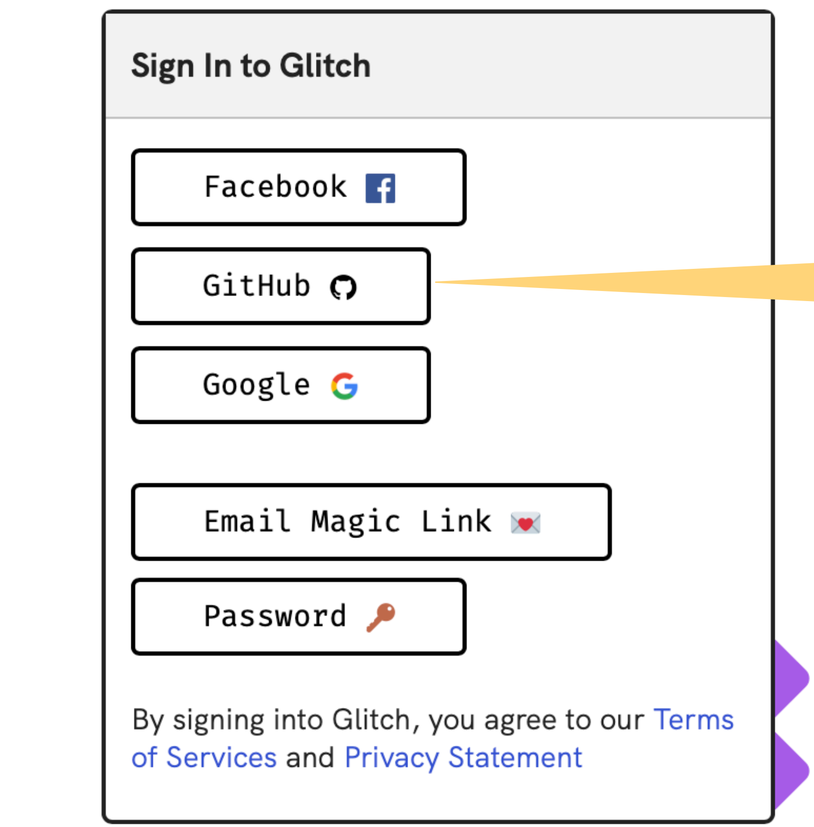
Guided Tour

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

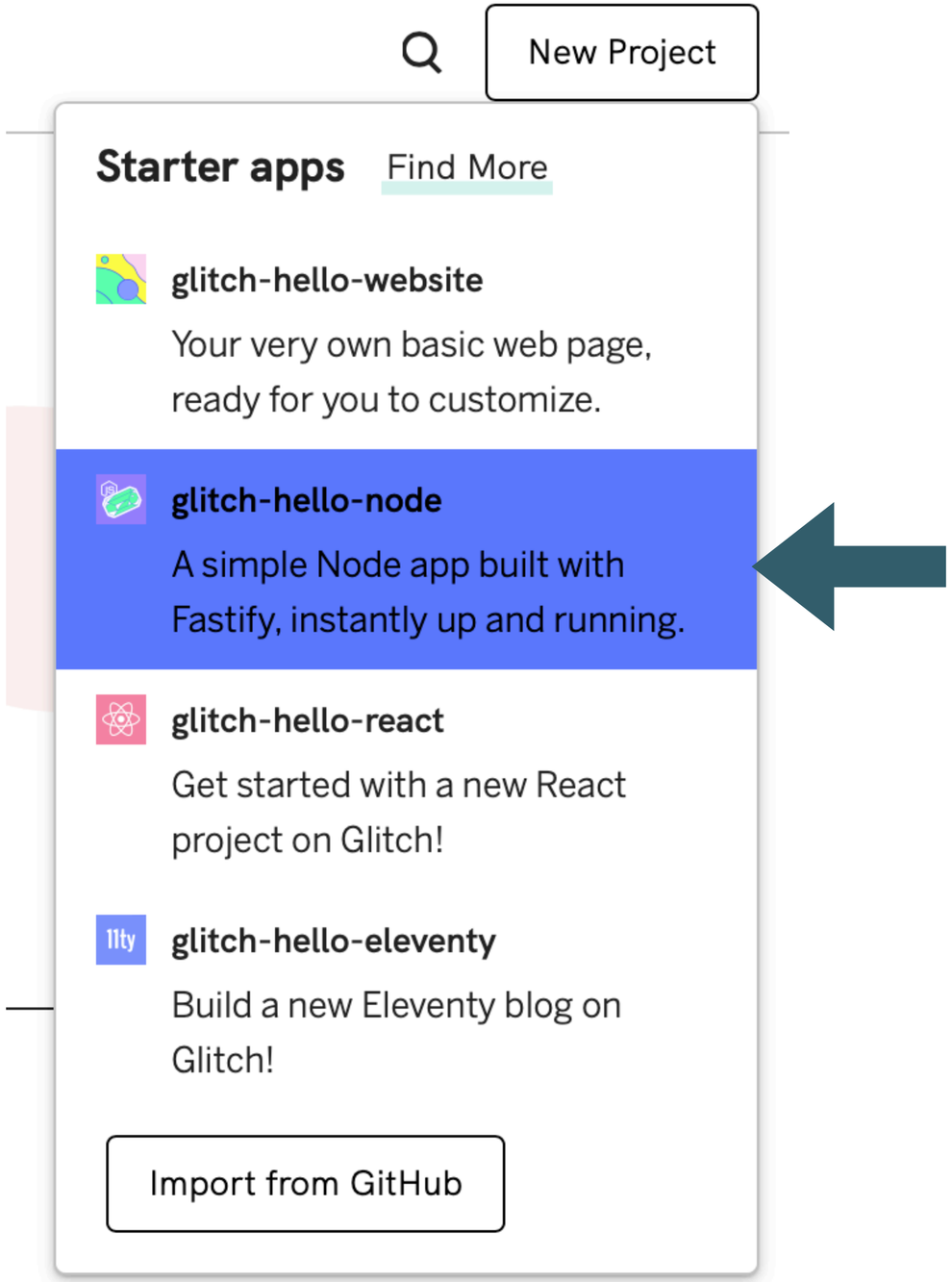
Prerequisite tools on your Workstation

none!

(apart from a browser + a github account)

