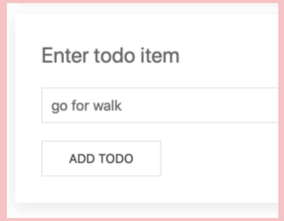
Svelte First Steps

Building your first Svelte Application

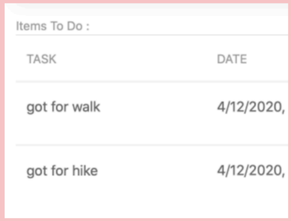
Todo Example - Recap

Lab-13a-Todo 1



Build a simple todo app in javascript.

Lab-13b-Todo 2



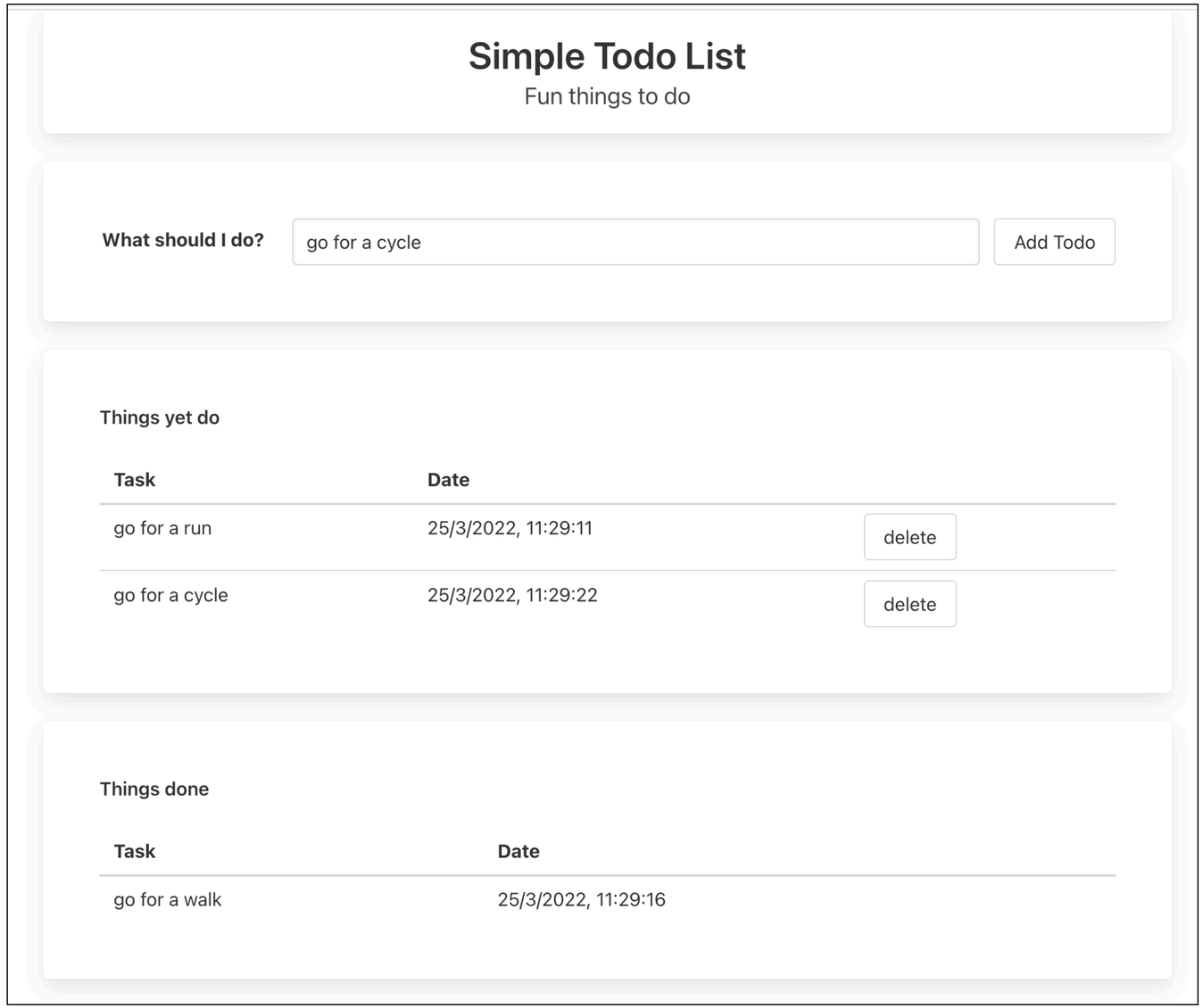
Evolve to Todo app further.

Lab-14-Todo 1

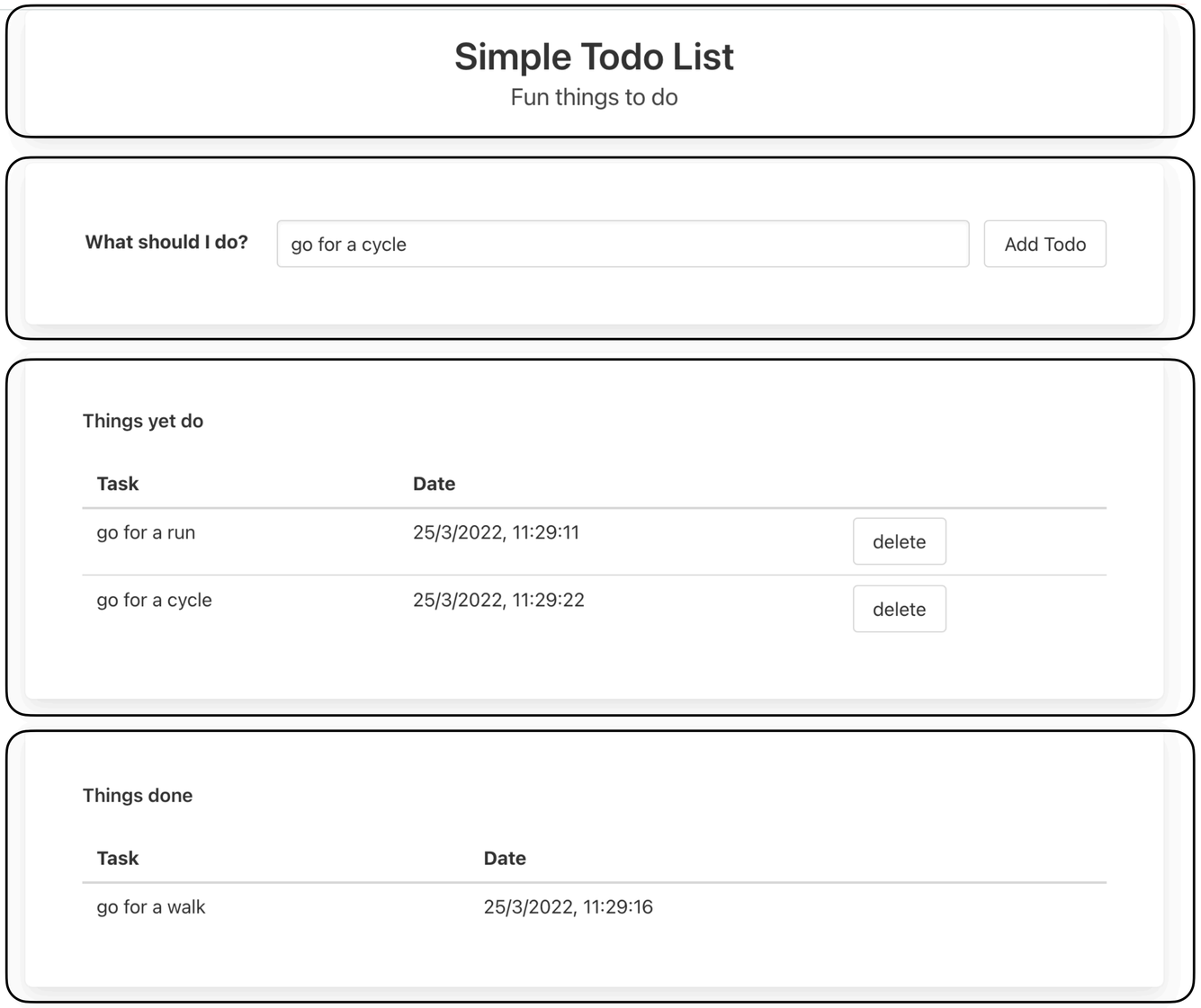
Create a first Svelte app.

