

Guided Tour



Building Blocks

A look at at the
components of a glitch
project. Also the types of...

Prerequisite tools on your Workstation

none!

(apart from a browser + a github account)



New Project

Starter apps

[Find More](#)



glitch-hello-website

Your very own basic web page,
ready for you to customize.



glitch-hello-node

A simple Node app built with
Fastify, instantly up and running.



glitch-hello-react

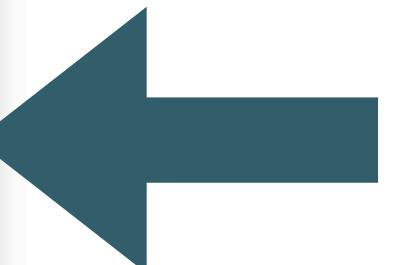
Get started with a new React
project on Glitch!



glitch-hello-eleventy

Build a new Eleventy blog on
Glitch!

[Import from GitHub](#)



- The type of project we will be working with

App Source

petite-spangle-stitch Show README.md

Share Markdown

New File ▾

assets

public/ style.css

src/ pages/ index.hbs colors.json seo.json

.env .gitignore LICENSE

README.md

package.json server.js

Hello Node

Node.js is a popular runtime that lets you run JavaScript on the server instead of in a browser. This project uses the [Fastify](#) framework to explore basic templating with [Handlebars](#) and submitting data using forms and querystrings.

What's in this project?

- ← README.md : That's this file, where you can tell people what your cool website does and how you built it.
- ← public/style.css : The styling rules for your pages and posts.
- ← server.js : The main server script for your new site.
- ← src/ : This folder holds the main template for your site along with some basic data files.

Working in the src/ folder

- ← src/pages/index.hbs : This is the main page template for your site.
- ← src/colors.json : A collection of CSS color names. We use this to pick a random color, and to match searches against color names.
- ← src/seo.json : When you're ready to share your new site or add a custom domain, change SEO/meta settings in here.

You built this with Glitch!

Glitch is a friendly community where millions of people come together to build web apps and websites.

- Need more help? [Check out our Help Center](#) for answers to any common questions.
- Ready to make it official? [Become a paid Glitch member](#) to boost your app with private sharing, more storage and memory, domains and more.

Status Tools ▾

glitch.com Change URL

Hello Node!

Now available in #0000ff!

Try another color?

Submit

Can't decide? [Use a random color!](#)

JS Gem

Remix on Glitch

Running App

Project name
(automatically generated)

Link to running app (to share)

Folders/Files in the project

Current File (editable)

The screenshot shows a Glitch project titled "petite-spangle-stitch". The left sidebar lists files and folders: assets, public/style.css, src/, pages/index.hbs, colors.json, seo.json, .env, .gitignore, LICENSE, README.md (selected), package.json, and server.js. The main content area shows the "Hello Node!" project page. It includes a "Hello Node" heading, a description of the project using Node.js and Fastify, a "What's in this project?" section with links to README.md, public/style.css, server.js, and src/, and a "Working in the src/ folder" section with links to index.hbs, colors.json, and seo.json. Below this is a "You built this with Glitch!" section about the Glitch community and links to the Help Center and成为付费会员.

Link to your Profile

Link to Community, resources, options

Glitch.com

petite-spangle-stitch Show README.md

New File Share

assets

public/ style.css

src/ pages/ index.hbs colors.json seo.json .env .gitignore LICENSE README.md

package.json server.js

In a New Window Next to The Code

README.md

← README.md : That's this file, where you can tell people what your cool website does and how you built it.

← public/style.css : The styling rules for your pages and posts.

← server.js : The main server script for your new site.

← src/ : This folder holds the main template for your site along with some basic data files.

Working in the src/ folder

← src/pages/index.hbs : This is the main page template for your site.

← src/colors.json : A collection of CSS color names. We use this to pick a color, and to match searches against color names.

← src/seo.json : When you're ready to share your new site or add a custom domain, change SEO/meta settings in here.

You built this with Glitch!

Glitch is a friendly community where millions of people come together to build apps and websites.

- Need more help? [Check out our Help Center](#) for answers to any common questions.
- Ready to make it official? [Become a paid Glitch member](#) to boost your app with private sharing, more storage and memory, domains and more.

petite-spangle-stitch.glitch.me

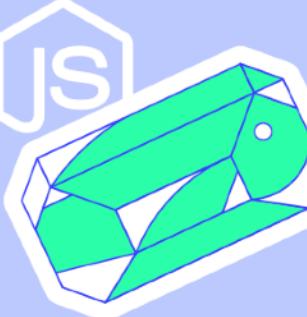
Hello Node!

