

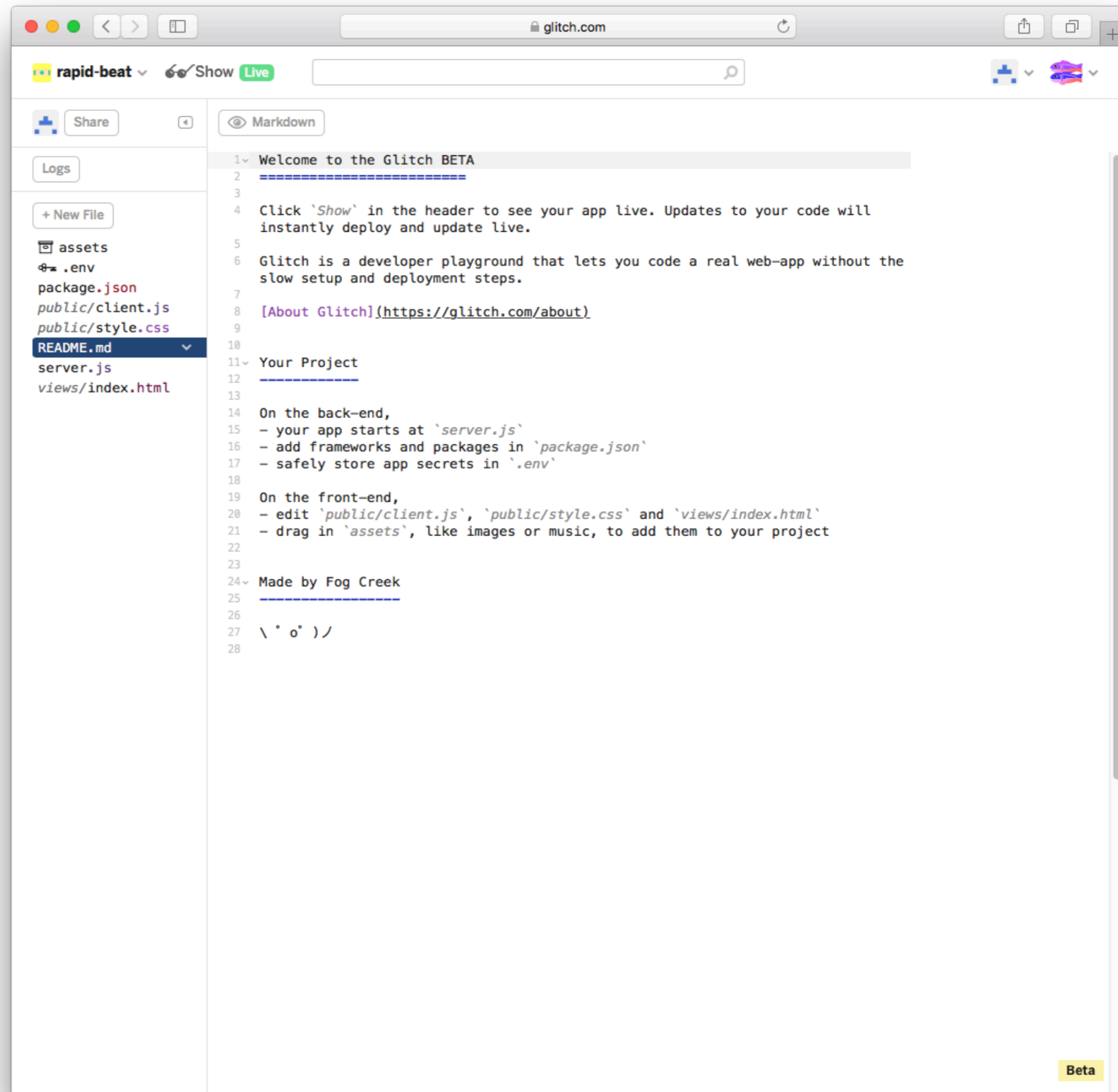
Glitch Tour

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)

