# Marking Justification

|  |  |  |
| --- | --- | --- |
| Criteria | Score | Reason |
| Presentation | D | The code parts are better when print out due to the word check within word. |
| Rich text | D | The code parts are clear and well formatted, different styles are used in this document. |
| Explanation | D | Personally, I think this note can be used for idea learning or quick switching manual from other developers with other languages skills.  Maybe reused as one or two sections in a comprehensive printed book. |
| Note Summary | HD | My note includes facts, information from other resources, as well as the tests from different codes for explaining the concept and researching some topics. |
| Lab Exercises | HD | Every weekly exercise is delivery in time and correct. |
| Project Progress | D | The project is developed as expected |

Contents

[Marking Justification 1](#_Toc513816013)

[Week 1 3](#_Toc513816014)

[Theory Note 3](#_Toc513816015)

[Lab Exercise Answers 3](#_Toc513816016)

[Section A 4](#_Toc513816017)

[Section B 4](#_Toc513816018)

[Section C, D 5](#_Toc513816019)

[Week 2 6](#_Toc513816020)

[Theory Note 6](#_Toc513816021)

[Lab Exercise Answers 7](#_Toc513816022)

[Week 3 13](#_Toc513816023)

[Theory Note 13](#_Toc513816024)

[Lab Exercise Answers 13](#_Toc513816025)

[Week 4 16](#_Toc513816026)

[Theory Note 16](#_Toc513816027)

[Lab Exercise Answers 17](#_Toc513816028)

[Excise A 17](#_Toc513816029)

[Excise B 17](#_Toc513816030)

[Excise C 19](#_Toc513816031)

[Excise D 20](#_Toc513816032)

[Week4c.js 21](#_Toc513816033)

[ui.js 21](#_Toc513816034)

[Project Progress 21](#_Toc513816035)

[Week 5 22](#_Toc513816036)

[Theory Note 22](#_Toc513816037)

[Lab Exercise Answers 23](#_Toc513816038)

[Excise A 23](#_Toc513816039)

[Excise B 24](#_Toc513816040)

[Project Progress 26](#_Toc513816041)

[Week 6 27](#_Toc513816042)

[Theory Note 27](#_Toc513816043)

[Lab Exercise Answers 29](#_Toc513816044)

[Project Progress 32](#_Toc513816045)

[WEEK 7 33](#_Toc513816046)

[Project Progress 33](#_Toc513816047)

[WEEK 8 33](#_Toc513816048)

[Project Progress 33](#_Toc513816049)

[WEEK 9 34](#_Toc513816050)

[Project Progress 34](#_Toc513816051)

[References 35](#_Toc513816052)

# Week 1

## Theory Note

The concept of web contains three general concepts: Internet, Servers, addresses and webs. Internet is the fundamental structure of modern technologies, servers are providing services and processing data. And in order to access certain servers, addresses are needed. Addresses have two types. The first type is URL, which is designed for easy remembering and contains specific query information, for example, [www.google.com.au](http://www.google.com.au) is considered easier to remember other than “216.58.196.131”. Another one is IP address, which is allocated to your workstation and servers by ISPs and IP address pools.

The web includes HTTP/HTTPS, HTML, CSS and JavaScript. Generally speaking, a website is constructed with HTML, CSS and JavaScript, and delivery through HTTP/HTTPS to the end users.

For mobile apps (web apps), while they are using the technologies similar to websites, their user interface and user experience also need to be considered carefully. Like (Google, 2018)and (Tom May, 2018), individual programmers and development companies need to spend more resources on design part than those in Symbian era (Hrissan, 2005). User interface includes layer design, spaces, fonts and colours.

There are many popular colour styles and guidelines on the internet, for example, material design concept from Google, flat design concept from Microsoft, in addition, many theme websites provide pre-defined colour sets for implement, which significant lower the requirement of design.

## Lab Exercise Answers

### Section A

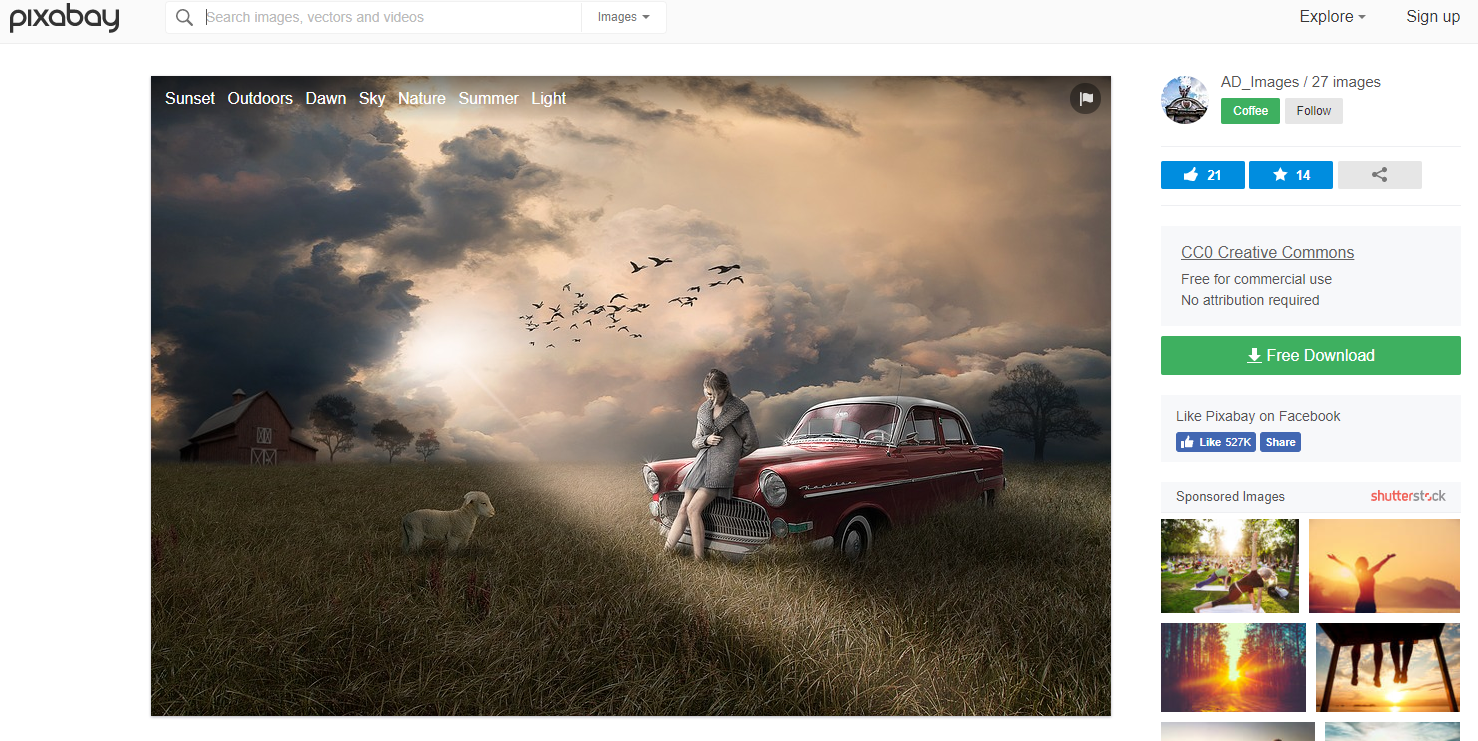
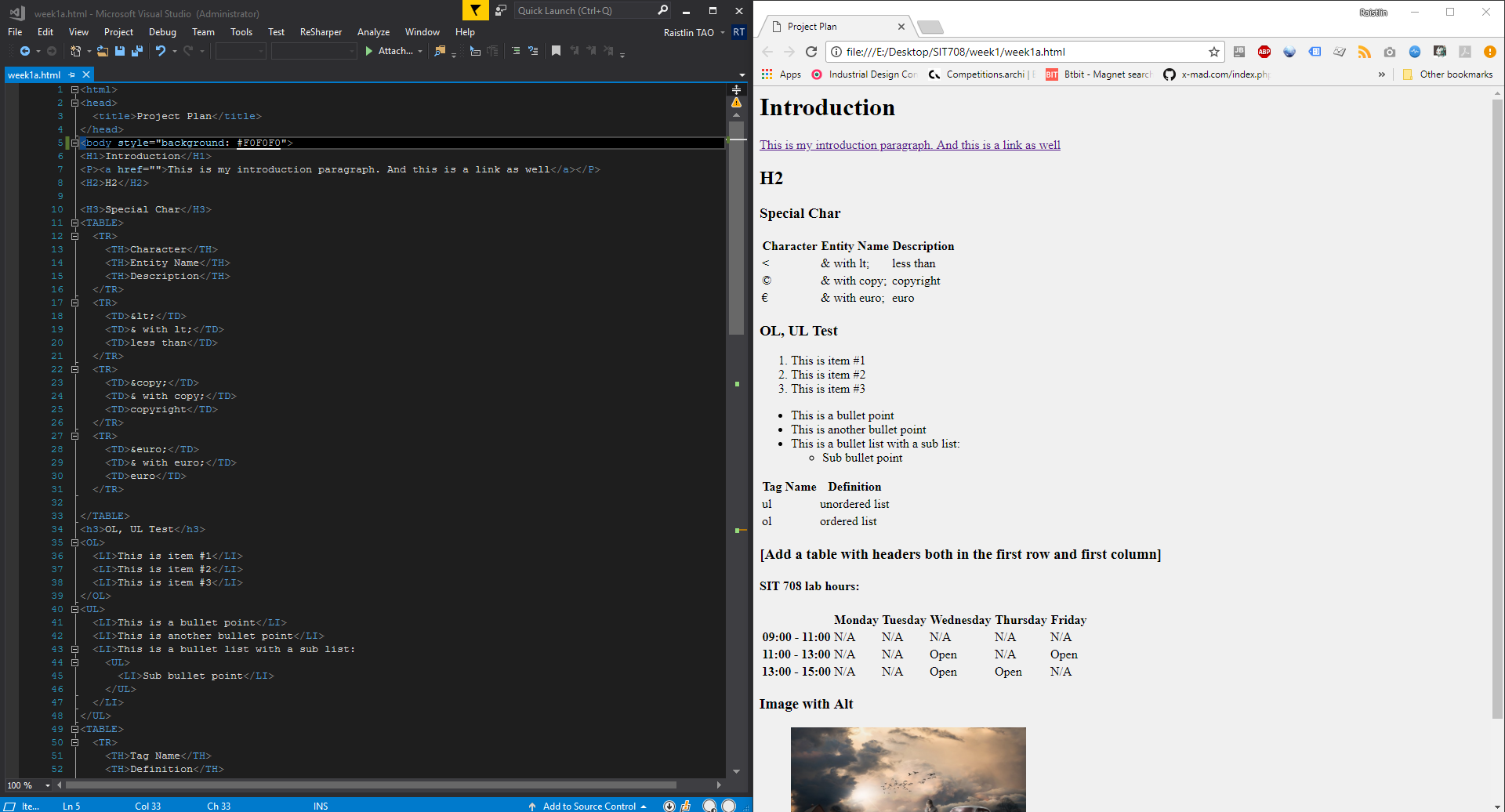


