

Guided Tour



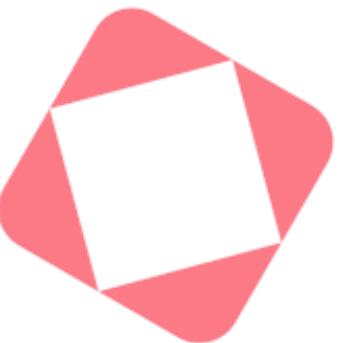
Building Blocks

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)



Sign In to Glitch

Facebook

GitHub

Google

Email Magic Link

Password

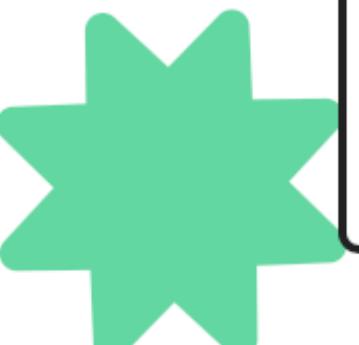
By signing into Glitch, you agree to our [Terms of Services](#) and [Privacy Statement](#)



Sign via
Github
Account

Don't have an account?

Create an account



The screenshot shows the Glitch website's 'Starter apps' section. At the top, there is a search icon and a 'New Project' button. Below this, the heading 'Starter apps' is followed by a 'Find More' link. The first item listed is 'glitch-hello-website', which has a small icon of a globe and a blue link. Its description is: 'Your very own basic web page, ready for you to customize.' The second item, 'glitch-hello-node', is highlighted with a large blue background and a dark blue arrow pointing to it from the left. Its icon is a green terminal window, and its description is: 'A simple Node app built with Fastify, instantly up and running.' The third item is 'glitch-hello-react', with a pink React logo icon, described as: 'Get started with a new React project on Glitch!' The fourth item is 'glitch-hello-eleventy', with a blue 11ty logo icon, described as: 'Build a new Eleventy blog on Glitch!' At the bottom of the list is a button labeled 'Import from GitHub'.

- The type of project we will be working with

App Source

The screenshot shows a Glitch project titled "Hello Node!". The interface includes a sidebar with file navigation, a main content area with project details and a "What's in this project?" section, and various status and tool tabs at the bottom.

Glitch README.md ⌂ Share

Your brand new node.js app uses server-side JavaScript to create a live full-stack application that updates instantly. Check out the [TODO](#) for some first edits to try out!

robust-spiny-cheque README.md EDIT MARKDOWN

Settings Assets Files

public/ src/ pages/ index.hbs colors.json seo.json .gitignore LICENSE README.md TODO.md package.json server.js

Hello Node!

This project includes a Node.js server script and a web page that connects to it. The front-end page presents a form the visitor can use to submit a color name, sending the submitted value to the back-end API running on the server. The server returns info to the page that allows it to update the display with the chosen color. 🎨

[Node.js](#) is a popular runtime that lets you run server-side JavaScript. This project uses the [Fastify](#) framework and explores basic templating with [Handlebars](#).

Prerequisites

You'll get best use out of this project if you're familiar with basic JavaScript. If you've written JavaScript for client-side web pages this is a little different because it uses server-side JS, but the syntax is the same!

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 the pages in your site.
- ← server.js : The [Node.js](#) server script for your new site. The JavaScript defines the endpoints in the site back-end, one to return the homepage and one to update with the submitted color. Each one sends data to a Handlebars template which builds these parameter values into the web page the visitor sees.
- ← package.json : The NPM packages for your project's dependencies.
- ← src/ : This folder holds the site template along with some basic data files.

New

STATUS LOGS TERMINAL TOOLS PREVIEW

Project name
(automatically
generated)

Folders/Files in
the project

Current File
(editable)

Link to running
app (to share)

Link to your
Profile

Link to
Community,
resources,
options

The screenshot shows a Glitch project titled "Hello Node!". The interface includes a sidebar with project files like README.md, public/style.css, server.js, package.json, and others. A preview pane shows the live application. A context menu is open over the package.json file, with options like "Open preview pane" and "Preview in a new window". The main content area contains sections about the project's purpose, prerequisites, and what's included.

