Svelte First Steps

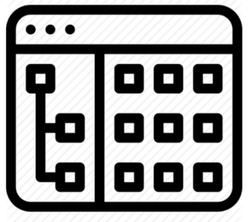
Building your first Svelte Application

Todo Example - Recap

The Browser Environment

The Web Browser Programming Model

Todo DOM



Manipulating the DOM to implement TODO features

Lab-12a-Todo 1

Enter todo item

go for walk

ADD TODO

Build a simple todo app in javascript.

Lab-12b-Todo 2

Items To Do :

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

Evolve to Todo app further.

Simple Todo List

Fun things to do

What should I do? go for a cycle Add Todo

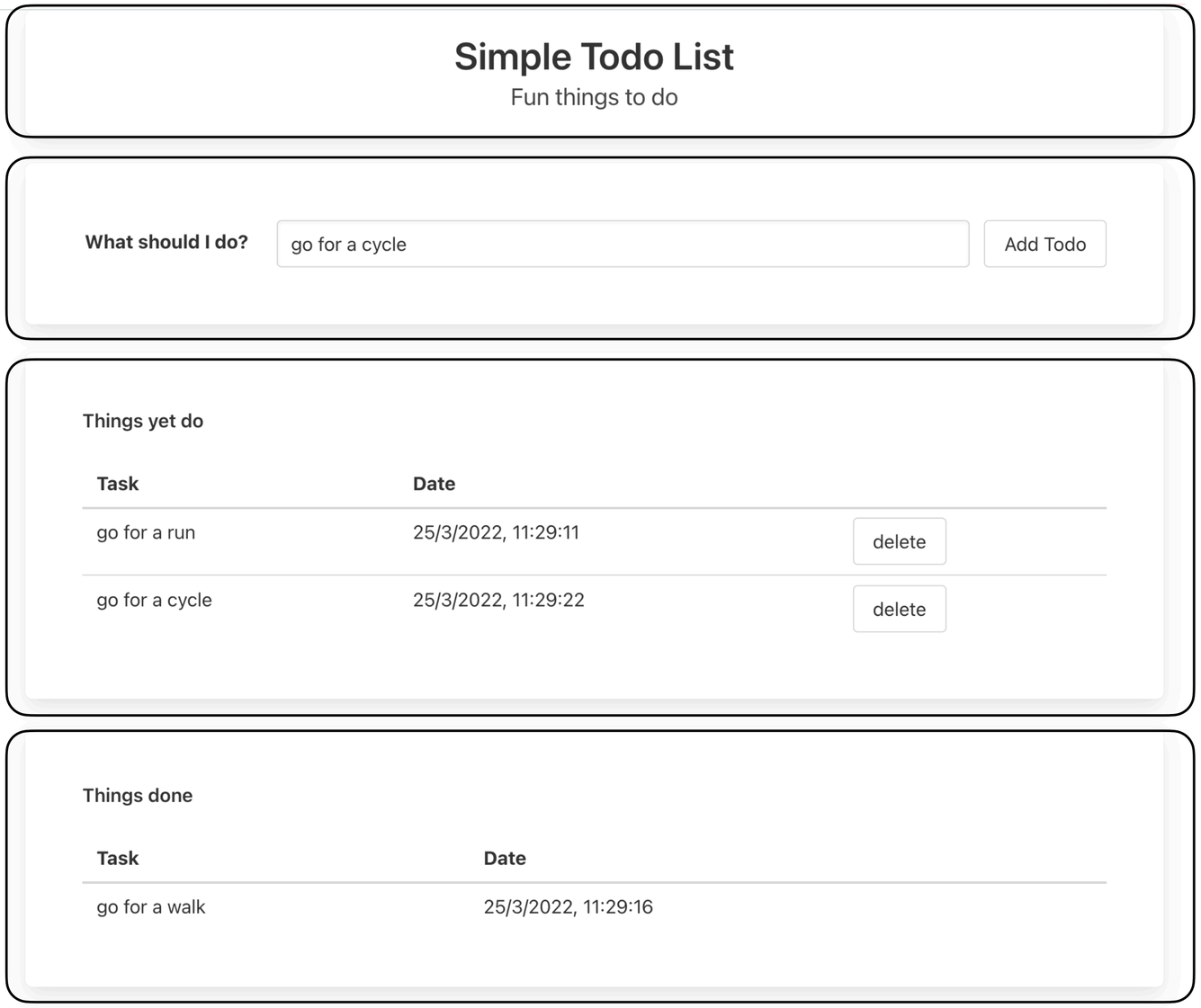
Things yet do

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

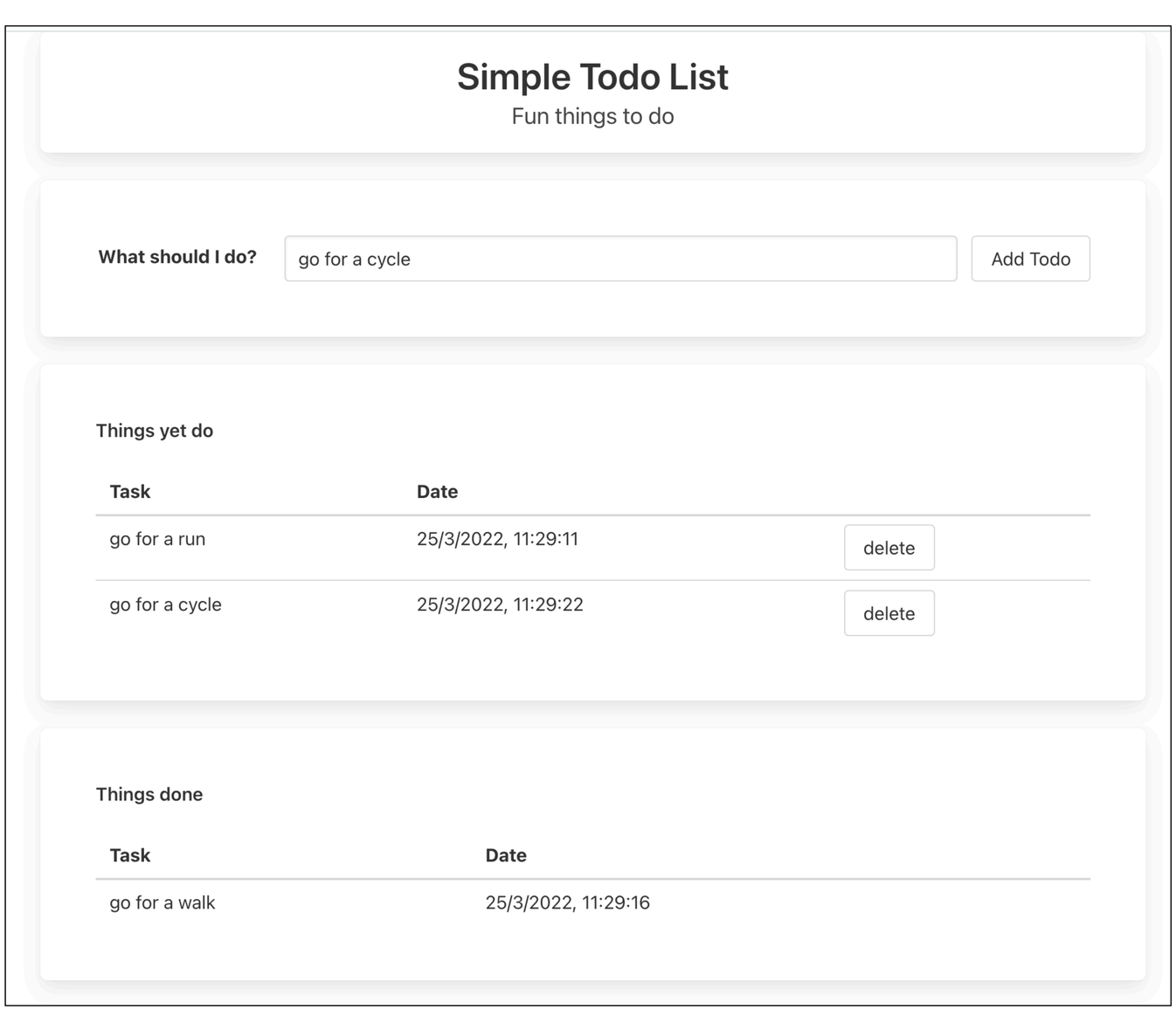
Todo UX




```
<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>
```

