Getting started with SvelteKit

Enda Lee 2023

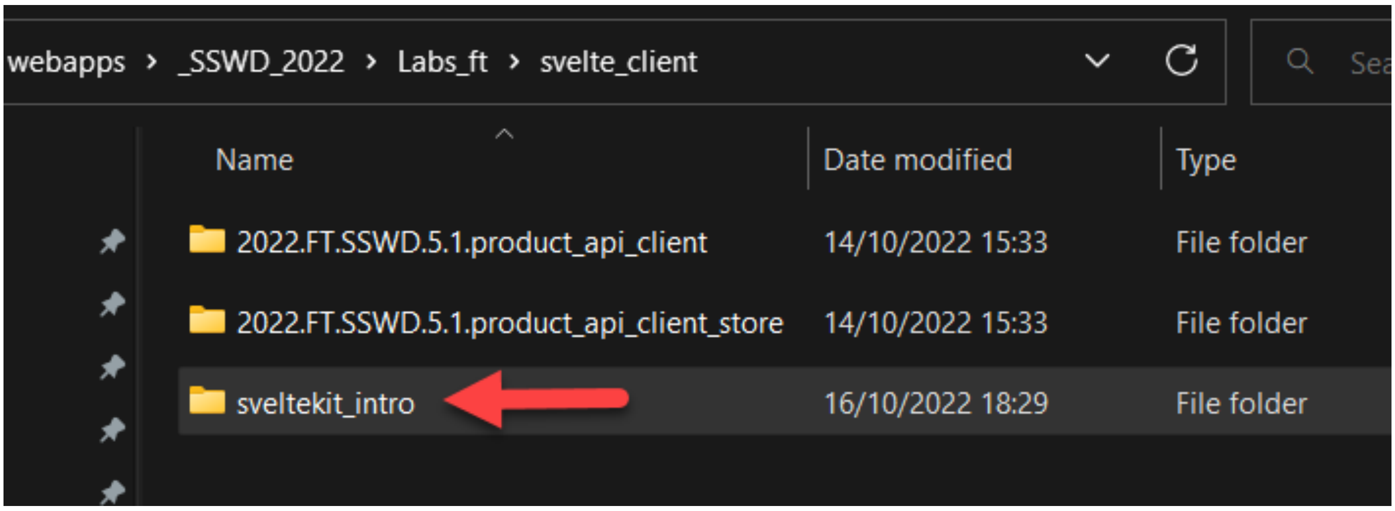
Introduction

From now on we will be using a front-end framework, **SvelteKit**, to build client web applications.

In this lab you will use **SvelteKit** to build and server a simple website.

1. Create a new SvelteKit application

Start by creating a folder for your application and the open it in VS Code, for example:



webapps › _SSWD_2022 › Labs_ft › svelte_client

Name	Date modified	Type
2022.FT.SSWD.5.1.product_api_client	14/10/2022 15:33	File folder
2022.FT.SSWD.5.1.product_api_client_store	14/10/2022 15:33	File folder
sveltekit_intro	16/10/2022 18:29	File folder

1. Execute the following command in a VS Code terminal to create the application

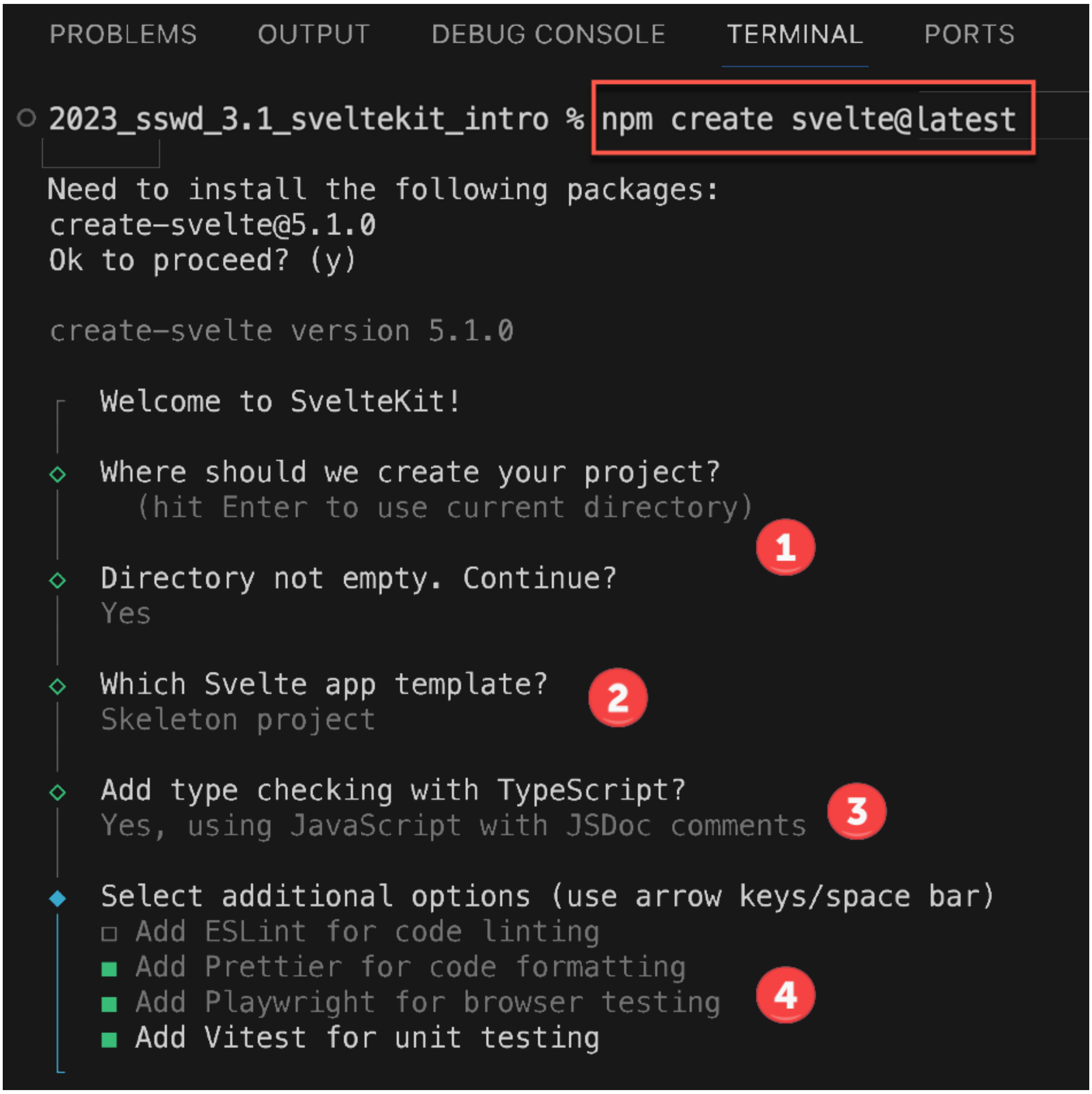
```
npm create svelte@latest
```

You will be asked a series of question:

1. Choose a directory name: **leave blank and press enter to use the current directory**
2. New project template: Use the arrow keys (up/ down) and press enter to **choose the Skeleton Project**
3. Choose language: **Select JavaScript**
4. a) Choose whether to use ESlint: **Choose No for this example**

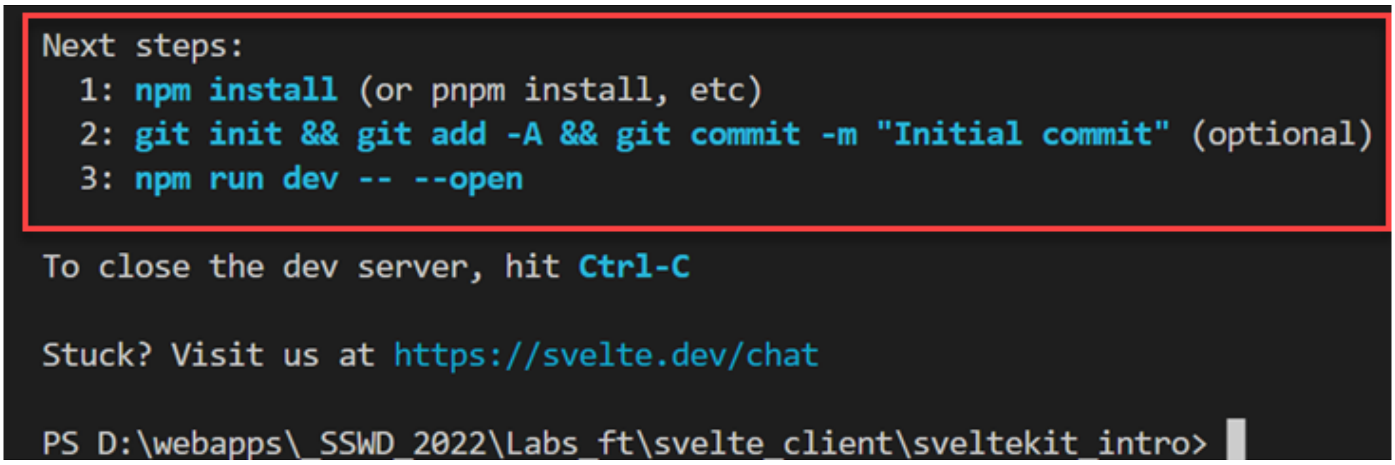
   b) Choose whether to use Prettier formatting: **Choose Yes**

   c) Testing options: **Choose No for this example**



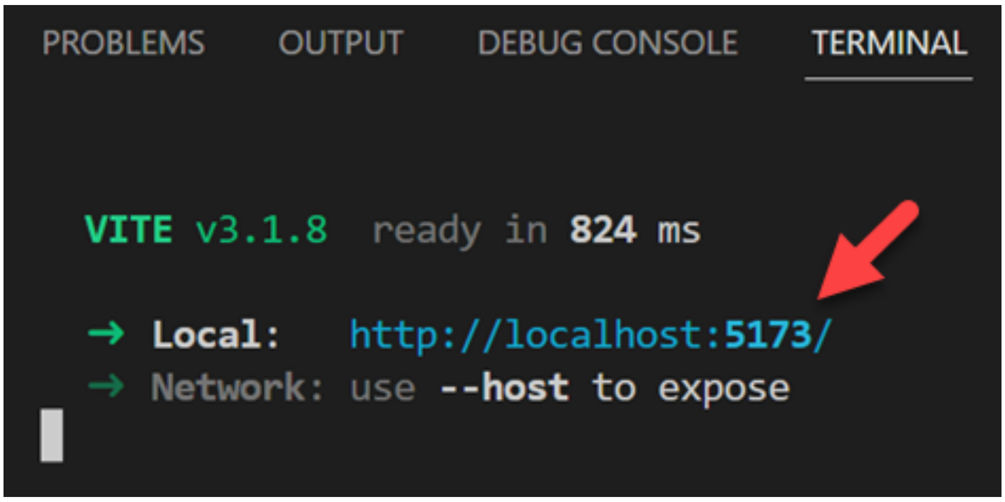
```
○ 2023_sswd_3.1_sveltekit_intro % npm create svelte@latest

  Need to install the following packages:
  create-svelte@5.1.0
  Ok to proceed? (y)

  create-svelte version 5.1.0

  ┌  Welcome to SvelteKit!
  │
  ◇  Where should we create your project?
  │    (hit Enter to use current directory)
  │
  ◇  Directory not empty. Continue?
  │  Yes
  │
  ◇  Which Svelte app template?
  │  Skeleton project
  │
  ◇  Add type checking with TypeScript?
  │  Yes, using JavaScript with JSDoc comments
  │
  ◆  Select additional options (use arrow keys/space bar)
  │  ◻ Add ESLint for code linting
  │  ◼ Add Prettier for code formatting
  │  ◼ Add Playwright for browser testing
  │  ◼ Add Vitest for unit testing
  └
```

The new SvelteKitapp is now created: run **npm install** and **npm run dev** to start it