Glitch

README.md

Your brand new node.js app uses server-side JavaScript to create a live full-stack application that updates instantly. Check out the [TODO](#) for some first edits to try out!

robust-spiny-cheque

Settings

Assets

Files

+ public/
src/
 pages/
 index.hbs
 colors.json
 seo.json
 .gitignore
LICENSE
README.md
TODO.md
package.json

README.md EDIT MARKDOWN

Hello Node!

This project includes a Node.js server script and a web page that connects to it. The front-end page presents a form the visitor can use to submit a color name, sending the submitted value to the back-end API running on the server. The server returns info to the page that allows it to update the display with the chosen color.

[Node.js](#) is a popular runtime that lets you run server-side JavaScript. This project uses the [Fastify](#) framework and explores basic templating with [Handlebars](#).

Prerequisites

You'll get best use out of this project if you're familiar with basic JavaScript. If you've written JavaScript for client-side web pages this is a little different because it uses server-side JS, but the syntax is the same!

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 the pages in your site.
- ← server.js : The [Node.js](#) server script for your new site. The JavaScript defines the endpoints in the site back-end, one to return the homepage and one to update with the submitted color. Each one sends data to a Handlebars template which builds these parameter values into the web page the visitor sees.
- ← Package.json : The [NPM](#) packages for your project's dependencies.
- ← src/ : The source code for your project, along with some basic data files.

New

Open preview pane

Preview in a new window

TOOLS PREVIEW

The screenshot shows a dual-browser setup. The left browser window is the Glitch project editor for a 'Hello Node!' project. It displays the file structure on the left, including 'public/' and 'src/' directories containing files like 'index.hbs', 'colors.json', 'seo.json', '.gitignore', 'LICENSE', 'README.md', 'TODO.md', 'package.json', and 'server.js'. The main pane shows the 'README.md' content, which includes a heading 'Hello Node!', a color input field, and a 'Submit' button. The right browser window shows the live preview of the project at 'robust-spiny-cheque.glitch.me', featuring a large 'Hello Node!' title, a color input field, and a 'Submit' button. A sidebar on the right of the preview window contains a 'Glitch' logo and navigation links.

Your brand new node.js app uses server-side Java instantly. Check out the [TODO](#) for some first edits.

robust-spiny-cheque

Settings

Assets

Files

> `public/`

< `src/`

< `pages/`

`index.hbs`

`colors.json`

`seo.json`

`.gitignore`

`LICENSE`

`README.md`

`TODO.md`

`package.json`

`server.js`

README.md

Hello Node!

This project includes a front-end page presented with a submitted value to the page that allows you to change the color of the page.

`Node.js` is a popular choice for server-side development, using the `Fastify` framework.

Prerequisites

You'll get best use of this project if you're familiar with written JavaScript for client-side web development or server-side JS, but it's not required.

What's in this project

- ← `README.md` : The file you're reading now, and how you built it.
- ← `public/style.css` : The CSS for the page.
- ← `server.js` : The code that handles the endpoints in the project, including the submitted color.
- ← `package.json` : The NPM packages for your project's dependencies.
- ← `readme.html` : A template along with some basic styling.

Using this project

This is the Glitch **Hello Node!** project. You can use it to build your own app. Check out `README.md` in the editor for more info and next steps you can take!

Open preview pane

Preview in a new window

STATUS LOGS TERMINAL TOOLS PREVIEW

glitch.com

robust-spiny-cheque.glitch.me

Hello Node!

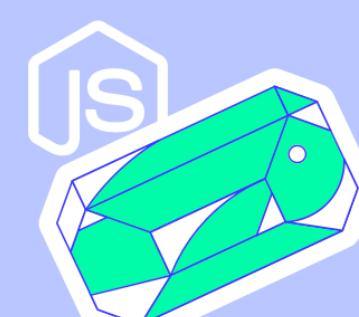
Hello Node!

Learn about communication with a server by talking about color.

What's your favorite color?