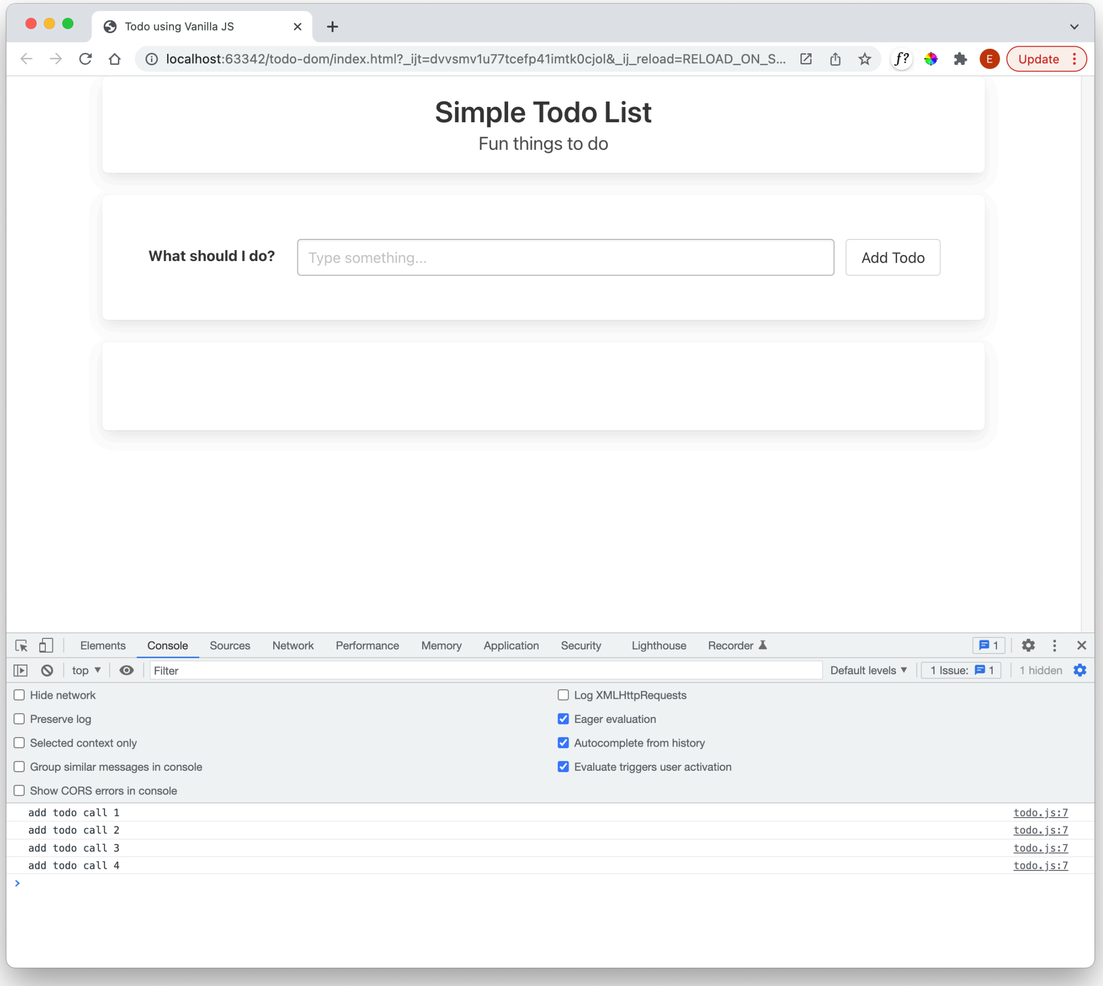


Incorporate Javascript into page

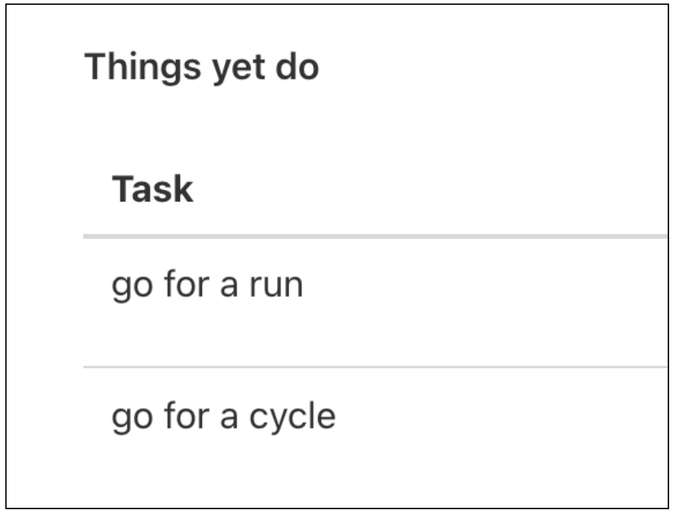
onClick

```
<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-12a-Todo 1

Enter todo item

go for walk

ADD TODO

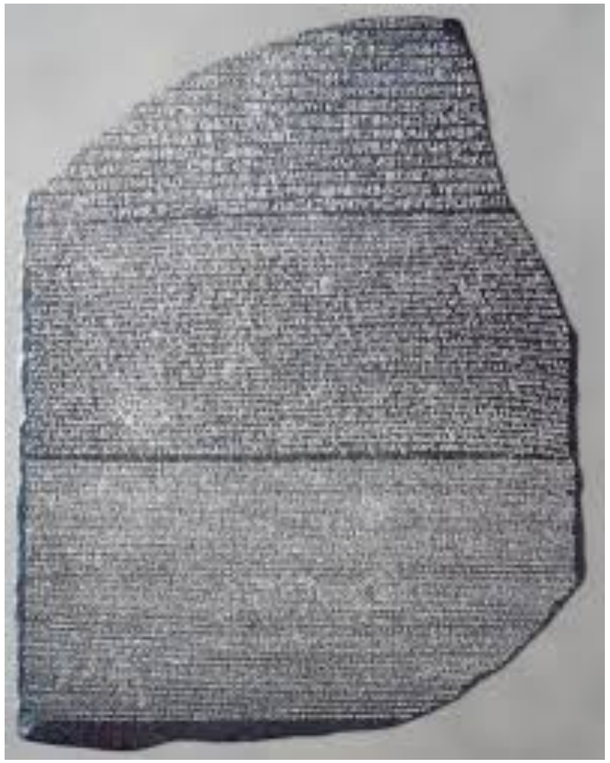
Build a simple todo app in javascript.

Lab-12b-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-13a-Todo 1

Create a first Svelte app.

Lab 13b-Todo 2

Explore Svelte components in the Todo app.

