

1809131 Anton Hall - ACCA5022

by Anton Hall

FILE	12316_ANTON_HALL_1809131_ANTON_HALL_- _ACCA5022_219797_370248525.DOCX (1.69M)		
TIME SUBMITTED	14-JAN-2020 12:04PM (UTC+0000)	WORD COUNT	2806
SUBMISSION ID	118279832	CHARACTER COUNT	14426

University of Wales, Trinity Saint David

Web Technologies (ACCA5022) (Javascript)

Anton Hall

Contents

Introduction	1
1. Component 1	1
1.1 Current time	1
1.2 Code	1
1.3 Method	2
2. Component 2	3
2.1 Login	3
2.1 Code	3
2.3 Method	4
3. Component 3	4
3.1 All Pipe Routes	4
3.2 Code	5
3.3 Method	7
4. Component 4/5	7
4.1 Questionnaire/Timer	7
4.2.1 Questionnaire Code	9
4.2.2 Questionnaire Method	11
4.3.1 Timer Code	12
4.3.2 Timer Method	12
Conclusion	12
References	13

Introduction

The following report evidences a website that has been created with JavaScript to allow user interaction. The chosen subject was one that is relevant to the developers work within the water engineering environment. The website makes use of five discrete JavaScript components each containing several functions across three HTML pages.

1. Component 1

1.1 Current time

The first component visible to the user is the current time when they open the index page of the website. It is displayed on the right-hand side of the navigation bar and initiated when the webpage is loaded. Once loaded the function is set to be executed at an interval of 1 second so that the time is always correct. The function makes use of an array to change the month number into the month name as this format better suits the website. The current time function is used throughout the website.

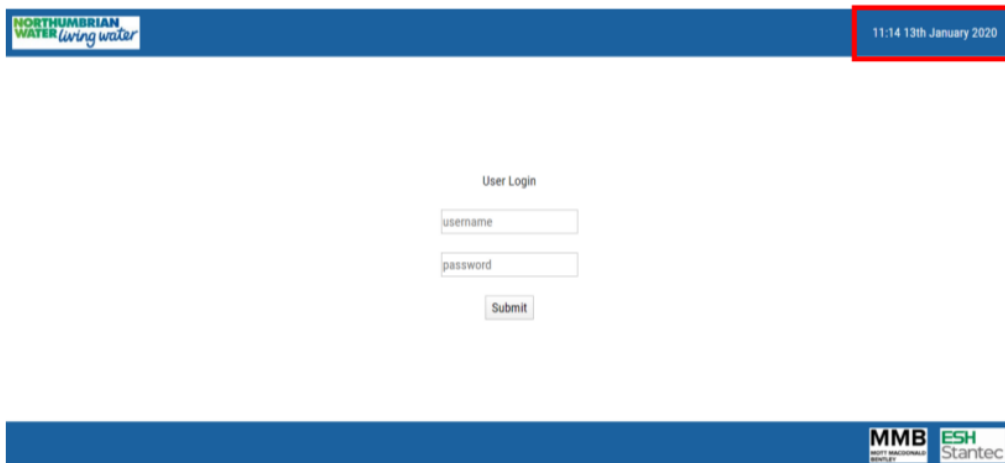


Figure 1 Current time

1.2 Code

```
function currenttime() { //function to show current time called onload of the
website
    showdate = document.getElementById("time"); //accesses the time div in the
html
    thedate = new Date(); //create variable for current date and time
    year = thedate.getFullYear(); //get the year
    day = thedate.getDate(); //get the day
    month = thedate.getMonth(); //get the month
    hour = thedate.getHours(); //get the hours
    minute = thedate.getMinutes(); //get the minutes
    monthnames = new Array(); //create an array to turn the month numbers into
month names
    monthnames = ["January", "February", "March", "April", "May", "June", "Jul
y", "August", "September", "October", "November", "December"];
```

```

    if (day == 1 || day == 21 || day == 31) { //if statement that checks the day and applies the relevant prefix
        day = day + "st ";
    }
    else if (day == 2 || day == 22) {
        day = day + "nd ";
    }
    else if (day == 3 || day == 23) {
        day = day + "rd ";
    }
    else {
        day = day + "th ";
    }
    if(minute < 10){ //if statement to put a 0 in front of the minutes if they are less than 10
        minute = "0" + minute
    }
    if(hour <10){ //if statement to put a 0 in front of the hours if they are less than 10
        hour = "0" + hour
    }
    showdate.innerHTML = hour + ":" + minute + " " + day + " " + monthnames[month] + " " + year; //builds up the date in the format I want it and inserts it into the HTML
}
function timer() { //timer function
    timer = setInterval(currenttime, 1000); // the function initiates the current time function every second intervals
}

