

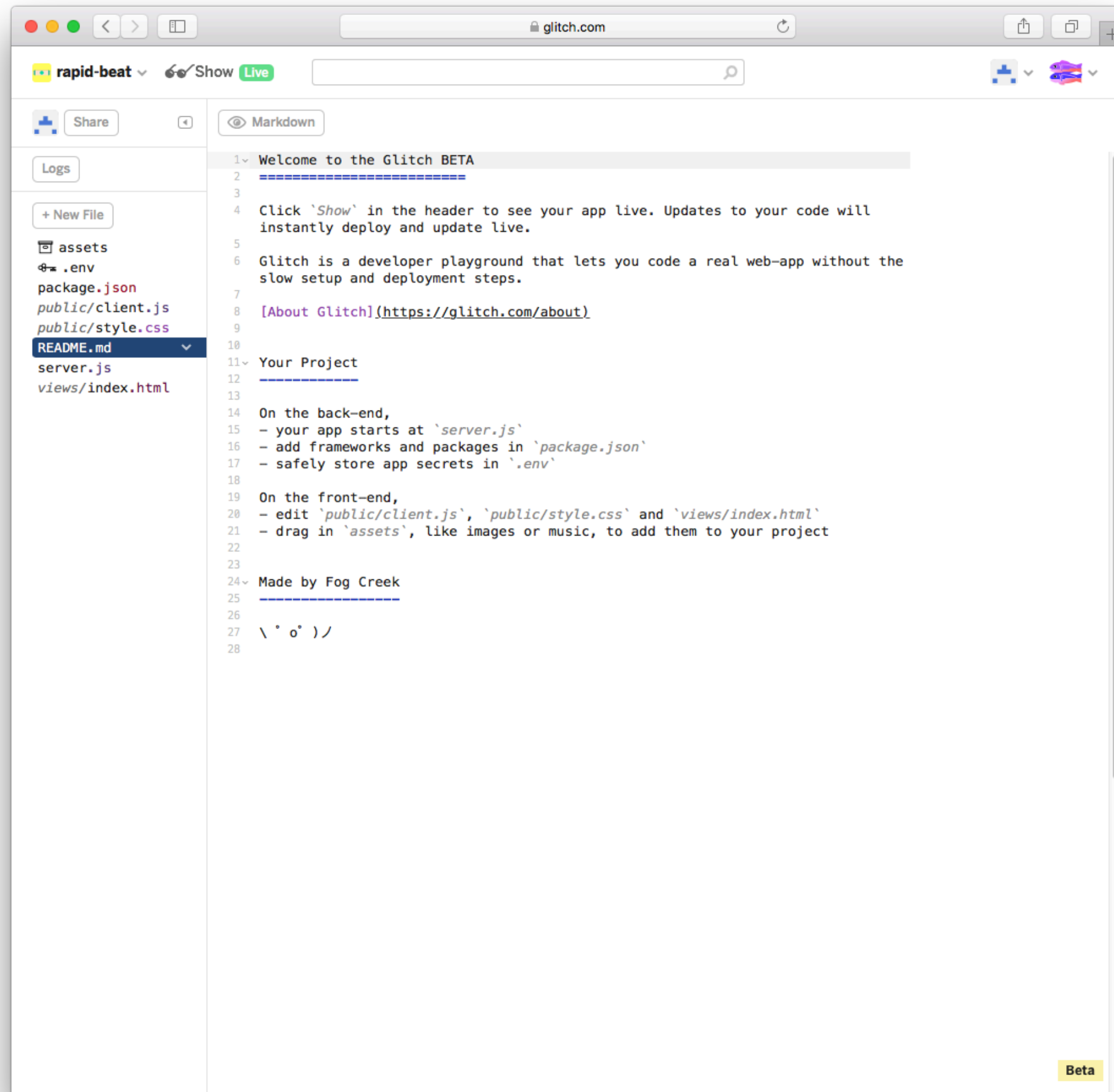
Glitch Tour

Prerequisite tools on your Workstation

none!

(apart from a browser + a github account)

First screen is the “source” for a running, live web project



Project name
(automatically
generated)