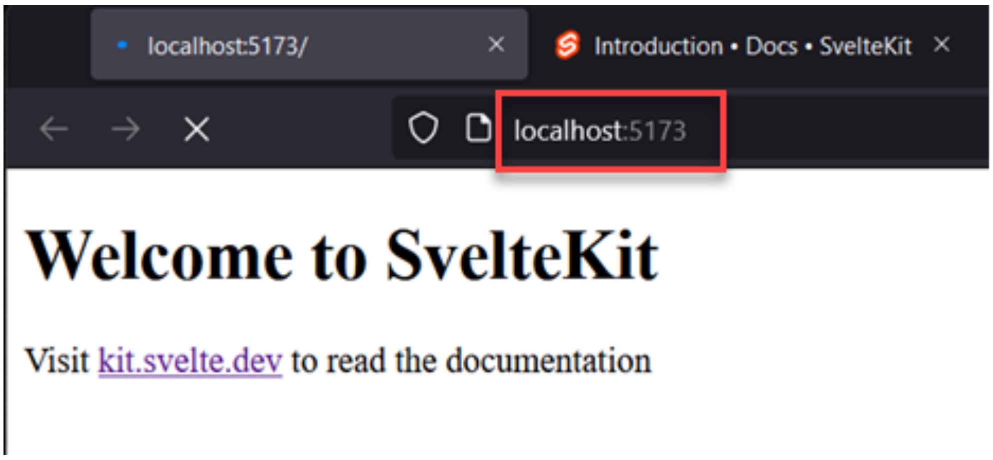
```
Next steps:
  1: npm install (or pnpm install, etc)
  2: git init && git add -A && git commit -m "Initial commit" (optional)
  3: npm run dev -- --open

To close the dev server, hit Ctrl-C

Stuck? Visit us at https://svelte.dev/chat

PS D:\webapps\_SSWD_2022\Labs_ft\svelte_client\sveltekit_intro>
```