Learn about communication with a server by talking about color.

What's your favorite color?

 Submit

Can't decide? [Use a random color!](#)



- 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/
        └── style.css
    └── src/
        └── pages/
            └── index.hbs
```

- 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

The screenshot shows a web browser window with the URL glitch.com. The project name is "petite-spangle-stitch". The current file is "server.js". The code is as follows:

```
1 const path = require("path");
2
3 // Require the fastify framework and instantiate it
4 const fastify = require("fastify")({
5   // set this to true for detailed logging:
6   logger: false
7 });
8
9 // Setup our static files
10 fastify.register(require("fastify-static"), {
11   root: path.join(__dirname, "public"),
12   prefix: "/" // optional: default '/'
13 });
14
15 // fastify-formbody lets us parse incoming forms
16 fastify.register(require("fastify-formbody"));
17
18 // point-of-view is a templating manager for fastify
19 fastify.register(require("point-of-view"), {
20   engine: {
21     handlebars: require("handlebars")
22   }
23 });
24
25 // load and parse SEO data
26 const seo = require("./src/seo.json");
27 if (seo.url === "glitch-default") {
28   seo.url = `https://${process.env.PROJECT_DOMAIN}.glitch.me`;
29 }
30
31 // Our home page route, this pulls from src/pages/index.hbs
32 fastify.get("/", function(request, reply) {
33   // params is an object we'll pass to our handlebars template
34   let params = { seo: seo };
35   // check and see if someone asked for a random color
36   if (request.query.randomize) {
37     // we need to load our color data file, pick one at random, and add it to the params
38     const colors = require("./src/colors.json");
39     const allColors = Object.keys(colors);
40     let currentColor = allColors[(allColors.length * Math.random()) << 0];
41     params = {
42       color: colors[currentColor],
43       colorError: null,
44       seo: seo
45     };
46   }
47   reply.view("/src/pages/index.hbs", params);
48 });
49
50 // A POST route to handle and react to form submissions
51 fastify.post("/", function(request, reply) {
52   let params = { seo: seo };
53 }
```

The Starter App

A screenshot of a web browser window showing the 'Hello Node!' application. The title 'Hello Node!' is displayed in large blue letters at the top. Below it, a sub-header reads 'Learn about communication with a server by talking about color.' A form field labeled 'What's your favorite color?' contains the placeholder text 'Type a color name here'. To its right is a 'Submit' button. Below the form is a link 'Can't decide? [Use a random color!](#)'. In the bottom right corner of the page area, there is a green diamond-shaped icon with a white 'JS' logo. At the very bottom of the page is a footer bar with a 'Remix on Glitch' button.

A screenshot of the Glitch web-based development environment. The top navigation bar shows the URL 'glitch.com' and the project name 'petite-spangle-stitch'. The left sidebar displays the project's file structure: assets, public (style.css), src (pages/index.hbs, colors.json, seo.json, .env, .gitignore, LICENSE, README.md, package.json, server.js). The main workspace shows the content of the 'server.js' file:

```
1 const path = require("path");
2
3 // Require the fastify framework and instantiate it
4 const fastify = require("fastify")({
5   // set this to true for detailed logging:
6   logger: false
7 });
8
9 // Setup our static files
10 fastify.register(require("fastify-static"), {
11   root: path.join(__dirname, "public"),
12   prefix: "/" // optional: default '/'
13 });
14
15 // fastify-formbody lets us parse incoming forms
16 fastify.register(require("fastify-formbody"));
17
18 // point-of-view is a templating manager for fastify
19 fastify.register(require("point-of-view"), {
20   engine: {
21     handlebars: require("handlebars")
22   }
23 });
24
25 // load and parse SEO data
26 const seo = require("./src/seo.json");
27 if (seo.url === "glitch-default") {
28   seo.url = `https://${process.env.PROJECT_DOMAIN}.glitch.me`;
29 }
30
31 // Our home page route, this pulls from src/pages/index.hbs
32 fastify.get("/", function(request, reply) {
33   // params is an object we'll pass to our handlebars template
34   let params = { seo: seo };
35   // check and see if someone asked for a random color
36   if (request.query.randomize) {
37     // we need to load our color data file, pick one at random, and add it to the params
38     const colors = require("./src/colors.json");
39     const allColors = Object.keys(colors);
40     let currentColor = allColors[(allColors.length * Math.random()) << 0];
41     params = {
42       color: colors[currentColor],
43       colorError: null,
44       seo: seo
45     };
46   }
47   reply.view("/src/pages/index.hbs", params);
48 });
49
50 // A POST route to handle and react to form submissions
51 fastify.post("/", function(request, reply) {
52   let params = { seo: seo };
53 }
```

The Starter App

A screenshot of a web application titled "Hello Node!". The main heading is "Hello Node!" in large blue letters. Below it is a sub-headline: "Learn about communication with a server by talking about color." A form field asks "What's your favorite color?" with a placeholder and a "Submit" button. Below the form is a link "Can't decide? [Use a random color!](#)". A green "JS" logo is centered on the page. At the bottom, there is a "Remix on Glitch" button.

A screenshot of the Glitch code editor showing the file "src/pages/index.hbs". The code is a Handlebars template for a web page. It includes meta tags for SEO and social sharing, imports the "style.css" stylesheet, and checks for a "color" parameter. The template uses Handlebars syntax to conditionally change the page's styling based on the user's input or a random color choice. The file also includes a "title" section with the text "Hello Node!" and a "color-form" section for user input.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="https://glitch.com/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>{{seo.title}}</title>
    <!-- meta tags for SEO and social sharing -->
    <link rel="canonical" href="{{seo.url}}>
    <meta name="description" content="{{seo.description}}>
    <meta property="og:title" content="{{seo.title}}>
    <meta property="og:type" content="article">
    <meta property="og:url" content="{{seo.url}}>
    <meta property="og:description" content="{{seo.description}}>
    <meta property="og:image" content="{{seo.image}}>
    <meta name="twitter:card" content="summary">
    <!-- import the webpage's stylesheet -->
    <link rel="stylesheet" href="/style.css" />
    <!-- this is a handlebars IF statement, checking for a custom "color" parameter set in server.js -->
    {{#if color}}
      <style>
        .title {
          color: {{color}};
        }
      </style>
    {{/if}}
  </head>
  <body>
    <div class="wrapper">
      <div class="content" role="main">
        <!-- this is the start of content for our page -->
        <h1 class="title">Hello Node!</h1>
        <!-- add text indicating that we've changed color OR had an error -->
        {{#if color}}
          <p class="color-info">
            Now available in <b>{{color}}</b>!
          </p>
        {{/if}}
        {{#if colorError}}
          <p class="color-info">
            Hmm. Couldn't find <b>"{{colorError}}</b>.
            <a href="?randomize=go">Try a random color?</a>
          </p>
        {{/if}}
      </div>
    </div>
    <div class="color-form">
      <!-- our default paragraph/message -->
      {{#unless colorError}}
        {{#unless color}}
          <!--
            If neither color nor colorError are present, then
            we'll just show a general message
          -->
          <p>Please enter a color below!</p>
        {{/unless}}
      {{/unless}}
    </div>
  </body>
</html>
```

Server side javascript

html

```
<div class="color-form">
  <!-- our default paragraph/message -->
  {{#unless colorError}}
    {{#unless color}}
      <!-- There are fancier ways to do this, but nesting two "#unless" statements sets up a "if these
      <p>
        Learn about communication with a server by talking about color.
      </p>
      {{/unless}}
    {{/unless}}
    <!-- the "what's your favorite color?" form sends a POST to the server, check server.js to see how
    <form class="color-search" method="post">
      <label for="color">
        <!-- we use the handlebars "#if" statement so the form can adapt to its situation -->
        {{#if color}}
          Try another color?<br>
        {{else}}
          What's your favorite color?<br>
        {{/if}}
        <input id="color" name="color" required="required" type="text">
      </label>

      <button type="submit">Submit</button>
    </form>
    <!-- this randomize link communicates with the server (server.js) using a querystring and GET req
    Can't decide? <a href="?randomize=go">Use a random color!</a>
```

