

CONTENTS

INTRODUCTION	1
PEER REVIEW	1
TABLES	1
WEB FORMS	3
HINT	5
WEB ACCESSIBILITY	5

INTRODUCTION

The initial part of the practical session is for you to carry out your peer review as outlined in the practice assignment specification. Attendance will be recorded at these sessions and you are expected to attend. This week we look in more depth at tables and how to manipulate them, we consider web forms and web accessibility.

PEER REVIEW

The practical will be organised so that you all will have the opportunity to carry out a peer review of your sitemaps. This is your chance to crystalize your ideas for your application and how you intend to set out the pages that will form the core of your application. You are expected to record who you spoke to, what constructive feedback were made and how you changed your ideas to take account of that feedback. A template is available on the DLE should you require.

TABLES

Download the file dlr_evenings.html from the folder 09 Files. After the initial paragraph in the section add a table with a class "schedule". Add 3 rows. Inside the first row of the table, add eight table header <th> elements and label them as follows:

- Time
- Mon
- Tue
- Wed
- Thu
- Fri
- Sat
- Sun

In the second row of the table insert a table header element with 6:00 PM in between the open and closing tags. Insert one table data <td> element after the initial table header element with the value "National News" and make it span 7 columns.

In the third row of the table insert a table header element with 6:30 PM in between the open and closing tags. Insert one table data <td> element after the initial table header element with the value "World News" and make it span 7 columns.

Save and view.

Open the dlr_tables.css file from the folder 09 Files. Add a style rule that will place a border around a table that has a class attribute of “schedule”

```
table.schedule {  
    background: white;  
    border : 10px outset rgb(153,0,153);  
    border-collapse: collapse;  
    font-size: 0.75em;  
    width: 100%;  
}
```

Within the Table Cells style section add the following style rule to place a border around each header cell and data cell within the tables belonging to the schedule class.

```
table.schedule th, table.schedule td {  
    border: 1px solid gray  
}
```

Link the dlr_tables.css stylesheet with the dlr_evenings.html page.

Save and view.

Add a new row to the table with a table header and says 7:00 PM. Insert the following:

- A table data element that spans 2 rows that says Opera Fest
- A table data element that spans 2 rows that says Radio U
- A table data element that spans 2 rows that says Science Week
- A table data element that spans 2 rows that says The Living World
- A table data element that says Word Play
- A table data element that says Agri-Week
- A table data element that spans 2 rows that says Folk Fest

Add another new row that has a table header and says 7.30 PM. Insert two more table data elements one saying Brain Stew, the other Bismarck Forum

Save and view.

Directly after the starting table tag insert a caption element saying All Times GMT.

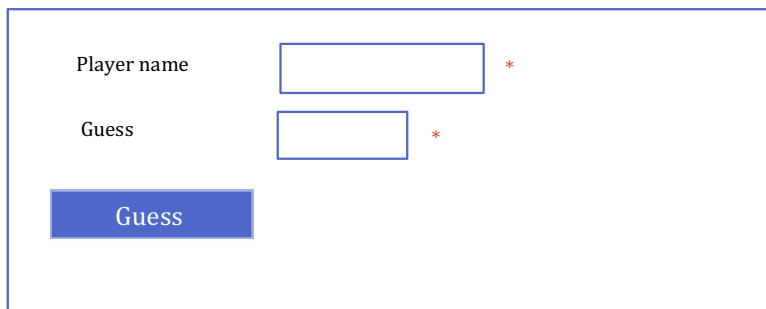
Finally, add a style to the dlr_tables.css stylesheet

```
table.schedule caption {  
  caption-side: bottom;  
  text-align: right;  
}
```

WEB FORMS

In this exercise we will create a web form to our existing application developed last week. Our new page will be part of a game where a player has to guess a number that the computer has randomly generated between 1 and 6, and the lights in SMB109 will display the guesses.

Our web form may well look something like this:



The image shows a web form layout within a blue border. It contains two rows of input fields. The first row has the label "Player name" on the left, a text input box in the middle, and a red asterisk on the right. The second row has the label "Guess" on the left, a text input box in the middle, and a red asterisk on the right. Below these fields is a blue rectangular button with the text "Guess" in white.