```
npm init vite@latest
```

In the subsequent menus, select **svelte** and call the project todo-svelte

```
Need to install the following packages:
  create-vite@latest
Ok to proceed? (y) y
✔ Project name: … todo-svelte
✔ Select a framework: › svelte
✔ Select a variant: › svelte
```

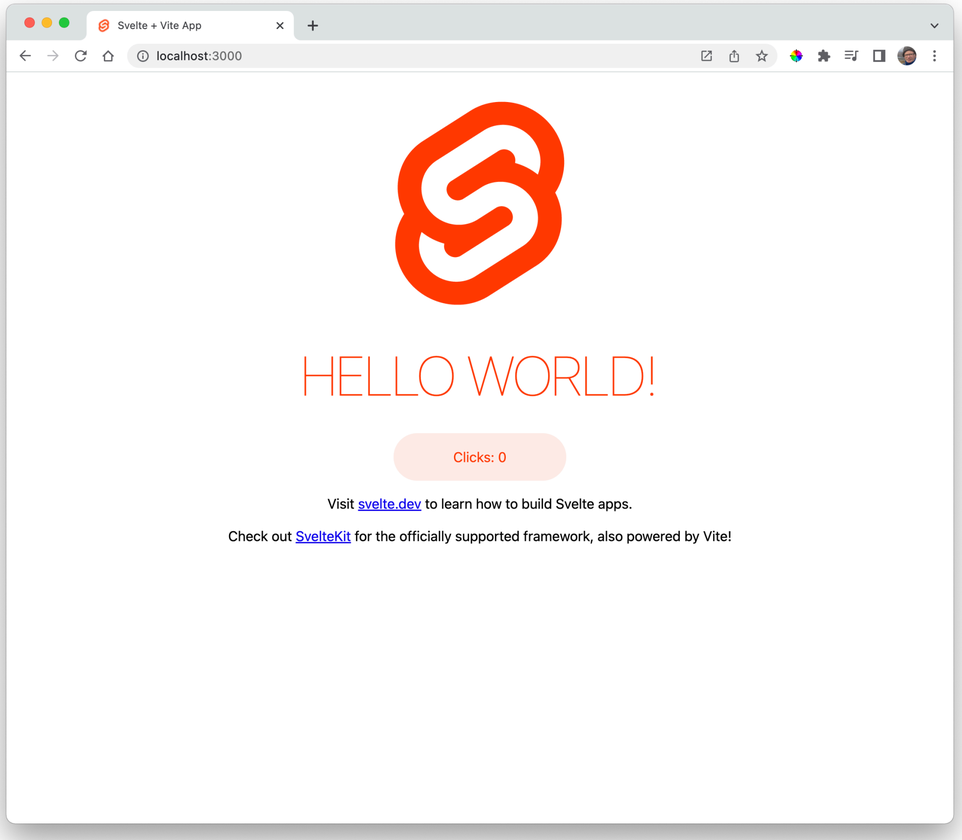
Now we can run the application:

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

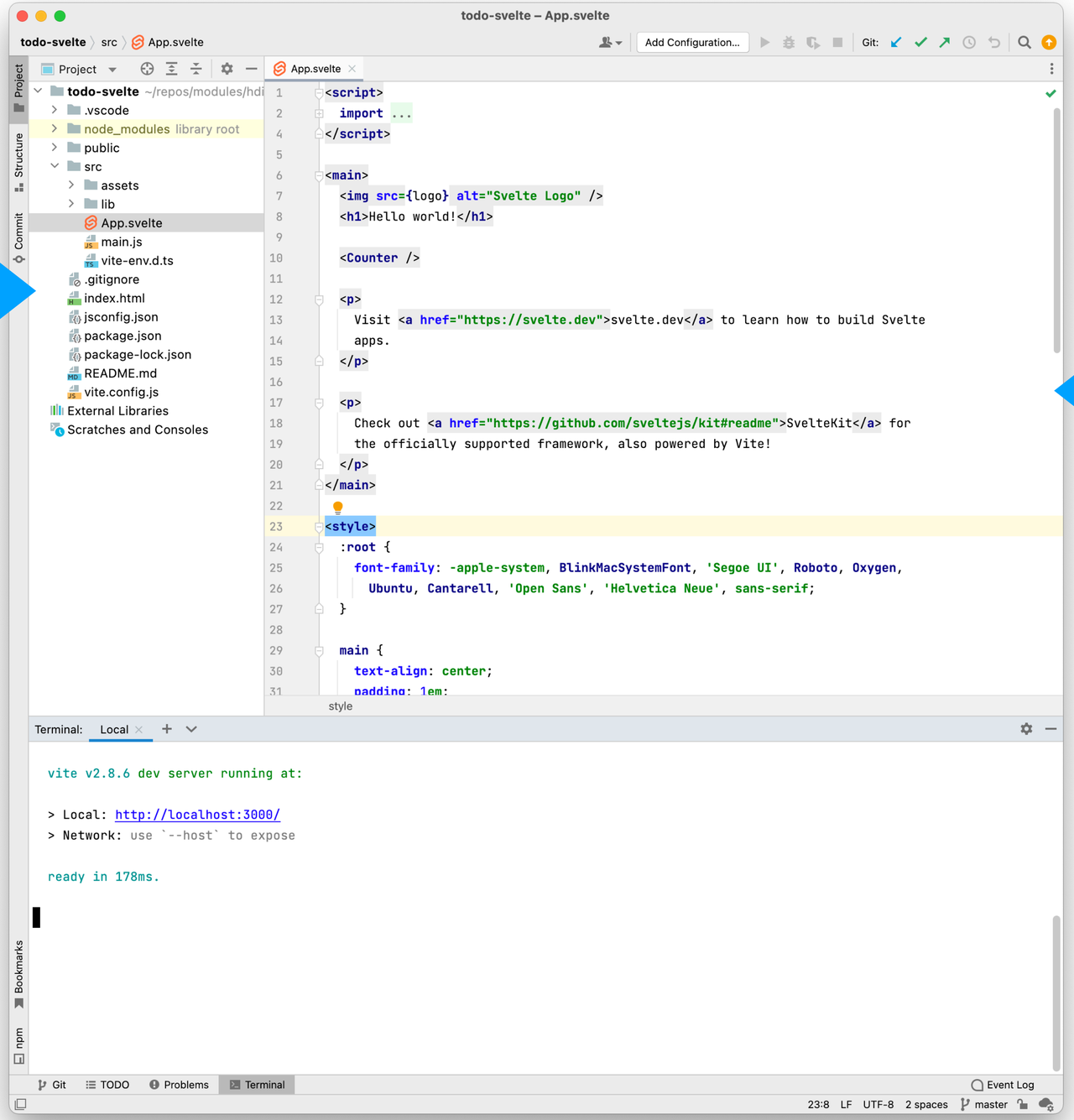
This should respond with:

```
  vite v2.8.6 dev server running at:

  > Local: http://localhost:3000/
  > Network: use `--host` to expose

  ready in 274ms.
```

Create Svelte App



Index.html



App.svelte

Starter Component

Svelte application launched (on front end)

index.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="/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title> Todo using Svelte </title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bulma@0.9.3/css/bulma.min.css">
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="/src/main.js"></script>
  </body>
</html>
```

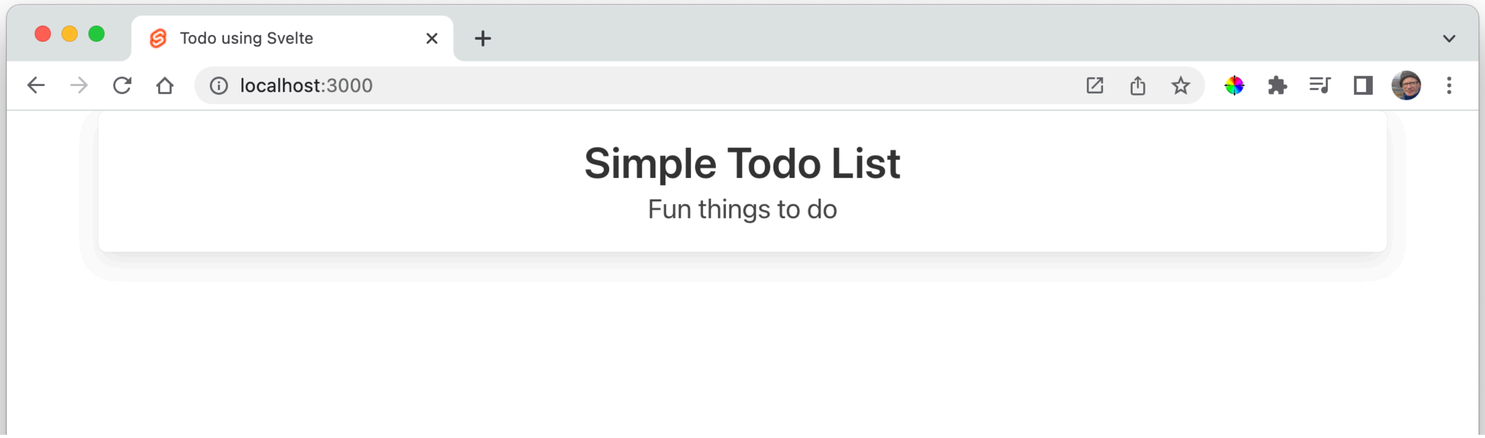
App.svelte

Component Code (empty)

