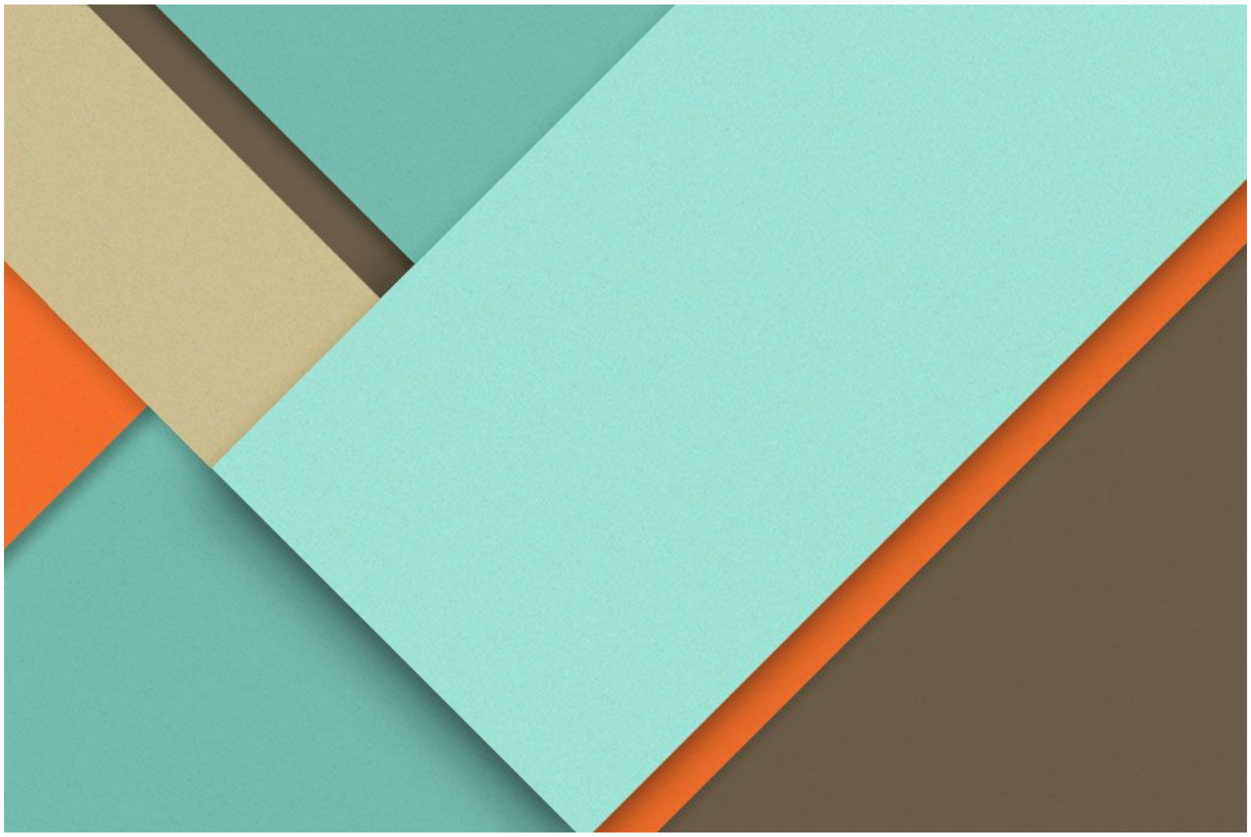


QR Code Stuff

Full Stack with Feathers & Quasar

Haris Hashim
© 2019 harishashim@gmail.com

Overview

A step by step training manual style instruction to accompany mentor and mentee discussion and learning session. This is a full stack guide how to accomplish “stuff” related to QR Code.

Table of Contents

Introduction

Searching on Google on how to implement QR Code with Node JS will result to the following npm package. https://www.npmjs.com/package/qrcode

So we wonder how to do stuff with this library!

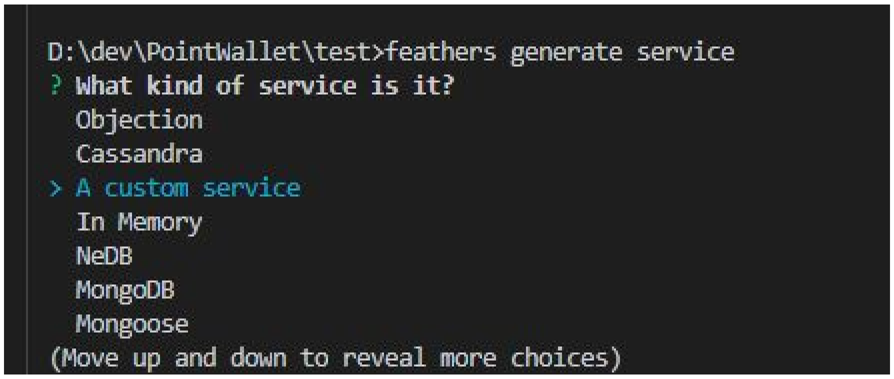
To demonstrate that we will create a backend service to generate QR code and front-end to consume that service and display the result.