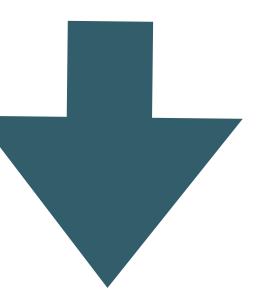
```
// Our home page route, this pulls from src/pages/index.hbs
fastify.get("/", function(request, reply) {
  // params is an object we'll pass to our handlebars template
  let params = { seo: seo };

  // check and see if someone asked for a random color
  if (request.query.randomize) {
    // we need to load our color data file, pick one at random, and add it to the params
    const colors = require("./src/colors.json");
    const allColors = Object.keys(colors);
    let currentColor = allColors[(allColors.length * Math.random()) << 0];
    params = {
      color: colors[currentColor],
      colorError: null,
      seo: seo
    };
  }
  reply.view("/src/pages/index.hbs", params);
});

// A POST route to handle and react to form submissions
fastify.post("/", function(request, reply) {
  let params = { seo: seo };

  // the request.body.color is posted with a form submission
  let color = request.body.color;
  // if it's not empty, let's try to find the color
  if (color) {
    // load our color data file
    const colors = require("./src/colors.json");
    // take our form submission, remove whitespace, and convert to lowercase
    color = color.toLowerCase().replace(/\s/g, "");
    // now we see if that color is a key in our colors object
    if (colors[color]) {
      // found one!
      params = {
        color: colors[color],
        colorError: null,
        seo: seo
      };
    } else {
      // try again.
      params = {
        colorError: request.body.color,
        seo: seo
      };
    }
  }
  reply.view("/src/pages/index.hbs", params);
});
```

Client side html runs in each users browser



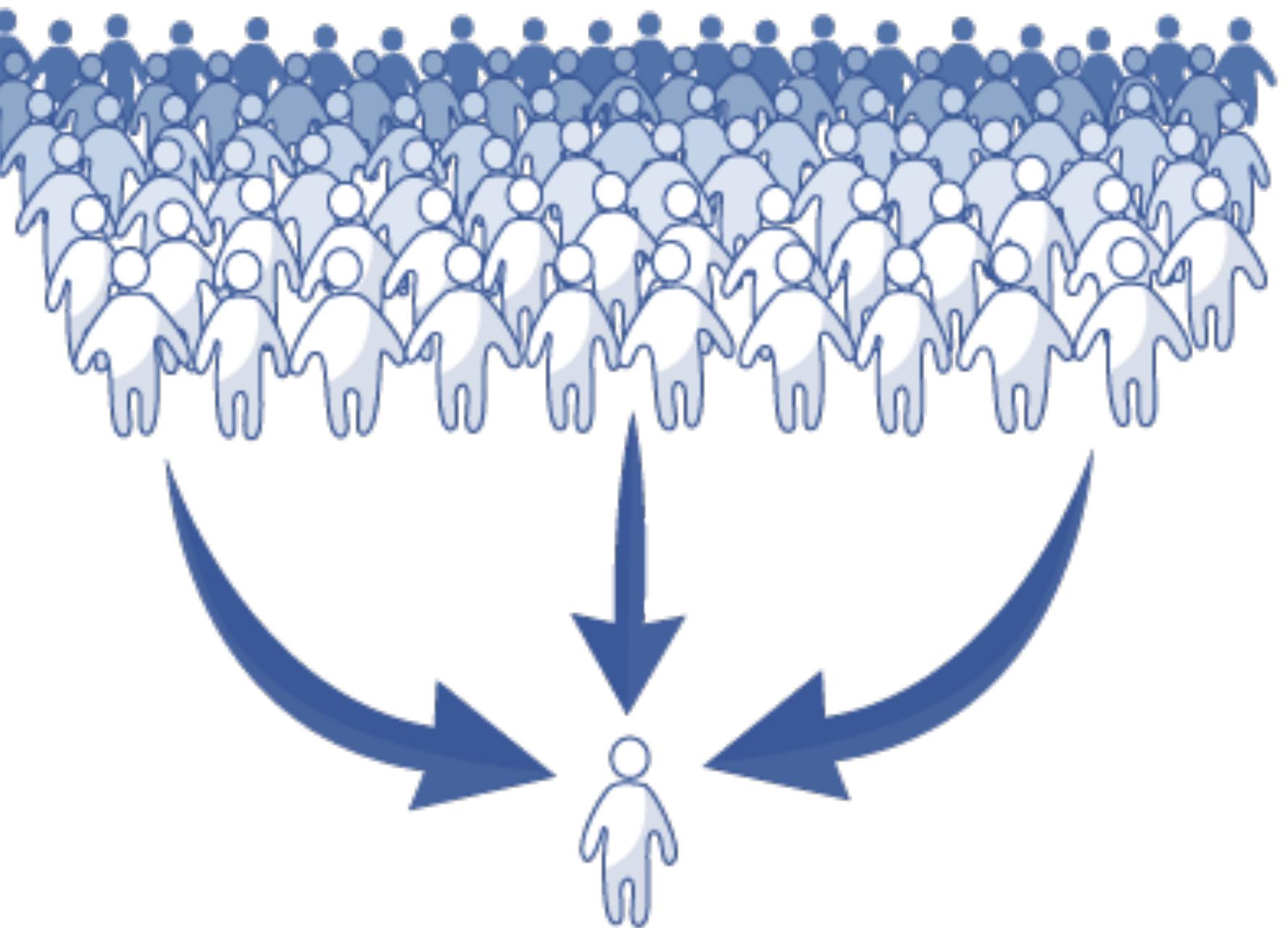
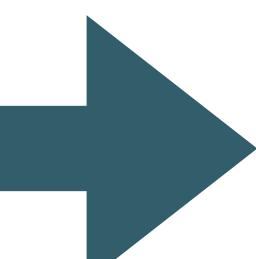
```
<div class="color-form">
  <!-- our default paragraph/message -->
  {{#unless colorError}}
    {{#unless color}}
      <!-- There are fancier ways to do this, but nesting two "#unless" statements sets up a "if these
      <p>
        Learn about communication with a server by talking about color.
      </p>
      {{/unless}}
    {{/unless}}
  <!-- the "what's your favorite color?" form sends a POST to the server, check server.js to see how
  <form class="color-search" method="post">
    <label for="color">
      <!-- we use the handlebars "#if" statement so the form can adapt to its situation -->
      {{#if color}}
        Try another color?<br>
      {{else}}
        What's your favorite color?<br>
      {{/if}}
      <input id="color" name="color" required="required" type="text">
    </label>

    <button type="submit">Submit</button>
  </form>

  <!-- this randomize link communicates with the server (server.js) using a querystring and GET request
  Can't decide? <a href="?randomize=q0">Use a random color!</a>

