworks available
- easy to learn
  - doesn't share the more complex syntax of Java
  - object oriented structure
- platform independence – all operating systems support it
- interfaces well with the DOM (Document Object Model)
- the most popular web development language
  - basis for JSON, jQuery, and AJAX technologies
- small overhead for browser resources
  - fast download of HTML and JavaScript script, even faster if they are in separate files
- JavaScript code files can be easily shared

JavaScript - weaknesses

- parts of the language can be difficult to master
  - concept of closures, anonymous methods in JavaScript
  - not as well suited for team development as other languages
- rendering varies by browser
  - browsers employ different JavaScript engines resulting in inconsistent functionality and interface
- no JavaScript code hiding
  - JavaScript visible within the HTML page
  - consensus has become JavaScript scripts are essentially 'freeware'
  - JavaScript executing within browser could potentially have malicious code exploits on the user's system
- users have option to disable JavaScript in browser
  - prevent storage of cookies, pop-ups, and other functionality
- search engines may ignore HTML pages containing a lot of JavaScript code
  - Web developers can isolate all the JavaScript code into a separate .js file
- JavaScript always stops running at the first sign of an error
  - Even if you have multiple errors in the JavaScript, only the first one is flagged

JavaScript – browser console



Firefox and Chrome consoles show all the methods for the object.

JavaScript and HTML

- HTML tags <script> </script> contain the JavaScript portion within the HTML file
- attribute "type" identifies the MIME type of the script (usually `text/javascript`)
- as of HTML 5, the "type" is optional and will default to `text/javascript` if not provided
- attribute language="JavaScript1.8" is deprecated

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

   JavaScript source script appears in here

</script>
```


JavaScript and HTML

- hiding scripts from older browsers (pre-IE 6 vintage) which do not support JavaScript, use the *HTML comment element*

<!-- -->

```
<script type="text/javascript">
<!-- Hide the script from some browsers

JavaScript program code …

// Stop hiding from other browsers -->
</script>
```

single line comments in JavaScript use // notation

JavaScript and HTML

- Where to place the JavaScript code within the HTML document?
  - JavaScript programs can be included anywhere in the header or body of the HTML
  - If there is JavaScript to execute prior to the HTML page rendering, then place it in the HTML header
  - JavaScript can be defined anywhere within the HTML body if JavaScript functionality is appropriate for that part of the HTML
  - Form validation of user entered data , mouse hover… etc.

JavaScript and HTML

- Longer or complex scripts can be placed in a separate text file which must have a `.js` file extension
- Usually the scripts that affects page layout are defined within the head element and external scripts (Google analytics, e.g.) at the bottom of the body element (just before `</body>`) to improve the page rendering time
- JavaScript code in the `.js` file should not contain the <script> element

```
<script type="text/javascript"
    src="http://www.you.com/myScript.js">
</script>
<script type="text/javascript"
    src="js/myJSLib.js">
</script>
```


JavaScript - sample 1

- Following page shows HTML having an embedded JavaScript script in the body element using DHTML to write some text (Have a nice day!) to the browser window.
- JavaScript is updating the HTML page content via the DOM object document

```
document.writeln
```



```
<html>
<head>
<title>JavaScript Sample 1</title>
</head>

<body>
   This is sample DHTML JavaScript:
    <script type="text/javascript">

      // Display a greeting message.
     document.writeln("Have a nice day!<br
/>");