```

1.3 Method

- The first stage of creating this component was to access the div element where the time would be shown on the website, this is done by using getElementById and the relevant ID from the HTML code.
- Then the date variable needed to be built, firstly setting the date variable to the current date with new Date, then pulling out the relevant parts needed to make up the date in the desired format.
- Once this was achieved an array was developed to replace the month numbers for the month words.
- Next was to create if statements to apply certain suffixes to the day so that it was fitting with the rest of the date format and to add zeros in front of numbers below ten.
- Once the date was created in the correct format it then needed to be fed back into the div element accessed at the start of the function. This is achieved by using innerHTML followed by the relevant variables.
- Finally, another function was created to run the previous function at a set interval of a second so that the date would refresh the same as a normal clock.

2. Component 2

2.1 Login

The next function that the user will initiate is the login function, it will authenticate the user's username and password and load the next page of the website when the user hits submit, and the details are correct. If the user's credentials are incorrect and alert will pop up and tell them. They can't move further into the website without the correct credentials. The username is username and password is password. This function was purely created as a proof of concept and it is by no means a way of securing a website as the credentials can be easily accessed through the source code.

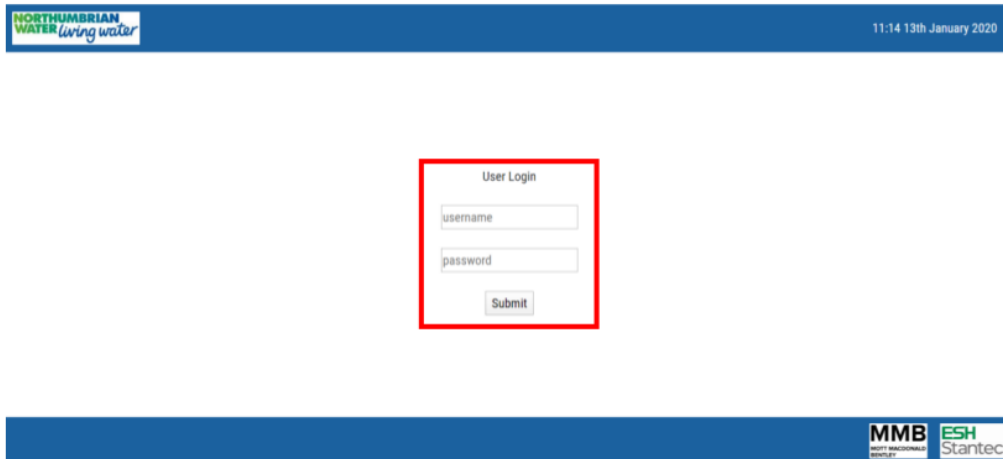


Figure 2: Login

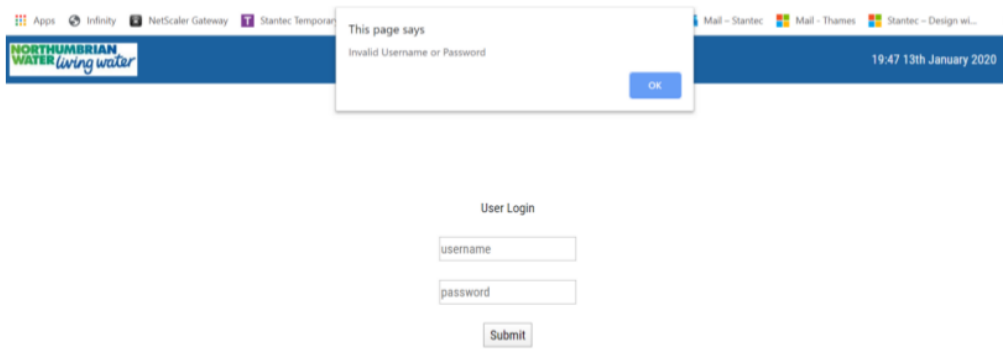


Figure 3: Failed login

2.1 Code

```
function login() //login function initiated by the submit button
{
    user = document.getElementById("user").value //stores the user text in the
    username box to the user variable
    pass = document.getElementById("pass").value //stores the user text in the
    password box to the pass variable
    if (user=="username" && pass=="password") { //checks whether the username
    and password entered match those in the statement
```

```

location="../HTML/piperoutes.html" //if true loads the website
} else {
alert("Invalid Username or Password") //if false alerts the user
}
}

