

Svelte First Steps



Building your first Svelte
Application

Todo Example - Recap

The diagram illustrates the progression of the Todo example:

- The Browser Environment:** Shows a laptop icon with a code editor and a lamp, labeled "The Browser Environment".
- Todo DOM:** Shows a grid icon with a path through it, labeled "Todo DOM". Below it is the text "Manipulating the DOM to implement TODO features".
- Lab-12a-Todo 1:** A form with an input field containing "go for walk" and a button "ADD TODO". Below it is the text "Build a simple todo app in javascript."
- Lab-12b-Todo 2:** A table titled "Items To Do:" with two rows:

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

Below the table is the text "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
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

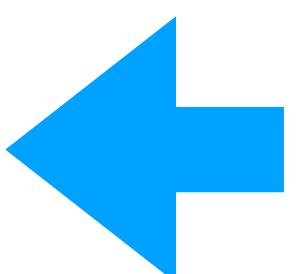
What should I do? go for a cycle Add Todo

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



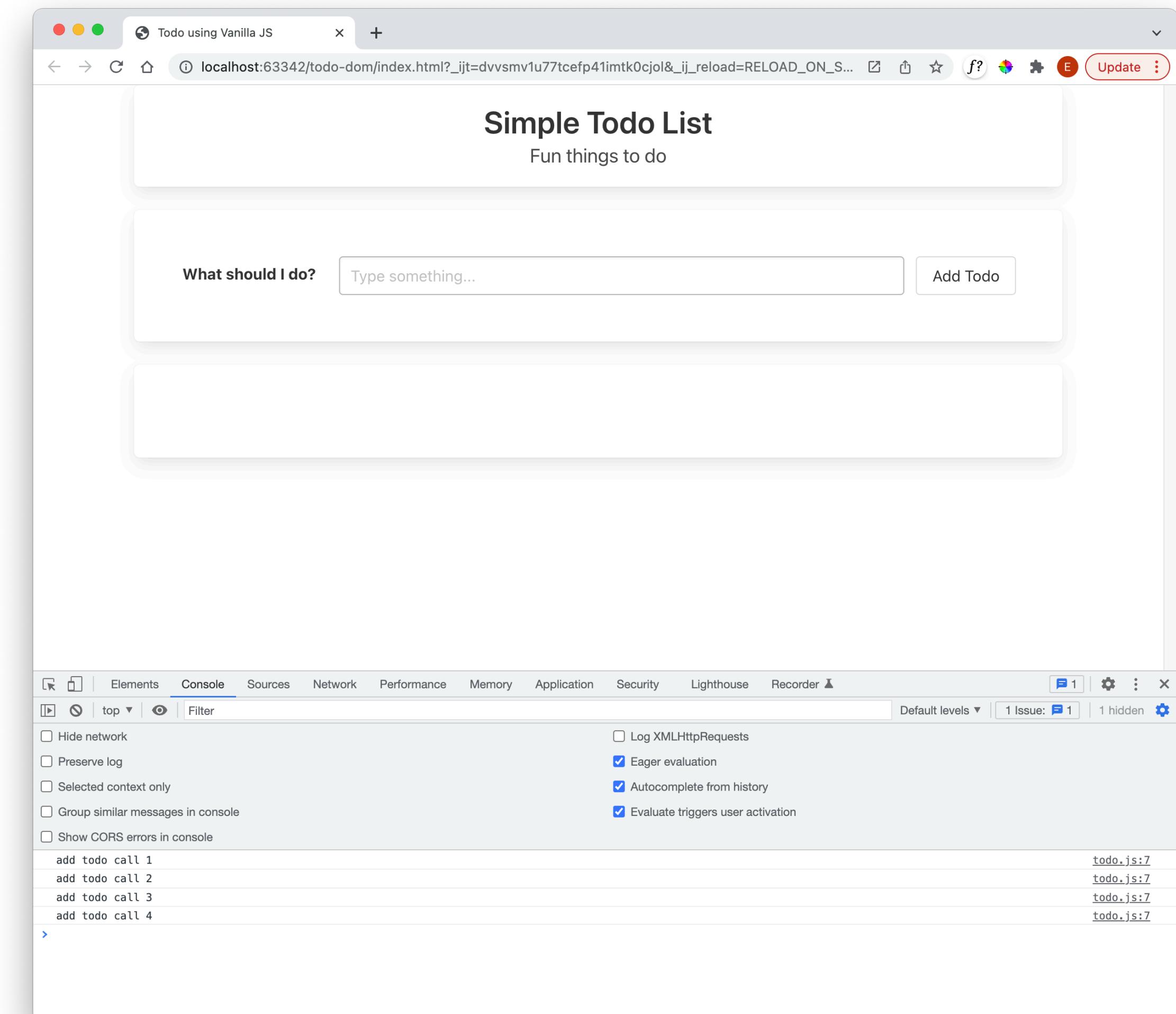
Incorporate Javascript into page

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

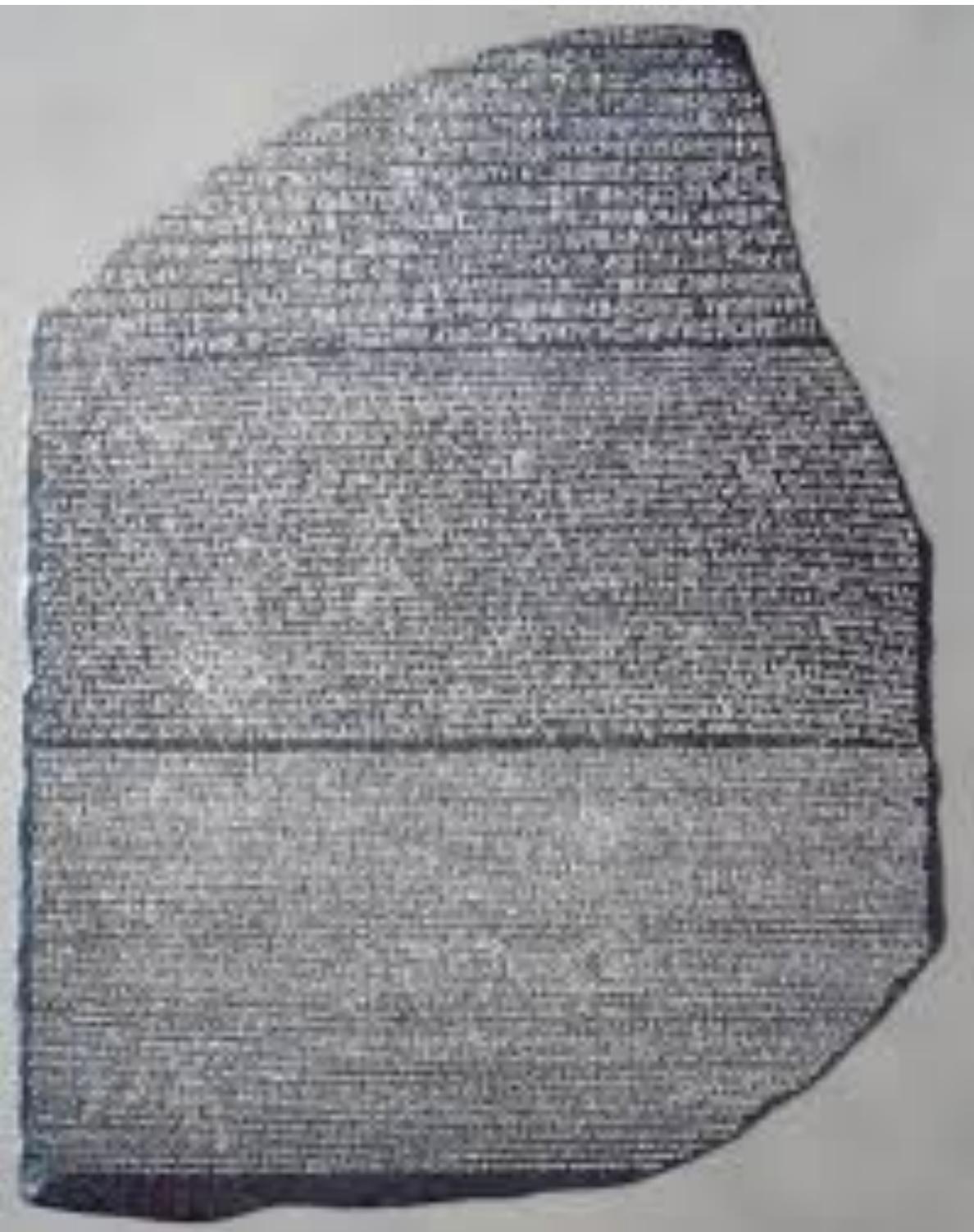
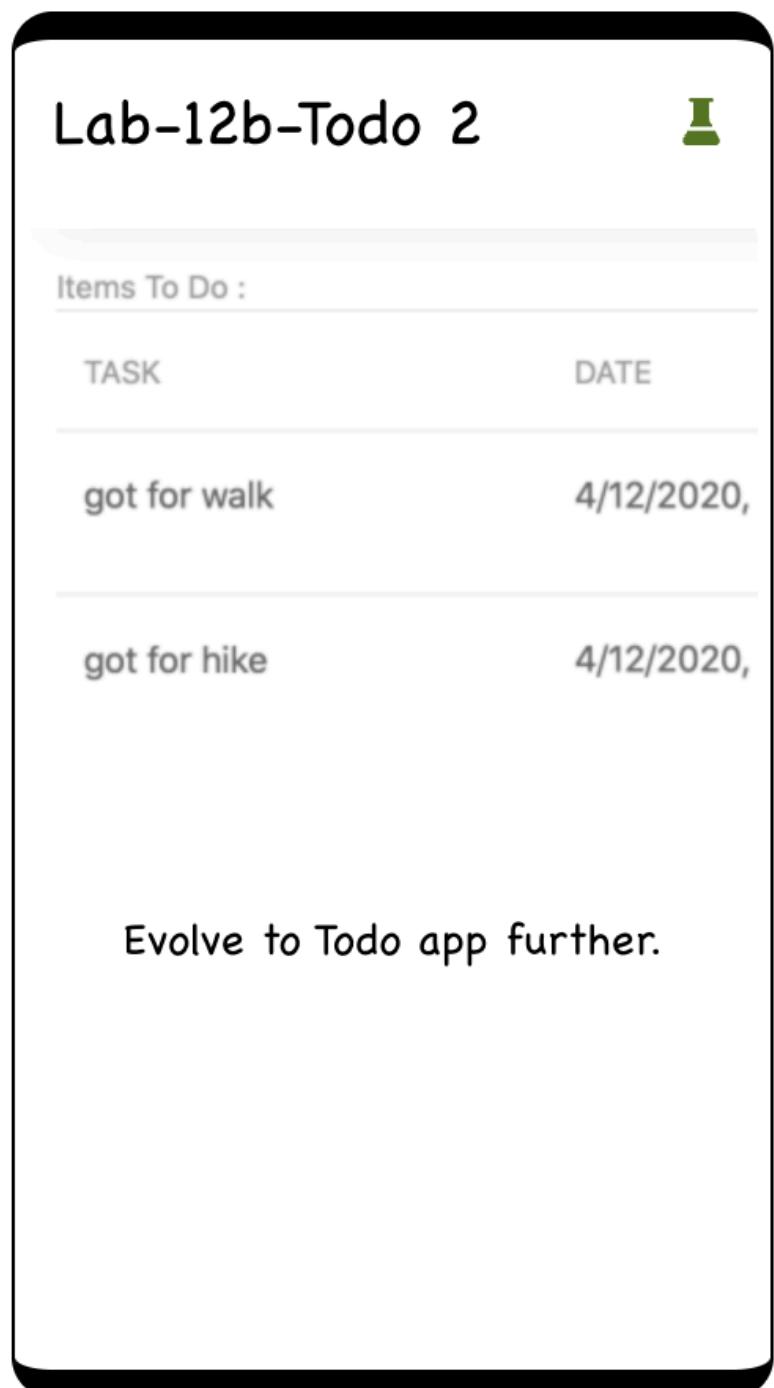
```
function addTodo() {
  count++;
  const todoText = document.getElementById("todo-id").value;
  console.log(`add todo call ${count}: ${todoText}`)

  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



