EBU6501 - Middleware
Week 3, Day 3: JavaScript Programming

Gokop Goteng & Ethan Lau

Queen Mary
University of London

Original version created by: Dr. Gokop Goteng

Lecture Aim and Outcome

- Aim
  - The aim of this lecture is to teach students how to programme in JavaScript
- Outcome
  - At the end of this lecture students should be able to:
    - Know the differences between Java and JavaScript
    - Write simple JavaScript programmes
    - Understand how to use JavaScript in HTML document

Lecture Outline

- Revision on JSP
  - Discussion on pre-lecture online assignment
- JavaScript (JS) as a Language
- JavaScript Structure
- JS Syntax
- JS Operators
- JS Variables
- JS Data Types
- JS Conditional and Loop statements
- JS Functions
- The HTML Document Object Model (DOM)
- JS Examples
- Summary

Revision on JSP

- What is a JSP?
- Write a simple “Hello World” JSP Programme
- What is MVC design Pattern?
- Which part of MVC is implemented using JSP
- Describe the following
  - Scriptlet
  - Declaration
  - Directive
  - Tag
- Is Java the same Thing as JavaScript? Explain.
- Which Programming Languages are Similar to JavaScript and Why?
- Scripting languages are good for what types of programming tasks?

JavaScript as a Language

- It is a **"scripting"** language
  - Uses interpreter as against compiler
- It is a **dynamic programming language** that is used in
  - Web application development
  - Mobile application development
  - Also used in non-web applications such as PDF
- It is used for both **client** side (web browsers) development as well as **server** side application development
- JavaScript is **NOT** Java
  - They have different semantics
  - The semantics of Java is derived from the C programming language
    - if statement, switch statement, while loop, do while loop, etc
- It is similar to PHP
- You can embed it in many places in a HTML programme
- You can use it with HTML and PHP
- It supports regular expression (regex) as Perl and is used for data mining (manipulating text) for this reason

JavaScript Structure

- The JavaScript **must start** with the **<script> tag** and **end** with the **</script> tag**

```
<script>
document.getElementById(“myJavaScript").innerHTML = "My JavaScript Programme";
</script>
```

- Can be **embedded** within the **<head> </head>** and **<body> </body> tags** of HTML

```
<html>
<body>
      <script>
            document.getElementById(“myJavaScript").innerHTML = "My JavaScript Programme";
   </script>
</body>
<html>
```

Or

```
<html>
<head>
      <script>
            document.getElementById(“myJavaScript").innerHTML = "My JavaScript Programme";
   </script>
</head>
<html>
```

JS Syntax

- The rules of how the language is **constructed** is called a "**Syntax**"
  - It is called **"syntax" for all programming languages – E.g. Syntax error!**

- **Expressions** are a combination of variables or constants that cannot be executed on their own
  - Examples of expressions

```
10 + 7
10 – 7
10/7
```

- **Statements** are the **complete "sentences"** in programming
  - Examples of JS statements

```
var mySum = 10 + 7;
Var mySub = 10 – 7;
myDiv = 10/7;

document.getElementById("myJavaScript").innerHTML = "My JavaScript Programme";
```

```
var x = document.getElementById("demo");   // Get the element with id="demo"
x.style.color = "red";                     // Change the colour of the element
```

JS Operators

- Arithmetic

  +, -, *, /, %, ++, --

- Comparison

  <, >, ==, <=, >=, !=

- Logical

  &, |, ~, &&, ||, !

JS Variables

- **Variables** are **declarations** that are used to **store values**
- The key word "var" is used to **declare variables**
- Examples of variable declaration

```
var myAge;
myAge = 35;
```

JS Assignment

- Variables are **assigned** some values using the equality (=) sign
- Examples

```
myAge = 35;

var thisYear;  // variable declaration
thisYear = "2014"; // assignment
```

JS Comments

- Comments are **good for documentation** in the programme. Why?
- They are **not executed** by the **interpreter**
- The comment starts with "//"
- Examples

```
//This is the beginning of the programme

// var y = 30;
```

