香港中文大學
The Chinese University of Hong Kong

CSCI2720 2023-24 Term 1: Building Web Applications

Lab 7: Node.js and Express

Dr Colin Tsang

Outline

- Getting started with Node.js
  - Local installation
  - Online Node.js playgrounds
- Hello World from Express
- Parsing URL parameters
- Obtaining POST parameters

Using Node.js

- For a web server to be able to serve contents, it must listen on a *TCP port*
  - e.g., port 3000
  - Administrative rights is needed to access it
- Pick one of these ways (or try both if you wish)
  - Install Node.js onto your laptop
  - Use online Node.js playgrounds

Getting Node.js

- *Local installation*:
  - Access http://nodejs.org and follow the link to download the latest version for the OS of your laptop
    - For the sake of *Assignment Two*, you may want to download the *18.17.0 version* instead:
      - Windows: https://nodejs.org/dist/v18.17.0/node-v18.17.0-x64.msi
      - MacOS: https://nodejs.org/dist/v18.17.0/node-v18.17.0.pkg
  - After installation, start **Command Prompt** (Windows) or **Terminal** (MacOS / Linux) and issue this command:
    - `node -v`
    - It shows the version number of your Node.js

- *Online playgrounds*:
  - http://stackblitz.com
  - You can sign in with your *GitHub* account to use the service (not necessary for this lab)
  - You must use *Google Chrome* for StackBlitz.
  - StackBlitz cannot perform Task 2 of this lab properly.

Setting up the first web app

- *Local*:
  - Create a new directory somewhere
    - E.g., Desktop/lab7
  - Navigate to this directory
    - E.g., `cd Desktop/lab7`
  - Type this command : `npm init`
    - Accept default answers for all questions with `Enter`
  - Install Express: `npm install express`

- *StackBlitz*:
  - Click New Project on the right-top corner.
  - Choose Node.js (blank project)
  - Install Express: `npm install express`