Submit

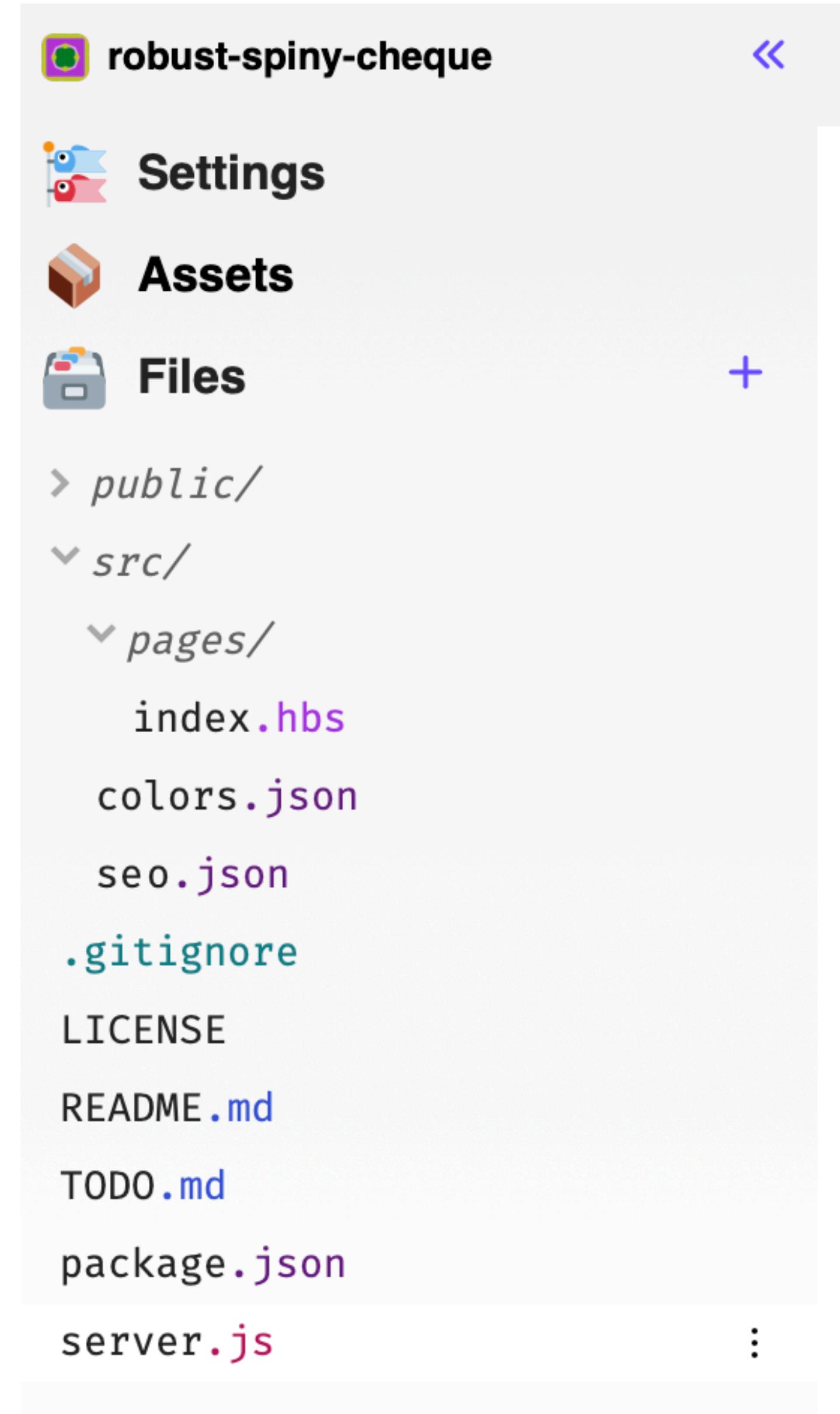
Can't decide? [Use a random color!](#)



- 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



- 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 the Glitch web interface with the project 'robust-spiny-cheque' open. The main view displays the contents of the 'server.js' file, which is a Node.js script using the Fastify framework. The file includes comments explaining its purpose, such as defining endpoints for a back-end API and setting up static files. The interface also shows a sidebar with project files like 'public', 'src' (containing 'pages', 'colors.json', 'seo.json', '.gitignore', 'LICENSE', 'README.md', 'TODO.md', and 'package.json'), and a preview window showing the application's output.

```
1  /**
2   * This is the main Node.js server script for your project
3   * Check out the two endpoints this back-end API provides in fastify.get and fastify.post below
4   */
5
6  const path = require("path");
7
8  // Require the fastify framework and instantiate it
9  const fastify = require("fastify")({
10    // Set this to true for detailed logging:
11    logger: false,
12  });
13
14 // ADD FAVORITES ARRAY VARIABLE FROM TODO HERE
15
16 // Setup our static files
17 fastify.register(require("@fastify/static"), {
18  root: path.join(__dirname, "public"),
19  prefix: "/", // optional: default '/'
20});
21
22 // Formbody lets us parse incoming forms
23 fastify.register(require("@fastify/formbody"));
24
25 // View is a templating manager for fastify
26 fastify.register(require("@fastify/view"), {
27  engine: {
28    handlebars: require("handlebars"),
29  },
30});
31
32 // Load and parse SEO data
33 const seo = require("./src/seo.json");
34 if (seo.url === "glitch-default") {
35  seo.url = `https://${process.env.PROJECT_DOMAIN}.glitch.me`;
36}
37
38 /**
```