Lab-15-Todo 2

Explore Svelte components in the Todo app.



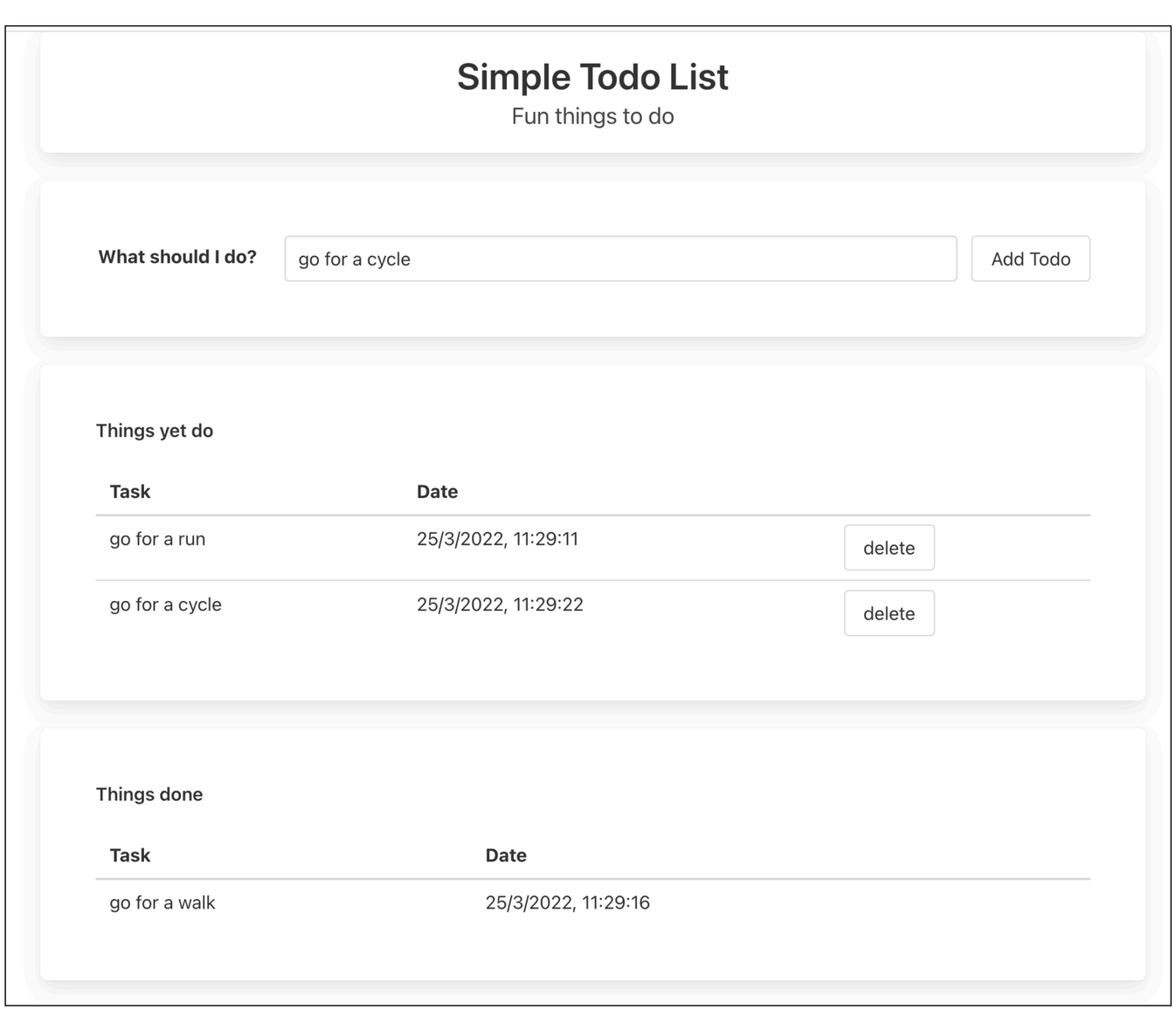
Todo UX




```
<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
        </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>
```