```
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
```

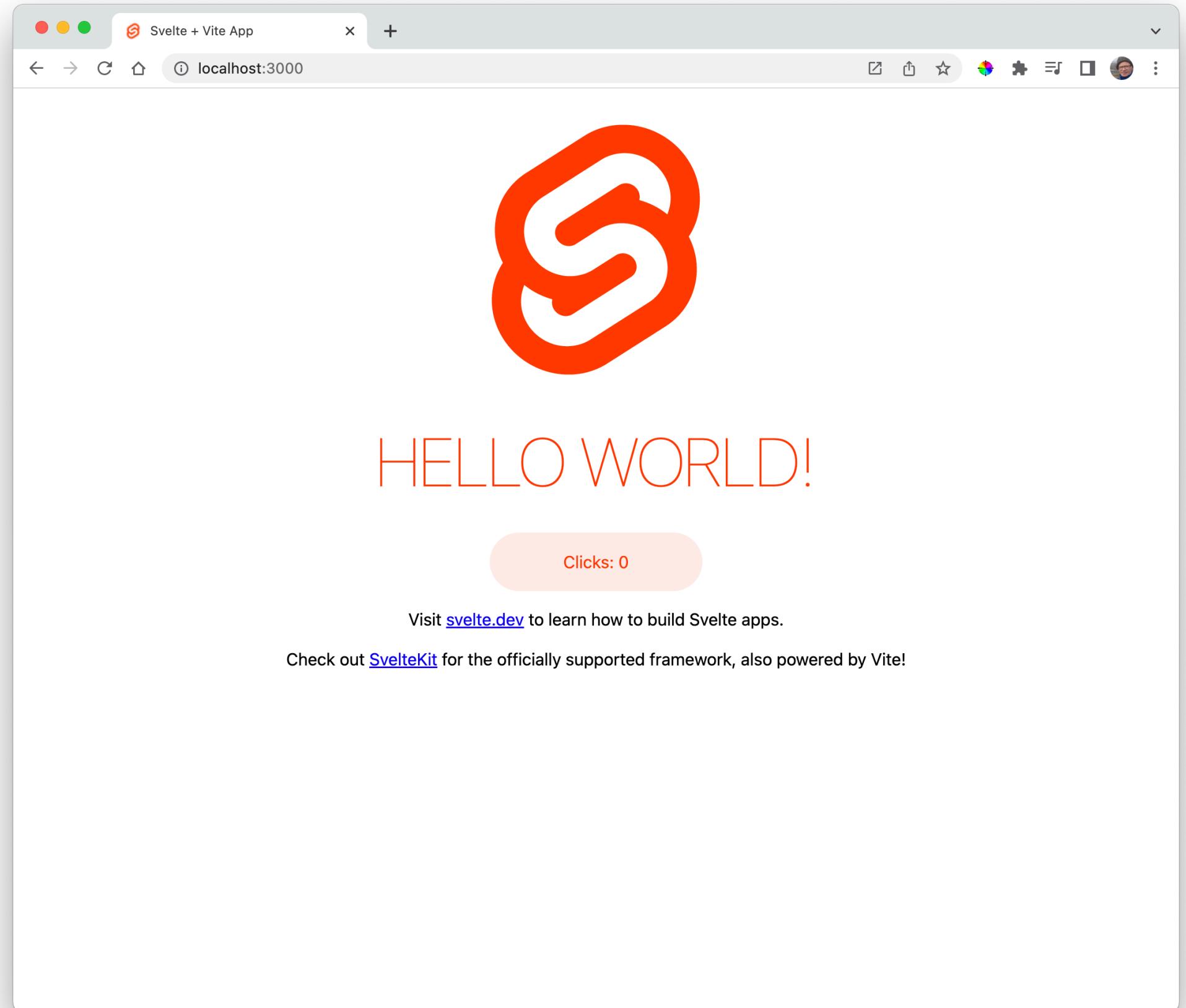
Now we can run the application:

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

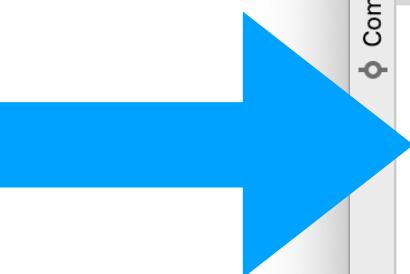
This should respond with:

```
vite v2.8.6 dev server running at:  
  
> Local: http://localhost:3000/  
> Network: use `--host` to expose  
  
ready in 274ms.
```

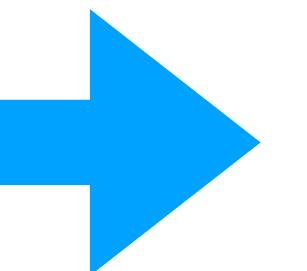
Create Svelte App



Index.html



Svelte application
launched (on front
end)



The screenshot shows the VS Code interface with the following details:

- Project Explorer:** Shows the project structure: todo-svelte (~/repos/modules/hdi), .vscode, node_modules library root, public, src (assets, lib), App.svelte, main.js, vite-env.d.ts, .gitignore, index.html, jsconfig.json, package.json, package-lock.json, README.md, vite.config.js, External Libraries, Scratches and Consoles.
- Code Editor:** The file App.svelte is open, displaying Svelte code. The code includes a script block, a main component with an img tag, an h1 heading, a Counter component, and two p tags providing links to Svelte and SvelteKit documentation. It also contains a style block with CSS for the root and main elements.
- Terminal:** The terminal shows the output of the Vite dev server command: "vite v2.8.6 dev server running at: Local: http://localhost:3000/ Network: use `--host` to expose ready in 178ms."
- Bottom Status Bar:** Shows the current branch as master, the file path as todo-svelte / src / App.svelte, and the file status as green.

App.svelte

Starter
Component

index.html

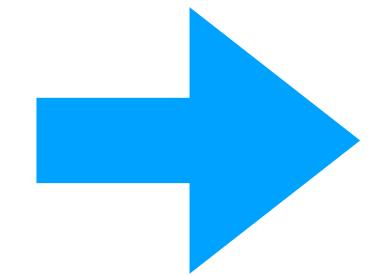
Bulma CSS
framework

Entire Front
End Application
'bundled' into
this Javascript
file

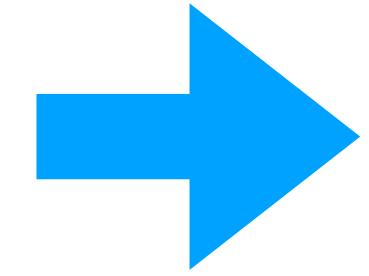
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" href="/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title> Todo using Svelte </title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bulma@0.9.3/css/bulma.min.css">
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="/src/main.js"></script>
  </body>
</html>
```

App.svelte

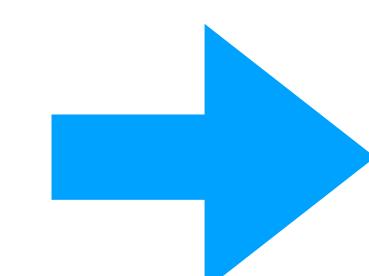
Component
Code (empty)



