CSIT128:

**Introduction to Web Technology**

**Assignment 2**

|  |  |  |
| --- | --- | --- |
| **No.** | **Name** | **SIM Student No.** |
| UOW ID 1 | Darayuth Chea | 8550864 |
| UOW ID 2 | Ho Ka Yan Jeslyn | 8535383 |

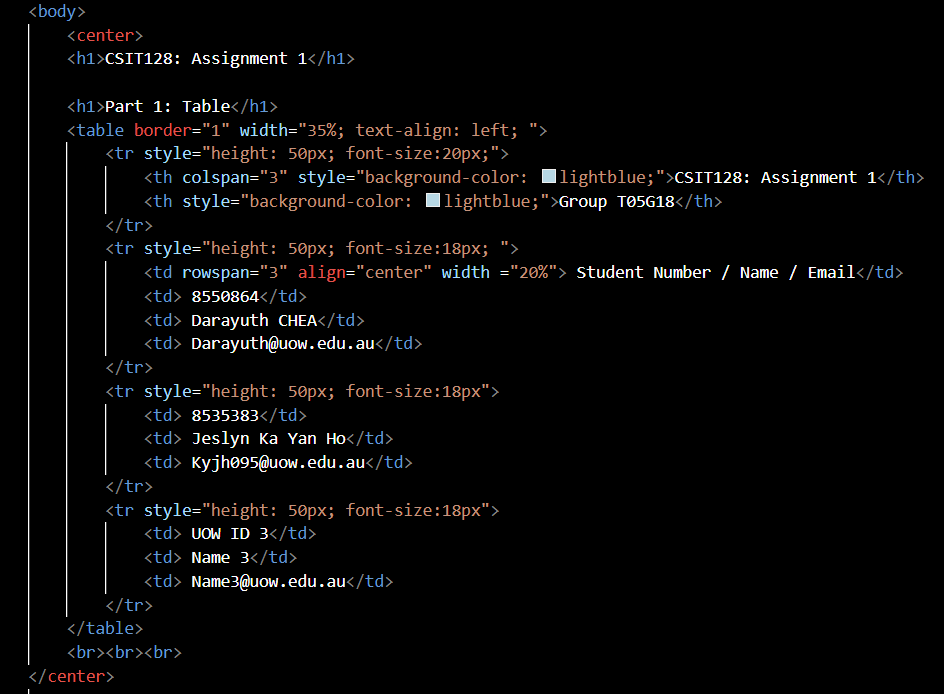
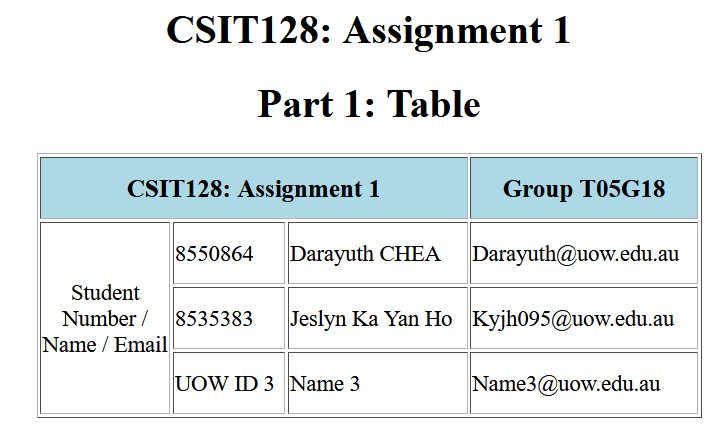
**Date of Submission: 6 November 2023Table of Content**