Figure 1 Screenshot of CC0 picture.

### Section B



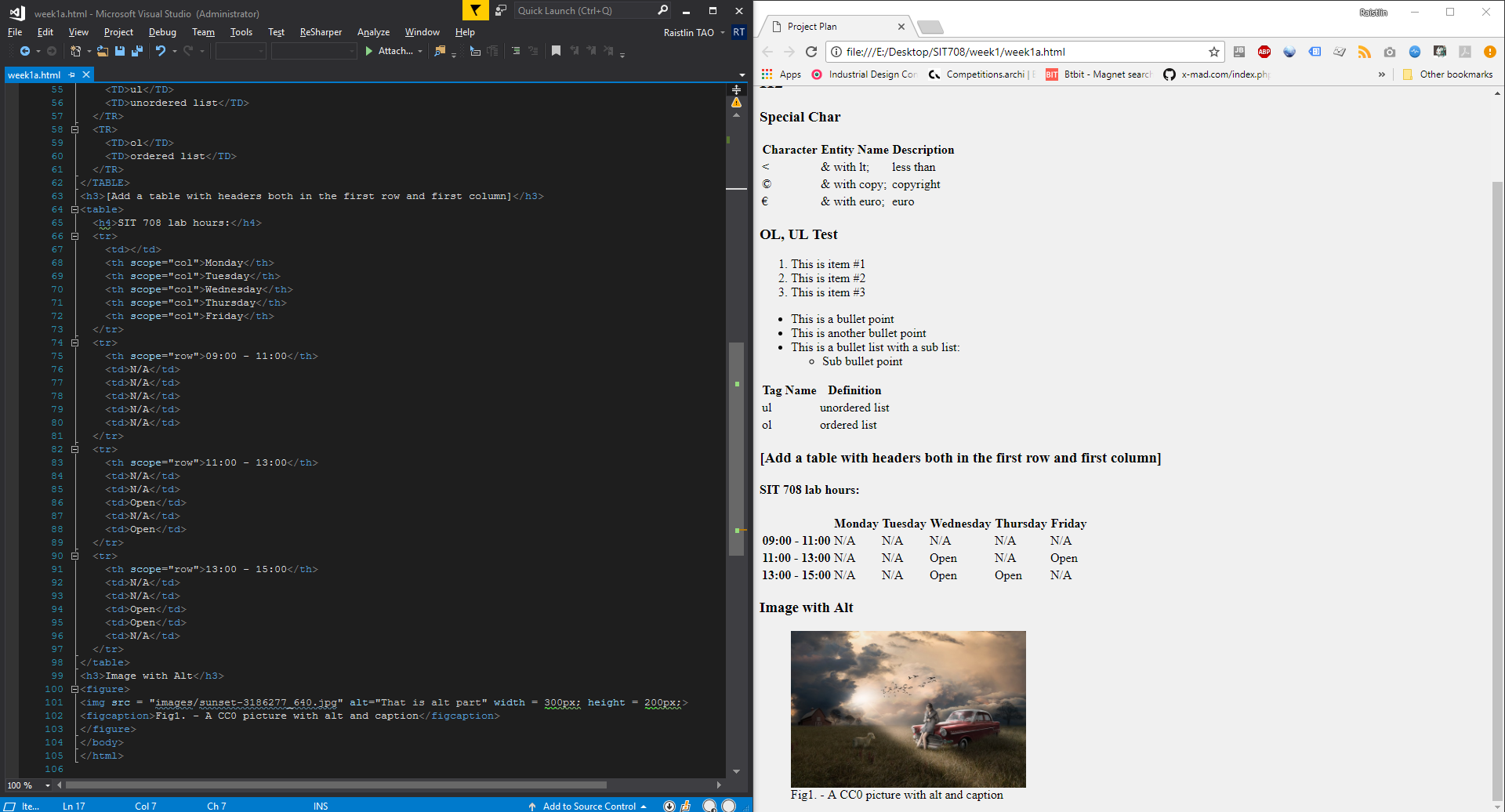
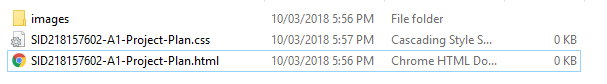


Figure 2,3 Screenshot of HTML coding

### Section C, D



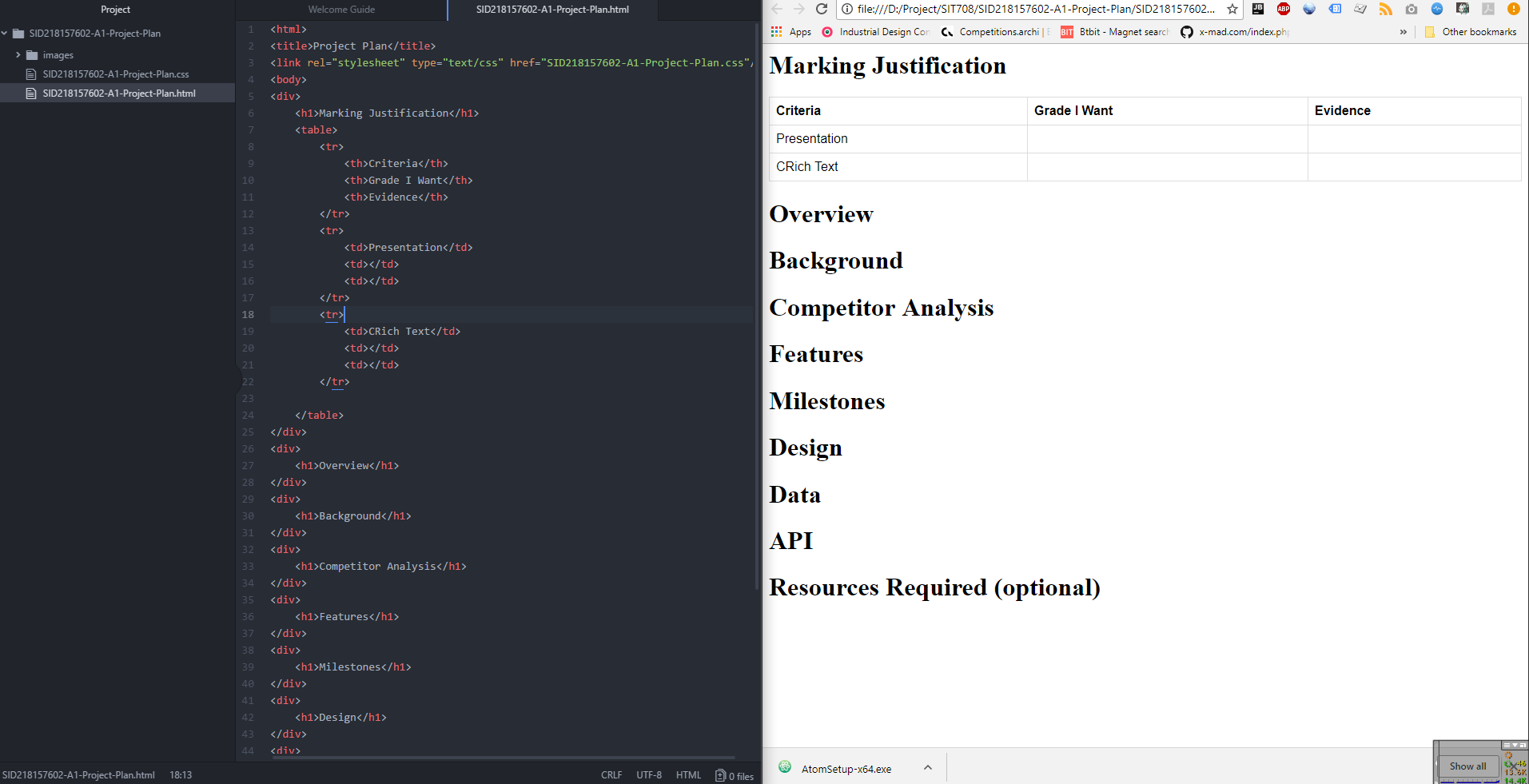


Figure 3 Screenshot of Project Plan Basic

# Week 2

## Theory Note

Style and colour consistency is key to professionalism in websites, both desktop and mobile application, as well as the book designs.

