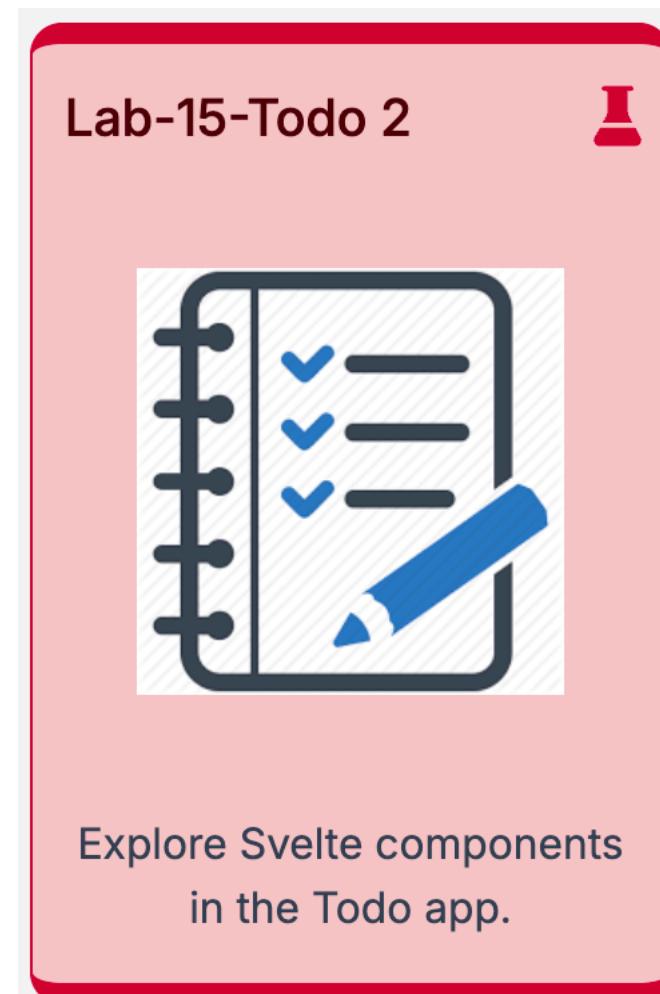
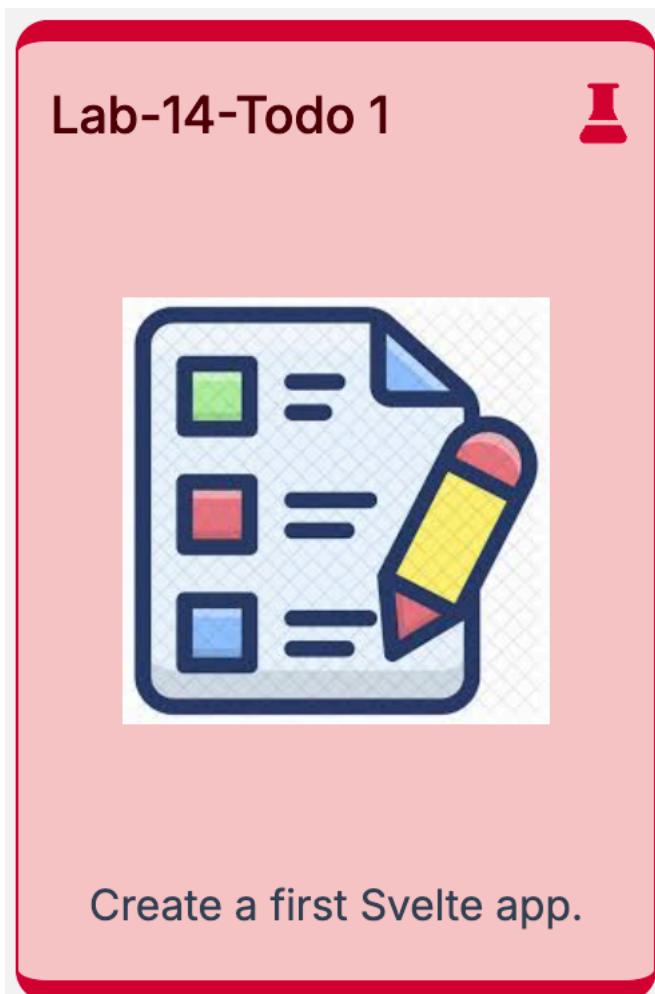
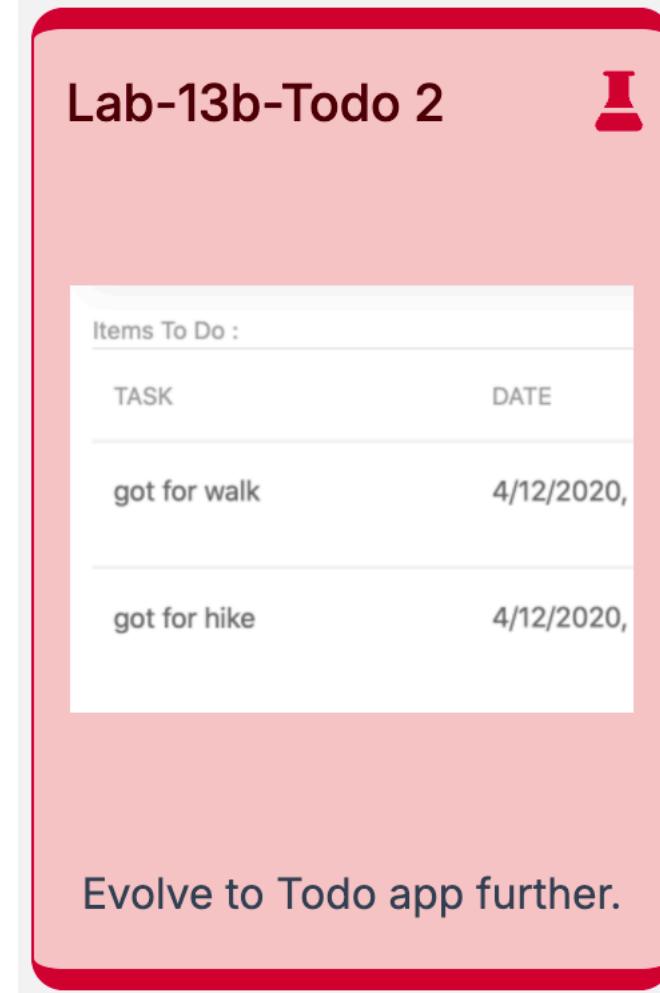
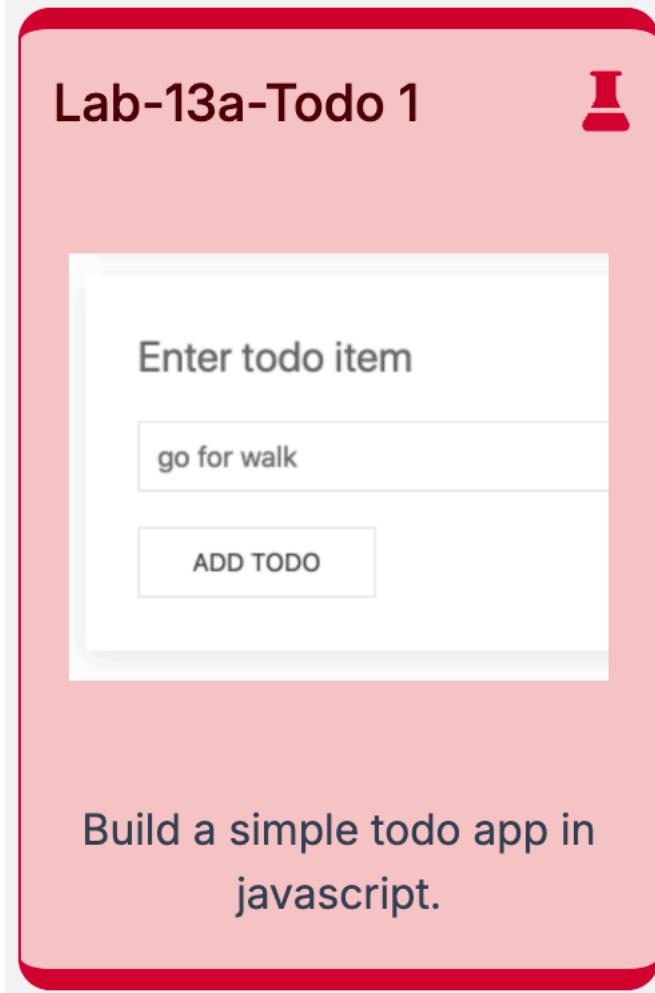


