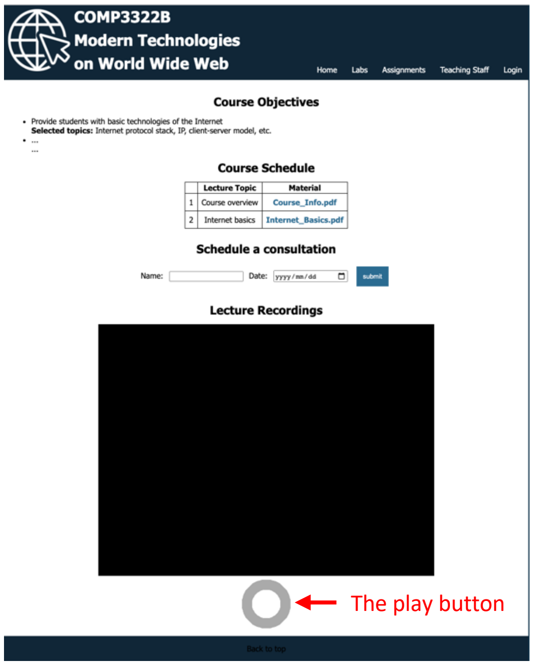
COMP3322B Modern Technologies on World Wide Web

Lab 3: HTML with Scripts

Overview

In this lab exercise, we add a feature to the web page we have created in Lab 1 and Lab 2 to better facilitate remote learning. We will display a lecture recording on the webpage and create a mini quiz after the user finishes viewing the video. Please see the following screenshots for details:



Course Objectives

- Provide students with basic technologies of the Internet
  Selected topics: Internet protocol stack, IP, client-server model, etc.
- ...
  ...

Course Schedule

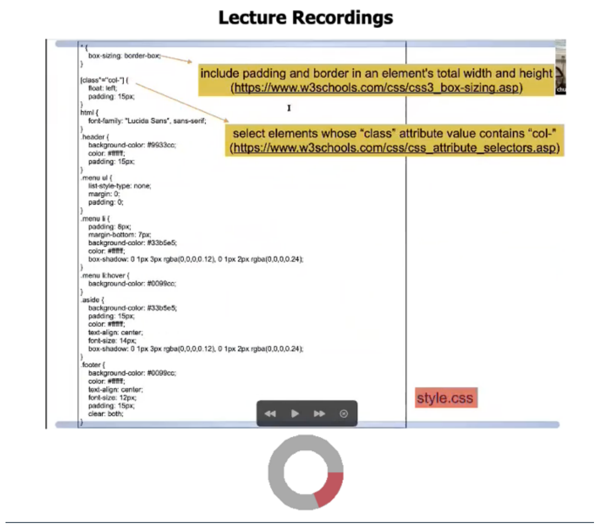
	Lecture Topic	Material
1	Course overview	Course_Info.pdf
2	Internet basics	Internet_Basics.pdf

Schedule a consultation

Name: Date: yyyy/mm/dd submit

Lecture Recordings

Webpage after loading



Lecture Recordings

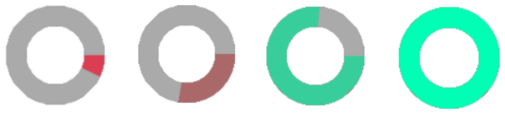
```
* {
   box-sizing: border-box;
}

[class*="col-"] {
   float: left;
   padding: 15px;
}
html {
   font-family: "Lucida Sans", sans-serif;
}
.header {
   background-color: #9933cc;
   color: #ffffff;
   padding: 15px;
}
.menu ul {
   list-style-type: none;
   margin: 0;
   padding: 0;
}
.menu li {
   padding: 8px;
   margin-bottom: 7px;
   background-color: #33b5e5;
   color: #ffffff;
   box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}
.menu li:hover {
   background-color: #0099cc;
}
.aside {
   background-color: #33b5e5;
   padding: 15px;
   color: #ffffff;
   text-align: center;
   font-size: 14px;
   box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}
.footer {
   background-color: #0099cc;
   color: #ffffff;
   text-align: center;
   font-size: 12px;
   padding: 15px;
   clear: both;
}
```

include padding and border in an element's total width and height (https://www.w3schools.com/css/css3_box-sizing.asp)