Part 1: Introduction 3

Part 2: Game 4

Part 3 9

# Part 1: Introduction



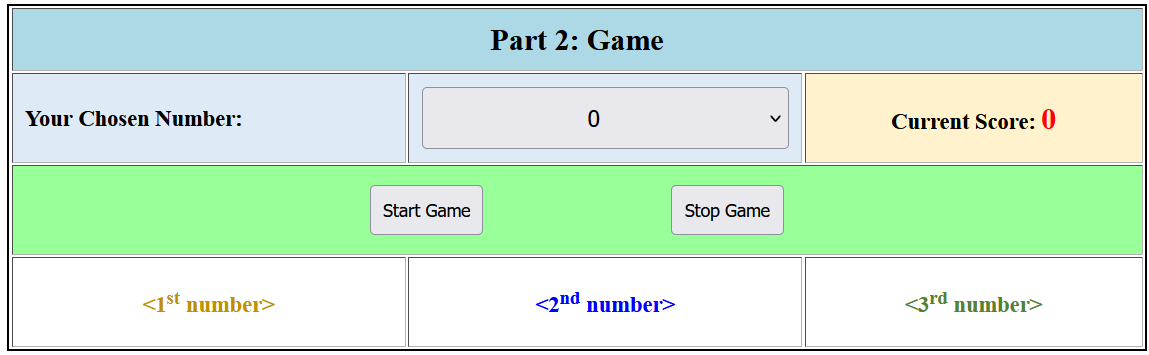
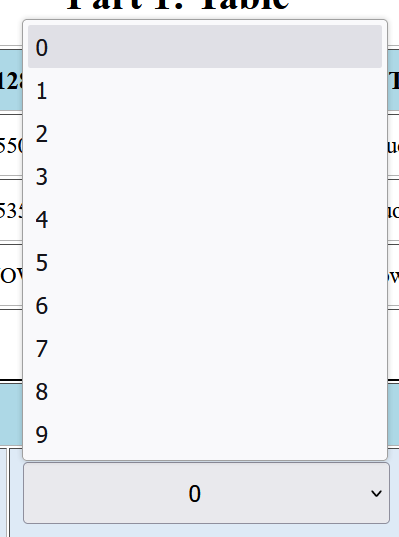
**Explanation:**

This section includes a table that will show our student ID, name, and email address. We decide to use the border properties of 1px, the element's width value of 35%, and text-align: left to make the table easier to read.

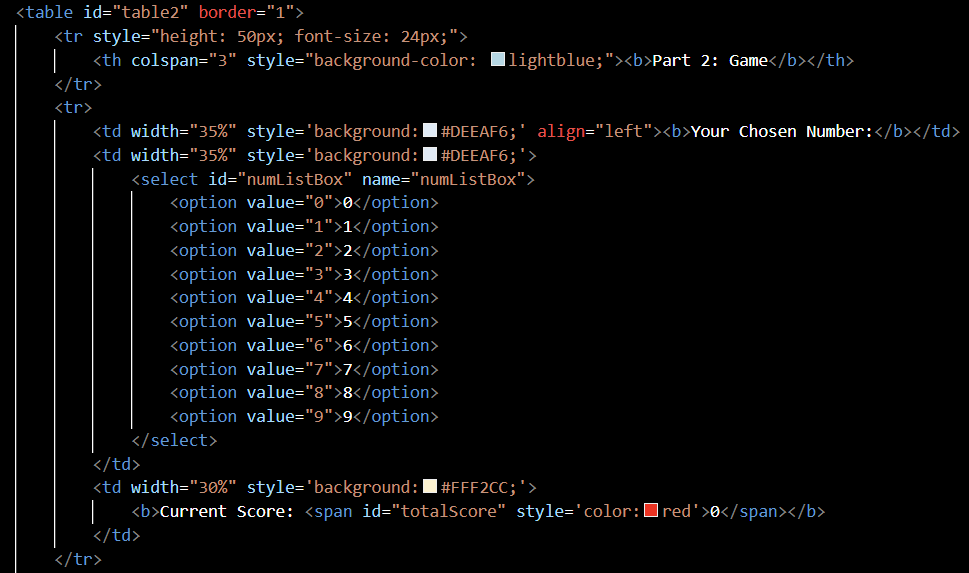
There is a header row in this table, and we utilize the element 'th'. To enable the header of CSIT128: Assignment 1 to have three columns underneath it, we use the attribute 'colspan' with a value of 3. We don't need to include any more characteristics beneath the header of Group T05G18 because there is just one row there.