Component UI

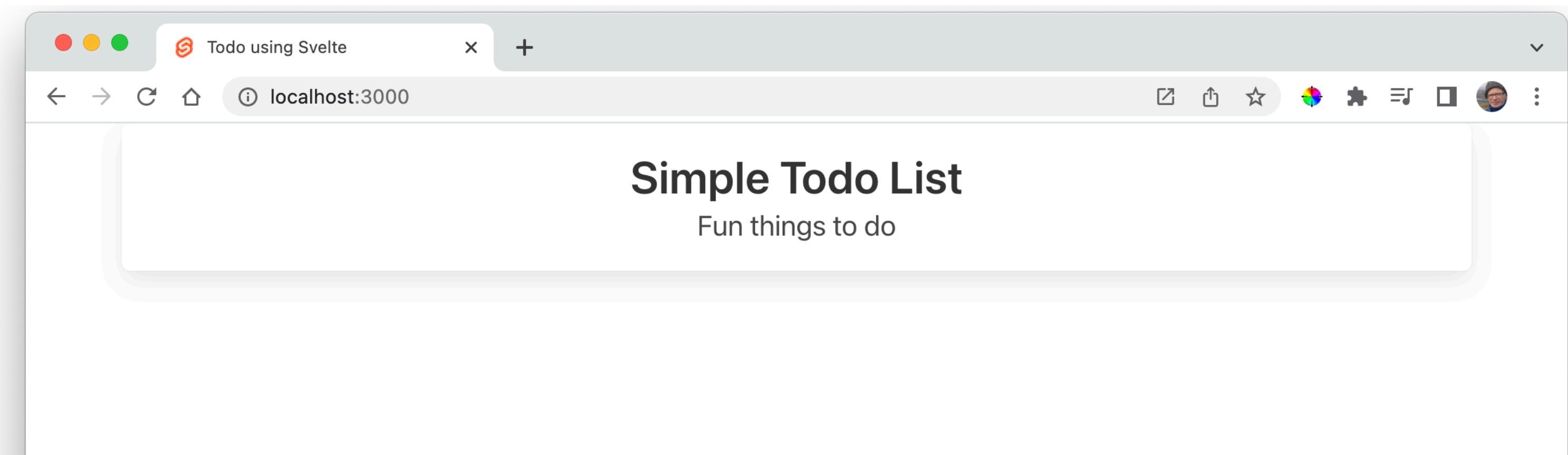


Running
Application



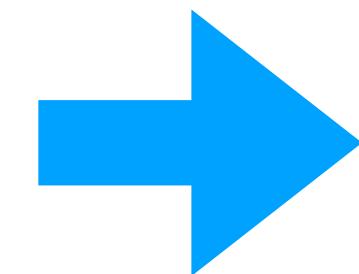
```
<script>
</script>

<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>
```



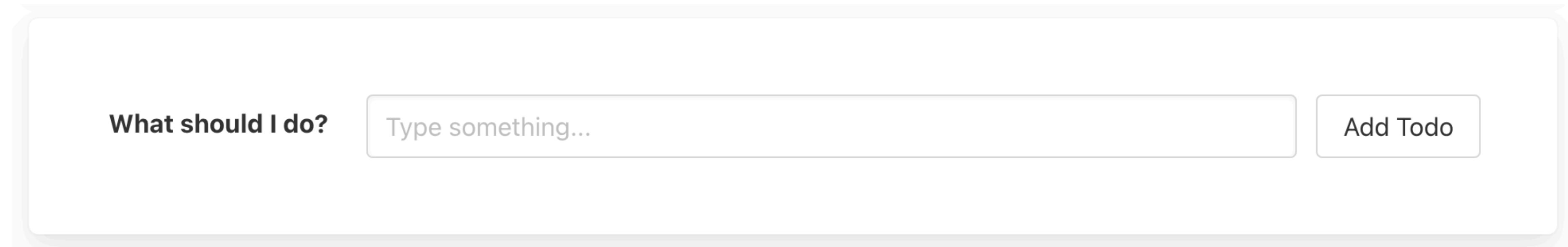
Incorporating Form UX

Additional
Form
elements
(in App.svelte)



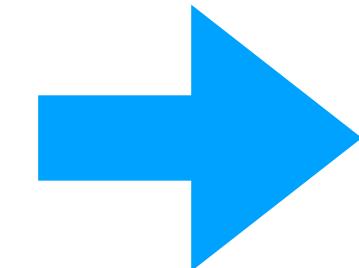
```
<div class="section box">
  <div class="field is-horizontal">
    <div class="field-label is-normal">
      <label for="todo" class="label">What should I do?</label>
    </div>
    <div class="field-body">
      <div class="field">
        <p class="control">
          <input id="todo" class="input"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

Form
unresponsive
(yet)



Component Logic

Simple
Variable +
function

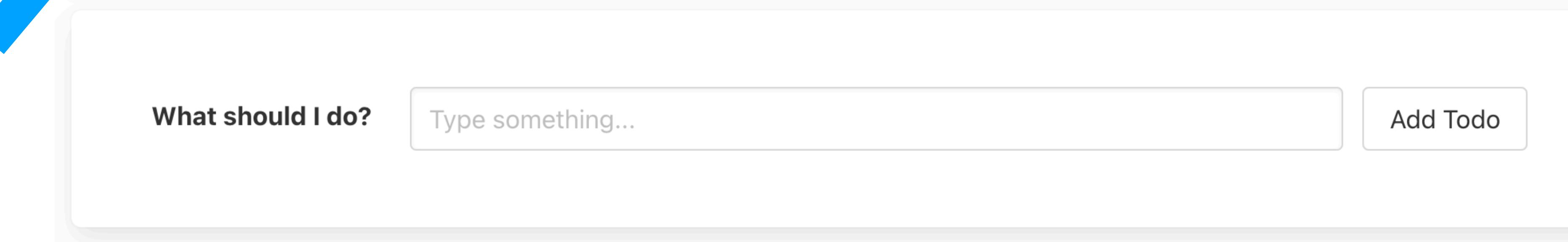