The Starter App

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!](#)

JS

Using this project

This is the Glitch [Hello Node](#) project. You can use it to build your own app. Check out README.md in the editor for more info and next steps you can take!

Remix on Glitch

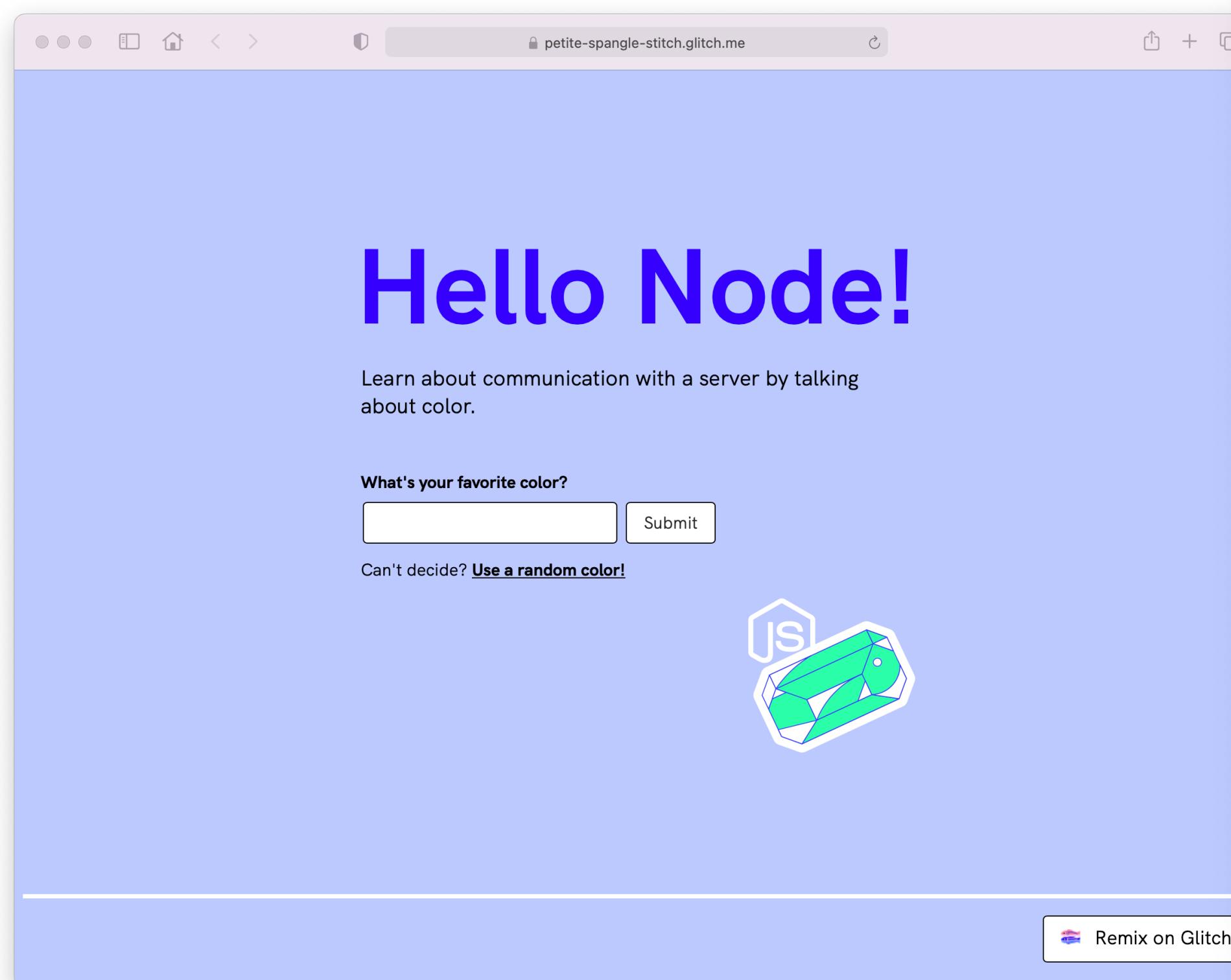
