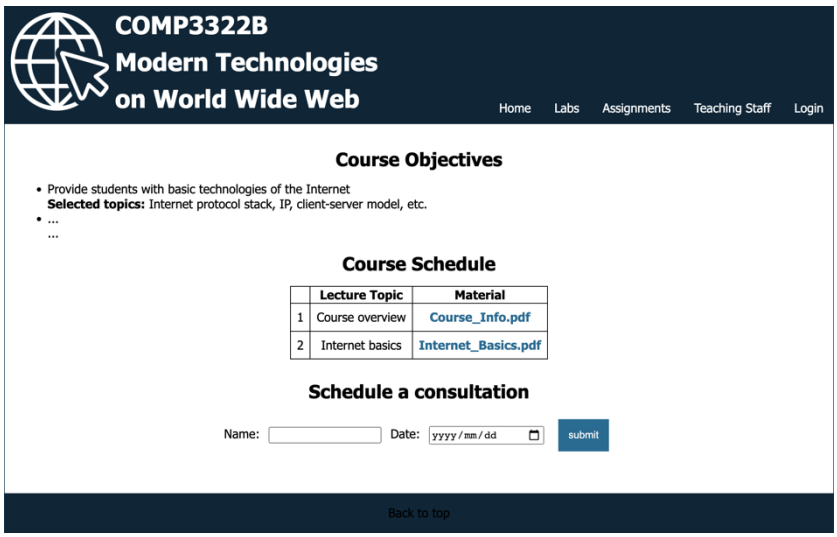
COMP3322B Modern Technologies on World Wide Web

Lab 2: Responsive Web Design and JavaScript

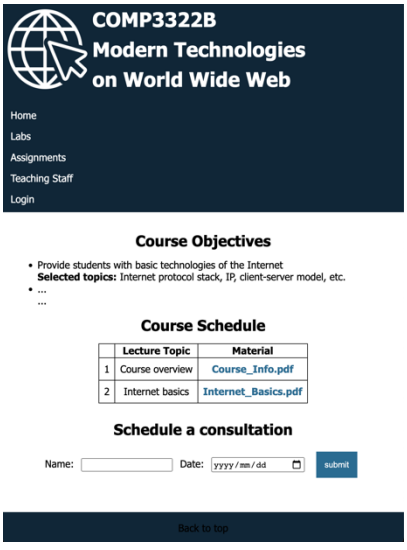
Overview

In this lab exercise, we will first enhance the web page we have created in Lab 1 with responsive web design. Then, we will use JavaScript to simulate a login feature and perform validation for the form.



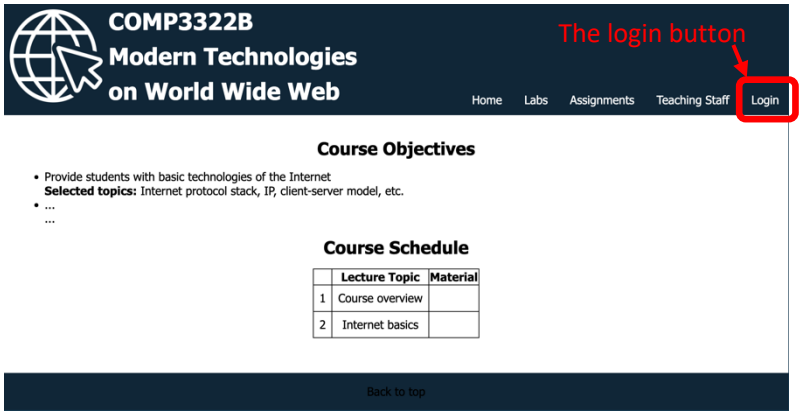
laptop

tablet

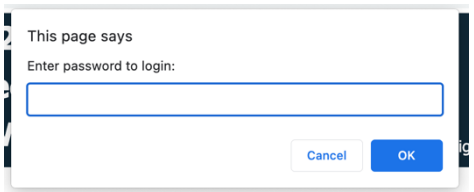


smaller mobile devices

*Figure 1. Responsive Web Design. The screenshots show the webpage (**after** "login") under different screen widths. (The format for the "Date" field may look different depending on your browser and localization settings.)*



(a) webpage before "login"



(b) prompt after clicking the login button

(c) alert if password does not match

Figure 2. The "login" feature. Course materials and the "Schedule a consultation" section are initially hidden. When the login button is clicked, a prompt box will pop up and ask you to enter a password. If the password is correct, then the webpage will show the hidden contents as in Figure 1; otherwise, an alert box shows up ("Login Failed") and the page remains the same.