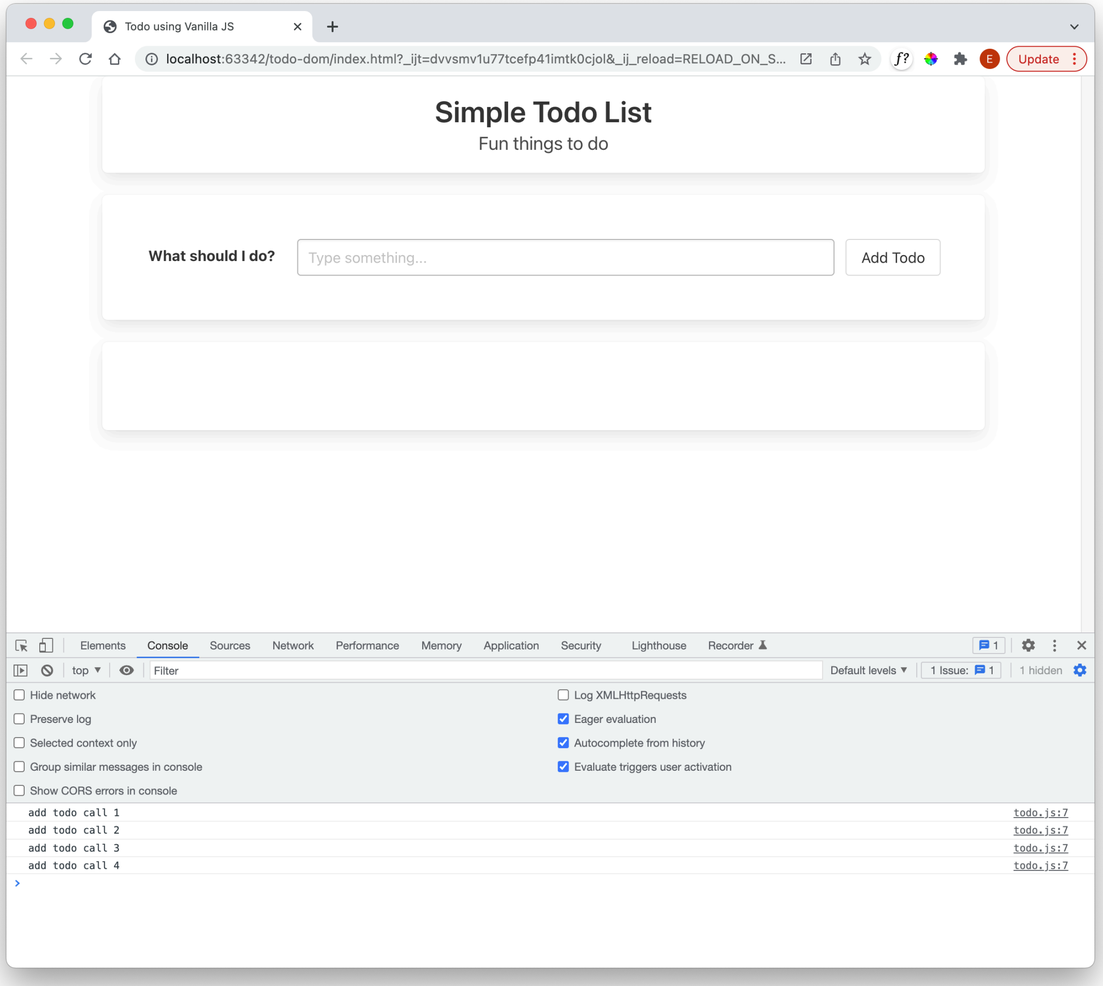


Incorporate Javascript into page

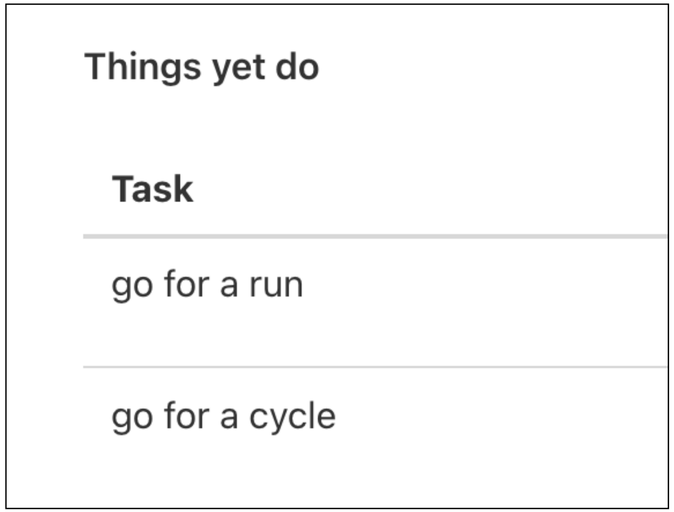
onClick

```
<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




```
<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;
}
```

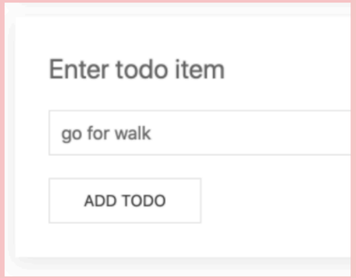
Retrieve Table DOM Object

DOM Interaction

- Rosetta Stone approach

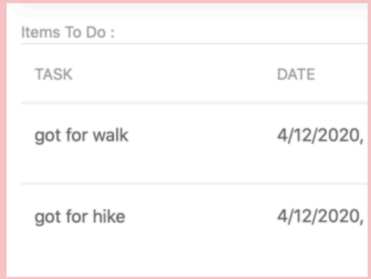
- Develop the same application(s) in Svelte

Lab-13a-Todo 1



Build a simple todo app in javascript.

Lab-13b-Todo 2



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.

Create Svelte App

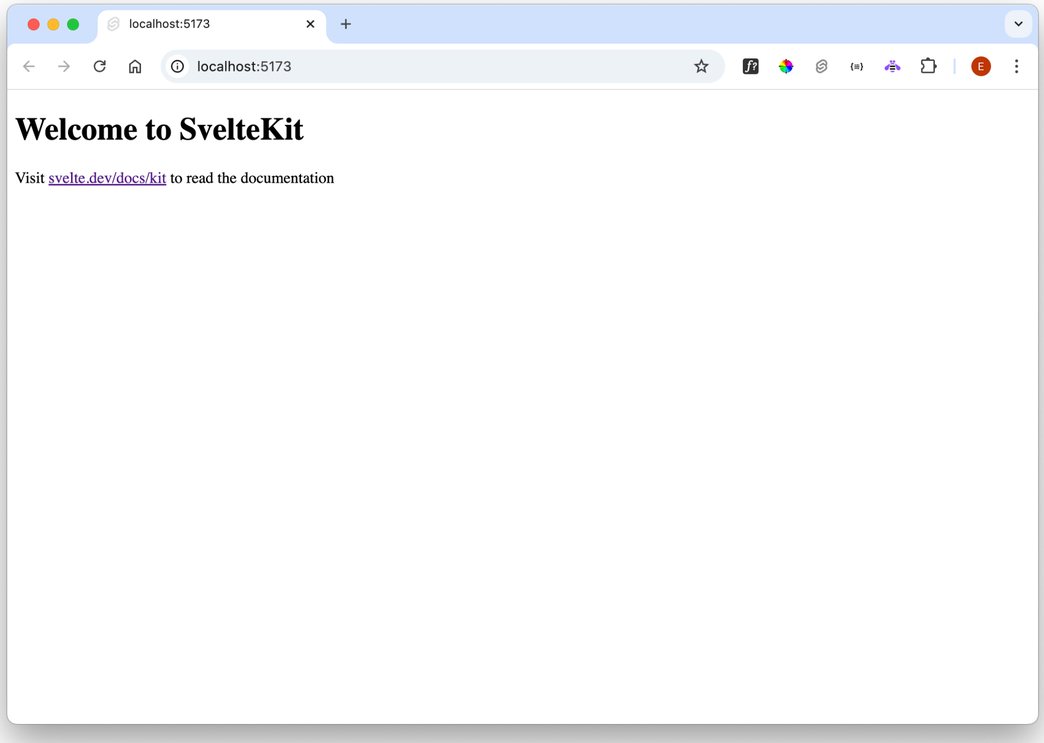
```
npx sv create todo-svelte
```