```

Server side run on a cloud server. All browsers connected to this node



```
// A POST route to handle and react to form submissions
fastify.post("/", function(request, reply) {
  let params = { seo: seo };
  // the request.body.color is posted with a form submission
  let color = request.body.color;
  // if it's not empty, let's try to find the color
  if (color) {
    // load our color data file
    const colors = require("./src/colors.json");
    // take our form submission, remove whitespace, and convert to lowercase
    color = color.toLowerCase().replace(/\s/g, "");
    // now we see if that color is a key in our colors object
    if (colors[color]) {
      // found one!
      params = {
        color: colors[color],
        colorError: null,
        seo: seo
      };
    } else {
      // try again.
      params = {
        colorError: request.body.color,
        seo: seo
      };
    }
  }
});
```

Skills for this Course

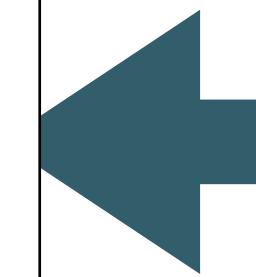
- Assumptions:
 - Foundation Knowledge in HTML + CSS
 - Working knowledge of Semantic UI CSS Framework
 - Familiarity with Play Framework + Todo & Playlist applications
- Major focus of this course:
 - Back end Javascript Programming
 - Node.js Web Application Development
 - Glitch is the platform + WebStorm (sibling of IDEA) as alternative
 - Front end javascript development will **not** be covered.

```

// Our home page route, this pulls from src/pages/index.hbs
fastify.get("/", function(request, reply) {
  // params is an object we'll pass to our handlebars template
  let params = { seo: seo };
  // check and see if someone asked for a random color
  if (request.query.randomize) {
    // we need to load our color data file, pick one at random, and add it to the params
    const colors = require("./src/colors.json");
    const allColors = Object.keys(colors);
    let currentColor = allColors[(allColors.length * Math.random()) << 0];
    params = {
      color: colors[currentColor],
      colorError: null,
      seo: seo
    };
  }
  reply.view("/src/pages/index.hbs", params);
});

// A POST route to handle and react to form submissions
fastify.post("/", function(request, reply) {
  let params = { seo: seo };
  // the request.body.color is posted with a form submission
  let color = request.body.color;
  // if it's not empty, let's try to find the color
  if (color) {
    // load our color data file
    const colors = require("./src/colors.json");
    // take our form submission, remove whitespace, and convert to lowercase
    color = color.toLowerCase().replace(/\s/g, "");
    // now we see if that color is a key in our colors object
    if (colors[color]) {
      // found one!
      params = {
        color: colors[color],
        colorError: null,
        seo: seo
      };
    } else {
      // try again.
      params = {
        colorError: request.body.color,
        seo: seo
      };
    }
  }
  reply.view("/src/pages/index.hbs", params);
});

