

Lab 3

Web Form and Fetch

CSCI2720 Building Web Applications

Agenda

- Get and understand a sample
 - Bootstrap components with form
- Improve a web form
 - Basic input and textarea
 - Radio buttons
 - Button and event
- JavaScript fetch
 - Get a file
 - Save to a file
 - Save and load the comments

Sample

- Get this file and read the code

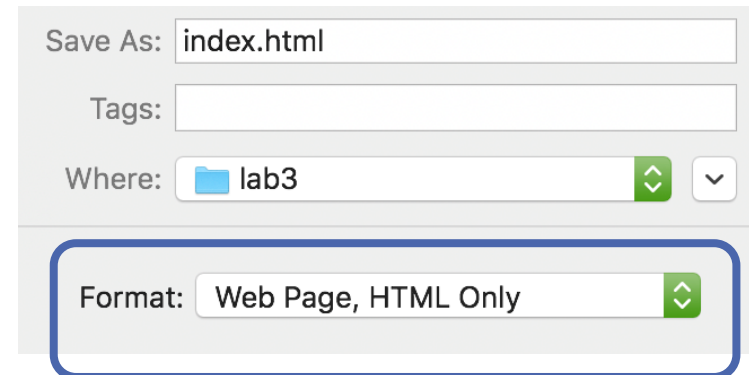
<http://www.cse.cuhk.edu.hk/~chuckjee/2720lab3/index.html>

- In Google Chrome, you can save the file by right clicking on the page and choose

“Save as...”,
keeping the
name

`index.html`

- *Read the
comments too!*



Save As: index.html

Tags:

Where: lab3

Format: Web Page, HTML Only

- Just a set of div's with two boxes in a flex layout
 - An SVG circle
 - A paragraph with a heading
 - And also the beginning of a form

Users > chuckjee > Desktop > lab3 > <> index.html > ...

4

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>CSCI2720 Comment System</title>
5  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css"
6  | <script src="script.js"></script>
7  </head>
8  <body>
9  <div class="container m-5">
10  | <div id="comments">
11  | | <div id="c1001" class="d-flex"> <!-- flex layout -->
12  | | | <div class="flex-shrink-0"> <!-- disallow div to shrink when page gets smaller -->
13  | | | | <svg height="100" width="100">
14  | | | | | <circle cx="50" cy="50" r="40" fill="green">
15  | | | | </svg>
16  | | | </div>
17  | | | <div class="flex-grow-1"> <!-- allow div to grow and take up all space -->
```

Read and understand the source code...
Relate it with the structure seen in DevTools



Web form

- You will work on this file with your favourite *text editor*, and view the results in *Google Chrome*
- You can find an HTML form `<form>` below the list of comment(s) with:
 - Email
 - Comment
 - Paragraph text using `<textarea>`
 - Color (to be used by SVG)
 - Red, green, yellow, blue
 - Or you can try other colors later



Web form

- The *Email* and *Comment* boxes came from “Example” from this page:
<https://getbootstrap.com/docs/5.0/forms/form-control/>
- Look at the classes **form-label** and **form-control**
- Observe the correspondence between **for** of **<label>** and **id** of **<input>**

Web form

- A color choice is available below the comment box
 - Only the radio button for “red” is built for you
 - Repeat div/input/label for more choices, with different **id** but same **name=new-color**
- *Feel free to style the buttons in your own way!*
 - See: <https://getbootstrap.com/docs/5.0/forms/checks-radios/#radios>

Web form event

- A button is built for submitting the comment

```
<button ...>Add comment</button>
```

- It is linked to an **onclick** event (as an attribute) for this button

```
onclick="processform()"
```

- Which means that, when this button is clicked on, the JS engine will run the **processform()** function
- This is an alternative way from setting the **onclick** property of the DOM element in JS

Setup JavaScript

- Create a new external JS file to be used, e.g. script.js

```
<script src="script.js"></script>
```

- In the JS file, create your event handler **processform()**

```
function processform() {  
    console.log("Testing");  
}
```

- Test the button, does it work?
- So, what should you do in the handler function body?

Event handler

a. Prepare comment element

1. Set up a new element

```
let newComment = document.createElement("div");
let element = '<div><svg height="100"
width="100"><circle cx="50" cy="50"
r="40"></svg></div><div><h5></h5><p></p></div>';
newComment.innerHTML = element;
```

2. Set the classes of the div and its children div's Note: instead of `class`, **`className`** is used

