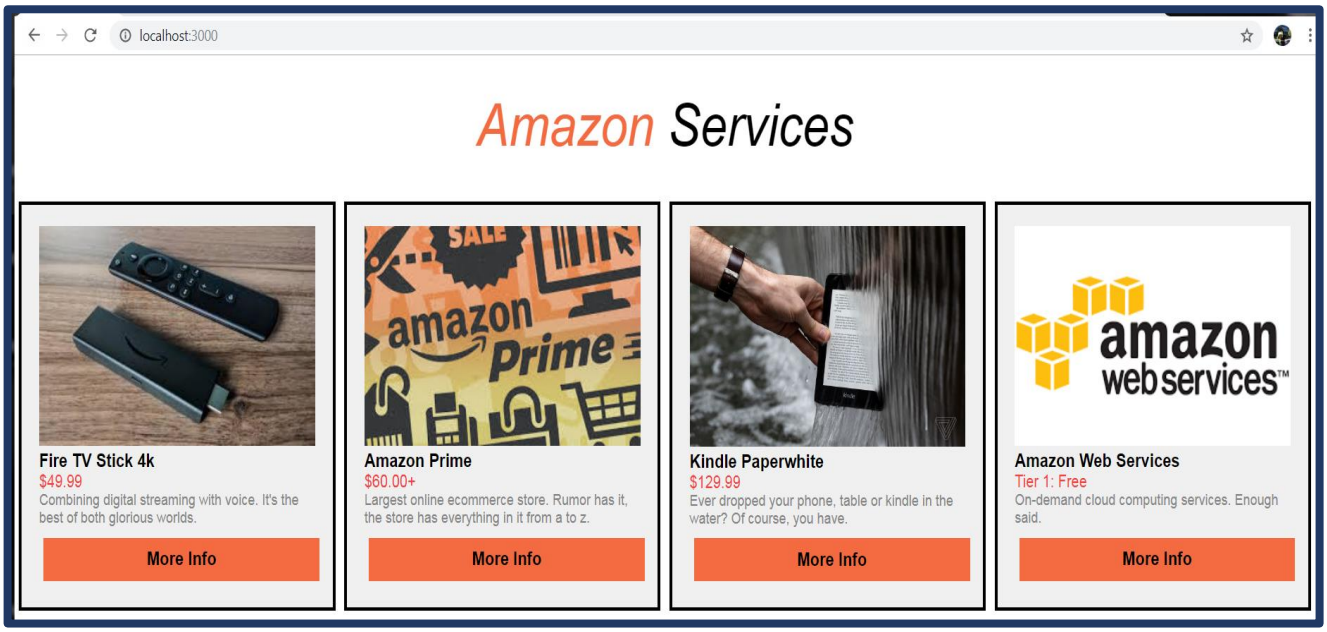
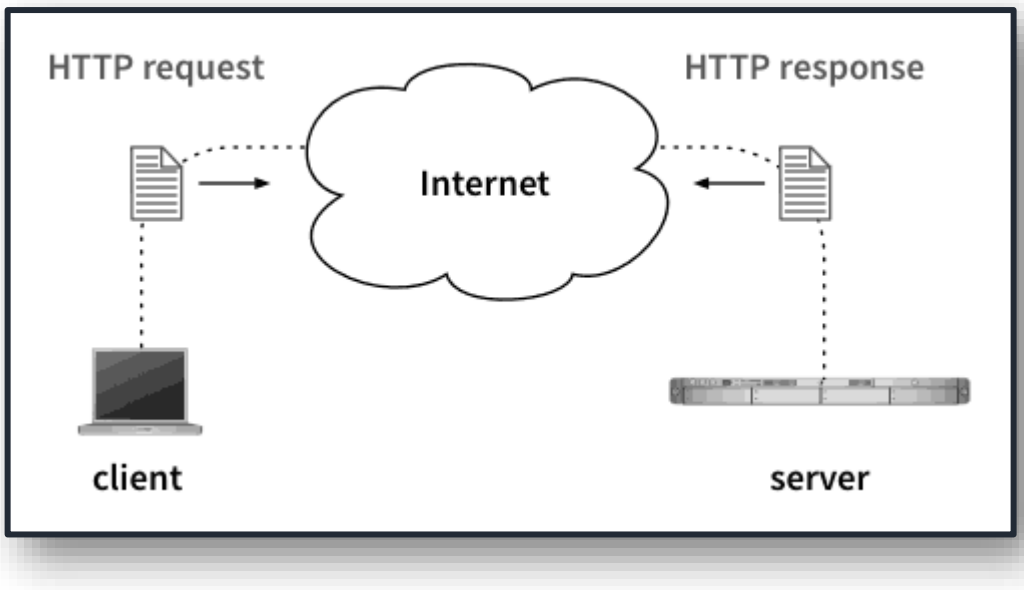
Introduction to Creating Basic Routes