JS Data Types

- JS has **dynamic data types**
  - This means that the **data type can change** on the **same variable** in the **same program**

```
var myValue;
var myValue = 10;
var myValue = “My Name”;
```

- **Numbers**
  - 60, 100, 56.89, 569e7, 567e-9
  - They can be integers (10, 1, 89, etc), float (10.0, 89.00, etc)

- **Strings**
  - They are enclosed with double or single quotes
  - “my Name”, ‘Her Name’, “23”, “24.7”, etc

JS Data Types

- **Arrays**
  - They are **declared** with **square brackets** ([])
  - The index starts from **zero by default**

```
var myIntArray = [4, 8, 200, 50];
var myStringArray = [“Name”, “StudID”, “Degree”];

myIntArray[0] = 4;
myStringArray[2] = “Degree”
```

JS Data Types

- **Objects**
  - They are written within **curly braces** ({})
  - The structure is
    - Name: value pairs
    - They are separated by commas

```
var student = {surName:”Wang”, firstName:”Liu”, age:20};
```

JS Data Types

- Objects
  - **Creating objects** with the **“new”** operator key word
  - The “new” key word **must be** followed by a **function invocation**
  - A function used in this way is called **a constructor**
  - This **constructor initialises the newly created object**
  - Core JavaScript consists of built-in constructors eg
    - var obj = new Object(); // creates an empty object similar to using {}
    - var arr = new Array(); // creates an empty array similar to using []
    - var dt= new date(); // creates a Date object with the current date

Example of using “new” key word

```
<!DOCTYPE html>
<html>
<body>
<p id=“newExample"></p>
   <script>
      var person = new Object();
      person.firstName = “Wang";
      person.lastName = “Zhang";
      person.age = 18;
      document.getElementById(" newExample ").innerHTML =
            person.firstName + " is " + person.age + " years old.";
  </script>

</body>
</html>
```

Source: W3 School’s Website

JS Data Types

- **Boolean**
  - There are only two types
    - True and false

```
var myBooleanValue1 = true;
var myBooleanValue2 = false;
```

typeof keyword

- Use the **“typeof”** keyword to **know the data type**
- Examples

```
typeof 23; // returns a number
typeof “Wang”; // returns a string
typeof false; // returns a boolean
```

JS Conditional Statements

- **If statement/ if else statement**

Structure:

```
if (condition) {
   execute the code here if the condition is true
}
```

Examples:

```
if (myAge < 16) {
    driving = “I cannot drive!";
}
```

```
if (myAge < 16) {
    driving = “I cannot drive!";
} else {
    driving = “I can drive!";
}
```

JS Conditional Statements

- **Switch statement**

Structure:

```
switch(expression) {
    case 1:
        code block
        break;
    case 2:
        code block
        break;
    default:
        default code block
}
```

JS Loop Statements

- **For loop statement**

```
for (i = 0; i < 5; i++) {
    myNumbers += “My number is " + i + "<br>";
}
```

- **While loop**

```
while (condition) {
    Execute code block if condition is true
}
```

JS Functions

- JS function is a complete code **that performs certain function as a block**

Create a function:

function functionName(*para1, para2, …….*) {
    return returnValue;
}

- function is a **keyword (or syntax)**
- functionName is the **name** given to the function
- para1, para2, etc are the optional parameters
- return is a **keyword**
- returnValue is the value returned when the function is called

Calling a function:

functionName(*para1, para2*) {
    code to be executed;
}

JS Functions

- Examples
- Calculate sum of two numbers *a* and *b*

```
var mySum = myAddFunction(4, 3);    // Function is called,
return value will end up in mySum

function myAddFunction(a, b) {
   return a + b;          // Function returns the sum of a and b
}
```

JS Functions

- Examples
- Convert Fahrenheit to Celsius

```
function myCelsius(myFahrenheit) {
   return (5/9) * (myFahrenheit-32);
}
document.getElementById(“myDemo").innerHTML = myCelsius(32);
```