```
┌  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
```

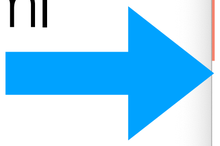
```
cd todo-svelte
npm run dev
```

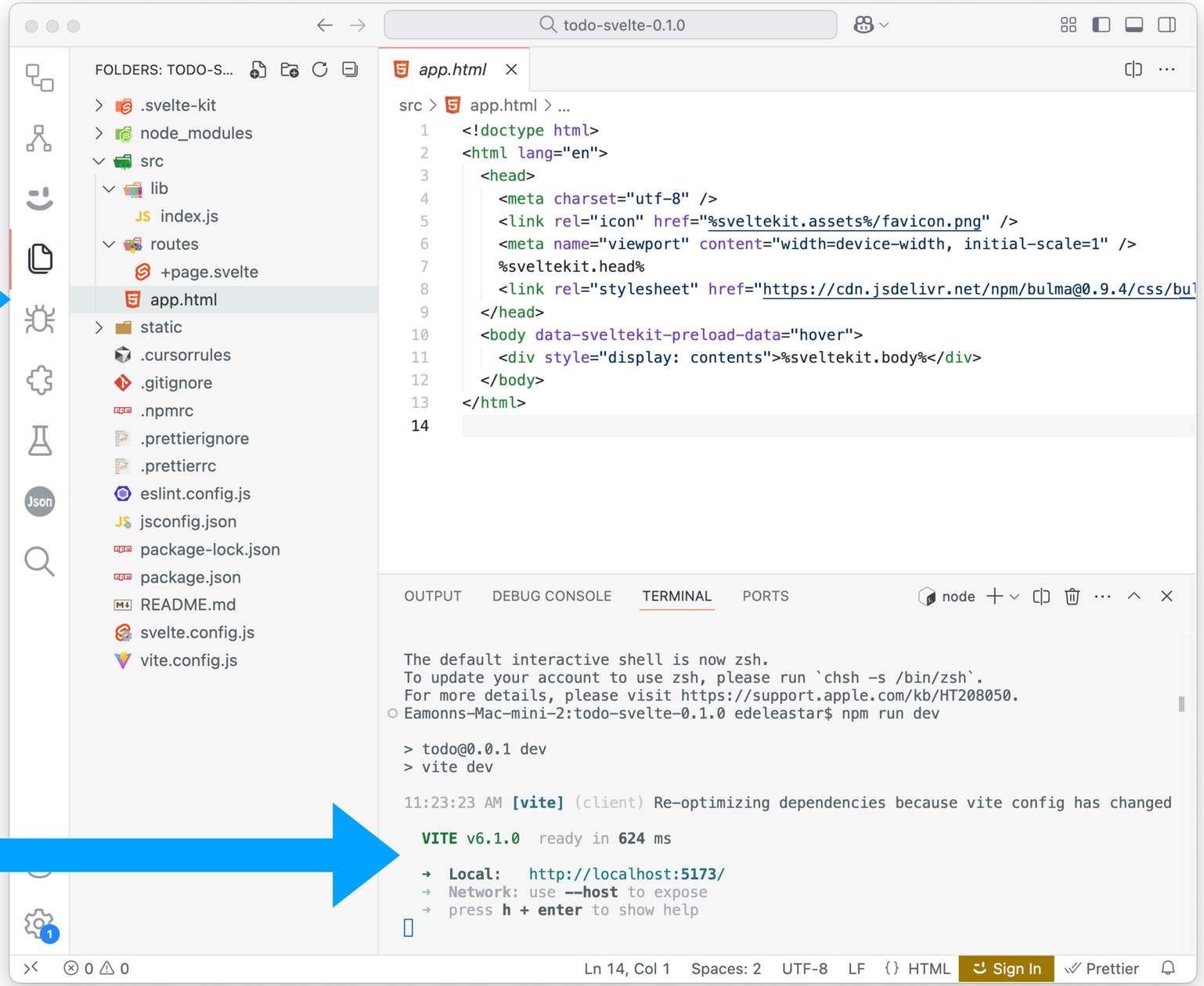
```
> todo@0.0.1 dev
> vite dev

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
```

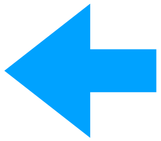
http://localhost:5173



App.html







+page.svelte

Application Home Page

Svelte application launched (on front end)

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>
```

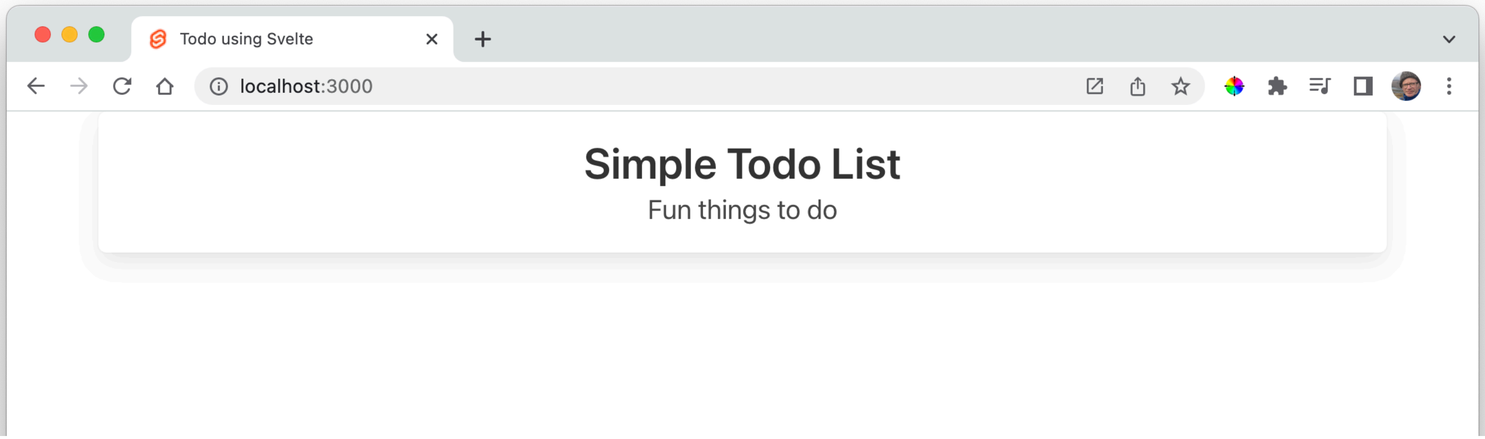
+routes/+page.svelte

Component Code (empty)

Component UI

Running Application

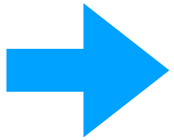
```svelte
<script>
</script>

<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>
```



Incorporating Form UX

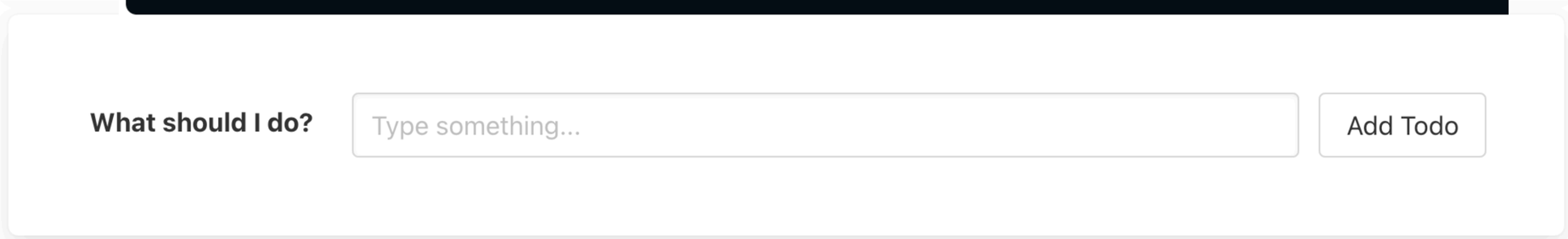
routes/+.page.svelte

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)



