COMP3322A Modern Technologies on World Wide Web

Lab 3: HTML with Scripts

Overview

In this lab exercise, we first add form validation to the web page we have created in Lab 1 using DOM API. Then, we add a pie plot to the statistics page that we have created in Lab 2 using canvas.

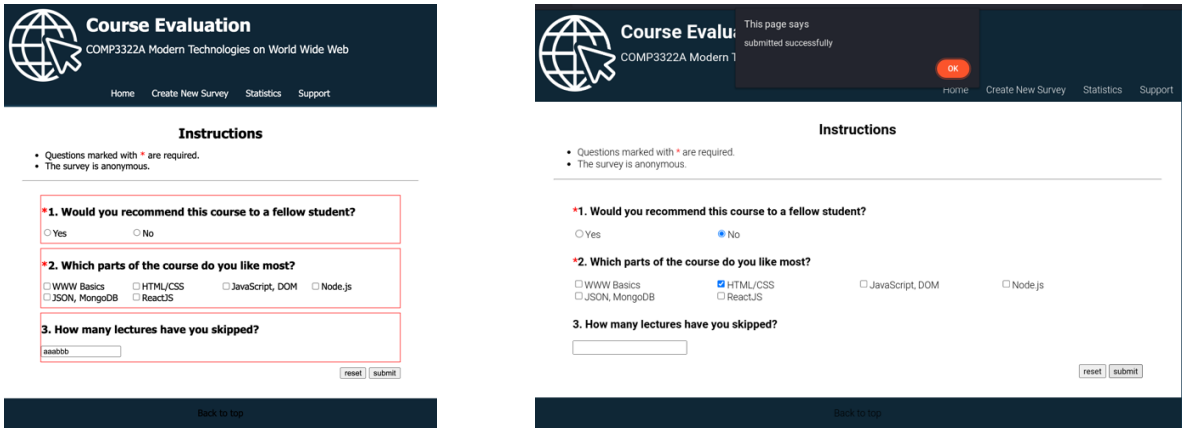


Figure 1. The index page. After clicking the "submit" button, invalid inputs are outlined in red (left). If all inputs are valid, a pop-up window shows "submitted successfully" (right).

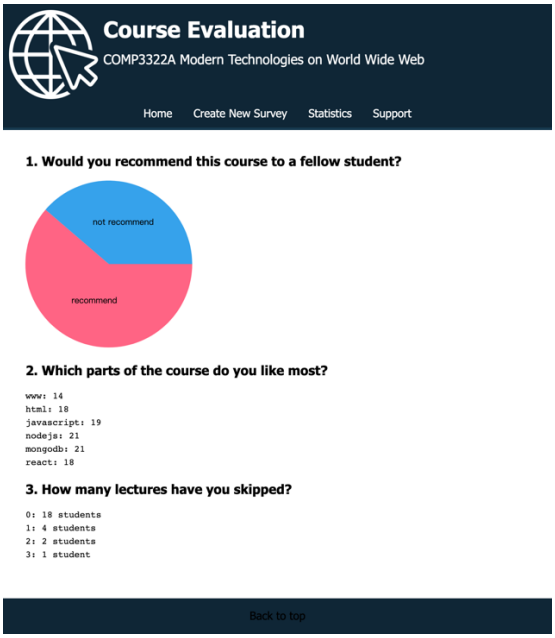


Figure 2. The statistics page.

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 "**lab3_materials.zip**" from HKU Moodle, and extract it to a folder. In this folder, you will find files structured like the following:

```
/
├── assets
│   ├── images
│   │   └── logo.png
│   ├── scripts
│   │   ├── index.js
│   │   └── statistics.js
│   └── stylesheets
│       ├── basics.css
│       └── main.css
├── index.html
└── statistics.html
```

We added a new file "**index.js**" and loaded it in "**index.html**". In this lab, we will modify the contents of "**index.js**" 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 2, with following modifications:

1. In the bottom of "**index.html**", we added a **<script>** tag to load the newly added "**index.js**".
2. An onsubmit event handler is set for the **<form>** element in **"index.html"**. The function body can be found in "**index.js**". We also changed the "action" attribute of the form to "javascript:alert(…)". This instructs the browser to show a pop-up window instead of actually submitting the data to server.
3. The **type** attribute of **<input>** for the third question in "**index.html**" has been changed to "text" and we will validate its value using Javascript.
4. At the bottom of "**main.css**", we added a new rule for showing a red border around elements that belong to both the "**question**" class and the "**invalid**" class. We will use this for outlining invalid inputs.
5. In "**statistics.html**", the statistical result of the first questions is replaced by **<canvas>**, on which we will draw a plot in this lab.
6. In the **show_results** function in "**statistics.js**", we now call **plot_recommend** to show the result of the first question instead of showing it in text. We will be working on this function in Part 3.

Part 2. Implement form validation

Step 2. Check the validity of user inputs

Complete the "**validation**" function in "**index.js**", such that it returns true or false according to whether the user inputs are valid. The validity of inputs are defined as follows:

- For the first question, either Yes or No needs to be checked.
- For the second question, at least one of the topics must be checked.

- For the third question, the value must either be empty, or contain only digit characters, e.g., “0”, “25”, “109”. [**Note**: the “**value**” property of a text input element in HTML DOM returns a string that represents the value of the text field.]