```
<script>  
  let todoText;  
  function addTodo() {  
    console.log(todoText)  
  }  
</script>
```

“Bind” the
variable to the
input field

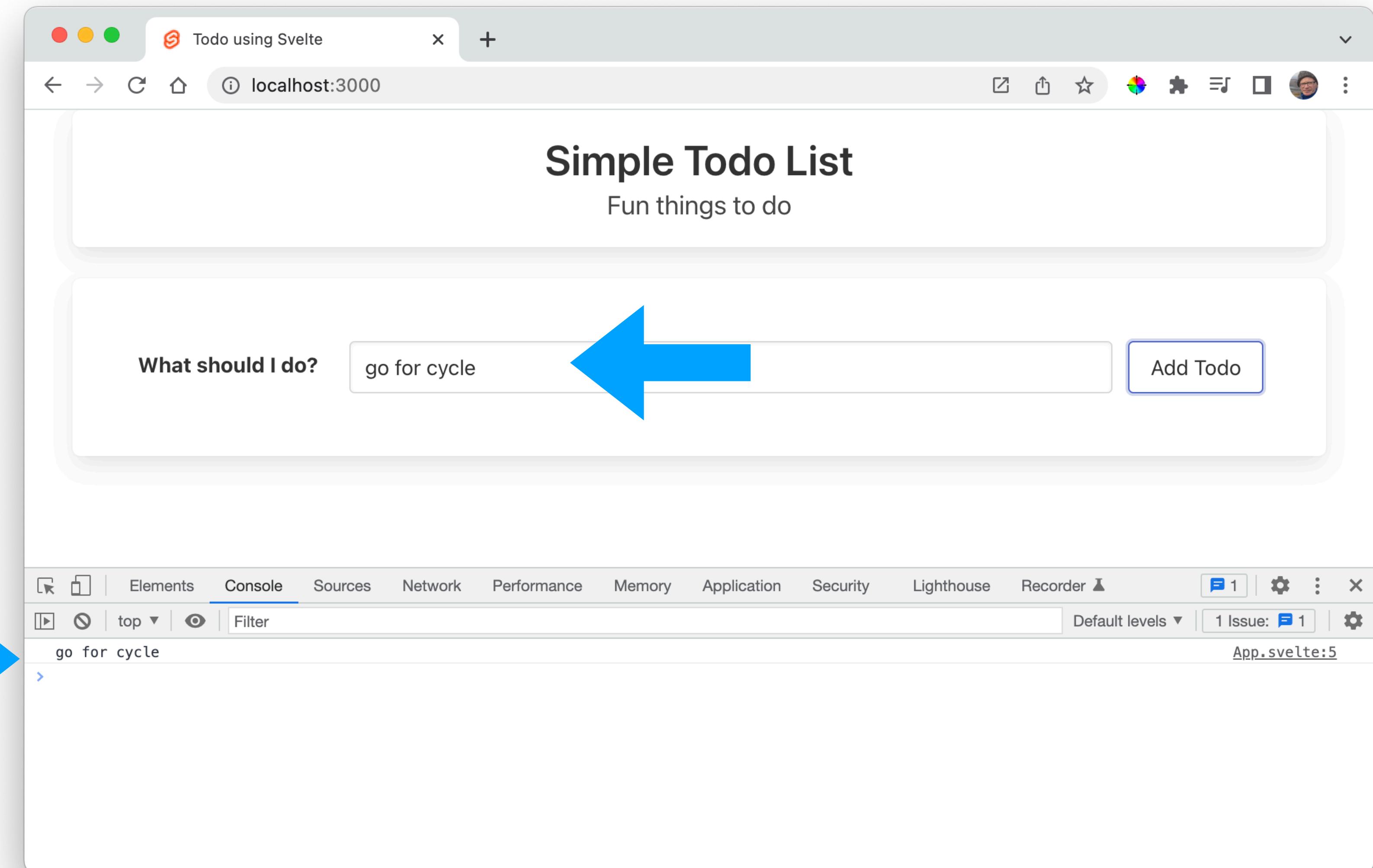
```
<input bind:value={todoText} id="todo" class="input" type="text" placeholder="Type something...">  
  
<button on:click={addTodo} class="button">Add Todo</button>
```

Trigger function
when button
pressed



Binding Behaviour

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



Debugging

Application
Structure +
Source
Debugging
available

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 form to "Enter todo item" containing a text input "Go for run" and a button "ADD TODO". A status bar at the top right says "Paused in debugger". Below the page content is the browser's developer tools, specifically the "Sources" tab. The left sidebar shows the file structure: "top", "localhost:5000" (with "build", "node_modules/svelte/internal", and "src" folders), and files like "bundle.css", "index.mjs", "App.svelte", "main.js", "(index)", and "global.css". The "App.svelte" file is open in the main pane, showing the following code:

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

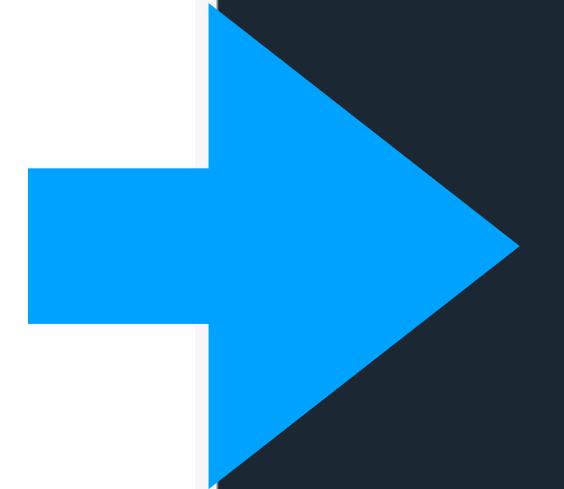
<div class="uk-container">
  <div class="uk-flex uk-flex-center uk-flex-middle uk-align-center">
    <div class="uk-width-2-3@m uk-card uk-card-default">
      <div class="title"> Simple Todo List </div>
      <div class="text-muted uk-text-small"> Fun things to do </div>
    </div>
  </div>
</div>
```

The line "console.log(todoText)" is highlighted with a blue arrow and selected. The right sidebar contains the debugger controls and a list of breakpoints, watches, and scopes. It shows a breakpoint is active at line 4 of "App.svelte", and a watch for "todoText" is set to "Go for run".

