# **Intro Express Practice**

*Introduction*

It's time to get some practice setting up an express server and writing a few routes. Feel free to test these routes with the browser's navbar.

### **Up and Running**

* Create a new directory
* run npm init -y
* install express and nodemon with npm i express nodemon

Add a start script to package.json:  
 "scripts": {

"start": "nodemon index.js"

* }
* create an index.js file

#### **Set up an Express server**

Let's write the boilerplate for an express server:

* First, import express at the top of the file with:

const express = require('express');

* Next, import fruits from fruits.js.

const fruits = require('./fruits');

* Go ahead and define a PORT constant variable, e.g., const PORT = process.env.PORT || 3000;
* Init an app instance const app = express();
* And finally bind the app to a port at the bottom of the file  
   app.listen(PORT, () => console.log(`up on port ${PORT}`))

Now start the server from the terminal with npm start. If all goes well, you should see the message from the console log above in the console. *Don't stop the server until you are done writing code or unless you just want to restart it.* If you save the index.js file the server should automatically re-load the changes.

#### **Adding the first Route**

Start off by defining a simple ping-pong route:

app.get('/ping', (req, res) => {

res.json('pong');

})

Try to test this route out.

#### **/greet/:name**

GET /greet/:name should return a greeting with the supplied name, e.g., 'Why hello there, !'

Recall you can access the url parameters with req.params

#### **/five**

GET /five should return an array of the numbers from 1-5

#### **evens**

GET /evens/:n should return an array of even numbers from 2 up to and including n. So when you call this route, n in the url should be a number.

Also, you may need to use parseInt to cast the supplied n parameter to a number

#### **/namelength/:name**

GET /namelength/:name should return the length of the supplied name

#### **/fruits**

GET /fruits should return an array with all the fruits.

Let's add a route that retrieves all the fruits from our fruits.js file. Recall we imported it on top of our index.js.

app.get('/fruits', (req, res) => {

//your code here

res.send()

})

#### **/fruits/:name**

Now let's add a route that takes a route parameter name and retrieves the fruit that matches the supplied name.

app.get('/fruits/:name', (req, res) => {

//your code here

// HINT - you can use a higher-order array method

})

Recall you can access the url parameters with req.params. So when you call this route, name in the url should be a capitalized string(refer to fruits.js, if you are unsure).

### **Bonus**

GET /fruits/sort should return the fruits array sorted alphabetically using bubble sort.

app.get('/fruits/sort', (req, res) => {

// implement bubble sort

res.send()

})