petite-spangle-stitch Show server.js

New File ↴

Format This 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



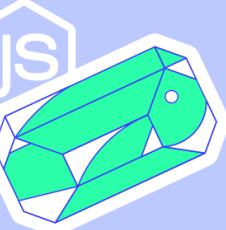
Hello Node!

Learn about communication with a server by talking about color.

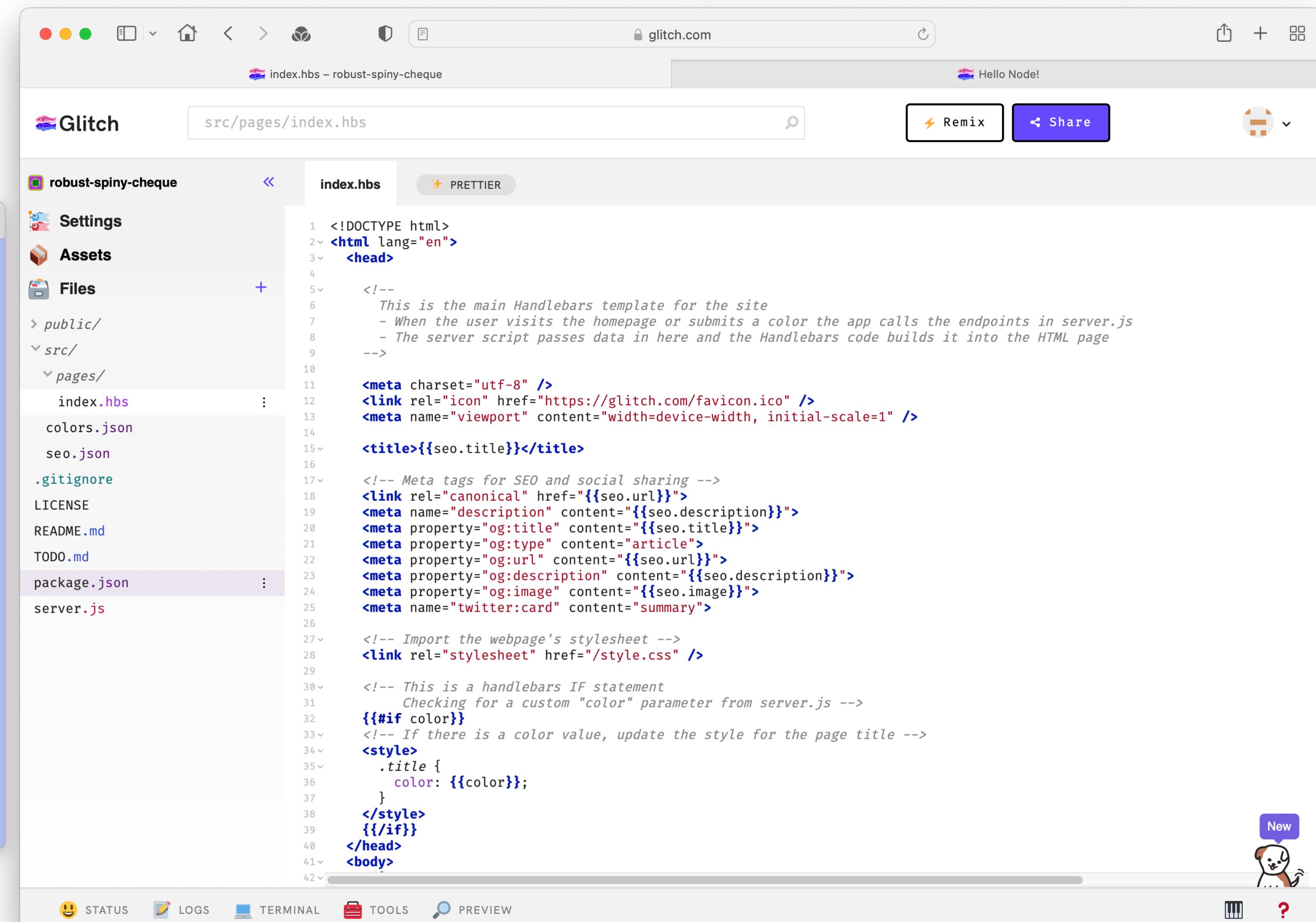
What's your favorite color?