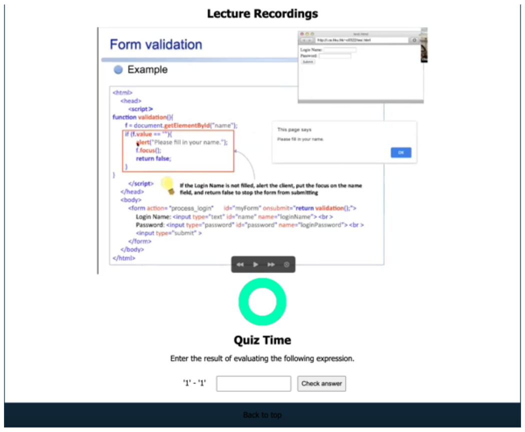
select elements whose "class" attribute value contains "col-" (https://www.w3schools.com/css/css_attribute_selectors.asp)

After moving your mouse over the play button, the video starts to play. Moving the mouse out of the play button will cause video to pause.



The play button's shape and color will change according to how long the video has been played. (It is a circular progress bar)

Figure 1. Displaying the lecture recording video.



Lecture Recordings

Form validation

Example

```
<html>
  <head>
    <script>
function validation(){
    f = document.getElementById("name");
    if (f.value == ""){
        alert("Please fill in your name.");
        f.focus();
        return false;
    }
}
    </script>
  </head>
  <body>
    <form action= "process_login"    id="myForm" onsubmit="return validation();">
        Login Name: <input type="text" id="name" name="loginName"> <br >
        Password: <input type="password" id="password" name="loginPassword"> <br >
        <input type="submit" >
    </form>
  </body>
</html>
```

If the Login Name is not filled, alert the client, put the focus on the name field, and return false to stop the form from submitting

This page says

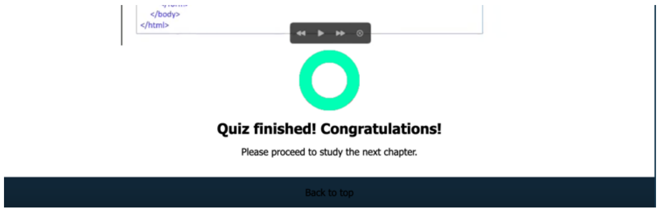
Please fill in your name.

Quiz Time

Enter the result of evaluating the following expression.

'1' - '1'

After finishing watching the lecture recording



```
  </body>
</html>
```

Quiz finished! Congratulations!

Please proceed to study the next chapter.

Once the quiz answer is correctly entered and the button is clicked, the line containing quiz input will be removed and the heading "Quiz Time" and the quiz description will be replaced by the texts above.

Figure 2. The "quiz" feature.

Note: Due to compatibility of difference browsers, the same CSS code may lead to different display; we will check the lab 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:

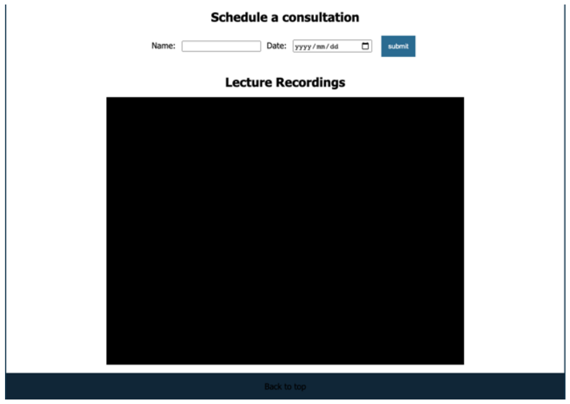
```
/
├─ assets
|  ├─ images
|  |  └─ logo.png
|  ├─ stylesheets
|  |    └─ main.css
|  |    └─ basics.css
|  └─ scripts
|  |    └─ index.js
|  └─ videos
|       └─ lecture_recording.mp4
└─ index.html
```

This time, you'll see a new folder named video containing the video file we will use in this lab. The other files are mostly the same as Lab 2 except for some modifications. In this lab, we will **only** modify the content of "**index.html**" and "**index.js**". The two files can be edited using any code editors or IDEs.