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

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

Simple template style
syntax to iterate array

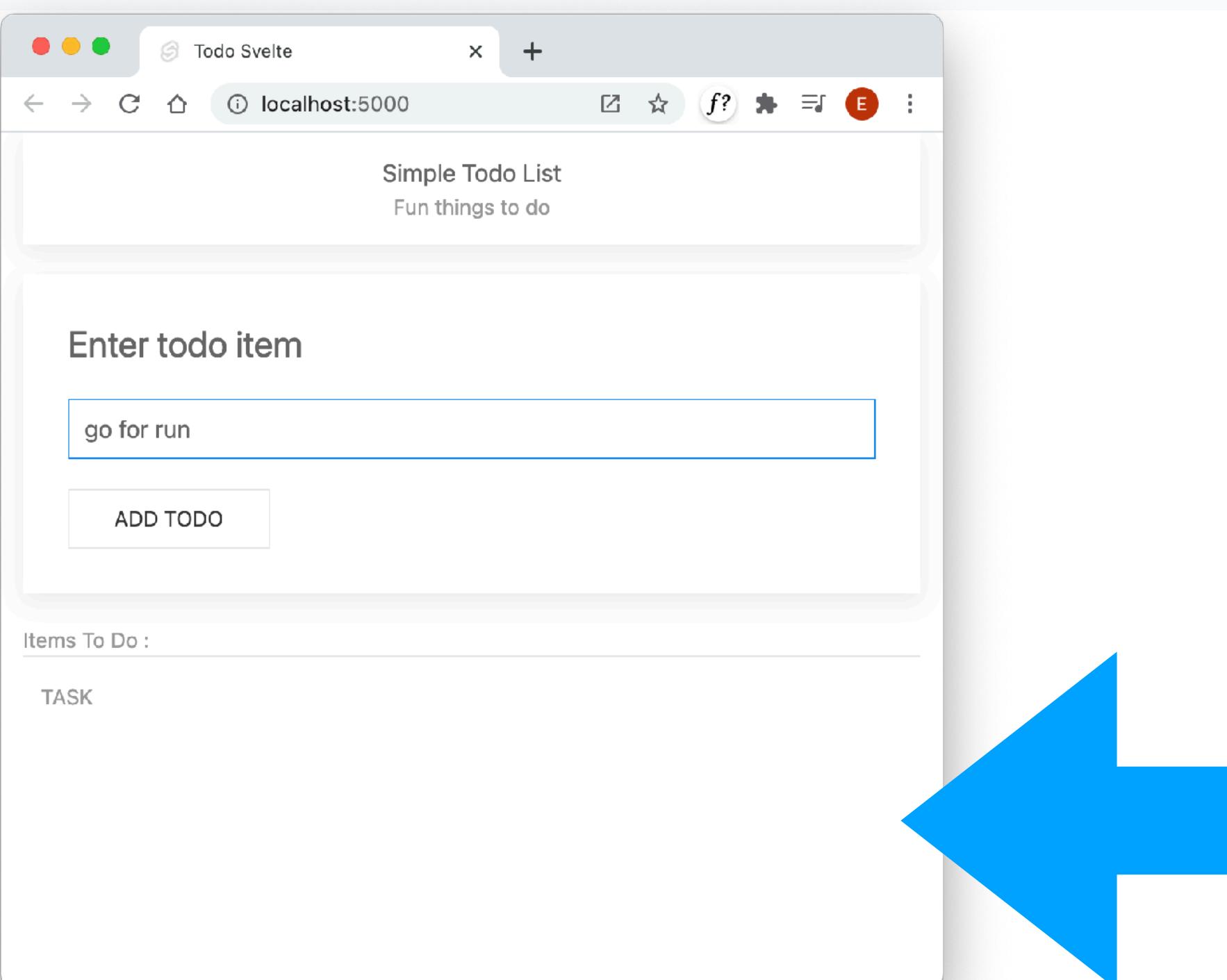


```
<table class="uk-table uk-table-divider">
  <caption> Items To Do :</caption>
  <thead>
    <tr>
      <th>Task</th>
    </tr>
  </thead>
  <tbody>
    {#each todoItems as todo}
    <tr>
      <td> {todo} </td>
    </tr>
    {/each}
  </tbody>
</table>
```

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>
```

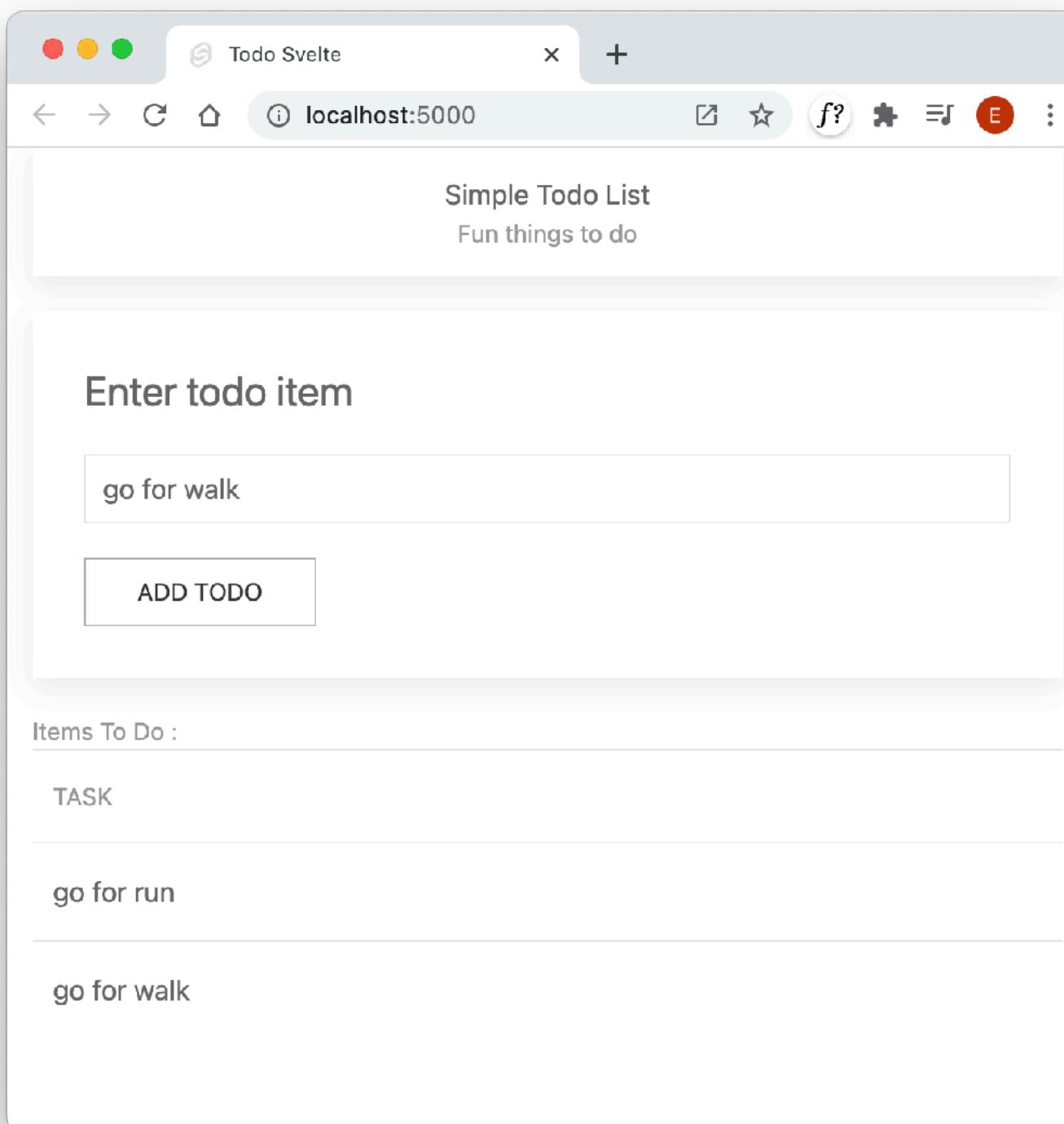
However, fails to update
list!

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

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

Spread Assignment

```
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 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.

- Spread Operator

 **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.

Binding Behaviours

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

Simple Todo List

Fun things to do

Spread
assignment
to trigger
array update
in DOM

Simple
assignment to
trigger input field
update

Things yet do

Task

go for run

go for cycle

go for walk

Todo Object

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

Items To Do :

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

```
<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>
```

The screenshot shows a browser window titled "Todo Svelte" at "localhost:5000". The page displays a simple todo application with a form to enter a todo item ("learn svelte") and a button to add it. Below the form is a section titled "Items To Do:" with two columns: "TASK" and "DATE". A blue arrow points from the "Scope" panel of the developer tools' Sources tab towards the "Inspect Data Structure" text.

