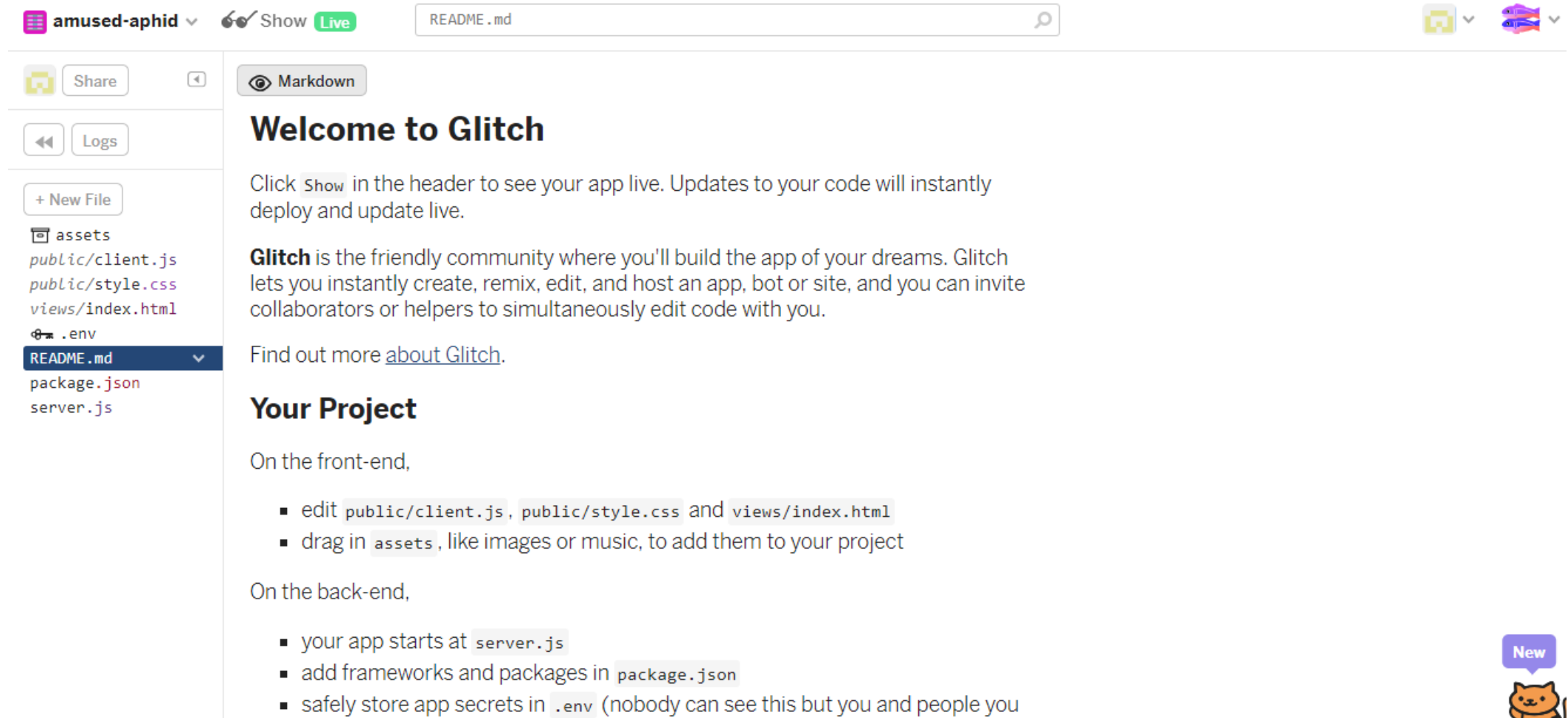# Glitch Tour

Prerequisite tools on your Workstation

- none!

- (apart from a browser + a github account)

- First screen is the "source" for a running, live web project

amused-aphid ⌄ 👓 Show `Live` README.md 🔍

Share ◁

⏪ Logs

+ New File

📦 assets
public/client.js
public/style.css
views/index.html
🔒 .env
README.md ⌄
package.json
server.js

👁 Markdown

Welcome to Glitch

Click `Show` in the header to see your app live. Updates to your code will instantly deploy and update live.

Glitch is the friendly community where you'll build the app of your dreams. Glitch lets you instantly create, remix, edit, and host an app, bot or site, and you can invite collaborators or helpers to simultaneously edit code with you.