We can enter our information in three rows, thus the Student Number, Name, and Email header uses a 'row span' element with a value of three.

# Part 2: Game

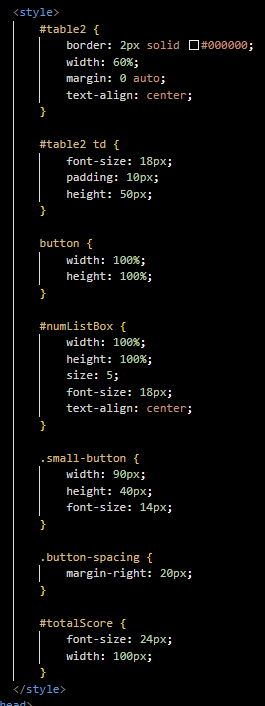


**HTML**



**Explanation CSS <style>**

**<select id="nucleator" name="numListBox">**



This line create a dropdown list where user can select a number, with the ID call”numListBox

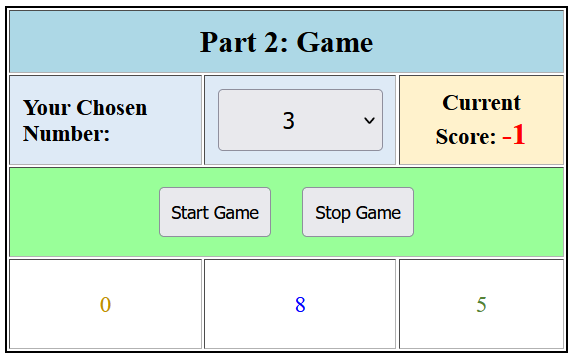
**<button id="startButton" class="small-button button-spacing"onclick="start()">Start Game</button>**

This line creates a start button. Pressing it will cause the script's "start()" function to be triggered. The same is true for the stop button.

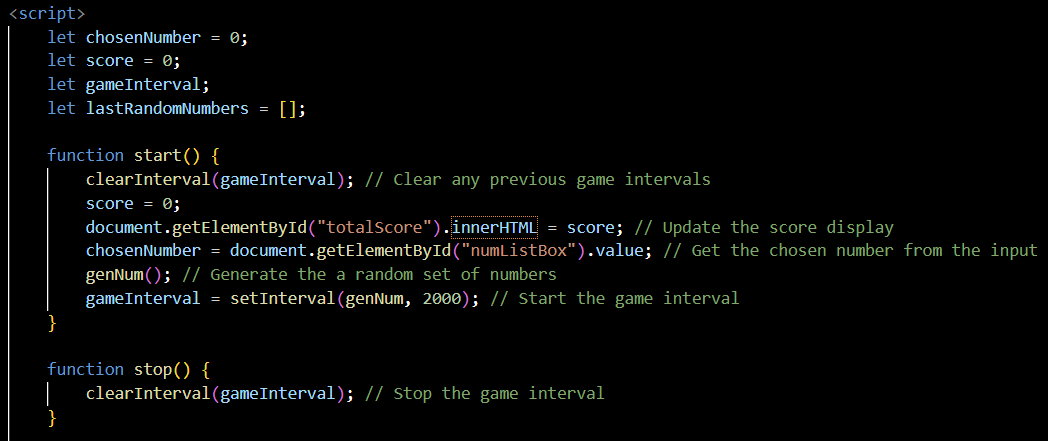
**<span id="intButton1" onclick="checkNumber(0)" style='color:#BF9000;'><b><1<sup>st</sup> number></b></span>**

This line contains a span element, which when clicked, will cause the checkNumber (0) function to be called. This also applies to the remaining span elements for the int Buttons.

**JavaScript <script>**



**for the start() and stop()**



**Explanation**

This is start function and a stop function, it will be triggered when a user press the start button or stop button.

**clearInterval(gameInterval)**

This line stops several game loops from running at the same time by clearing any previously defined game interval.

**document.getElementById("totalScore").innerHTML = score;**

The HTML element with the ID "totalScore" is updated to show the starting score of 0.

