TEESSIDE UNIVERSITY

SCHOOL OF COMPUTING

ICA SPECIFICATION Lab Work

This form should be attached to the front of the ICA specification and include all details.

Module Title: Client Side Scripting	**Module Leader**: Barry Hebbron
	Module Code: **MUL1062-N**
Assignment Title: Client Side Scripting Part 2 2016	**Deadline Date:** 23/4/18
	Deadline Time: Midnight
	Submission Method: ONLINE **ONLY: into your IWS space with a named folder created for each task.**

FULL DETAILS OF THE ASSIGNMENT ARE ATTACHED

INCLUDING MARKING & GRADING CRITERIA

Assessment Marking Criteria

%	**Overall Grade**
40% and above	Pass: A, B, C, D
Less than 40%	Fail

Client Side Scripting: In-Course Assessment

This ICA requires you to undertake Task 1 through to Task 9 and is worth 80% of the overall mark for this module.

Hand-In Requirements

ONLINE Submission Notes:

You are required to upload ALL your files to a sub-directory called CSS2016 with each answer clearly visible as a response to a specific task – eg Task 1.doc and Task9.html or Task9.js

Upload ALL your work to your Intranet Work Space inside a directory called **CSS2018.**
Create a sub folder for each task

Task1 – Task1.doc / or Task1.ppt

Task2 – 9 , HTML/CSS & JS FIles

NO PRINTOUTS REQUIRED.

ALL Code MUST be commented.

Scenario: The Worksheets

Throughout the year you have been required to undertake a collection of tutorial tasks within the worksheets associated with each lecture.

This ICA is an opportunity for you to make visible this work for a selection of those tasks.

The list of tasks for this ICA is in the next section.

Please NOTE – A Separate FILE or FOLDER is required for EACH Task…

Task1.doc (ppt) / Task2.html / Task3.html….

There are special instructions for Task 9 – Read them.

Task List

The following must be completed.

Task 1 DOM (6 Marks)

Create a DOM (a diagram) for this HTML code:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Master Detail AJAX with DropDown - My ASP.NET MVC Ap"</title>
    </head>
    <body>
        <header>
          <p class="site-title">Playing with Master Detail Drop Downs</p>
          <section id="login">
     <ul>
        <li><a href="/Account/Register" id="registerLink">Register</a></li>
        <li><a href="/Account/Login" id="loginLink">Log in</a></li>
    </ul>
                        </section>
                        <nav>
                            <ul id="menu">
                                <li><a href="/MasterDetailDDL/MasterV1">Version 1</a></li>
                                <li><a href="/MasterDetailDDL/MasterV2">Version 2</a></li>
                                <li><a href="/MasterDetailDDL/MasterV3">Version 3</a></li>
                                <li><a href="/MasterDetailDDL/MasterV4">Version 4</a></li>
                                <li><a href="/Grade/MasterV5">Version 5</a></li>
                                <li><a href="/Grade/MasterV6">Version 6</a></li>
                                <li><a href="/Grade/MasterV7">Version 7</a></li>
                                <li><a href="/Grade/MasterV8">Version 8</a></li>
                                <li><a href="/Grade/MasterV9">Version 9</a></li>

                            </ul>
                        </nav>
        </header>
        <div id="body">
    <section class="featured">
            <hgroup class="title">
                <h1>Master Detail AJAX with DropDown.</h1>
            </hgroup>
            <p>
                In this Demo, I have broken down the solution into 9 separate versions with relatively small steps between each. Each
step allows us to introduce an idea that is Key to MVC.
            </p>
    </section>
<section class="content-wrapper main-content clear-fix">
<ol class="round">
    <li >
Version 1.  View 1 a HTML Only Select Element posting back to a controller that writes a simple result to separate View.
    </li>
    <li >
       Version 2. Behaves the same as version 1 but introduces abstraction by using html helpers and the SelectList Class
    </li>
<li >
Version 3. This behaves the same as version 2 but introduces a ViewModel in as the mechanism for coupling the Controller to the Views.
</li>
<li>Version 4. Is different from the previous 3 versions as it uses a single view to capture the request and subsequently display the
data requested.</li>
<li>Version 5. Is the first attempt at single page Master-Detail, in that a selection of a single item in the Master collection requires
filtering of a related child collection and the listing the result of that selection-filtering.</li>
<li>Version 6. Is slightly more ambitious that Version 5 in that it will initiate a post back when the drop down list detects a change
in its selection, hence removing the need for a submit button.</li>
<li>Version 7.  Introduces the Partial View concept. Version 6 provided a Master-detail mechanism with a single view, but this view is
becoming increasingly complex. One mechanism available to us that helps us manage this complexity and encourage reuse of view code is
the Partial View.</li>
<li>Version 8. In our previous versions we had to regenerate the entire view via a post back if the content should change. AJAX allows
elements of views to be updated asynchronously by exchanging data with the controller behind the scenes. This means that it is possible
to update parts of a view, without having to regenerate the whole page. This example reintroduces the submit button uses but AJAX to
provide a more efficient and sophisticated user experience.</li>
<li>Version 9.  In this our final version for now, we are going to make visible JQuery post back code that does the work, and remove
(once again) the need for a submit button.</li>
</ol>
</section>
</div>
<footer>
<p>&copy; 2013 - B.D. Hebbron SCM Teesside University</p>
</footer>
</body>
</html>
```

Task 2 Understand JavaScript and CSS (6 Marks)

Use the following JavaScript/HTML/CSS code to build a simple application. Then describe by commenting each line how it works?