Svelte First Steps



Building your first Svelte
Application

Todo Example - Recap



Simple Todo List

Fun things to do

What should I do?

go for a cycle

Add Todo

Things yet do

Task	Date
go for a run	25/3/2022, 11:29:11
go for a cycle	25/3/2022, 11:29:22

delete

delete

Things done

Task	Date
go for a walk	25/3/2022, 11:29:16

Todo UX

Simple Todo List

Fun things to do

What should I do?

go for a cycle

Add Todo

Things yet do

Task	Date
------	------

go for a run	25/3/2022, 11:29:11
--------------	---------------------

delete

go for a cycle	25/3/2022, 11:29:22
----------------	---------------------

delete

Things done

Task	Date
------	------

go for a walk	25/3/2022, 11:29:16
---------------	---------------------

```
<div class="box has-text-centered">
  <div class="title"> Simple Todo List</div>
  <div class="subtitle">Fun things to do</div>
</div>

<div class="section box">
  <div class="field is-horizontal">
    <div class="field-label is-normal">
      <label class="label">What should I do?</label>
    </div>
    <div class="field-body">
      <div class="field">
        <p class="control">
          <input id="todo-id" class="input" type="text" placeholder="What should I do?">
        </p>
      </div>
      <button onClick="addTodo()" class="button">Add Todo</button>
    </div>
  </div>
</div>

<div class="section box">
  <div class="title is-6">Things yet do</div>
  <table id="todo-table" class="table is-fullwidth">
    <thead>
      <th>Task</th>
      <th>Date</th>
      <th></th>
    </thead>
    <tbody>
      <tr></tr>
    </tbody>
  </table>
</div>

<div class="section box">
  <div class="title is-6">Things done</div>
  <table id="done-table" class="table is-fullwidth">
    <thead>
      <th>Task</th>
      <th>Date</th>
      <th></th>
    </thead>
    <tbody>
      <tr></tr>
    </tbody>
  </table>
</div>
```

Include Javascript

```
</div>
<div class="section box">
  <div class="title is-6">Things yet do</div>
  <table id="todo-table" class="table is-fullwidth">
    <thead>
      <th>Task</th>
      <th>Date</th>
      <th></th>
    </thead>
    <tbody>
      <tr></tr>
    </tbody>
  </table>
</div>
<div class="section box">
  <div class="title is-6">Things done</div>
  <table id="done-table" class="table is-fullwidth">
    <thead>
      <th>Task</th>
      <th>Date</th>
      <th></th>
    </thead>
    <tbody>
      <tr></tr>
    </tbody>
  </table>
</div>
</div>
<script src="todo.js" type="text/javascript"></script>
</body>
</html>
```

Simple Todo List
Fun things to do

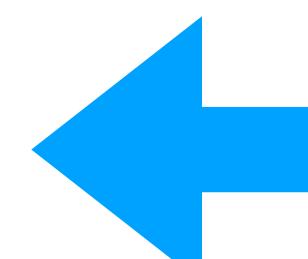
What should I do? go for a cycle Add Todo

Things yet do

Task	Date	
go for a run	25/3/2022, 11:29:11	delete
go for a cycle	25/3/2022, 11:29:22	delete

Things done

Task	Date
go for a walk	25/3/2022, 11:29:16



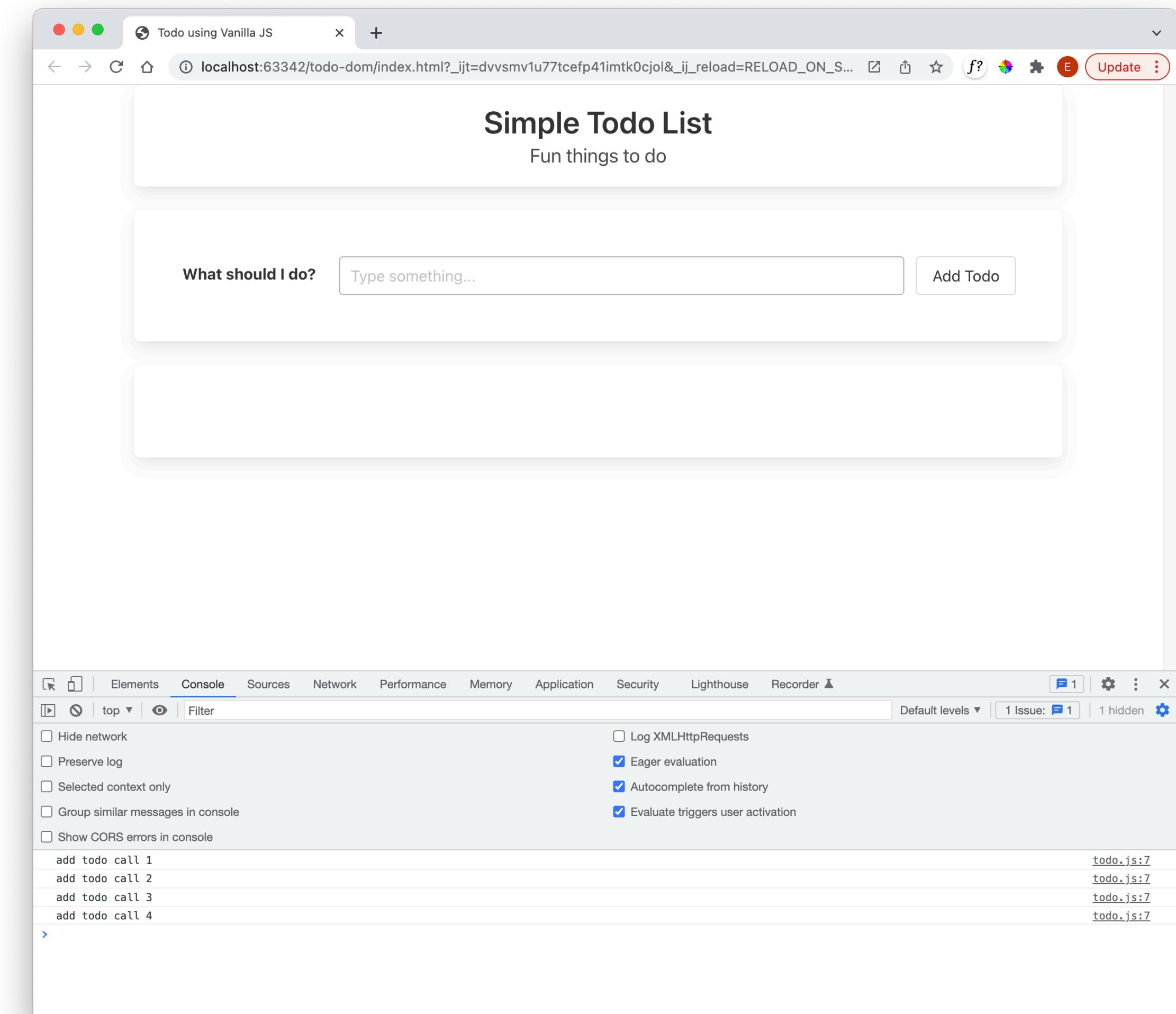
Incorporate Javascript into page

onClick

```
<div class="uk-width-1-2@m uk-card uk-card-default uk-padding">
  <fieldset class="uk-fieldset">
    <legend class="uk-legend">Enter todo item</legend>
    <div class="uk-margin">
      <input id="todo-id" class="uk-input" type="text" placeholder="Todo">
    </div>
  </fieldset>
  <button onClick="addTodo()" id="add-btn"
    class="uk-button uk-button-default">Add Todo</button>
</div>
```

```
let count = 0;

function addTodo() {
  count++;
  console.log(`add todo call ${count}`)
}
```



