

# Modules



Using npm modules in  
Svelte

# Todo Example - Recap

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.

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

## Simple Todo List

Fun things to do

What should I do?

Add Todo

Things yet do

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

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>
```



```
let todoItems = [];

function uuidv4() {
  return "xxxxxxx-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 renderAllTodos() {
  for (let i = 0; i < todoItems.length; i++) {
    renderTodo(todoItems[i]);
  }
}

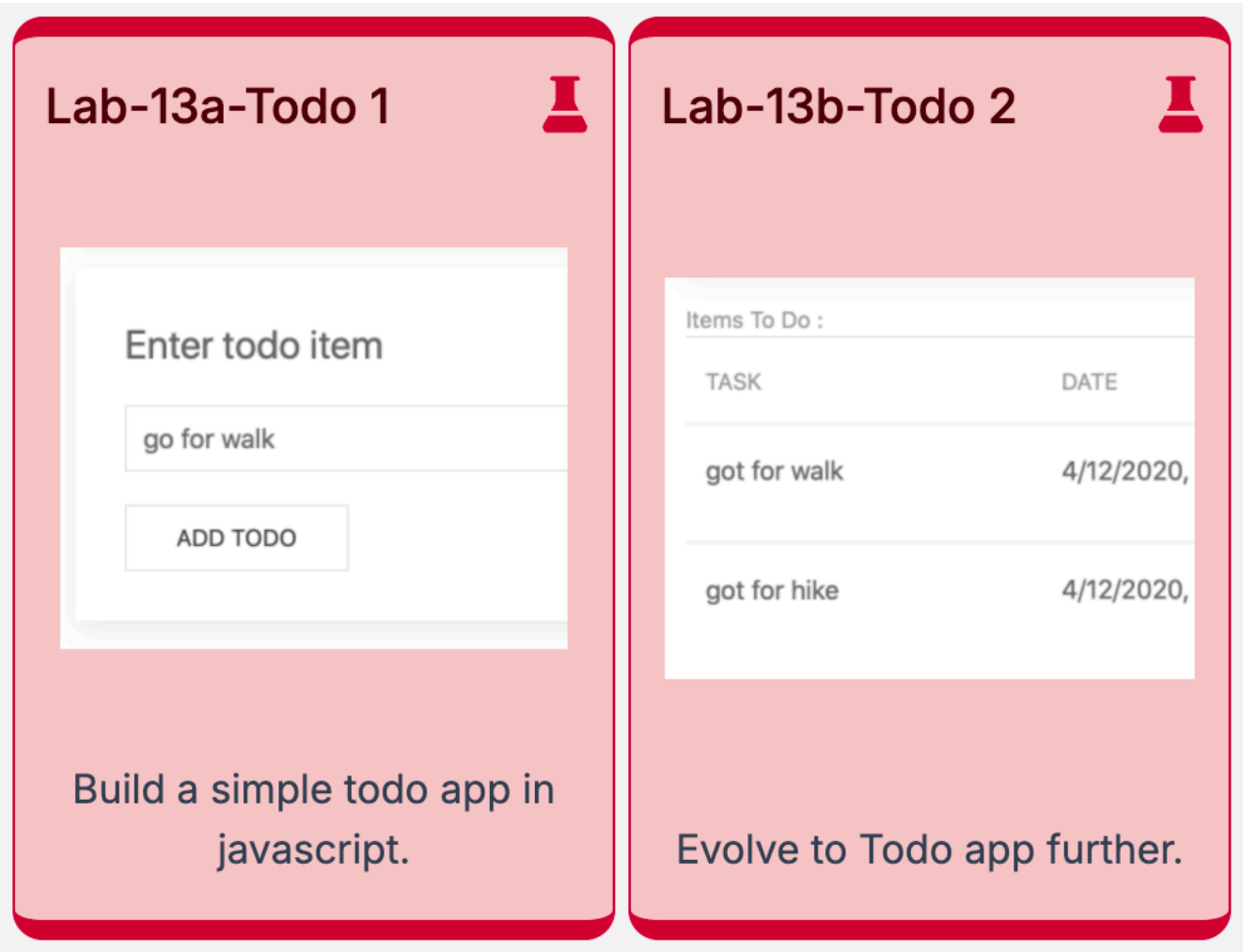
function deleteAllTodos() {
  let table = document.getElementById("todo-table");
  for (let i = 0; i < todoItems.length; i++) {
    table.deleteRow(-1);
  }
}

function renderTodo(todo) {
  const table = document.getElementById("todo-table");
  const row = table.insertRow(-1);
  const textCell = row.insertCell(0);
  textCell.innerText = todo.text;
  const dateCell = row.insertCell(1);
  dateCell.innerText = todo.date;
  const deleteCell = row.insertCell(2);
  deleteCell.innerHTML = `delete</a>`;
}

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

function deleteTodo(id) {
  deleteAllTodos();
  const found = todoItems.findIndex((todo) => todo.id === id);
  const done = todoItems[found];
  todoItems.splice(found, 1);
  renderAllTodos();
  addDone(done);
}

function addDone(doneItem) {
  const table = document.getElementById("done-table");
  const row = table.insertRow(-1);
  const textCell = row.insertCell(0);
  textCell.innerText = doneItem.text;
  const dateCell = row.insertCell(1);
  dateCell.innerText = doneItem.date;
}
```



# Todo Vanilla JS Project

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset='utf-8'>
    <meta name='viewport' content='width=device-width,initial-scale=1'>
    <title> Todo using Vanilla JS </title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bulma@0.9.3/css/bulma.min.css">
  </head>
  <body>
    <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 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="Type something...">
              </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>
    </div>
    <script src="todo.js" type="text/javascript"></script>
  </body>
</html>
```