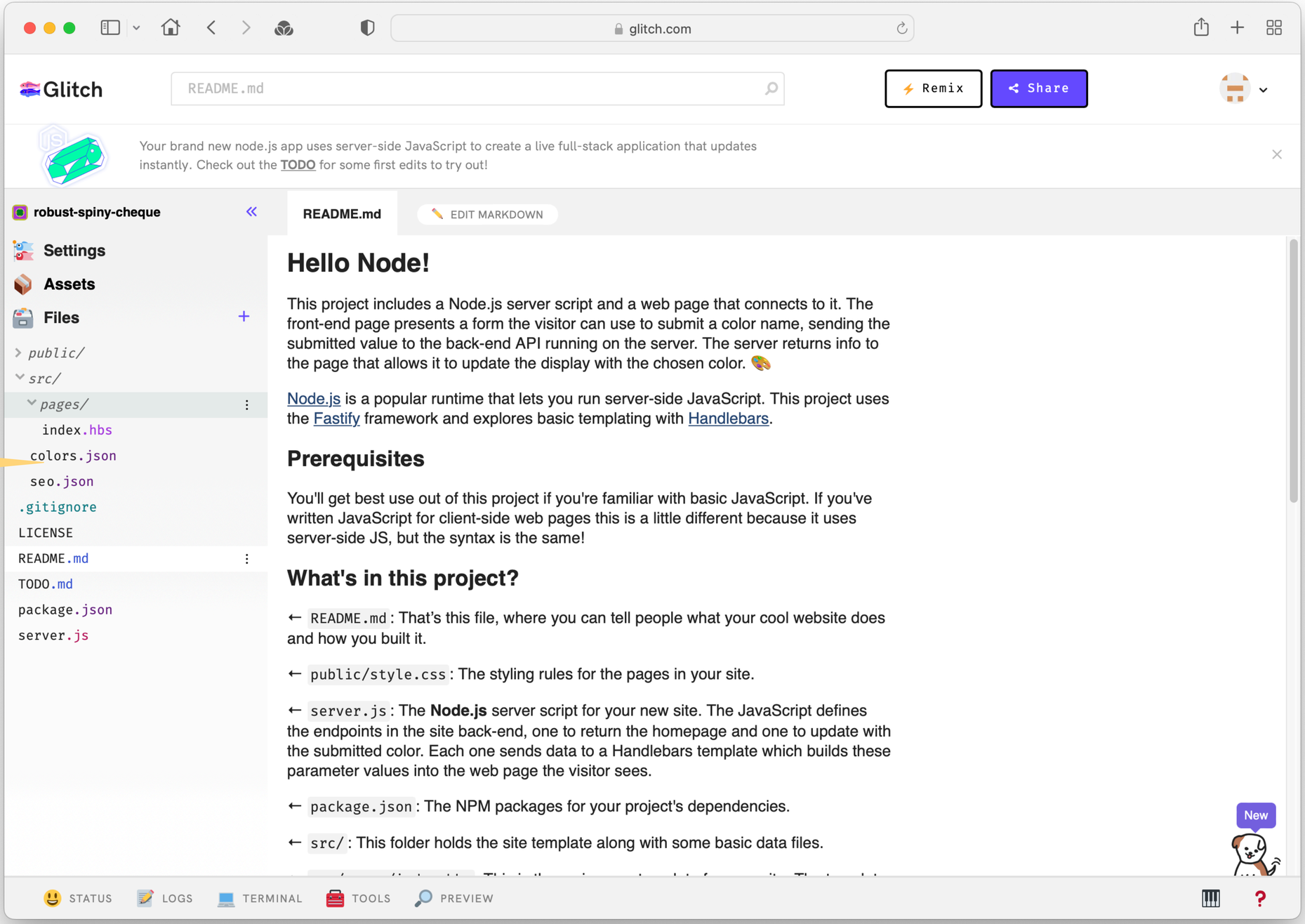


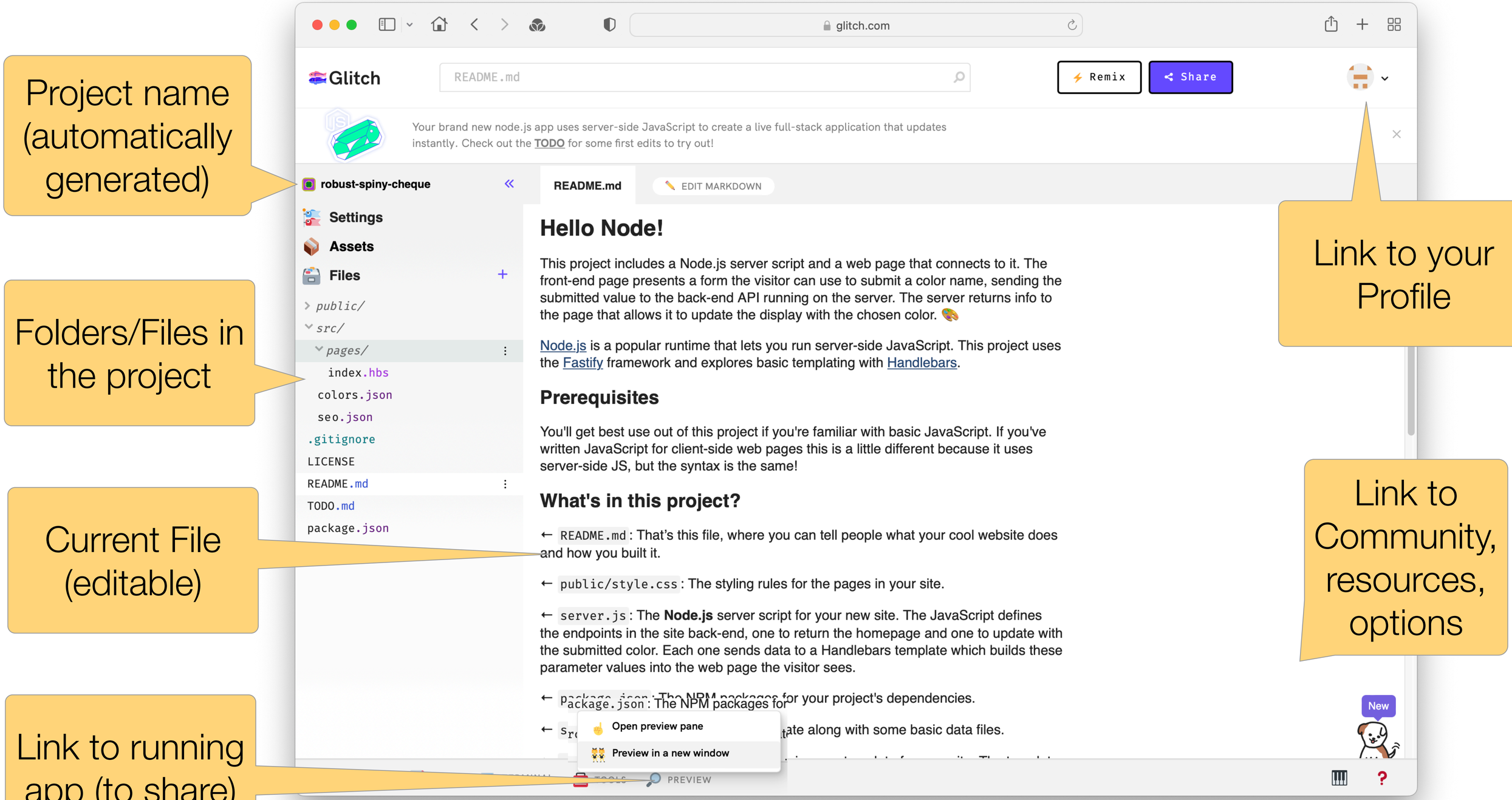
Sign via Github Account

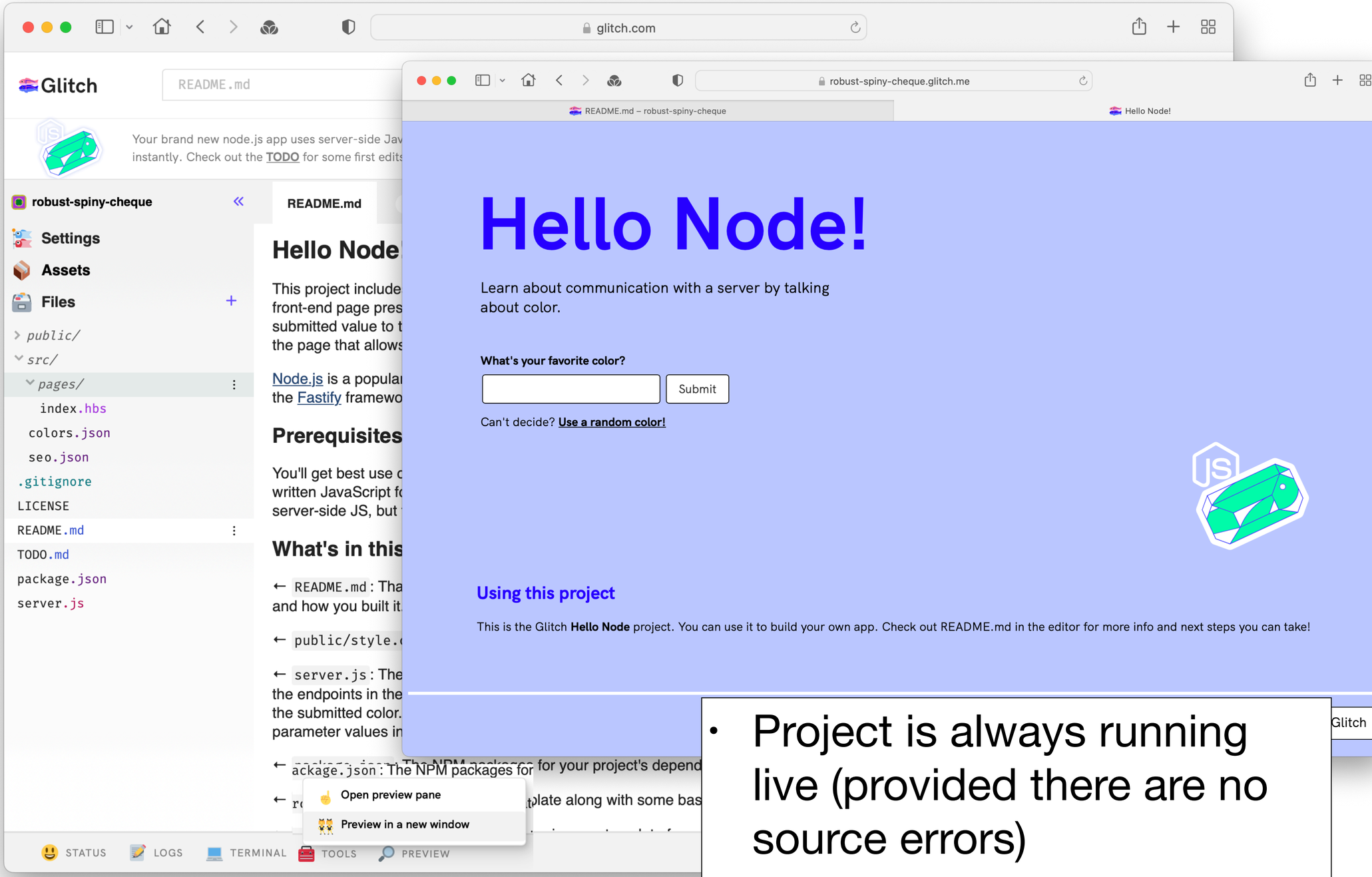


- The type of project we will be working with

App Source


glitch.com
Glitch
README.md
Remix
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
Settings
Assets
Files
public/
src/
pages/
index.hbs
colors.json
seo.json
.gitignore
LICENSE
README.md
TODO.md
package.json
server.js
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/ : This folder holds the site template along with some basic data files.
New
STATUS
LOGS
TERMINAL
TOOLS
PREVIEW