```

2.3 Method

- The first stage of this function is to gather the information provided by the user in both text boxes and store them as variables. This is achieved by using getElementById linked to the correct ID in the HTML .value.
- Next an if statement is used to compare what the user has inputted to the required credentials. If they are correct the java script loads the page in the location specified. If it is incorrect then it will display a pop up telling the user their credentials are incorrect.

3. Component 3

3.1 All Pipe Routes

The All pipe routes function was designed as a proof of concept so that a water company could be potentially more engaged with their customers by allowing some interaction with planned engineering works. It allows the user to toggle pipeline routes on and off a map, letting them see any number of the planned routes and the ability to clear all routes. The map is fully interactive, they can zoom in to see exactly where the route will be located, allowing them to fully understand the extent of the planned engineering routes. This JavaScript was developed using the google maps JavaScript API. [1]

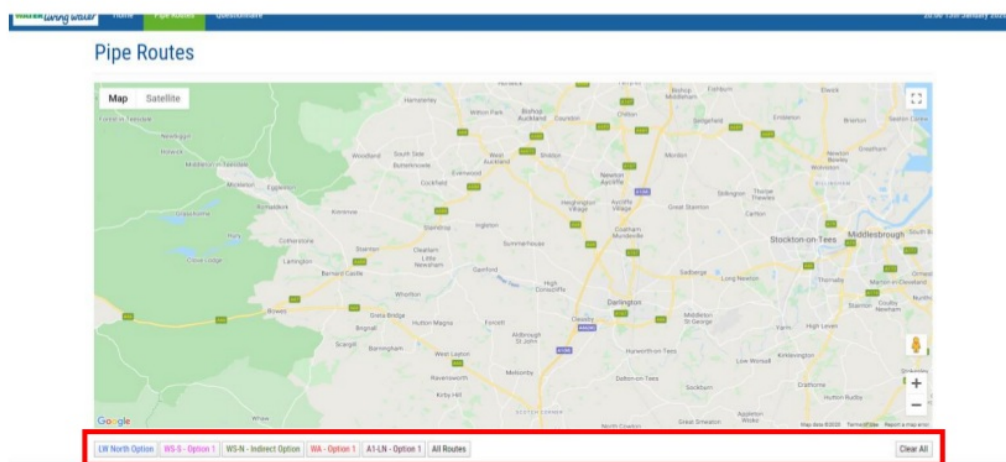


Figure 4: Pipe routes

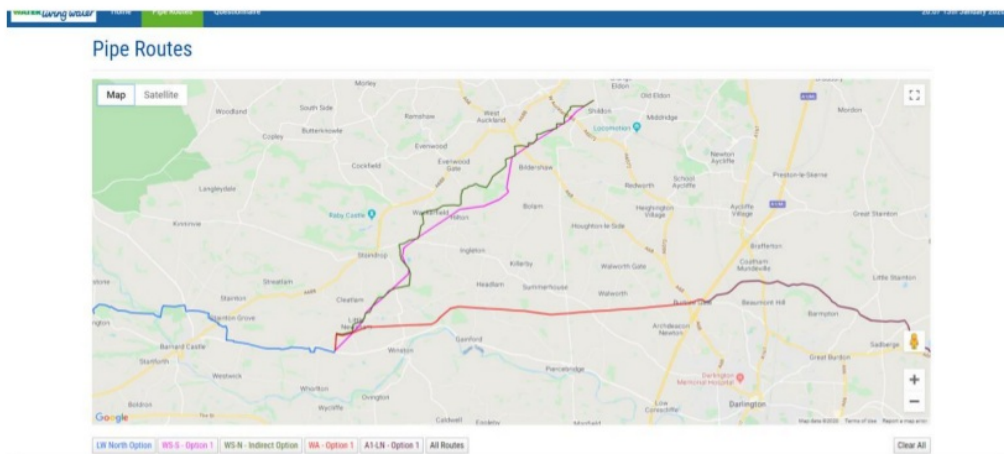


Figure 5: All pipe routes selected

3.2Code

```

4. var map //global variable
5. function initMap() { //function to initialise the map on the page
6.     map = new google.maps.Map(document.getElementById('map'), { //variable to create the map inside the map div
7.         center: {lat: 54.558459, lng: -
8.             1.7035124}, //options for the map, the centre point and zoom level
9.         zoom: 11
10.    });
11.
12. function toggleoverlay(a) { //function to toggle the pipe route overlay
13.     taking the variable a from the HTML
14.     if (a==1 || a==6) //If you user presses the first button then
15.         a=1 if the user presses all routes then a=6
16.     {var pipeRoute1 = [ //create the pipe route using a lat and long for each vertex
17.         {lat: 54.557526, lng: -1.9810411},...
18.         ...{lat: 54.543778, lng: -1.8254228}];
19. var pipeLine = new google.maps.Polyline({ //creates a new polyline with options
20.     path: pipeRoute1, //tells the polyline which route to plot
21.     strokeColor: '#0066ff', //sets the colour
22.     strokeOpacity: 1.0, //sets the opacity of the colour
23.     strokeWeight: 2, //sets the thickness of the line
24. });
25. pipeLine.setMap(map); //draws the pipeline on the previously initialised map if the statement is true
26. }
27.

```

```

28. if (a==2 || a==6) {
29.     var pipeRoute2 = [
30.         {lat: 54.543778, lng: -1.8254538},...
31.
32.         ...{lat: 54.636994, lng: -1.6588959},,];
33.