Find out more about Glitch.

Your Project

On the front-end,

- edit `public/client.js`, `public/style.css` and `views/index.html`
- drag in `assets`, like images or music, to add them to your project

On the back-end,

- your app starts at `server.js`
- add frameworks and packages in `package.json`
- safely store app secrets in `.env` (nobody can see this but you and people you

`New`

Project name
(automatically
generated)

Link to
running app
(to share)

Files in
the
project

Current
File
(editable)

Link to your
Profile

Link to
Community,
resources,
options

amused-aphid ▾   👓 Show Live   README.md 🔍

Share ◁

◀◀ Logs

+ New File

📦 assets
public/client.js
public/style.css
views/index.html
🔑 .env
README.md ▾
package.json
server.js

◀ Markdown

Welcome to Glitch

Click `Show` in the header to see your app live. Updates to your code will instantly deploy and update live.

Glitch is the friendly community where you'll build the app of your dreams. Glitch lets you instantly create, remix, edit, and host an app, bot or site, and you can invite collaborators or helpers to simultaneously edit code with you.

Find out more about Glitch.

Your Project

On the front-end,

- edit `public/client.js`, `public/style.css` and `views/index.html`
- drag in `assets`, like images or music, to add them to your project

On the back-end,

- your app starts at `server.js`
- add frameworks and packages in `package.json`
- safely store app secrets in `.env` (nobody can see this but you and people you

New

A Dream of the Future

Oh hi,

Tell me your hopes and dreams:

[Dreams!]

[Submit Dream]

- Find and count some sheep
- Climb a really tall mountain
- Wash the dishes

Made with Glitch!

- Project is always running live (provided there are no source errors)

Project Structure

- Glitch projects not just web sites!

- They are web apps, divided into:

  - Front-end files

  - Back-end files

```
📦 assets
public/client.js
public/style.css
views/index.html
🔑 .env
README.md          ⌄
package.json
server.js
```

Front End

```
assets
public/client.js
public/style.css
views/index.html
```

- Comparable to the web site you developed in previous module(s).

  - html files + stylesheets + images

- Templating also possible.

- Also, access to the server side is implicit.

- This means you can build apps that have behaviour + state (much more on this later).

Back End

```
🔑 .env
README.md          ⌄
package.json
server.js
```

- An application - written in JavaScript - and hosted in the cloud.

- Many types of application supported.

- We will focus on JavaScript applications written using **node.js**.

- This is the default toolkit for Glitch.

The Starter App

The Starter App

Share   ◀

⏪ Logs

+ New File

📁 assets
public/client.js
public/style.css
views/index.html ⌄
🔌 .env
README.md
package.json
server.js

A Dream of the Future

Oh hi,

Tell me your hopes and dreams:

Dreams!

Submit Dream

- Find and count some sheep
- Climb a really tall mountain
- Wash the dishes

Made with Glitch!

```html
1  <!-- This is a static file -->
2  <!-- served from your routes in server.js -->
3
4  <!-- You might want to try something fancier: -->
5  <!-- html/nunjucks docs: https://mozilla.github.io/nunjucks/ -->
6  <!-- pug: https://pugjs.org/ -->
7  <!-- haml: http://haml.info/ -->
8  <!-- hbs(handlebars): http://handlebarsjs.com/ -->
9
10 <!DOCTYPE html>
11 <html lang="en">
12   <head>
13     <title>Welcome to Glitch!</title>
14     <meta name="description" content="A cool thing made with Glitch">
15     <link id="favicon" rel="icon" href="https://glitch.com/edit/favicon-app.ico" type="image/x-icon">
16     <meta charset="utf-8">
17     <meta http-equiv="X-UA-Compatible" content="IE=edge">
18     <meta name="viewport" content="width=device-width, initial-scale=1">
19
20     <!-- import the webpage's stylesheet -->
21     <link rel="stylesheet" href="/style.css">
22
23     <!-- import the webpage's client-side javascript file -->
24     <script src="/client.js" defer></script>
25   </head>
26   <body>
27     <header>
28       <h1>
29         A Dream of the Future
30       </h1>
31     </header>
32
33     <main>
34       <p class="bold">Oh hi,</p>
35
36       <p>Tell me your hopes and dreams:</p>
37
38       <form>
39         <input name="dream" type="text" maxlength="100" placeholder="Dreams!" aria-labelledby="submit-dream">
40         <button type="submit" id="submit-dream">Submit Dream</button>
```