Component Logic

Simple Variable + function

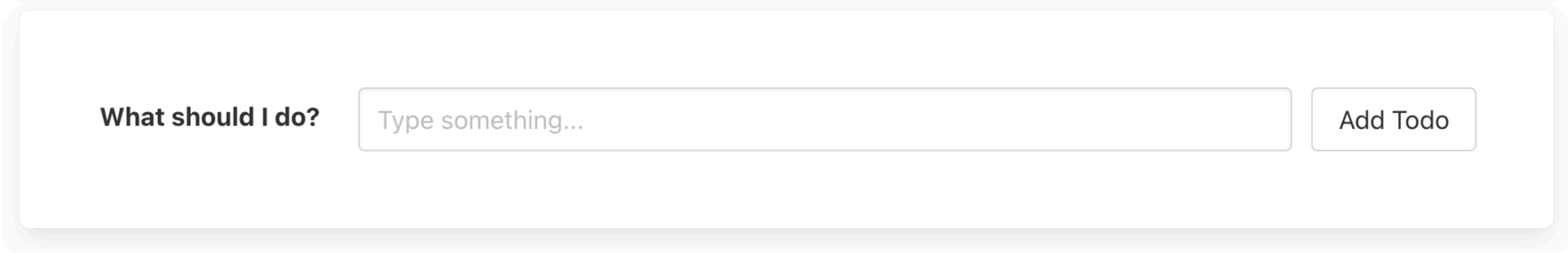
```
<script>
  let todoText;
  function addTodo() {
    console.log(todoText)
  }
</script>
```