```
newComment.className = "d-flex";
newComment.querySelectorAll("div")[0].className
= "flex-shrink-0"; // 1st div
newComment.querySelectorAll("div")[1].className
= "flex-grow-1"; // 2nd div
```

3. Increment the comment id

```
let lastComment =
document.querySelector("#comments").lastElementC
hild; // instead of lastChild for div element
newComment.id = 'c' +
(Number(lastComment.id.substr(1)) + 1);
```

Event
handler

b. Apply
contents
from form

4. Change contents of **<h5>** and **<p>** according to form input with **id**

```
newComment.querySelector("h5").innerHTML =  
document.querySelector("#new-email").value;  
newComment.querySelector("p").innerHTML =  
document.querySelector("#new-  
comment").value;
```

Event
handler

c. Draw the
circle!

- You haven't learnt SVG in this course yet, but it's simple and easy!
 - More shapes:
https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Basic_Shapes
- 5. Get the color choice from the radio buttons

```
let color =
document.querySelectorAll("input[name=
new-color]:checked")[0].value; // look
for checked radio buttons
```
- 6. Change the fill color of the SVG circle

```
newComment.querySelector("circle").set
Attribute("fill", color);
```

Almost
there!

- Now the **newComment** element is ready

7. Append it to the div **#comment** using
`document.querySelector("#comments").appendChild(newComment);`
8. After this, *reset* the form to clear the contents
`document.querySelector("form").reset();`

Recapping the steps

- You need to finish these
 1. Read and understand the given file
 2. Complete the web form
 3. Point to event handler
 4. Create JS event handler
 - Prepare a new element
 - Get contents from user form
 - Set up the circle SVG
 - Append to the HTML content
 - Reset web form



chuckjee@cse.cuhk.edu.hk

This lab is so fun! Let's explore further with Bootstrap and JS fetch.



good@student.me

I can't agree more



lazy@guy.it

Too many steps... but I still finished!

■ Now you can try your
comment form!

Add your comment:

Email address

name@example.com

Comment

■ Everything is generated by
JavaScript

■ Yet, all new comments are lost
after refresh

- ☐ Red
☐ Green
☐ Yellow
☐ Blue

Add comment

JS Fetch

- Before we add save and load features, let's try to fetch contents from server
- Get this file using your browser
<http://www.cse.cuhk.edu.hk/~chuckjee/2720lab3/file.txt>
- Place it in the same folder as your **index.html** file from Lab 3

Fetching a file

- In the HTML file, create a new button for ***Load file***
 - It can be next to the ***Add comment*** button
 - Create an onclick event for it, *e.g.*,
... `onclick="loadfile()"` ...
 - And set up the event handler

```
function loadfile() {  
    alert("testing");  
};
```
- Check that your event fires successfully

Fetching a file

- If the event works well, it's time to start the loading work
- Using JS fetch, obtain the contents of file.txt given to you earlier, *e.g.*,

```
fetch('file.txt')  
  .then(res => res.text())  
  .then(txt => console.log(txt))
```

Working on a web server

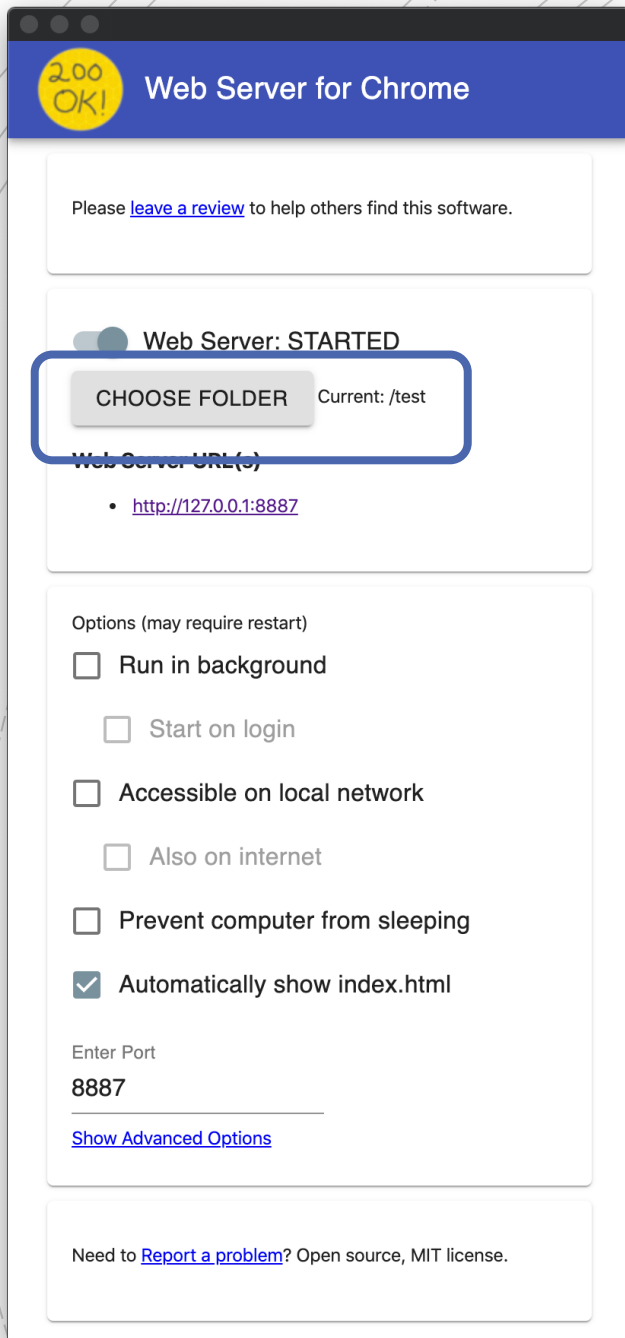
- No matter how hard you try, you won't succeed:

```
✖ ▶ Fetch API cannot load file:///User script.js:33  
s/chuckjee/Desktop/lab3/file.txt. URL scheme must  
be "http" or "https" for CORS request.
```

- Chrome's security settings doesn't allow using fetch() on local (file://...) filesystem
- You have to upload your page (html, js, txt, ...) onto a web server to try
- For local development, you may start a local web server within your Google Chrome!
 - <https://chrome.google.com/webstore/detail/web-server-for-chrome/ofhbbkphhbklhfoeikjpcbhemplcgigb/>

A local web server

- A web server is a software which serves documents from a specific folder, when requested
 - E.g. <http://www.cse.cuhk.edu.hk> calls the CSE web server to serve some files at, for example, */var/www* of the server storage
- Our local web server allows you to serve a folder, and then you can access documents via the browser
 - Whether others can access or not depends on firewall settings



- Set up this new web server
 - Basically you just need to choose a folder and start it!
 - Put your lab 3 exercise there
- Visit your lab 3 exercises at <http://127.0.0.1:8887>
 - Include folder names if necessary
 - “Port” 8887 is a point of entry
- *If you try Googling, you will find other ways to **allow file access from local files**, but there are security concerns so a local web server may serve better!*

Writing to the server

- A web server not only allows users to obtain documents
- Usually, users can also upload data
 - A specific script is needed to handle the data, e.g. where to save it, what to do with it
- If you are successful with the previous step, you can carry on trying to save text into a file

Options (may require restart)

☐ Run in background

☐ Start on login

☐ Accessible on local network

☐ Also on internet

☐ Prevent computer from sleeping

☒ Automatically show index.html

Enter Port

8887

[Hide Advanced Options](#)

☐ Set CORS headers

☐ Listen on IPV6

☐ Plain (static) files view

☒ Allow File upload

☒ Verbose logging

The files directory listing allows drag-and-drop to upload small files

☐ Enable mod-rewrite (for SPA)

- Uploading data can be done using `fetch()`, if the **web server supports POST or PUT** methods
- Let's enable **PUT** in Web Server for Chrome for uploading
 - PUT is easier as no script is needed
 - 1. Options >> *Show Advanced Options*
 - 2. Enable *Allow File upload*
- The fetch operation looks like this:


```
fetch('file2.txt', {
  method: 'PUT',
  body: "Hello world"
}); // a file called file2.txt
with content: Hello world
```

Save and
load the
comments

- If you are able to write a testing file, congratulations!
- Now it's time for some *real work*
- Think of a way to ***save all the comments into a file*** for loading later
 - Now this can be done manually using the load and save buttons



Save and
load the
comments

- There are too many possibilities
 - XML? JSON? HTML? Plain text?
- For this lab, start with the simplest:
 - Get the HTML contents under the element **div#comments** into a string
 - **Save file**: Save this string into a file for storing on the web server
 - **Load file**: Load this string into **div#comment** to *replace* its HTML content

- ☐ Red
- ☐ Green
- ☐ Yellow
- ☐ Blue

Add comment

Load file

Save file



Submission

- No submission is needed for labs
- What you have done for the comment system will be useful for your assignment
- **Please keep your own file safely**