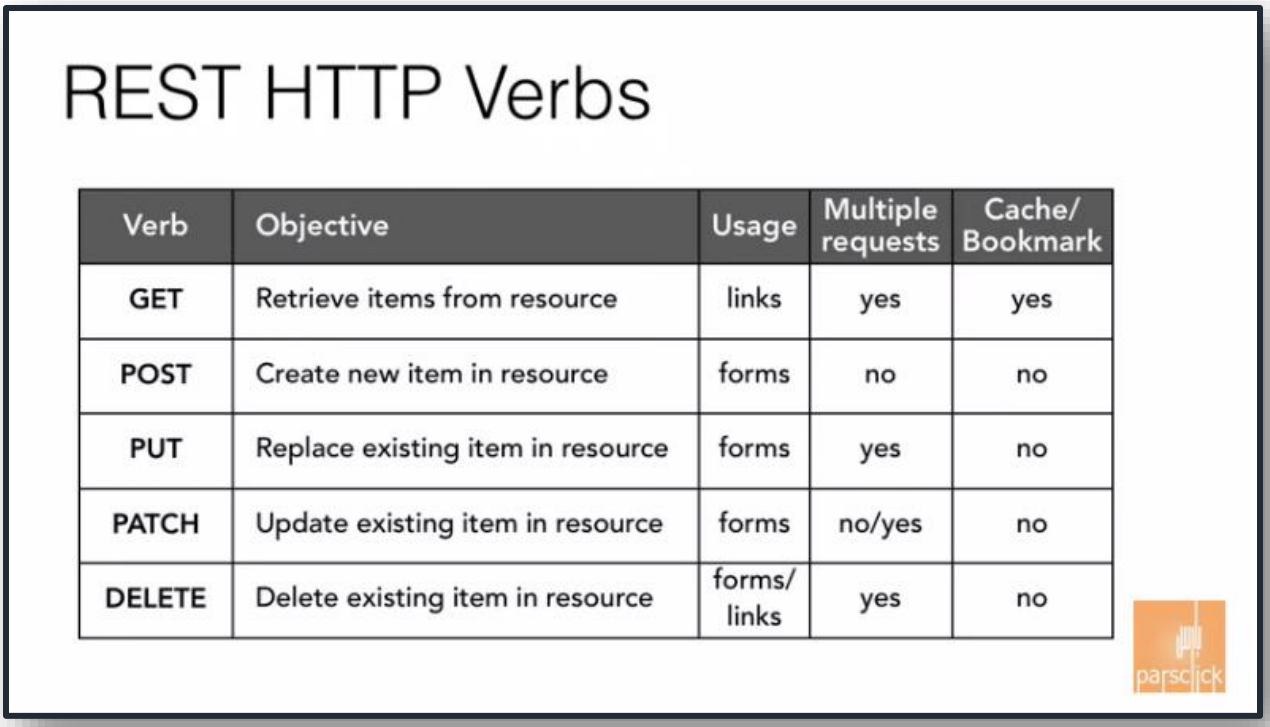


Creating Basic Routes in Node and Express

Routing refers to how an application's endpoints (URL or path) responds to a client's request. For example, when you type in a Uniform Resource Locator or URL, you, the client, send a request to a server. The server listens for your request, and then serves up your request by displaying the website, images, and or data to you.