In order to archive code reuse and inheritance like object-oriented programming, we use Cascading Style Sheets in HTML Generally, users can place CSS definitions within the HTML or write definitions in external files and including them at the head part of HTML file. A CSS file can list any number of style rules, normally, there are three important parts in a CSS file: selector, property and value. For example, in order to define one certain element in CSS, we can use multiple properties in one selector:

H5 {

Margin-top:10px;

font-size: 12px;

}

This define H5 element’s distance before top is 10px and the font size is 12px.

The CSS can define HTML elements’ sizes, colours, responsive media rules and fonts

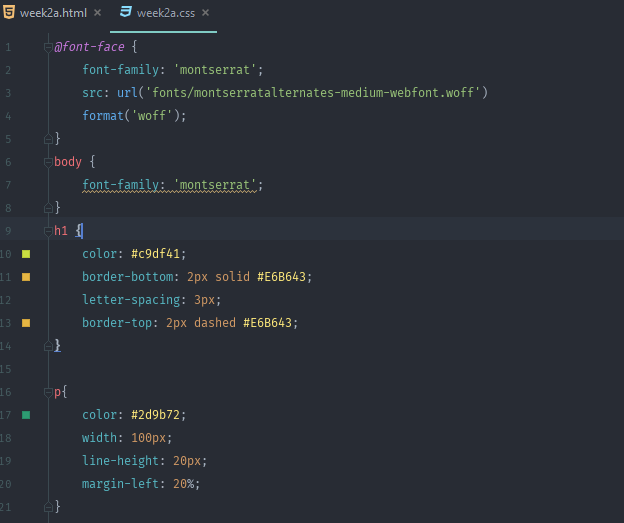
Personally, I think the use of CSS is coming from the OOP concept:

1. The CSS can be reused after one project
2. Even users can add “style” in HTML element directly, put them into one file is best practice.
3. Switch CSS can directly change the website, it is resource-wise by doing so.

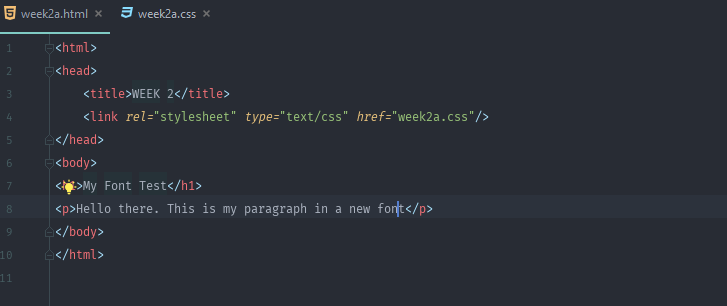
## Lab Exercise Answers



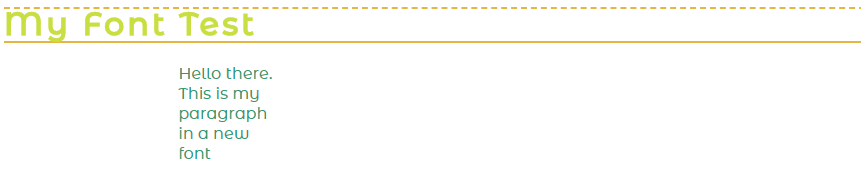
CSS Screenshot:



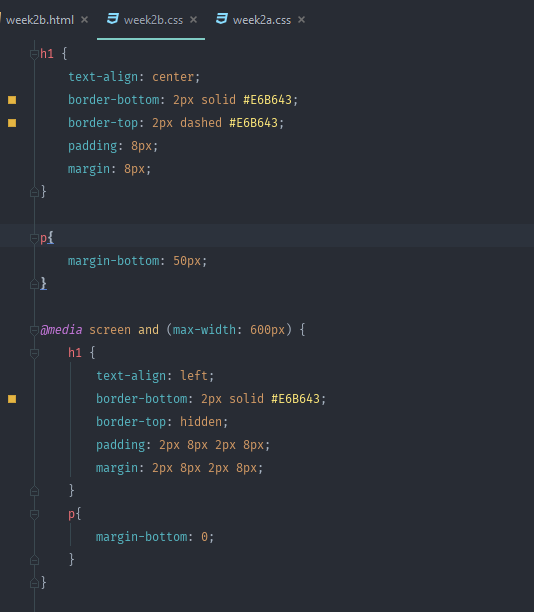
HTML:



WEBPAGE:



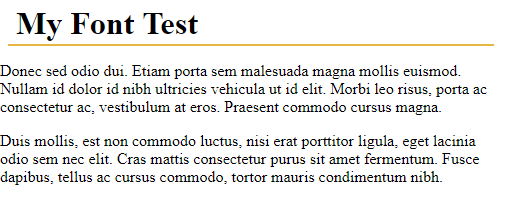
CSS B:



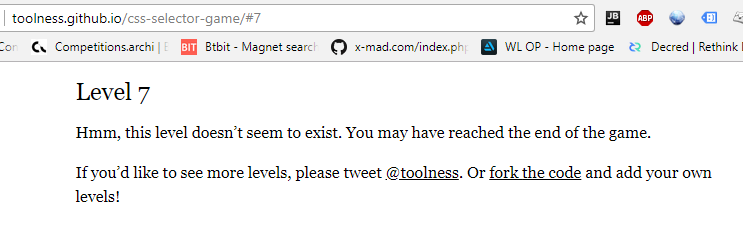
WEBPAGE:



WEBPAGE <600:



C http://toolness.github.io/css-selector-game/

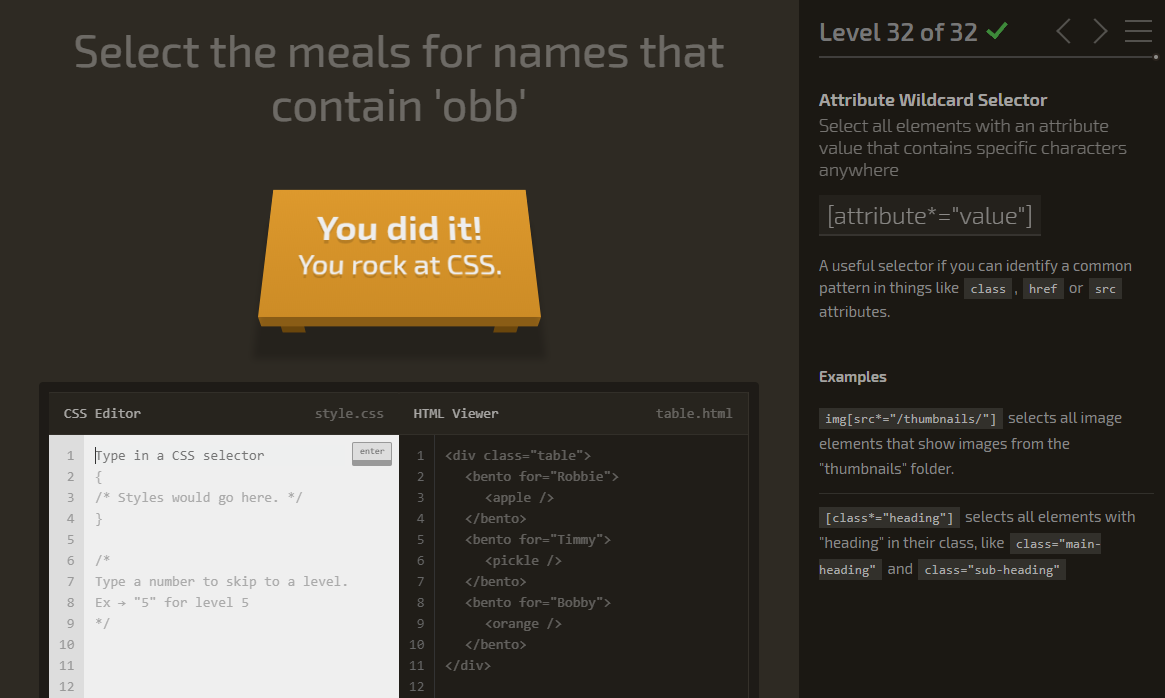


If you want select element with id, use ‘#’;

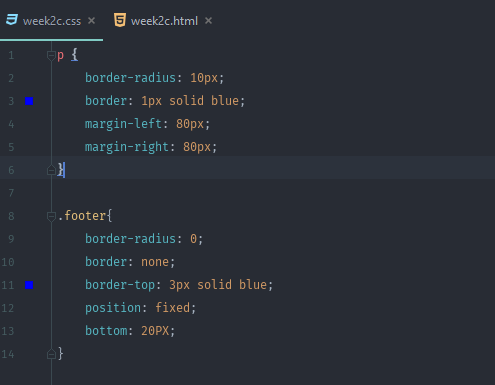
If you want select element with class, use ‘.’;