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

“Bind” the variable to the input field

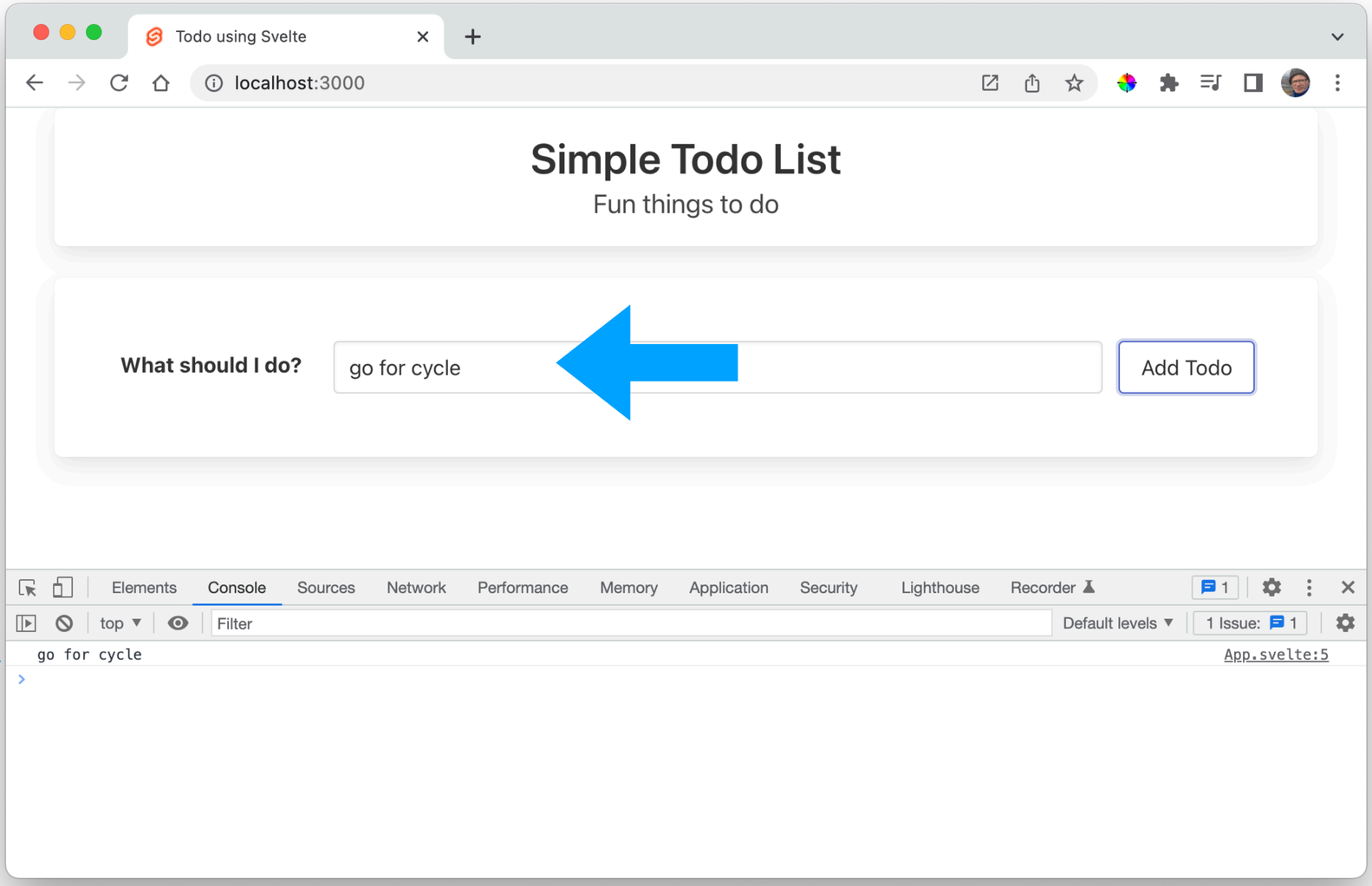
Trigger function when button pressed

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



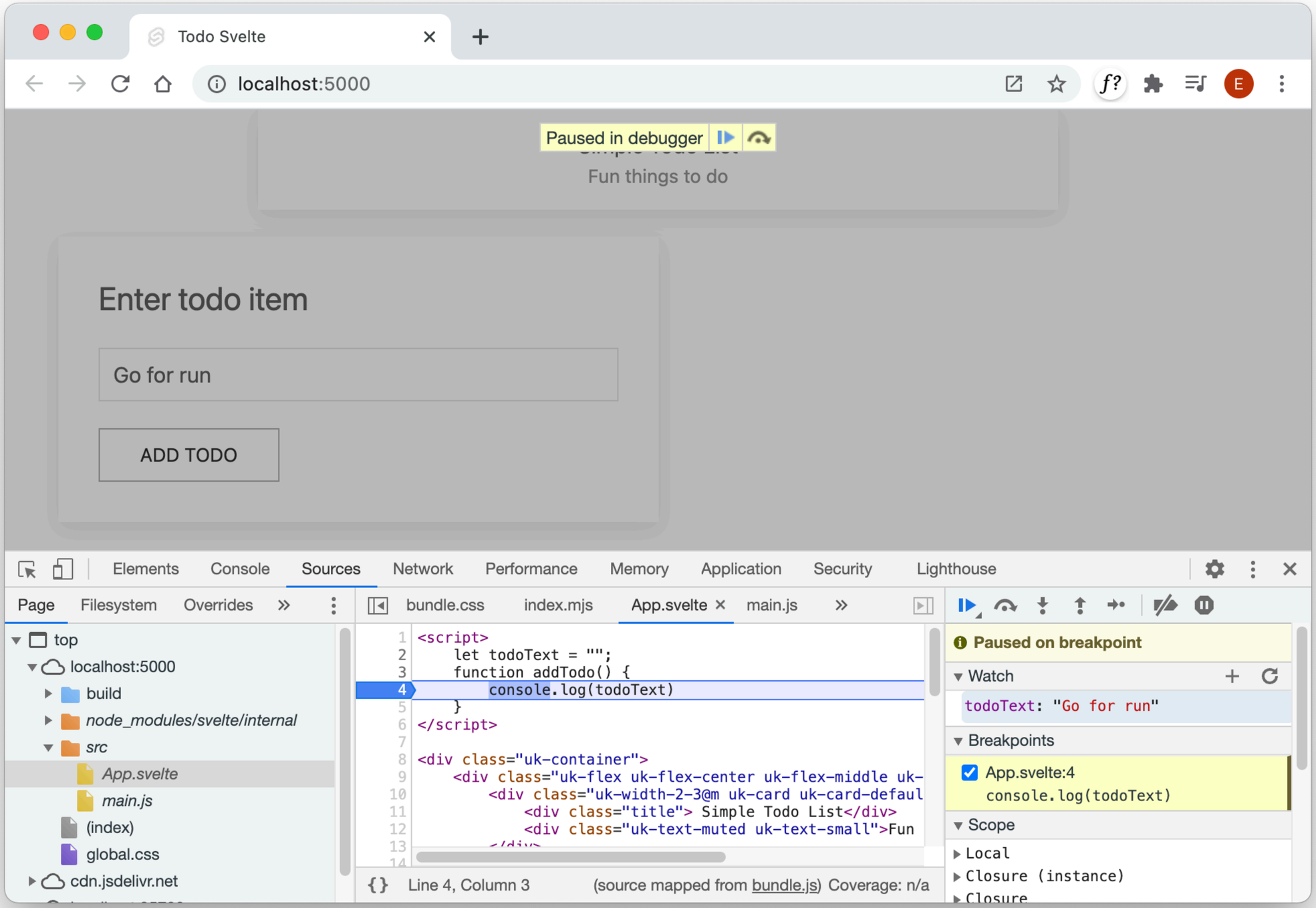
Binding Behaviour

```
<script>
  let todoText = "";

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



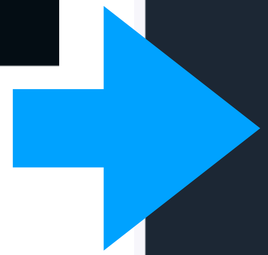
Debugging

Application Structure + Source Debugging available



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

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

Simple template style syntax to iterate array



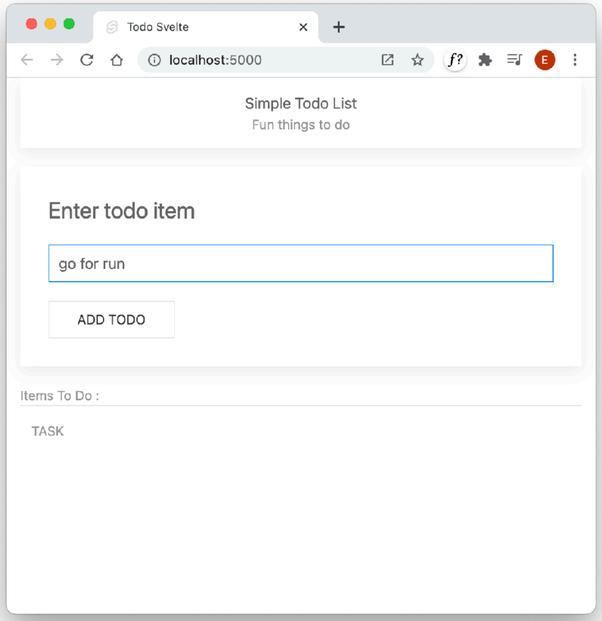
```
<table class="uk-table uk-table-divider">
  <caption> Items To Do :</caption>
  <thead>
    <tr>
      <th>Task</th>
    </tr>
  </thead>
  <tbody>
    {#each todoItems as todo}
      <tr>