This manual assume that Node JS, Feathers JS, Postman and Quasar is already installed and reader already familiar with how to create project with Feathers and Quasar.

Backend with Feathers

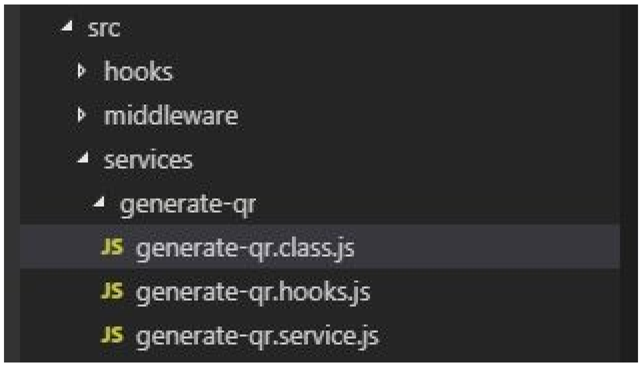
Creating the service

1. Generate a project.
2. Generate a custom service. Emphasize on "custom".



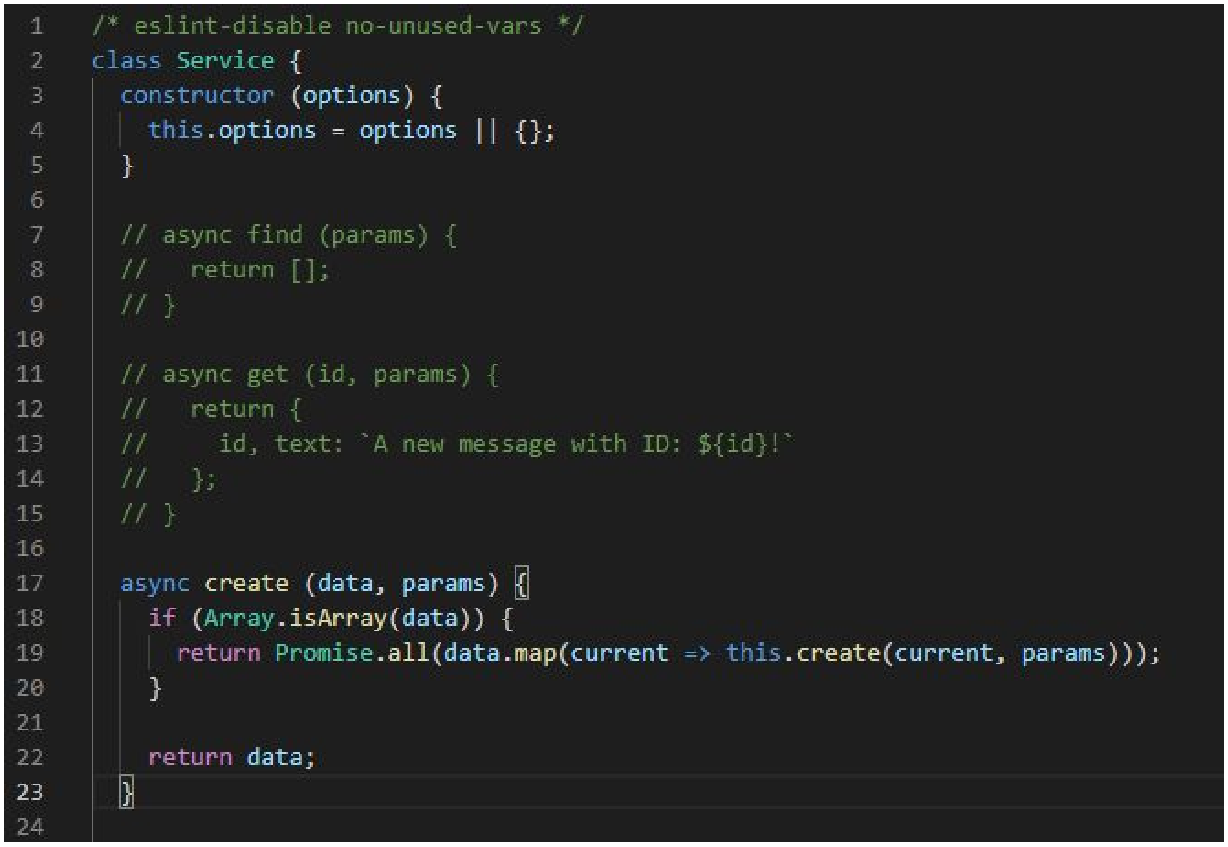
```
D:\dev\PointWallet\test>feathers generate service
? What kind of service is it?
  Objection
  Cassandra
> A custom service
  In Memory
  NeDB
  MongoDB
  Mongoose
(Move up and down to reveal more choices)
```

3. Name the above service GenerateQR. By doing this the service endpoint will be automatically created as "/generate-qr".
4. Find the source code generated as per below screenshot and open up generate-qr.class.js file.



```
src
  hooks
  middleware
  services
    generate-qr
      generate-qr.class.js
      generate-qr.hooks.js
      generate-qr.service.js
```