**chosenNumber = document.getElementById("numListBox").value;**

This line of code here will retrieve the selection entered by the user from the drop down selection list that have an ID called “numListBox” and place it inside the chosenNumber variable.

**genNum()**

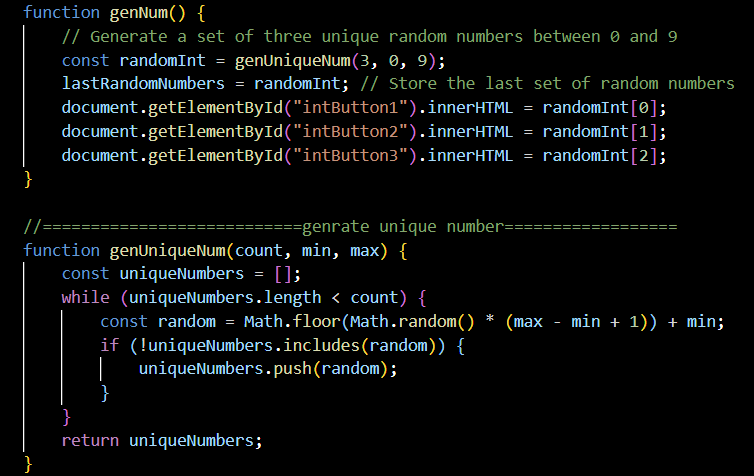
This function has been called to generate a random set of numbers.

**gameInterval = setInterval(genNum, 2000)**

The function “genNum() will be called every 2000milliseconds, which is 2 seconds.

**JavaScript <script>**

**for the genNum() and genUniqueNum()**



**Explanation**

As u can see from the code above there is a function that help generate a random number and another function that make sure that a set of number generated are all unique.

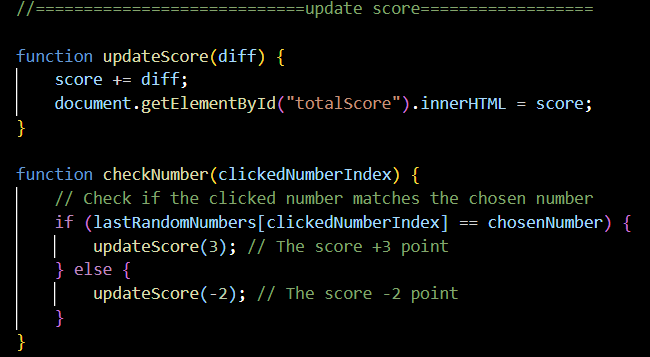
In the genNum() function, the code “const randomInt = genUniqueNum(3, 0, 9);” make sure that it will generate 3 unique number between 0 to 9. It will then stores this set of numbers inside an array called “randomInt”

In the genUniqueNum () function, its using a while loop to generate random integer using the ‘Math.random()’ that’s between ‘min’ and ‘max which is 0 and 9 as shown in the genNum() function.

We use the ‘include’ method to check if the number is already included inside the set.