8. Running the app



```
  VITE v3.1.8  ready in 824 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
```

9. Open in a browser using http://localhost:5173

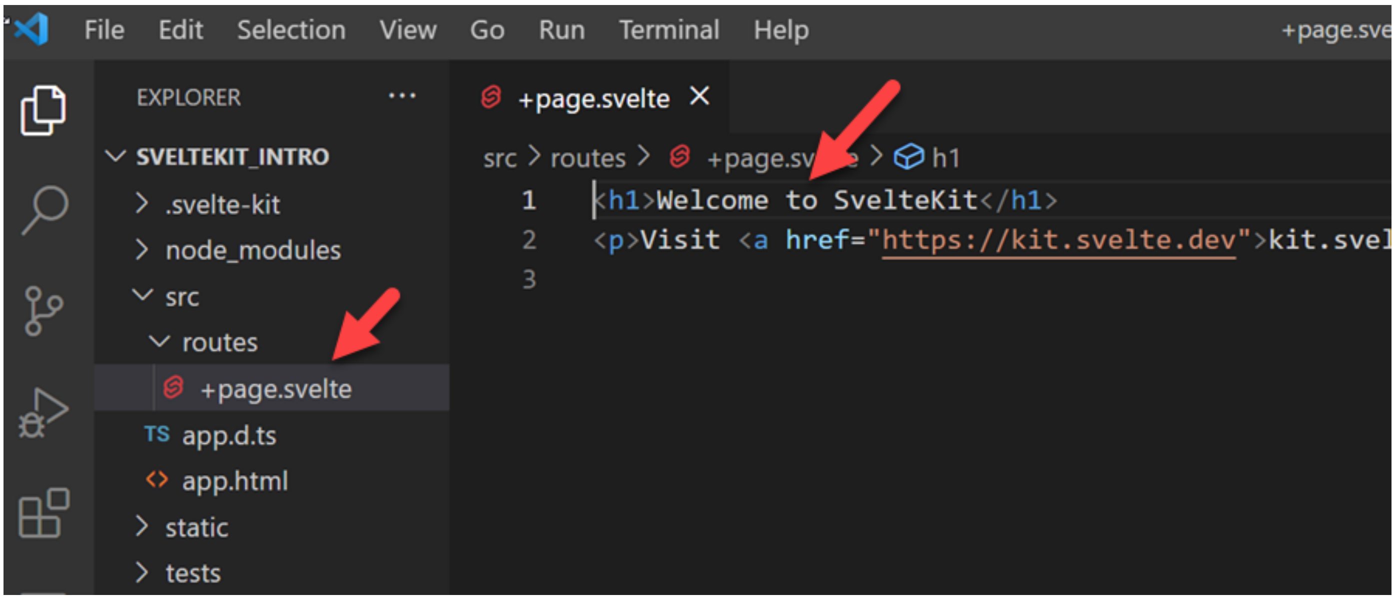


Welcome to SvelteKit

Visit kit.svelte.dev to read the documentation

2. Examine the App structure

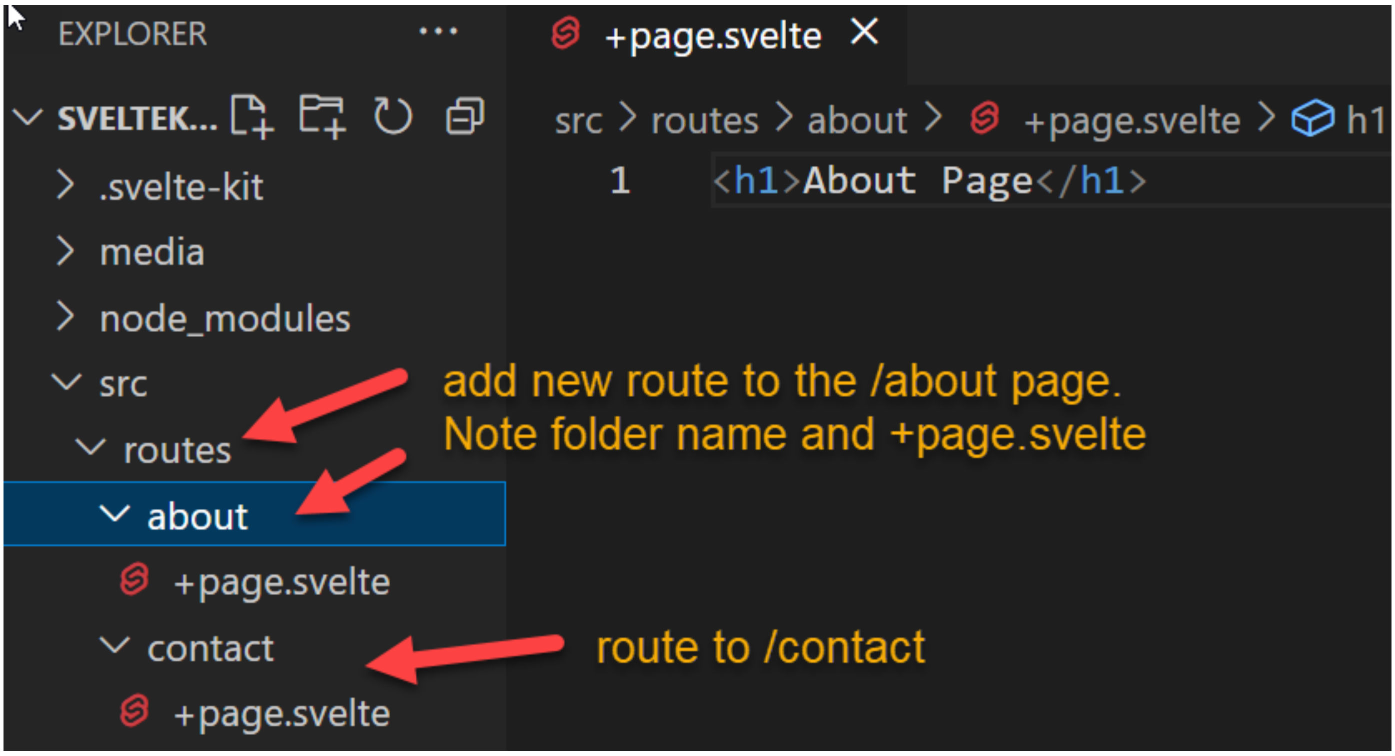
The home page served to the browser was generated from the `\src\routes` folder. The file `routes\+page.svelte` contains the home page. Note that this file naming scheme is a requirement of the framework.



```
<h1>Welcome to SvelteKit</h1>
<p>Visit <a href="https://kit.svelte.dev">kit.svel
```

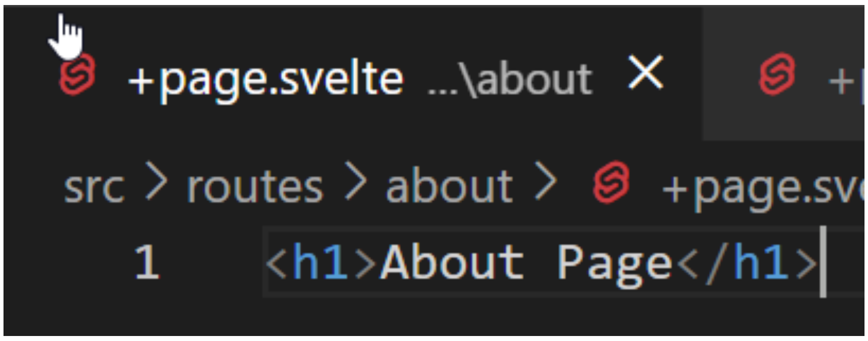
2.1. Adding new page routes

To add an **about** and **contact** page to the site, ad two new folders also named about and contact. Then add a new file to each folder named **+page.svelte**

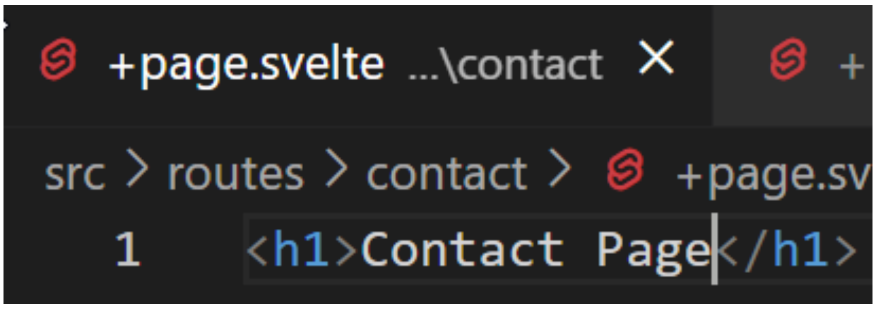


```
<h1>About Page</h1>
```

Add a H1 element to both pages to indicate the page content. Note that only html body content is required. The rest of the HTML page is loaded from `src/app.html`