34.     var pipeline = new google.maps.Polyline({
35.         path: pipeRoute2,
36.         strokeColor: '#FF00ff',
37.         strokeOpacity: 1.0,
38.         strokeWeight: 2,
39.     });
40.     pipeline.setMap(map);
41. }
42.
43. if (a==3 || a==6) {
44.     var pipeRoute3 = [
45.         {lat: 54.543778, lng: -1.8254538},...
46.
47.
48.         ...{lat: 54.636994, lng: -1.6588959},,];
49.
50.     var pipeline = new google.maps.Polyline({
51.         path: pipeRoute3,
52.         strokeColor: '#336600',
53.         strokeOpacity: 1.0,
54.         strokeWeight: 2,
55.     });
56.     pipeline.setMap(map);
57. }
58.
59. if (a==4 || a==6) {
60.     var pipeRoute4 = [
61.         {lat: 54.543778, lng: -1.8254538},...
62.
63.         ...{lat: 54.562146, lng: -1.5875338},,];
64.
65.     var pipeline = new google.maps.Polyline({
66.         path: pipeRoute4,
67.         strokeColor: '#FF0000',
68.         strokeOpacity: 1.0,
69.         strokeWeight: 2,
70.     });
71.     pipeline.setMap(map);
72. }
73.
74.     if (a==5 || a==6) {
75.         var pipeRoute5 = [

```

```

76.         {lat: 54.562109, lng: -1.5874413},...
77.
78.         ...{lat: 54.543247, lng: -1.4419618},,];
79.
80.     var pipeline = new google.maps.Polyline({
81.         path: pipeRoute5,
82.         strokeColor: '#660033',
83.         strokeOpacity: 1.0,
84.         strokeWeight: 2,
85.     });
86.     pipeline.setMap(map);
87. }
88. }
89.

```

3.3 Method

- The first thing need in this component is a global variable for the map. This needs to be a global variable as it is accessed by more than one function.
- Next, we need to create the function that will create the map on the page of the website. To do this map is made equal to a new google map which will access the map div in the HTML using getElementById. There are also a few options that are applied to the map at this point, including a centre point and a zoom scale.
- Once the map had been created the pipelines could be created to be toggled on top of it. This was done by using buttons in the HTML that pass parameters to the JavaScript. An if statement then verifies what variable has been passed from the code and draws the relevant line on the map that corresponds with the button pressed.
- Firstly the function uses google.maps.Polyline to draw the polyline on the map using setMap with the map variable. There are also options involved when creating the polyline, the most important being the actual pipe route which is an array variable containing the latitude and longitude of each vertex on the pipeline. It also sets colour and line weight.
- The last function needed was to be able to clear the map, this is done by re-running the first function to initialise a blank map