- onClick attribute establishes link from <button> to javascript function

Things yet do

Task

go for a run

go for a cycle

```
<table id="todo-table" class="table">
  <caption> Items To Do :</caption>
  <thead>
    <tr>
      <th>Task</th>
    </tr>
  </thead>
  <tbody>
    <tr></tr>
  </tbody>
</table>
```

- Retrieve text from input field
- Create new row & insert text element

```
function addTodo() {
  count++;
  const todoText = document.getElementById("todo-id").value;
  console.log(`add todo call ${count}: ${todoText}`)

  const table = document.getElementById("todo-table");
  const row = table.insertRow(-1);
  const textCell = row.insertCell(0);
  textCell.innerText = todoText;
}
```

Retrieve Table
DOM Object

DOM Interaction

- Rosetta Stone approach
- Develop the same application(s) in Svelte

Lab-13a-Todo 1 

Enter todo item

go for walk

ADD TODO

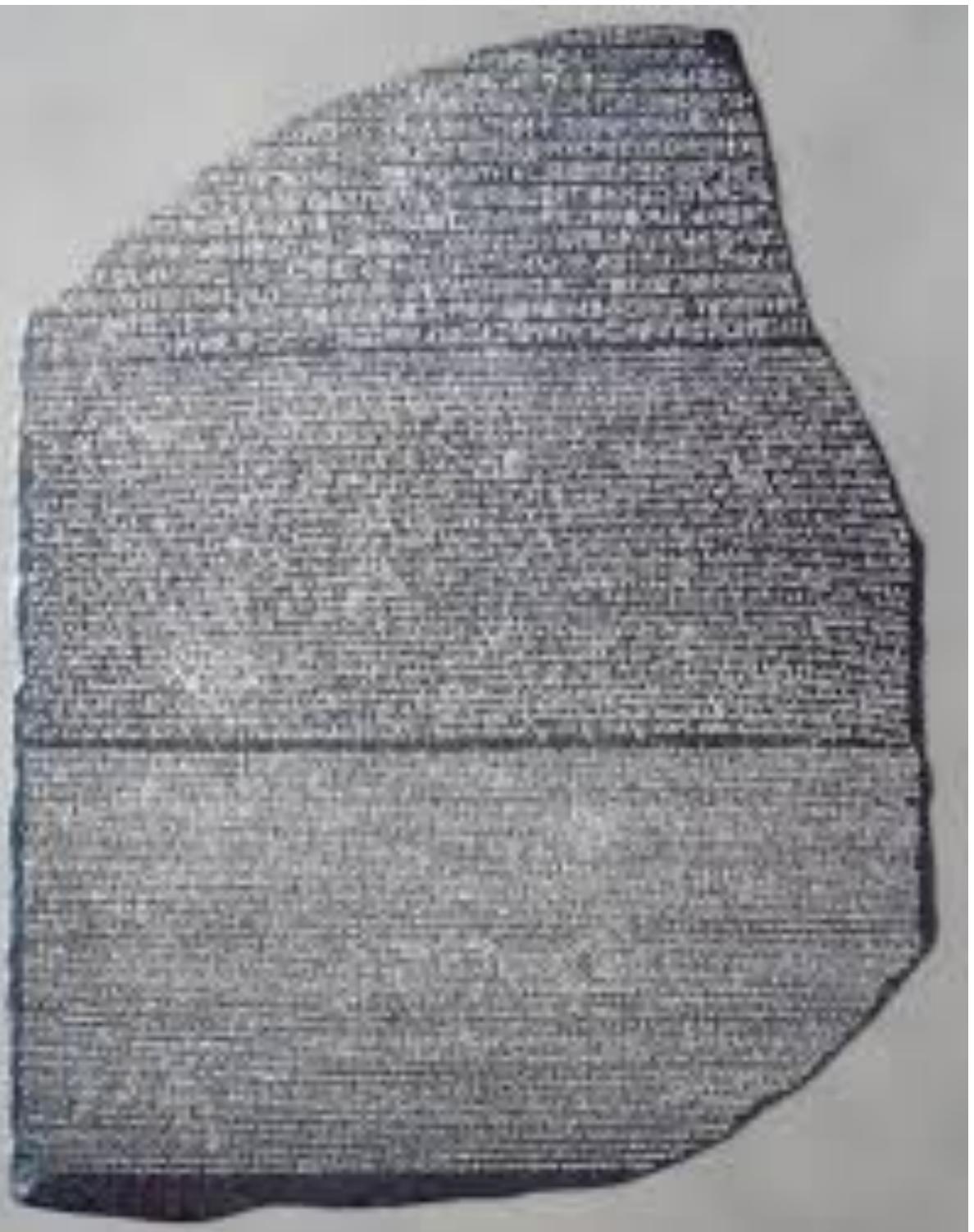
Build a simple todo app in javascript.

Lab-13b-Todo 2 

Items To Do :

TASK	DATE
got for walk	4/12/2020,
got for hike	4/12/2020,

Evolve to Todo app further.



Lab-14-Todo 1 



Create a first Svelte app.

Lab-15-Todo 2 



Explore Svelte components in the Todo app.

```
npx sv create todo-svelte
```

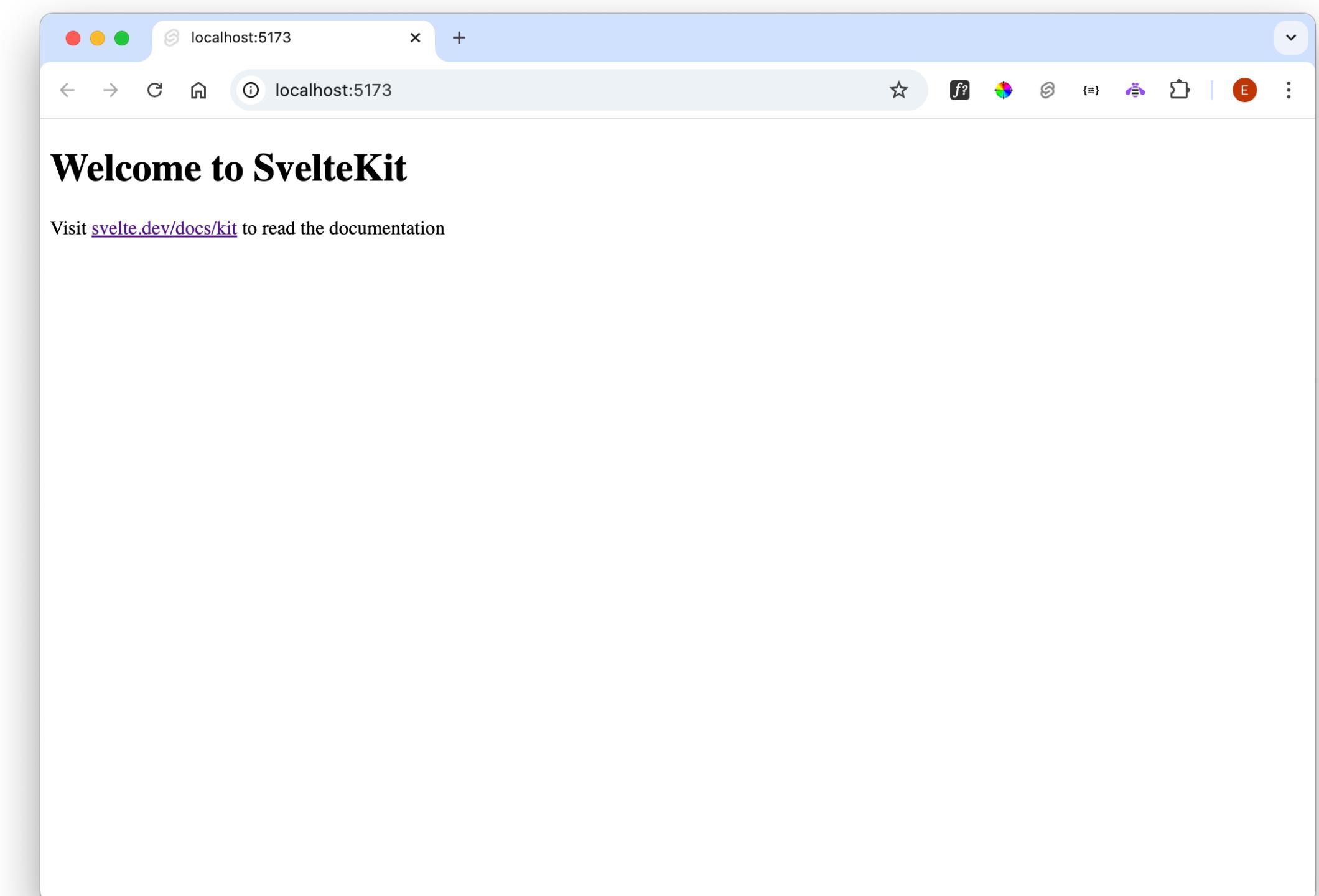
```
| Welcome to the Svelte CLI! (v0.6.21)  
|  
◆ Which template would you like?  
|   ● SvelteKit minimal (barebones scaffolding for your new app)  
|   ○ SvelteKit demo  
|   ○ Svelte library  
◆ Add type checking with Typescript?  
|   ○ Yes, using Typescript syntax
```

```
cd todo-svelte  
npm run dev
```