```



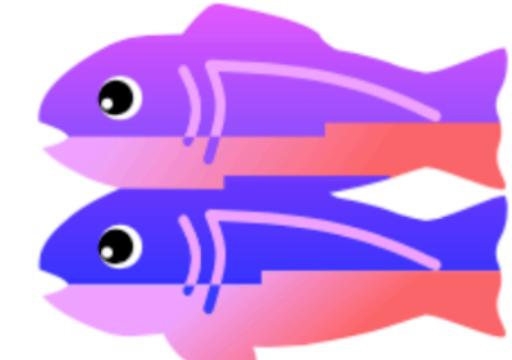
- We will learn what all of this means.
- + how to build a fully featured web app including:
 - templating
 - forms to submit information
 - How store information in models
 - create user accounts, and tie account to a each user

All of this requires beginner/intermediate level
Javascript skills

Preview of Course Labs

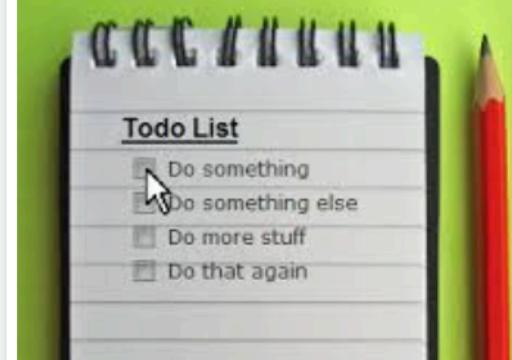
- Introductory
- Javascript
- Playlist
- Todolist

Lab-00a Glitch Intro



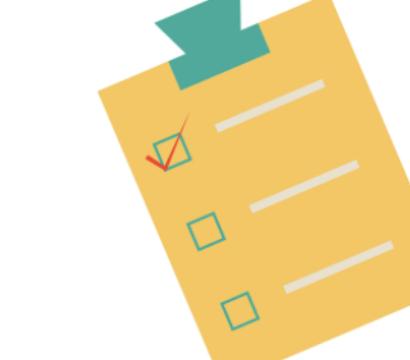
Create, modify and view your first Glitch project.

Lab-00b Todo 1



Create a Todo application in Glitch

Lab-00c Todo 2



Incorporate IDs into the todo model

Lab-00d Todo 3



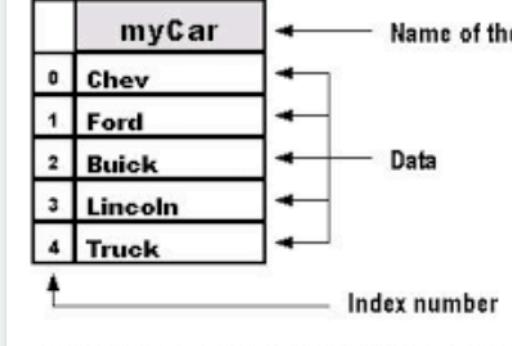
Incorporate Sessions / User accounts into Todo

Lab-01 JS Intro



Background & Tools, Variables & Boolean Logic

Lab-02 JS Arrays



Comparison of an array to a column of data

Array Basics, Array Methods & Iteration

Lab-03a Playlist 1



Import and run a new starter project. Extend this project to include multiple..