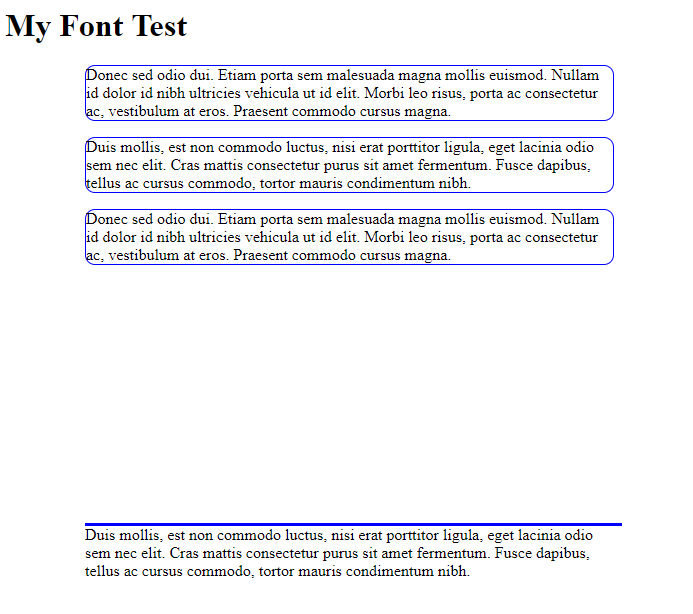
For logic connection, use ‘,’;

D. https://flukeout.github.io/



Week2c.css





# Week 3

## Theory Note

For dynamically change the HTML document and running functions within a website, we need a runtime script language – JavaScript.

Like CSS, JS also has internal and external version, which similar to CSS definition. Both of them have same statement rules:

* Each single instruction ends with a semi-colon “;”
* A group of instructions are put inside { } braces.
* We can save a group of instructions by using the keyword function, followed by (),
* then wrapping the code inside { } braces.
* We can execute a named group of instructions (a function) by writing its name,
* followed by () parentheses.
* We can save information by creating a memory space for it, using the keyword var,
* and then giving the memory space a name (any single word, without spaces).
* We can save information into a memory space using the “=” equals sign

Like other popular languages nowadays. JavaScript features function, function overloading as well as basic process result, log. JavaScript can also trigger by event similar to Java or C#. There are many events can be used to trigger a call-back function

However, JS has some drawbacks which leads to several limitations:

1. For security reason, website cannot calculate values in JS function since the data can be manipulated within browser.
2. Cannot trust the result from local JS functions since the previous reason, which means the important calculation still need POST to servers.
3. In certain circumstances, server need to revaluate and recalculate the data from local JS when clients submit them to server.

JS also has some advanced features:

* Optional parameter

In function “function functionName (P1,P2,P3){…}”, the developers can call this function like:

functionName(P1,P2) or functionName(P1),

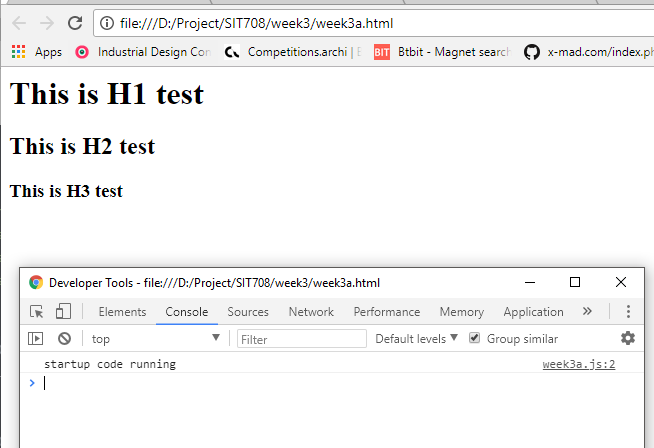
but cannot call this function as: functionName(P2,P3). In this case, P1 is required, and P2, P3 are optional, however, P2 must be submitted as well if P3 is required.

* Namespaces

com.domain.namespace = function() { define functions }, this can accept several functions with same name in different namespaces.

## Lab Exercise Answers









# Week 4

## Theory Note

Since JavaScript is widely used in website and mobile application design, it is not surprisingly that JavaScript has multiple overlaps from other programming languages on the market, which including some concepts like variables and its scope, operators, variable type, objects as well as array.

There 7 types of data type (Variable Types) in JavaScript:

* Null / **undefined**

To be “null”, an object has to be defined first, otherwise it will be “undefined”.

Therefore, in a try-catch situation,

Try{

if (myObj !== null && typeof myObj !== "undefined") }

catch(err){

…

}

Above code will fail, since the “myObj” maybe does not define first and NULL is actually a value.

* Boolean

Typical type for true/false judgement

* Number

Not very strict as other native languages, any “int”,” float” can be considered as “number”, which may lead to problems in practice.

* String

Typical type, array of char

* Object

As an object-oriented programming language, the object can be considered as fundamental base of JavaScript or can be considered as “Class” in Java. One object can be structured with multiple variable types.

* Array

Typical type, a set of fixed amount of same type of values.

* Function

This is very confusing in most situation, however, considering the below situation

var myFunction = function() {

statements;

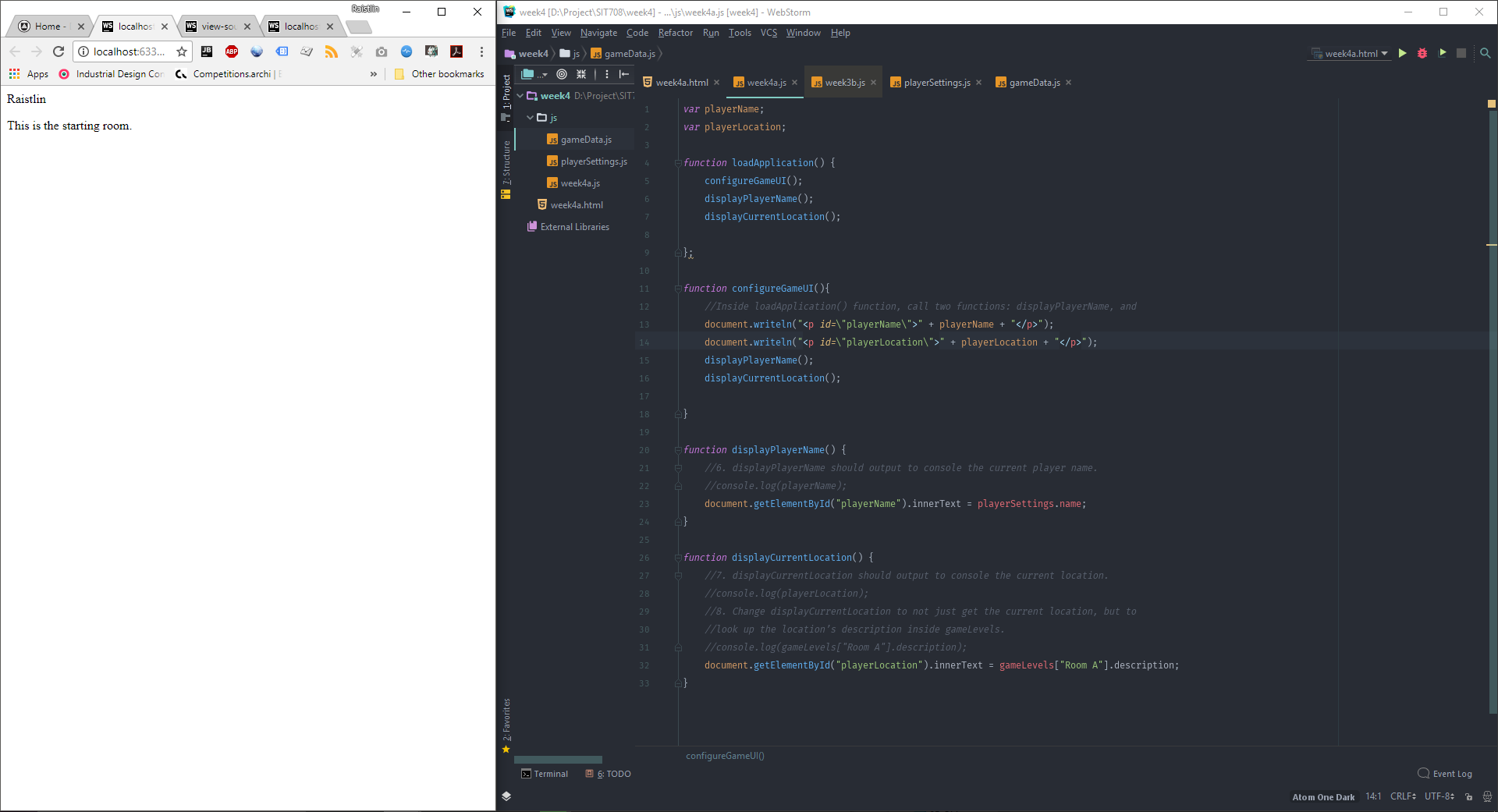
}

The right side newly defined function transfers its “content” to “myFunction”

As the above mentioned, the types are designed to be light-weighted and not suitable for heavy calculation, which lower the performance and increasing the response time of a webpage.

## Lab Exercise Answers

### Excise A



### Excise B

200 Words

The variables in ‘window’ are similar to ‘global variable’ concept in other languages, which means those variables are accessible for any JavaScript function, especially those in external JavaScript files, from the entire webpage (or mobile app JS in this subject) scope.

