

COMP3322A Modern Technologies on World Wide Web

Lab 2: Responsive Web Design and JavaScript

Overview

In this lab exercise, we will first enhance the web page we have created in Lab 1 with responsive web design. Then, we will practice JavaScript by processing the survey results.

desktop/laptop

Course Evaluation
COMP3322A Modern Technologies on World Wide Web

Home Create New Survey Statistics Support

Instructions

- Questions marked with * are required.
- The survey is anonymous.

***1. Would you recommend this course to a fellow student?**

☐ Yes ☐ No

***2. Which parts of the course do you like most?**

☐ WWW Basics ☐ HTML/CSS ☐ JavaScript, DOM ☐ Node.js
☐ JSON, MongoDB ☐ ReactJS

3. How many lectures have you skipped?

reset submit

Back to top

tablet

Course Evaluation
COMP3322A Modern Technologies on World Wide Web

Home Create New Survey Statistics Support

Instructions

- Questions marked with * are required.
- The survey is anonymous.

***1. Would you recommend this course to a fellow student?**

☐ Yes ☐ No

***2. Which parts of the course do you like most?**

☐ WWW Basics ☐ HTML/CSS ☐ JavaScript, DOM ☐ Node.js
☐ JSON, MongoDB ☐ ReactJS

3. How many lectures have you skipped?

reset submit

Back to top

smaller mobile devices

Course Evaluation
COMP3322A Modern Technologies on World Wide Web

Home
Create New Survey
Statistics
Support

Instructions

- Questions marked with * are required.
- The survey is anonymous.

***1. Would you recommend this course to a fellow student?**

☐ Yes ☐ No

***2. Which parts of the course do you like most?**

☐ WWW Basics
☐ HTML/CSS
☐ JavaScript, DOM
☐ Node.js
☐ JSON, MongoDB
☐ ReactJS

3. How many lectures have you skipped?

reset submit

Back to top

Figure 1. Responsive Web Design. The screenshots show the webpage under different screen widths

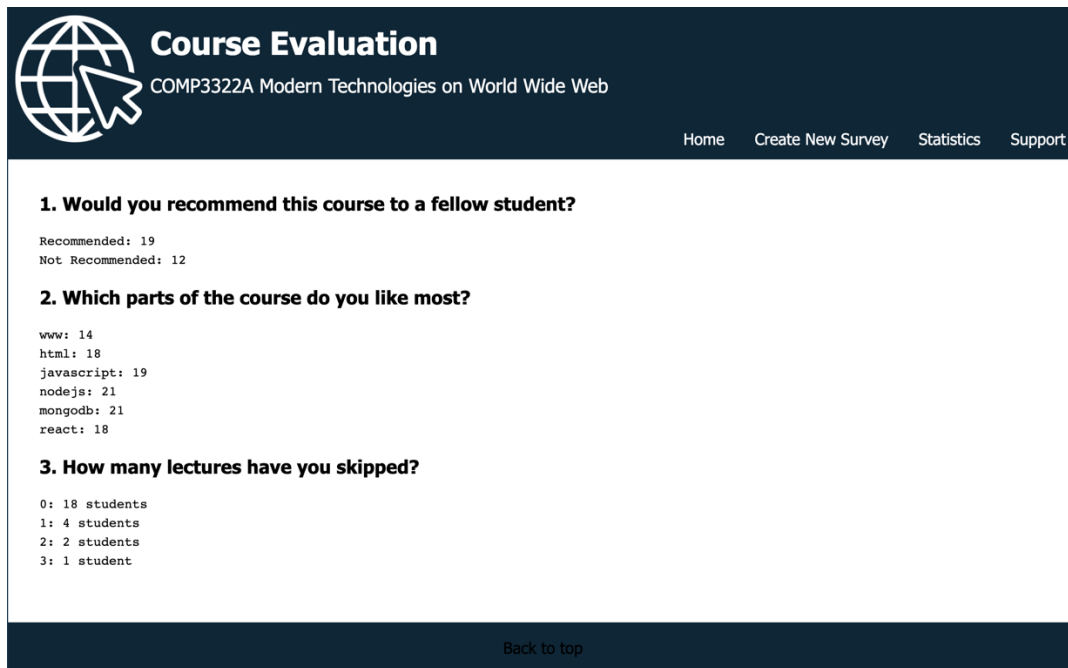


Figure 2. The statistics page.