The screenshot shows a Glitch project page for a project named "rapid-beat". The page is in "Live" mode, as indicated by the "Show Live" button in the top header. The left sidebar contains a file explorer with the following files: `assets`, `.env`, `package.json`, `public/client.js`, `public/style.css`, `README.md` (highlighted), `server.js`, and `views/index.html`. The main content area displays the content of the selected `README.md` file, which includes a welcome message, instructions on how to use Glitch, and a list of files to edit. The page also features a "Share" button, a "Logs" button, and a "Markdown" toggle. In the bottom right corner, there is a "Beta" badge.

```
1 Welcome to the Glitch BETA
2 =====
3
4 Click 'Show' in the header to see your app live. Updates to your code will
5 instantly deploy and update live.
6
7 Glitch is a developer playground that lets you code a real web-app without the
8 slow setup and deployment steps.
9
10 [About Glitch](https://glitch.com/about)
11
12 Your Project
13 =====
14
15 On the back-end,
16 - your app starts at `server.js`
17 - add frameworks and packages in `package.json`
18 - safely store app secrets in `.env`
19
20 On the front-end,
21 - edit `public/client.js`, `public/style.css` and `views/index.html`
22 - drag in `assets`, like images or music, to add them to your project
23
24 Made by Fog Creek
25 =====
26
27 \ ` o` )/
28
```

Link to your
Profile

Link to
Community,
resources, options

The image shows a Glitch project editor interface. On the left, a file explorer shows a project named 'rapid-beat' with files like 'assets', '.env', 'package.json', 'public/client.js', 'public/style.css', 'README.md', 'server.js', and 'views/index.html'. The 'README.md' file is selected and its content is visible in the main editor. A dark blue arrow points from the 'Show Live' button in the top header to the 'README.md' file in the file explorer. The main editor displays the following text:

```
1 Welcome to Glitch BETA
2 =====
3
4 Click 'Show' in the header to
  instantly deploy and update l
5
6 Glitch is a developer playgro
  slow setup and deployment ste
7
8 [About Glitch](https://glitch
9
10
11 Your Project
12 =====
13
14 On the back-end,
15 - your app starts at `server.js`
16 - add frameworks and packages
17 - safely store app secrets in
18
19 On the front-end,
20 - edit `public/client.js`, `pu
21 - drag in `assets`, like image
22
23
24 Made by Fog Creek
25 =====
26
27 \ ` o ` ) /
28
```

On the right, a live preview window titled 'rapid-beat.glitch.me' shows the rendered README content:

A Dream of the Future

Oh hi,

Tell me your hopes and dreams:

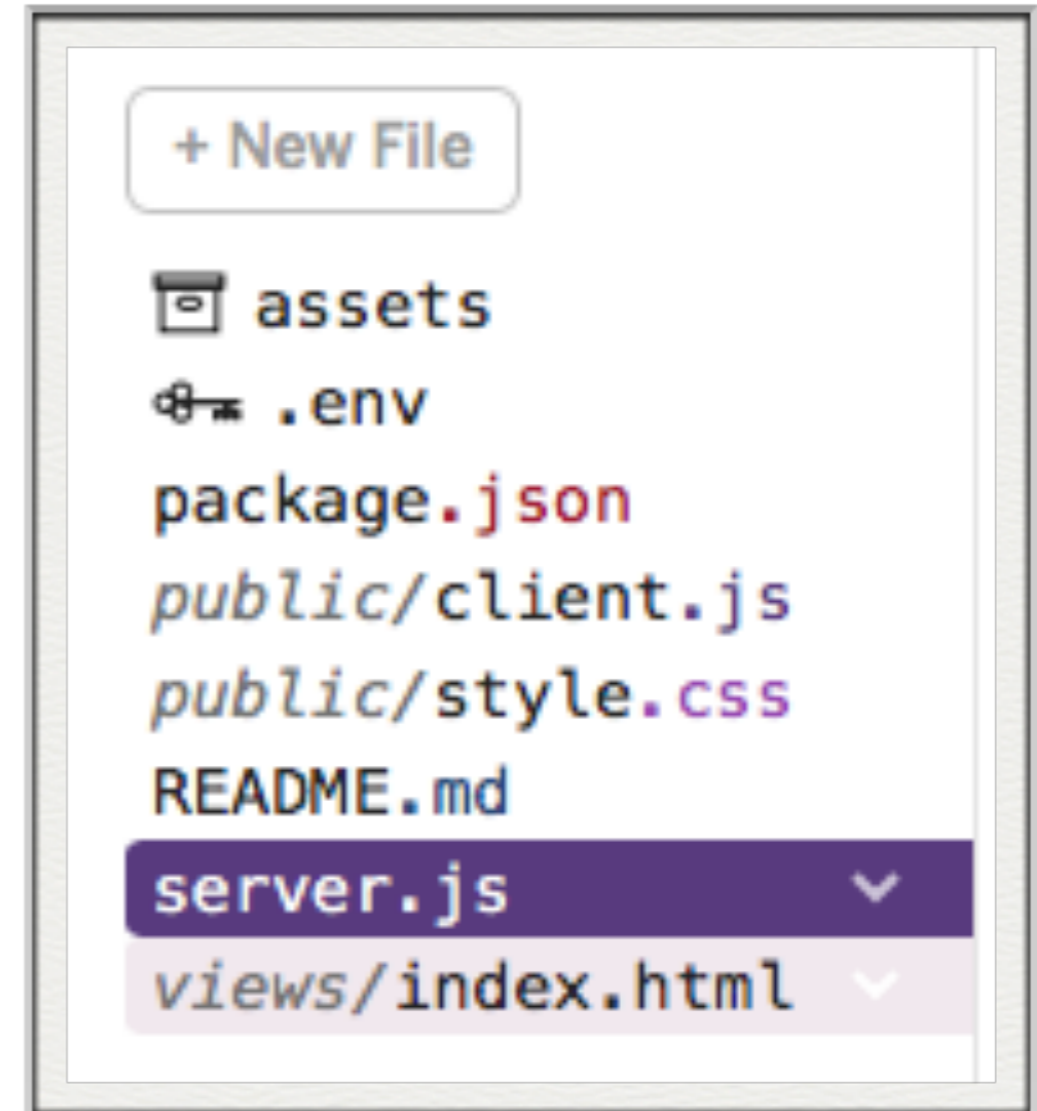
- Find and count some sheep
- Climb a really tall mountain
- Wash the dishes

[Remix this in Glitch](#)