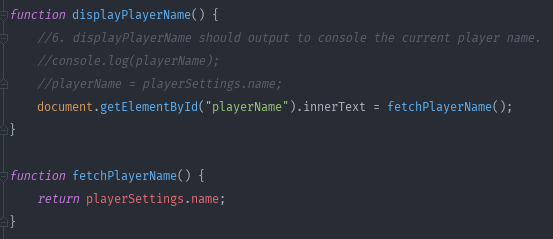
In previous exercise above, there are two reasons for using global scope variables, firstly, these two variables can be access from all functions in ‘week4.js’ file, for example, if we read the value from “playerSetting.js” and put it into the “playerName” variable in function “fetchPlayerName”:

function fetchPlayerName() {

playerName = playerSettings.name; //Example Value: “Example”

}

The value has been changed, and after we use “playerName” again in other functions, the value is “Example”. This is easier approach compares to others like the picture shown below.



Other methods (if we try to avoid using the global variables) require either more skills or programming time.

Another reason is window variables are suitable for quick expansion, especially in small project which does not require too much resource consideration.

“window” level variables are not best practice from my perspective, giving the difficulty level of value manipulation in runtime environment, for example, if there is one new added JavaScript code:

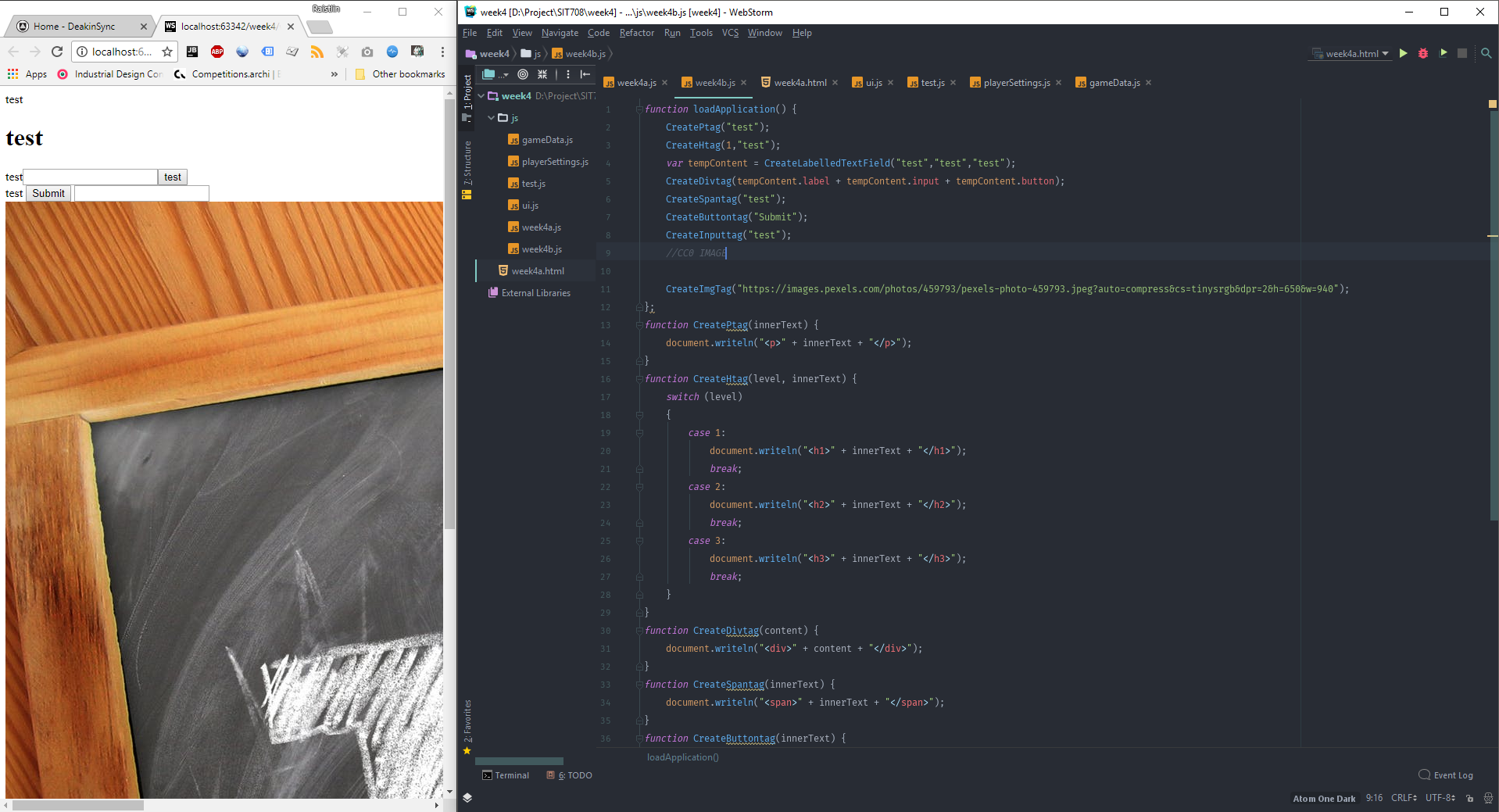
*function* test() {  
 *var* playerName = "TEST";  
 document.getElementById("playerName").innerText = playerName;  
}

Or



The examples are simple in this case, however, if the project is large and requires multiple programmers at the same time, such things will raise a lot of confusions.