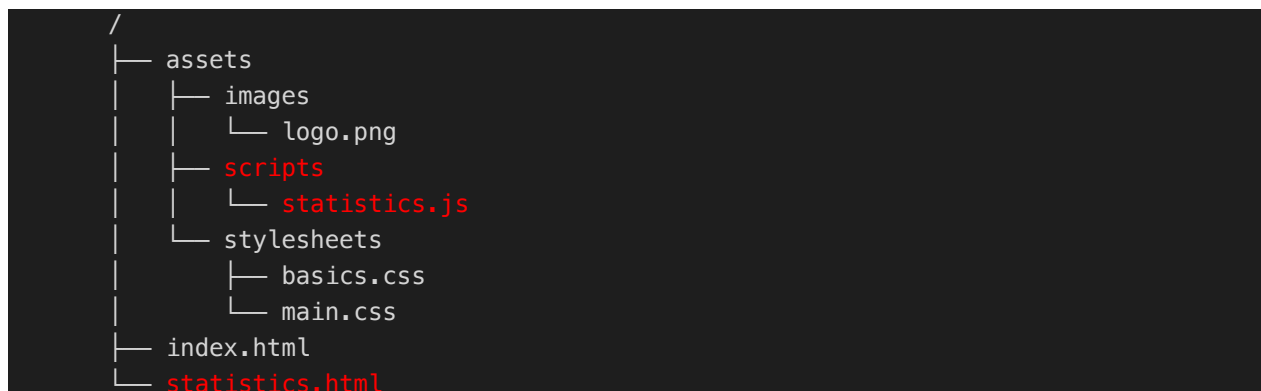
Note: Due to compatibility of different browsers, the same CSS code may lead to different display; we will check the assignment in the Chrome browser. Therefore, you are recommended to use the Chrome browser to develop/test your page as well.

Lab Exercise

Part 1. Preparation

Step 1. Download the template code from Moodle

Download "**lab2_materials.zip**" from HKU Moodle, and extract it to a folder. In this folder, you will find files structured like the following:



This time, you'll see a new page (statistics.html) and a directory containing the JavaScript file we will be working on (statistics.js). In this lab, we will **only** modify the contents of "**index.html**",

“statistics.html”, “main.css” and “statistics.js”. These files can be edited using any code editors or IDEs. The contents of other files are largely the same as in Lab 1, with some modifications.

Part 2. Add responsiveness to our web pages

Step 2. Setting the viewport

First, we set the viewport width to fit the device by adding the following element in the **<head>** section of “index.html” and “statistics.html”:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Step 3. Add media queries to change CSS styles according to screen width

We first make our nav bar responsive using media queries. In “main.css”, we can find two commented blocks specifying the rules to enable when the screen width is no larger than **1080px** and **720px**, respectively, as follows.

When the screen width is no larger than **1080px**, add the following rules:

```
header,nav {
  overflow: hidden;
}
nav {
  display: inline-block;
  width: 100%;
  text-align: center;
}
nav div {
  margin-top: 0;
}
```

This removes the top margin of the nav bar contents and sets the nav bar’s width to 100%, so it takes up the width of the entire page and is wrapped around (read more about CSS overflow property at https://www.w3schools.com/cssref/pr_pos_overflow.asp).

When the screen width is no larger than **720px**, further add the following rules:

```
header img {
  display: none;
}
header h1 {
  padding: 0 0.5em;
}
nav {
  text-align: left;
}
nav div {
  width: 100%;
}
```

This further allows each element in the nav bar to take up an entire row, changes the text alignment of the nav bar to the left, hides the logo and adjusts the padding around the page title.

Add appropriate media queries to implement the above responsive web design. After this step, when you adjust the width of your browser, you'll see the nav bar changes its appearance under different screen widths, as shown in Figure 1.

Step 4. Implement a responsive grid-view

We use the grid-view (refer to pages 35-39 of lecture notes "5_cssbasics_COMP3322A_f2022.pdf") to align the choices under question 1 and question 2 on webpage **index.html**. We have enclosed the input and label of each choice in a **<div>** element in "**index.html**" and provided some CSS rules in "**main.css**". **Set the class attribute in these <div> elements in "index.html"** and **add CSS rules in "main.css"**, such that: when the screen width is larger than 720px, at most 4 choices are displayed on the same row under each question, with each choice taking up 1/4 of the form width; when the screen width is no larger than 720px, one choice is displayed per row under each question; in both cases, each choice should float to the left, as shown in Figure 1.

After this step, the "**index.html**" page should look the same as Figure 1 under different screen widths.

Part 3. Summarize the survey statistics using JavaScript

In this part, we summarize some statistics of the survey results and display them on the "**statistics.html**" page. The JavaScript code for showing the results on the page are given (function `show_results()` in `statistics.js`) and we focus on implementing the results counting code.