JS Functions

- Convert Fahrenheit to Celsius

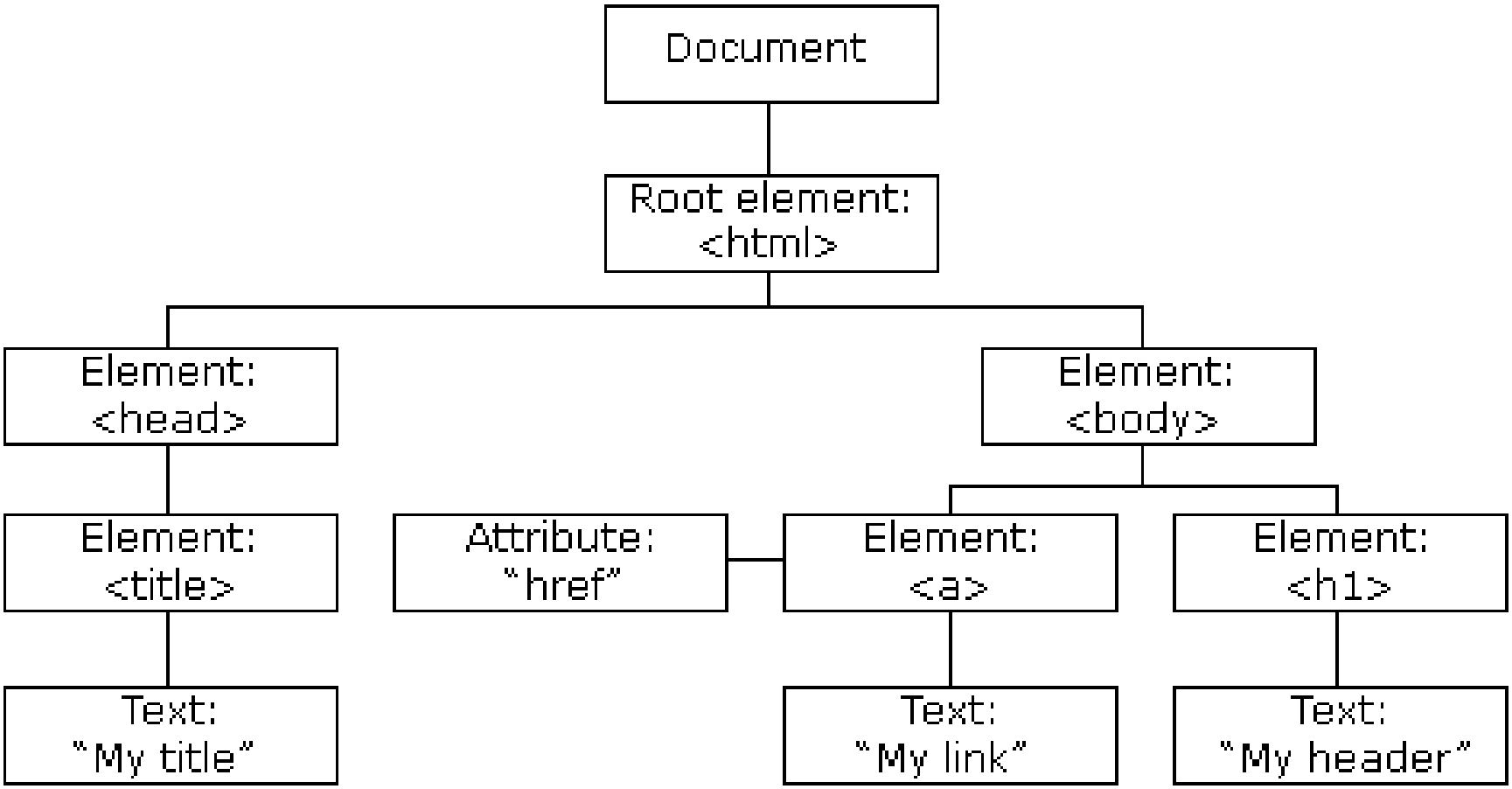
```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Celcius to Fahrenhet</h1>
<p>Insert a number into one of the input fields below:</p>
<p><input id="c" onkeyup="convert('C')"> degrees Celsius</p>
<p><input id="f" onkeyup="convert('F')"> degrees Fahrenheit</p>
<p>Note that the <b>Math.round()</b> method is used, so that the result will be returned as an integer.</p>

<script>
function convert(degree) {
   var x;
   if (degree == "C") {
      x = document.getElementById("c").value * 9 / 5 + 32;
      document.getElementById("f").value = Math.round(x);
   } else {
      x = (document.getElementById("f").value -32) * 5 / 9;
      document.getElementById("c").value = Math.round(x);
   }
}
</script>

</body>
</html>
```