Lab-03b WebStorm



Using WebStorm instead of Glitch for Web App development

Lab-04 Playlist 2



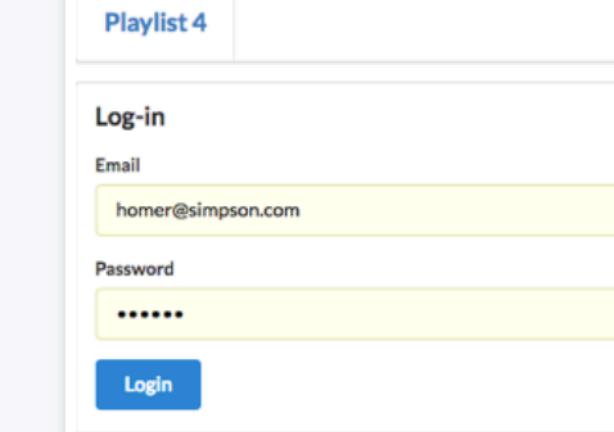
Refactor the dashboard controller to show summary on of the playlists + link to..

Lab-05 Playlist 3



Enable Songs and Playlists to be added via simple forms.

Lab-06a Playlist 4



Log-in

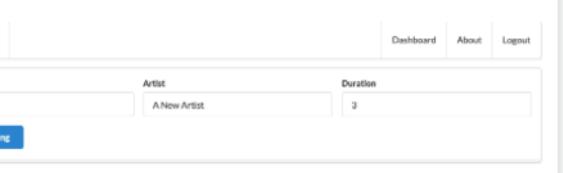
Email: homer@simpson.com

Password: *****

Login

Introduce Sessions onto the Playlist application, enabling user accounts a...

Lab-06b Playlist 5



Title: My Song

Artist: A New Artist

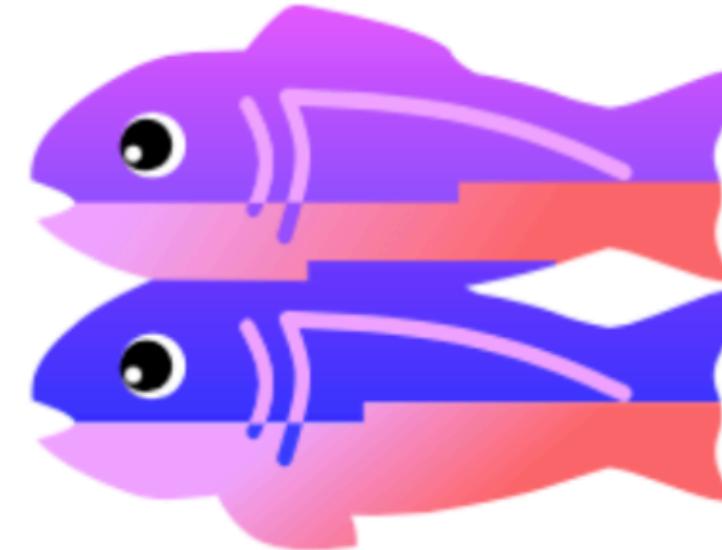
Duration: 2

Update Song

Implement a feature to update an existing Song

Introductory

Lab-00a Glitch Intro



Create, modify and view
your first Glitch project.

Exploring the
Look & Feel of
Glitch

Lab-00b Todo 1



Create a Todo application in
Glitch

Lab-00c Todo 2



Incorporate IDs into the
todo model

Lab-00d Todo 3



Incorporate Sessions / User
accounts into Todo

Advance preview of course. Recreate the play todo
list applications in Javascript Glitch.
Feel free to Skip these three labs (for the moment)!