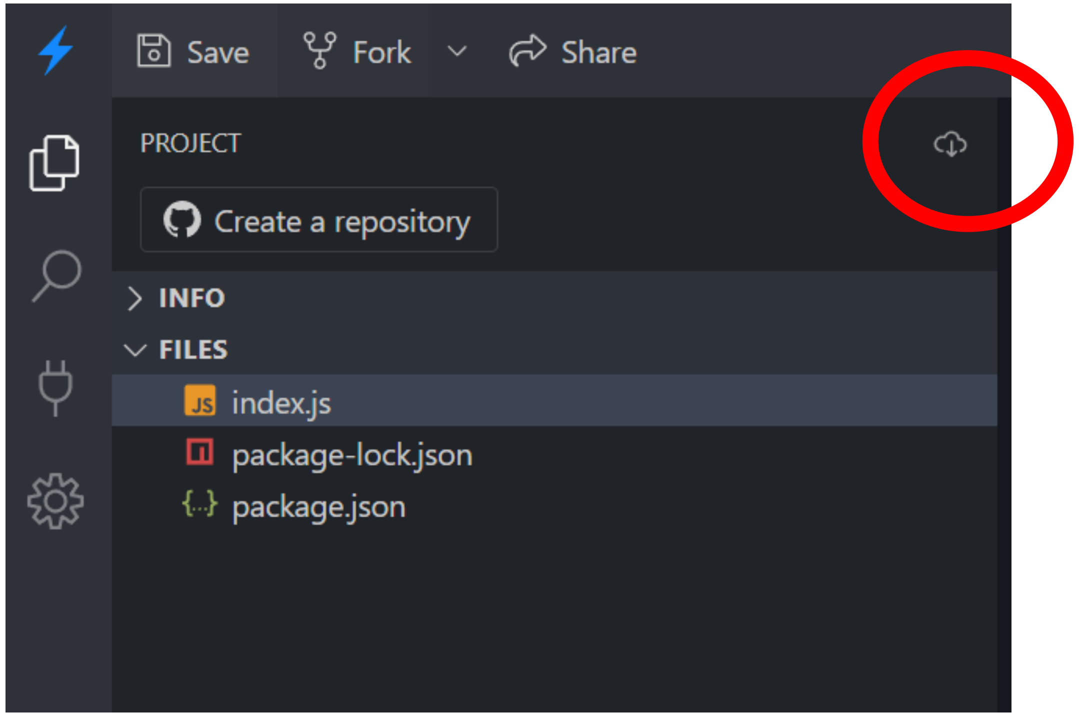
Hello World from Express

- *Local*:
  - Set up a new file in this directory, e.g., **`server.js`**, with the below contents
  - Start the server in the directory by **`node server.js`**
  - Check this out in browser: http://localhost:3000
- *StackBlitz*:
  - Put down the below contents into **`index.js`**
  - Run the program by **`node index.js`**
  - The result will be displayed on the right side
    - You can also copy the URL and open it in a new tab

```
const express = require('express');

const app = express();

// handle ALL requests

app.all('/*', (req, res) => {
    // send this to client
    res.send('Hello World!');

});

// listen to port 3000

const server = app.listen(3000);
```

Download from StackBlitz

- You can download the project from StackBlitz:

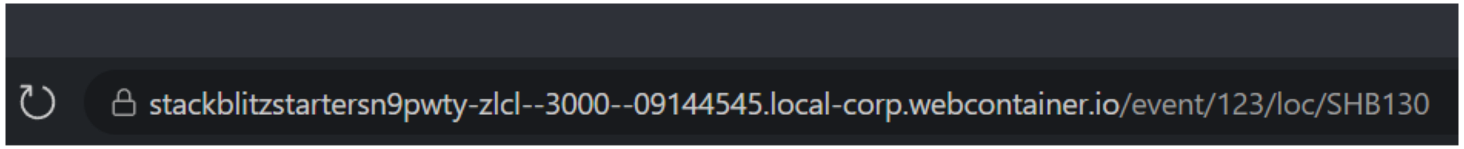


TASK 1: Parsing URL parameters

- You can read parameters from URL segments using the **:** operator
- You can try to parse the URL with the following code

```
app.get('/event/:eventID/loc/:locID', (req, res) => {
  res.send(req.params);
});
```

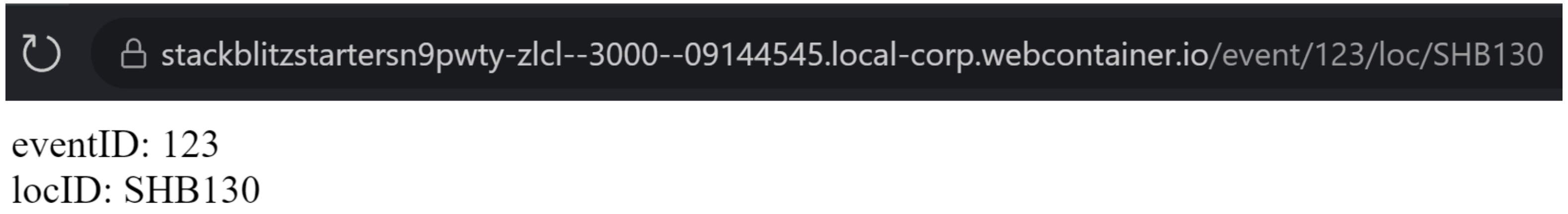
- The result should look like this:


stackblitzstartersn9pwty-zlcl--3000--09144545.local-corp.webcontainer.io/event/123/loc/SHB130

```
{"eventID":"123","locID":"SHB130"}
```

TASK 1: Parsing URL parameters

- TASK 1: Adjust the **res.send()** contents into this format:
  - You may need a combination of **<br>**, **req.params**, etc.



```
stackblitzstartersn9pwty-zlcl--3000--09144545.local-corp.webcontainer.io/event/123/loc/SHB130
```

```
eventID: 123
locID: SHB130
```

- Can you do it? The solution code will be uploaded to Blackboard later.

TASK 2: Obtaining POST parameters

- GET is usually used for the server to deliver contents
- POST is usually for putting up contents to the server
  - Advantages: contents are put inside the request body.
- Using the same URL before with **eventId** and **locID**, set up a POST rule to accept **loginID** from user.