        <td> {todo} </td>
      </tr>
    {/each}
  </tbody>
</table>
```

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

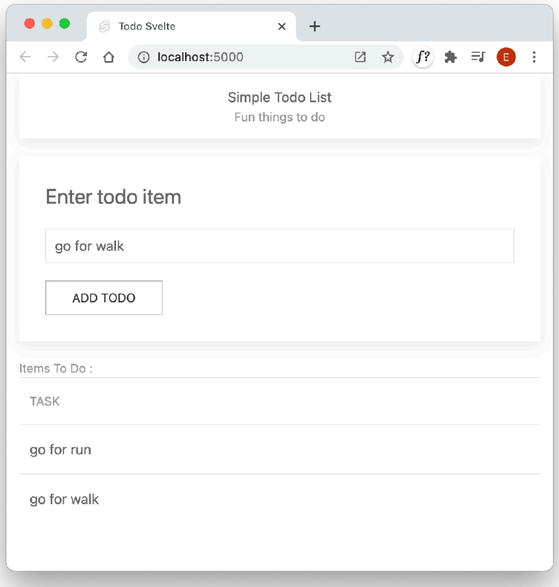
```
let todoText = "";
let todoItems = [];

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

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

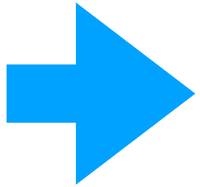
This is called a *rune*, and it's how you tell Svelte that **todoItems** isn't an ordinary variable

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

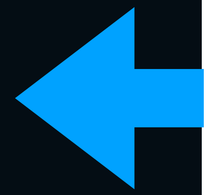


```
<script>
  let todoText = "";
  let todoItems = $state([]);

  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>
</div>
```



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

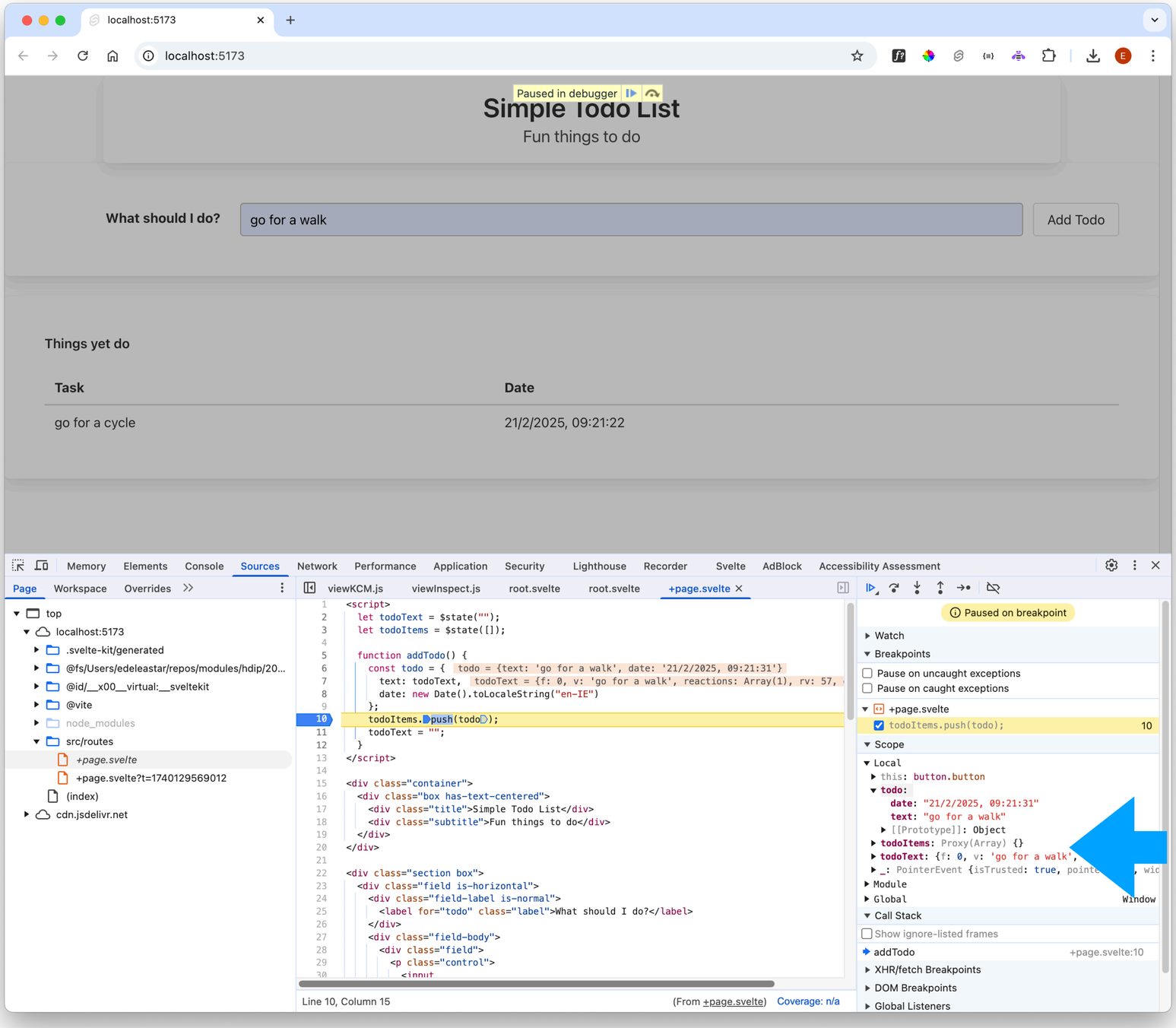
Todo Object

```
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>
```