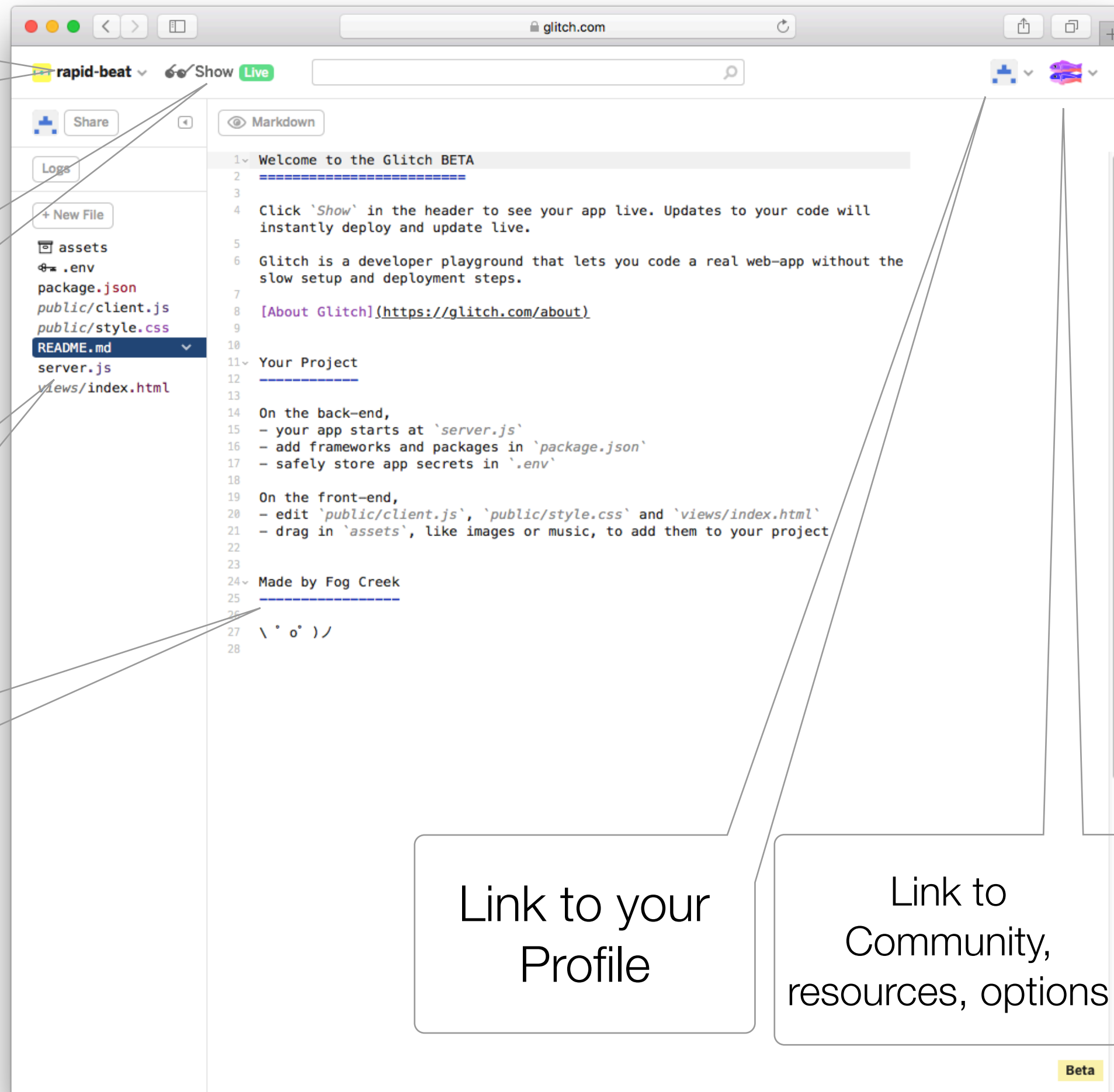
Link to running
app (to share)

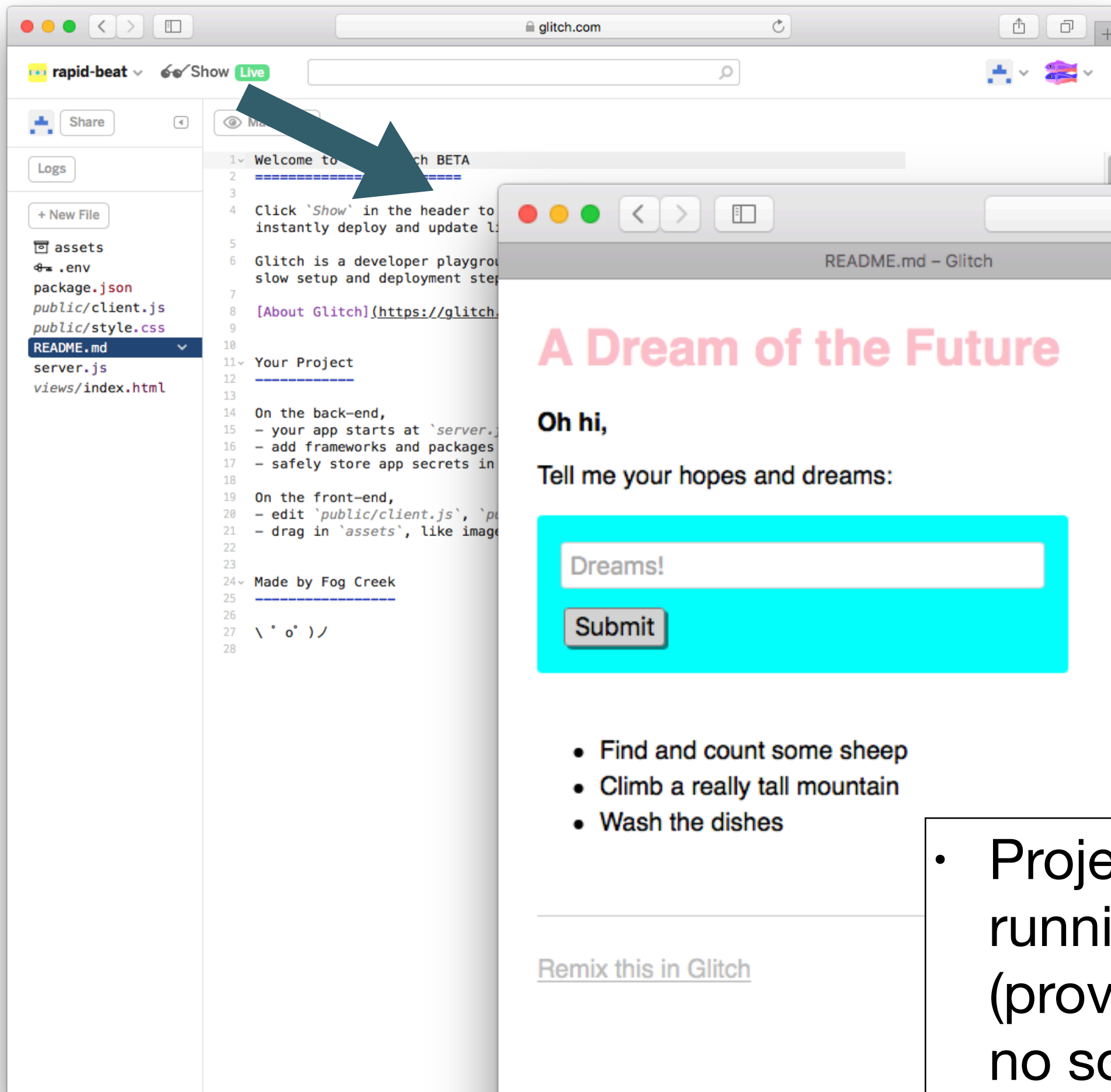
Files in the
project

Current File
(editable)