Component UI

Running Application

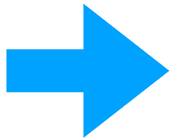
```
<script>
</script>

<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

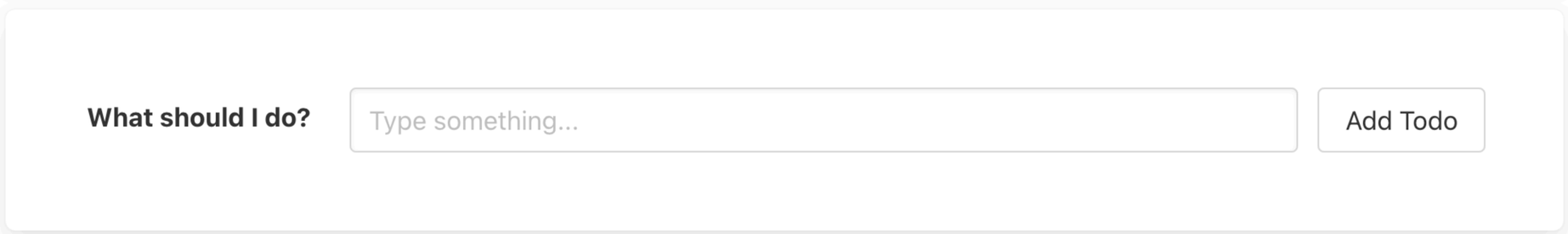
Additional Form elements

(in App.svelte)



```
<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

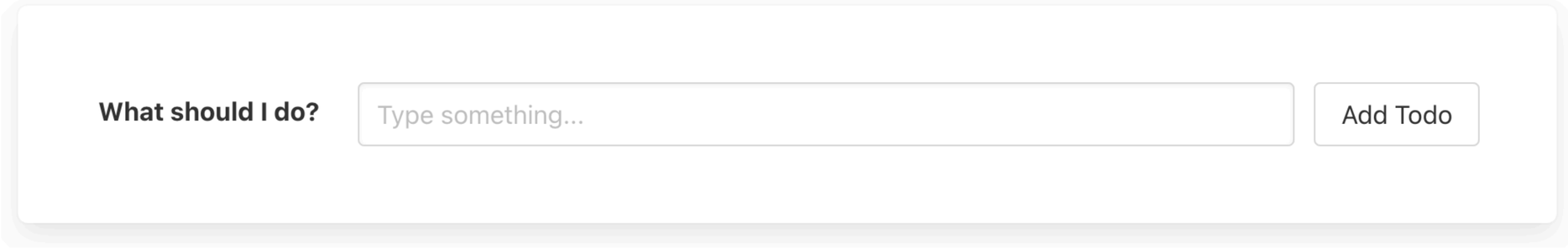
```
<script>
  let todoText;
  function addTodo() {
    console.log(todoText)
  }
</script>
```

“Bind” the variable to the input field

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

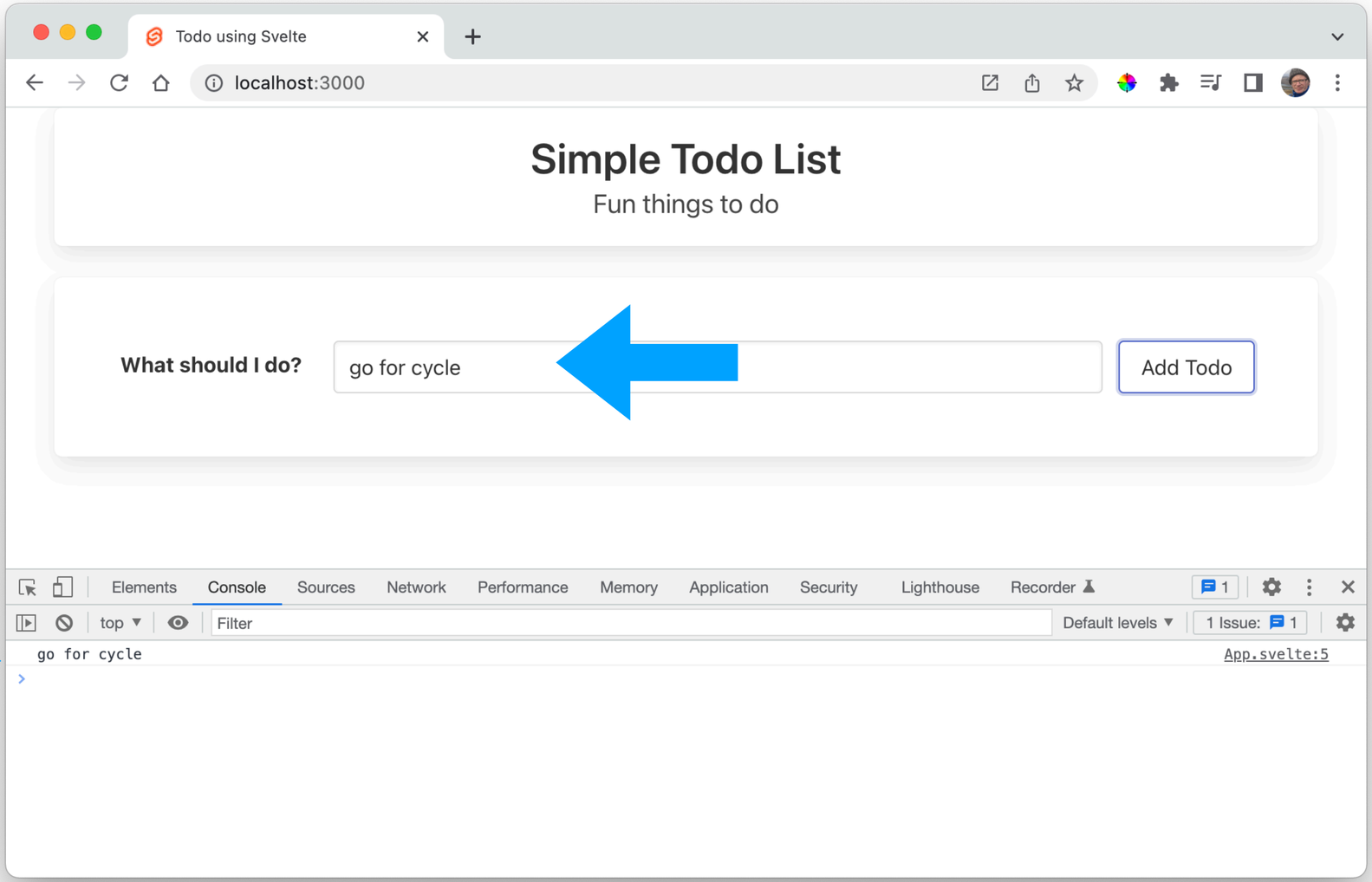
```
<button on:click={addTodo} class="button">Add Todo</button>
```