5. Comment out all methods in Service class except "create" method. Basically this means that we can only perform POST request, the rest will return error.



```
/* eslint-disable no-unused-vars */
class Service {
  constructor (options) {
    this.options = options || {};
  }

  // async find (params) {
  //   return [];
  // }

  // async get (id, params) {
  //   return {
  //     id, text: `A new message with ID: ${id}!`
  //   };
  // }

  async create (data, params) {
    if (Array.isArray(data)) {
      return Promise.all(data.map(current => this.create(current, params)));
    }

    return data;
  }
```

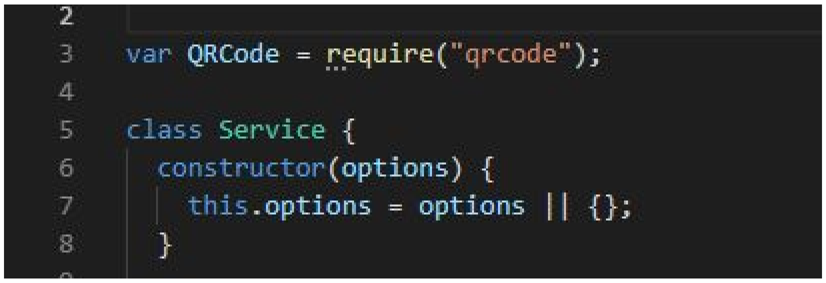
Adding node-qrcode library package

It is time to follow instruction from node-qrcode npm page.

1. Type this command to add the library to our project. Type it in shell or command prompt after navigating or cd to our project root directory.

```
npm install --save qrcode
```

2. After adding the library to our project, we need to import it inside source code that will use it. Import it in generate-qr.class.js.

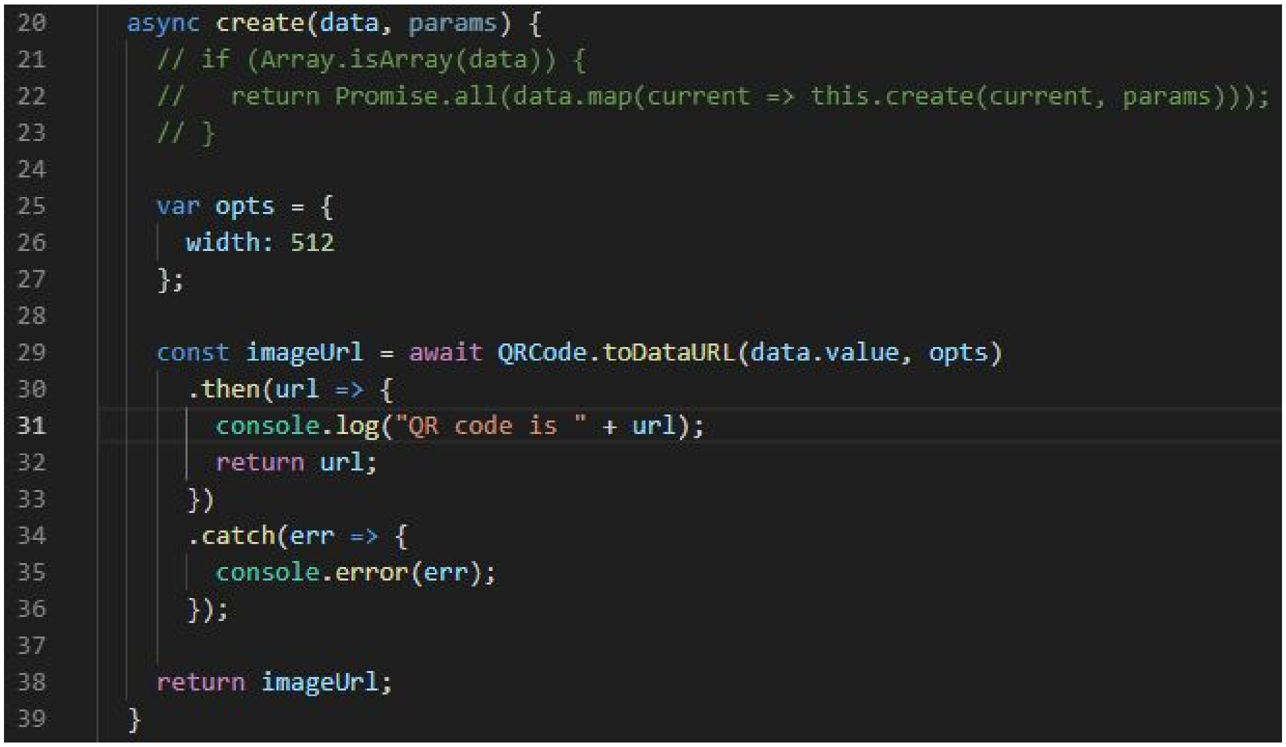


```

var QRCode = require("qrcode");

class Service {
  constructor(options) {
    this.options = options || {};
  }
```

3. Reading though node-qrcode documentation, we will discover how to use the API to generate image by using toDataURL(). What is not clear early on is that we can also pass option to control how QR code is created.