Link to your
Profile

Link to
Community,
resources, options

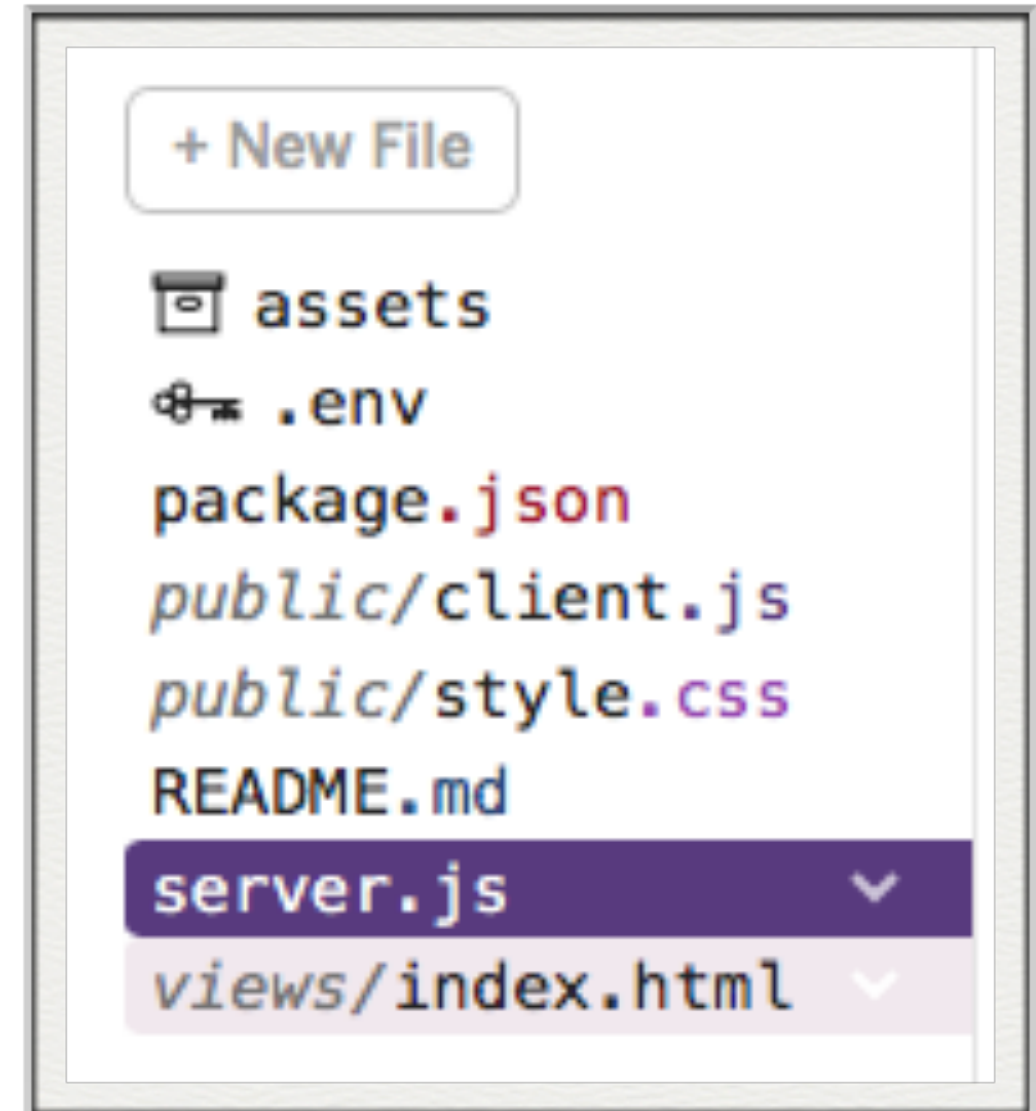





- Project is always running live (provided there are no source errors)

Project Structure

- Glitch projects not just web sites!
- They are fully featured web apps - with full server-side resources



Front End

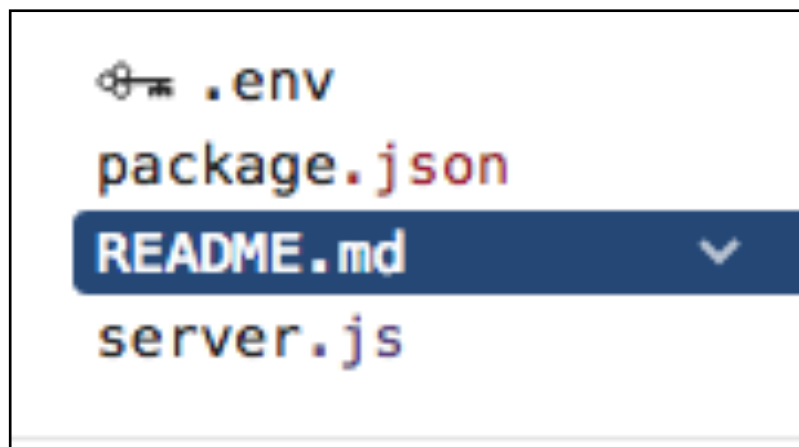


```
assets  
public/client.js  
public/style.css  
views/index.html
```