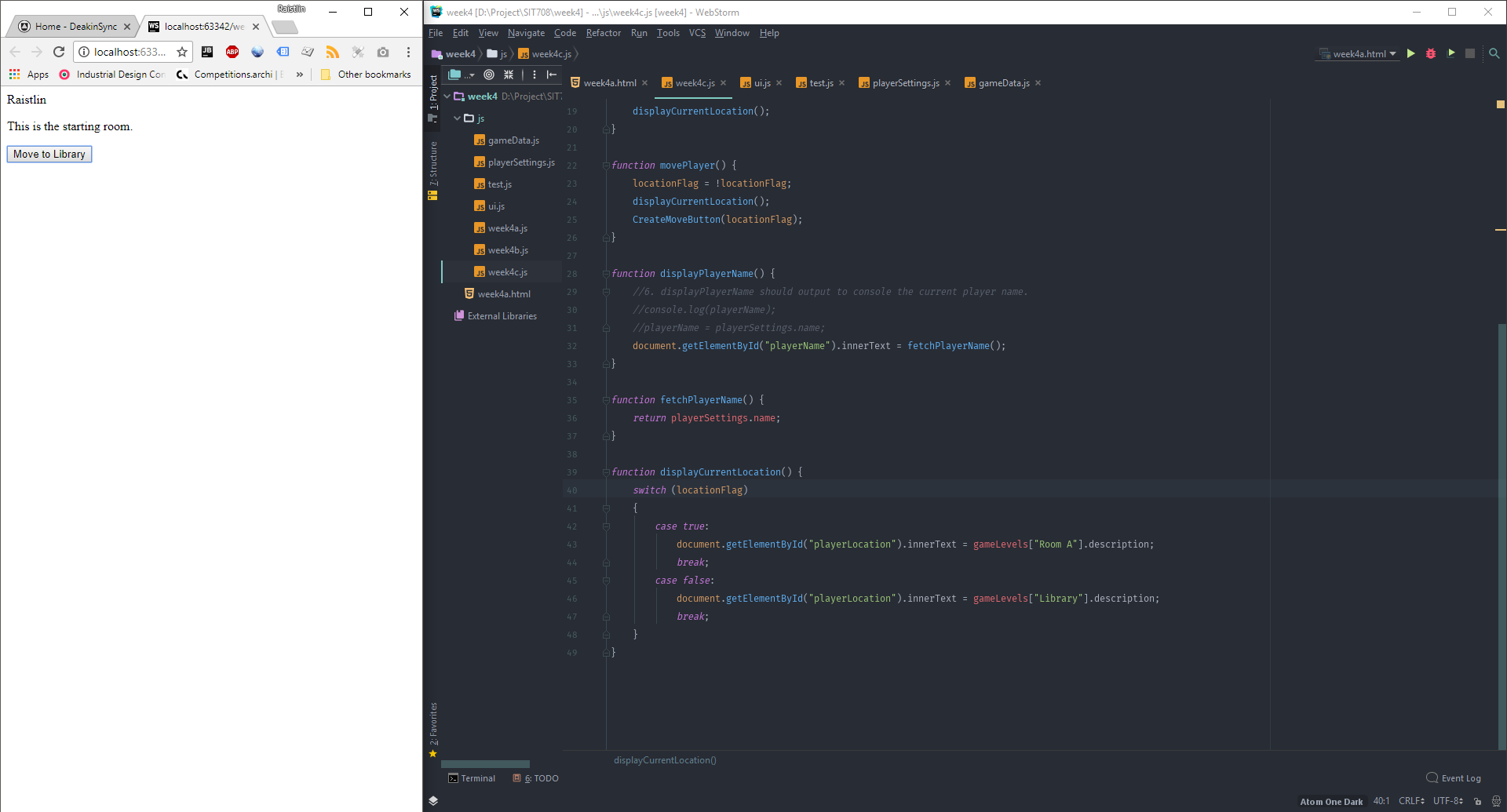
### Excise C



Detail code

*function* loadApplication() {  
 CreatePtag("test");  
 CreateHtag(1,"test");  
 *var* tempContent = CreateLabelledTextField("test","test","test");  
 CreateDivtag(tempContent.label + tempContent.input + tempContent.button);  
 CreateSpantag("test");  
 CreateButtontag("Submit");  
 CreateInputtag("test");  
 *//CC0 IMAGE* CreateImgTag("https://images.pexels.com/photos/459793/pexels-photo-459793.jpeg?auto=compress&cs=tinysrgb&dpr=2&h=650&w=940");  
};  
*function* CreatePtag(innerText) {  
 document.writeln("<p>" + innerText + "</p>");  
}  
*function* CreateHtag(level, innerText) {  
 *switch* (level)  
 {  
 *case* 1:  
 document.writeln("<h1>" + innerText + "</h1>");  
 *break*;  
 *case* 2:  
 document.writeln("<h2>" + innerText + "</h2>");  
 *break*;  
 *case* 3:  
 document.writeln("<h3>" + innerText + "</h3>");  
 *break*;  
 }  
}  
*function* CreateDivtag(content) {  
 document.writeln("<div>" + content + "</div>");  
}  
*function* CreateSpantag(innerText) {  
 document.writeln("<span>" + innerText + "</span>");  
}  
*function* CreateButtontag(innerText) {  
 document.writeln("<button type=\"button\">" + innerText + "</button>");  
}  
*function* CreateInputtag(inputName) {  
 document.writeln("<input type=\"text\" name=\""+inputName+"\">");  
}  
*function* CreateLabelledTextField(lblText, inputName, buttonText) {  
 *return* {  
 label: "<label>" + lblText + "</label>",  
 input: "<input type=\"text\" name=\""+inputName+"\">",  
 button: "<button type=\"button\">" + buttonText + "</button>"  
 }  
}  
*function* CreateImgTag(Source) {  
 document.writeln("<img src=\"" + Source + "\">");  
}

### Excise D



The default button:

document.writeln("<button id=\"move\" onclick='movePlayer()'>Move to Library</button>");

Major functions:

### Week4c.js

*function* displayCurrentLocation() {  
 *switch* (locationFlag)  
 {  
 *case true*:  
 document.getElementById("playerLocation").innerText = gameLevels["Room A"].description;  
 *break*;  
 *case false*:  
 document.getElementById("playerLocation").innerText = gameLevels["Library"].description;  
 *break*;  
 }  
}

*function* movePlayer() {  
 locationFlag = !locationFlag;  
 displayCurrentLocation();  
 CreateMoveButton(locationFlag);  
}

### ui.js

*function* CreateMoveButton(location) {  
 *if*(location)  
 {  
 document.getElementById("move").innerText = "Move to Library";  
 *//document.writeln("<button id=\"move\" onclick='movePlayer()'>Move to Library</button>");* }  
 *else* {  
 document.getElementById("move").innerText = "Move to Room A";  
 *//document.writeln("<button id=\"move\" onclick='movePlayer()'>Move to Room A</button>");* }  
}

## Project Progress

\* Project structure Initialisation

Make the project structure thought thoughtful consideration, 5 hours

Next week: will dig more on next week.

# Week 5

## Theory Note

Control structure is decision-making code which triggered by conditions.

**if-statement triggered when condition matched.**

If(condition)

{

//statement Part

}

In above code, when “condition” result equals true, statement Part will be executed, otherwise the “statement part” will be ignored.

Here is the list of logic execution.

|  |  |
| --- | --- |
| === | The variable type and value are same |
| >,< | Greater than, less then |
| && | And |
| || | Or |
| ! | Not |

**Loop-statement triggered when condition matched as well, but the statement execute multiple times. The loop-statement have for-loop and while-loop.**

The most important points when writing loop-statement are:

1. The loop must have an exit point to prevent endless loop.
2. Cannot declare variable with same name as condition variable.

In following function, even though the “i” is re-defined within for loop, hance, the “i” declare within for loop still share the same memory address with the one defined in for-condition.

for(i=0; i<10; i++)

{

var i = 40;

document.write("Hello World!" + i);

}

**Event handler (Anonymous function declare)**

Basically, it is a function declared inside a function, then attach it to an event. However, the event cannot attach functionB directly.

function functionB(p){

//statement

}

function functionA(p) {

return function() {

functionB(p);

}

}

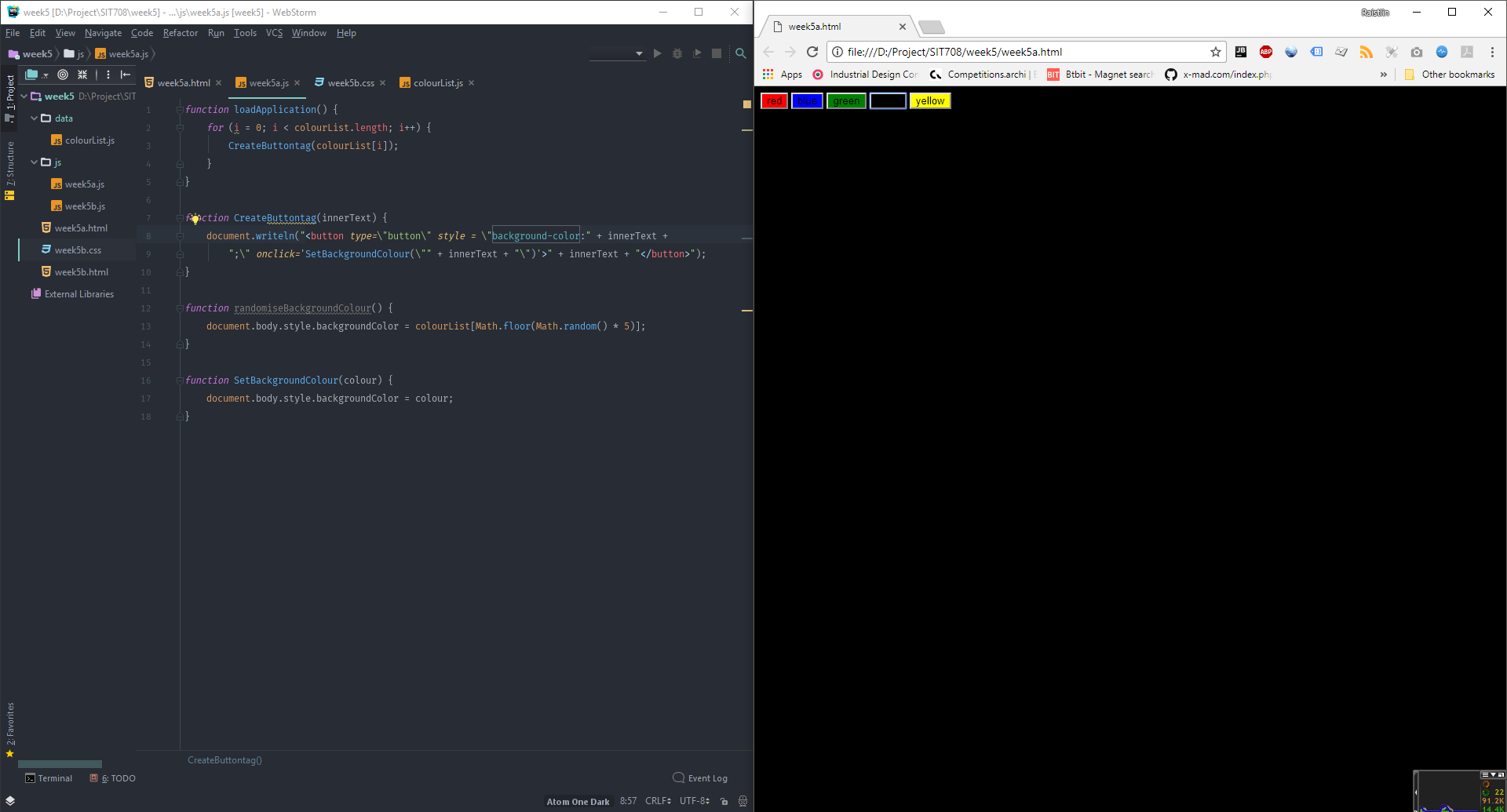
element.event = functionA(“something”);

This equals to:

element.addEventListener("event", function(){ //statement });