- 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

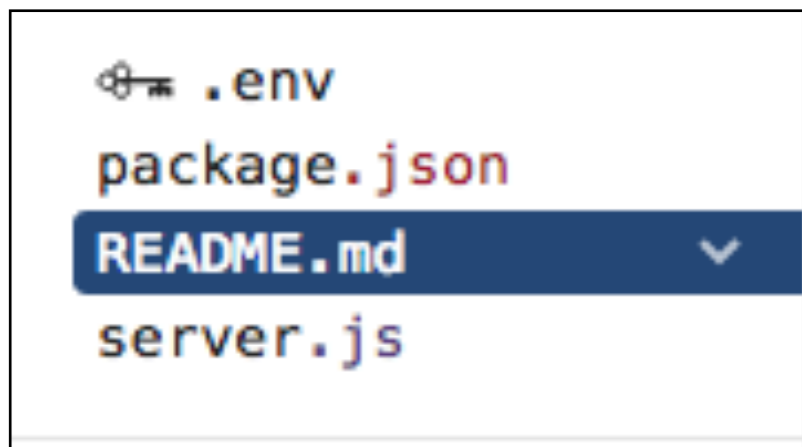


Front End

```
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.
- Many types of application supported.
- We will focus on Javascript applications written using node.js
- This is the default toolkit for Glitch - but other variants are planned.

The Starter App

index.html - Gomix

https://gomix.com/#!/project/actually-specialist

actually-specialist Show Live

Share

Logs

back-end +

- .env
- package.json
- README.md
- server.js

front-end +

- assets
- public/client.js
- public/style.css
- views/index.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: http://mozilla.github.io/nunjucks/ -->
6 <!-- jade: http://jade-lang.com/ -->
7 <!-- haml: http://haml.info/tutorial.html -->
8 <!-- hbs(handlebars): http://handlebarsjs.com/expressions.html -->
9
10 <!DOCTYPE html>
11 <html>
12 <head>
13   <title>Welcome to Gomix!</title>
14   <meta name="description" content="A cool thing made with Gomix">
15   <link id="favicon" rel="icon" href="https://gomix.com/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 </head>
21 <body>
22 <header>
23   <h1>
24     A Dream of the Future
25   </h1>
26 </header>
27
28 <main>
29   <p class="bold">Oh hi,</p>