In Node, routes are defined by using methods of the Express app object that correspond to an HTTP method. Meaning, routes are defined in node using verbs. These verbs indicate the actions to be performed for a given resource. Commonly used app methods are GET, POST, PATCH and DELETE.



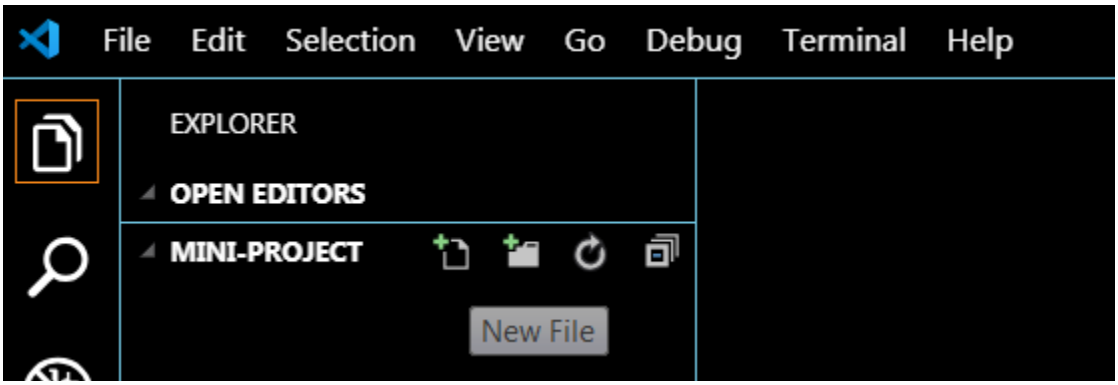
In this session, you will learn how to create basic routes. At the end of the course, you will have a fully functioning web application that runs on your local machine.

Before starting the course, please download a text editor of your choice, preferably Visual Studio Code. Also, download node.js.

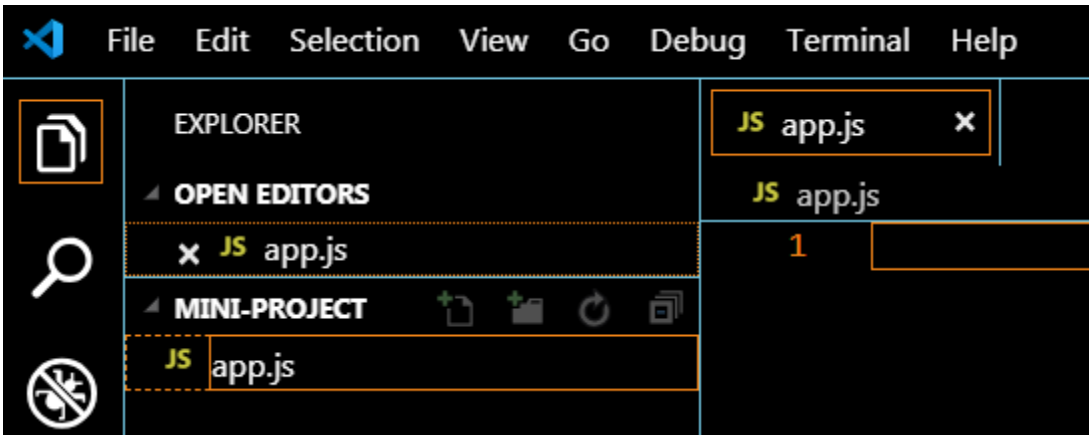
Let's dive in.

Creating Project File Structure

1. Create an empty project folder inside Visual Studio code called **Mini-Project**.
2. From the File Explorer toolbar, press the **New File** button.



3. Name the newly created file, app.js.



4. Create **three** directories: **Routes**, **Views** and **Public**.
5. Within each directory, create your files. In Routes, create two files named, about.js and home.js. In Views, create three files named, 404.html, about.html, and home.html. And in Public, add all project images. Then create two folders named, about.css and home.css.