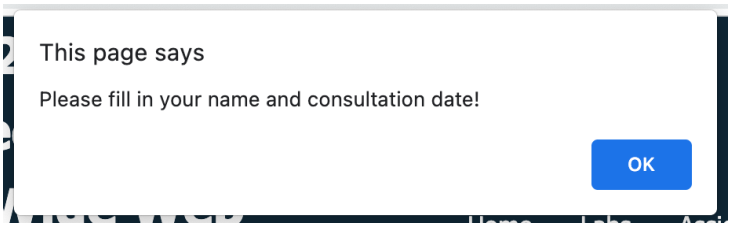


*Figure 3. Form validation. A pop-up box appears if you leave the name **or** date empty and click the form's submit button. The form will not be submitted in this case.*

Note: Due to compatibility of difference 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:

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

This time, you'll see a new directory containing the JavaScript file we will be working on. The contents of other files are mostly the same as Lab 1 except for some minor modifications. In this lab, we will **only** modify the content of "**index.html**", "**main.css**" and "**index.js**". The three files can be edited using any code editors or IDEs.

Part 2. Add responsiveness to our webpage

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**":

```
<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 make our webpage responsive by changing the style of the nav bar using media query. Near the end of "**main.css**", we can find two block comments specifying the rules to enable when the screen width is no larger than **1080px** and **720px**.

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

```
header {
  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. We also adjust the overflow property of the header so its size will adjust according to the nav bar.

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

```
nav {
  text-align: left;
}
nav div {
  width: 100%;
}
```

This further lets each element in the nav bar to take up an entire row, and change the text alignment of the nav bar to the left.

Add appropriate conditions using media query to implement such responsive web design (https://www.w3schools.com/css/css_rwd_mediaqueries.asp). After this step, when you adjust the width of your browser, you'll see the nav bar changes its appearance under different screen widths, like in Figure 1.

Part 3. Add the "login" feature

In this part, we simulate a login feature which hides the course materials and the "Schedule a consultation" section until the user enters the correct password.

Step 4. Link the JavaScript code to the HTML page

Now we will add our JavaScript template code to the HTML page. This is done by adding

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

in the HTML file's **<head>** element.

Step 5. Handle "onload" event on the page

Opening "**index.js**", you will see several skeleton functions and objects. Locate the function named "**hideContents**", which we want to run when the page is loaded. We call this function when the "**onload**" event is triggered on the **<body>** element of the HTML page. The content of the function will be implemented in later steps.

Step 6. Implement the "hideElement" function

This function hides the input element by setting its display style. It also stores the original display style into a cache so we can "unhide" it later.

Note that in the beginning of "**index.js**", we have defined a global variable named "**hidden_elements**" as an empty array. This will be our temporary cache. For each element we hide using "**hideElement**", we create an object with the following properties:

```
var hidden_element = {
        "element": the HTML element object
        "display": the original display value
}
```

The first property named "element" refers to the HTML object that we hide. The second property stores its display style which can be obtained as:

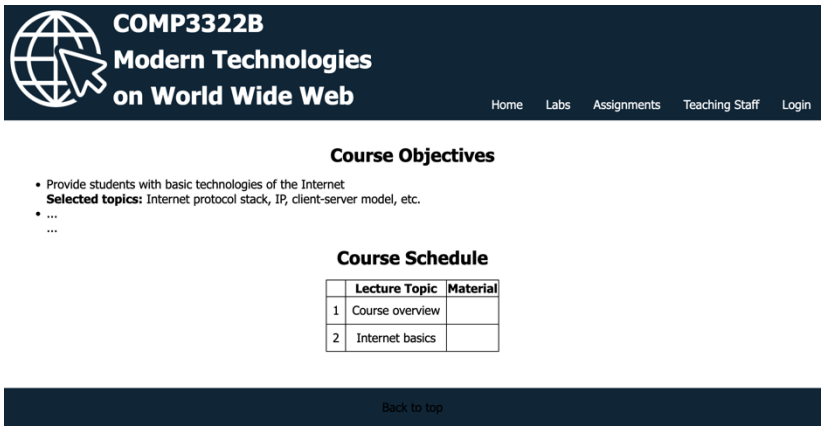
```
element.style.display
```

In "**hideElement**", you will notice a statement setting this property of the input element to "none". Before we do that, **create the object** as decribed above and **push** it into the "**hidden_elements**" array (https://www.w3schools.com/jsref/jsref_push.asp).