+page.svelte ...\about
src > routes > about > +page.sv

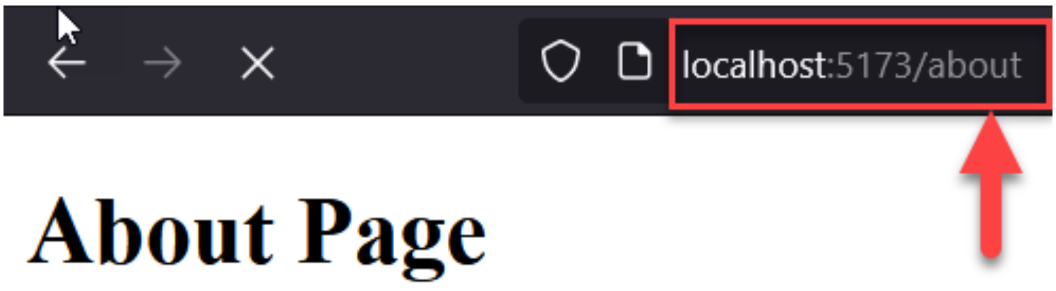
```
<h1>About Page</h1>
```

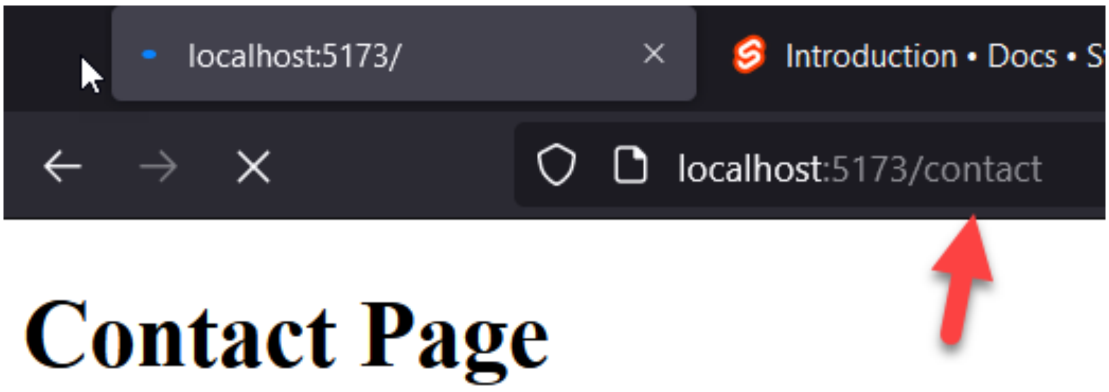

+page.svelte ...\contact
src > routes > contact > +page.sv

```
<h1>Contact Page</h1>
```

2.2. Testing the new routes

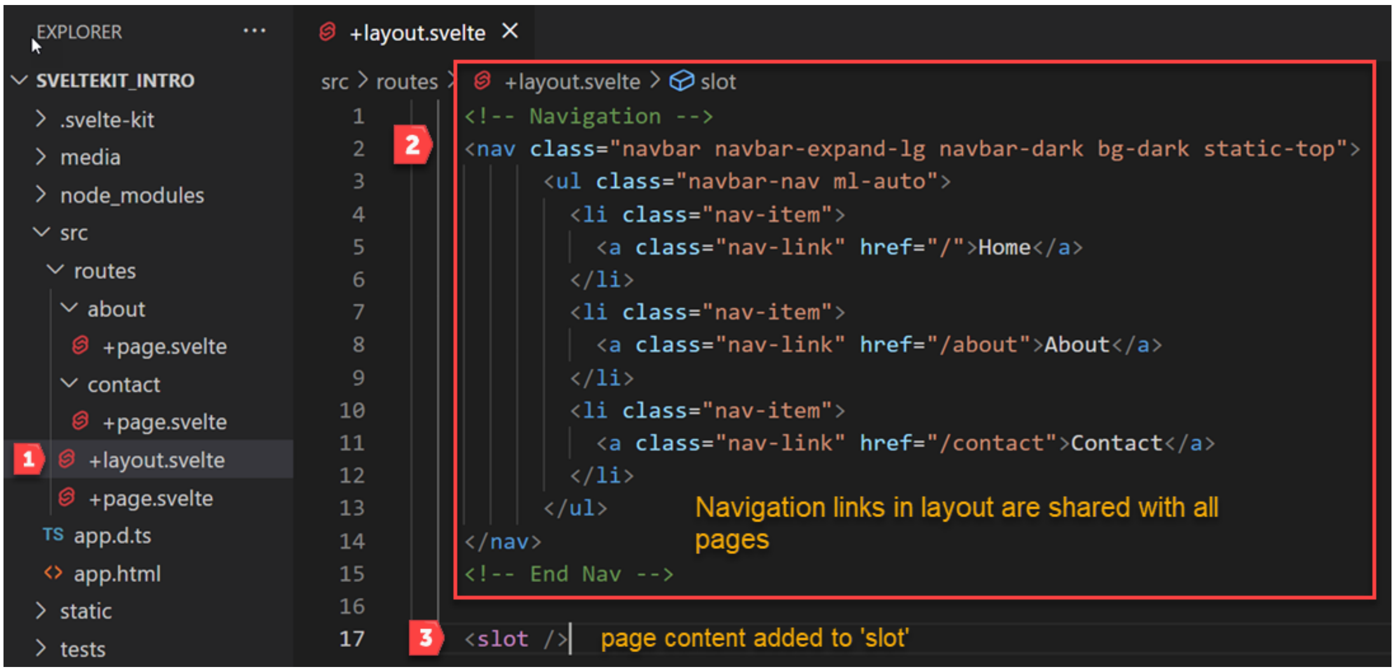
Open the new page routs in a browser. You will see that the the page/ route names are derived from the names of the folders added to `/src/routes/`





3. Adding a navigation menu

A shared layout definition can be used to easily add navigation links to all the pages. Add `+layout.svelte` to the **routes** folder:



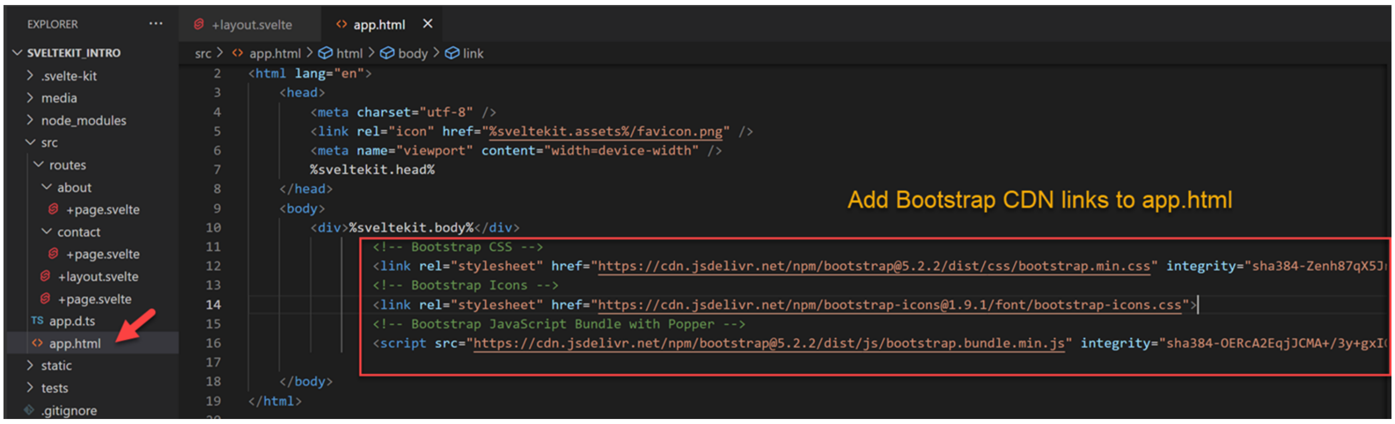
```html
<!-- Navigation -->
<nav class="navbar navbar-expand-lg navbar-dark bg-dark static-top">
      <ul class="navbar-nav ml-auto">
        <li class="nav-item">
          <a class="nav-link" href="/">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/about">About</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/contact">Contact</a>
        </li>
      </ul>
</nav>
<!-- End Nav -->

<slot />
```

The navigation menu uses Bootstrap 5 for styling. Also note the `<slot />` element. This indicates where page content will be added when the page is generated by Sveltekit.