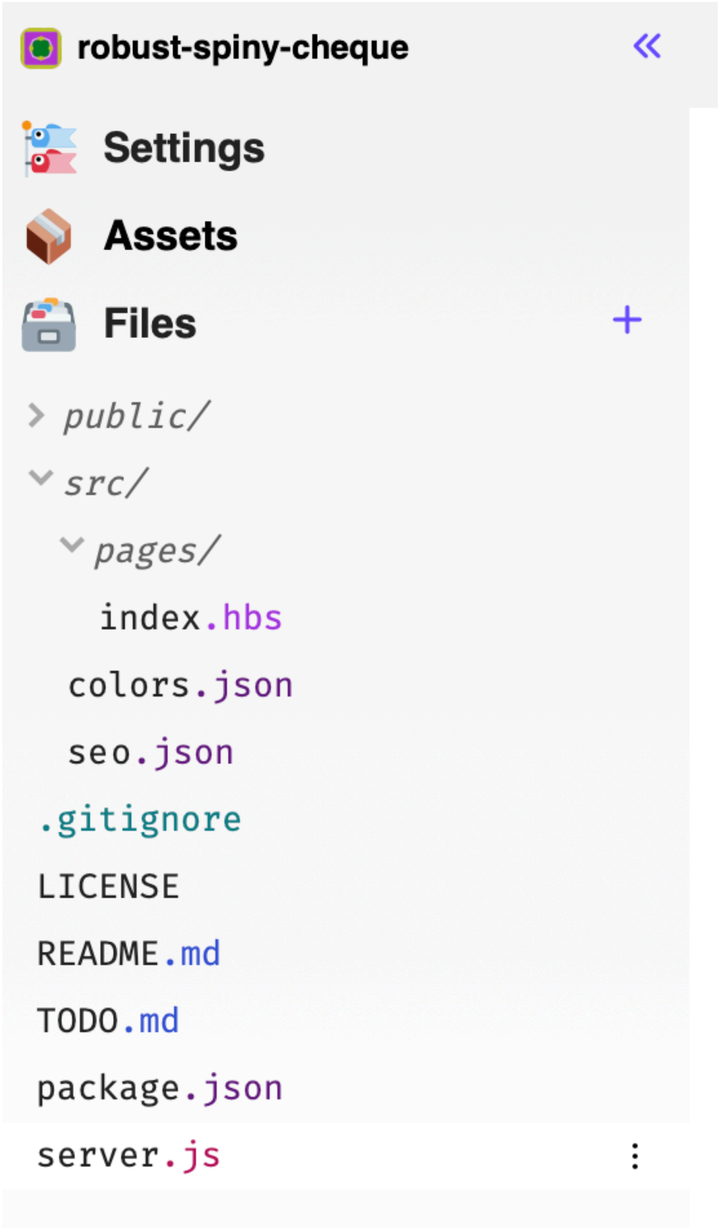

Project name (automatically generated)
Folders/Files in the project
Current File (editable)
Link to running app (to share)
Link to your Profile
Link to Community, resources, options
glitch.com
Glitch
README.md
Remix
Share
Your brand new node.js app uses server-side JavaScript to create a live full-stack application that