Creating the Main file

1. Navigate to the app.js file and paste the following code.

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


const homeRoutes = require("./routes/home");
const aboutRoutes = require ("./routes/about");

const app = express();

app.use(express.static("public"));
```

```
app.use('/', homeRoutes);
app.use('/about', aboutRoutes);

app.use((req, res, next) => {
    res.status(404).sendFile(path.join(__dirname, "views", "404.html"));

});

app.listen(3000);
```

2. Save file.

Code Explanation: App.Js is our main file. It's where we require core modules that come with express, link to our home and about routes, serve up our public files, define URL paths or endpoints, look in the views folder to serve up the 404.html file, and start the server on **localhost:3000** to access the app.

Installing Express

Express is a framework for Node.Js that creates rules for how our folders/code is structured in our web application. This structure provides consistency in how our server is set up, how we route URLs to responses, and how we render views for html responses. To be able to access our app and start the server on **localhost:3000**,

1. Run the following command in your terminal in the root directory: **npm init**. The npm init command creates a configuration file, called package.json for your application. This command prompts you for several things, such as name and version of your application. For now, simply hit **ENTER** to accept the defaults for most of them, with the following exception: entry point: (index.js). Enter **app.js** because we want app.js as our main file, from which to start the server.
   Note: By default, npm installs the module to the dependencies list in the **package.json** file. If you open the file, you will see express listed as a dependency.

2. Now install Express in the mini-project directory. Run the following command in your terminal in the root directory: **npm install express - - save**. Your package.json, or configuration file, will look like the following.

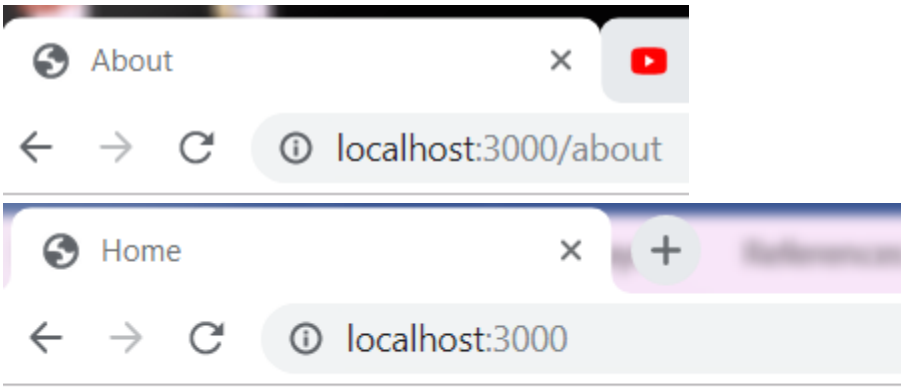
```
{
  "name": "mini-project",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1",
    "nodemon": "^1.19.1"
  }
}
```

Viewing & Accessing our app

1. In your terminal in the mini-app directory, type **node app.js** to start the server. The node app.js command, starts the project and allows you to view the project in your browser.

```
C:\Users\Talya\Desktop\mini-project>node app.js
```

2. In your browser, type **localhost:3000**. Remember, in the app.js file we instructed the app to listen on port 3000 and then send a response. For now, the expected response are two endpoints one at **localhost:3000 (homeRoutes)** and **/about (aboutRoutes)**.



Creating Routes

1. Navigate to the **Routes** folder.
2. In the **home.js** file, paste the following code:

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

const express = require('express');

const router = express.Router();

router.get('/', (req, res, next) => {
    res.sendFile(path.join(__dirname, '../', 'views', 'home.html'));

});


module.exports = router;
```

Code explanation: Home.js contains the path to the home or landing page, in this case, it is localhost:3000. Because our code is considered a mini app, express.router is used instead of app.get ("/"). This method makes our code more manageable by breaking the code into separate files and linking to the main app. Router.get is an HTTP verb that allows us to retrieve data when the end point is accessed. In the code, the res.sendFile method will go down two directories and look for the views directory and render home.html to the user on the homepage. Lastly, module.exports = router is used to mount the router module on a path to the main app.