Source: http://www.w3schools.com/js/tryit.asp?filename=tryjs_celsius

The HTML Document Object Model (DOM)



Source: W3C (World Wide Web consortium) Site

DOM Methods

- **A method is a variable** that **performs an action** such as adding, subtracting, uploading, etc
- Some built in methods

getElementById

- Use this property to get the contents of the element

innerHTML

- This method access an HTML element via the Id of the element

DOM Methods

- Example

```
<html>
<body>

<p id=“myElement"></p>

<script>
document.getElementById(“myElement").innerHTML = "Hello World!";
</script>

</body>
</html>
```

DOM Nodes

- HTML Elements are known as Nodes
- You can create new HTML elements (Nodes)
- Example of creating ne Node

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);
var element = document.getElementById("div1");
element.appendChild(para);
</script>
```

Source: W3C (World Wide Web consortium) Site

DOM Nodes

- Explanations of the code

First **create a new** <p> **element**:

```
var para = document.createElement("p");
```

To add text to the <p> element, you must **create a text node** first:

```
var node = document.createTextNode("This is a new paragraph.");
```

You must **append the text node** to the <p> element:

```
para.appendChild(node);
```

You find an existing element:

```
var element = document.getElementById("div1");
```

The you finally **append the new element** to the existing element:

```
element.appendChild(para);
```

DOM Nodes

- Remove existing elements

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
   var parent = document.getElementById("div1");
   var child = document.getElementById("p1");
   parent.removeChild(child);
</script>
```

Source: W3C (World Wide Web consortium) Site

DOM Node List

- The **“length” property** is used **to define the number of nodes** in a node list
- A **node list** is like **an array** consisting of a **collection of nodes**
- Use the “getElementsByTagName()” method to return a **node list**.
- Example

```
var myNodeList = document.getElementsByTagName("p");
```

- Use the “length” property to get define the number of nodes in a node list

```
var myNodeList = document.getElementsByTagName("p");
document.getElementById(“myNodes").innerHTML = myNodeList.length;
```

Classes and Modules

- It is important to describe **objects** in terms of **their classes**
- JavaScript **classes** are based on **JavaScript prototypes** and **inheritance**
- So, **a class** is a **set of objects** that **inherit properties** from the **same prototype object**
- An important reason for using classes in any program is for **modularity**
- The goal of modular programming is to allow **large tasks implemented by codes** to be **divided** into simple and understandable smaller parts **called modules**

Example of Classes

```
<!DOCTYPE html>
<html>
<body>
<p id=“classExample"></p>
<script>
function Person(first, last, age) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
}
Person.prototype.nationality = “Chinese";
var myFather = new Person(“ZHANG", “Wang", 18);
document.getElementById(" classExample ").innerHTML =
"My father is " + myFather.nationality;
</script>
</body>
</html>
```

Source: W3 School’s website

JS Examples

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
  window.alert(5 + 6);
</script>

</body>
</html>
```

Source: W3C (World Wide Web consortium) Site

JS Examples

```
<!DOCTYPE html>
<html>
<body>

<p>Use of Expressions for Computation</p>

<p id=“myProduct"></p>

<script>
document.getElementById(“myProduct").innerHTML = 5 * 10;
</script>

</body>
</html>
```

Source: W3C (World Wide Web consortium) Site

Class Work

- Work in 3 Groups
- Group 1: Write a simple JS programme to take in student number, student name and print them
- Group 2: Write a simple JS program to calculate the area of a circle (pi r square=(22/7)*r*r)
- Group 3: Write a simple JS program to take two variables a and b and compute their sum

Summary

- Revision on JSP
- JavaScript Introduction
- HTML and JavaScript
- Examples