   To cut the explanation short. I will execute toDataURL function with "width" option. This option allows us to create QR code image with the width specified in pixels (or px). Since QR Code is square, height is always equal to width.

   Update create method to have bellow codes.



```
async create(data, params) {
  // if (Array.isArray(data)) {
  //   return Promise.all(data.map(current => this.create(current, params)));
  // }

  var opts = {
    width: 512
  };

  const imageUrl = await QRCode.toDataURL(data.value, opts)
    .then(url => {
      console.log("QR code is " + url);
      return url;
    })
    .catch(err => {
      console.error(err);
    });

  return imageUrl;
}
```

4. Run the Feathers backend.

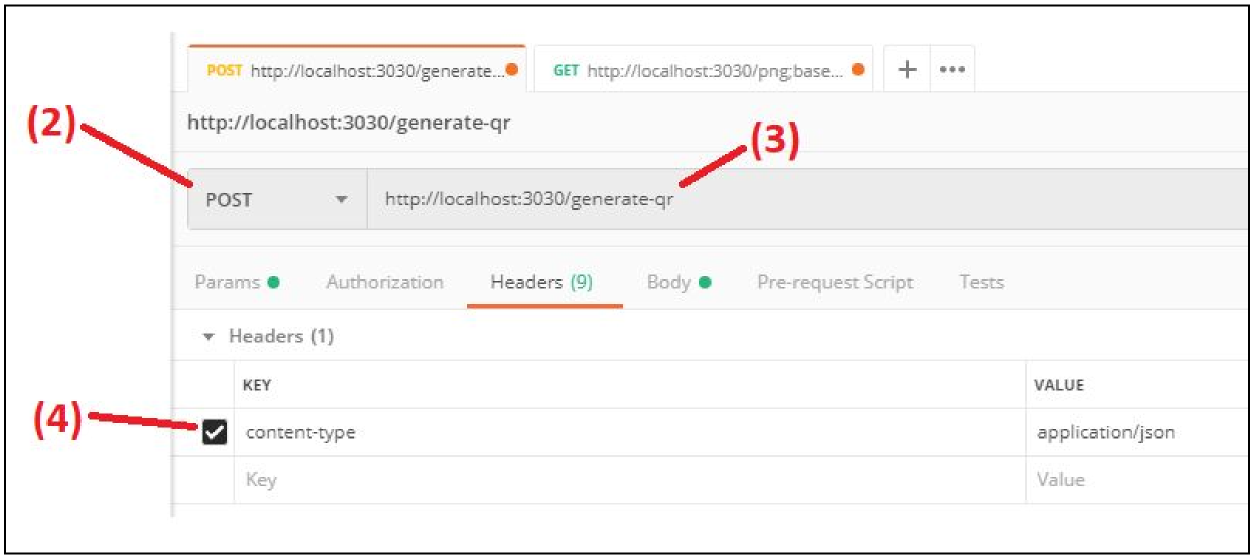
Testing with Postmen

There is a backend already running our QR code generation service. What is the quickest way to test and consume the service? The answer is Postman.

Take note that The Feathers backend will be running in localhost and by default on port 3030. Our service is a POST that is expecting a JSON data with value field.

Quick intro to postmen

1. Run Postman.
2. Change the request type to POST.
3. Use the generate-qr service as URL in below picture.
4. Set header content-type as below picture.