The Starter App

A Dream of the Future

Oh hi,

Tell me your hopes and dreams:

> [Dreams!]
>
> [Submit Dream]

- Find and count some sheep
- Climb a really tall mountain
- Wash the dishes

Made with Glitch!

```html
<body>
  <header>
    <h1>
      A Dream of the Future
    </h1>
  </header>

  <main>
    <p class="bold">Oh hi,</p>

    <p>Tell me your hopes and dreams:</p>

    <form>
      <input name="dream" type="text" maxlength="100" placeholder="Dreams!"
                                      aria-labelledby="submit-dream">
      <button type="submit" id="submit-dream">Submit Dream</button>
    </form>

    <section class="dreams">
      <ul id="dreams"></ul>
    </section>
  </main>

  <footer>
    Made with <a href="https://glitch.com">Glitch</a>!
  </footer>
```

html

client side JavaScript

```html
<body>
  <header>
    <h1>
      A Dream of the Future
    </h1>
  </header>

  <main>
    <p class="bold">Oh hi,</p>

    <p>Tell me your hopes and dreams:</p>

    <form>
      <input name="dream" type="text" maxlength="100" placeholder="Dreams!"
                                        aria-labelledby="submit-dream">
      <button type="submit" id="submit-dream">Submit Dream</button>
    </form>

    <section class="dreams">
      <ul id="dreams"></ul>
    </section>
  </main>

  <footer>
    Made with <a href="https://glitch.com">Glitch</a>!
  </footer>
</body>
```

```javascript
console.log('hello world :o');

// our default array of dreams
const dreams = [
  'Find and count some sheep',
  'Climb a really tall SNOWY mountain',
  'Wash the dishes'
];

// define variables that reference elements on our page
const dreamsList = document.getElementById('dreams');
const dreamsForm = document.forms[0];
const dreamInput = dreamsForm.elements['dream'];

// a helper function that creates a list item for a given dream
const appendNewDream = function(dream) {
  const newListItem = document.createElement('li');
  newListItem.innerHTML = dream;
  dreamsList.appendChild(newListItem);
}

// iterate through every dream and add it to our page
dreams.forEach( function(dream) {
  appendNewDream(dream);
});

// listen for the form to be submitted and add a new dream when it is
dreamsForm.onsubmit = function(event) {
  // stop our form submission from refreshing the page
  event.preventDefault();

  // get dream value and add it to the list
  dreams.push(dreamInput.value);
  appendNewDream(dreamInput.value);

  // reset form
  dreamInput.value = '';
  dreamInput.focus();
};
```

server side JavaScript

```javascript
// server.js
// where your node app starts

// init project
const express = require('express');
const app = express();

// we've started you off with Express,
// but feel free to use whatever libs or frameworks you'd like through `package.json`.

// http://expressjs.com/en/starter/static-files.html
app.use(express.static('public'));

// http://expressjs.com/en/starter/basic-routing.html
app.get('/', function(request, response) {
  response.sendFile(__dirname + '/views/index.html');
});

// listen for requests :)
var listener = app.listen(process.env.PORT, function () {
  console.log('Your app is listening on port ' + listener.address().port);
});
```

The 'Dreams' App

- This app displays the content of the 'dreams' array in an unordered list and allows the user to insert a new dream.

- However, if we refresh the app the original array is displayed and new dream items are lost.

- At the moment, the array is stored in **client.js.** We will move it to the server (**server.js**).

- We will also modify some of the code so the new dream items are added to the array on the server, so if we refresh the app the new dream items are still there.