Javascript

Accelerated
introducing to
Javascript

Explore a subset
of the language
suitable for our
purposes

NOT using glitch
for these labs -
just a web browser

Lab-01 JS Intro



Background & Tools,
Variables & Boolean Logic

Lab-02 JS Arrays

myCar	
0	Chev
1	Ford
2	Buick
3	Lincoln
4	Truck

← Name of the array
← Data
↑ Index number

Comparison of an array to a column of data

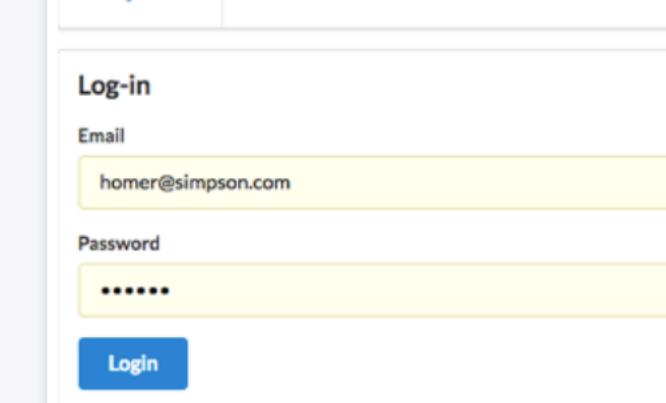
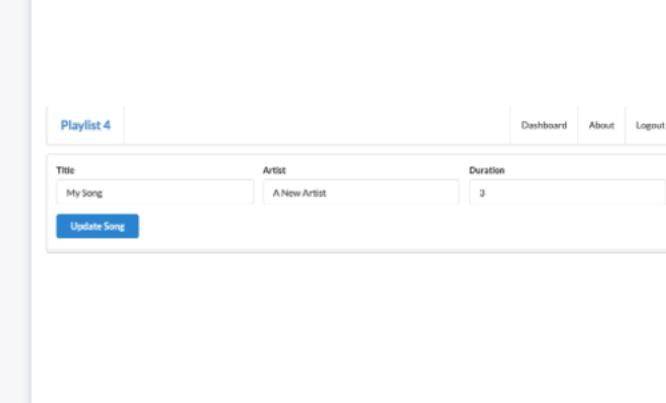
Array Basics, Array
Methods & Iteration

Playlist

Rebuild the
Playlist
Applications

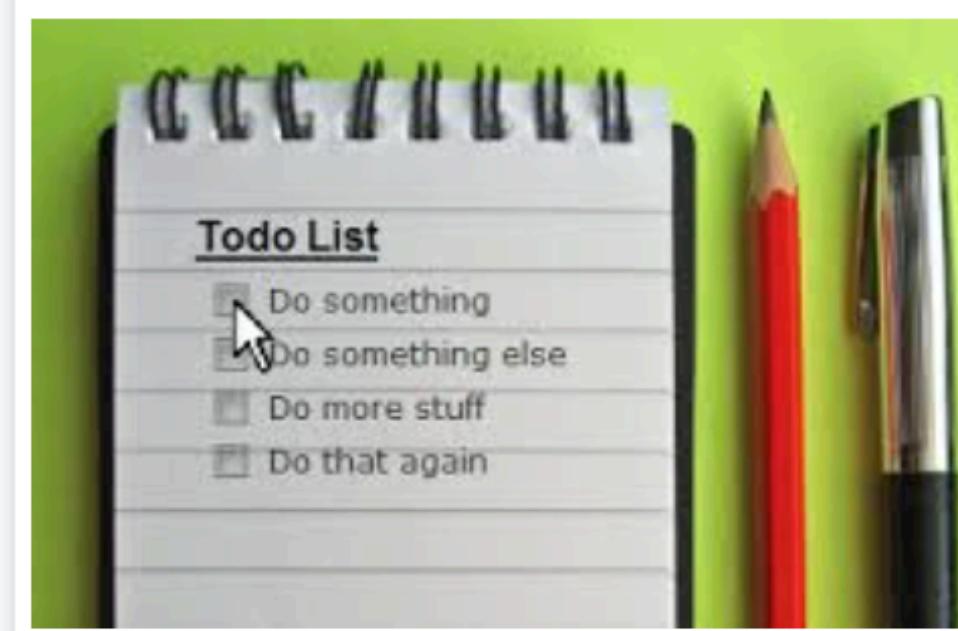
Identical features
from Play/Java

Implemented in
Javascript/
Express

<p>Lab-03a Playlist 1</p>  <p>Import and run a new starter project. Extend this project to include multiple..</p>	<p>Lab-03b WebStorm</p>  <p>Using WebStorm instead of Glitch for Web App development</p>	<p>Lab-04 Playlist 2</p>  <p>Refactor the dashboard controller to show summary on of the playlists + link to..</p>
<p>Lab-05 Playlist 3</p>  <p>Enable Songs and Playlists to be added via simple forms.</p>	<p>Lab-06a Playlist 4</p>  <p>Introduce Sessions onto the Playlist application, enabling user accounts a...</p>	<p>Lab-06b Playlist 5</p>  <p>Implement a feature to update an existing Song</p>

Todolist

Lab-00b Todo 1



Create a Todo application in
Glitch

Lab-00c Todo 2



Incorporate IDs into the
todo model

Lab-00d Todo 3



Incorporate Sessions / User
accounts into Todo

Do/Redo these labs as a revision of the
core concepts

Guided Tour



Building Blocks

A look at at the components of a glitch project. Also the types of...