CITS1231 Web Technologies
JavaScript
Contents

• Introduction to JavaScript
• Variables
• Operators
• Conditional Statements
• Program Loops
• Popup Boxes
• Functions
User Interaction

- User interaction requires web page content to change.
- Web server needs to send new web pages.
  - Delays in communicating with server (slow interaction).
  - Server load goes up (cost and delay).
  - User experience goes down.
  - Cost of servers can go up.
- Recall fast client side image maps versus slow server side maps. But this interaction is just hyperlinks.
- How can we have richer user interaction that is fast?
Introduction to JavaScript

• Designed to add interactivity to HTML page
• Client side processing means fast response
• Can generate new HTML code on the fly without server interaction
• Most popular scripting language on the internet
• Scripting language is a lightweight programming language
• Usually embedded directly into HTML pages
• Works in all major browsers
• No license required
Java and JavaScript

• Are Java and JavaScript the same?
  – NO!
• They are two completely different languages!
• Java is much more complex (as per C and C++).
• JavaScript’s official name is ECMAScript
  – Invented by Brendan Eich at Netscape (with Navigator 2.0)
  – developed and maintained by the ECMA organization
What Can JavaScript Do?

• Gives HTML designers a programming tool with a very simple syntax!

• Dynamically generate text and put into an HTML page.

• React to events (eg when page completes loading, user clicks on HTML element).

• Read and change the content of an HTML element.

• Validate form data before submission to server.

• Detect the visitor's browser (and load another page specifically designed for that browser).

• Store and retrieve information on the visitor's computer (cookie).
Example: Hello World

```html
<html>
<body>
<script type="text/javascript">
    document.writeln("<h1>Hello CITS1231 Students!</h1>");
</script>
</body>
</html>
```
JavaScript Element

• General form:

  <script type="text/javascript">
  JavaScript Commands
  </script>

• JavaScript is Case Sensitive (unlike HTML)
  – Watch your capitalization closely when you write JavaScript statements.
JavaScript Comments

- Use comments to explain your JavaScript.
- Single line comments start with //.
- Multi line comments start with /* and end with */.

```javascript
<script type="text/javascript">
    //
    // this script is used to illustrate the use of comments
    //
    //document.writeln("<p>This paragraph has been commented out</p>");
    document.writeln("<p>Normal paragraph</p>"leneck);
    /*
    document.writeln("<p>This paragraph commented out using multi-line</p>");
    document.writeln("<p>Another paragraph commented out using multi-line</p>"leneck);
    */
    document.writeln("<p>Another paragraph</p>"); // comments at end of line
</script>

- You can use comments at end of line.
Browsers Without JavaScript

• Some simple browsers don’t support JavaScript.
  – They will just display the JavaScript as page content.
  – Use HTML comments to prevent this:

    <script type="text/javascript">
    <!--
      document.writeln("<h1>Hello CITS1231 Students!</h1>"),
      //-->
    </script>

• The “//” is a JavaScript comment symbol - prevents JavaScript from executing -->. More later.
JavaScript Variables: Revision

- High School Algebra Revision:
  
  \[ x=1, \ y=6, \ z=2^x+y \]

- A letter (like \( x \)) can be used to hold a value (like 1).

- These letters are called variables.

- Variables can be used to hold values (\( x=1 \)) or expressions (\( z=2^x+y \)).

- In the above example, the value of variable \( z \) is 8.
JavaScript Variables

• As per algebra, JavaScript variables can hold values or expressions.

• Can have short names (eg x), or descriptive names (eg income).

• Names are case sensitive (z and Z are different variables).

• Names must begin with a letter or the underscore character.

• Use the `var` statement to create/declare a variable:

```javascript
var x; // not initialized;
var z = 100;
var _myIncome = 1000;
var _yourIncome = _myIncome + z;
```

• As per algebra, you can do arithmetic with Java variables.

  `_yourIncome` evaluates to 1100.
Identifiers

• Identifiers are the names of your variables, functions and objects. Try to make them meaningful, e.g. name rather than x21

• Function Identifiers must be unique within a page.

• They must start with a letter or underscore and can contain letters and the digits 0 - 9
  
  – Valid identifiers: Fred29 _foo AbCd
  – Invalid identifiers: hello! 97abc

• They cannot include spaces, or be a JavaScript keyword.
Variables

- A variable is an identifiable or named element (often referred to as an identifier). It is used to store and retrieve data.

  \[
  \text{radius} = 2.54; \quad \text{ThisMonth} = \text{“May”};
  \]

- Once the variable is assigned a value, referring to the name is equivalent to referring to its current value.

- To use a variable it must be declared (created). This is done in one of 3 ways

  \[
  \begin{align*}
  \text{var identifier} &; \quad \text{eg} \quad \text{var radius}; \\
  \text{var identifier} &\quad = \text{value} ; \quad \text{eg} \quad \text{var day} = \text{“Friday”}; \\
  \text{identifier} &\quad = \text{value} ; \quad \text{eg} \quad \text{c_to_i} = 2.54;
  \end{align*}
  \]

- “=” sign is used for assigning a value to a variable.

- Don’t confuse with “==” which is used for comparison in JavaScript (see later).
JavaScript Variables

• The power of programming comes from being able to store, reuse and manipulate values in variables

• JavaScript variables should be declared

  var variable;

• Then they can be assigned, eg

  var userName="mark";

• And processed

  document.writeln(userName + "@" + deptAddr);

• Variables can store strings, numbers, booleans, or null.
An Example

<table>
<thead>
<tr>
<th>Dr Fred Lecturer</th>
<th>CS1.15</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;script type=text/javascript&gt;</code></td>
<td></td>
</tr>
<tr>
<td>var userName=&quot;fred&quot;;</td>
<td></td>
</tr>
<tr>
<td>var deptAddr=&quot;csse.uwa.edu.au&quot;;</td>
<td></td>
</tr>
<tr>
<td>var addr=userName+&quot;@&quot;+deptAddr;</td>
<td></td>
</tr>
<tr>
<td>document.writeln(&quot;&lt;a href='mailto:'+addr+''&gt;&quot;);</td>
<td></td>
</tr>
<tr>
<td>document.writeln(addr);</td>
<td></td>
</tr>
<tr>
<td>document.writeln(&quot;&lt;/a&gt;&quot;);</td>
<td></td>
</tr>
<tr>
<td><code>&lt;/script&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;/td&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;/tr&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>
Variables and Literals

- There are two different ways to use values in your scripts.
  - Variables hold data that can be changed at any time
  - Literals are fixed values that don’t change (eg 1, “blah”).
- Values have an associated type. Some important JavaScript types are:
  - integers (27)
  - floating point numbers (42.03948208)
  - strings (“hello”)
  - booleans (TRUE or FALSE)
- JavaScript is not strong typed, you do not need to declare the variable type. Eg:
  
  var x = 27; var y = 42.03948208; var z = “hello”; var t = TRUE;
## Arithmetic Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Combines or adds two items</td>
<td>Men = 20; Women = 25; Total = Men + Women;</td>
</tr>
<tr>
<td>-</td>
<td>Subtracts one item from another</td>
<td>Income = 1000; Expense = 750; Profit = Income - Expense;</td>
</tr>
<tr>
<td>*</td>
<td>Multiplies two items</td>
<td>Width = 50; Length = 20; Area = Width * Length;</td>
</tr>
<tr>
<td>/</td>
<td>Divides one item by another</td>
<td>Persons = 50; Cost = 200; CostPerPerson = Cost / Persons</td>
</tr>
<tr>
<td>%</td>
<td>Calculates the remainder after dividing one value by another</td>
<td>TotalEggs = 64; CartonSize = 12; EggsLeft = TotalEggs % CartonSize;</td>
</tr>
</tbody>
</table>
## Unary Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Equivalent To</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>Increases the item’s value by 1</td>
<td>x++</td>
<td>x = x + 1</td>
</tr>
<tr>
<td>--</td>
<td>Decreases the item’s value by 1</td>
<td>x--</td>
<td>x = x - 1</td>
</tr>
<tr>
<td>-</td>
<td>Changes the sign of the item’s value</td>
<td>-x</td>
<td>x = 0 - x</td>
</tr>
</tbody>
</table>
# Assignment Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Equivalent To</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Assigns the value of the expression on the right to the expression on the left</td>
<td>x = y</td>
<td>x = x + y</td>
</tr>
<tr>
<td>+=</td>
<td>Adds two expressions</td>
<td>x += y</td>
<td>x = x + y</td>
</tr>
<tr>
<td>-=</td>
<td>Subtracts the expression on the right from the expression on the left</td>
<td>x -= y</td>
<td>x = x - y</td>
</tr>
<tr>
<td>*=</td>
<td>Multiplies two expressions</td>
<td>x *= y</td>
<td>x = x * y</td>
</tr>
<tr>
<td>%=</td>
<td>Calculates the remainder from dividing the expression on the right from the expression on the left</td>
<td>x %= y</td>
<td>x = x % y</td>
</tr>
</tbody>
</table>
Conditional Statements

• Allow you to perform different actions for different decisions.

• **if** statement
  – used to execute some code if specified conditions true.

• **if...else** statement
  – execute some code if the condition is true and another code if the condition is false

• **if...else if....else** statement
  – select one of many blocks of code to execute

• **switch** statement
  – select one of many blocks of code to execute
if Statement

• Syntax

\[
\text{if (condition) }
\begin{align*}
\{ \\
\text{commands} \\
\}
\end{align*}
\]

• Example

\[
x = 3; \\
\text{if (x==3)} \\
\begin{align*}
\{ \\
\text{document.writeln("Yes x equals 3");} \\
\}
\end{align*}
\]

\[
// x is assigned value 3 \\
// test if x has value 3
\]
if...else Statement

• Example