Modified client.js

```javascript
// client-side js
// run by the browser each time your view template is loaded

console.log('hello world :o');

// our default array of dreams

// define variables that reference elements on our page
const dreamsList = document.getElementById('dreams');
const dreamsForm = document.forms[0];
const dreamInput = dreamsForm.elements['dream'];

// a helper function that creates a list item for a given dream
const appendNewDream = function(dream) {
  const newListItem = document.createElement('li');
  newListItem.innerHTML = dream;
  dreamsList.appendChild(newListItem);
}

// iterate through every dream and add it to our page
$.get('/dreams', function(dreams) {
 dreams.forEach( function(dream) {
  appendNewDream(dream);
});
});

// listen for the form to be submitted and add a new dream when it is
dreamsForm.onsubmit = function(event) {
  // stop our form submission from refreshing the page
  event.preventDefault();
  const dream = dreamInput.value;
  // get dream value and add it to the list
  $.post('/dreams?' + $.param({dream: dream}), function() {
   console.log('New Dream added');
   appendNewDream(dream);
  // reset form
   dreamInput.value = '';
   dreamInput.focus();
  });
};
```

Modified server.js

```javascript
// server.js
// where your node app starts

// init project
const express = require('express');
const app = express();

// we've started you off with Express,
// but feel free to use whatever libs or frameworks you'd like through `package.json`.

// http://expressjs.com/en/starter/static-files.html
app.use(express.static('public'));

// http://expressjs.com/en/starter/basic-routing.html
app.get('/', function(request, response) {
  response.sendFile(__dirname + '/views/index.html');
});

app.get('/dreams', function (request, response) {
  response.send(dreams);
});

app.post('/dreams', function (request, response) {
  dreams.push(request.query.dream);
  response.sendStatus(200);
});

const dreams = [
  'Find and count some sheep',
  'Climb a really tall SNOWY mountain',
  'Wash the dishes'
];

// listen for requests :)
const listener = app.listen(process.env.PORT, function() {
  console.log('Your app is listening on port ' + listener.address().port);
});
```

- Client side JavaScript runs in each users browser

```javascript
dreamsForm.onsubmit = function(event) {
  // stop our form submission from refreshing the page
  event.preventDefault();
  const dream = dreamInput.value;
  // get dream value and add it to the list
  $.post('/dreams?' + $.param({dream: dream}), function() {
    console.log('New Dream added');
    appendNewDream(dream);
  // reset form
    dreamInput.value = '';
    dreamInput.focus();
  });
};
```

```javascript
// could also use the POST body instead of query string: http://expressjs.com/en/api.html#req.body
app.post("/dreams", function (request, response) {
  dreams.push(request.query.dream);
  response.sendStatus(200);
});
```
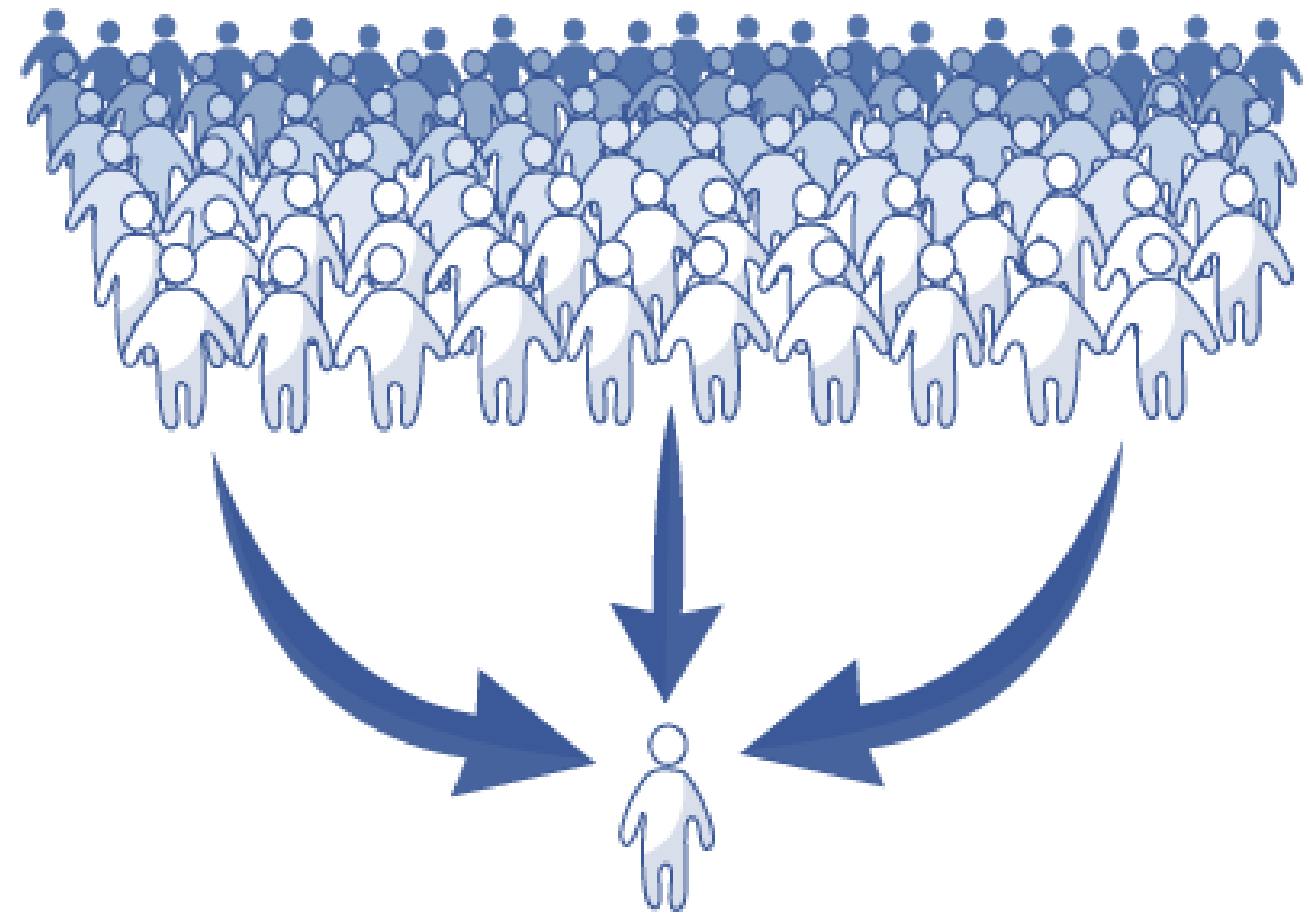
- A node runs the server side JavaScript. All browsers are connected to this node.