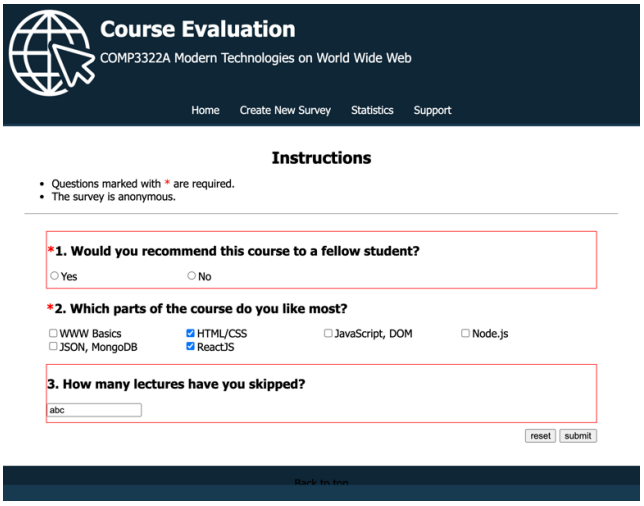
Hint: Use **document.getElementById** or **document.getElementsByTagName** to get the **<input>** elements; then read the “**checked**” property for radio buttons and checkboxes, and the “**value**” property of the text input element. You may use the “**name**” property of input elements to identify particular elements (see https://www.w3schools.com/jsref/prop_radio_name.asp, https://www.w3schools.com/jsref/prop_checkbox_name.asp and https://www.w3schools.com/jsref/prop_text_name.asp).

In addition, when the inputs are invalid, we add a red border around the related question. The following CSS rules has already be given at the bottom of “**main.css**”.

```
.question.invalid {
    border: 1px solid red;
}
```

In the “**validation**” function, add the “invalid” class to each ** ** element (of id “q1”, “q2” or “q3”) that contains a question with invalid inputs. [**Hint**: use the **classList** property of elements in HTML DOM to manipulate their class attributes. add(), remove(), contains() functions of the **classList** property might be useful. See https://developer.mozilla.org/en-US/docs/Web/API/Element/classList.

After this step, when clicking the submit button while some inputs are invalid, the pop-up window will not show, and the corresponding questions will be surrounded by red boxes, like in the following screenshot:



Step 3. Handle input events

In this step, we will handle the **input** event (https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/input_event) of the **<input>** elements: when the value of an **<input>** element has been changed, the **input** event is triggered, and we will remove the red box of the corresponding question if its inputs become

valid. In "**index.js**", we have provided the JavaScript code that registers event handler function to handle the **input** event on the ** ** of each question (that contains the **<input>** elements and hence captures the **input** event as well). Implement the three handler functions: each handler should check if the inputs in the respective question are valid and if so, remove the "invalid" class from this containing ** ** (whose **input** event we are handling).

Further, add an event handler for the "**reset**" event of the **<form>**, which is trigger when the "reset" button in the form is clicked (see https://developer.mozilla.org/en-US/docs/Web/API/HTMLFormElement/reset_event). In the event handler, remove the "invalid" class from all ** ** elements that contain questions.

After this step, when a question's input is fixed (e.g., clicking either Yes or No in the first question), the surrounding red box will immediately disappear. When the "reset" button is clicked, all red boxes will disappear.

Part 3. Show the statistics on canvas

Now we move to the "**statistics.html"** page. We will draw a pie plot for the survey results of the first question, similar to Figure 2. We have replaced the **<pre>** element used in Lab 2 with **<canvas>** in "**statistics.html"**. We are going to write JavaScript in "**statistics.js**" to draw on the **<canvas>**.

In "**statistics.js**", a function called "**set_up_canvas**" is provided to make the canvas look more clear on high-resolution displays (refer to https://developer.mozilla.org/en-US/docs/Web/API/Window/devicePixelRatio for details). We have called this function in **plot_recommend**, the function we will be working on in this part.

Step 4. Draw the pie plot

As Figure 2 shows, the pie plot contains two slices showing the ratio of students who recommend this course and who don't. Complete the function **plot_recommend** using canvas APIs to draw the pie as in Figure 2.

Hint 1: to draw a slice, first call **ctx.beginPath**, then **ctx.moveTo** to the center of the canvas and use **ctx.arc** to draw the arc (you should figure out the parameters to call **ctx.arc** according to the percentage of students who recommend this course or who don't), and finally call **ctx.closePath** and use **ctx.fill** to fill the colors (refer to pages 7 and 8 of lecture notes "**7_HTML_withScript_COMP3322A_f2022.pdf**").
Hint 2: **ctx.arc** uses radian as the unit for angles. You may use **Math.PI** (=3.14…) for related calculation.
Hint 3: to draw the text in each slice, use **ctx.fillText** (https://www.w3schools.com/tags/canvas_filltext.asp). You can place the text anywhere inside the corresponding slice. The **ctx.textAlign**

(https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/textAlign) property can be used to better place the text.

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

Submission:

Please finish this lab exercise before 23:59 Thursday Oct. 6th, 2022. Please compress the entire project folder (i.e., the folder containing your two **html files** and the entire **assets** folder with **index.js** and **statistics.js** modified) into a .zip file and submit it on Moodle.

(1) Login Moodle.
(2) Find "Labs" section and click "Lab 3".
(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.