30   <p>Tell me your hopes and dreams:</p>
31   <form>
32     <input type="text" maxlength="100" placeholder="Dreams!">
33     <button type="submit">Submit</button>
34   </form>
35   <section class="dreams">
36     <ul id="dreams">
37     </ul>
38   </section>
39 </main>
40
41 <footer>
42   <a href="https://gomix.com">
43     Remix this in Gomix
44   </a>
45 </footer>
46
47 <!-- Your web-app is https, so your scripts need to be too -->
48 <script src="https://code.jquery.com/jquery-2.2.1.min.js"
49   integrity="sha256-gvQgAFzTH6trSrAwOHiPo9Xc96QxSZ3feW6kem+000="
50   crossorigin="anonymous"></script>
51 <script src="/client.js"></script>
52
53 </body>
54 </html>
55
```

Beta

The Starter App

```
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);
```

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](#)

```
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

client side javascript

```
<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 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);
});

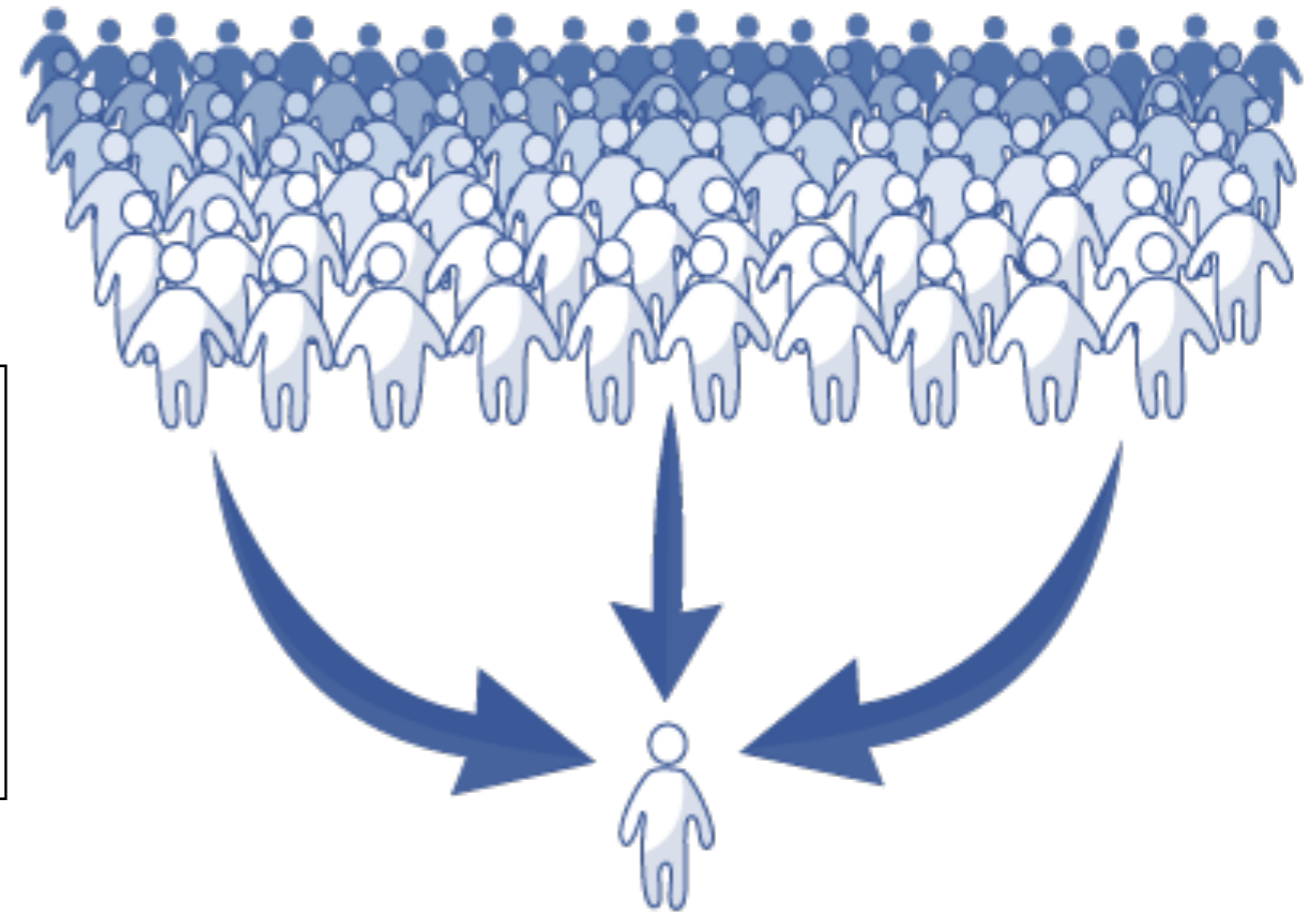
// could also use the POST body instead of query string: http://expressjs.com/en/api.html#req.body
app.post("/dreams", function (request, response) {
  dreams.push(request.query.dream);
  response.sendStatus(200);
});