```
<!DOCTYPE html>
<head>
    <title>Please Comment</title>
         <style>
        body {
}

ul
{
list-style-type:none;
margin:30px,0,0,0;
padding:0,0,0,0;
}

li
{
display:inline;
padding-left:25px;
padding-right:25px;
}

#demo {
    position:relative;
    top:10px;
    left:100px;
    width:200px;
    height:200px;
    background-color:#0094ff;
}

    </style>
<script type="text/javascript">
        window.onload = initall;

        function initall() {
            document.getElementById("siz").onmouseover = changesizeover;
            document.getElementById("siz").onmouseout = changesizeout;
        }

        function changesizeover() {
            document.getElementById("demo").style.width = "50px";
        }

        function changesizeout() {
            document.getElementById("demo").style.width = "100px";
        }
    </script>
</head>
<body>
    <div id="demo">
        JS and CCS DEMO
    </div>
    <ul>
      <li><span id ="col">colour</span></li>
      <li><span id="siz">size</span></li>
      <li><span id="vis">visibility</span></li>
      <li><span id="pos">CLICK position</span></li>
</ul>
</body>
</html>
```

Task 3 Modify JavaScript and CSS (6 Marks)

Modify the code in TASK 2 so that when the first span tag in the DOM experiences a mouse over event the element <div id="demo"> changes both width by 50px and to a new colour. When the span experience a mouse out return to its original dimensions and colour.

Task 4 Extend JavaScript and CSS (6 Marks)

Extend the code in TASK 2 so that the element <div id="demo"> changes its visibility when the visibility span is **clicked. (Element should remain in the DOM)**

Task 5 Understand JavaScript and HTML (6 Marks)

Describe what this JavaScript/HTML application demonstrates, and then in detail (each line) how it works?

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

   <title></title>

    <link rel="stylesheet" type="text/css" href="Style/demo.css" />

  <script type="text/javascript">

      window.onload = init;

function init() {

            document.getElementById("s1").onclick = part1;

           document.getElementById("s2").onclick = part2;

           document.getElementById("s3").onclick = part3;

}

      function part1() {

           var demo = document.getElementById("demo");

           demo.innerHTML=  "It is a period of civil war. Rebel spaceships, striking from a hidden base, have won their first victory against the evil Galactic Empire. ";

      }


function part2() {

           var demo = document.getElementById("demo");

           demo.innerHTML = "During the battle, Rebel spies managed to steal secret plans to the Empire's ultimate weapon, the Death Star, an armoured space station with enough power to destroy an entire planet.";

      }
    function part3() {
        var demo = document.getElementById("demo");
        demo.innerHTML = "Pursued by the Empire's sinister agents, Princess Leia races home aboard her starship, custodian of the stolen plans that can save her people and restore freedom to the galaxy....";
    }

  </script>
</head>
<body>
  <div id="demo">
     A long time ago, in a galaxy far, far away..
  </div>
  <ul>
   <li><span id ="s1">Part 1</span></li>
   <li><span id="s2">Part 2</span></li>
   <li><span id="s3">Part 3</span></li>
   <li><span id="s4">Reset</span></li>
</ul>
</body>
</html>
```

Task 6 Extend JavaScript and HTML (6 Marks)

Modify the HTML and JavaScript code in Task 5 so that a Part 4 element generates some new narrative and when the Reset element is clicked the page displays the original message “A long time ago, in a galaxy far, far away..