Part 2. Display and control the lecture recording video

Step 2. Adding the video tag

First, open "**index.html**". At the end of the <section> tag, we can see a new <div> container with a <h2> heading which reads "Lecture Recordings". Right under the heading, we insert a **<video>** element with (1) class **"center"** (this makes the video appear in the center of the page), make it (2) **muted** (https://www.w3schools.com/tags/att_video_muted.asp), (3) **disable the controls**, and (4) assign it an **id** of "**lecture_video**". Inside the video element, we include a **<source>** which links to the **mp4 video** file we have included in the *assets* folder. After this step, you should see a black box appearing:

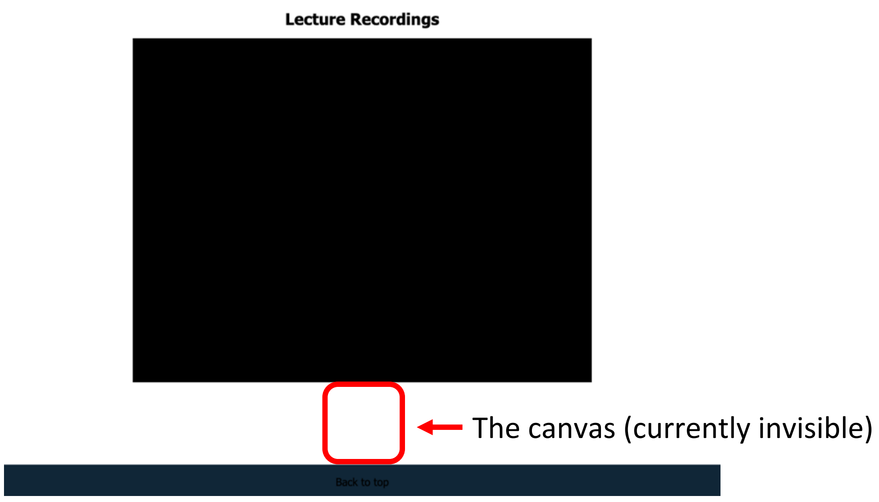


This is because we haven't started to play the video yet. We now add our own controls for the video.

Step 3. Add the canvas

In this lab, we use HTML5 canvas to draw the play button. Below the video element we just created, insert a **<canvas>** with **width="100"** and **height="100"**, and assign it an **id** of "**playButtonCanvas**" so that we can access it from JavaScript easily.

After this step, you should see a blank space appearing under the video. The newly added canvas is already in the middle of the blank space, but is pure white because we have not drawn anything on it.



Step 4. Controlling the video

Now open "**index.js**". This time we have removed the login related functions, and changed the function to call when page is loaded from "hideContents" to "**init**". In "**init**", use "**addEventListener**" to handle events of the canvas, so that "**playVideo**" is called when **mouse enters** the canvas ("mouseenter" event), and "**pauseVideo**" is called when **mouse leaves** the canvas ("mouseleave" event). (For reference, see page 42 of *6_JavaScript_COMP3322B_s2022.pdf*).

After that, **implement** functions "**playVideo**" and "**pauseVideo**" to actually play and pause the video that we added in Step 2 (For reference, see pages 18-20 of *7_HTML_with_Script_COMP3322B_s2022.pdf*). You will be able to control the video by moving your mouse over the canvas after this step.
(Note: the video we provided is a lecture recording that has been cut and sped up significantly. It only has a duration of around 6 seconds.)

Step 5. Implement functions to draw the play button

Locate the functions "**colorGradient"**, "**drawArc**" and "**tick**". These are the functions we need to draw the play button.

1. “**colorGradient**” is already implemented, which takes a number between 0 and 1 (“**percentage**”) as input and returns a string representing a RGB color. This function calculates the linear interpolation between “start_rgb” (red) and “end_rgb” (green).
2. “**drawArc**” takes two arguments: “**color**”, a color string which defines color of the arc to draw, and “**percentage**”, a number between 0 and 1 which controls the length of the arc (in percentage of a full circle).
3. “**tick”** does not take any argument. Inside “**tick**”, we will call “**colorGradient”** and “**drawArc**” to update the circular progress bar of the canvas. The percentage arguments of “**colorGradient**” and “**drawArc**” will be determined by the current progress of the video when “**tick**” is called. “**tick**” will be called periodically so the circular progress bar always matches the progress of the video.

5.1. We first implement “**drawArc**”. We set the origin of the circle to the **center** of the canvas, use a **radius** of **half canvas width – 10**, and set the **line width** of the stroke to **20**. The arc begins at a angle of 0 and its ending point is determined by the “**percentage**” argument. (Hint: “**percentage**” is a value between 0 (corresponding to an arc of 0 degrees) and 1 (an arc of 360 degrees), and degree can be converted to radian by $radian = \frac{degree}{180} * \pi$). Its color should be determined by the “**color**” argument.
(For reference, see https://www.w3schools.com/tags/canvas_arc.asp,
https://www.w3schools.com/tags/canvas_linewidth.asp, and
https://www.w3schools.com/tags/canvas_strokestyle.asp.)