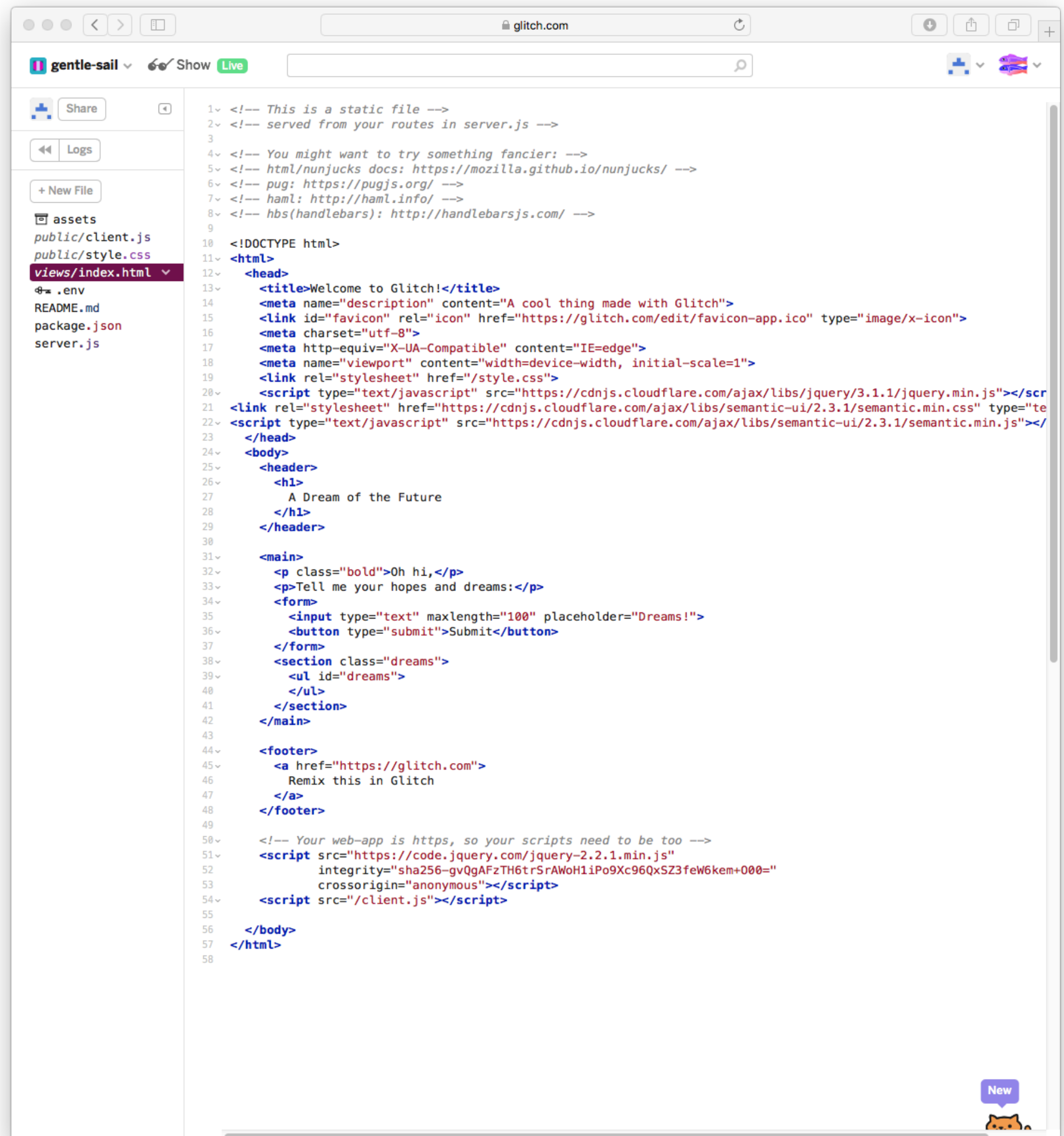
- Comparable to a static web site:
 - html files + stylesheets + images
- Templating also possible.
- Also, access to the server side is implicit.
- This means you can build apps that have behaviour + state (much more on this later)

Back end

- An application - written in javascript - and hosted in the cloud.
- Application built in Javascript using a technology called node.js