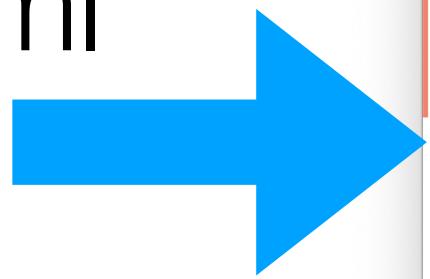
```
> todo@0.0.1 dev  
> vite dev  
  
17:10:27 [vite] (client) Forced re-optimization of dependencies  
  
VITE v6.1.0 ready in 1184 ms  
  
→ Local: http://localhost:5173/  
→ Network: use --host to expose  
→ press h + enter to show help
```

Create Svelte App

<http://localhost:5173>



App.html



Svelte
application
launched (on
front end)

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows the project structure with folders like '.svelte-kit', 'node_modules', 'src' (containing 'lib/index.js', 'routes/+page.svelte', and 'app.html'), 'static', and various configuration files.
- Code Editor:** The 'app.html' file is open, displaying Svelte code for a home page. It includes imports for 'sveltekit.assets%/favicon.png' and 'https://cdn.jsdelivr.net/npm/bulma@0.9.4/css/bulma.min.css'.
- Terminal:** The 'TERMINAL' tab is active, showing the command 'npm run dev' being run. The output indicates the default shell is zsh, provides instructions to update the account, and shows the vite server starting at port 5173.
- Status Bar:** Shows file statistics (Ln 14, Col 1), encoding (UTF-8), and other developer tools.

+page.svelte

Application
Home Page

App.html

Bulma CSS
framework

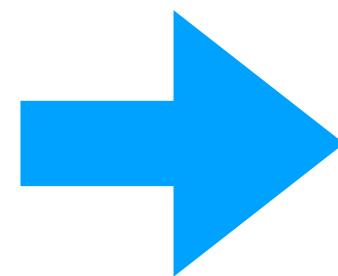
Entire Front
End Application
'bundled' into
this Javascript
file

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%sveltekit.assets%/favicon.png" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    %sveltekit.head%
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bulma@0.9.4/css/bulma.min.css" />
  </head>
  <body data-sveltekit-preload-data="hover">
    <div style="display: contents">%sveltekit.body%</div>
  </body>
</html>
```



+routes/+page.svelte

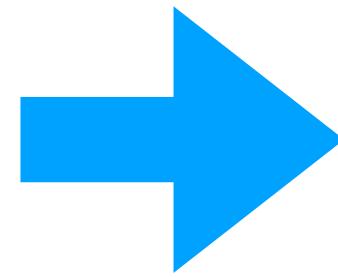
Component
Code (empty)



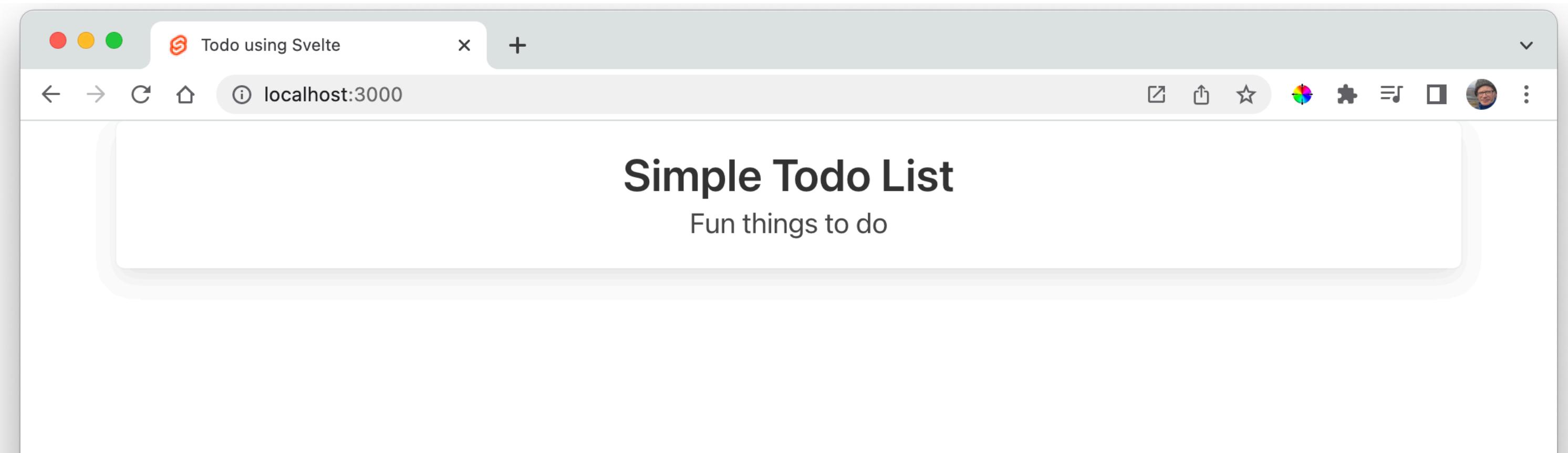
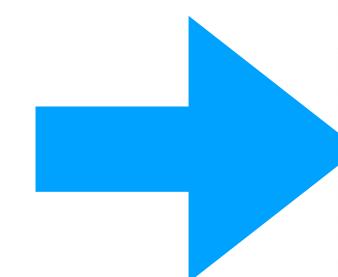
```
<script>
</script>

<div class="container">
  <div class="box has-text-centered">
    <div class="title"> Simple Todo List</div>
    <div class="subtitle">Fun things to do</div>
  </div>
</div>
```

Component UI



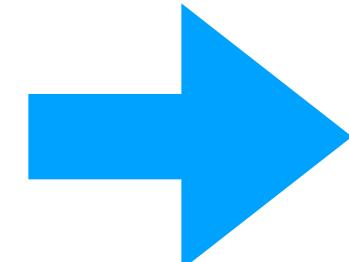
Running
Application



Incorporating Form UX

routes/+.page.svelte

Additional
Form
elements



```
<div class="section box">
  <div class="field is-horizontal">
    <div class="field-label is-normal">
      <label for="todo" class="label">What should I do?</label>
    </div>
    <div class="field-body">
      <div class="field">
        <p class="control">
          <input id="todo" class="input" type="text" placeholder="Type something...">
        </p>
      </div>
      <button class="button">Add Todo</button>
    </div>
  </div>
</div>
```

Form
unresponsive
(yet)

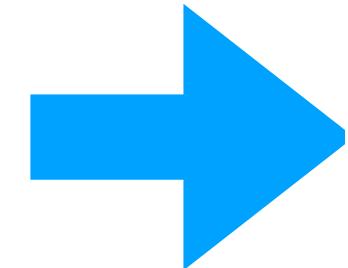
What should I do?

Type something...

Add Todo

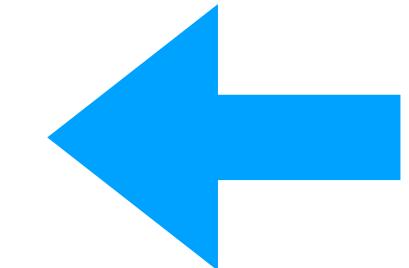
Component Logic

Simple
Variable +
function



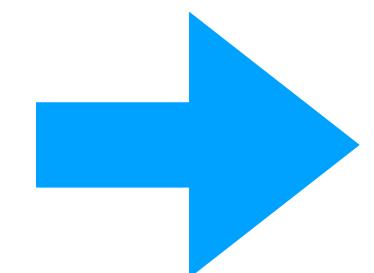
```
<script>
  let todoText;
  function addTodo() {
    console.log(todoText)
  }
</script>
```

