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

**PE05 - Basic Node and Express - Serve JSON on a Specific Route**

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-json-on-a-specific-route

**How to Submit**

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

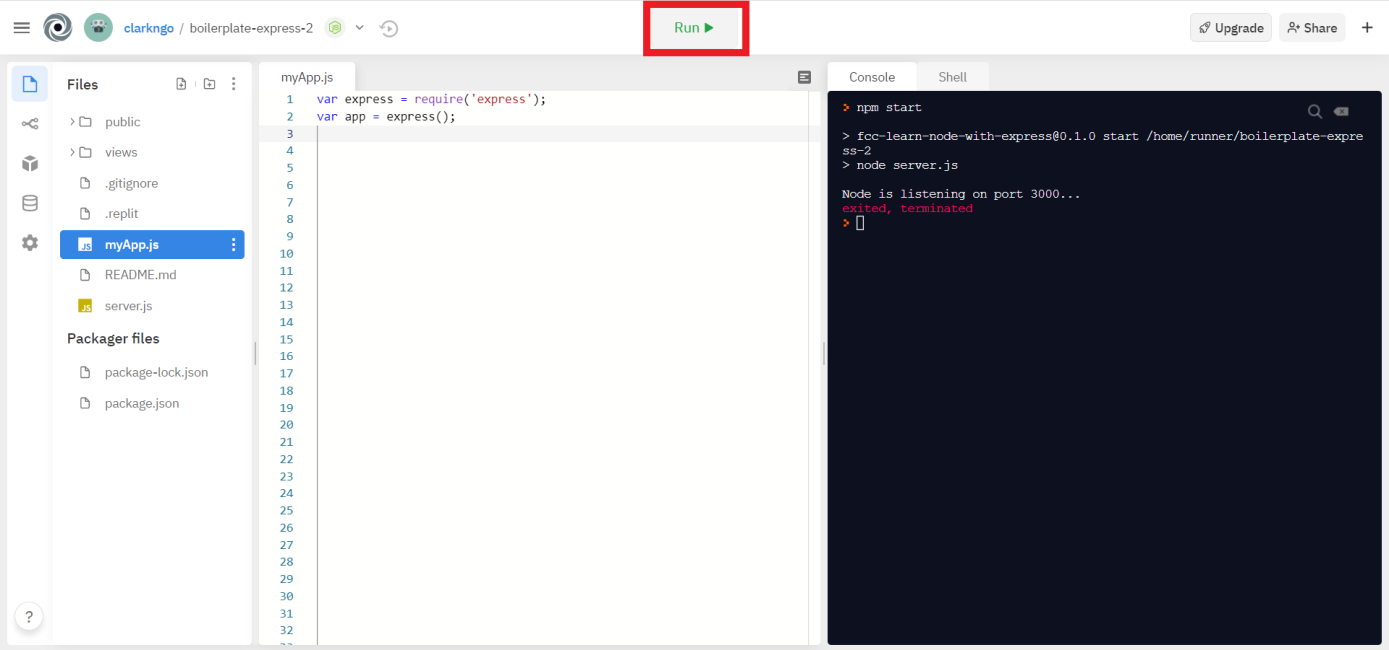
## **Basic Node and Express - Serve JSON on a Specific Route**

While an HTML server serves (you guessed it!) HTML, an API serves data. A *REST* (REpresentational State Transfer) API allows data exchange in a simple way, without the need for clients to know any detail about the server. The client only needs to know where the resource is (the URL), and the action it wants to perform on it (the verb). The GET verb is used when you are fetching some information, without modifying anything. These days, the preferred data format for moving information around the web is JSON. Simply put, JSON is a convenient way to represent a JavaScript object as a string, so it can be easily transmitted.

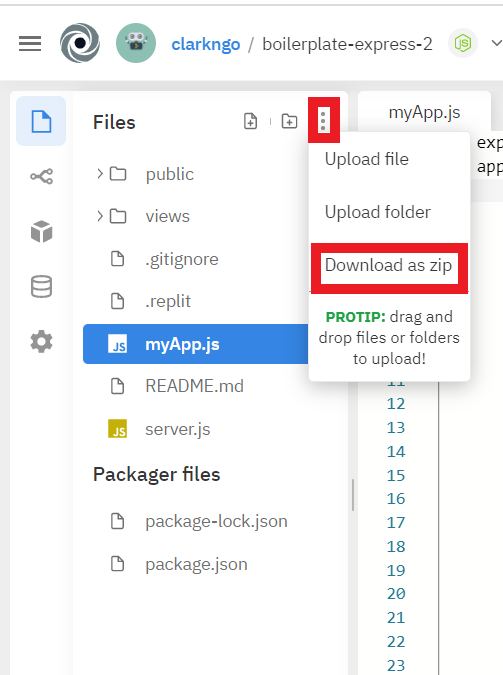
Let's create a simple API by creating a route that responds with JSON at the path /json. You can do it as usual, with the app.get() method. Inside the route handler, use the method res.json(), passing in an object as an argument. This method closes the request-response loop, returning the data. Behind the scenes, it converts a valid JavaScript object into a string, then sets the appropriate headers to tell your browser that you are serving JSON, and sends the data back. A valid object has the usual structure {key: data}. data can be a number, a string, a nested object or an array. data can also be a variable or the result of a function call, in which case it will be evaluated before being converted into a string.