```javascript
x = 4; // x is assigned value 3
if (x==3) { // test if x has value 3
    document.writeln("Yes x equals 3");
}
else // all other cases...
    { // all other cases...
        document.writeln("No x does not equal 3");
    }
```
if...else if...else Statement

• Example

```javascript
x = 4; // x is assigned value 3
if (x==3) { // test if x has value 3
    document.writeln("Yes x equals 3");
}
else if (x==4) // test if x has value 4
    {document.writeln("x equals 4");
}
else // all other cases...
    {document.writeln("x does not equal 3 nor 4");
}
```
switch Statement

- Syntax

```java
switch(condition)
{
    case 1:
        execute code block 1
        break;
    case 2:
        execute code block 2
        break;
    default:
        for all other cases, execute code default block
}
```
Example: switch

- Example

```javascript
switch(x)
{
    case 3:
        document.writeln("x equals 3");
        break;
    case 4:
        document.writeln("x equals 4");
        break;
    default:
        document.writeln("x neither equals 3 nor 4");
}
```
Comparison Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>Returns true if the values are equal</td>
<td>x == y</td>
</tr>
<tr>
<td>!=</td>
<td>Returns true if the values are not equal</td>
<td>x != y</td>
</tr>
<tr>
<td>&gt;</td>
<td>Returns true if the value on the left is greater than the value on the right</td>
<td>x &gt; y</td>
</tr>
<tr>
<td>&lt;</td>
<td>Returns true if the value on the left is less than the value on the right</td>
<td>x &lt; y</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Returns true if the value on the left is greater than or equal to the value on the right</td>
<td>x &gt;= y</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Returns true if the value on the left is less than or equal to the value on the right</td>
<td>x &lt;= y</td>
</tr>
</tbody>
</table>
### Logical Operators

In the following examples, assume that \( x = 20 \) and \( y = 25 \)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>Returns true when both expressions are true</td>
<td>((x == 20) &amp;&amp; (y == 25))</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Returns true when at least one expression is true</td>
</tr>
<tr>
<td>!</td>
<td>Returns true if the expression is false and false if the expression is true</td>
<td>(! (x == 20))</td>
<td>false</td>
</tr>
</tbody>
</table>
Program Loops

- Allow you to execute same block of code repeatedly.
- **for** loop:
  - loops through block of code a specific number of times
- **while** loop:
  - loops through block of code while a specified condition is valid
Working for Loop

- Syntax:
  ```
  for (initial;test;iterate)
  {
    code to be executed
  }
  ```
- **initial** is an expression evaluated before the program loop.
- **test** is an expression evaluated at the start of each loop. The loop is terminated if test evaluates to false.
- **iterate** is an expression that is evaluated at the end of each loop.
## Counter Values in the For loop

<table>
<thead>
<tr>
<th>For Loop</th>
<th>Counter Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>for (i=1; i &lt;= 5; i++)</td>
<td>i = 1, 2, 3, 4, 5</td>
</tr>
<tr>
<td>for (i=5; i &gt; 0; i--)</td>
<td>i = 5, 4, 3, 2, 1</td>
</tr>
<tr>
<td>for (i=0; i &lt;= 360; i+=60)</td>
<td>i = 0, 60, 120, 180, 240, 300, 360</td>
</tr>
<tr>
<td>for (i=1; i &lt;= 64; i*=2)</td>
<td>i = 1, 2, 4, 8, 16, 32, 64</td>
</tr>
</tbody>
</table>
Example: for loop

```javascript
<script type="text/javascript">
    var i=0;
    for (i=0;i<=5;i++)
    {
        document.writeln("The number is "+i);
        document.writeln("<br />");
    }
</script>
```
Example: while loop

```javascript
<script type="text/javascript">
    var i = 0;
    while (i <= 5) {
        document.writeln("The number is " + i);
        document.writeln("\n");
        i++;
    }
</script>
```
JavaScript Popup Boxes

- JavaScript has three kind of popup boxes

- Alert box
  - Syntax: `alert("sometext");`
  - Pops up a box with text.

- Prompt box
  - Syntax: `prompt("question, defaultvalue");`
  - Pops up a box with question and obtains user’s answer.
  - The answer will be pre-filled with defaultvalue.

- Confirm box
  - Syntax: `confirm("question");`
  - Pops up a box with question.
  - The user can choose “Ok” or “Cancel”.
Example: Alert Popup

```html
<body>
<script type="text/javascript">
    alert("This is an example of an alert box");
</script>
</body>
```

Example: Prompt Popup

```html
<body>
<pre>
<script type="text/javascript">
nn=prompt("What is your name?", "patrick");
document.writeln("Hello "+nn);
document.writeln("Hope your day is going well!");
</script>
<pre>
</body>
```

Example: Confirm Popup

```html
<body>
<pre>
<script type="text/javascript">
    gender = confirm("Press a button");
    if (gender==true)
    {
        document.writeln("You pressed OK.");
    }
    else
    {
        document.writeln("You pressed Cancel.");
    }
</script>
</pre>
</body>
```

JavaScript Functions

- Contains code to be executed by an event or by a call to the function.

- Browser does not automatically execute code inside function unless the function is explicitly called.

- Can be called from anywhere within a page.

- Can be embedded in an external .js file.

- Can be in the <head> or <body>.

- Better to put in <head> so it is available to whole page.
JavaScript Function

- Syntax:
  
  ```javascript
  function functionname(var1,var2,...,varN)
  {
    code
  }
  ```

- Example:
  
  ```html
  <head>
  <script type="text/javascript">
  function handleClick()
  {
    alert("You clicked the button. Well done!");
  }
  </script>
  </head>

  <body>
  <form>
    <input type="button" value="Click here!" onclick="handleClick()" />
  </form>
  </body>
  ```

Summary

JavaScript can provide fast and rich user interaction.

- Variables
- Operators
- Conditional Statements
- Program Loops
- Popup Boxes
- Functions