Step 5. Link JavaScript file to the HTML page

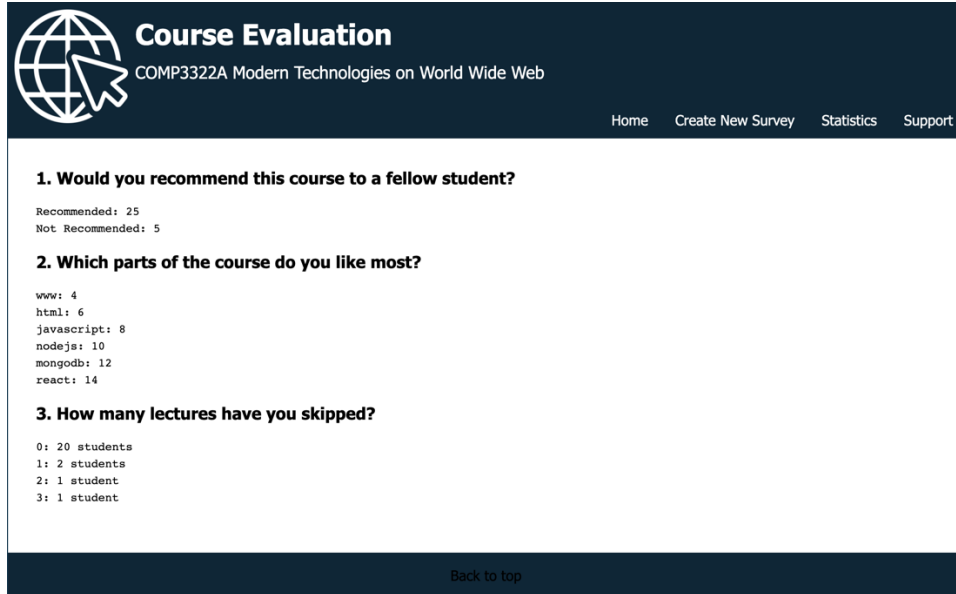
Now we add the JavaScript code in "**statistics.js**" to "**statistics.html**". This is done by adding the following code in the HTML file's **<head>** section:

```
<script src="assets/scripts/statistics.js"></script>
```

Step 6. Handle "onload" event on the page

Locate the function named "**show_results**" in "**statistics.js**", which we want to run when the page is loaded. To achieve this, call this function when "**onload**" event is triggered on the **<body>** element in "**statistics.html**".

After this step, reloading the page leads to some statistics displayed under each question as in the following screenshot. Note that we are currently using some placeholder statistics; in the following steps, we will write JavaScript code to count the actual statistics.



Course Evaluation
COMP3322A Modern Technologies on World Wide Web

Home Create New Survey Statistics Support

1. Would you recommend this course to a fellow student?

Recommended: 25
Not Recommended: 5

2. Which parts of the course do you like most?

www: 4
html: 6
javascript: 8
nodejs: 10
mongodb: 12
react: 14

3. How many lectures have you skipped?

0: 20 students
1: 2 students
2: 1 student
3: 1 student

Back to top

Step 7. Implement the “get_recommendation_data” function

In “**statistics.js**”, there is a function called “**get_raw_data**” that simulates getting the survey results from the server. It returns an **array of objects**, where each element of the array is an object that contains the following properties:

- **recommend**: a **boolean** value indicating whether the student recommends this course or not.
- **favourites**: an **array of strings** that contains a student’s favourite topics. Possible values include "www", "html", "javascript", "nodejs", "mongodb", and "react". There are no duplicates in this array.
- **skipped**: a **number** representing the number of lectures the student has skipped. Since the corresponding question is optional, this property can be **null**.

In this step we implement the “**get_recommendation_data**” function, which processes the data returned by `get_raw_data()` and counts the number of students who recommend or do not recommend this course. You should revise the placeholder numbers given in the function in “**statistics.js**” and implement the above. [Hint: use **for (..of..)** loop on the array returned by `get_raw_data()`.]

Step 8. Implement the “get_favourites_data” function

Similar to Step 7, implement the “**get_favourites_data**” function: for each topic (among "www", "html", "javascript", "nodejs", "mongodb" and "react"), count the number of students who select it as favourite. You should revise the placeholder numbers given in the function in “**statistics.js**” and count the numbers from the data returned by `get_raw_data()`. The return value of the function should be an **object** that contains a property for each topic (the value of the property is the number of students who select the topic as favourite).

Note: The order of the topics in the result page can differ from that shown in Figure 2 and it does not matter.

Step 9. Implement the “get_skipping_data” function

Similar to Step 7 and Step 8, implement the “**get_skipping_data**” function that counts the numbers of students who skipped 0, 1, ... lectures. Note that as answering the corresponding question on the index.html page is optional, the “skipped” property in an element in the array returned by get_raw_data() may be **null**; those records should be ignored in the number counting. The return value of the function should be an array in which the i-th element is the number of students who skipped i lectures.

Now you have finished the entire lab! The resulting web page should look like the screenshot on the first two pages of this document.

Submission:

Please finish this lab exercise before **23:59 Thursday Sep. 29th, 2022**. Please compress the entire project folder (i.e., the folder containing your two **html files** and the entire **assets** folder with **main.css** and **statistics.js** modified) into a .zip file and submit it on Moodle.

- (1) Login Moodle.
- (2) Find “Labs” section and click “Lab 2”.
- (3) Click “Add submission”, browse your .zip file and save it. Done.
- (4) You will receive an automatic confirmation email, if the submission was successful.
- (5) You can “Edit submission” to your already submitted file, but **ONLY** before the deadline.