3.1. Bootstrap dependencies

These should be added to `app.html` so that they are loaded globally.

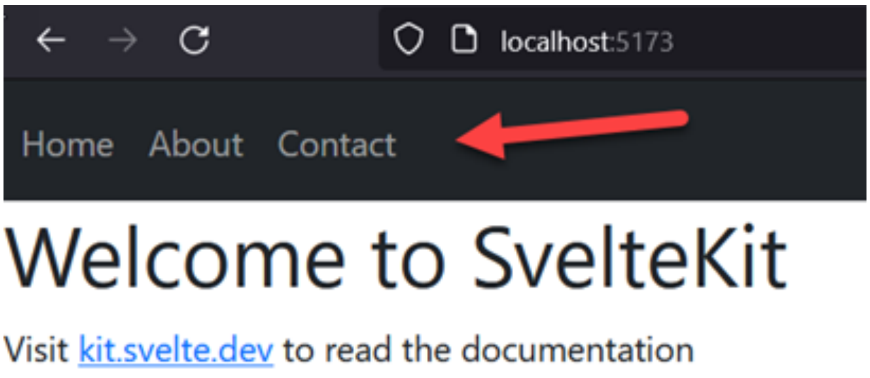
Get the latest CDN links from https://getbootstrap.com/ and add them to the end of the `<body>` section of `app.html`



```html
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <link rel="icon" href="%sveltekit.assets%/favicon.png" />
        <meta name="viewport" content="width=device-width" />
        %sveltekit.head%
    </head>
    <body>
        <div>%sveltekit.body%</div>
                <!-- Bootstrap CSS -->
                <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" integrity="sha384-Zenh87qX5J
                <!-- Bootstrap Icons -->
                <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.9.1/font/bootstrap-icons.css">
                <!-- Bootstrap JavaScript Bundle with Popper -->
                <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-OERcA2EqjJCMA+/3y+gxI

    </body>
</html>
```

3.2 Test the Navigation

Save all open files and reload the app in your browser. You should now have working navigation.

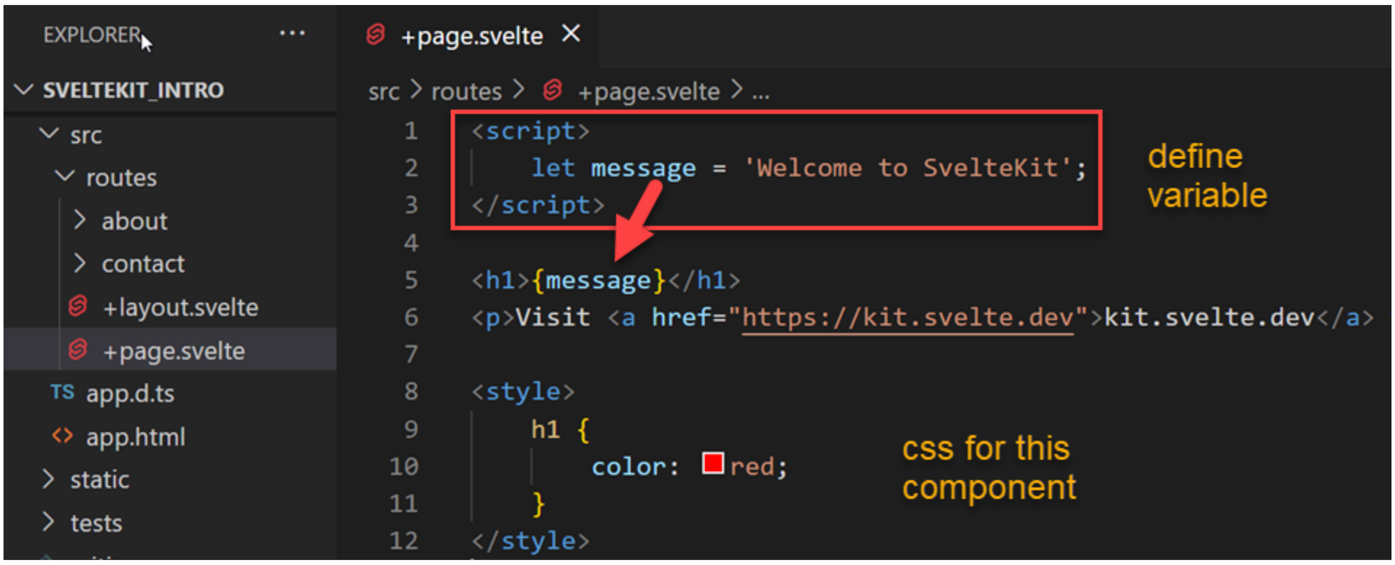


Welcome to SvelteKit

Visit kit.svelte.dev to read the documentation

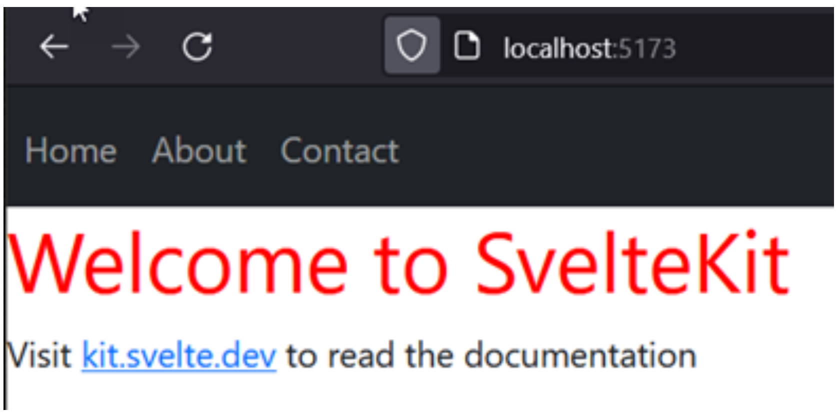
4. Component variable example

Each `.svelte` page is a self contained component which can include its own script, style, and HTML content. This simple example shows how to define a variable and use it in the page.



```
<script>
	let message = 'Welcome to SvelteKit';
</script>

<h1>{message}</h1>
<p>Visit <a href="https://kit.svelte.dev">kit.svelte.dev</a>

<style>
	h1 {
		color: red;
	}
</style>
```

Also notice how the CSS is only applied to the home page and not the others.