```
<input
  bind:value={todoText}
  id="todo"
  class="input"
  type="text"
  placeholder="Type something...">
```

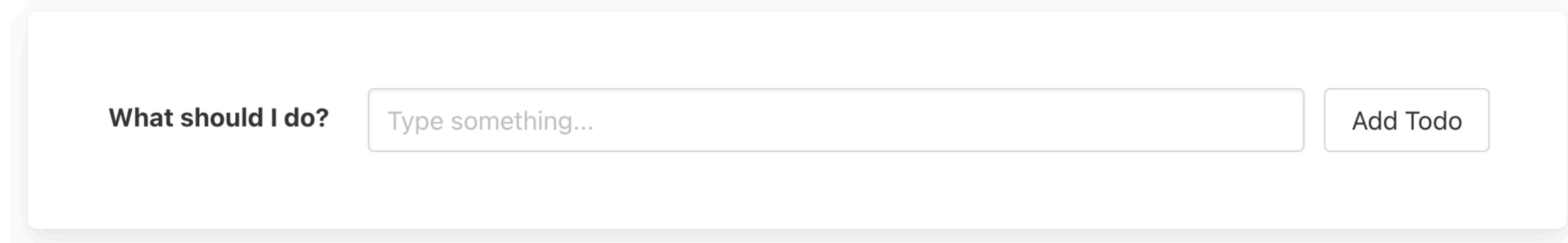


“Bind” the
variable to the
input field

Trigger
function
when button
pressed



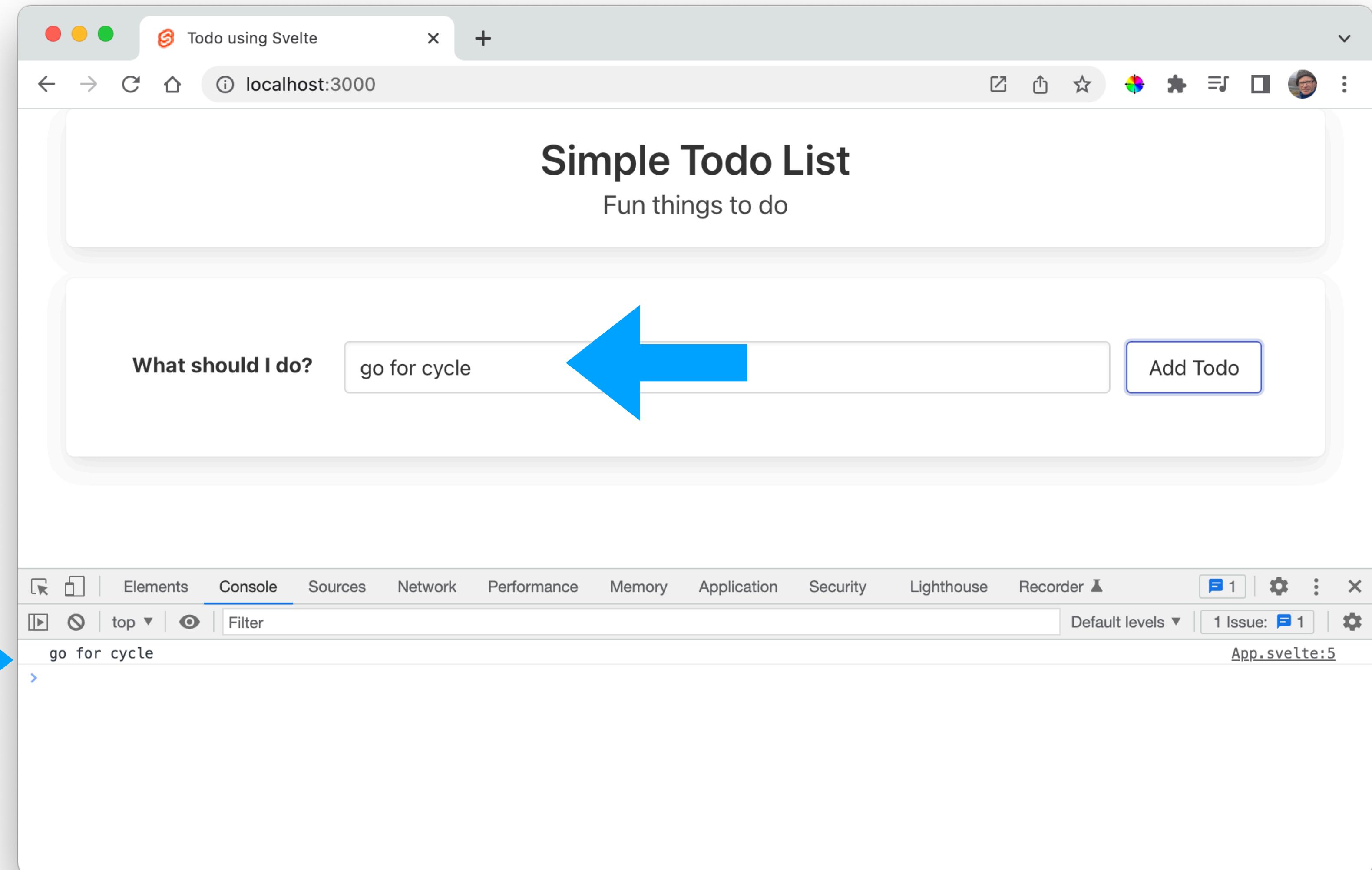
```
<button onclick={() => addTodo()} class="button">Add Todo</button>
```



A screenshot of a simple todo application interface. It features a text input field with the placeholder "Type something...", a button labeled "Add Todo", and a text area labeled "What should I do?".

Binding Behaviour

```
<script>  
  let todoText = "";  
  
  function addTodo() {  
    console.log(todoText);  
  }  
</script>
```



Debugging

Application
Structure +
Source
Debugging
available

The screenshot shows a browser window titled "Todo Svelte" at "localhost:5000". The page displays a simple todo list with the heading "Fun things to do" and a card asking to "Enter todo item". Below it is a text input field with "Go for run" and a button labeled "ADD TODO". The browser's developer tools are open, specifically the "Sources" tab, which is currently selected. The "bundle.css" and "index.mjs" tabs are also visible. The code editor shows a script block with the following content:

```
<script>
  let todoText = "";
  function addTodo() {
    console.log(todoText)
  }
</script>

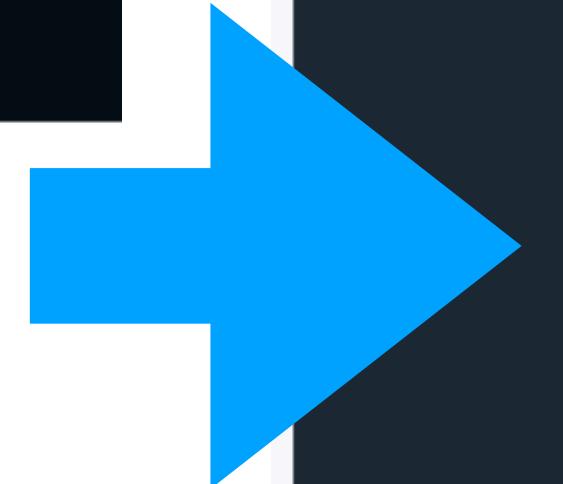
<div class="uk-container">
  <div class="uk-flex uk-flex-center uk-flex-middle uk-width-2-3@m">
    <div class="uk-card uk-card-default">
      <div class="title"> Simple Todo List </div>
      <div class="text-muted uk-text-small"> Fun </div>
    </div>
  </div>
</div>
```

The line `4 console.log(todoText)` is highlighted with a blue selection bar. The right panel of the developer tools shows the "Breakpoints" section with a checked breakpoint on this line. The "Scope" section shows a "todoText" variable with the value "Go for run" in red, indicating it is being watched. The status bar at the bottom indicates "Paused in debugger" and "Line 4, Column 3".