**JavaScript <script>**

**for the updateScore() and checkNumber()**

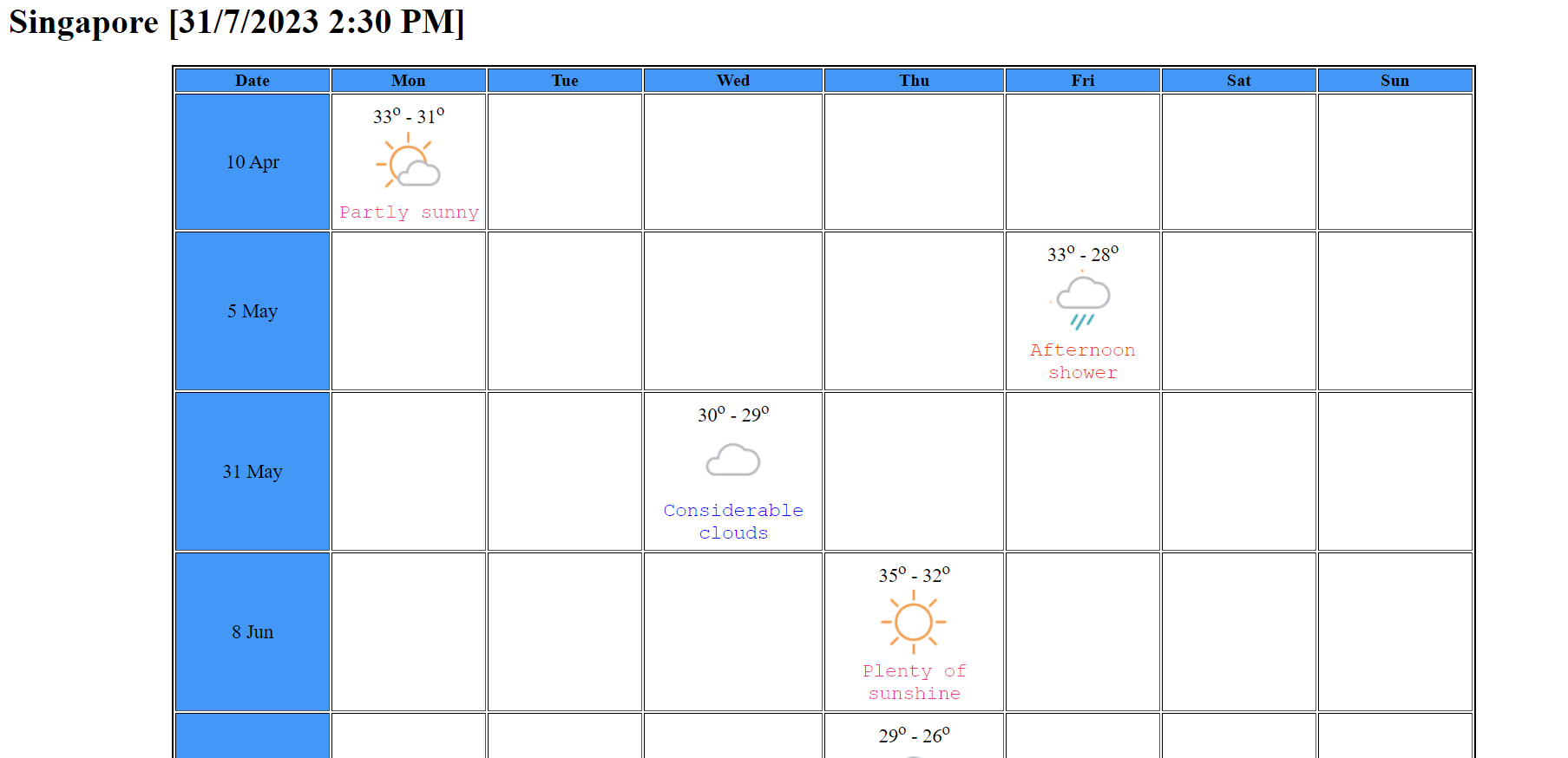


**Explanation**

The updateScore() function have a numeric value “diff” to help updates the user’s score.