The 'Dreams' App

- This modified Glitch app (as basically any other webapp) is composed of two parts: **a client** and **a server**.

- The client is run by your browser when you load a page, the server is run by the machine on which your Glitch app is hosted.

- **server.js** is the entry point for the server-side of your web application. It is responsible for storing data you want to share among all the "clients" (that is, all the browsers that will connect to your application), and to serve (hence the name… server) the requests that the clients make.

The 'Dreams' App

- So, you can see that server.js defines a few "routes": for example app.get("/", ...).

- Routes tell the server how to respond to client requests.

- **app.get("/", ...)** describes what to respond when a client asks for the **/ path** of your webapp: that is, it serves *index.html.*

- If you look in index.html, you'll see that one of the lines "includes" client.js: so when a client (the browser) goes to **https://your-app.glitch.me/,** it also receives the "client-side" of your web application.

The 'Dreams' App

- So **server.js** and **client.js** run on two completely separate machines, on two completely separate environments, and serve two completely separate uses:

  - server.js is responsible of responding to requests, while

  - client.js… makes the requests to the server, and shows them to the user through the browser!

The 'Dreams' App

- So when in client.js you see **$.get("/dreams", ...),** it's the client-side that is asking *server.js* to serve the "route" /dreams. If you look in *server.js*, you'll see what app.get("/dreams", ...) does:

```
app.get("/dreams", function (request, response) {
    response.send(dreams);
});
```

- It sends the content of the dreams array to the client. This is how the client knows what's in your current dream list.

Skills developed in this Module

- Web App Development 1

  - Basic JavaScript knowledge.

  - Back end development in JavaScript .