    </script>
</body> </html>
```


JavaScript syntax

- Basic unit is one-line statement or expression followed by a semicolon (not mandatory but strongly recommended)
- `document.writeln("...");`
  - JavaScript command invokes the DOM's document object method `writeln( )`
- In JavaScript, as in Java, everything is *case-sensitive*
  - *use document NOT Document or DOCUMENT*
  - *use writeln NOT WRITELN or WriteLN*

JavaScript syntax

Terms:

- Method
  - name of a function associated with an object
  - e.g. write() and writeln() are methods of object document
    - write() and writeln() only differ in that writeln() adds a new line.
- Parameter
  - In the definition of the method or function, the placeholder values passed into the method or function
  - e.g function add(n) { return n+1; } // n is parameter
- Argument
  - The actual values used in the invocation of the method or function
  - e.g. document.write("Hello "); // "Hello" is argument
- When more than one is used, parameters and arguments are separated by commas

JavaScript syntax

- JavaScript layout is free-format
  - It does not matter how you format your JavaScript with white spaces (tabs, new lines)
  - Multiple statements on one line separated by ;
  - Readability is key if you are maintaining the JavaScript code for development
  - Many third party JavaScript libraries are provided in *minimized* form to speed up download (all newlines and unnecessary spaces stripped out) and may obfuscate the JavaScript code (hinder reverse engineering)
  - There are JavaScript code formatters which make JavaScript more easily readable to humans
  - http://javascript.crockford.com/code.html

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

document.writeln ( "This ") ;
   document.writeln(
              "  is  "
    );
document.writeln( "       a   "  )
 ;
     document.writeln(     "line   \
                                .  \
                               " )  ;

</script>
```

Free-format demo – do not write JavaScript like this!

If text spans more than one line, use the backslash \ to continue.

JavaScript tools

- JSLint is a JavaScript program that checks for problems in your JavaScript programs (improves code quality)
  - jslint.com
- YUI Compressor minimizes JavaScript and CSS
  - developer.yahoo.com/yui/compressor
- Dojo Toolkit allows you to use and build custom web page widgets for many platforms
  - dojotoolkit.org

JavaScript comments

- use comments in JavaScript to explain the code purpose and make it human readable.
- use // for one line comment and /* */ for multiline

```
// Use the numeric sort function.
function s(a,b) {
   return (a-b);
}
/*
  This code will write to a heading.
*/
document.getElementById("theHeading").innerHT
ML = "This is the beginning.";
```


JavaScript methods

- `document.writeln("Hello<br />");`
  - Outputs the text `Hello<br />` to the browser, which interprets it as HTML information
  - `writeln` is one of many *methods* associated with the object `document`

All methods are functions in JavaScript

- In JavaScript, methods are called by combining the object name with the method
  - objectname.methodname
- If the object name is omitted, the *window* object is assumed (e.g. `alert` is `window.alert`)

JavaScript methods

- Data that the method needs to perform is provided as an argument within the parenthesis:

```
document.write( "Welcome !" );
document.writeln( "Have a great
  day!");
```

- Script container does not affect the HTML structures where it occurs, so any format tags or elements in the HTML file will affect the text produced by `write()` and `writeln()` methods


```
<html>
<head>
<title>JavaScript Sample 2</title>
</head>

<body>
<p>Here is a sample JavaScript </p><strong>
<script type="text/javascript">

// Display a message
document.writeln("This text appears as bold.
");
document.writeln("</strong>");

</script> </body> </html>
```

JavaScript - prompt

- JavaScript can interact directly with the user
- Simplest way is with the JavaScript `prompt()` method
- Prompt displays its first argument as the prompt text
- Optional second argument is displayed as the default value within the dialog box
- Empty string is returned if user clicks OK without providing any text
- This method of interacting with users is obtrusive and no longer best-practice.


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

var name = prompt("Enter your name:",

"visitor"));

document.write("Welcome " + name );

var sign = prompt("What is your zodiac
sign?");

document.write("<br />");

document.write("Your sign is " + sign + ".");
</script>
```

What do these functions output in a pop-up window?

JavaScript - prompt

- The `prompt` method doesn't need to be prefixed by *`document.`* because it is a method of the `window` object (a built-in functions)
- If the object name is missing, then `window` object is assumed
  - e.g. JavaScript functions `parseInt`, `parseFloat`, `isNaN` do not require to be prefixed by `window`.

JavaScript - alert

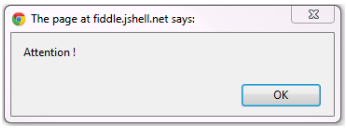
- Alert dialog box
- Useful to show some information in a dialog box
- `Alert("Click OK to continue.");`
- Useful to point out...
  - incorrect information in a form
  - invalid result from a calculation
  - other immediate messages


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

document.write('<img src="welcome.gif">');

alert("Attention !");

</script>
```