Paused in debugger

Fun things to do

Enter todo item

learn svelte

ADD TODO

Items To Do :

TASK DATE

Elements Console Sources Network Performance Memory Application Security Lighthouse

bundle.js bundle.css index.mjs App.svelte

Paused on breakpoint

Breakpoints

App.svelte:10
todoItems.push(todo);

Scope

Local

this: button.uk-button.uk-button-default

todo:

date: "23/3/2021, 13:25:46"
text: "learn svelte"

__proto__: Object

Closure (instance)

Closure

Global

Call Stack

addTodo App.svelte:10

XHR/fetch Breakpoints

{ } Line 10, Column 5 (source mapped from bundle.js) Coverage: n/a

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>
```

Simple Todo List
Fun things to do

ADD TODO

Items To Do :

TASK	DATE	DELETE
Learn Svelte	23/3/2021, 13:34:55	DELETE
Learn React	23/3/2021, 13:35:03	DELETE

Items Completed :

TASK	DATE
go for walk	23/3/2021, 13:34:48

Done List - Behaviour

Add deleted Todos to
doneItems

Update DOM

Include ID in Todo Object

Generate Unique ID

Declare doneItems

```
<script>

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

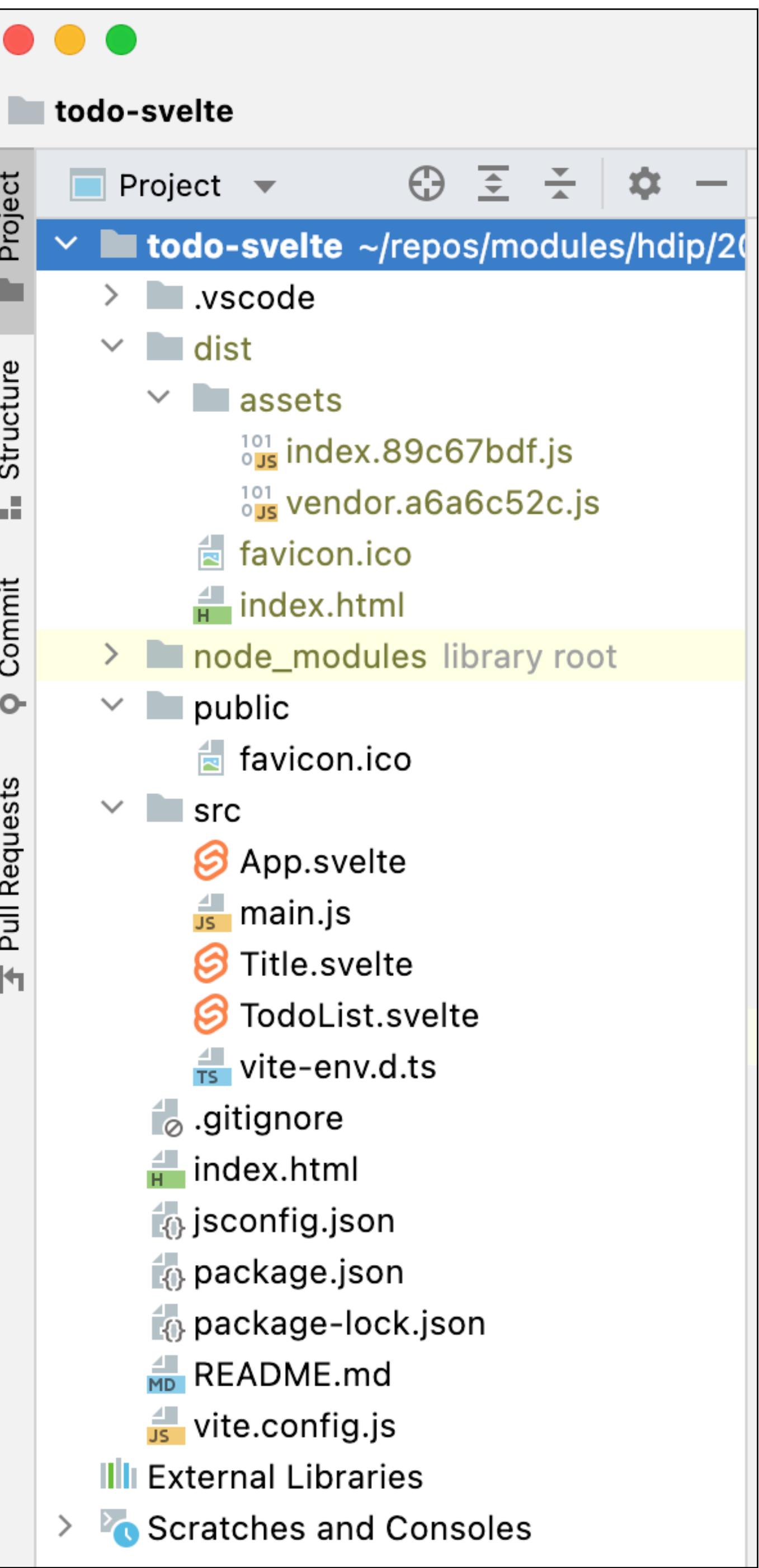
function uuidv4() {
  return "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx".replace(/[xy]/g, function (c) {
    var r = Math.random() * 16 | 0, v = c == "x" ? r : (r & 0x3 | 0x8);
    return v.toString(16);
  });
}