Trigger function when button pressed



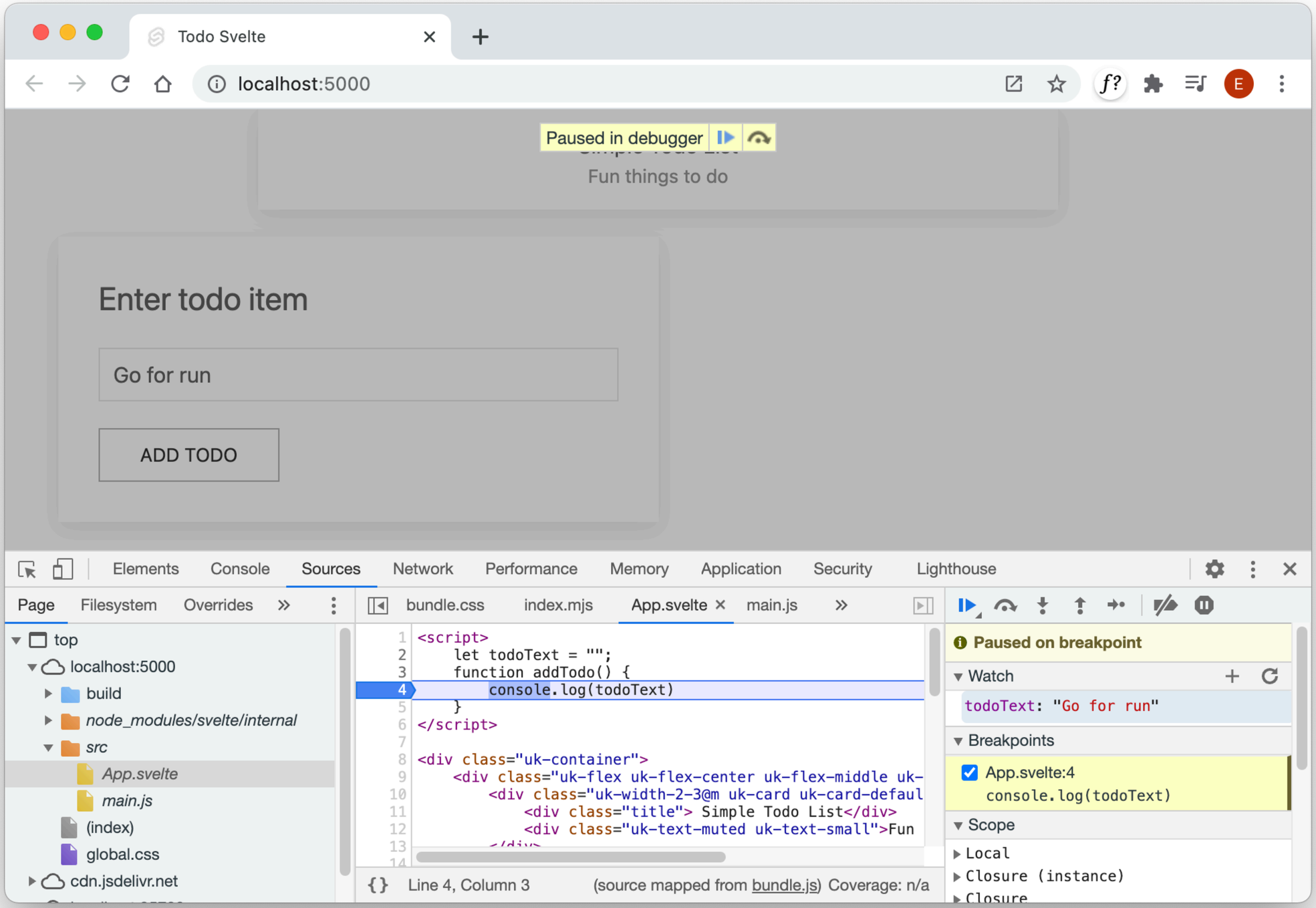
Binding Behaviour

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



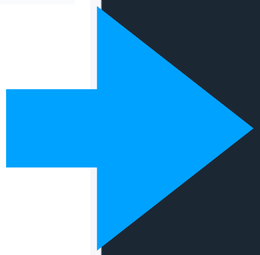
Debugging

Application Structure + Source Debugging available



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

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

Simple template style syntax to iterate array

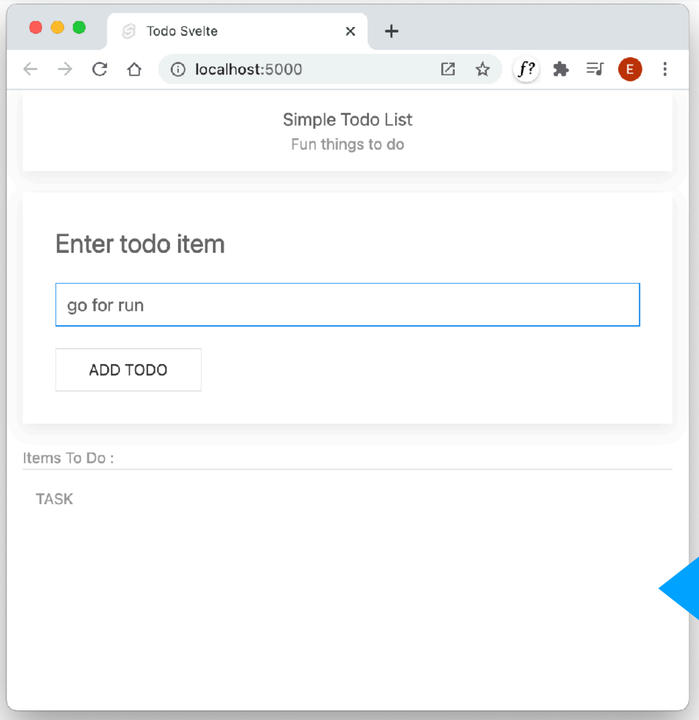


```
<table class="uk-table uk-table-divider">
  <caption> Items To Do :</caption>
  <thead>
    <tr>
      <th>Task</th>
    </tr>
  </thead>
  <tbody>
    {#each todoItems as todo}
      <tr>
        <td> {todo} </td>
      </tr>
    {/each}
  </tbody>
</table>
```

Svelte Template Syntax

Binding Behaviour - Arrays

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

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



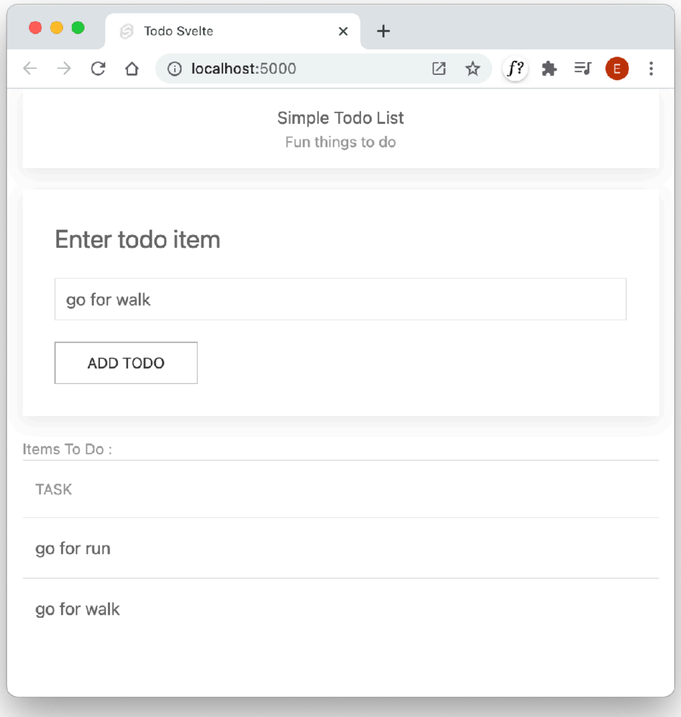

```
<table class="uk-table uk-table-divider">
  <caption> Items To Do :</caption>
  <thead>
    <tr>
      <th>Task</th>
    </tr>
  </thead>
  <tbody>
    {#each todoItems as todo}
      <tr>
        <td> {todo} </td>
      </tr>
    {/each}
  </tbody>
</table>
```

However, fails to update list!

Spread Assignment

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

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

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



"Spread" operator updates all elements (with a self assignment)

This has the effect to 'touching' the state of the array element.

Which in turns triggers update to the DOM

- Spread Operator

Spread syntax (...)

Spread syntax (`...`) allows an iterable such as an array expression or string to be expanded in places where zero or more arguments (for function calls) or elements (for array literals) are expected, or an object expression to be expanded in places where zero or more key-value pairs (for object literals) are expected.