 Submit

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



Remix on Glitch



Glitch

src/pages/index.hbs

robust-spiny-cheque index.hbs PRETTIER

Settings Assets Files

public/ src/ pages/

index.hbs colors.json seo.json .gitignore LICENSE README.md TODO.md package.json server.js

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4
5   <!--
6     This is the main Handlebars template for the site
7     - When the user visits the homepage or submits a color the app calls the endpoints in server.js
8     - The server script passes data in here and the Handlebars code builds it into the HTML page
9   -->
10
11   <meta charset="utf-8" />
12   <link rel="icon" href="https://glitch.com/favicon.ico" />
13   <meta name="viewport" content="width=device-width, initial-scale=1" />
14
15   <title>{{seo.title}}</title>
16
17   <!-- Meta tags for SEO and social sharing -->
18   <link rel="canonical" href="{{seo.url}}>
19   <meta name="description" content="{{seo.description}}>
20   <meta property="og:title" content="{{seo.title}}>
21   <meta property="og:type" content="article">
22   <meta property="og:url" content="{{seo.url}}>
23   <meta property="og:description" content="{{seo.description}}>
24   <meta property="og:image" content="{{seo.image}}>
25   <meta name="twitter:card" content="summary">
26
27   <!-- Import the webpage's stylesheet -->
28   <link rel="stylesheet" href="/style.css" />
29
30   <!-- This is a handlebars IF statement
31     Checking for a custom "color" parameter from server.js -->
32   {{#if color}}
33     <!-- If there is a color value, update the style for the page title -->
34     <style>
35       .title {
36         color: {{color}};
37       }
38     </style>
39   {{/if}}
40   </head>
41   <body>
```

New

STATUS LOGS TERMINAL TOOLS PREVIEW

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 an
          "if these are both missing..." condition -->
      <p>
        Learn about communication with a server by talking about color.
      </p>
      {{/unless}}
    {{/unless}}

    <!-- The "What's your favorite color?" form sends a POST request to the server
        Check server.js to see how it works -->
    <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>

      <!-- If the user submits a value through the form
          it'll be passed to the server in the request body -->
      <button type="submit">Submit</button>
    </form>

    <!-- This randomize link communicates with the server (server.js)
        using a querystring and GET request instead of a form POST -->
    Can't decide? <a href="?randomize=go">Use a random color!</a>
  </div>
```

```
/***
 * Our POST route to handle and react to form submissions
 */
* Accepts body data indicating the user choice
*/
fastify.post("/", function (request, reply) {
  // Build the params object to pass to the template
  let params = { seo: seo };

  // If the user submitted a color through the form it'll be passed here in the request body
  let color = request.body.color;

  // If it's not empty, let's try to find the color
  if (color) {
    // ADD CODE FROM TODO HERE TO SAVE SUBMITTED FAVORITES

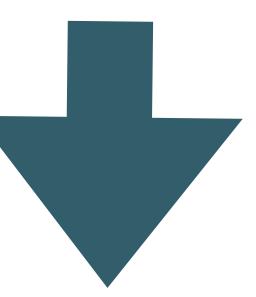
    // Load our color data file
    const colors = require("./src/colors.json");

    // 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 {
      // No luck! Return the user value as the error property
      params = {
        colorError: request.body.color,
        seo: seo,
      };
    }
  }

  // The Handlebars template will use the parameter values to update the page with the chosen color
  return reply.view("/src/pages/index.hbs", params);
});
```