updates instantly. Check out the TODO for some first edits to try out!
robust-spiny-cheque
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.
Open preview pane
Preview in a new window
PREVIEW
New



- 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



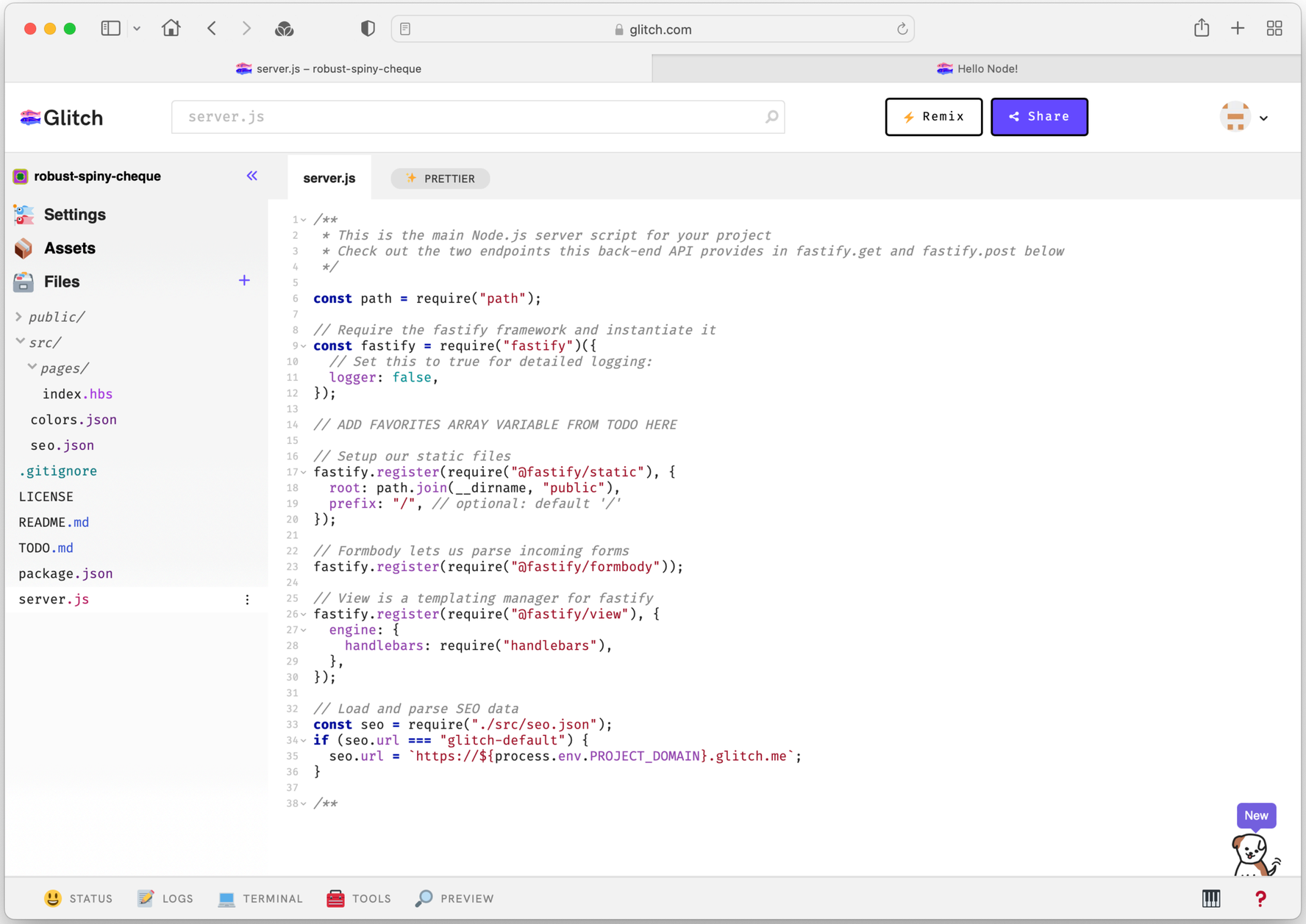
Front End

- 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)

