**CS628 Full-Stack Development II – Backend**

**PE04 - Basic Node and Express - Serve Static Assets**

Developed by Clark Ngo on December 29, 2020

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Before You Start**

* Some steps are not explained in the tutorial**.** If you are not sure what to do:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

Students will be able to:

* Learn Basics of Node and Express

**Resources**

* https://www.freecodecamp.org/learn/apis-and-microservices/basic-node-and-express/serve-static-assets

**How to Submit**

* **Upload .zip file**
* **Write a 150-word summary to explain your understandings and findings from this lab assignment.**

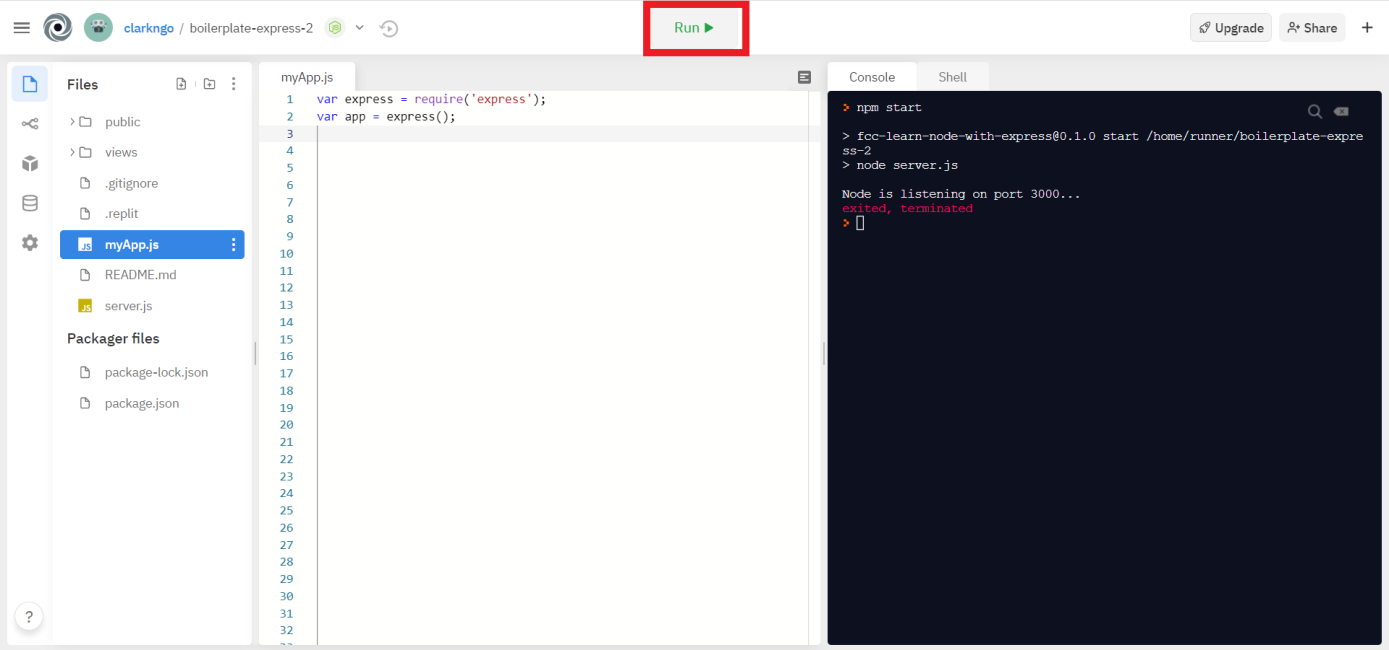
## **Basic Node and Express - Serve Static Assets**

An HTML server usually has one or more directories that are accessible by the user. You can place there the static assets needed by your application (stylesheets, scripts, images). In Express, you can put in place this functionality using the middleware express.static(path), where the path parameter is the absolute path of the folder containing the assets. If you don’t know what middleware is... don’t worry, we will discuss in detail later. Basically, middleware are functions that intercept route handlers, adding some kind of information. A middleware needs to be mounted using the method app.use(path, middlewareFunction). The first path argument is optional. If you don’t pass it, the middleware will be executed for all requests.

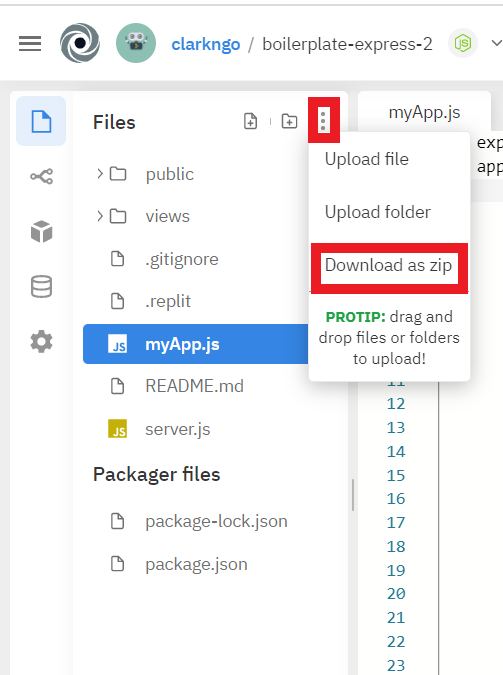
Mount the express.static() middleware for all requests with app.use(). The absolute path to the assets folder is \_\_dirname + /public.

Now your app should be able to serve a CSS stylesheet. From outside, the public folder will appear mounted to the root directory. Your front-page should look a little better now!