Client side html runs in each users browser



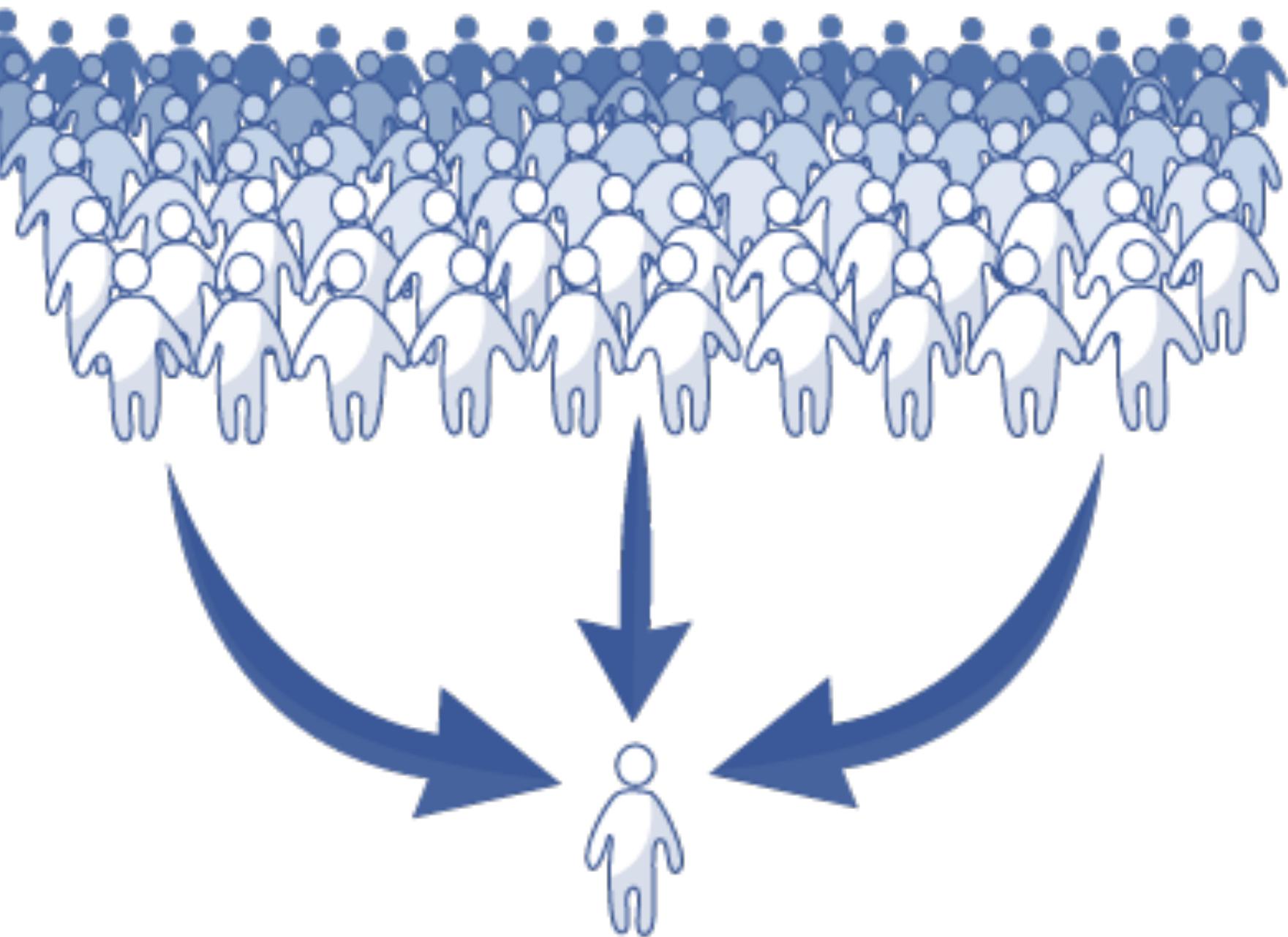
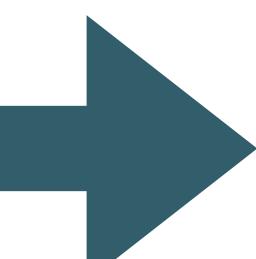
```
<div class="color-form">
  <!-- our default paragraph/message -->
  {{#unless colorError}}
    {{#unless color}}
      <!-- There are fancier ways to do this, but nesting two "#unless" statements sets up a "if these
      <p>
        Learn about communication with a server by talking about color.
      </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 Module

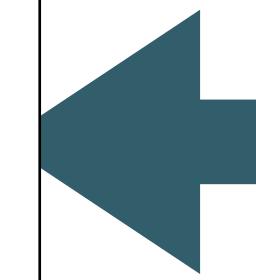