Step 7. Finish the "hideContents" function

Locate the "**hideContents**" function. You will see a statement which returns an array of objects of HTML elements in the "needs_login" class and stores them in the variable "**elements**". The course material links and all elements in the "Schedule a consultation" section are in the "needs_login" class. You should **apply** the "**hideElement**" function we implemented in the last step on each element in "**elements**" to hide them.

After this step, after you refresh the page, you will no longer see the course material links and the "Schedule a consultation" section:



Step 8. Implement the "displayHiddenElements" Function

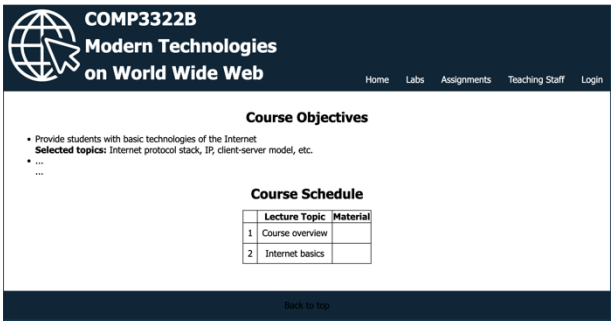
Locate the "**displayHiddenElements**" function. In this function, we set the **display style** (element.style.display) back to its **original value** for every element we have hidden. Implement the function by using information stored in "**hidden_elements**". The "**hidden_elements**" array should be **emptied** afterwards.

Step 9. Implement the "login" function

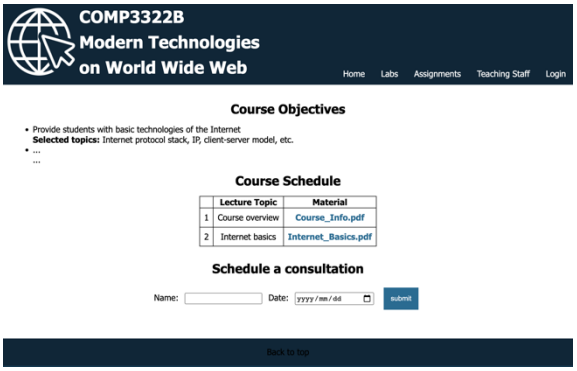
First, in "**index.html**", in the **<nav>** section of the HTML page, locate the **<a>** element with text "Login". We turn it into a login button by calling the JavaScript function "**login()**" upon its **onclick** event.

Next, in "**index.js**", implement the function "**login**". When it's called, a **prompt** should show, displaying the text: "**Enter password to login:**". It then checks if the user's input value exactly matches our password "**comp3322**" (case sensitive). If so, it then calls the function: "**displayHiddenElements**". Otherwise, it **alerts** "**Login Failed.**" and the page will stay the same.

After this step, the behavior of the button will look like Figure 2. When the password is entered correctly, the hidden elements will be back:



before "login"



after "login"

Part 3. Form validation

Step 10. Implement form validation

Finally, we add simple validation feature for the form in "Schedule a consultation" section. This should be implemented in the "**formValidation**" function (in "**index.js**") and the function should be called when the form is submitted (in "**index.html**"). We check if the user has filled in the contents for **both "Name" and "Date" fields** when the form is submitted. If not, an **alert** showing "**Please fill in your name and consultation date!**" will be displayed, and the **form will not be submitted** (Figure 3). Please see lecture notes p26-p29 of 6_JavaScript_COMP3322B_s2022.pdf for refererence. If both fields have been filled, after clicking the submit button, the page will stay the same or display a error (since we did not provide the server-side code "reg_consult" for handling the form submission in this lab exercise).

Congratulations! By now we have finished Lab 2. You can test if the login and form validation features are working correctly. The final result should look similar to the screenshots at the front of this document.

Submission:

Please finish this lab exercise before 23:59 Wednesday Feb. 16th, 2022. Please compress the entire project folder (i.e., the folder containing your **index.html** and the entire **assets** folder with **main.css** and **index.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.