Task 7 Manipulating the DOM (6 Marks)

Using the HTML framework supplied in other tasks, build an application that uses the following code. Ensure you comment each line, and then extend its behaviour so that the element s4 adds to the content.

```
window.onload = init;

    function init() {

        document.getElementById("s1").onclick = part1;

        document.getElementById("s2").onclick = part2;

        document.getElementById("s3").onclick = part3;

    }

    function part1() {

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

        var node=document.createTextNode("It is a period of civil war. Rebel spaceships, striking from a hidden base, have won their first victory against the
evil Galactic Empire. ");

        para.appendChild(node);

        var element=document.getElementById("demo");

        element.appendChild(para);

    }

    function part2() {

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

        var node = document.createTextNode("During the battle, Rebel spies managed to steal secret plans to the Empire's ultimate weapon, the Death
Star, an armored space station with enough power to destroy an entire planet.");

        para.appendChild(node);

        var element = document.getElementById("demo");

        element.appendChild(para);

    }



    function part3() {

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

        var node = document.createTextNode("Pursued by the Empire's sinister agents, Princess Leia races home aboard her starship, custodian of the stolen
plans that can save her people and restore freedom to the galaxy....");

        para.appendChild(node);

        var element = document.getElementById("demo");

        element.appendChild(para);

    }
```

Task 8 (12 Marks)

Redesign your solution to **Task 6** (not Task 7) to take into account two key development principles: DRY and SOC.

It is important that you comment the code justifying the decisions you take and demonstrating you understand the code delivered.

See marking criteria for guidance.

Task 9 (46 Marks)

This will be developing of Bingo Card / Bingo Caller Application using the following sequence of steps. YOU MUST FOLLOW THESE STEPS AND PRODUCE A SEQUENCE OF PROTOTYPES. SO YOU ARE REQUIRED TO SUBMIT.

Task9_ BingoCardPhase1_ Version1.js / Task9_ BingoCardPhase1_ Version2.js / Task9_ BingoCardPhase1_ Version3.js …..

Bingo Card Phase 1

- Bingo Card Version 1: Generate a Random Number for a single cell.
- Bingo Card Version 2: Generate any Random Number for every cell.
- Bingo Card Version 3: Refactor Bingo Card Version 2 into an MVC pattern of Objects

Bingo Caller Phase 2

- Bingo Caller Version 1: Generate and Display a Sequence of Unique Numbers between 1 and 78.
- Bingo Caller Version 2: Refactor Bingo Caller Version 1 into an MVC pattern of Objects.

Bingo Card Phase 3

- Bingo Card Version 4: Start from Bingo Card Version 2 (or Version 3), Generate a numbers for every cell without duplicates.
- Bingo Card Version 5: Refactor Version 4 into an MVC pattern of Objects
- Step 5 - Bingo Card Version 6: Generate a numbers for every cell without duplicates, each column displaying the correct set of numbers.
- Step 6a- Bingo Card Version 7: Version 6 refactored into an MVC pattern of objects.

  – Step 7 – Version 8, Re-Engineering the Bingo Card to include a view model.

Integrated Application Phase 3

- Step 8 – Version 9, integrating the Bingo Card & Bingo Caller into a single Application.

Key points:

You are required to submit all versions for assessment with each clearly labelled as version1.js …. Version9.js within a directory of your **CSS2017 space eg.. CSS2017/Task9/…**

You are required to comment every line of every version demonstrating you understand every line of code submitted.

Assessment Criteria

You must demonstrate all learning outcomes during this assessment

Knowledge & Understanding
1. Demonstrate knowledge of basic client-side programming. 2. Demonstrate knowledge of fundamental object based concepts such as classes, objects, methods and encapsulation.
Cognitive & Intellectual Skills
3. Design an algorithmic solution to simple problem specification. 4. Select and use the appropriate JavaScript objects for a given task.
Practical & Professional Skills
5. Build an efficient JavaScript solution to a simple problem specification. 6. Test a JavaScript application using a set of test cases.
Key Transferable Skills
7. Use a range of tools to design, build, debug and test a simple software application. 8. Produce software documentation using source code comments.

These learning outcomes have been referred to throughout the module and will be the basis for your assessment criteria.

Task	Complete and Correct and clearly understood	Good attempt, may not be totally correct or understood	Adequately Demonstrated	Some Attempt
1 to 7	6	5	4	3
	Refactored using an MVC pattern that both delivers DRY and SOC architecture	Refactored so the code attempts to delivers DRY and SOC architecture	Refactored so the code attempts to delivers DRY or SOC architecture	Some attempt at refactoring
Task 8	12	10	8	6

Task 9	Complete and Correct and clearly understood	Good attempt, may not be totally correct or understood	Adequately Demonstrated	Some Attempt
Card Version 1	1	0	0	0
Card Version 2	1	0	0	0
Card Version 3	2	1	0	0
Caller Version 1	4	3	2	1
Caller Version 2	4	3	2	1
Card Version 4	3	2	1	1
Card Version 5	4	3	2	1
Card Version 6	5	4	3	1
Card Version 7	6	5	4	2
Card Version 8	7	5	4	2
Card Version 9	9	6	4	2

Note the following:

Code will be tested using Firefox.

Demonstration of understanding will be your use of comments.

Correctness will take into account the structure of your code (consider the DRY principle)