What will these alert messages say?

a.
```
var foo = 5;
foo += 5;
alert(foo);
```
10

b.
```
i = 5;
i++;
alert( i );
```
6

c.
```
var foo = 2;
alert(foo + " " + "remaining");
```
2 remaining

d.
```
var foo = "Mat";
var bar = "Jennifer";
if (foo.length > bar.length) {
    alert( foo + " is longer." );
}
else {
    alert ( bar + " is longer.");
}
```
Jennifer is longer

e. `alert( 10 === "10");` false

JavaScript - variables

- Variable names are case sensitive and must start with a letter, dollar sign, or underscore; subsequent characters can be digits 0-9; no reserved JavaScript keywords* allowed
- Best practice: variable name starts with a-z
- Valid JavaScript variable names:
  a rangeRow x1 p_input salary2012
- Invalid JavaScript variable names:
  a# @tag 4H X factor true
- https://developer.mozilla.org/en/JavaScript/Reference/Reserved_Words

JavaScript - variables

- Keyword `var` declares variables
- Subsequent use of `var` for the same variable within the same script block is unnecessary

```
var a;
var selection;
var b, c, d;
```


JavaScript - constants

- A read-only named constant is created with the `const` keyword
- Same name rules as for variables
- Constants cannot change value or be re-declared
- Cannot use same name as an existing function or variable

```
const g = 10.5;
```


JavaScript - assignment

- The single equals sign = is the assignment operator e.g. *variable* = *expression*;
  - The expression on the right is evaluated and the variable name on the left represents that value

```
var a = 0;   // declare variable a having value 0
a  = 100+1;    // variable a now has value 101
a = "cat";     // variable a now has value "cat"
var b = 0, c = true, d = "atom";   // 3 variables
a = b;          // variable b now has value zero
```


JavaScript – scope rules

- In general, always preface the declaration of new variables with the `var` keyword
- If you declare a new variable without the `var` keyword *(implicit declaration)*, you may be accidentally changing the value of the same variable name found in a higher scope…but you are permitted to use `delete` statement on it … not so if you use the var keyword

```
// myobj is a property of the global object, not a variable, // so it can be deleted
delete myobj;
```

More about this later in the scoping section in functions

JavaScript - block

- A block statement is used to group one or more statements within braces { }
- Commonly used with control flow as in loops

```
{
   statement_1;
   statement_2;
    …
   statement_n;
}
```

JavaScript - block

- JavaScript does **not** have block scope. Variables declared within a block are scoped to the containing function or script, and any assignment of values to them continue beyond the block itself.
- (v1.7 JavaScript introduces a `let` keyword which changes this – to be discussed later)

```
var a = 1;
{
   var a = 5;
}
 // variable a is 5
```


JavaScript - variables

- Multiple variables may be declared with one var statement – each separated by a comma

```
var a = 0, b, c = 100, d = "blue sky",
  e = a;
```

- This practice is slightly more execution efficient than declaring each variable with a separate `var` but not as maintainable
  - potentially, an error will occur if you remove a declared variable and the comma separator
  - e.g `var a = 0 , b d = "blue sky", e = a;`

JavaScript – data types

JavaScript provides five primitive data types

- Numeric as in 0, -21, and 32.62
- Strings as in "Hello" and 'There'
- Boolean (logical) either `true` or `false`
- `null` special keyword for a nothing value; null is primitive and case-sensitive (not NULL or Null)
- `undefined` for something not yet assigned a value or an unknown variable; also primitive
  - For example, a string is *undefined* before you give it a value.