JavaScript Demo: Expressions - Spread syntax

```
1  function sum(x, y, z) {
2    return x + y + z;
3  }
4
5  const numbers = [1, 2, 3];
6
7  console.log(sum(...numbers));
8  // expected output: 6
9
10 console.log(sum.apply(null, numbers));
11 // expected output: 6
12
```

Run ›

Reset

Description

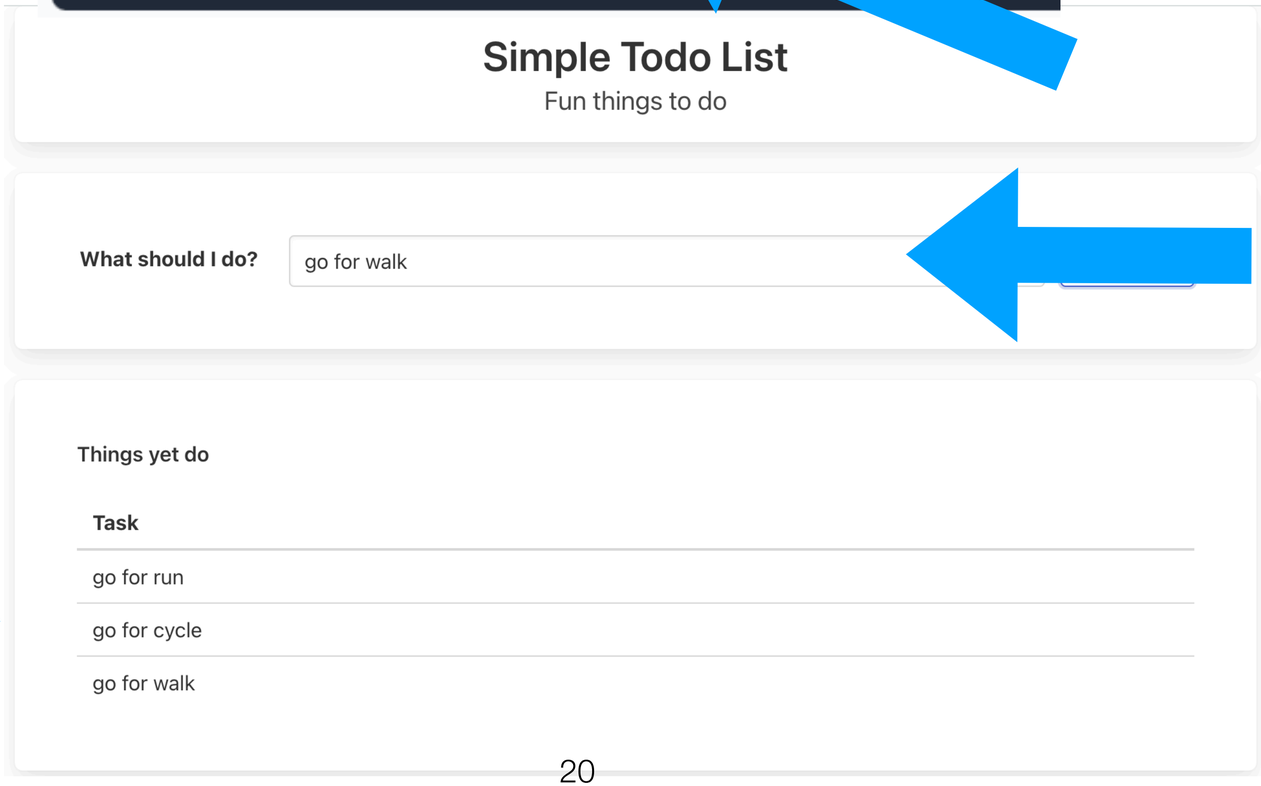
Spread syntax can be used when all elements from an object or array need to be included in a list of some kind.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax

Binding Behaviours

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

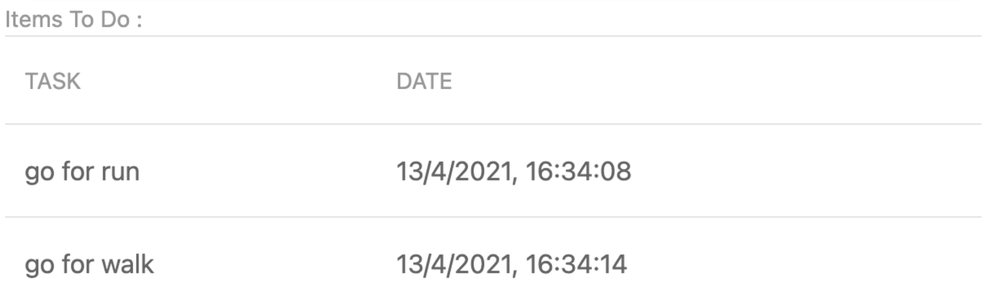
Spread assignment to trigger array update in DOM



Simple assignment to trigger input field update

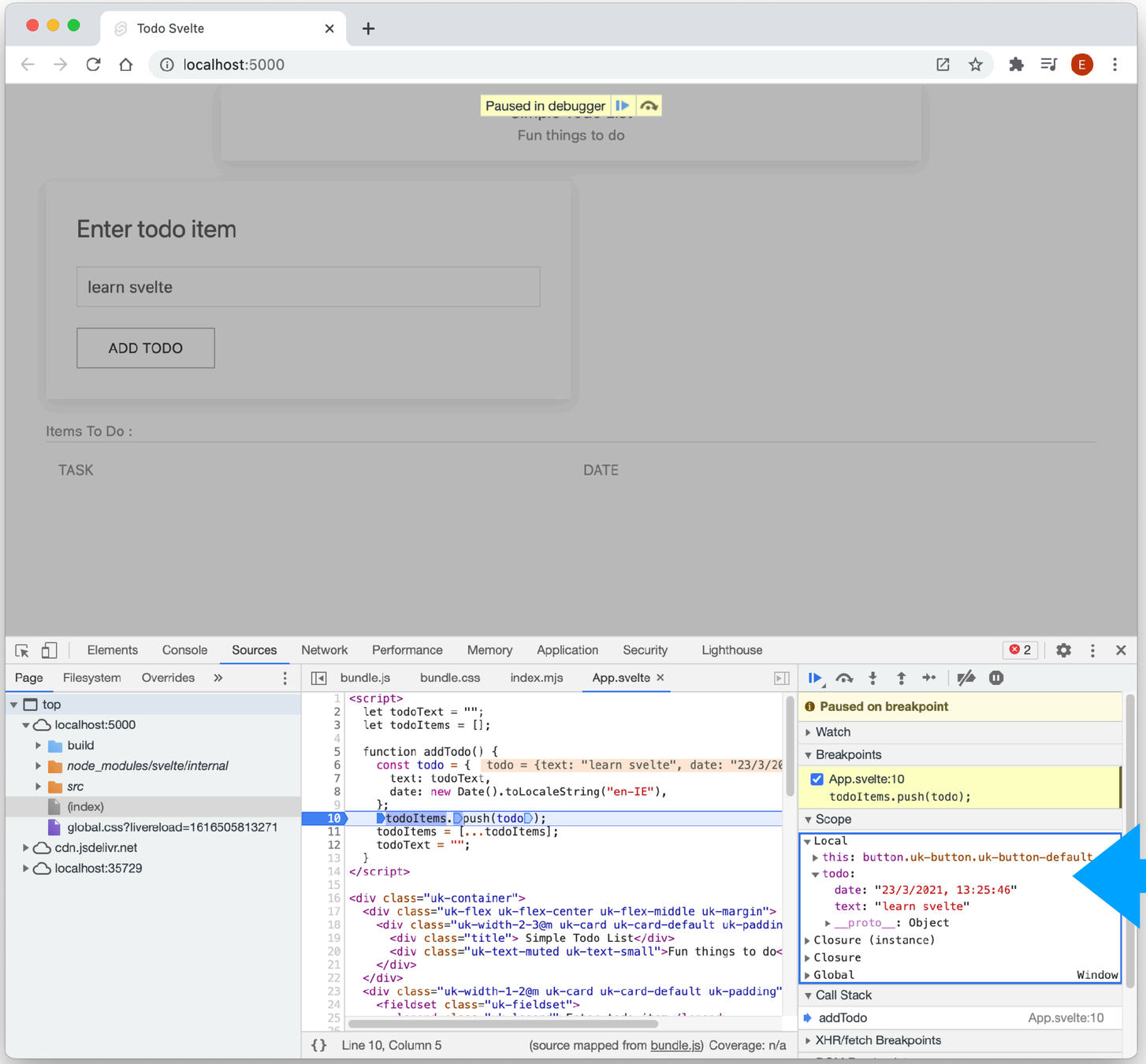
Todo Object

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




```
<table class="uk-table uk-table-divider">
  <caption> Items To Do :</caption>
  <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>
```