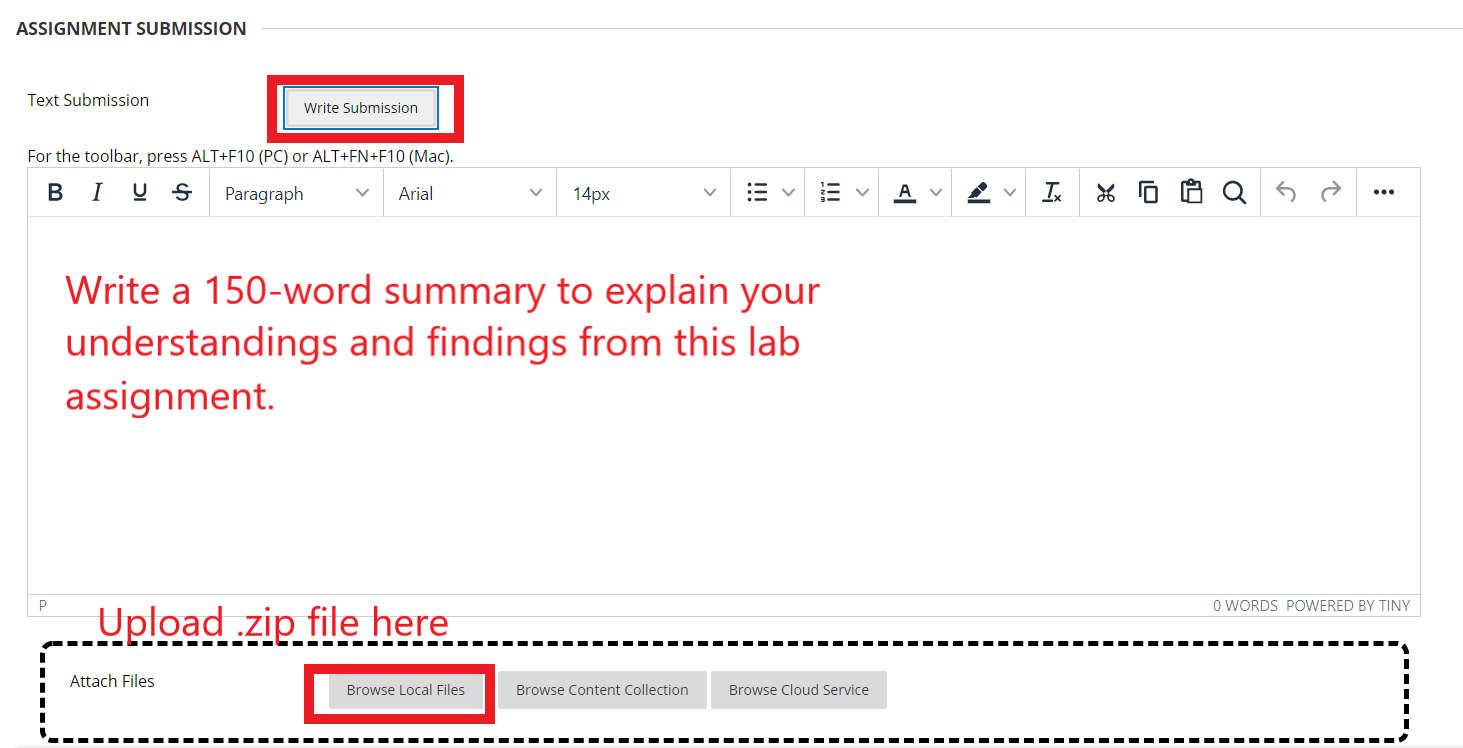
1. Start a new project on Repl.it using [this link](https://repl.it/github/freeCodeCamp/boilerplate-express).
2. Click Run to test your code



1. Download the project as .zip.



1. In BlackBoard, upload the .zip file and write a 150-word summary to explain your understandings and findings from this lab assignment.



When we look at middlewares in node.js, we can see that we can either set a target for the middleware we are creating, or we can let it run for all requests.

There are two types of web programming when it comes to it’s staticness. Static and dynamic. Static webpages are simple html css javascript (front end) webpages that don’t have any parts that speak to the server such as an html form. We can think of a simple html webpage with text and images. Dynamic webpages can contain elements such as html forms that send data to the server, the server does the necessary computations and sends back a response.

Looks like express doesn’t support static pages by default and we need to enable it. Our static assests are in the directory “/public”. So we take the root directory, concatenate it with “/public”. And then pass it to express.static().

Here is what this does. When we have images in our views, these images will be looked for in the public folder. So when we menion a bug.jpg in a view, it will try to statically serve the bug.jpg from the public folder.

I downloaded a cartoon bug image, uploaded to replit, added the html code to my index file, added the middleware to the js file, and I ended up seeing the image on the webpage. When I remove the middleware, the image can not be seen.

<img src="/bug.jpg" alt="Bug Image">

app.use(express.static(\_\_dirname + "/public"));

Middleware reminds me of the filter function of dynamic programming with java servlets. Like the challenge says “intercept route handlers”.

In the code above we define a servlet and a filter that has the servlet as its target.

Lets say we have an AdminServlet which is an admin menu. We login with a username and password from index.html. This form sends the data to AdminServlet with action=”AdminServlet”. But before we let the user to the admin menu page, we need to check if he is really admin or not. We can use a filter here. That filter will intercept every request and response coming in and out of its target. We set filter’s target in web.xml in filter’s url-pattern line.

<servlet>

<servlet-name>AdminServlet</servlet-name>

<servlet-class>AdminServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>AdminServlet</servlet-name>

<url-pattern>/AdminServlet</url-pattern>

</servlet-mapping>

<filter>

<filter-name>MyFilter</filter-name>

<filter-class>MyFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>MyFilter</filter-name>

<url-pattern>/AdminServlet</url-pattern>

</filter-mapping>