3. In the about.js, paste the following code:

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

const express = require ('express');

const router = express.Router();

router.get('/', (req, res, next) => {
   res.sendFile(path.join(__dirname, '../', 'views', 'about.html'));

});



module.exports = router;
```

Code explanation: About.js contains the path to the /about page, in this case, it is localhost:3000/about. Because our code is considered a mini app, express.router is used instead of app.get ("/"). This method makes our code more manageable by breaking the code into separate files and linking to the main app. Router.get is an HTTP verb that allows us to retrieve data when the end point is accessed. In the code, the res.sendFile method will go down two directories and look for the views directory and render about.html to the user on the /about page. Lastly, module.exports = router is used to mount the router module on a path to the main app.

Why can't we see anything on the page? We haven't added our html! If we tried to run our code, the app crashes. Let's create our views and then our app will start taking shape.

Creating Views

1. Navigate to the **Views** folder.
2. In the **home.html** file, paste the following code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Home</title>
    <link rel="stylesheet" type="text/css" href="css/home.css">
  </head>
  <div class ="header1"><span>Amazon</span> Services</div>
  <body>
```

```
    <div id="p-flex">
      <div class="p-flex"><div class="p-flex-in">
        <img class="p-img" src="http://localhost:3000/firestick.jpg"/>
        <div class="p-name">Fire TV Stick 4k</div>
        <div class="p-price">$49.99</div>
        <div class="p-desc">Combining digital streaming with voice. It's the best
of both glorious worlds.</div>
        <button class="p-add"><a href="../about">More Info</a></button>
      </div></div>
      <div class="p-flex"><div class="p-flex-in">
        <img class="p-img" src="http://localhost:3000/prime.jpg"/>
        <div class="p-name">Amazon Prime</div>
        <div class="p-price">$60.00+</div>
        <div class="p-desc">Largest online ecommerce store. Rumor has it, the
store has everything in it from a to z.</div>
        <button class="p-add"><a href="../about">More Info</a></button>
      </div></div>
      <div class="p-flex"><div class="p-flex-in">
        <img class="p-img" src="http://localhost:3000/kindle.jpg"/>
        <div class="p-name">Kindle Paperwhite</div>
        <div class="p-price">$129.99</div>
        <div class="p-desc">Ever dropped your phone, table or kindle in the
water? Of course, you have.</div>
        <button class="p-add"><a href="../about">More Info</a></button>
      </div></div>
      <div class="p-flex"><div class="p-flex-in">
        <img class="p-img" src="http://localhost:3000/ws.png"/>
        <div class="p-name">Amazon Web Services</div>
        <div class="p-price">Tier 1: Free</div>
        <div class="p-desc">On-demand cloud computing services. Enough
said.</div>
        <button class="p-add"><a href="../about">More Info</a></button>
      </div></div>
      <div class="p-flex"><div class="p-flex-in">
        <img class="p-img" src="http://localhost:3000/echo.jpg"/>
        <div class="p-name">Echo Dot</div>
        <div class="p-price">$29.99+</div>
        <div class="p-desc">"Hey Alexa!" We hope you never lose your voice
again.</div>
        <button class="p-add"><a href="../about">More Info</a></button>
      </div></div>
      <div class="p-flex"><div class="p-flex-in">
        <img class="p-img" src="http://localhost:3000/go.jpg"/>
        <div class="p-name">Amazon Go</div>
        <div class="p-price">$30.00+</div>
```

```
        <div class="p-desc">300 people in the store, two cashiers. Frustrating.
          Have no fear Amazon Go is here.</div>
        <button class="p-add"><a href ="../about">More Info</a></button>
      </div></div>
    </div>
  </body>
</html>
```