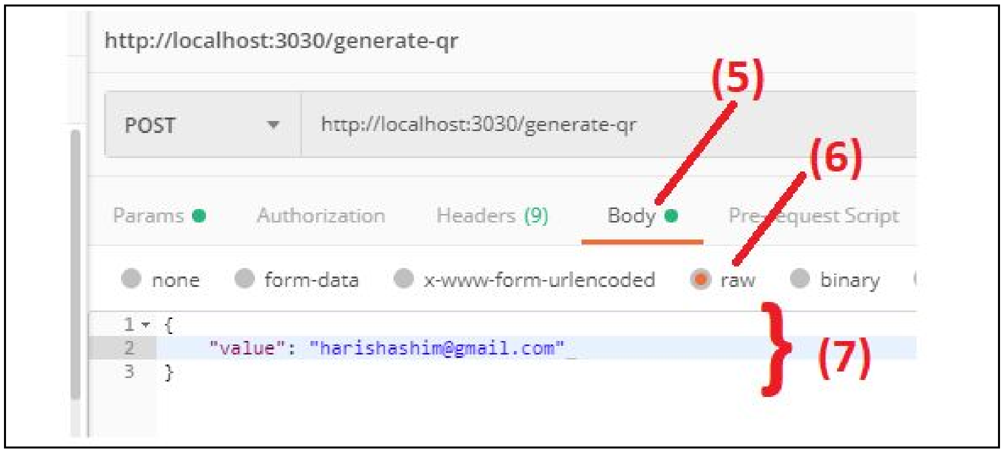


http://localhost:3030/generate-qr

POST http://localhost:3030/generate-qr

KEY	VALUE
✓ content-type	application/json
Key	Value

5. Change to body tab.
6. Select raw
7. Type in the JSON with value field.



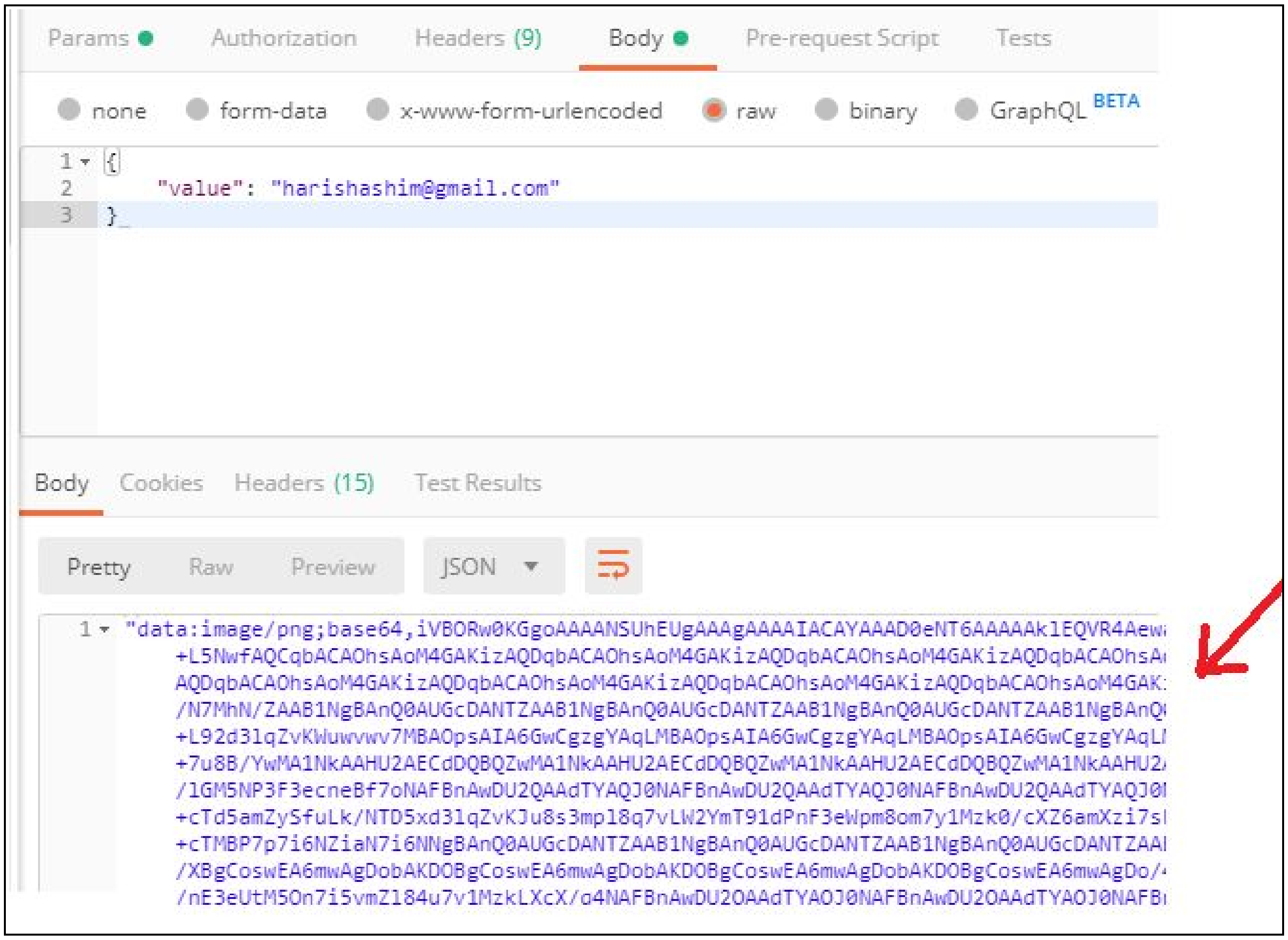
http://localhost:3030/generate-qr

POST http://localhost:3030/generate-qr

```
{
    "value": "harishashim@gmail.com"
}
```