5.2. Next, we implement “**tick**”. In “**tick**”, we first get the **current time** of the video (https://www.w3schools.com/tags/av_prop_currenttime.asp) and divide it by the video’s duration (https://www.w3schools.com/tags/av_prop_duration.asp) to get the current percentage played. To draw the play button, we first call “**drawArc**” with color **#AAAAAA** and percentage of **1**. This draws a grey circle on the canvas as the background for the play button. We then add a second call to “**drawArc**”, this time with color returned by calling “**colorGradient**”. The percentage argument of the second “**drawArc**” call and the “**colorGradient**” call should be the video’s percentage played, which we calculated earlier.

5.3. Finally, we periodically update the canvas by adding a “**setInterval**” (https://www.w3schools.com/jsref/met_win_setinterval.asp) call in the “**init**” function. The “**setInterval**” should execute the function “**tick**” every **20ms**.

After this step, you will see the same play button as in Figure 1.

Part 2. Implement the quiz

Step 6. Call the “showQuiz” function when the video ends

Use “**showQuiz**” to handle the “**ended**” event of the video
(https://www.w3schools.com/tags/av_event_ended.asp).

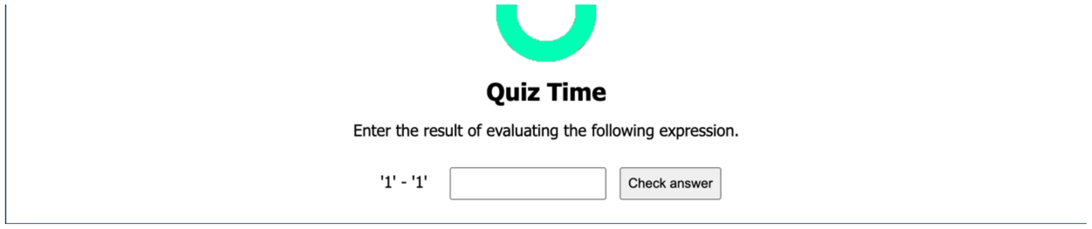
Step 7. Create the quiz

In this step, we insert the quiz underneath the canvas. At the end of the **<section>** element (which has id "contents"), we add the following elements using JavaScript in "**showQuiz**":

1. An **<h2>** heading with **id** "**quiz_heading**" and text "**Quiz Time**"
2. A **<p>** with **id** "**quiz_question**" and text: "**Enter the result of evaluating the following expression.**"
3. A **<div>** with **id** "**quiz_container**" (for setting the style) and child elements:
    a. A **<p>** with text "**\'1\' - \'1\'**".
    b. An **<input>** box of **type** "**text**" and **id** "**quiz_answer**".
    c. A **<button>** with text "**Check answer**" written on it, and event handler "**checkQuizAnswer**" registered with "**click**" event on the button (use "**addEventListener**")

Hint: use DOM APIs such as createElement, innerHTML, appendChild and setAttribute (https://www.w3schools.com/jsref/met_element_setattribute.asp).

After this step, when the video finishes playing, the quiz will show up:



If your page looks different, make sure that the **<p>, <input>** and **<button>** elements are placed properly **inside** the created **<div>** container, and the **<div>**'s id is set to "**quiz_container**". Also check if the quotation marks in the <p> tag are escaped properly.

Step 8. Remove the quiz container when answer matches

Implement function "**checkQuizAnswer**", which is called when the "Check answer" button is clicked. In the function, check if the value inside the <input> box matches the correct answer **"0"** (numerical zero, no quotation marks). If the answer does not match, then nothing will happen; otherwise, we remove the quiz container (Hint: call **removeChild** on **parentNode** of the quiz container). Then set the "**innerHTML**" of the created **<h2> in Step 7.1** to "**Quiz finished! Congratulations!**", and the **<p>** (underneath the <h2>) created in **Step 7.2** to "**Please proceed to study the next chapter.**"

Congratulations! Now you have finished Lab 3. You can test if the play button and the quiz are working correctly. The final result should look similar to the screenshots at the beginning of this document.

Submission:

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