3. In the **about.html** file, paste the following code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>About</title>
  </head>
  <link rel="stylesheet" type="text/css" href="css/about.css">
  <div class ="header1"><span>Amazon</span> Services</div>
  <body>
    <div id="p-flex">
      <div class="p-flex"><div class="p-flex-in">
        <img class="p-img" src="http://localhost:3000/firestick.jpg"/>
        <div class="p-name">Fire TV Stick 4k</div>
        <div class="p-price">$49.99</div>
        <div class="p-desc">Fire TV Stick 4K allows you to enjoy a more complete
4K Ultra HD streaming experience. Launch and control all your favorite movies and
TV shows with the next-gen Alexa Voice Remote.</div>
      </div></div>

      <div class="p-flex"><div class="p-flex-in">
        <img class="p-img" src="http://localhost:3000/prime.jpg"/>
        <div class="p-name">Amazon Prime</div>
        <div class="p-price">$60.00+</div>
        <div class="p-desc">The best online store in the world with approximately
120 million products sold annually. The highest selling product on Amazon.com are
books.</div>
      </div></div>

      <div class="p-flex"><div class="p-flex-in">
        <img class="p-img" src="http://localhost:3000/kindle.jpg"/>
        <div class="p-name">Kindle Paperwhite</div>
        <div class="p-price">$129.99</div>
        <div class="p-desc">Use your sleek e-reader, Kindle Paperwhite, at the
pool, beach or in the bath. Guess what? You no longer have to worry about getting
your "book" wet.</div>
      </div></div>
```

```
      <div class="p-flex"><div class="p-flex-in">
        <img class="p-img" src="http://localhost:3000/ws.png"/>
        <div class="p-name">Amazon Web Services</div>
        <div class="p-price">Tier 1: Free</div>
        <div class="p-desc">Secure cloud services platform, offering compute
power, database storage, content delivery and other functionality to help
businesses scale and grow.</div>
      </div></div>

      <div class="p-flex"><div class="p-flex-in">
        <img class="p-img" src="http://localhost:3000/echo.jpg"/>
        <div class="p-name">Echo Dot</div>
        <div class="p-price">$49.99+</div>
        <div class="p-desc">Amazon’s Dot is a smart speaker that packs all the
technology and functionality of the original Echo into a much smaller package.
It's Amazon's best-selling smart speaker, primarily due to its low entry
cost.</div>
        </div></div>

      <div class="p-flex"><div class="p-flex-in">
        <img class="p-img" src="http://localhost:3000/go.jpg"/>
        <div class="p-name">Amazon Go</div>
        <div class="p-price">$30.00+</div>
        <div class="p-desc">Using their smart phones, customers scan the Amazon
Go app when entering the store. Computer vision technology keeps track of what
they pull from the shelves and charges their credit card when they walk out.
        </div>
      </div></div>
    </div>
  </body>
</html>
```

4. In the **404.html** file, paste the following code:

```
<!DOCTYPE html>
<html lang ="en">
<head>
    <meta charset="UTF-8">
    <meta name ="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie-edge">
    <title>Page not found</title>
</head>
<body>
    <h1>Oops, page not found!</h1>
```

```
</body>
</html>
```

Code explanation: The home.js, about.hmtl, and 404.html files provides structure for what our app will look like but without any styling. If you run node app.js in your terminal, the routes will work, and you will see the html and that's about it. The 404.html page will render an error message if you decide to go to an undefined route. For example, going to localhost:3000/happy. Since /happy is not defined in our main application, you will receive an "oops, page not found!" error message.

Serving Static Files

All files in the public directory will serve static files, such as JavaScript, CSS, and images to the user when our routes are loaded.

1. In the **home.css** file, paste the following code:

```
.header1{
  font-size: 55px;
  font-style: italic;
  text-align: center;
  padding: 35px;
}
span {
  color: #f46b41;
}

   /* Flex container */
#p-flex {
  max-width: 1500px;
  margin: 0 auto;
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
}
/* Product items */
div.p-flex {
  width: 25%;
}
div.p-flex-in {
  box-sizing: border-box;
  margin: 5px;
```