8. Press the Send button.

9. Observe the result.



```
{
    "value": "harishashim@gmail.com"
}
```

```
"data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAgAAAAIACAYAAAD0eNT6AAAAAk1EQVR4Aew
    +L5NwfAQCqbACAOhsAoM4GAKizAQDqbACAOhsAoM4GAKizAQDqbACAOhsAoM4GAKizAQDqbACAOhsA
    AQDqbACAOhsAoM4GAKizAQDqbACAOhsAoM4GAKizAQDqbACAOhsAoM4GAKizAQDqbACAOhsAoM4GAK
    /N7MhN/ZAAB1NgBAnQ0AUGcDANTZAAB1NgBAnQ0AUGcDANTZAAB1NgBAnQ0AUGcDANTZAAB1NgBAnQ
    +L92d31qZvKWuwvwv7MBAOpsAIA6GwCgzgYAqLMBAOpsAIA6GwCgzgYAqLMBAOpsAIA6GwCgzgYAqL
    +7u8B/YwMA1NkAAHU2AECdDQBQZwMA1NkAAHU2AECdDQBQZwMA1NkAAHU2AECdDQBQZwMA1NkAAHU2
    /1GM5NP3F3ecneBf7oNAFBnAwDU2QAAdTYAQJ0NAFBnAwDU2QAAdTYAQJ0NAFBnAwDU2QAAdTYAQJ0
    +cTd5amZySfuLk/NTD5xd31qZvKJu8s3mpl8q7vLW2YmT91dPnF3eWpm8om7y1Mzk0/cXZ6amXzi7s
    +cTMBP7p7i6NZiaN7i6NNgBAnQ0AUGcDANTZAAB1NgBAnQ0AUGcDANTZAAB1NgBAnQ0AUGcDANTZAA
    /XBgCoswEA6mwAgDobAKDOBgCoswEA6mwAgDobAKDOBgCoswEA6mwAgDobAKDOBgCoswEA6mwAgDo/
    /nE3eUtM5On7i5vmZ184u7v1MzkLXcX/a4NAFBnAwDU2OAAdTYAOJ0NAFBnAwDU2OAAdTYAOJ0NAFB
```

How to Test and See?

The quickest way to see QR Code as per above response returned by generate-qr POST service is to create html file with bellow text and open it in a browser

<html>

<img

alt="QR Code" src="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAg ... ”>

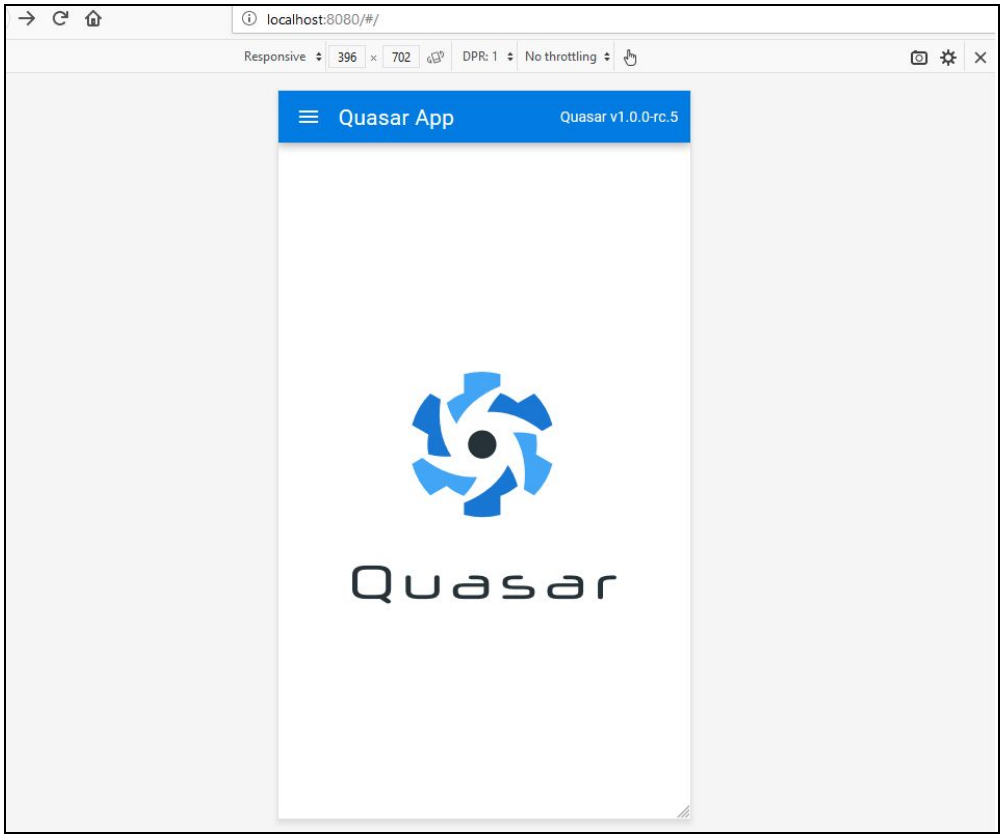
</html>

But we don't need to do this right? Because we will create a fully functional front end app that will display returned QR Code in the next section!

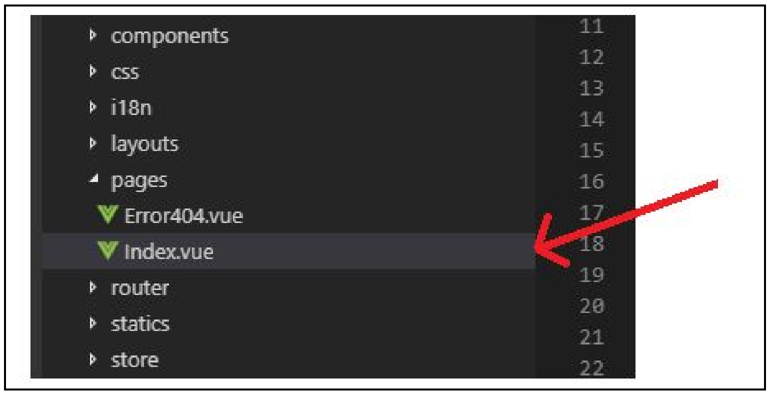
Frontend with Quasar

Starting Up!