## Lab Exercise Answers

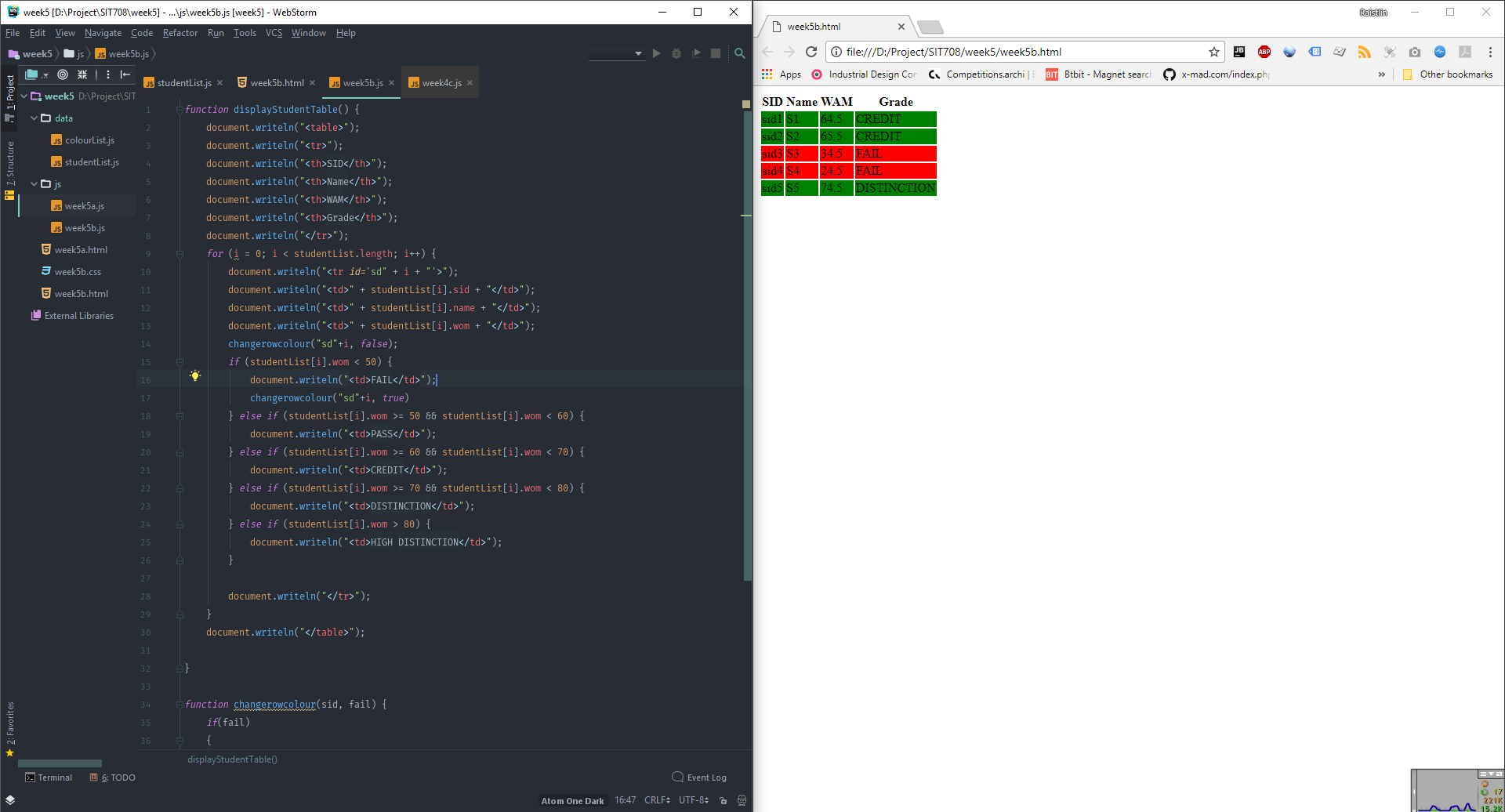
### Excise A



Code:

*function* loadApplication() {  
 *for* (i = 0; i < colourList.length; i++) {  
 CreateButtontag(colourList[i]);  
 }  
}  
  
*function* CreateButtontag(innerText) {  
 document.writeln("<button type=\"button\" style = \"background-color:" + innerText +  
 ";\" onclick='SetBackgroundColour(\"" + innerText + "\")'>" + innerText + "</button>");  
}  
  
*function* randomiseBackgroundColour() {  
 document.body.style.backgroundColor = colourList[Math.floor(Math.random() \* 5)];  
}  
  
*function* SetBackgroundColour(colour) {  
 document.body.style.backgroundColor = colour;  
}

### Excise B



Code:

*function* displayStudentTable() {  
 document.writeln("<table>");  
 document.writeln("<tr>");  
 document.writeln("<th>SID</th>");  
 document.writeln("<th>Name</th>");  
 document.writeln("<th>WAM</th>");  
 document.writeln("<th>Grade</th>");  
 document.writeln("</tr>");  
 *for* (i = 0; i < studentList.length; i++) {  
 document.writeln("<tr id='sd" + i + "'>");  
 document.writeln("<td>" + studentList[i].sid + "</td>");  
 document.writeln("<td>" + studentList[i].name + "</td>");  
 document.writeln("<td>" + studentList[i].wom + "</td>");  
 changerowcolour("sd" + i, *false*);  
 *if* (studentList[i].wom < 50) {  
 document.writeln("<td>FAIL</td>");  
 changerowcolour("sd" + i, *true*)  
 } *else if* (studentList[i].wom >= 50 && studentList[i].wom < 60) {  
 document.writeln("<td>PASS</td>");  
 } *else if* (studentList[i].wom >= 60 && studentList[i].wom < 70) {  
 document.writeln("<td>CREDIT</td>");  
 } *else if* (studentList[i].wom >= 70 && studentList[i].wom < 80) {  
 document.writeln("<td>DISTINCTION</td>");  
 } *else if* (studentList[i].wom > 80) {  
 document.writeln("<td>HIGH DISTINCTION</td>");  
 }  
  
 document.writeln("</tr>");  
 }  
 document.writeln("</table>");  
  
}  
  
*function* changerowcolour(sid, fail) {  
 *if* (fail) {  
 document.getElementById(sid).style.backgroundColor = "red";  
 }  
 *else* {  
 document.getElementById(sid).style.backgroundColor = "green";  
 }  
}

## Project Progress

\* Add music

\* Finished UI library functions

\* Find a suitable font, implement the dynamic css loading

\* Finished main menu, will move to game level UI when select->click play game

UI library cost more times, but after that, every HTML element can be done with one function call. This is the best practice

Next week: Main logic.

# Week 6

## Theory Note

Since JavaScript is not a multi-tasking program, therefore programmers cannot initialise background thread to monitor the process and run executions automatically by itself. However, programmers can define event handles in order to place the execution first and run them when contain event triggered.

Each HTML element has a set of events, for example, element DOM has “onclick”,” onmouseover”,” onmouseout”,”onfocus” and “onload”, which will trigger a function when:

Onclick: clicked the element

Onmouseover: when mouse pointer entering the element

Onmouseout: when mouse point out of the element area

Onfocus: when the element highlighted as focus

Onload: when the element finished loading.

There are two ways to implement event handlers:

1. In html code

<button onclick=”the handler function()”>

1. In JS code

Button name.onclick = the handler function;

In addition, the drag & drop feature, which normally used in online file transfer or submit progress, has 4 steps to implement:

1. Set the attribute draggable to true

imageElement.setAttribute("draggable", "true");

1. Place event handle on “ondragstart”

imageElement.ondragstart = dragStartHandle;

1. Place event handle on “ondragover”

imageElement. ondragover= dragOverHandle;

1. The code behind the drop progress

//This will handle ondragstart event

function dragStartHandle (event) {

currentDraggingObject = event.target;

};

//This will handle on dragover event.

function dragOverHandle (event) {

event.preventDefault();

};

//This will handle ondrop event for image element

imageElement.ondrop = function(event) {

//Check if the currentDragging object is not null

if (currentDraggingElement == null) return;

var receivingElement = event.target;

//Now run the code to do something on the dropped object

console.log("Receiving element: " + receivingElement + "

receives: " + currentDraggingElement);

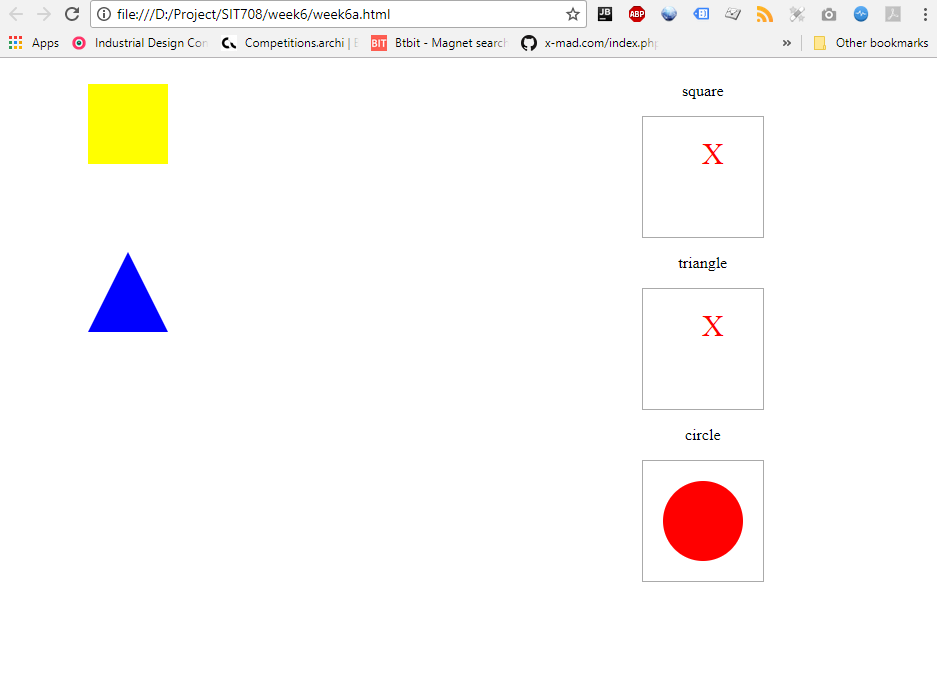
// MUST have this to allow the drop to complete, before submit.

event.preventDefault();