4. Component 4/5

4.1 Questionnaire/Timer

The next part of the website is a questionnaire based on water, keeping in theme with the main pipe routes page. Potentially used again as customer interaction but something for the younger members of the customers families. It's an eight-question multiple choice quiz that uses radio buttons for the user to enter their choice for each question. Once the user submits their answers feedback is given to the user so they can see which answers were right or wrong as well as their score. A 30 second timer has also been built into the quiz which starts counting down as soon as the user clicks the start quiz button. The quiz isn't too difficult, so a timer adds an element of pressure to the quiz, as it doesn't allow for any hesitation. As the quiz is timed the questions are hidden until the user clicks starts quiz.

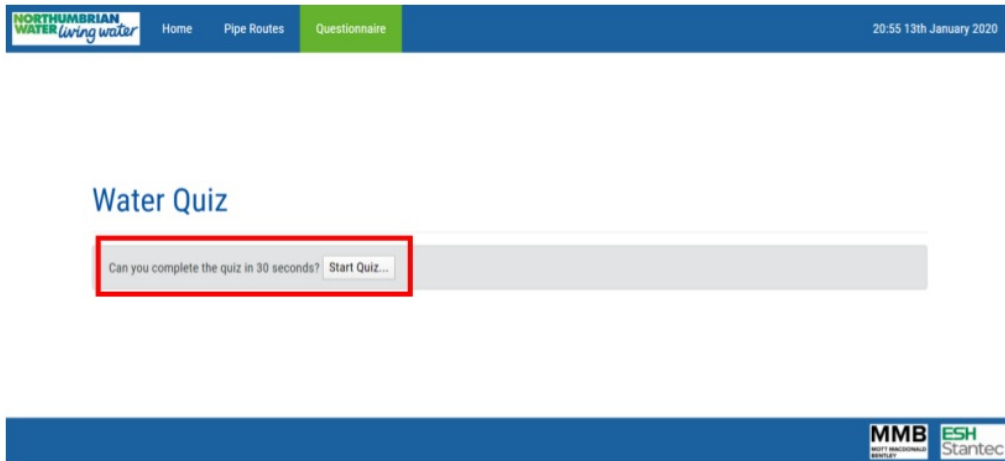


Figure 6: Questionnaire start

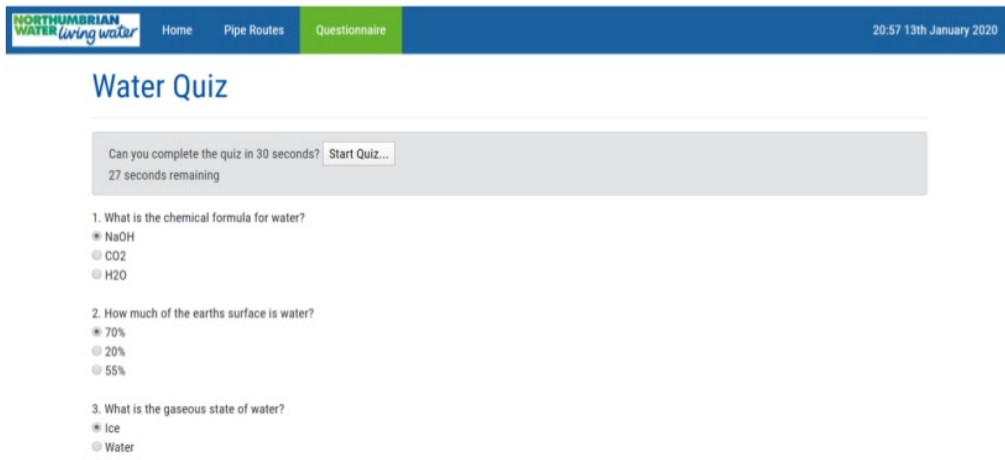


Figure 7: Questionnaire in progress

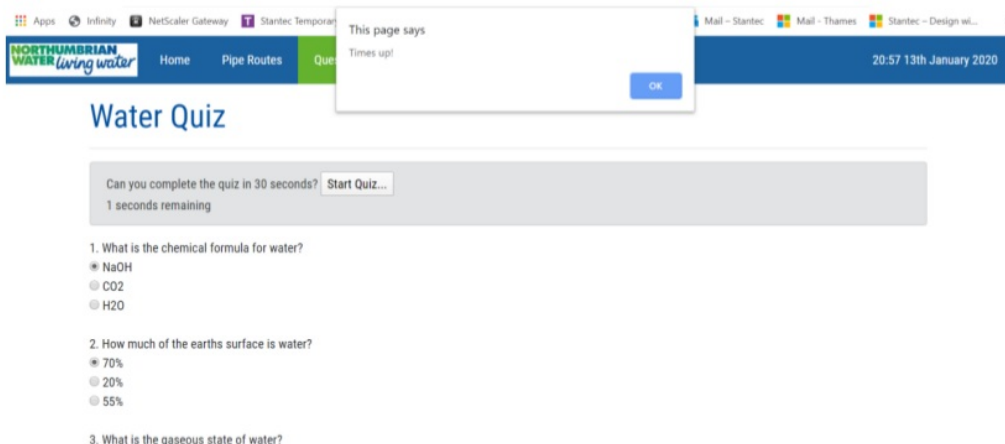


Figure 8: Questionnaire out of time

6. At what temperture does water freeze?

☒ 0 degrees C
☐ -3 degrees C
☐ -1 degrees C

7. At what temperture does water vapourise?

☒ 88 degrees C
☐ 50 degrees C
☐ 100 degrees C

8. How much of the human body is made up of water?

☒ 50%
☐ 66%
☐ 80%

1: Incorrect 2: Correct 3: Incorrect 4: Incorrect 5: Incorrect 6: Correct 7: Incorrect 8: Incorrect
 Well done! You scored: 2

MMB ESH
 water management Stantec

Figure 9: Completed Questionnaire

4.2.1 Questionnaire Code

```
function showquiz() { //function to make the quiz appear
  var show = document.getElementById("quiz"); //connects to the quiz div
  if (show.style.display === "none") { //if the quiz is not displayed then i