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

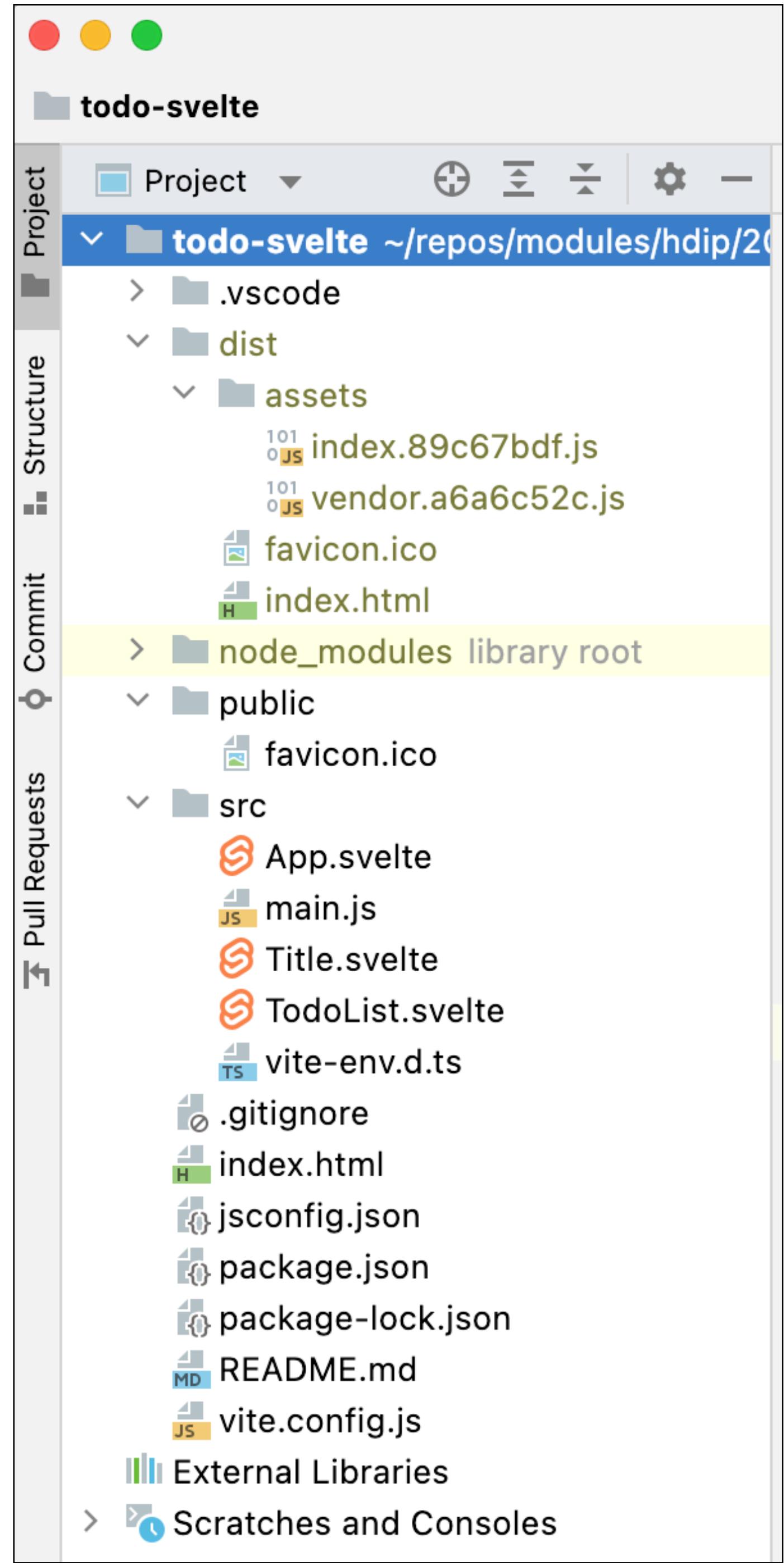
Generated (bundled) application

Application Modules

Static resources

Application (Source)

Start Page (source)



Project Structure

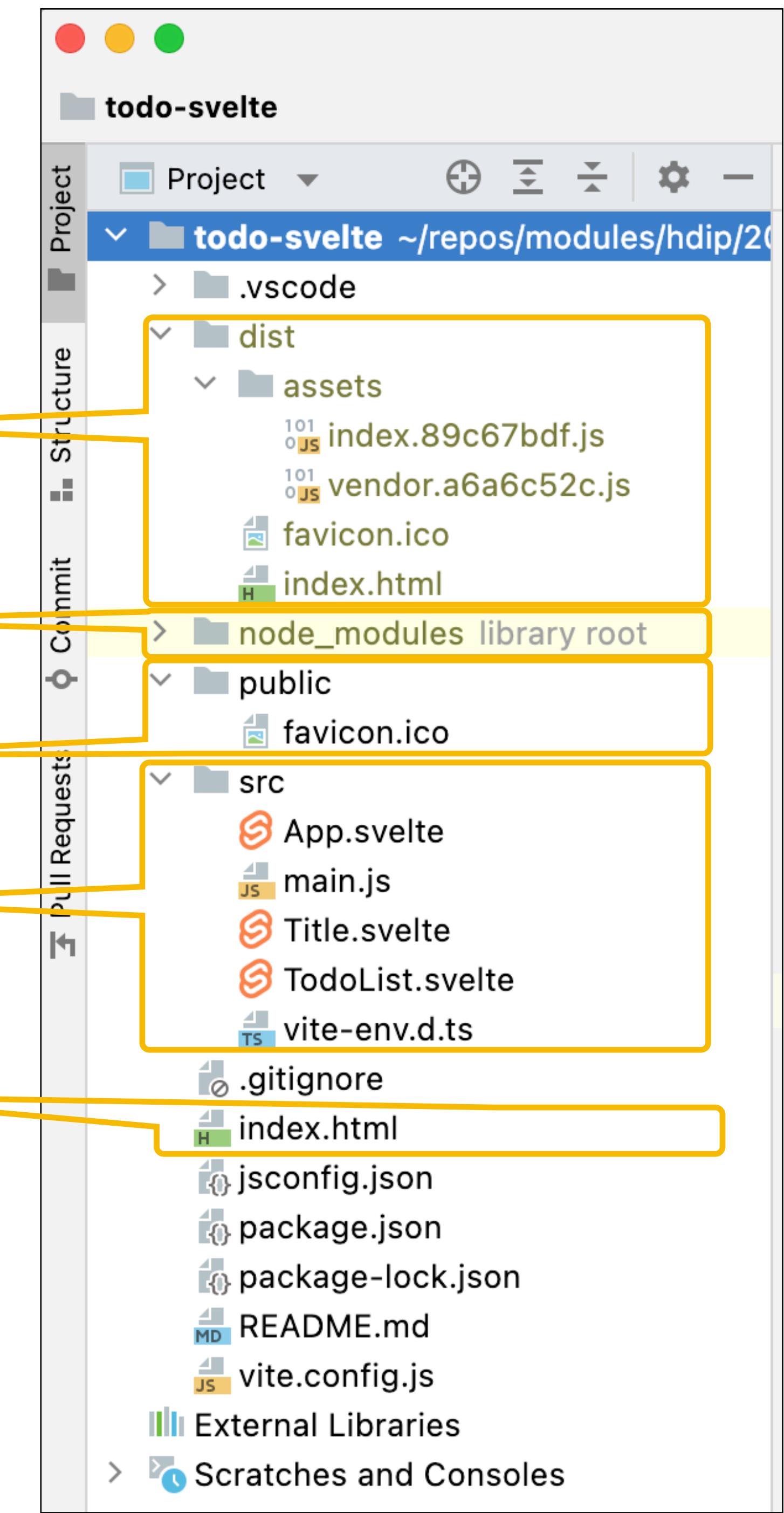
Generated (bundled) application

Application Modules

Static resources

Application (Source)

Start Page (source)



Svelte First Steps



Building your first Svelte
Application