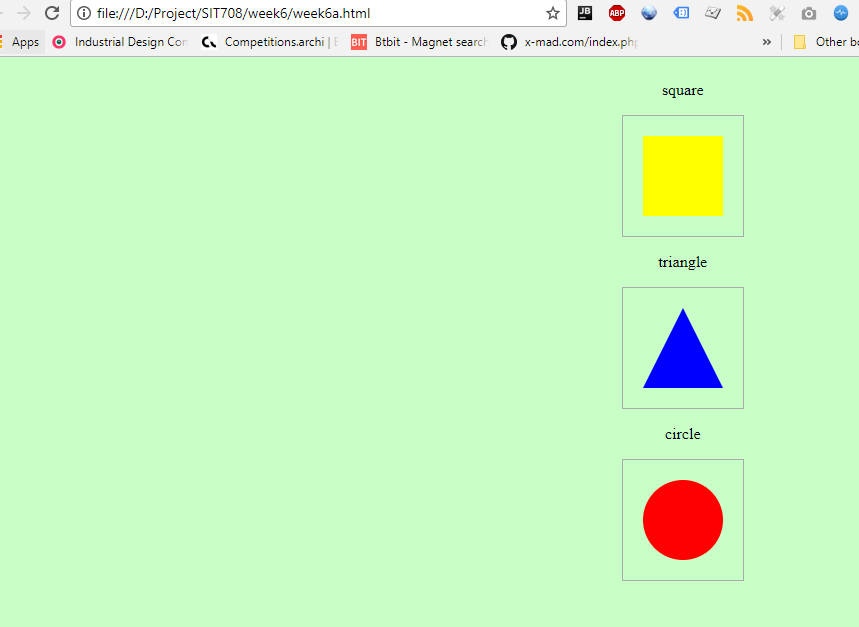
}

## Lab Exercise Answers

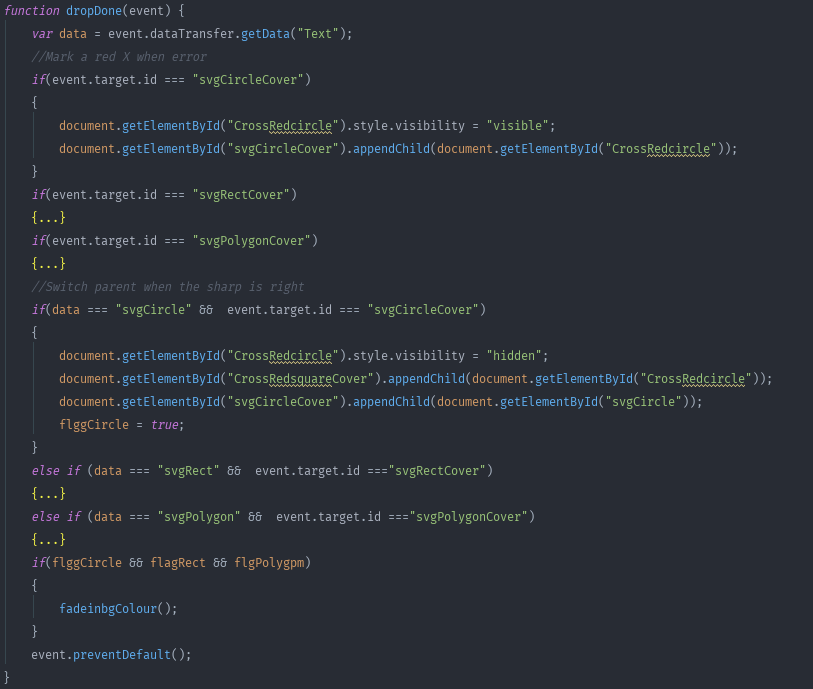
Half won



Won



Screenshot of drop event handler



Code

window.onload = *function* () {  
  
 *//to id the win situation, win condition = all true  
 var* flggCircle = *false*;  
 *var* flagRect = *false*;  
 *var* flgPolygpm = *false*;  
  
 document.getElementById('CrossRedsquare').style.visibility = "hidden";  
 document.getElementById('CrossRedcircle').style.visibility = "hidden";  
 document.getElementById('CrossRedtriangle').style.visibility = "hidden";  
 *//Prepare Drag-able elements* document.getElementById('svgCircle').setAttribute("draggable", "true");  
 document.getElementById('svgRect').setAttribute("draggable", "true");  
 document.getElementById('svgPolygon').setAttribute("draggable", "true");  
 document.getElementById('svgCircle').ondragstart = onDragStart;  
 document.getElementById('svgRect').ondragstart = onDragStart;  
 document.getElementById('svgPolygon').ondragstart = onDragStart;  
 *function* onDragStart(event) {  
 *//Put the id here, then later judge if win or not* event.dataTransfer.setData("Text", event.target.id);  
 }  
 *//Prepare contains* document.getElementById('svgCircleCover').ondragover = Dragover;  
 document.getElementById('svgRectCover').ondragover = Dragover;  
 document.getElementById('svgPolygonCover').ondragover = Dragover;  
 document.getElementById('svgCircleCover').ondrop = dropDone;  
 document.getElementById('svgRectCover').ondrop = dropDone;  
 document.getElementById('svgPolygonCover').ondrop = dropDone;  
 *function* Dragover(event) {  
 event.preventDefault();  
 }  
  
 *function* dropDone(event) {  
 *var* data = event.dataTransfer.getData("Text");  
 *//Mark a red X when error  
 if*(event.target.id === "svgCircleCover")  
 {  
 document.getElementById("CrossRedcircle").style.visibility = "visible";  
 document.getElementById("svgCircleCover").appendChild(document.getElementById("CrossRedcircle"));  
 }  
 *if*(event.target.id === "svgRectCover")  
 {  
 document.getElementById("CrossRedsquare").style.visibility = "visible";  
 document.getElementById("svgRectCover").appendChild(document.getElementById("CrossRedsquare"));  
 }  
 *if*(event.target.id === "svgPolygonCover")  
 {  
 document.getElementById("CrossRedtriangle").style.visibility = "visible";  
 document.getElementById("svgPolygonCover").appendChild(document.getElementById("CrossRedtriangle"));  
 }  
 *//Switch parent when the sharp is right  
 if*(data === "svgCircle" && event.target.id === "svgCircleCover")  
 {  
 document.getElementById("CrossRedcircle").style.visibility = "hidden";  
 document.getElementById("CrossRedsquareCover").appendChild(document.getElementById("CrossRedcircle"));  
 document.getElementById("svgCircleCover").appendChild(document.getElementById("svgCircle"));  
 flggCircle = *true*;  
 }  
 *else if* (data === "svgRect" && event.target.id ==="svgRectCover")  
 {  
 document.getElementById("CrossRedsquare").style.visibility = "hidden";  
 document.getElementById("CrossRedsquareCover").appendChild(document.getElementById("CrossRedsquare"));  
 document.getElementById("svgRectCover").appendChild(document.getElementById("svgRect"));  
 flagRect = *true*;  
 }  
 *else if* (data === "svgPolygon" && event.target.id ==="svgPolygonCover")  
 {  
 document.getElementById("CrossRedtriangle").style.visibility = "hidden";  
 document.getElementById("CrossRedtriangleCover").appendChild(document.getElementById("CrossRedtriangle"));  
 document.getElementById("svgPolygonCover").appendChild(document.getElementById("svgPolygon"));  
 flgPolygpm = *true*;  
 }  
 *if*(flggCircle && flagRect && flgPolygpm)  
 {  
 fadeinbgColour();  
 }  
 event.preventDefault();  
 }  
 *//Fade in without jquery  
 function* fadeinbgColour() {  
 *var* temp = 1;  
 *var* intervalId = setInterval(*function* () {  
 *//Basically change the alpha value of rgb* document.getElementById("bg").style.backgroundColor = "rgba(0,255,0," + temp/100 + ")";  
 *if* (++temp > 100) clearInterval(intervalId);  
 }, 200);  
 }  
};

## Project Progress

\* Auto play music has now against Google chrome's policy (https://developers.google.com/web/updates/2017/09/autoplay-policy-changes), will find a way to implement

\* Make the menu reinitialise new web contents

\* Doing the Image searches try to get low pressure on content making progress

\* Change the structure a bit, I am trying to dynamic generate game (with random), not fixed level thing which looks unwisely

Fail to delivery main logic within this week due to other assignments undergoing.

Next week: main game logic

# WEEK 7

## Project Progress

\* Define colour resource js for colour confusion game

\* Define random colour pick up function

\* Define shape confusion data js

\* Define number order data js

\* Write comments for functions

Finished all resources required for 3 games.

Next week: Game logic

# WEEK 8

## Project Progress

\* Colour confusion game functional

\* Working on score system

\* Make changes on ui\_base.js (UI library to match new requirement)

\* Dynamic creation feature will implement on number order within new few days

\* Changes on ui.css to match new HTML elements

Colour confusion Game logic is finished in this week, the logic can be reused in other two.

Next week: finished the other mini games.

# WEEK 9

## Project Progress

\*Finished all games

\*The logic reusing is work as intended

\*Add audio effects for each element.

The previous library and general logic implement are really helpful for delivery on time

Next week: Polish some user interface, video

# References

Google, 2018. *Material Design.* [Online]   
Available at: https://material.io/

Hrissan, 2005. *Symbian OS design faults.* [Online]   
Available at: https://www.codeproject.com/Articles/9357/Symbian-OS-design-faults  
[Accessed 10 3 2018].

Tom May, L. C., 2018. *Creative Bloq.* [Online]   
Available at: https://www.creativebloq.com/graphic-design/what-flat-design-3132112  
[Accessed 10 3 2018].