Debugging

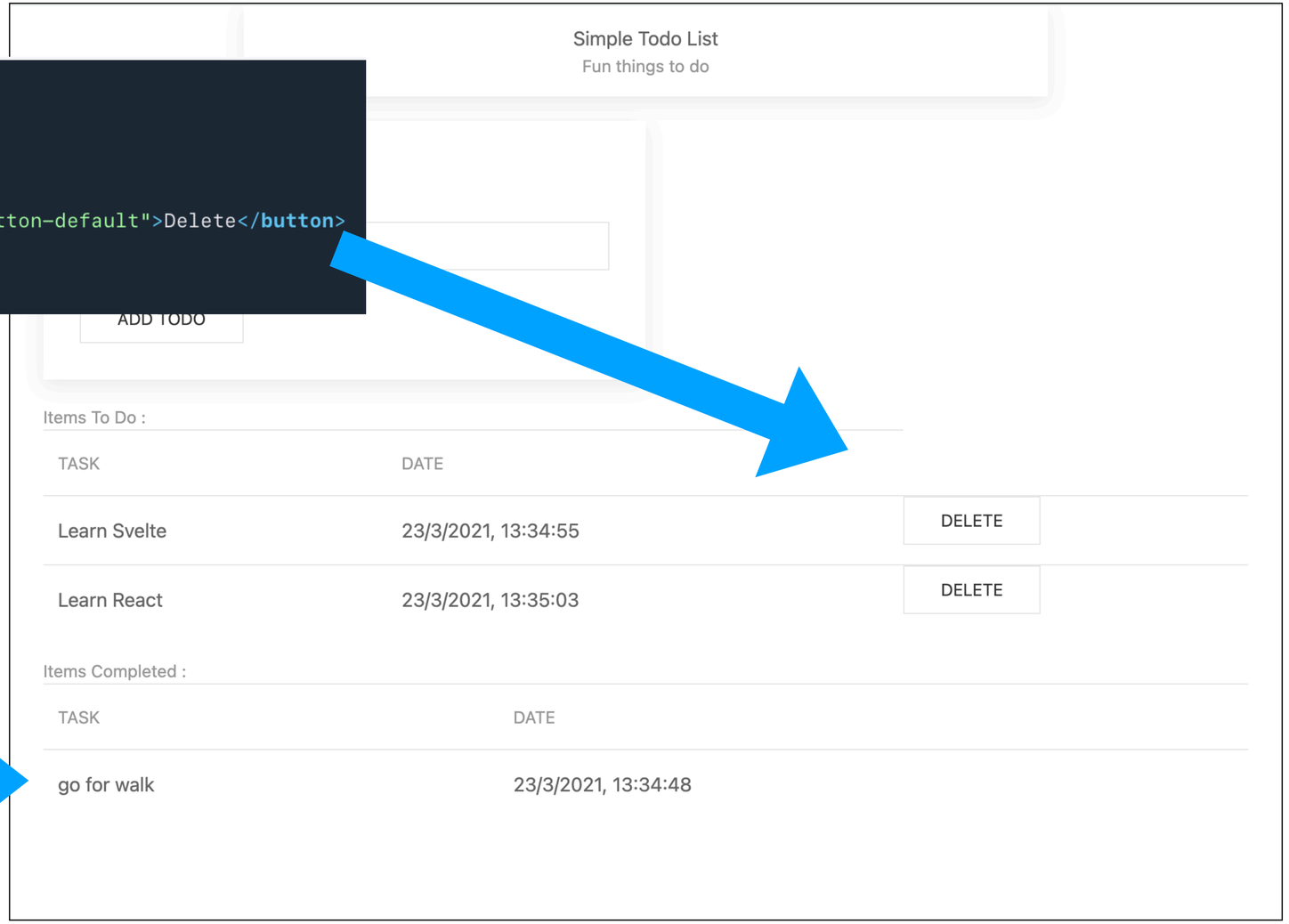


Inspect Data Structure

Done List - UX

```
{#each todoItems as todo}
  <tr>
    <td> {todo.text} </td>
    <td> {todo.date} </td>
    <button on:click={deleteTodo(todo.id)}  class="uk-button uk-button-default">Delete</button>
  </tr>
{/each}
```

```
<table class="uk-table uk-table-divider" id="done-table">
  <caption> Items Completed :</caption>
  <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>
```



Done List - Behaviour

Declare doneItems

Generate Unique ID

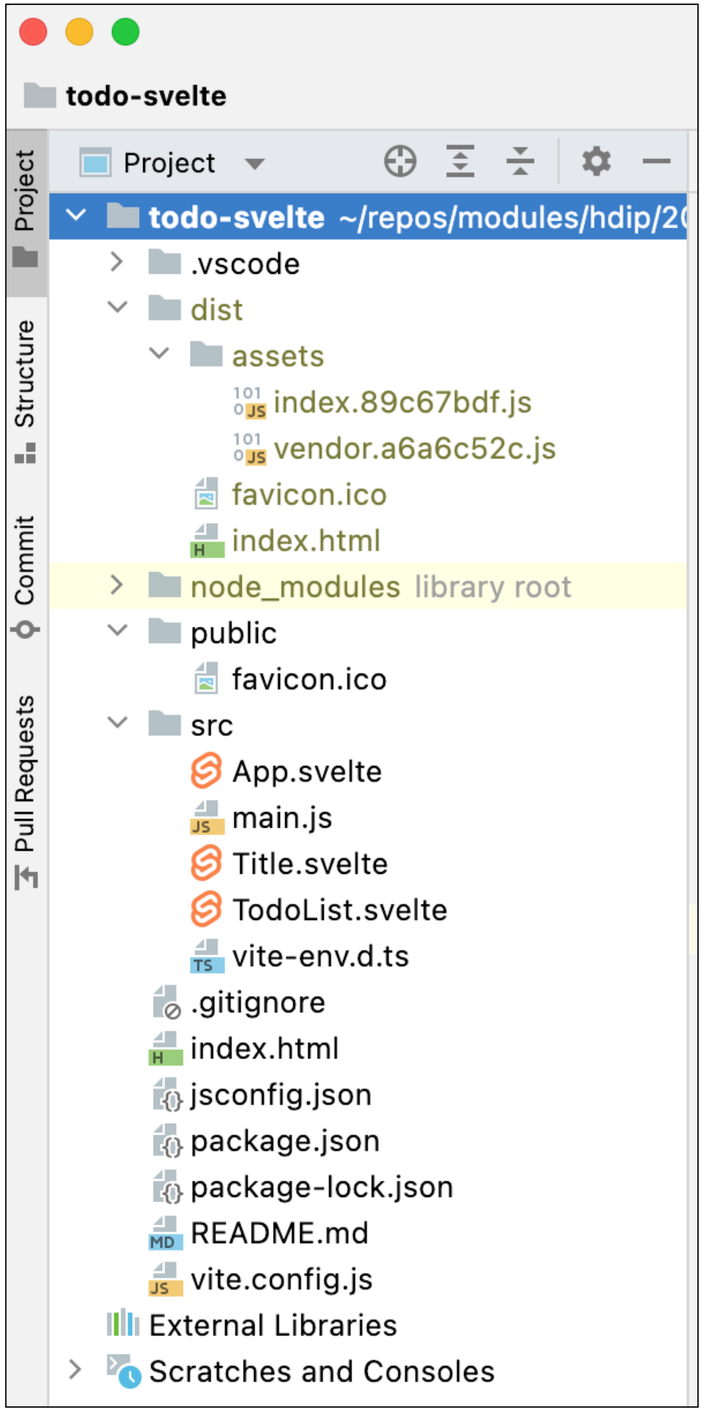
Include ID in Todo Object

Add deleted Todos to doneItems

Update DOM

```
<script>
  let todoText = "";
  let todoItems = [];
  let doneItems = [];

  function uuidv4() {
    return "xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx".replace(/[xy]/g, function (c) {
      var r = Math.random() * 16 | 0, v = c == "x" ? r : (r & 0x3 | 0x8);
      return v.toString(16);
    });
  }

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

  function deleteTodo(id) {
    const found = todoItems.findIndex((todo) => todo.id == id);
    const done = todoItems[found];
    todoItems.splice(found, 1);
    todoItems = [...todoItems];
    doneItems.push(done);
    doneItems = [...doneItems];
  }
</script>
```