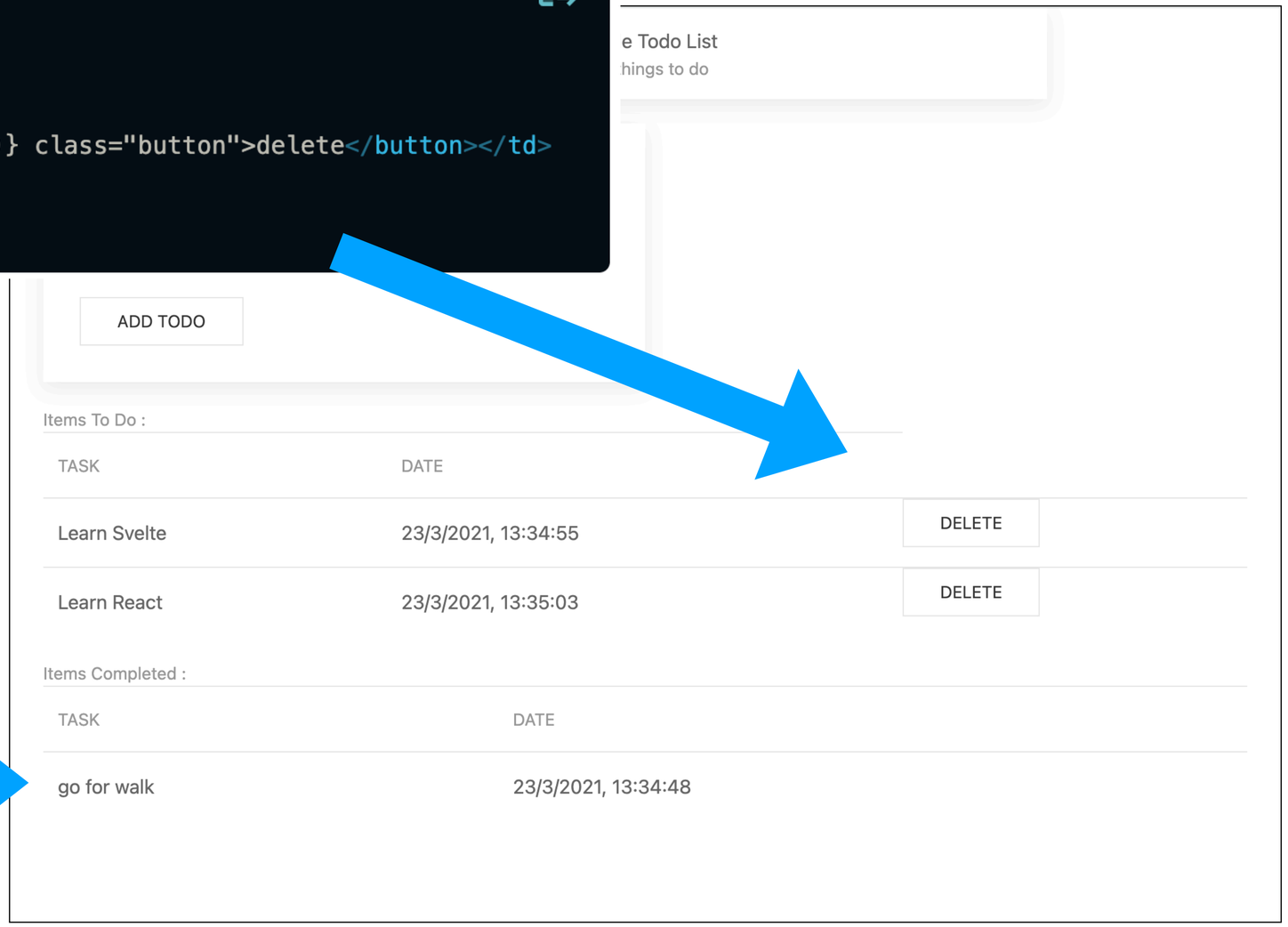
Debugging

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>
      {/each}
    </tbody>
  </table>
</div>
</div>
```



Done List - Behaviour

Declare Items

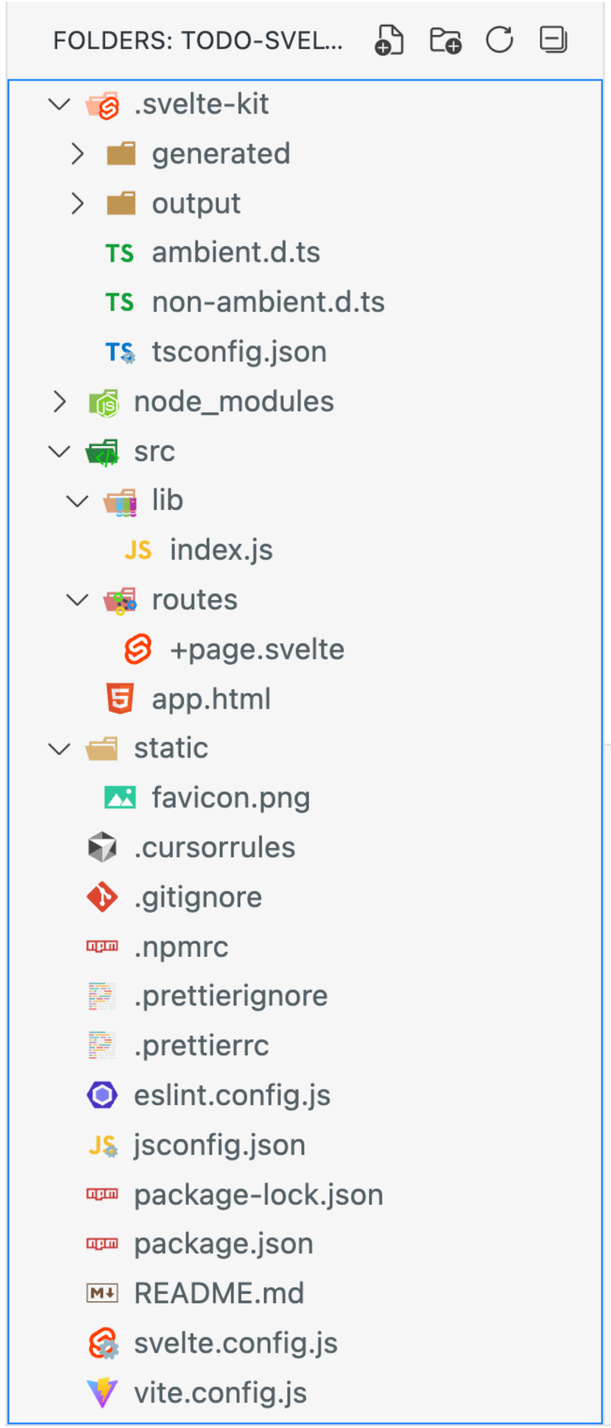
Create todo, including date + id

Push on to todoItems 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



Project Structure

Generated (bundled) application

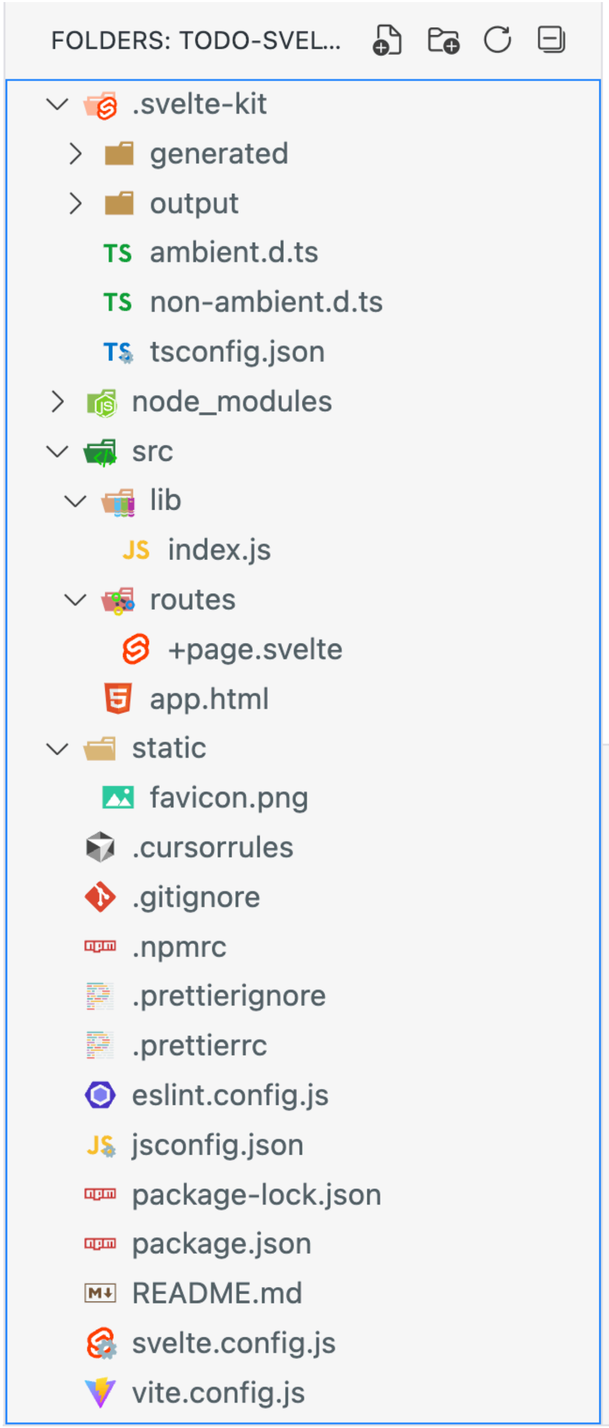
Application Modules

Shared Components

Application Source

Static resources

Configuration files



Project Structure

Generated (bundled) application

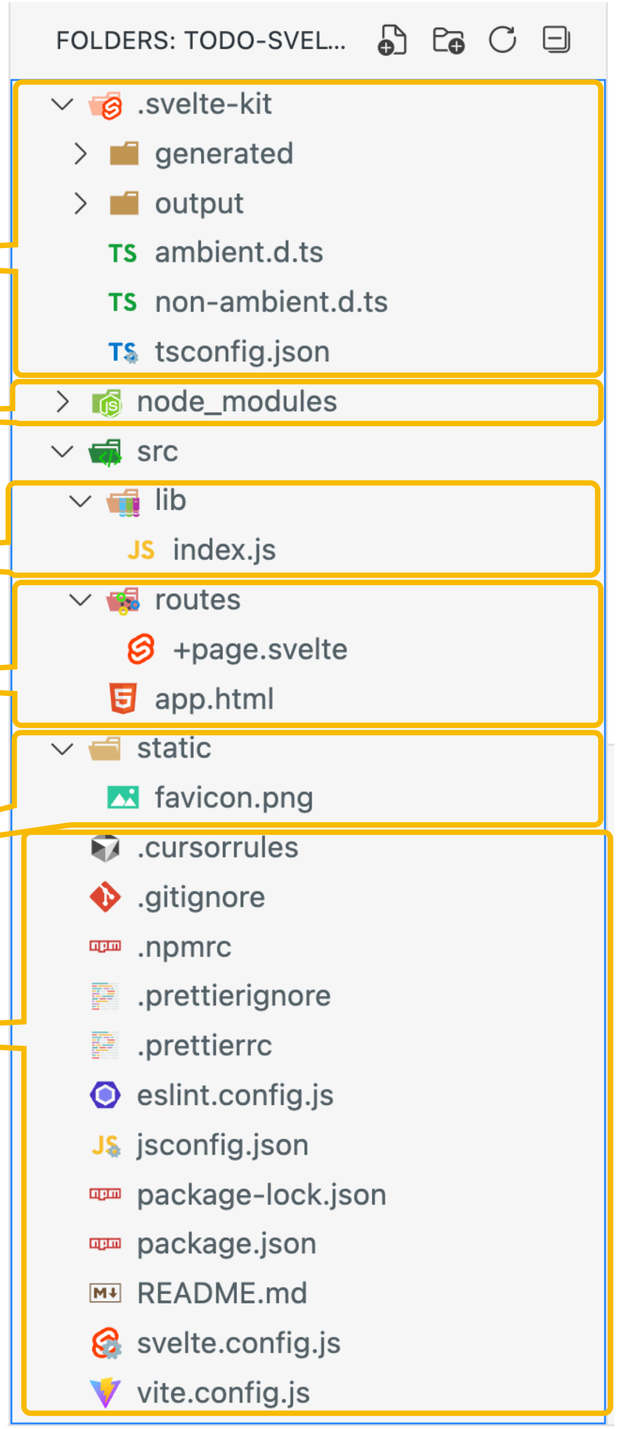
Application Modules

Shared Components

Application Source

Static resources

Configuration files



Project Structure

Svelte First Steps

Building your first Svelte Application