Welcome to SvelteKit

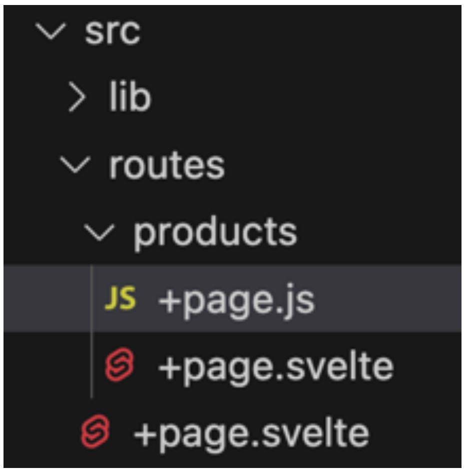
Visit kit.svelte.dev to read the documentation

5. Loading data (from an API)

The next step is to fetch some products from https://dummyjson.com/products and dsplay them in a page.

5.1 Add a new route for products

Follow the same process as above and add a new `products` route and `+page.svelte`. also add `+page.js` which will contain the `load()` function.



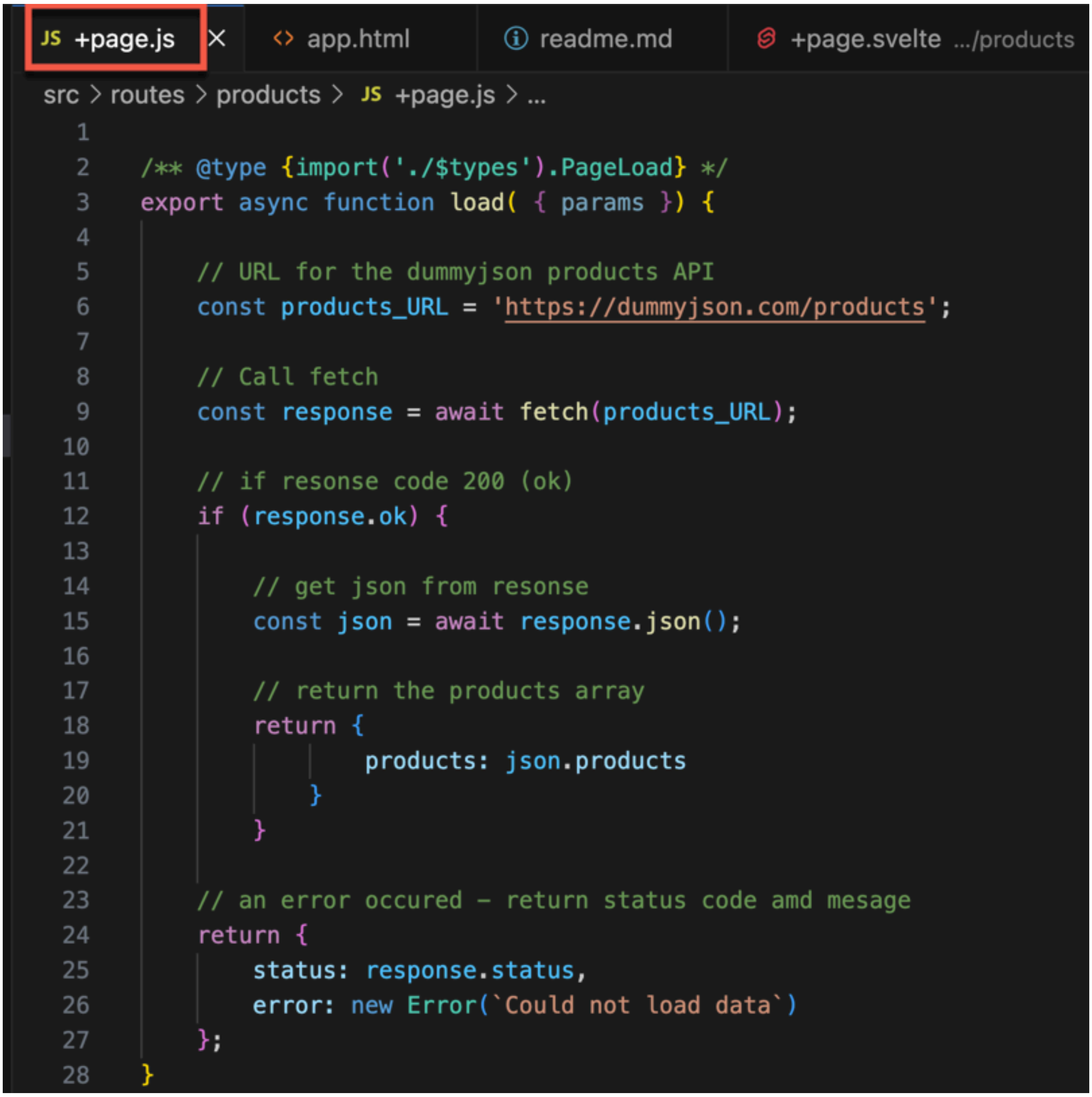
```
src
  lib
  routes
    products
      +page.js
      +page.svelte
    +page.svelte
```

5.2 Loading data

The product data must be read from the `API` before the page can be rendered. This is achieved in a `load` function in `+page.js`.   (see:https://kit.svelte.dev/docs/load)

1. Open `src/routes/products/+page.js` in VS Code and add the `load()` function.

   Read the comments for details



JS +page.js
app.html
readme.md
+page.svelte .../products
src > routes > products > JS +page.js > ...

