Lab Exercises

Web Development Frameworks

zenika
ARCHITECTURE INFORMATIQUE

Lab 4: Svelte Todo App

Let's create a Todo app in Svelte!

In this lab, we will explore building a Todo App using Svelte.

Let's create a new boilerplate Svelte project from the official docs.

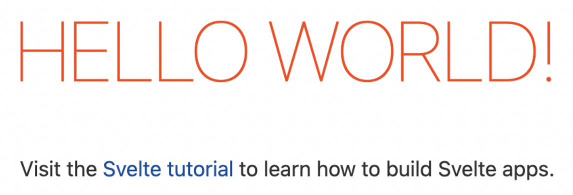
```
npx degit sveltejs/template svelteTodoApp
```

After the project has been initialized, change directory into `svelteTodoApp` .

Run the following commands:

```
cd svelteTodoApp
npm install
npm run dev
```

Open a new browser tab and visit `http://localhost:5000/` . You should see a welcome page!


HELLO WORLD!
Visit the Svelte tutorial to learn how to build Svelte apps.

Open the project folder in you code editor and open the `App.svelte` file. Delete all contents within `script` , `main` and `style` tags. You should only have a skeleton like so:

```
<script> </script>
<main></main>
<style></style>
```

Lets add the UI of our todoApp first. Paste the following into the `main` and `style` tags.

```
<main>
  <section class="todo-wrapper">
    <h2 class="todo-title">My Todo List</h2>
    <input type="text" />
    <div class="btn btn-add">+</div>
    <ul class="todo-list">
        <!-- list of todos here -->
    </ul>
  </section>
</main>

<style>
  :global(body) {
    padding: 0;
  }

  main {
```

```css
  display: flex;
  align-items: center;
  justify-content: center;
  background-color: #fefefe;
  background-image: linear-gradient(#fc6c48 0%, #ef5081 100%);
  background-repeat: no-repeat;
  background-size: cover;
  height: 100%;
}

.todo-wrapper {
  width: 400px;
  max-width: 100%;
  min-height: 500px;
  margin: 20px auto 40px;
  border: 1px solid #eee;
  border-radius: 4px;
  padding: 40px 20px;
  -webkit-box-shadow: 0 0 15px 0 rgba(0, 0, 0, 0.05);
  box-shadow: 0 0 15px 0 rgba(0, 0, 0, 0.05);
  background-color: #f4f7fc;
  overflow: hidden;
  position: relative;
}

.todo-title {
  font-size: 1.2em;
  color: #f65c65;
  font-weight: normal;
}

.btn,
input {
  line-height: 2em;
  border-radius: 3px;
  border: 0;
  display: inline-block;
  margin: 15px 0;
  padding: 0.2em 1em;
  font-size: 1em;
}

input[type="text"] {
  border: 1px solid #ddd;
  min-width: 80%;
  transition: all ease-in 0.25s;
}

input:focus {
  outline: none;
  border: 1px solid #a3b1ff;
}

.btn {
  text-align: center;
  font-weight: bold;
```

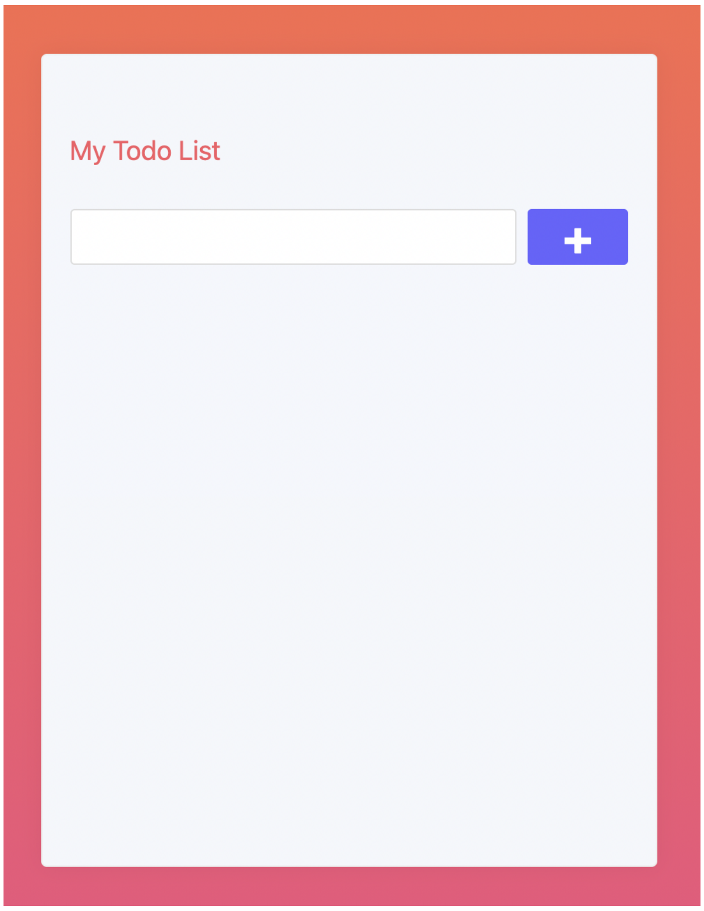
```
    cursor: pointer;
    border-width: 1px;
    border-style: solid;
  }

  .btn-add {
    background: #6664ff;
    border-color: #6664ff;
    pointer-events: visible;
    color: #fefefe;
    min-width: 12%;
    transition: all ease-in 0.25s;
    font-size: 2.2em;
    line-height: 0.5em;
    padding: 0.3em 0.3em;
    float: right;
  }

  ul.todo-list {
    padding: 0;
    margin-bottom: 30px;
  }

  ul.todo-list li {
    position: relative;
    list-style-type: none;
    display: block;
    margin: 10px 0;
    background: #e0e8f5;
    border-radius: 3px;
    padding: 12px;
    overflow: hidden;
  }
</style>
```

The app should now look like this.



Now inside the script tag, we will first declare a variable for the list of todos and the input value.

```
<script>
  let inputValue = "";
  let todos = [];
</script>
```

We will now bind the value of the `input` element with the variable `inputValue`.

```
<input type="text" bind:value={inputValue} />
```

In the `ul` tag, we will need to loop the `todos` array using a Svelte syntax `#each`.

```
    <ul class="todo-list">
      {#each todos as todo}
        <li>
          {todo}
        </li>
      {/each}
    </ul>
```

Finally we need to add the logic to insert a new todo value each time user types in the `input` element and clicks on the add button.

```
<script>
  let inputValue = "";
  let todos = [];

  function addTodo() {
    todos = [...todos, inputValue];
    inputValue = "";
  }
</script>
...
...
...
      <input type="text" bind:value={inputValue} />
      <div class="btn btn-add" on:click={addTodo}>+</div>
...
...
...
```

That's all! You can now insert new todo into the list!