// 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 in
- 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');
});

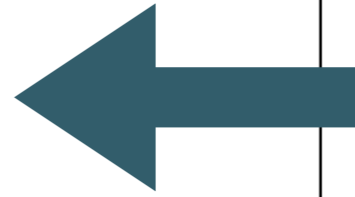
app.get("/dreams", function (request, response) {
  response.send(dreams);
});

// 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);
});

```

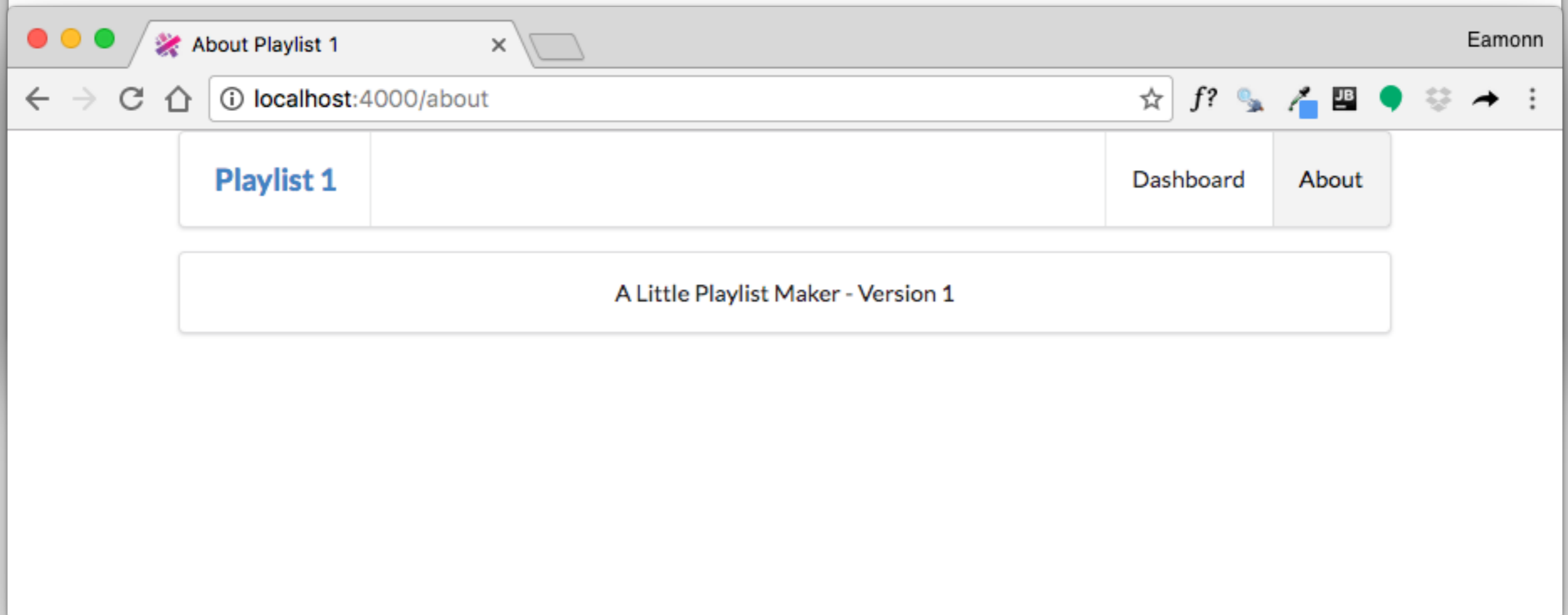
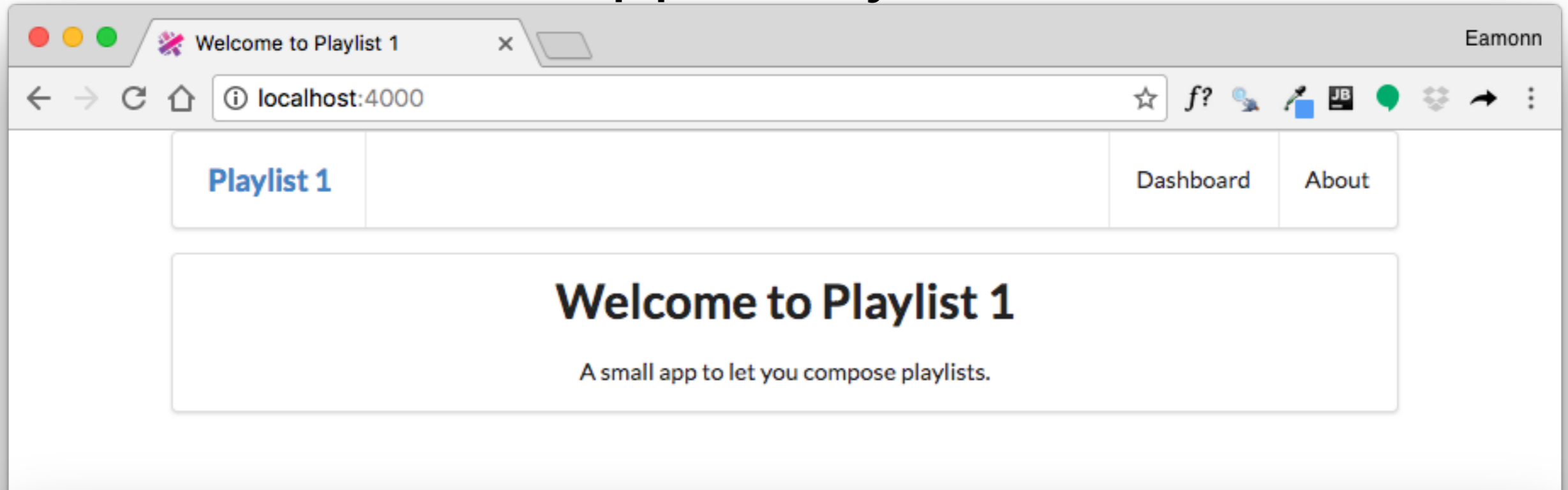


We will learn what all of this means.

- + how to build a fully featured web app including:
 - templating (like semester 1)
 - forms to submit information
 - How store information in models
 - create user accounts, and tie account to a each user

All of this requires intermediate level Javascript skills

A tour of our first app - Playlist





Playlist Dashboard x Eamonn

localhost:4000/dashboard

Playlist 1 Dashboard About



Beethoven Sonatas

Total Duration: 35



Beethoven Concertos

Total Duration: 23

Beethoven Variations

Total Duration: 67

Title

Add Playlist

Beethoven Sonatas

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

Title	Artist
<input type="text" value="Title"/>	<input type="text" value="Artist"/>
<input type="button" value="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