t changes the display to make it appear
    show.style.display = "block";
  } else {
    show.style.display = "none";
  }
}

function waterquiz() //function to check the answers and give feedback on the
quiz
{
question1=document.getElementById("H2O") //connects to the correct radio butto
n for each question on the quiz
question2=document.getElementById("70")
question3=document.getElementById("vapour")
question4=document.getElementById("ice2")
question5=document.getElementById("drink")
question6=document.getElementById("0")
question7=document.getElementById("100")
question8=document.getElementById("50")
themessage=document.getElementById("message") //connects to message box to giv
e feedback at the end of the quiz
correct=0 //variable to keep track of correct answers

if(question1.checked) // if statement to check whether the answer to each ques
tion was correct or not
{
```

```

themessage.innerHTML="1: Correct"; // creates a message to the user so they can see what they had correct
correct= correct+1 //adds a correct answer to the total score
}
else{
themessage.innerHTML="1: Incorrect";
}
if(question2.checked)
{
themessage.innerHTML+=" 2: Correct";
correct= correct+1
}
else{
themessage.innerHTML+=" 2: Incorrect";
}
if(question3.checked)
{
themessage.innerHTML+=" 3: Correct";
correct= correct+1
}
else{
themessage.innerHTML+=" 3: Incorrect";
}
if(question4.checked)
{
themessage.innerHTML+=" 4: Correct";
correct= correct+1
}
else{
themessage.innerHTML+=" 4: Incorrect";
}
if(question5.checked)
{
themessage.innerHTML+=" 5: Correct";
correct= correct+1
}
else{
themessage.innerHTML+=" 5: Incorrect";
}
if(question6.checked)
{
themessage.innerHTML+=" 6: Correct";
correct= correct+1
}
else{
themessage.innerHTML+=" 6: Incorrect";
}
if(question7.checked)