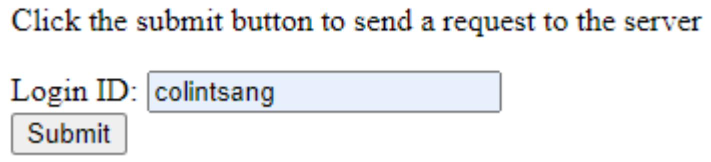
TASK 2: Obtaining POST parameters

- In this task, you are going to send a request to the server with POST.
  - Download the *lab7_task2.html* from Blackboard.
    - Read and try to understand the HTML code!
  - Set up a local server to open *lab7_task2.html* (use *Simple Web Server*).
  - Input a login ID and then click the submit button
  - A request will be sent to the server. In the HTML code, change the server address to:
    - Local: http://localhost:3000/event/.../loc/...
    - Stackblitz: http://......webcontainer.io/event/.../loc/...
  - If it receives a response with **Content-Type = text/html**, the HTML code from the server will be rendered.

TASK 2: Obtaining POST parameters

- Your task is to write the code on the *server-side* to generate the HTML code as a response to the POST request.

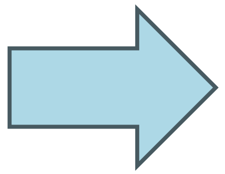
- From the *client-side*, you should see the following after clicking the Submit button:

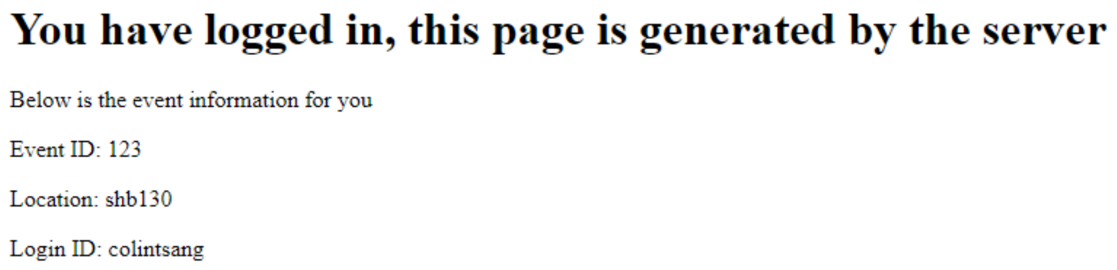


Click the submit button to send a request to the server

Login ID: colintsang

Submit





You have logged in, this page is generated by the server

Below is the event information for you

Event ID: 123

Location: shb130

Login ID: colintsang

- Can you finish Task 2? The solution code will be uploaded to Blackboard later

TASK 2: Obtaining POST parameters

- Hints for Task 2:

```
app.post('/event/:eventID/loc/:locID', (req, res) => {
  // step 1: get the parameters from params and body
  // step 2: write down the html code
  // step 3: res.setHeader() to set the header to html
  // step 4: res.send() to send the html
});
```

TASK 2: Obtaining POST parameters

- *Local*:
  - There could be a CORS error
    - It can be resolved in your local Node.js
  - The data is in JSON from the POST parameters
  - Do the following:
    - Run **`npm install cors`**
      - Restart the server after the installation
    - Add these codes on top of your **`server.js`**

```
const cors = require('cors');
app.use(cors());
app.use(express.json());
```

- *StackBlitz*:
  - It seems like StackBlitz cannot fix the CORS error……
  - Use a locally installed Node.js to finish Task 2.
  - Your *project* should be done in a locally installed Node.js too.
  - See the *appendix* for a trick to perform Task 2 on StackBlitz.

Submission

- No submission is needed for labs
- But what you have done will be useful for your assignment and project
- Please keep your own code safely

Appendix: CORS error in StackBlitz

- I couldn't find a good way to solve the CORS problem on StackBlitz. It's blocked by StackBlitz.
- A get-around is to generate the *lab7_task2.html* on StackBlitz too (i.e., on the *server-side*).
- Do the same things as the local solution, i.e., run **`npm install cors`**, etc.
- Use the following code:

```
app.get('/example', (req, res) => {
  const htmlCode = `The code copied from lab07_task2.html`; // copy and paste the code
  res.send(htmlCode);
});
```

- Then, you can "open" the *lab7_task2.html* on StackBlitz (by http://......webcontainer.io/example), instead of using *Simple Web Server* to open it as a client.