```
    padding: 20px;
    border: 3px solid #000;
    background: #F0F0F0;
  }
  img.p-img {
    width: 100%;
    height: auto;
  }
  div.p-name {
    font-weight: bold;
    font-size: 1.1em;
  }
  div.p-price {
    color: #f44242;
  }
  div.p-desc {
    color: #888;
    font-size: 0.9em;
  }

  a {
    text-decoration: none;

  }
  a:link{
    color: white;
  }

  a:visited{
    color: black;
  }

  a:hover{
    color: white;
  }
  button.p-add {
    background: #f46b41;
    color: #fff;
    border: none;
    width: 100%;
    padding: 10px;
    margin: 10px 5px 5px 5px;
    font-size: 1.1em;
    font-weight: bold;
    cursor: pointer;
```

```
    }
    /* Responsive */
    @media only screen and (max-width: 1024px) {
      div.p-flex { width: 33%; }
    }
    @media only screen and (max-width: 600px) {
      div.p-flex { width: 50%; }
    }
    /* [DOES NOT MATTER] */
    html, body {
      padding: 0;
      margin: 0;
      font-family: arial, sans-serif;
    }
```

2. In the **about.css** file, paste the following code:

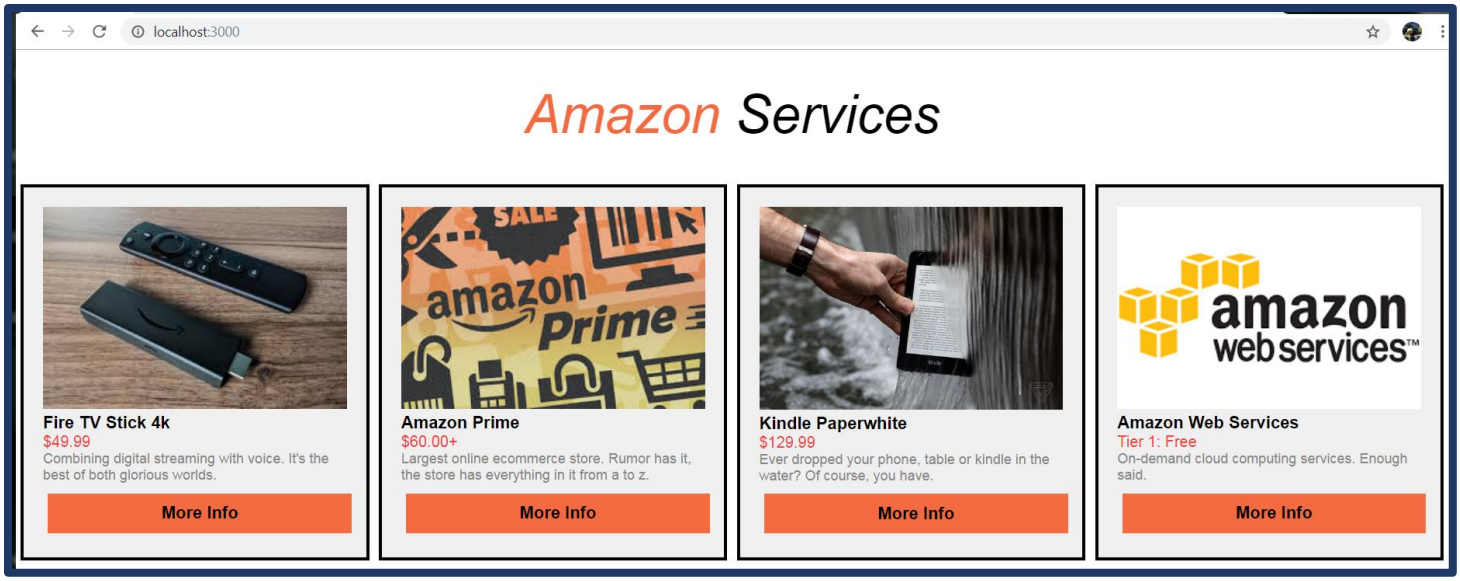
```
.header1 {
      font-size: 55px;
      font-style: italic;
      text-align: center;
      padding: 35px;
    }
    span {
      color: #f46b41;
    }