```

```

{
thmessage.innerHTML+=" 7: Correct";
correct= correct+1
}
else{
thmessage.innerHTML+=" 7: Incorrect";
}
if(question8.checked)
{
thmessage.innerHTML+=" 8: Correct";
correct= correct+1
}
else{
thmessage.innerHTML+=" 8: Incorrect";
}

if(correct>1){ //if statement to change the feedback and colour of the text based on the amount the user had correct.
    thmessage.innerHTML+=" <br>Well done! You scored: " + correct + "<br><br>"
    thmessage.style.color = "green"
}
else{
    thmessage.innerHTML+=" <br>You scored: " + correct + "<br><br>"
    thmessage.style.color = "red"
}
}

```

4.2.2 Questionnaire Method

- The initial stage of building the quiz function was to connect to all the correct ID's from the HTML for each of the eight questions and connecting to the div that will display the feedback using document.getElementById.
- Once this was achieved the next stage was to validate whether each of the questions were correct. An if statement was created for each of the questions, if the question was correct it added to the feedback message this question was correct and if its incorrect it gets added to the feedback message as incorrect. Also, if a question is correct a score of 1 gets added to the correct variable.
- An if statement is then created to check the number of correct answers. If the user gets more than one correct, the feedback says well done and will be displayed in green. If the user gets one or less correct, the feedback just tells them their score and will be displayed in red.
- The next stage was to create the timer which will be described below.
- After the timer was created it was realised the quiz will need to be hidden before the time is started so a function to show the quiz at the same time the timer starts was created. It connects to the quiz div and checks the style with and if statement. If the style was none then the statement changes it to block using style.display making the quiz appear.

4.3.1 Timer Code

```
function timer2(time) { //timer function to create a second countdown, receive
s the amount of time from the HTML
    myTimer = setInterval(timeLeft, 1000); //sets the countdown interval to
o a second for the timeLeft function
    var timeleft = time; //stores the time from the html

    function timeLeft(){ //function to display to time left
        document.getElementById("timer").innerHTML = --
timeleft + " seconds remaining"; // access the timer ID and deduct time
        if (timeleft == 0) { //when the timer runs out clear the interval and
alert the user
            clearInterval(myTimer);
            alert("Times up!");
        }
    }
}
function end() { //function to clear the interval when the user completes the
quiz
    clearInterval(myTimer);
}
```

4.3.2 Timer Method

- Firstly, a function was created to control the interval in which the countdown function would be run. This was set to one second. This function also receives the amount of time variable from the HTML code and stores it in a variable called timeleft. This function is called when the user presses starts quiz where it will execute every second.
- The next function created was to control the countdown from 30 seconds to zero and to notify the user when the time runs out. It uses the document.getElementById to access the timer div in the HTML so that the user can see the seconds counting down. It counts down by using the decrement operator which decreases the number by one each time the function is executed. It also contains an if statement so that when the timer reaches 0 it clears the interval function and notifies the user the time is up.
- Another function was added at the end so that when the user completes the quiz in the allotted time the interval function is cleared so that the timer stops and shows the user how much time they had left.

Conclusion

This report has evidenced the website that has been created for the web technologies module. Each discrete JavaScript component within the website has been described, annotated and explained. The website contains proof of concepts that will be used in a real work environment to help clients explore different approaches to visualizing their engineering data.

References

- [1] "Shapes | Maps JavaScript API | Google Developers." [Online]. Available: <https://developers.google.com/maps/documentation/javascript/shapes>. [Accessed: 14-Jan-2020].

FINAL GRADE

78/100

GENERAL COMMENTS

Instructor

Range of coding - Couldn't work out how to log in for a while(!)

I know this assignment is JS and not server side, but just be clear that Javascript is not appropriate to use for logins - you did mention this in your report. (It took me a while to work out what the username and password was!)

The pipe work is good, but you might want to think about removing pipelines after use - once you've clicked "All routes" you can't go back to one route unless you refresh the page.

Quiz is quite extensive and I like the countdown - I felt under pressure! :)

48/60 (80%)

Overall integration of website - Generally good with a suitable banner at top and bottom. I would style the Water quiz so it doesn't "jump around" and perhaps make the Google Map a little smaller - I have to scroll to see the buttons on my resolution 7.5 /10

Analysis of technical development -

You've talked well about what you did technically, but I'd like a bit more on WHY you took the technical approach you did. eg Why did you use the API rather than write your own Javascript,

Annotated code listing(s) -

Clearly annotated and quite easy to follow from the code comments.

13/15

Total: 77.5%

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15