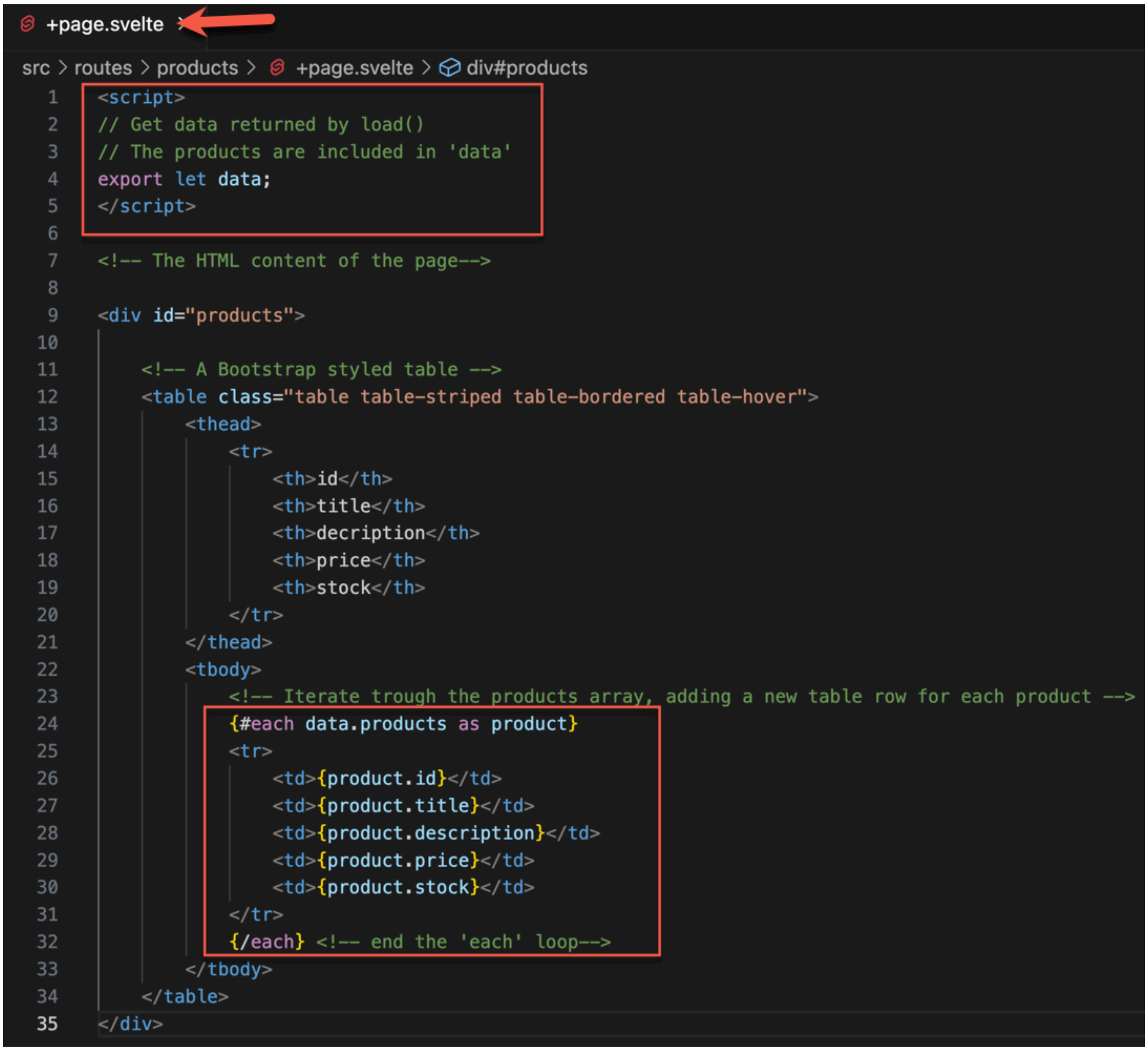
```javascript

/** @type {import('./$types').PageLoad} */
export async function load( { params }) {

    // URL for the dummyjson products API
    const products_URL = 'https://dummyjson.com/products';

    // Call fetch
    const response = await fetch(products_URL);

    // if resonse code 200 (ok)
    if (response.ok) {

        // get json from resonse
        const json = await response.json();

        // return the products array
        return {
                products: json.products
            }
        }

    // an error occured - return status code amd mesage
    return {
        status: response.status,
        error: new Error(`Could not load data`)
    };
}
```

2. Now open Open `src/routes/products/+page.scelte` in VS Code

   Add the following `<script>` and `HTML`.

   1. **Line 4:** The `data` object returned by `+page.js` is loaded by the `export let data` statement.
   2. **Line 24-32:** The products table is filled by iterating through `each` `product` in `data.products`

Read the comments for details



```
<script>
// Get data returned by load()
// The products are included in 'data'
export let data;
</script>

<!-- The HTML content of the page-->

<div id="products">

    <!-- A Bootstrap styled table -->
    <table class="table table-striped table-bordered table-hover">
        <thead>
            <tr>
                <th>id</th>
                <th>title</th>
                <th>decription</th>
                <th>price</th>
                <th>stock</th>
            </tr>
        </thead>
        <tbody>
            <!-- Iterate trough the products array, adding a new table row for each product -->
            {#each data.products as product}
            <tr>
                <td>{product.id}</td>
                <td>{product.title}</td>
                <td>{product.description}</td>
                <td>{product.price}</td>
                <td>{product.stock}</td>
            </tr>
            {/each} <!-- end the 'each' loop-->
        </tbody>
    </table>
</div>
```

6. Exercices

1. Add another route and page for `users`. Use this to display data from https://dummyjson.com/users
2. Add a route and page which will display the NASA Astronomy Picture of the Day.
    - The API endpoint is https://api.nasa.gov/planetary/apod?api_key=MY_KEY
    - You will need a free API key to use this service, see https://api.nasa.gov/

7. References

SvelteKit Docs: https://kit.svelte.dev/docs/introduction

Svelte Docs: https://svelte.dev/docs/introduction