JavaScript - numeric

- An integer number is a sequence of digits
  - range is -2^{53} to 2^{53} (-9007199254740992 to 9007199254740992 inclusive)
  - base 10 integers (decimal) do not start with a zero
  - base 8 integers (octal) start with a zero (deprecated)
  - base 16 integers (hexadecimal) start with 0x

```
var a = 0100;  // a is 64
var b = 100;   // b is 100
var c = 0x010; // c is 16

var d = 0x3a - 0200;  // d is -70
var e = -073 - 0x0b;  // e is -68
```


JavaScript - numeric

Floating point literals

- floating-point literals must have at least one digit and either a decimal point or "e" (or "E")
- range is 5e-324 to 1.797e308
- JavaScript keyword: Infinity or –Infinity
- `Number.POSITIVE_INFINITY, Number.NEGATIVE_INFINITY` and `Number.MAX_VALUE, Number.MIN_VALUE`

```
var a = 10.010101;
var b = -0.99;
var c = 1.45E10;
var d = 2e-2;
var bigNum  = 2/0;   // bigNum  is Infinity
```


JavaScript - string

- Strings store a piece of text
- JavaScript has two kinds of strings: primitives and objects
- Primitives: can use JavaScript String() or assignment

```
var txt = String("Hello");
var txt = "Hello";
```

- Objects: use new String()

```
var txt = new String("Hello");
```

- Use primitive form unless object form is required.

JavaScript - string

- String length displayed using `length` method

```
var txt_len = "hello".length;
// txt_len is 5
```

- Empty string `""` has a length of zero
- Special characters such as `"` `'` `\` and backspace, newline, tab, carriage return can be defined within a string this way: "\b" "\"", '\' ',"\\", "\n", "\t", "\r" respectively

```
var t = "He said, \"Welcome\".";
```


JavaScript - string

- Concatenation operators are `+` and `+=`

  `"Welcome to " + "my house"` makes the string `"Welcome to my house"`

  `welcome += " Thank-you."` adds the string "Thank-you." to the end of the string variable named `welcome`

  also, *string1*.concat(*string2*) method

```
var n = "abc";
var t = n.concat("xyz");
                // t is "abcxyz"; n is "abc"
```


JavaScript - string

- Access an individual character within a string in two ways, using the `charAt` method or as an array (first character is index zero)
  - `"mouse".charAt(1)` is "o"
  - `"mouse"[1]` is "o"

```
var pet = "mouse";
var c = pet.charAt(1);   // c is "o"
c = pet[1];              // c is "o"
```


JavaScript - string

- `substr` method returns a portion of a string
  - *string*.substr(*start_index*, *length*) length is optional but if not provided, extract characters until end of string

```
var answer = "quick";

var n1 = answer.substr(1, 2); // ui

var n2 = answer.substr(2);    // ick

var n3 = answer.substr(-1);   // k
```

https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/String#Properties_2

JavaScript - string

- `replace` method substitutes one substring with another
  - *string*.replace(*search_string*, *new_string*)

```
var t = "white car with white seat";

var n = t.replace("white", "blue");
var p = t.replace(/white/g, "red");
    // n is "blue car with white seat"
    // p is "red car with red seat"
    // t is "white car with white seat"
```


JavaScript - string

- `toLowerCase` and `toUpperCase` convert the string's case
  - these two methods require no arguments

```
var city = "Victoria, BC";

var n1 = city.toLowerCase();  // victoria, bc

var n2 = city.toUpperCase();  // VICTORIA, BC
                              // city is still
                              // Victoria, BC
```

JavaScript - string

Important to remember...

- string "null" is not the same as null
- string "undefined" is not the same as undefined
- string "" is not the same as null or undefined

JavaScript - boolean

- boolean values are either true or false
- double equals operator == tests if two operands represent the same value (but not the same *type*)
- triple equals operator === tests if two operands represent the same value **and** the same type
- non-zero numeric values evaluate to `true`
- null, undefined, NaN, and "" evaluate to `false`

```
var a = true;
var b = false;
var c = (1 == 1);      // c is true
var d = (a = 2);       // d is true, a is 2
var e = (1 == "1");    // e is true
var f = (1 === "1"); // f is false
```