1. Create the project. Don't forget to include Axios plugin, since we will be using it to consume backend service.
2. Run it to get the feel! Open http://localhost:8080 in browser.



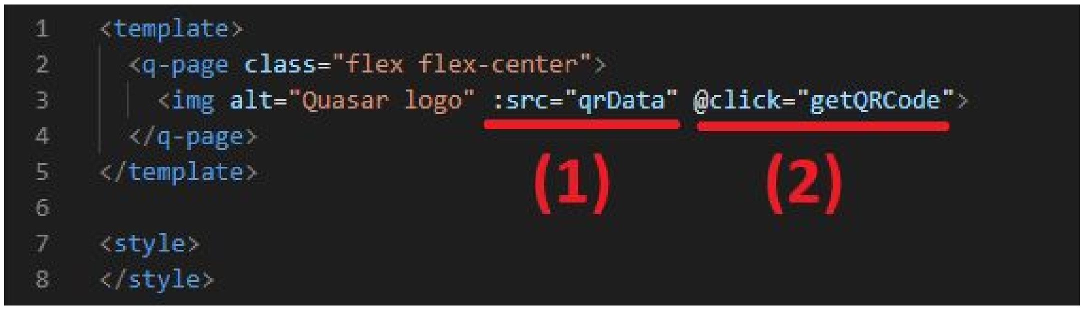
3. Above is how it looks like when run and shown in browser mobile view. We are going to add some codes so that when Quasar logo is clicked, a POST request is sent to backend and Quasar logo will change into QR code.
4. We are going to change index.vue file among the pages.



-+

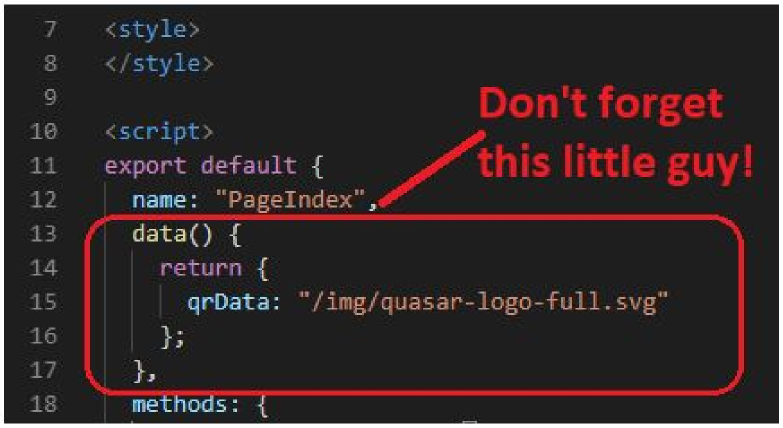
Some Coding Required!

1. Change template to bellow:



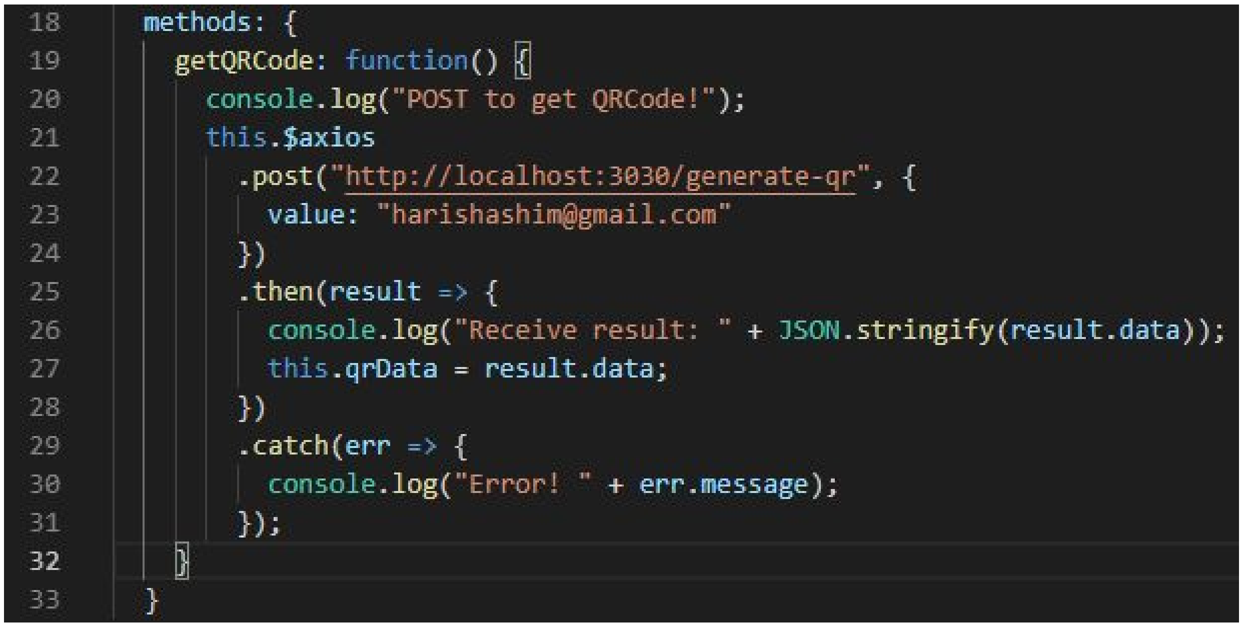
```
<template>
  <q-page class="flex flex-center">
    <img alt="Quasar logo" :src="qrData" @click="getQRCode">
  </q-page>
</template>

<style>
</style>
```