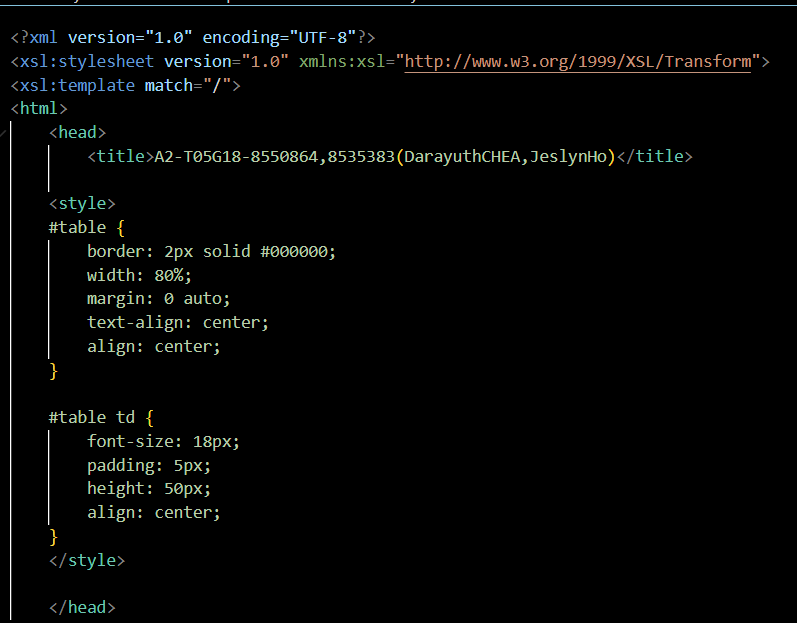
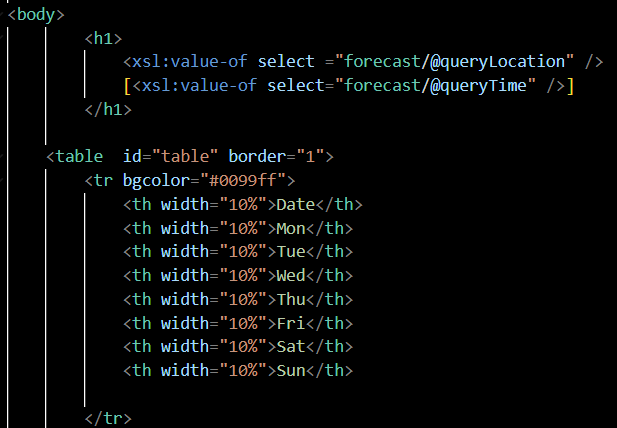
The checkNumber() function checks if the clicked number match the selected number in the array, it will compare the number with the ‘chosenNumber’ If and else method is use, If match the score will increase by 3, if not the score will decrease by 3.

# Part 3

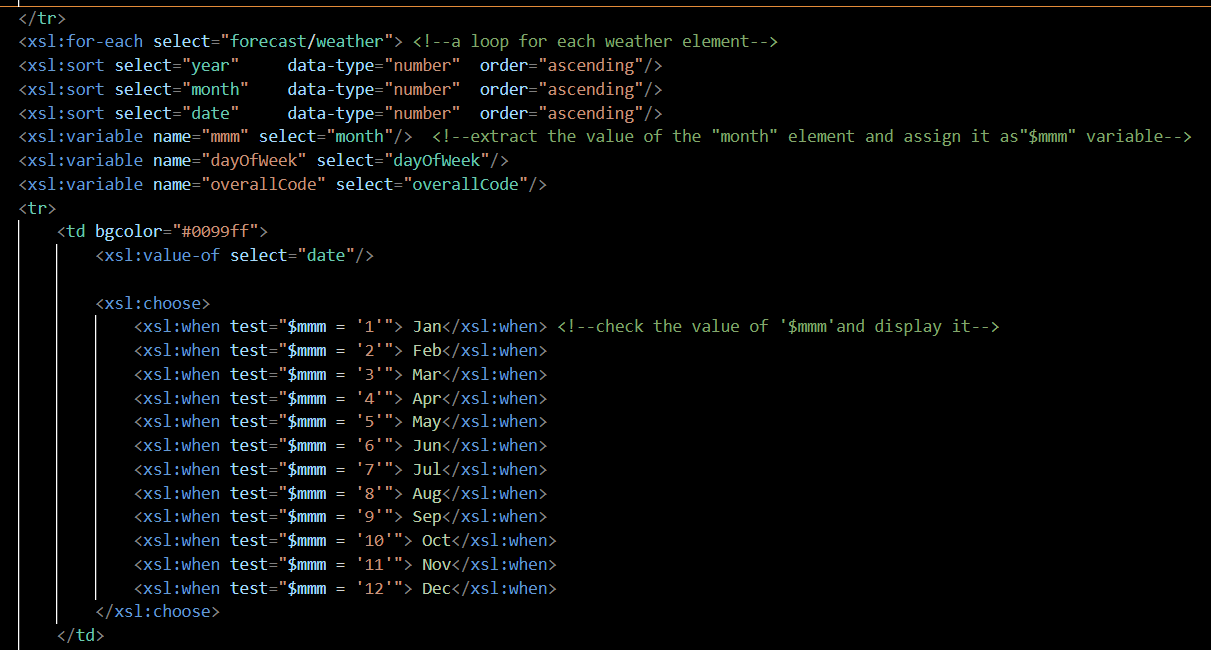


**S2.xsl**

**CSS <style>**



**S2.xsl**



**Exaplaination  
<xsl:for-each select="forecast/weather">**

This starts a loop that goes over every "weather" element that is part of the "forecast" element.