```
let todoText = "";
let todoItems = [];

function addTodo() {
  console.log(todoText)
  todoItems.push(todoText);
}
```

Simple template style
syntax to iterate array

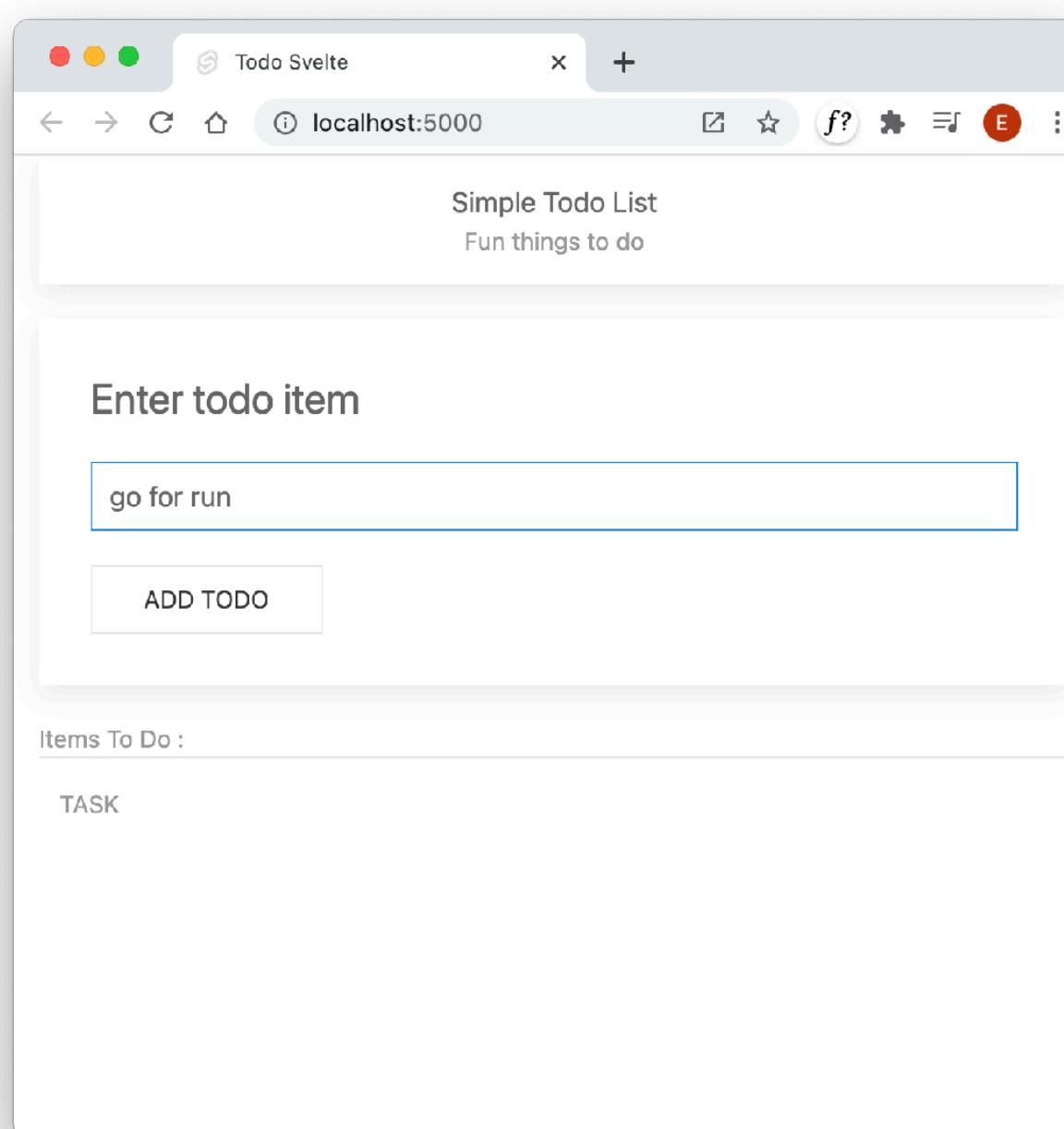


```
<table class="uk-table uk-table-divider">
  <caption> Items To Do :</caption>
  <thead>
    <tr>
      <th>Task</th>
    </tr>
  </thead>
  <tbody>
    {#each todoItems as todo}
    <tr>
      <td> {todo} </td>
    </tr>
  {/each}
  </tbody>
</table>
```

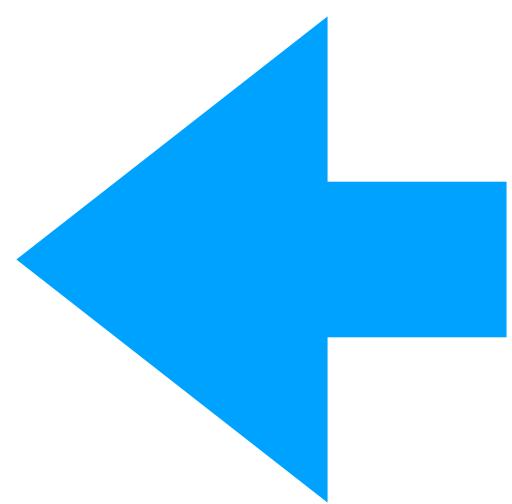
Svelte Template Syntax

Binding Behaviour - Arrays

```
let todoText = "";  
let todoItems = [];  
  
function addTodo() {  
  console.log(todoText)  
  todoItems.push(todoText);  
}
```



```
<table class="uk-table uk-table-divider">  
  <caption> Items To Do :</caption>  
  <thead>  
    <tr>  
      <th>Task</th>  
    </tr>  
  </thead>  
  <tbody>  
    {#each todoItems as todo}  
    <tr>  
      <td> {todo} </td>  
    </tr>  
   {/each}  
  </tbody>  
</table>
```



However, fails to update list!

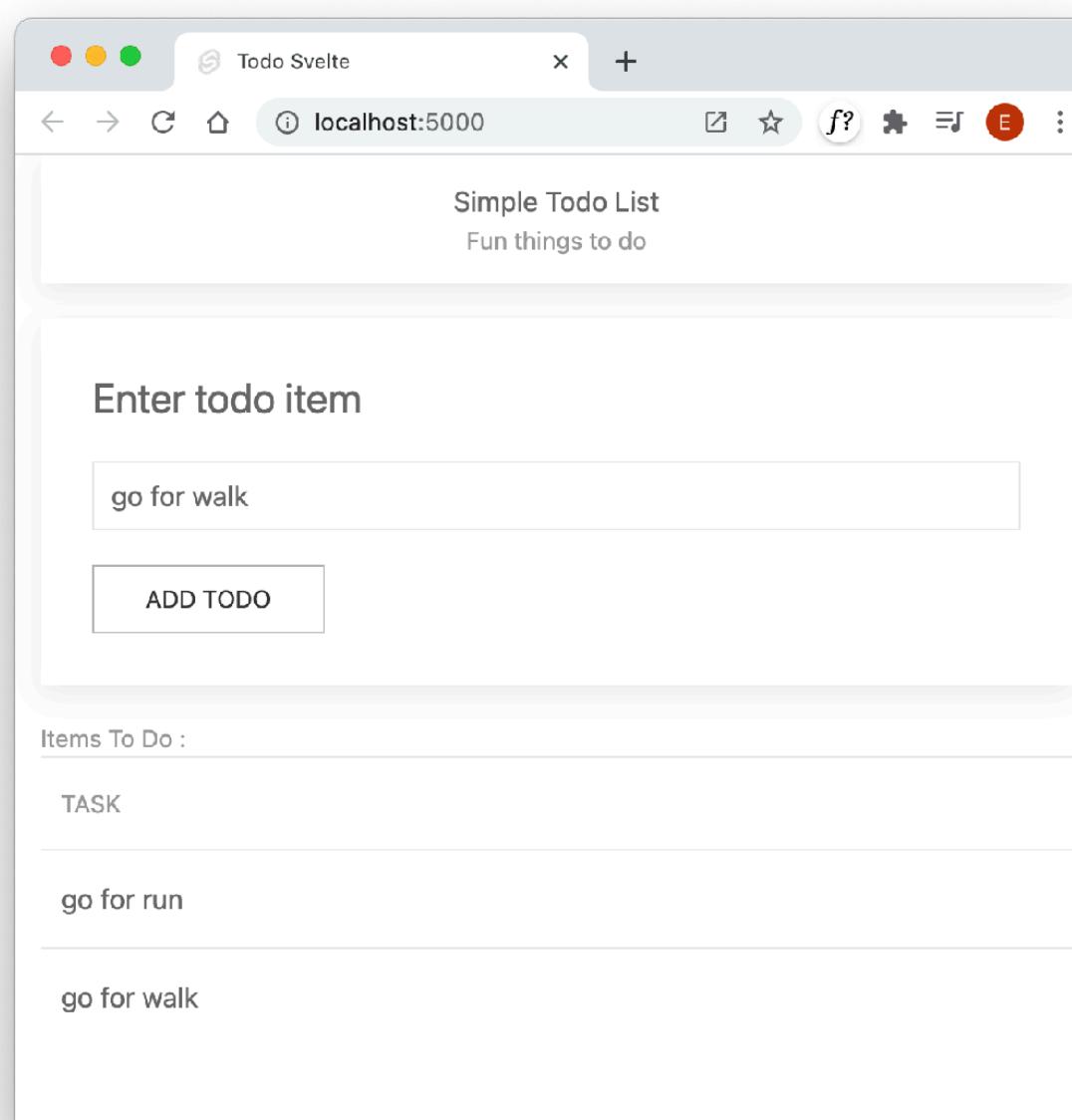
State Rune