2. Change number 1 is so that we can use qrData to store image src prop rather than just hard coding it in the template. Define qrData as below.



```
<style>
</style>

<script>
export default {
  name: "PageIndex",
  data() {
    return {
      qrData: "/img/quasar-logo-full.svg"
    };
  },
  methods: {
```

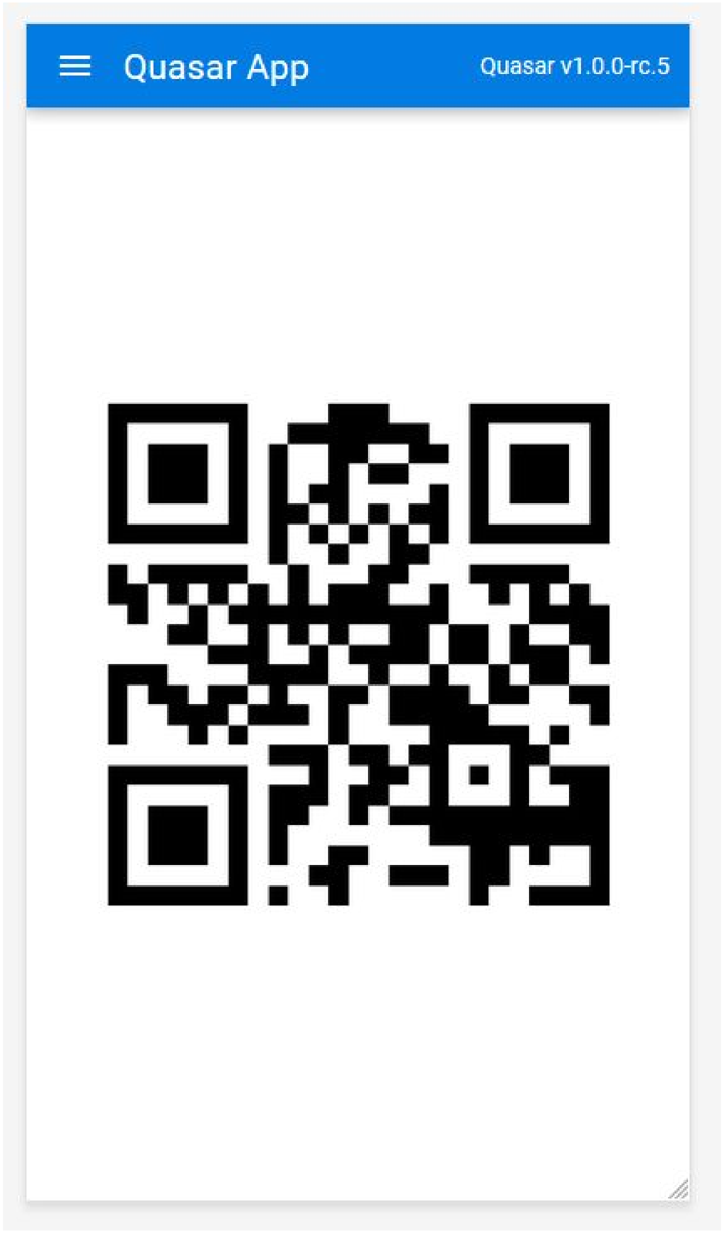
3. Change number 2 is so that when the image is clicked, we will call getQRCode method. Define getQRCode method as below.



```
  methods: {
    getQRCode: function() {
      console.log("POST to get QRCode!");
      this.$axios
        .post("http://localhost:3030/generate-qr", {
          value: "harishashim@gmail.com"
        })
        .then(result => {
          console.log("Receive result: " + JSON.stringify(result.data));
          this.qrData = result.data;
        })
        .catch(err => {
          console.log("Error! " + err.message);
        });
    }
  }
```

Testing

1. Save Index.vue and if there is no error, http://localhost:8080 opened in browser will hot update and display the same UI we see earlier with Quasar logo.
2. Click Quasar logo and QR code image will be displayed instead!



3. If doesn’t work. Ensure that backend is running and service call work with Postman. Ensure that quasar frontend is running an serving localhost port 8080. Pressing F12 will also help by displaying Chrome Dev Tool and find console tab to see if there is any error.