Project Structure



Project Structure

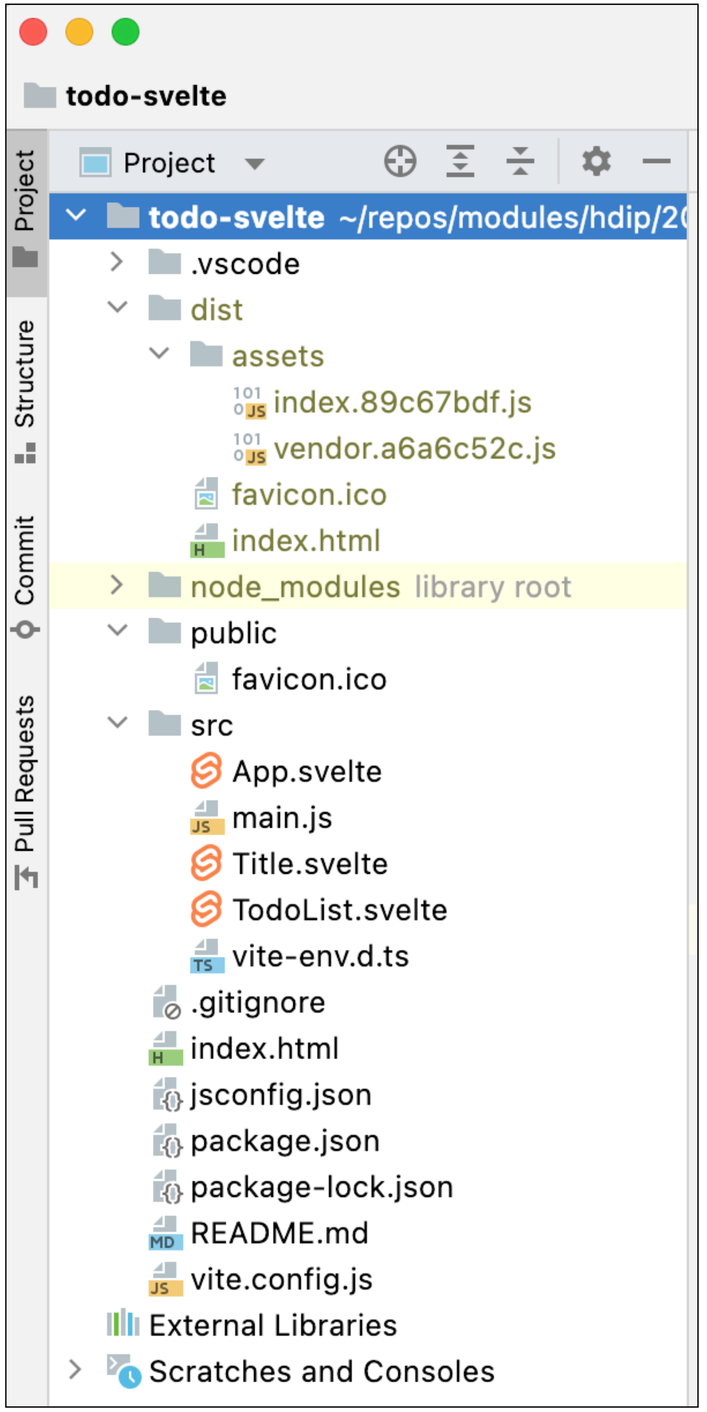
Generated (bundled) application

Application Modules

Static resources

Application (Source)

Start Page (source)



Project Structure

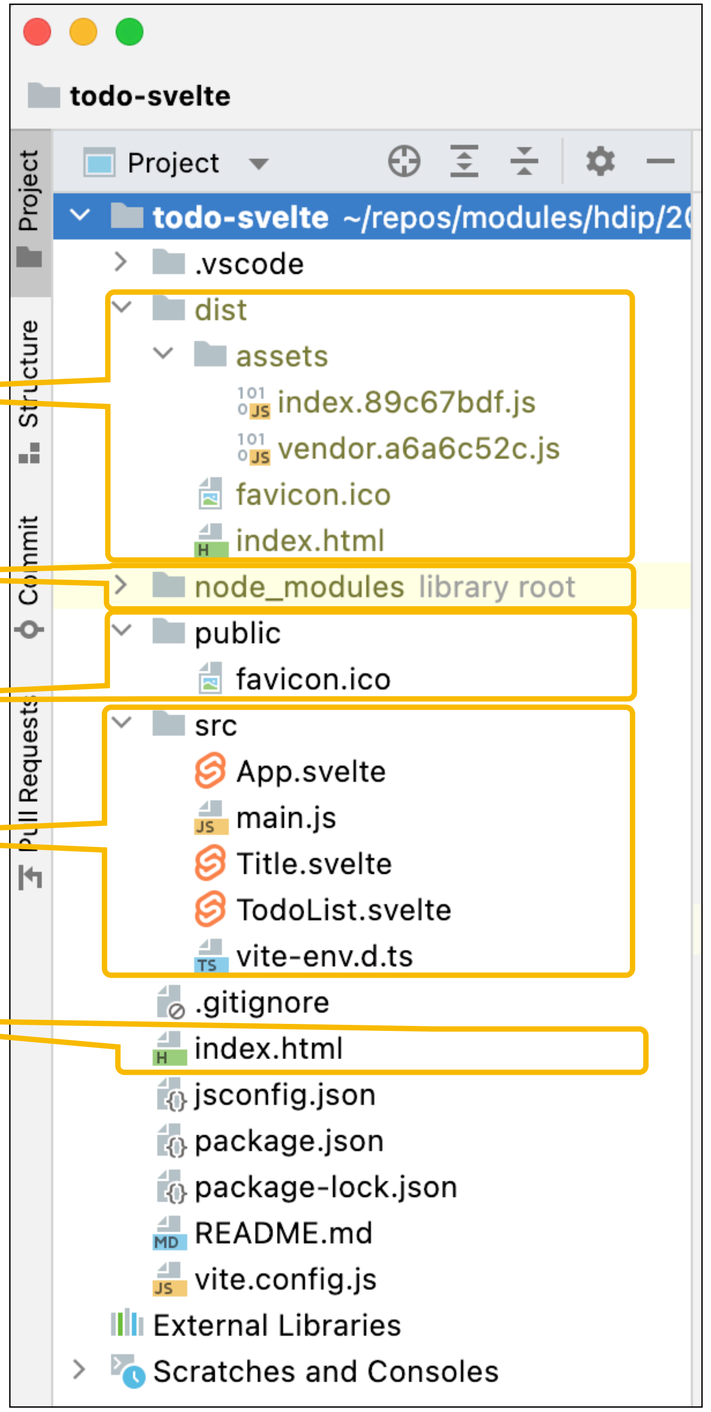
Generated (bundled) application

Application Modules

Static resources

Application (Source)

Start Page (source)



Svelte First Steps

Building your first Svelte Application