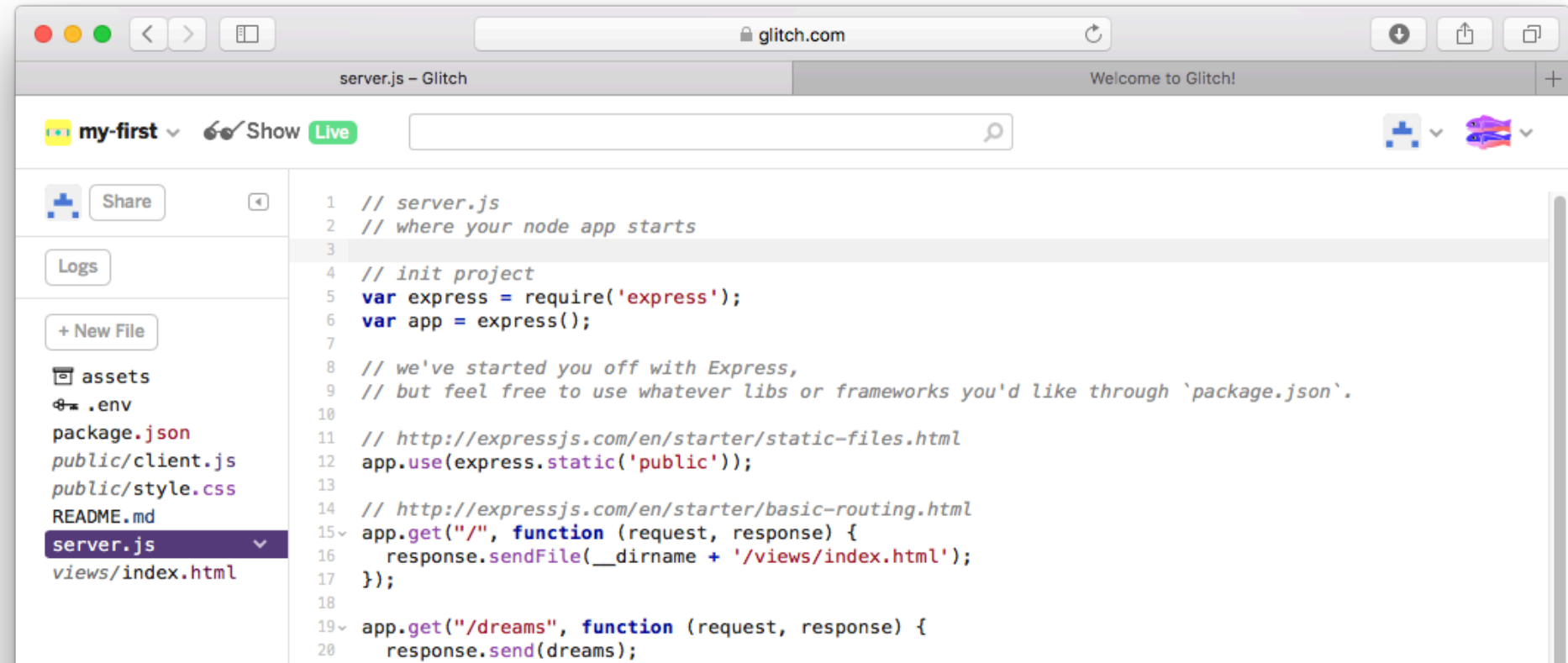


The Starter App



```
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>
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 <link rel="stylesheet" href="/style.css">
20 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.1/jquery.min.js"></scr
21 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.3.1/semantic.min.css" type="te
22 <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.3.1/semantic.min.js"></
23 </head>
24 <body>
25 <header>
26 <h1>
27 A Dream of the Future
28 </h1>
29 </header>
30
31 <main>
32 <p class="bold">Oh hi,</p>
33 <p>Tell me your hopes and dreams:</p>
34 <form>
35 <input type="text" maxlength="100" placeholder="Dreams!">
36 <button type="submit">Submit</button>
37 </form>
38 <section class="dreams">
39 <ul id="dreams">
40 </ul>
41 </section>
42 </main>
43
44 <footer>
45 <a href="https://glitch.com">
46 Remix this in Glitch
47 </a>
48 </footer>
49
50 <!-- Your web-app is https, so your scripts need to be too -->
51 <script src="https://code.jquery.com/jquery-2.2.1.min.js"
52 integrity="sha256-gvQgAFzTH6trSrAWoH1iPo9Xc96QxSZ3few6kem+000="
53 crossorigin="anonymous"></script>
54 <script src="/client.js"></script>
55
56 </body>
57 </html>
58
```

The Starter App



The screenshot shows a web browser window at glitch.com. The editor displays a file named 'server.js' with the following code:

```
1 // server.js
2 // where your node app starts
3
4 // init project
5 var express = require('express');
6 var app = express();
7
8 // we've started you off with Express,
9 // but feel free to use whatever libs or frameworks you'd like through `package.json`.
10
11 // http://expressjs.com/en/starter/static-files.html
12 app.use(express.static('public'));
13
14 // http://expressjs.com/en/starter/basic-routing.html
15 app.get("/", function (request, response) {
16   response.sendFile(__dirname + '/views/index.html');
17 });
18
19 app.get("/dreams", function (request, response) {
20   response.send(dreams);
21 });
```

The left sidebar shows a file explorer with the following structure:

- assets
- .env
- package.json
- public/client.js
- public/style.css
- README.md
- server.js (selected)
- views/index.html

A Dream of the Future

Oh hi,

Tell me your hopes and dreams:

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

[Remix this in Gornix](#)

```
use the POST body instead of query string: http://expressjs.com/en/api.html#req.body
ams", function (request, response) {
  request.query.dream);
  dStatus(200);

emory store for now

unt some sheep",
lly tall mountain",
shes",
n"

requests :)
app.listen(process.env.PORT, function () {
  'Your app is listening on port ' + listener.address().port);
```

A Dream of the Future

Oh hi,

Tell me your hopes and dreams:

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

[Remix this in Gomix](#)

```
<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 type="text" maxlength="100" placeholder="Dreams!">
      <button type="submit">Submit</button>
    </form>
    <section class="dreams">
      <ul id="dreams">
        </ul>
      </section>
  </main>

  <footer>
    <a href="https://gomix.com">
      Remix this in Gomix
    </a>
  </footer>
```

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 type="text" maxlength="100" placeholder="Your dream" />
      <button type="submit">Submit</button>
    </form>
    <section class="dreams">
      <ul id="dreams">
      </ul>
    </section>
  </main>

  <footer>
    <a href="https://gomix.com">
      Remix this in Gomix
    </a>
  </footer>
```

client side javascript

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

// by default, you've got jQuery,
// add other scripts at the bottom of index.html

$(function() {
  console.log('hello world :o');

  $.get('/dreams', function(dreams) {
    dreams.forEach(function(dream) {
      $('<li></li>').text(dream).appendTo('ul#dreams');
    });
  });

  $('form').submit(function(event) {
    event.preventDefault();
    dream = $('input').val();
    $.post('/dreams?' + $.param({dream: dream}), function() {
      $('<li></li>').text(dream).appendTo('ul#dreams');
      $('input').val('');
      $('input').focus();
    });
  });
});
```

server side javascript

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

// init project
var express = require('express');
var 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);
});