For this we need to use the `<form>` element, a couple of input boxes which are `<input>` elements with a type of "text", a couple of labels for the input boxes and a submit button which is an `<input>` element with a type of "Submit".

In this example we will not be using server-side coding but JavaScript to control and validate our form.

1. Add a new html file to your application and give it a name something like "game.html".
2. Use your layout from before to keep a consistent look and feel.
3. Download the new treasure.js file and replace the one from last week. Take a look at the changes – the function switchLight has been added and the toggleLight function has been amended to make use of the new function.
 - a. This is a little bit of refactoring carried out so that we do not have repeated code in different places. This is part of the development process that you need to carry out as you create your code. Make sure you do not repeat code – the principle of DNR
4. Add a div where your form will go and add the components needed to make the form shown above. Ensure you use the "required" attribute and place an asterisk to indicate which fields are mandatory.
5. Take a look at the Bootstrap classes that make forms look good and apply those you think are appropriate - <https://getbootstrap.com/docs/4.0/components/forms/>
6. Turn to your JavaScript file and create the function given below that will generate a random number between 1 and 6. This uses the Math.Random and Math.Floor to generate the number you need. See more at https://www.w3schools.com/js/js_random.asp

```
function generateRandomNumberToGuess()
{
    var NumberToGuess = Math.floor(Math.random() * 6)+1;
    return NumberToGuess;
}
```

7. Inside the document.ready() function, create a variable that will call your new function and hold it in memory. See code example below.
8. Now use the JQuery selector to react to the click event of your SubmitGuess button. You are looking to evaluate the value of the guess against the random number already generated. I suggest you create a new function to compare the two which returns a Boolean. Use the result of this function to decide whether you turn your light on or return an alert to tell the user it is a wrong guess.
 - a. Your light in the lab will depend on the row you are sitting at. There are six lights and six rows. If you are looking at the lights the top left of the screen is 1 and they number down and then up. Eg: Top left = 1, middle left = 2, bottom left = 3, top right = 4 etc.
 - b. The reason for doing it this way is because there will be a number of you in the lab and you will need to communicate with the others on your row to take it in turns to use the lights. Otherwise how will you know if your light has been turned on or not!

```
var GuessMe = generateRandomNumberToGuess();

$(html: '#SubmitGuess').click(handler: function()
{
    if(guess($(html: '#Guess').val(), GuessMe))
    {
        switchLight(rowNumber, boolValue: true);
    }
    else
    {
        window.alert("Sorry, you got it wrong");
    }
})
```

9. Create a new function which will compare the guessed value to the generated value. Have two inputs to the function and return the result of the comparison as a Boolean. Suggested code below.
 - a. Note: I am using parseInt as the value that will come in from the JQuery selecting the input box will be a string value – it automatically takes it as a string. Because I am dealing with numbers, I want them to behave like numbers so I use parseInt. However – what do you think might be a problem here? What would happen if somebody typed in a word?

```
function guess(guessedNumber, numberToGuess)
{
    return parseInt(guessedNumber) == parseInt(numberToGuess);
}
```

HINT

When developing your application there will be times when the information is cached inside the browser. In Chrome you can disable the cache when developing by doing the following:

Settings

Preferences

Workspace

Blackboxing

Devices

Throttling

Geolocations

Shortcuts

Preferences

Elements

- ☐ Show rulers
- ☐ Show user agent shadow DOM
- ☒ Word wrap
- ☒ Show HTML comments
- ☒ Reveal DOM node on hover
- ☒ Show detailed inspect tooltip

Network

- ☐ Preserve log
- ☐ Enable request blocking
- ☒ Disable cache (while DevTools is open)
- ☐ Color-code resource types
- ☐ Group network log by frame
- ☐ Force ad blocking on this site

WEB ACCESSIBILITY

Carry out some research on available tools for testing the accessibility aspects of a website. You might like to start here <https://www.softwaretestinghelp.com/accessibility-testing-tools/>

Evaluate the usefulness of the free tools. What help do they give?