**<xsl:sort>:**  
These elements sort the ‘weather’ element by year, month and date in an ascending order.

**<xsl:variable name="mmm" select="month"/>**

extract the value of the "month" element and assign it as"$mmm" variable, it’s the same goes for dayOfWeek and overAllCode.

**<xsl: choose>**

This conditional XSLT component is used to choose a certain value depending on specified requirements.

**<xsl:when>**

Its an element assists in determining the value of the "$mmm" variable (month) and displaying the appropriate three-letter month abbreviation. (E.g. Jan, Feb)

**S2.xsl “Mon”**



**Explanation**

**<xsl:when test="$dayOfWeek = 'Mon'">:**

This <xsl:when> element of the <xsl:choose> construct determines whether the value of the "$dayOfWeek" variable, which indicates the day of the week, equals 'Mon' (Monday).

In the event that the condition is met, Monday is represented by the current weather data.

**The nested <xsl:choose>:**

Another <xsl:choose> construct within the Monday-specific condition is used to make a decision based on the "overallCode" variable (which is the overall weather condition).

**Test="$overallCode = 'cloudy'"> <xsl:when**

This <xsl:when> element checks to see if the "overallCode" equals "cloudy" inside the nested <xsl:choose>. If the condition have been met, Monday's weather will be overcast overall.

**<textsl:value-of select="overall"/></font><font face="courier" color="blue">**

This code displays the "overall" weather condition text from the XML data on a cloudy day, sets the font to "Courier," and changes the text colour to blue.

The remaining days of a week, which are listed below, will follow the same pattern.