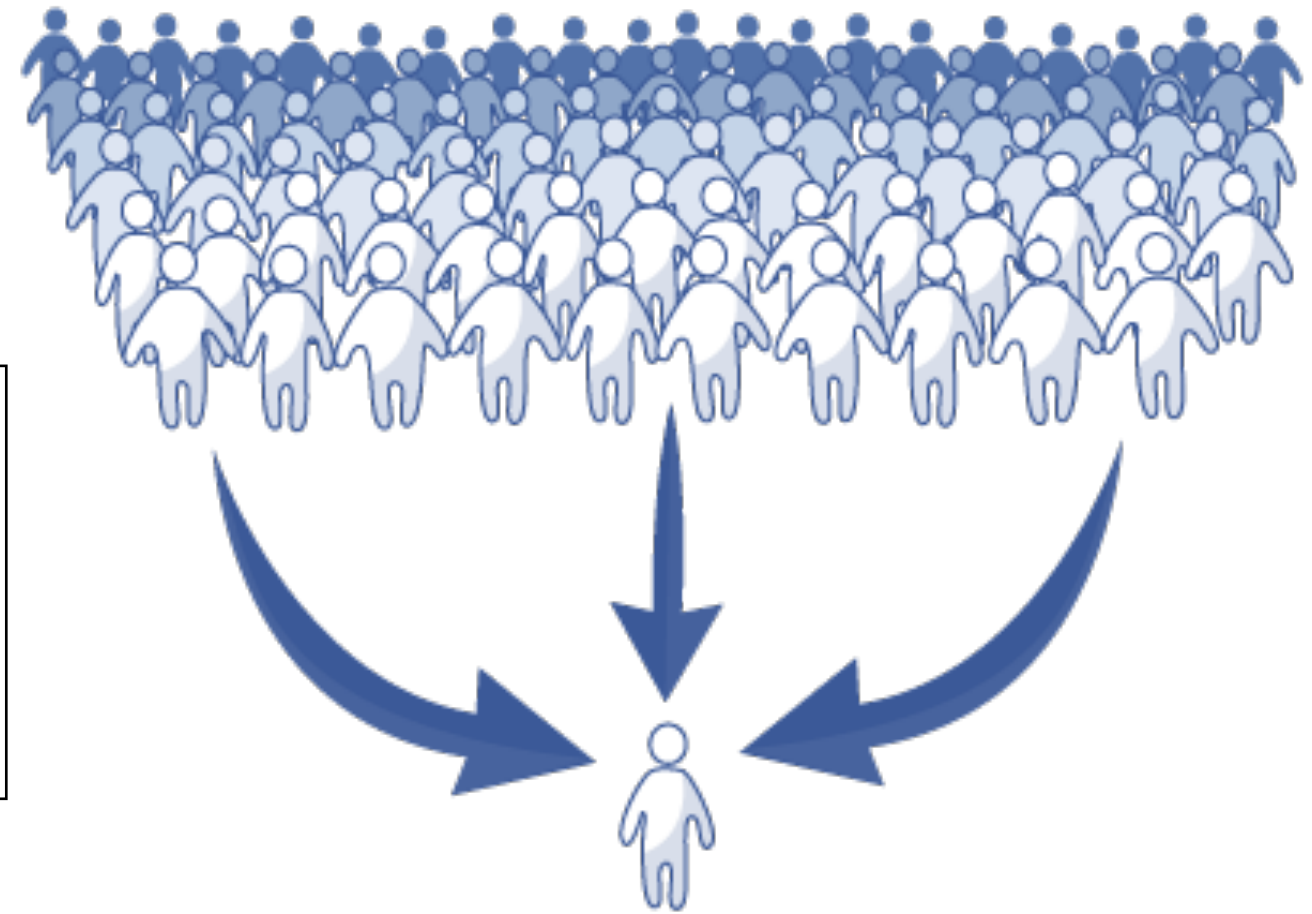
// could also use the POST body instead of query string: http://expressjs.com/en/api.html#req.body
~ app.post("/dreams", function (request, response) {
  dreams.push(request.query.dream);
  response.sendStatus(200);
});

// Simple in-memory store for now
~ var dreams = [
  "Find and count some sheep",
  "Climb a really tall mountain",
  "Wash the dishes"
];