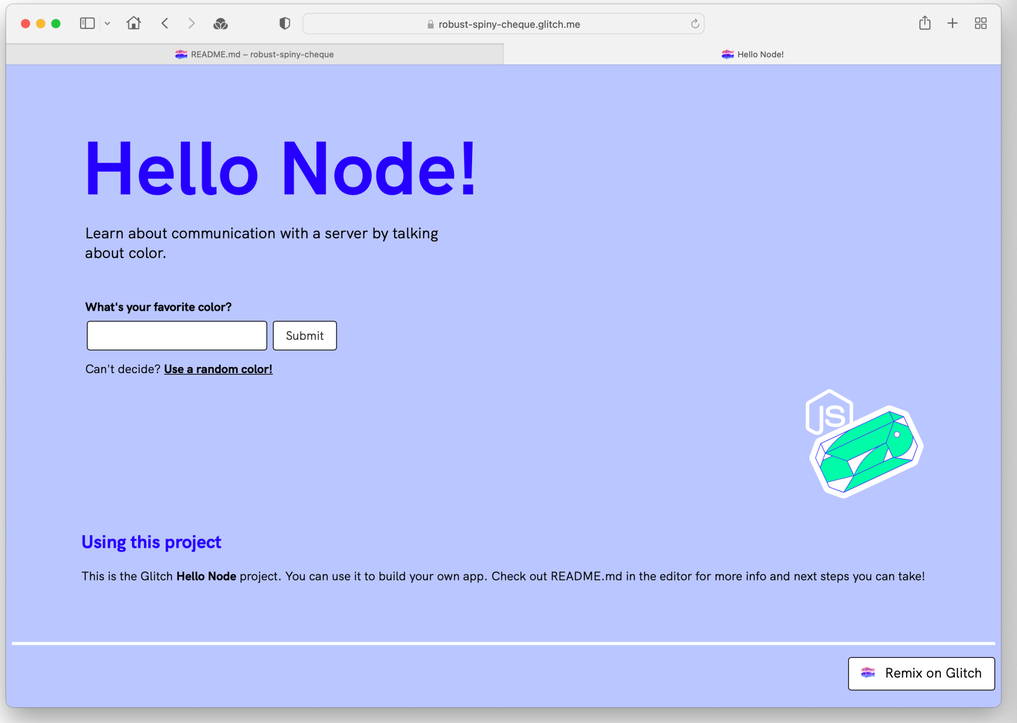
Back end

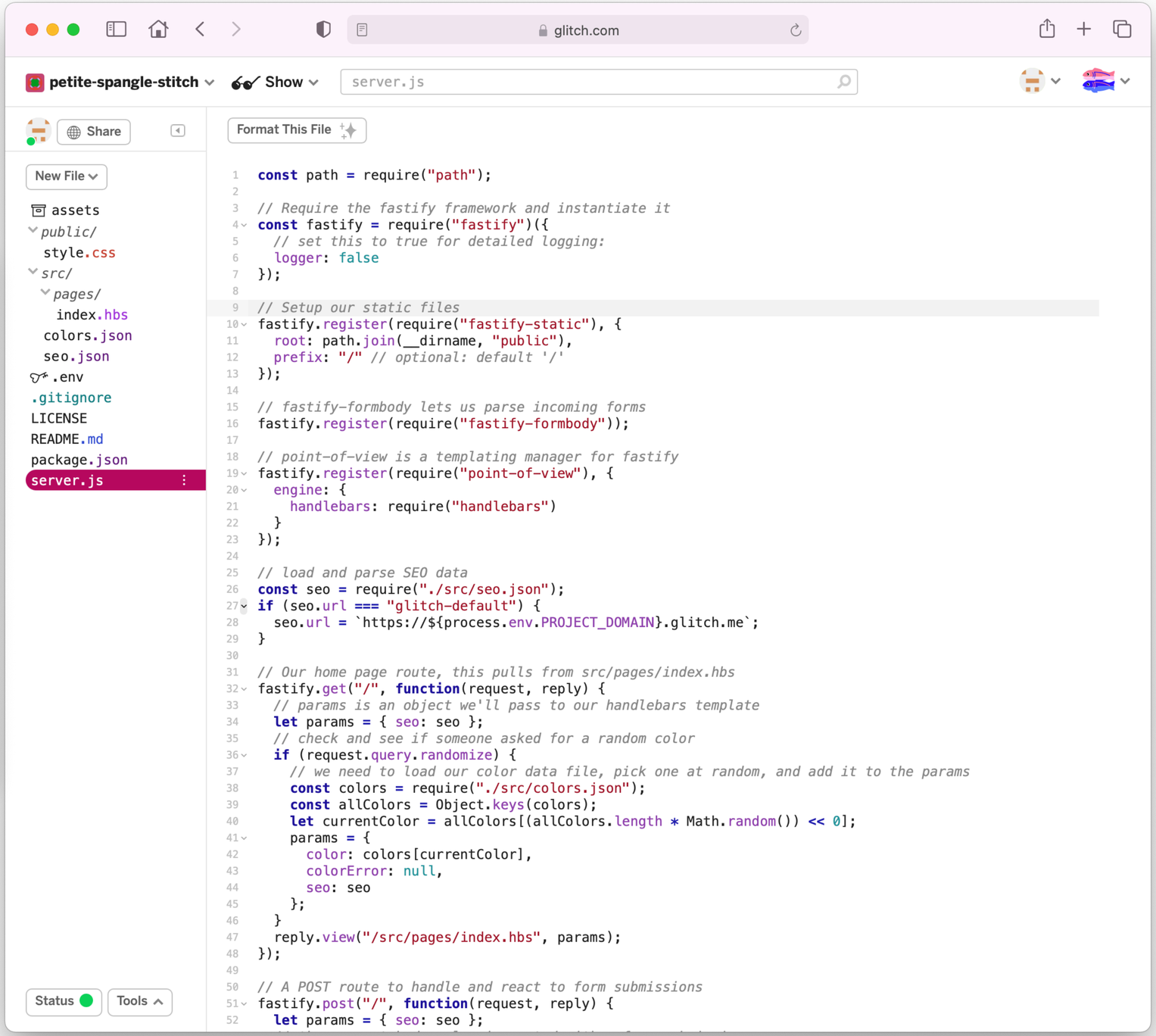
- An application - written in javascript - and hosted in the cloud.
- Application built in Javascript using a technology called node.js

