

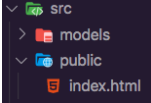
# JavaScript Frameworks

## Section 2: Node and Express Deployment

### 4.1 Deploying to Heroku

To deploy an app to Heroku you would need a Heroku account as well as a Github account. In the index.js file we would use the following line of code to make reference to a public folder which will have the entry point i.e. index.html file.

```
app.use(express.static('src/public'));
```



To test that the above app.use static method is working and the index.html page will work with our web server we can run the terminal command npm start (nodemon) script to run our server, when we visit the <http://localhost:3000> root route, the index.html page should be served.

The Heroku page provides a documented guide on deploying a Node.js application to their platform (<https://devcenter.heroku.com/articles/getting-started-with-nodejs>).

Once the Heroku CLI has been installed on your machine (using the command line heroku --version will confirm which version is installed on your local machine) we would need to prepare the application for Heroku deployment.

In the terminal cd into the root folder directory of the app we want to deploy we would want to run the following command:

```
$ heroku create $ git remote -v
```

This will run and come up with a unique name for our app and create a git remote attached to our GitHub. We can run the git remote - v command which will display the heroku and GitHub remotes we have which were created by the first command above.

We should ensure that we are on our master branch before running the next command. To do this we can run the following commands:

```
$ git branch $ git checkout master
```

The first command will tell us which branch we are on while the second command will change us to the desired branch. Once we are on the master branch we can run the following command:

```
$ git push heroku master
```

 This will push our remote git app to heroku.

If this command should not work we can do this manually by running the following command:

```
$ heroku git:remote -a [appName]
```

The [appName] should be replaced with the unique name of the app that was created with the heroku create command. This command is for Heroku setting up a connect between the git remote. The -a flag refers to the app which is why we pass in the name of the app after this flag.

This command should set our git remote heroku to the Heroku app URL which ends with a “.get” which is important. Now that our git remote is connected with Heroku we can run the previous command git push heroku master. This should now run the process of compressing our application and configuring it in a way that Heroku would be able to then deploy the app. We should see the following status printed in the terminal:

```
Build Succeeded!
Compressing...
Launching...
Verifying deploy...
done.
```

If we now run the below command this should open our Heroku application in a new browser window/tab:

```
$ heroku open
```

This will serve the file in our static method i.e. the index.html file in the public folder. If we see this in the browser we have successfully deployed our Node/Express application to Heroku and is available to the world on the web.

---

## 4.2 Build Input Form

In the index.html file we can build out a form where the user can input values which will allow users to use the RESTful API that we build with Node and Express and Mongoose via the browser.

So far we have been using Postman which is fine for testing routes during production and making sure the API routes are doing what they should be doing. We use input forms so that the user has somewhere to interact and enter values from the browser which will enable them to interact with the database (i.e. API). Below is an example HTML form:

```
<form>
  <field>
    <label>RESTful API</label>
    <input type="text">
  </field>
</form>
```

Nothing will happen with the form because it is not connected to an event handler. This is the beginning of the RESTful API i.e. an input field/form for GET request, one for POST, PUT and a DELETE requests. We can then use JavaScript to wire the form together with the eventHandler request logic.