```
let todoText = "";  
let todoItems = [];  
  
function addTodo() {  
    console.log(todoText)  
    todoItems.push(todoText);  
}
```

```
let todoItems = $state([]);
```

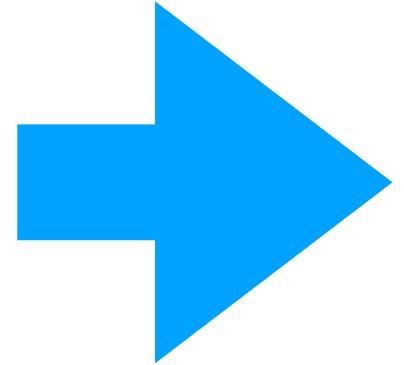
This is called a *rune*, and it's how you tell Svelte that todoItems isn't an ordinary variable

It is a special variable, whose values will be automatically reflected in the DOM



So when items are pushed into the array
- they will also trigger an update to the page.

Append the string in todoText to the todoItems array



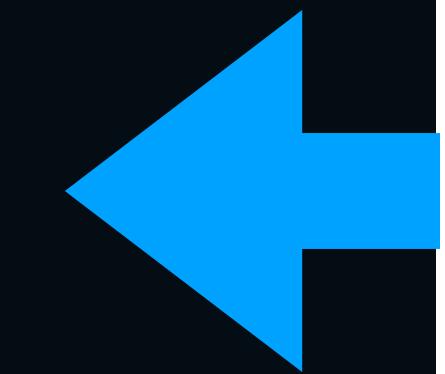
```
<script>
  let todoText = "";
  let todoItems = $state([]);

  function addTodo() {
    console.log(todoText);
    todoItems.push(todoText);
  }
</script>
```

```
<div class="field-body">
  <div class="field">
    <p class="control">
      <input
        bind:value={todoText}
        id="todo"
        class="input"
        type="text"
        placeholder="Type something...">
    </p>
  </div>
  <button onclick={() => addTodo()} class="button">Add Todo</button>
```

Pressing the button triggers the addTodo() function

```
<div class="section box">
  <div class="title is-6">Things yet do</div>
  <table class="table is-fullwidth">
    <thead>
      <tr>
        <th>Task</th>
      </tr>
    </thead>
    <tbody>
      {#each todoItems as todo}
        <tr>
          <td> {todo} </td>
        </tr>
      {/each}
    </tbody>
  </table>
</div>
```



If the array is updated, this loop reruns updating the DOM

Todo Object

```
function addTodo() {  
  const todo = {  
    text: todoText,  
    date: new Date().toLocaleString("en-IE"),  
  };  
  todoItems.push(todo);  
  todoText = "";  
}
```

Items To Do :

TASK	DATE
go for run	13/4/2021, 16:34:08
go for walk	13/4/2021, 16:34:14

```
<div class="section box">  
  <div class="title is-6">Things yet do</div>  
  <table class="table is-fullwidth">  
    <thead>  
      <tr>  
        <th>Task</th>  
        <th>Date</th>  
      </tr>  
    </thead>  
    <tbody>  
      {#each todoItems as todo}  
        <tr>  
          <td>{todo.text}</td>  
          <td>{todo.date}</td>  
        </tr>  
      {/each}  
    </tbody>  
  </table>  
</div>
```

A screenshot of a browser window showing a "Simple Todo List" application. The page title is "Simple Todo List" with the subtitle "Fun things to do". A text input field contains "go for a walk" and an "Add Todo" button is visible. Below this, a table titled "Things yet do" lists one item: "Task" (go for a cycle) and "Date" (21/2/2025, 09:21:22). The browser's developer tools are open, specifically the "Sources" tab, showing the code for "viewKCM.js" and "+page.svelte". The "+page.svelte" file contains Svelte code. A yellow arrow points to line 10 of the "+page.svelte" code, which is "todoItems.push(todo);". The right panel of the developer tools shows the "Breakpoints" section with a checked breakpoint for this line. The "Scope" panel shows the variable "todo" with its properties: "date: '21/2/2025, 09:21:31'" and "text: 'go for a walk'".

```
<script>
let todoText = $state("");
let todoItems = $state([]);

function addTodo() {
    const todo = { todo: {text: 'go for a walk', date: '21/2/2025, 09:21:31'}
    text: todoText, todoText: {f: 0, v: 'go for a walk', reactions: Array(1), rv: 57, date: new Date().toLocaleString("en-IE")};
};

todoItems.push(todo);
todoText = "";
}

</script>

<div class="container">
<div class="box has-text-centered">
<div class="title">Simple Todo List</div>
<div class="subtitle">Fun things to do</div>
</div>
</div>

<div class="section box">
<div class="field is-horizontal">
<div class="field-label is-normal">
<label for="todo" class="label">What should I do?</label>
</div>
<div class="field-body">
<div class="field">
<p class="control">
<input type="text" value="go for a walk" />
<button>Add Todo</button>

```

Debugging

Inspect Data Structure

Done List - UX

```
{#each todoItems as todo}  
  <tr>  
    <td> {todo.text} </td>  
    <td> {todo.date} </td>  
    <td><button onclick={() => deleteTodo(todo.id)} class="button">delete</button></td>  
  </tr>  
{/each}
```

```
<div class="section box">  
  <div class="title is-6">Things done</div>  
  <table id="done-table" class="table is-fullwidth">  
    <thead>  
      <tr>  
        <th>Task</th>  
        <th>Date</th>  
      </tr>  
    </thead>  
    <tbody>  
      #each doneItems as todo  
        <tr>  
          <td> {todo.text} </td>  
          <td> {todo.date}</td>  
        </tr>  
    </tbody>  
</table>  
</div>  
</div>
```

Items To Do :	
TASK	DATE
Learn Svelte	23/3/2021, 13:34:55
Learn React	23/3/2021, 13:35:03
Items Completed :	
TASK	DATE
go for walk	23/3/2021, 13:34:48

Done List - Behaviour

Declare Items

Create todo, including
date + id

Push on to todoltems
array

Add deleted Todos to
doneitems