The Starter App

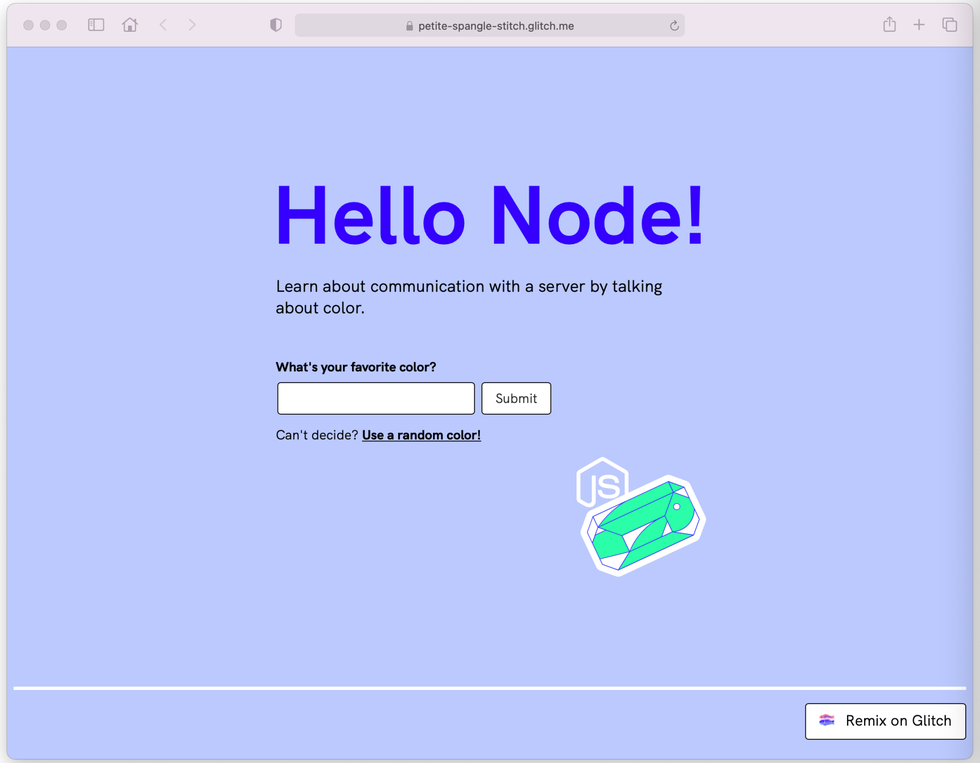


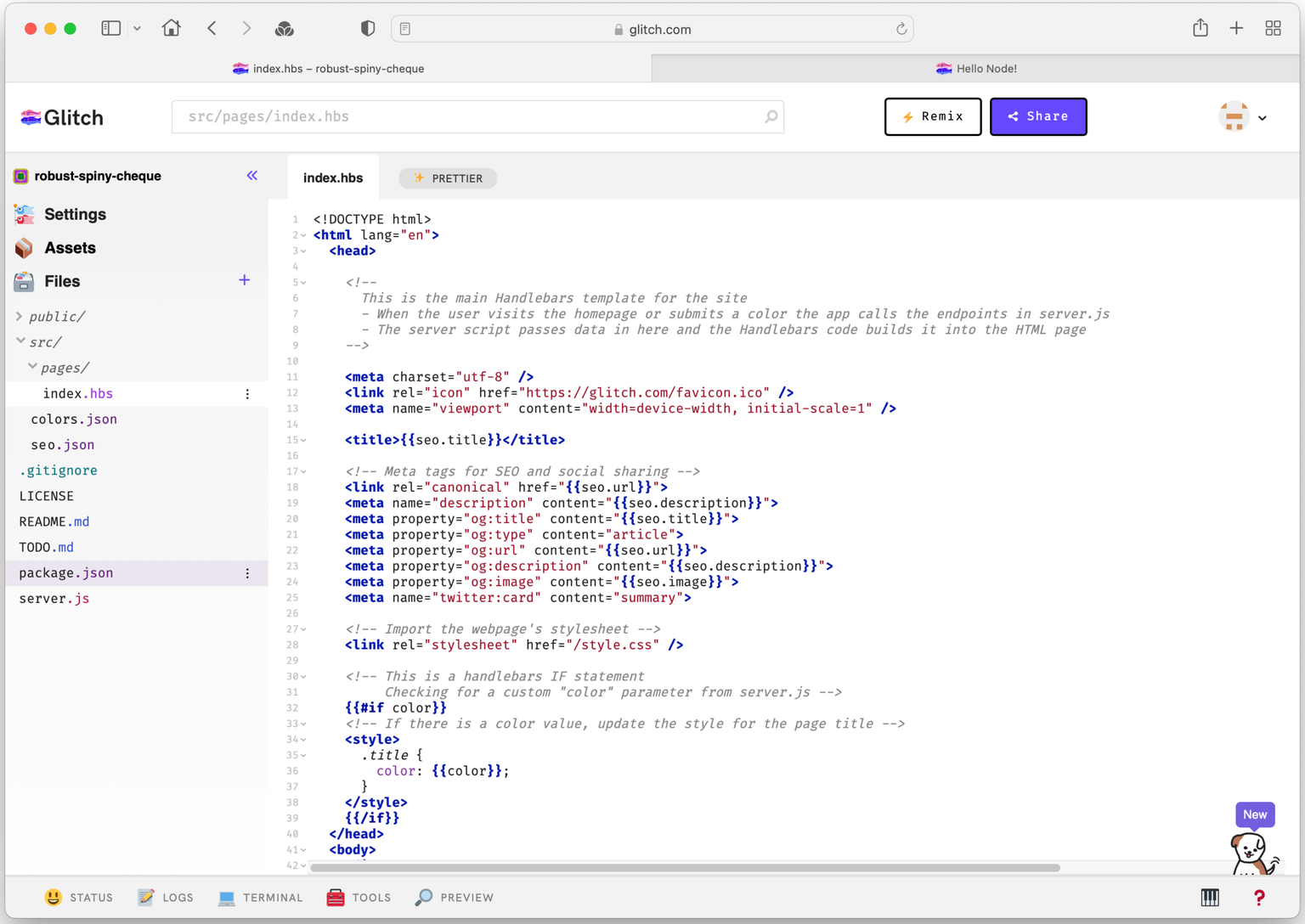
The Starter App





The Starter App





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>
```

Server side javascript

```
/**
 * 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);
});
```

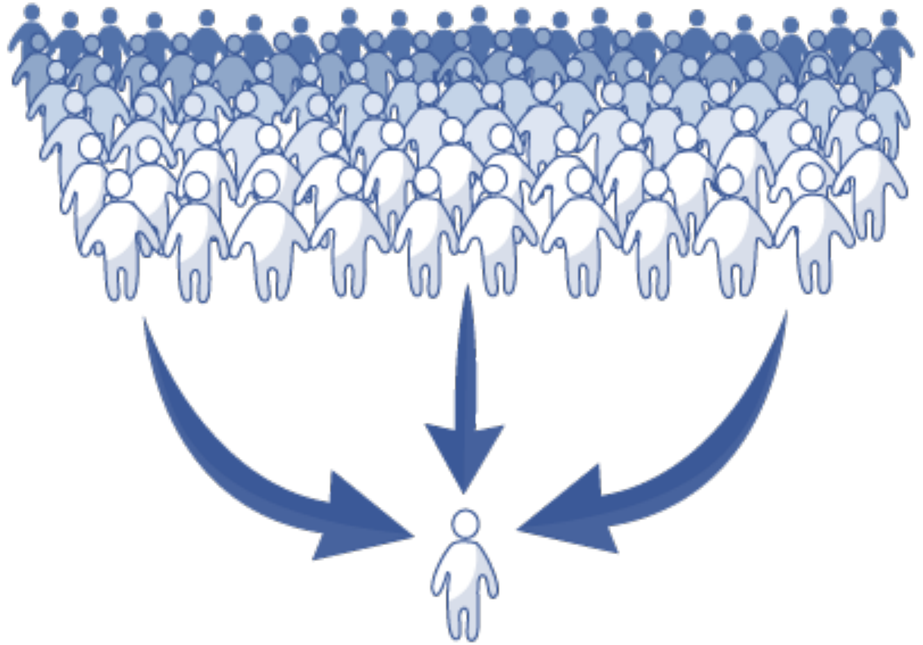
Client side html runs in each users browser

```
<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 ho
  <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>
```



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

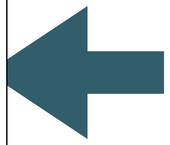
```
// A POST route to handle and react to form submissions
fastify.post("/", function(request, reply) {
  let params = { seo: seo };
  // the request.body.color is posted with a form submission
  let color = request.body.color;
  // if it's not empty, let's try to find the color
  if (color) {
    // 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

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