Serve the object {"message": "Hello json"} as a response, in JSON format, to GET requests to the /json route. Then point your browser to your-app-url/json, you should see the message on the screen

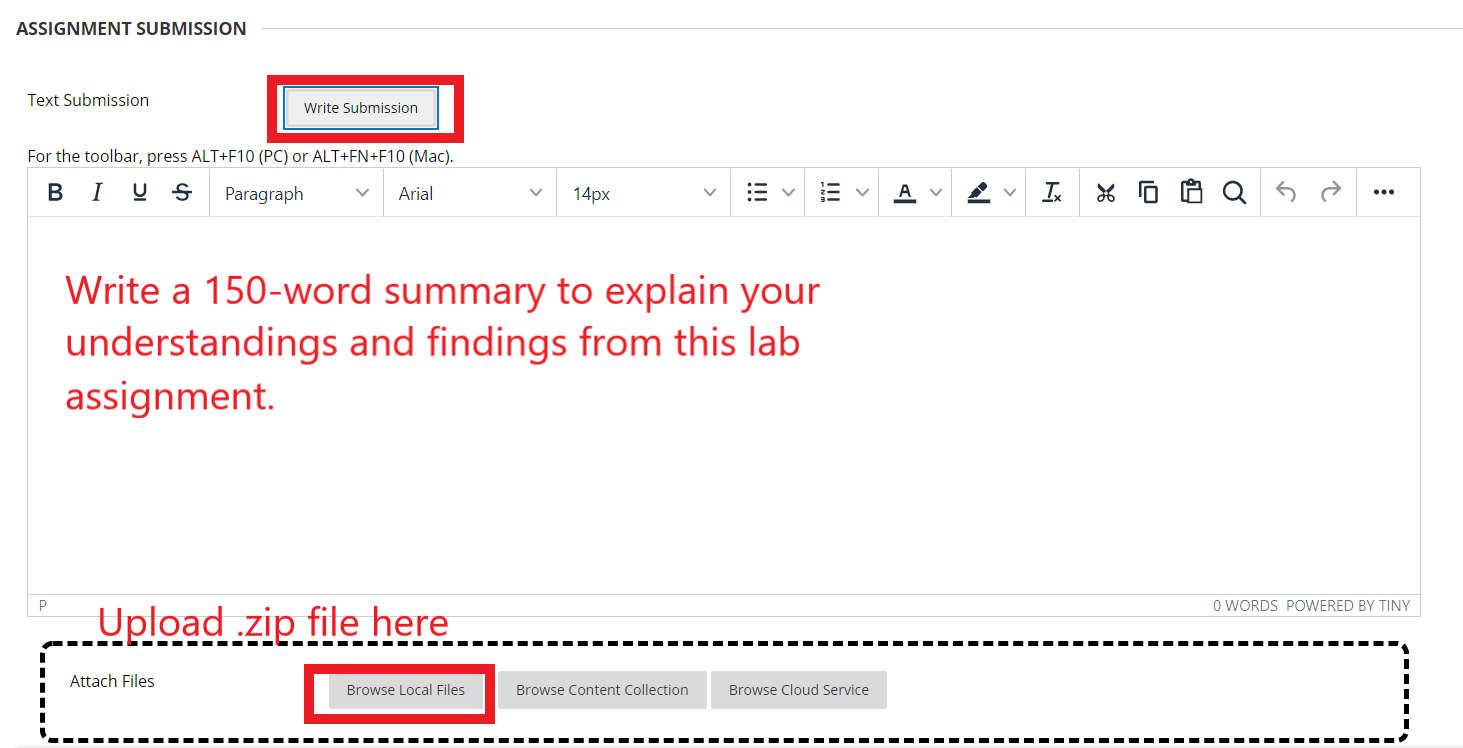
1. Start a new project on Repl.it using [this link](https://repl.it/github/freeCodeCamp/boilerplate-express).
2. Click Run to test your code



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.



There are two different types of communication we will be using on the internet. In earlier modules we have been working on the first type. Sending and receiving full web pages. The second type is sending and receiving data without a webpage. There are different formats for this data. And these formats allow different types of servers that use different programming languages to speak to each other.

We can think of this as yet another service that the server offers to the outside world. Just like dynamic web programming, the server side is hidden from clients. The clients only see what they are served.

We need a url to correspond to a controller. This controller will be serving data, not a webpage. The client can send a get request for example and we can add a json data to the response to the get request.

app.get("/json", (req, res) => {

res.json({

message: "Hello json"

});

});

I recently studied the serialization and deserialization feature of java and it is the same logic here. We turn the values of instance variables of the object into a text file. We can turn this file into an object later. Static variables don’t need to be serialized.

We can navigate to the /json directory of the website, which send a get request at that url to the server. This will make the server return the json object we specified. And we can see it on the browser as the client.