```
<script>

import { v4 as uuidv4 } from "uuid";
let todoText = $state("");
let todoItems = $state([]);
let doneItems = $state([]);

function addTodo() {
  const todo = {
    text: todoText,
    date: new Date().toLocaleString("en-IE"),
    id: uuidv4()
  };
  todoItems.push(todo);
  todoText = "";
}

function deleteTodo(id) {
  doneItems.push(todoItems.find((todo) => todo.id === id));
  todoItems = todoItems.filter((todo) => todo.id !== id);
}

</script>
```

Project Structure

FOLDERS: TODO-SVEL...

The screenshot shows a file explorer window with the title "FOLDERS: TODO-SVEL...". The tree view displays the following directory structure:

- ✓ .svelte-kit
 - > generated
 - > output
 - TS ambient.d.ts
 - TS non-ambient.d.ts
 - TS tsconfig.json
- > node_modules
- ✓ src
 - > lib
 - JS index.js
 - > routes
 - S +page.svelte
 - HTML app.html
 - > static
 - IMG favicon.png
 - SH cursorrules
 - RD gitignore
 - NPM .npmrc
 - RD .prettierignore
 - RD .prettierrc
 - INP eslint.config.js
 - JS jsonconfig.json
 - NPM package-lock.json
 - NPM package.json
 - M README.md
 - S svelte.config.js
 - V vite.config.js

Project Structure

Generated (bundled) application

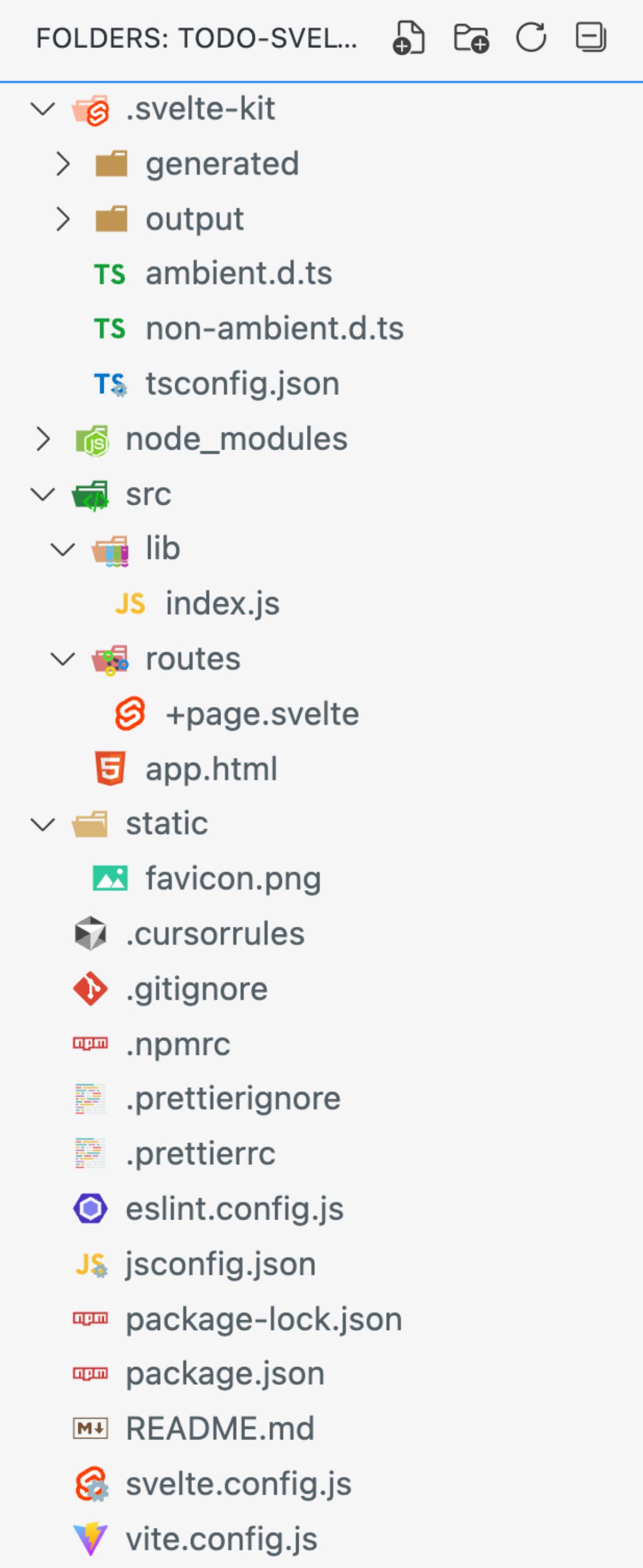
Application Modules

Shared Components

Application Source

Static resources

Configuration files



Project Structure

Project Structure

Generated (bundled) application

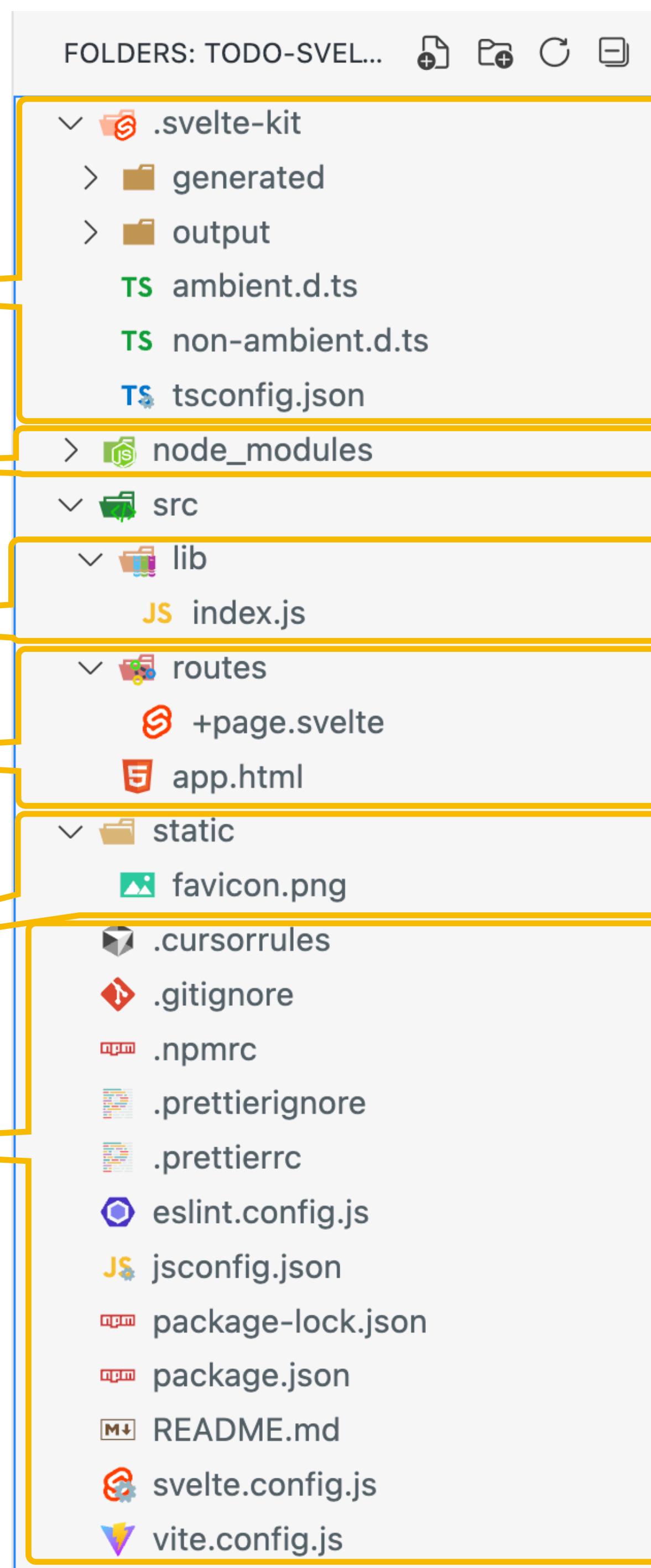
Application Modules

Shared Components

Application Source

Static resources

Configuration files



Svelte First Steps



Building your first Svelte
Application