- Assumptions:
 - Foundation Knowledge in HTML + CSS
 - Working knowledge of Semantic UI CSS Framework
 - Familiarity with Play Framework + Todo & Playlist applications
- Major focus of this course:
 - Back end Javascript Programming
 - Node.js Web Application Development
 - Glitch is the platform + 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);
});

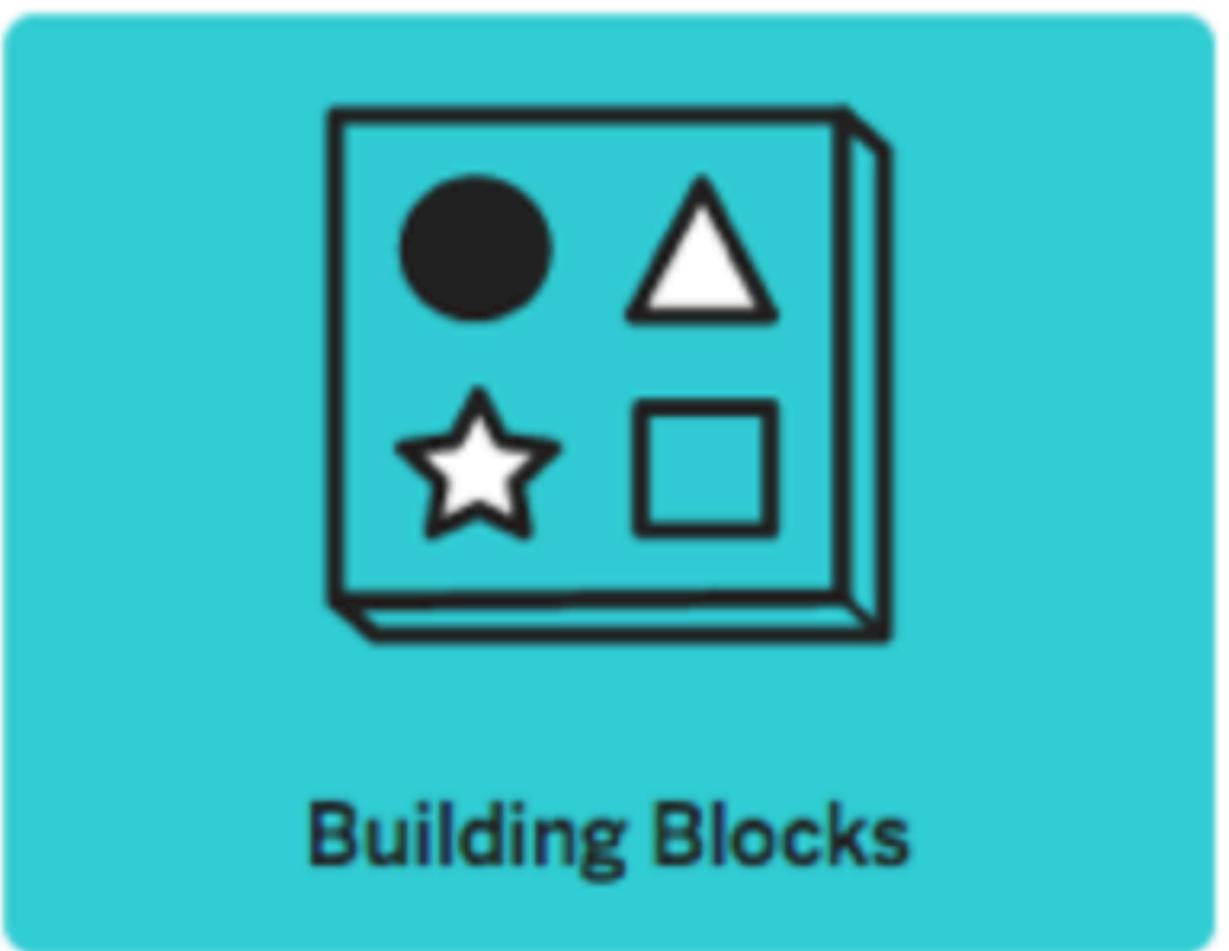
```



- 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

Guided Tour



Building Blocks

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