       /* Flex container */
    #p-flex {
      max-width: 1500px;
      margin: 0 auto;
      display: flex;
      flex-direction: row;
      flex-wrap: wrap;
    }
    /* Product items */
    div.p-flex {
      width: 25%;
    }
    div.p-flex-in {
      box-sizing: border-box;
      margin: 5px;
      padding: 20px;
```

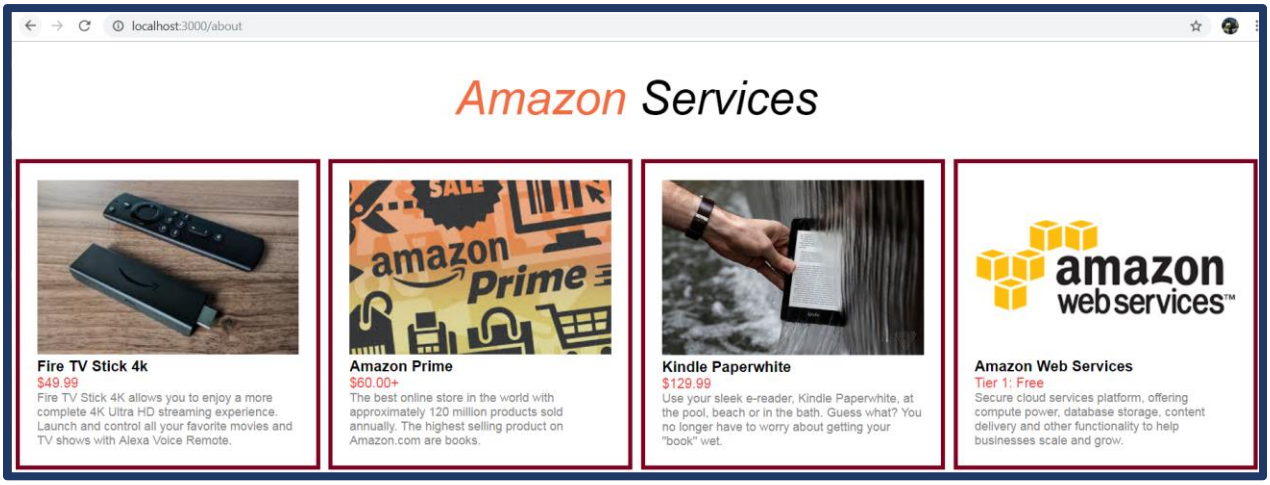
```
    border: 5px solid #800020 ;
    background #F0F0F0;
  }
  img.p-img {
    width: 100%;
    height: auto;
  }

  div.p-name {
    font-weight: bold;
    font-size: 1.1em;
  }
  div.p-price {
    color: #f44242;
  }
  div.p-desc {
    color: #888;
    font-size: 0.9em;
  }
  button.p-add {
    background: #f46b41;
    color: #fff;
    border: none;
    width: 100%;
    padding: 10px;
    margin: 10px 5px 5px 5px;
    font-size: 1.1em;
    font-weight: bold;
    cursor: pointer;
  }
  /* Responsive */
  @media only screen and (max-width: 1024px) {
    div.p-flex { width: 33%; }
  }
  @media only screen and (max-width: 600px) {
    div.p-flex { width: 50%; }
  }
  /* [DOES NOT MATTER] */
  html, body {
    padding: 0;
    margin: 0;
    font-family: arial, sans-serif;
  }
```

3. In the terminal, run **node app.js**.
4. Refresh **localhost:3000**, you should see the following on the landing page:



5. Go to **localhost:3000/about**, you should see the following on the about page:



Free Resources

Node.js

NPM

Express

Books/Tutorials

Getting Mean by Simon Holmes

Udemy: NodeJs: The Complete Guide by Maximillian S.