// listen for requests :)
~ var 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

```
$('#form').submit(function(event) {  
  event.preventDefault();  
  dream = $('#input').val();  
  $.post('/dreams?' + $.param({dream: dream}), function() {  
    $('#<li></li>').text(dream).appendTo('ul#dreams');  
    $('#input').val('');  
    $('#input').focus();  
  });  
});
```



```
// 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 connected to this node

Skills for this Course

- Assumptions:
 - Foundation Knowledge in HTML + CSS
 - Working knowledge of Semantic UI CSS Framework
- Major focus of this course:
 - Javascript Programming
 - Node.js Web Application Development
- Glitch is the platform
- Front end javascript development will **not** be covered.

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

// init project
var express = require('express');
var 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'));

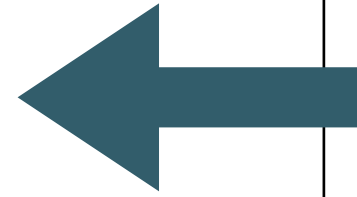
// http://expressjs.com/en/starter/basic-routing.html
app.get("/", function (request, response) {
  response.sendFile(__dirname + '/views/index.html');
});

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

// 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);
});

// Simple in-memory store for now
var dreams = [
  "Find and count some sheep",
  "Climb a really tall mountain",
  "Wash the dishes"
];

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

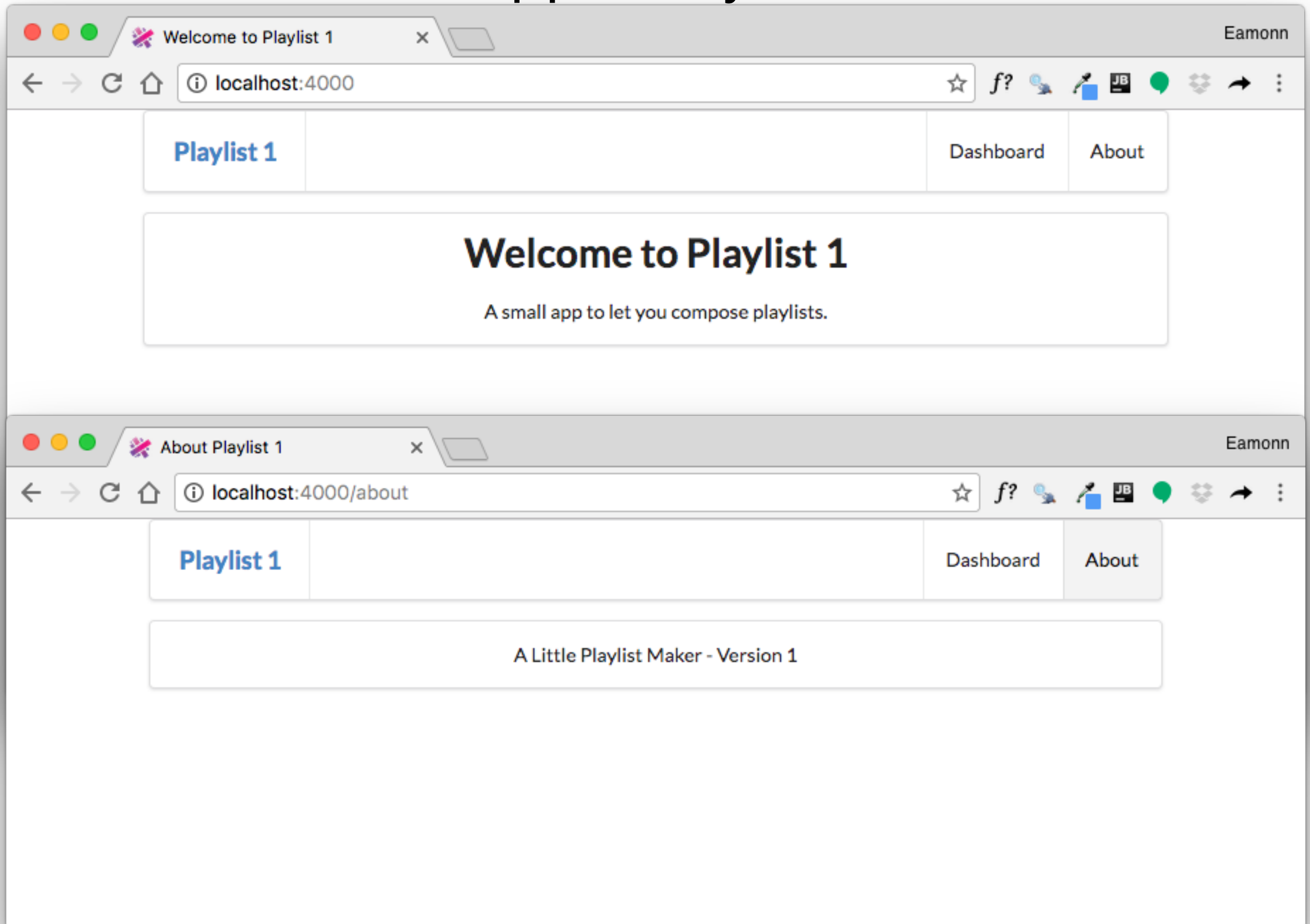


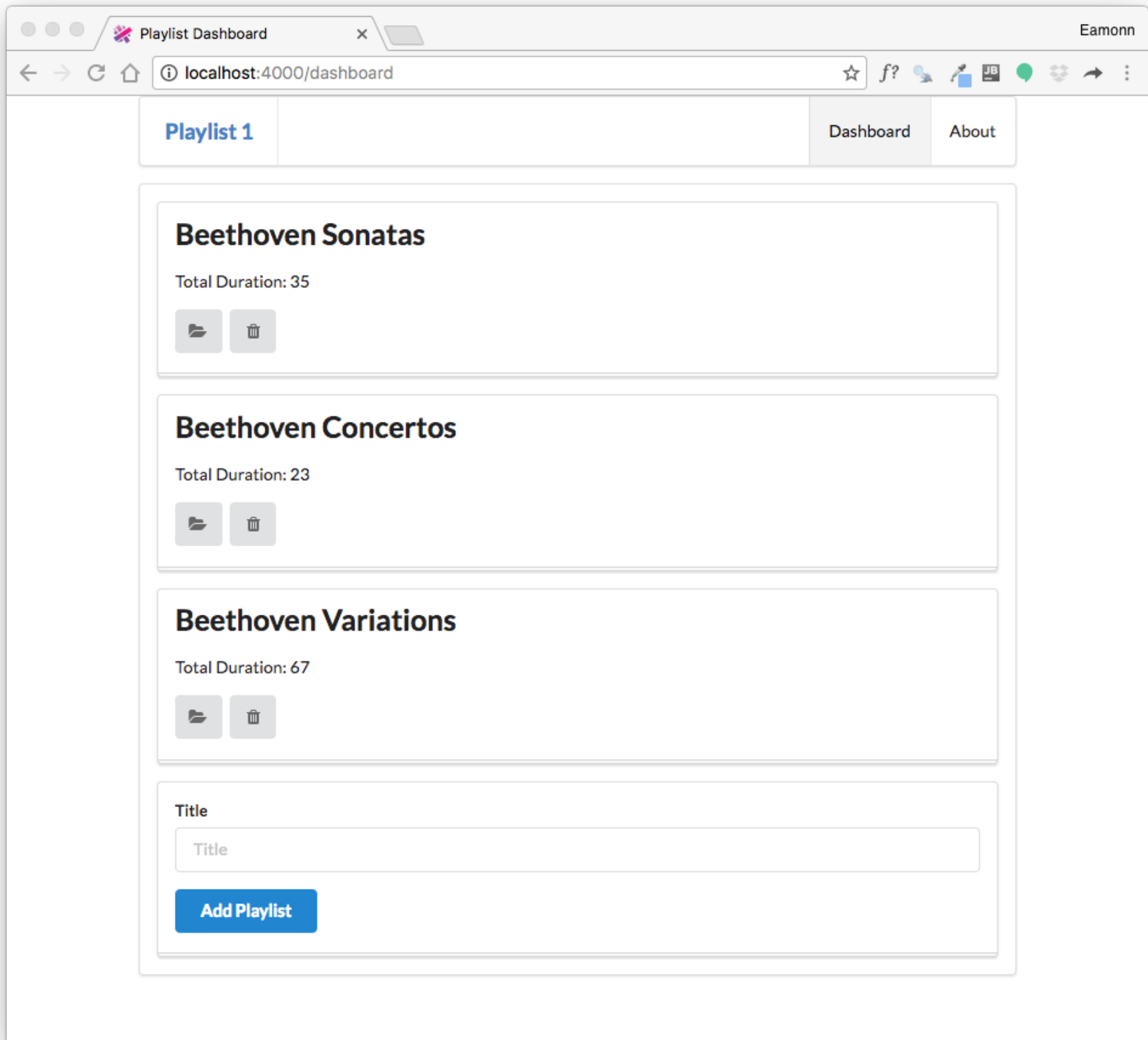
We will learn what all of this means.

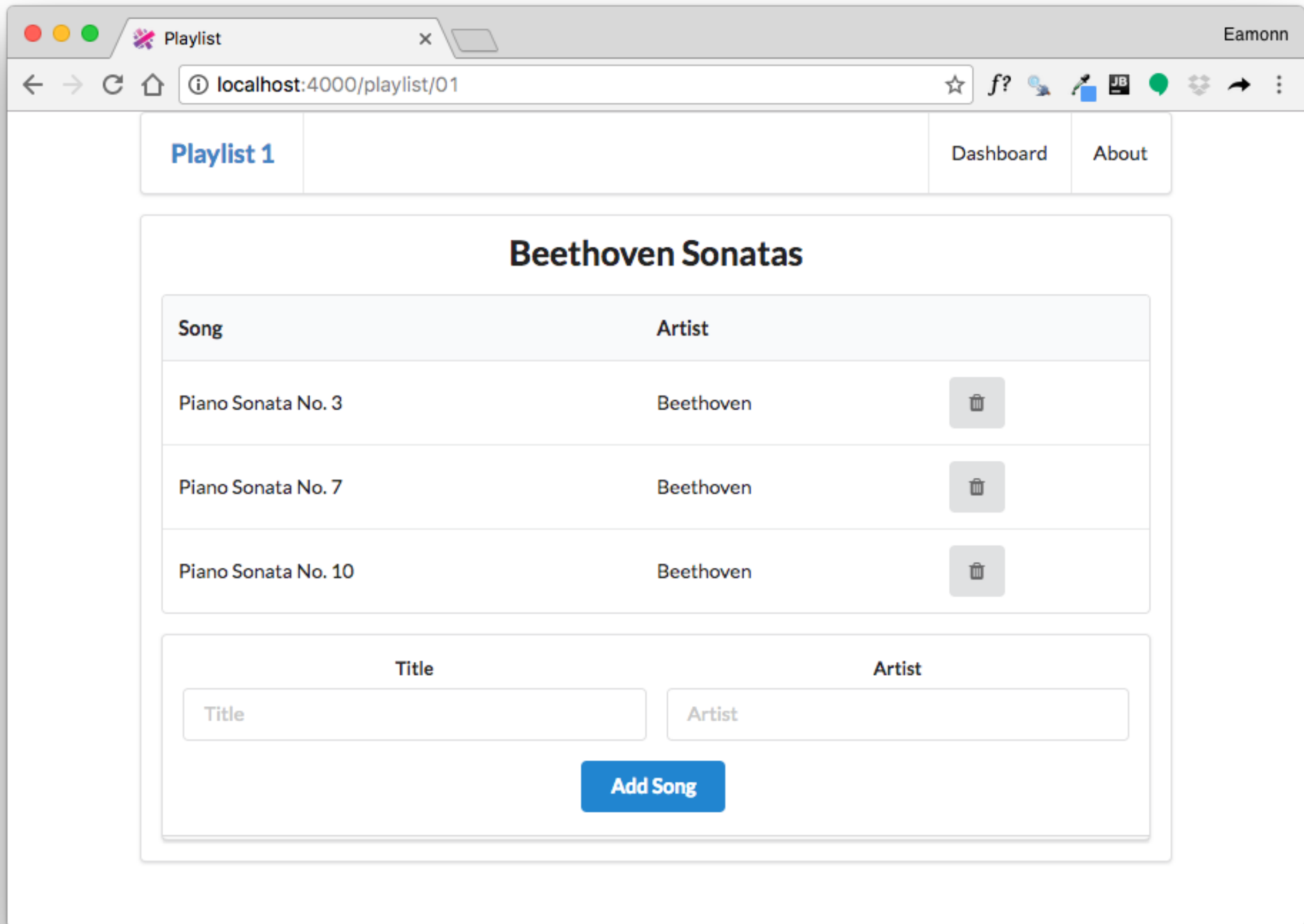
- + how to build a fully featured web app including:
 - templating
 - forms to submit information
 - How store information in models
 - create user accounts, and tie account to a each user

All of this requires beginner/
intermediate level
Javascript skills

A tour of our first app - Playlist










Playlist 1

Dashboard

About

Beethoven Sonatas

Song	Artist	
Piano Sonata No. 3	Beethoven	
Piano Sonata No. 7	Beethoven	
Piano Sonata No. 10	Beethoven	

Title

Artist

Title

Artist

Add Song

Playlist Labs

- We will do Four playlist labs in the next few sessions
 - 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.
 - Playlist 4: ability to support different users in the same application
- These labs will be interleaved with Javascript Introductory labs, which will gradually introduce you to the language