- Front end JavaScript development is delivered in a different module.

```
// server.js
// where your node app starts

// init project
const express = require('express');
const app = express();

// we've started you off with Express,
// but feel free to use whatever libs or frameworks you'd like through `package.json`.

// http://expressjs.com/en/starter/static-files.html
app.use(express.static('public'));

// http://expressjs.com/en/starter/basic-routing.html
app.get('/', function(request, response) {
  response.sendFile(__dirname + '/views/index.html');
});

app.get('/dreams', function (request, response) {
  response.send(dreams);
});

app.post('/dreams', function (request, response) {
  dreams.push(request.query.dream);
  response.sendStatus(200);
});

const dreams = [
  'Find and count some sheep',
  'Climb a really tall SNOWY mountain',
  'Wash the dishes'
];

// Listen for requests :)
const listener = app.listen(process.env.PORT, function() {
  console.log('Your app is listening on port ' + listener.address().port);
});
```
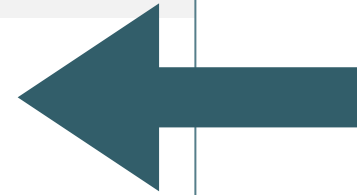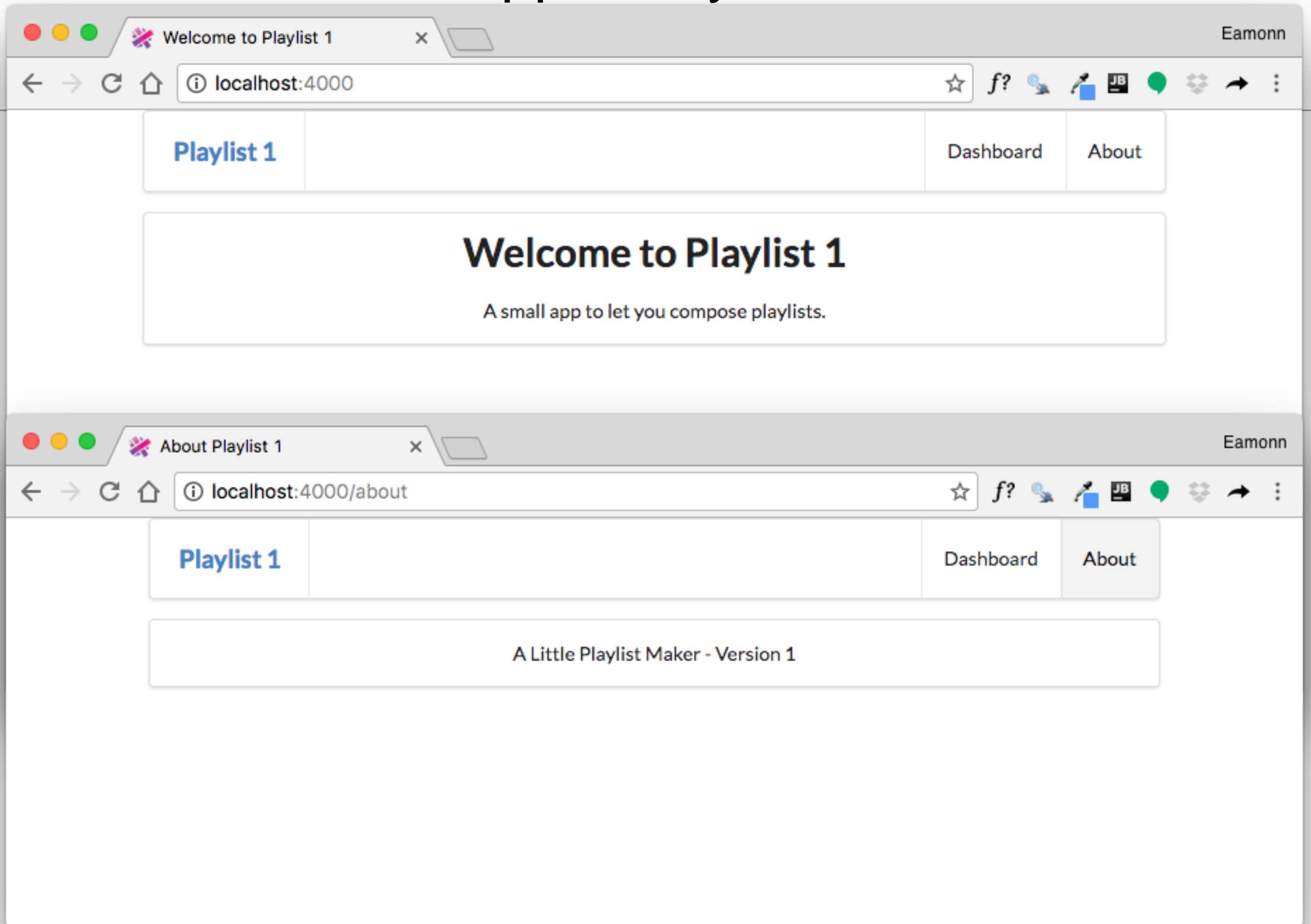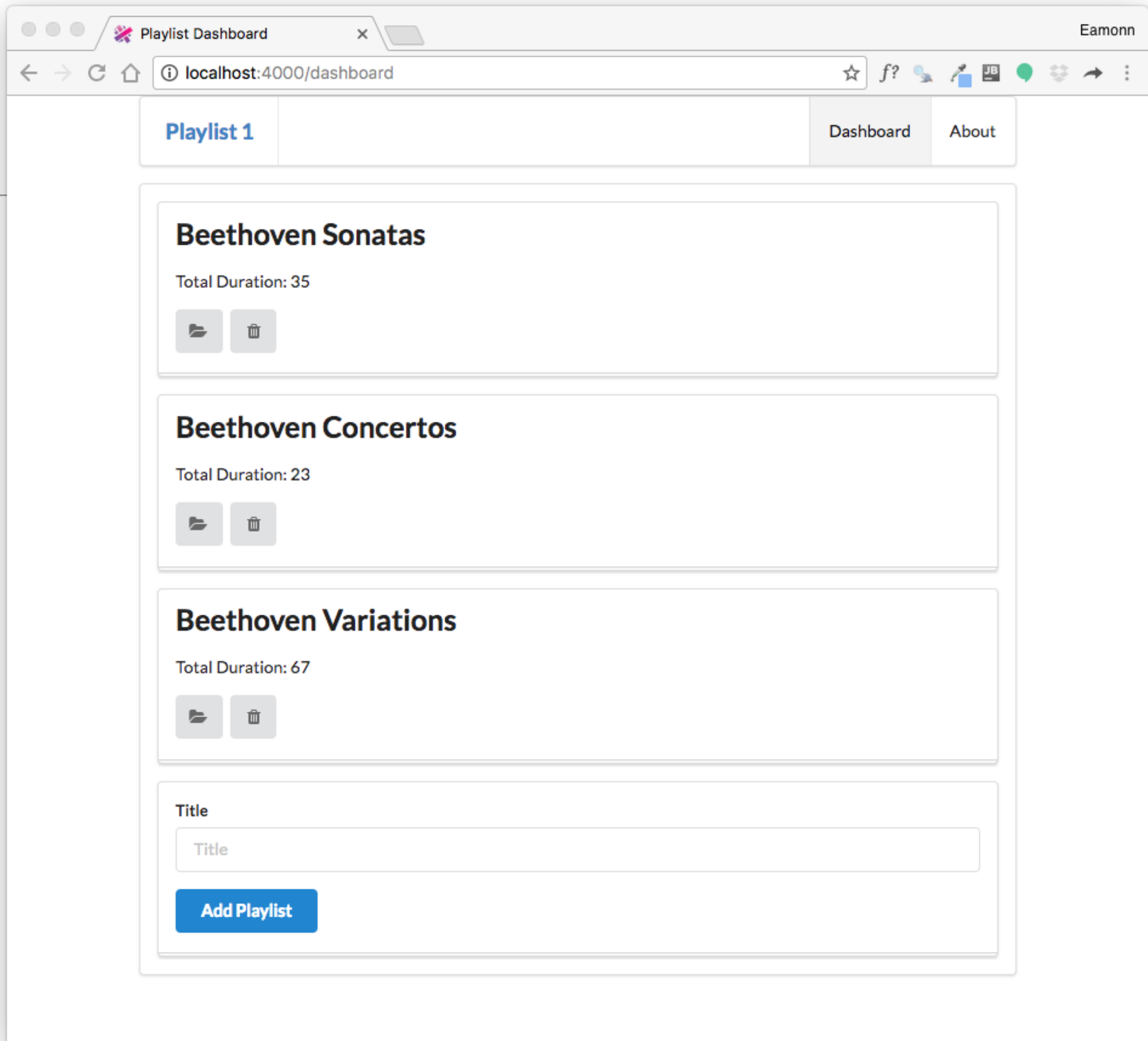
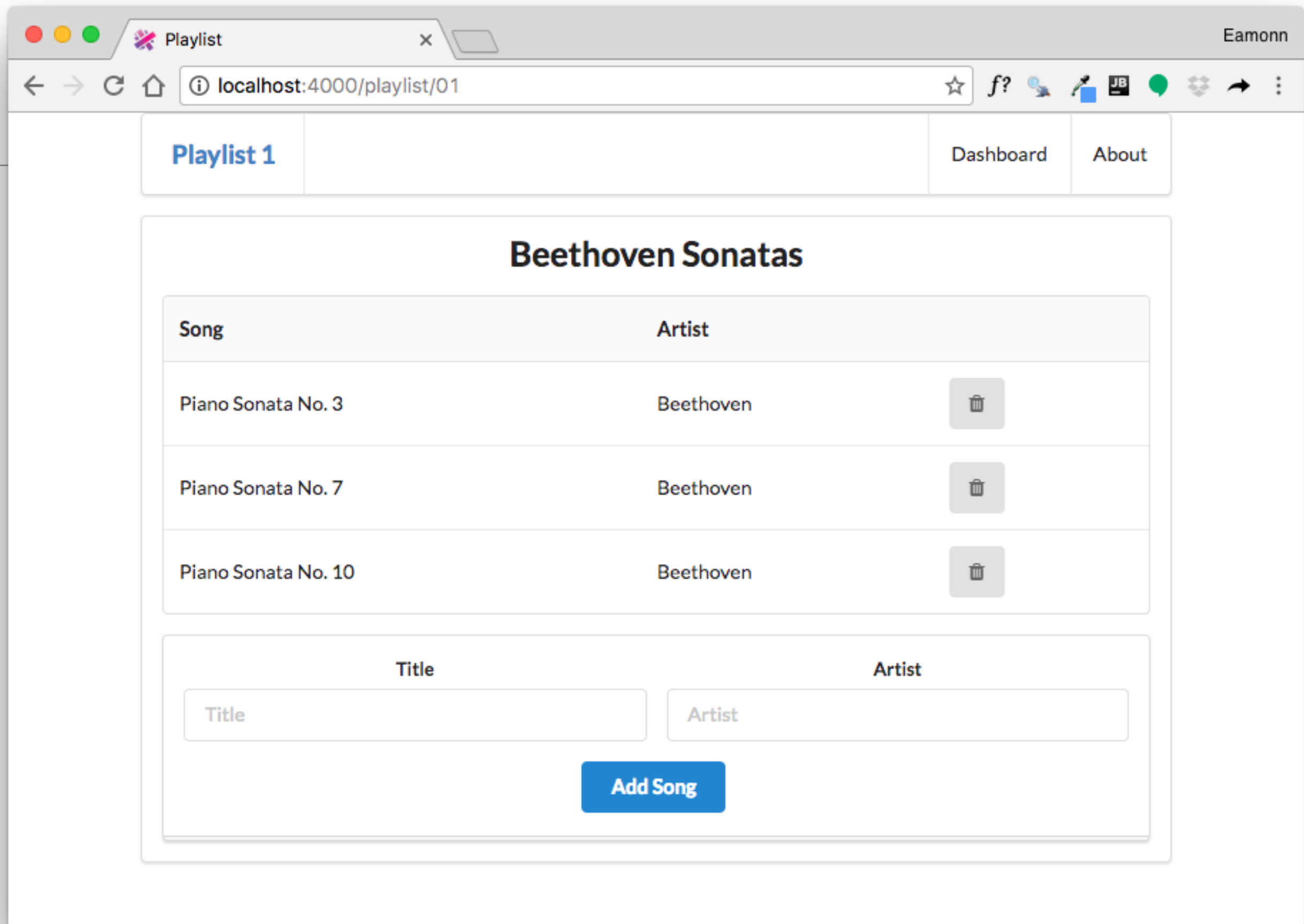We will learn what all of this means.

- + how to build a fully featured web app including:

  - templating;

  - creating forms to submit information;

  - how to store data in models;

  - creating user accounts, and how to tie an account to each user.

A tour of our first app - Playlist

Playlist 1

Dashboard   About

Beethoven Sonatas

Total Duration: 35

📂 🗑

Beethoven Concertos

Total Duration: 23

📂 🗑

Beethoven Variations

Total Duration: 67

📂 🗑

Title

Title

Add Playlist

Playlist Labs

- We will do three playlist labs

  - Playlist 1: simple rendering of static playlist.

  - Playlist 2: render multiple playlists, ability to delete playlists.

  - Playlist 3: ability to create playlists. Store playlists long term.

- These labs will be interleaved with JavaScript Introductory labs, which will gradually introduce you to the language.