**S2.xsl “Tue”**



**S2.xsl “Wed”**



**S2.xsl “Thu”**



**S2.xsl “Fri”**



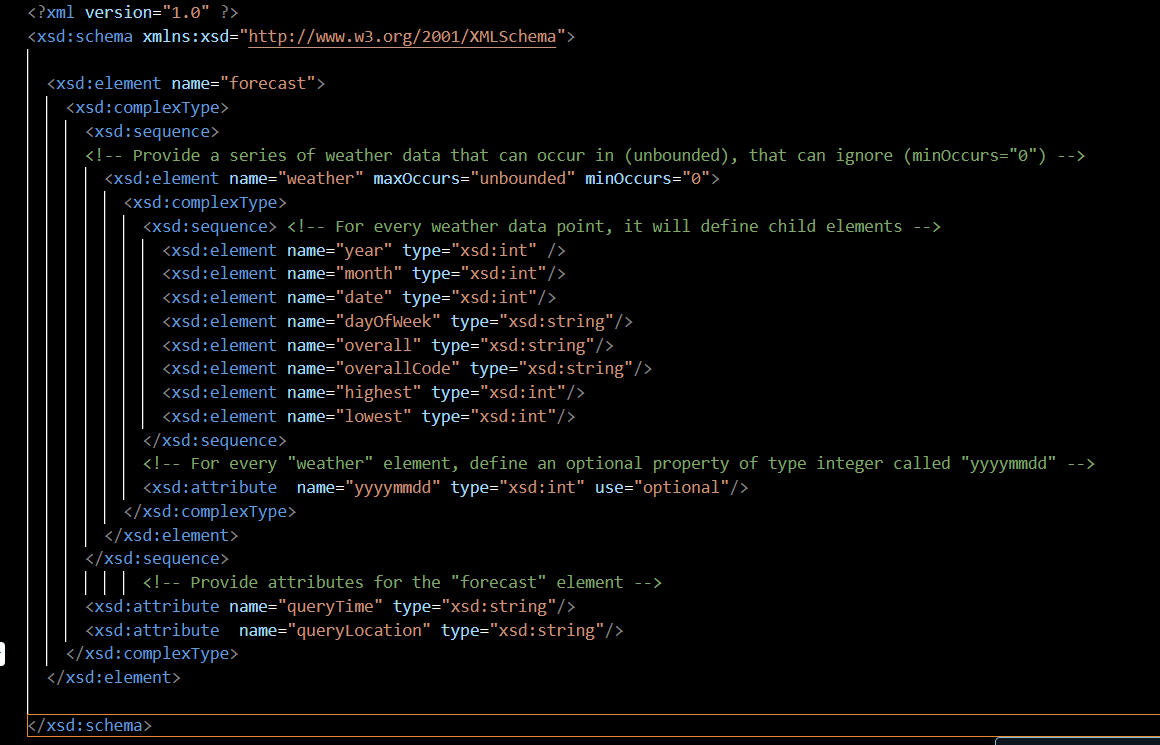
**S2.xsl “Sat”**



**S2.xsl “Fri”**



**S2.xsd**



**Explanation**

**<xsd:element name="forecast">**

This code defines the XML document's root element, "forecast." It outlines the forecast data's general structure.

**<xsd:element name="weather" maxOccurs="unbounded" minOccurs="0">:**

Describes the "weather" element, which is optional ("minOccurs="0") and can occur more than once ("unbounded"). It permits the forecast to contain zero or more weather entries.

**The child Elements (Embedded in the Term "weather"):**

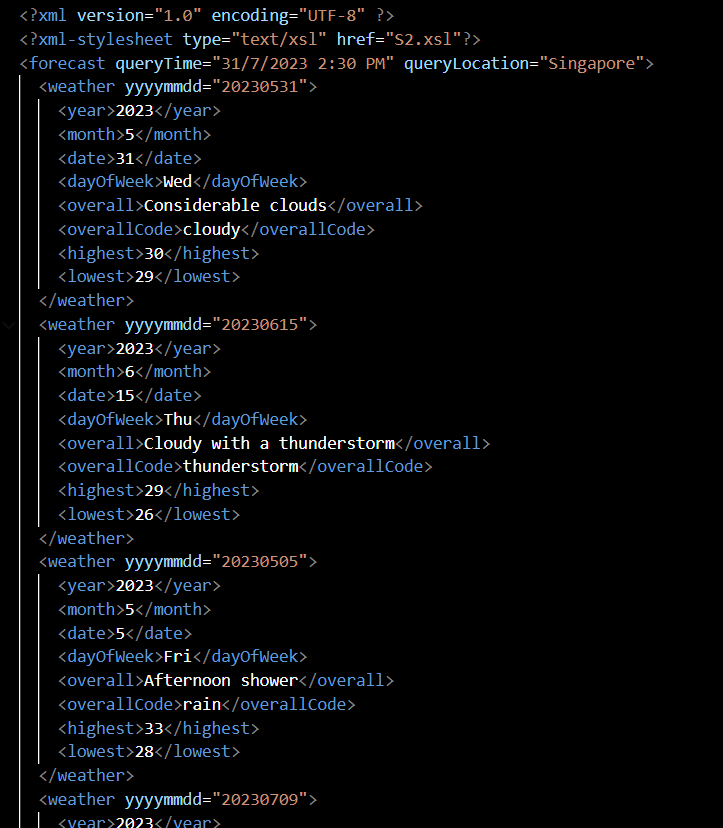
The weather data is defined by these child elements: "year," "month," "date," "dayOfWeek," "overall," "overallCode," "highest," and "lowest." Every child element possesses a specific data type.

**Attributes for "forecast" Element:**

Gives the "yyyymmdd" attribute for every element in the "weather" category. It is of integer (xsd:int) type and optional.

Properties of the "forecast" Element:

**A2.xml**



**<?xml-stylesheet type="text/xsl" href="S2.xsl"?>**

Only this line of code have been change to locate the file call “S2.xsl”