# Svelte Project Structure

Generated (bundled) application

Application Modules

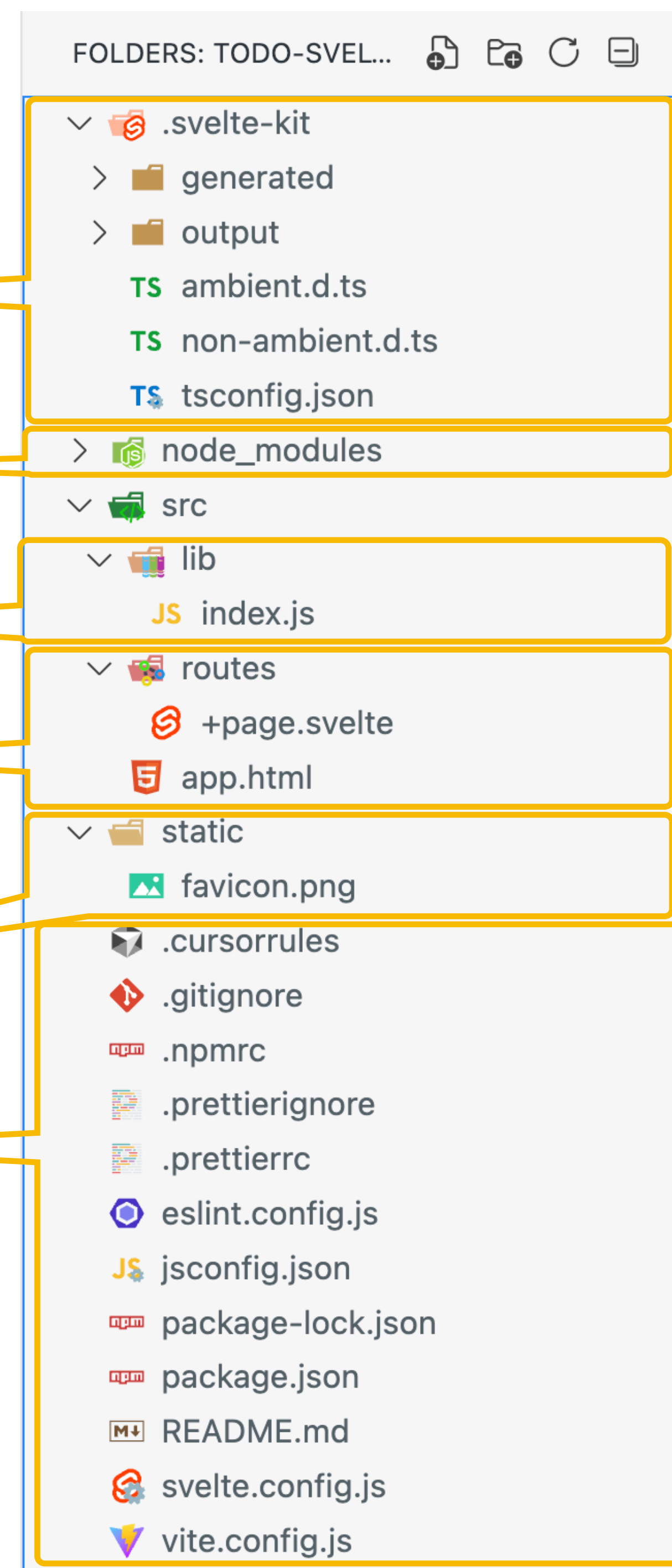
Shared Components

Application Source

Static resources

Configuration files

## Project Structure



```
<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>
```

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="container">
  <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 for="todo" class="label">What should I do?</label>
      </div>
      <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>
      </div>
    </div>
  </div>

  <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>
            <td><button onclick={() => deleteTodo(todo.id)} class="button">delete</button></td>
          </tr>
        {/each}
      </tbody>
    </table>
  </div>

  <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>
```

# Svelte Todo Sources

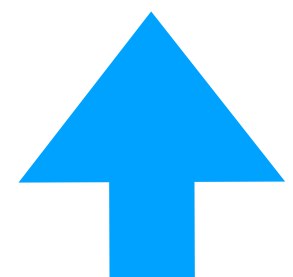
```

<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 = "";
}

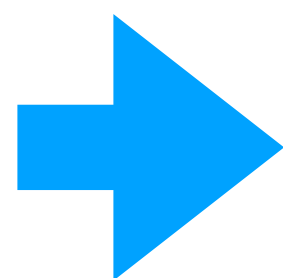
function deleteTodo(id) {
  doneItems.push(todoItems.find((todo) => todo.id === id));
  todoItems = todoItems.filter((todo) => todo.id !== id);
}
</script>

```



Svelte

Vanilla.js



```

let todoItems = [];

function uuidv4() {
  return "xxxxxxx-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 renderAllTodos() {
  for (let i = 0; i < todoItems.length; i++) {
    renderTodo(todoItems[i]);
  }
}

function deleteAllTodos() {
  let table = document.getElementById("todo-table");
  for (let i = 0; i < todoItems.length; i++) {
    table.deleteRow(-1);
  }
}

function renderTodo(todo) {
  const table = document.getElementById("todo-table");
  const row = table.insertRow(-1);
  const textCell = row.insertCell(0);
  textCell.innerHTML = todo.text;
  const dateCell = row.insertCell(1);
  dateCell.innerHTML = todo.date;
  const deleteCell = row.insertCell(2);
  deleteCell.innerHTML = `<a onclick="deleteTodo('${todo.id}')" class="uk-button uk-button-default">delete</a>`;
}

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

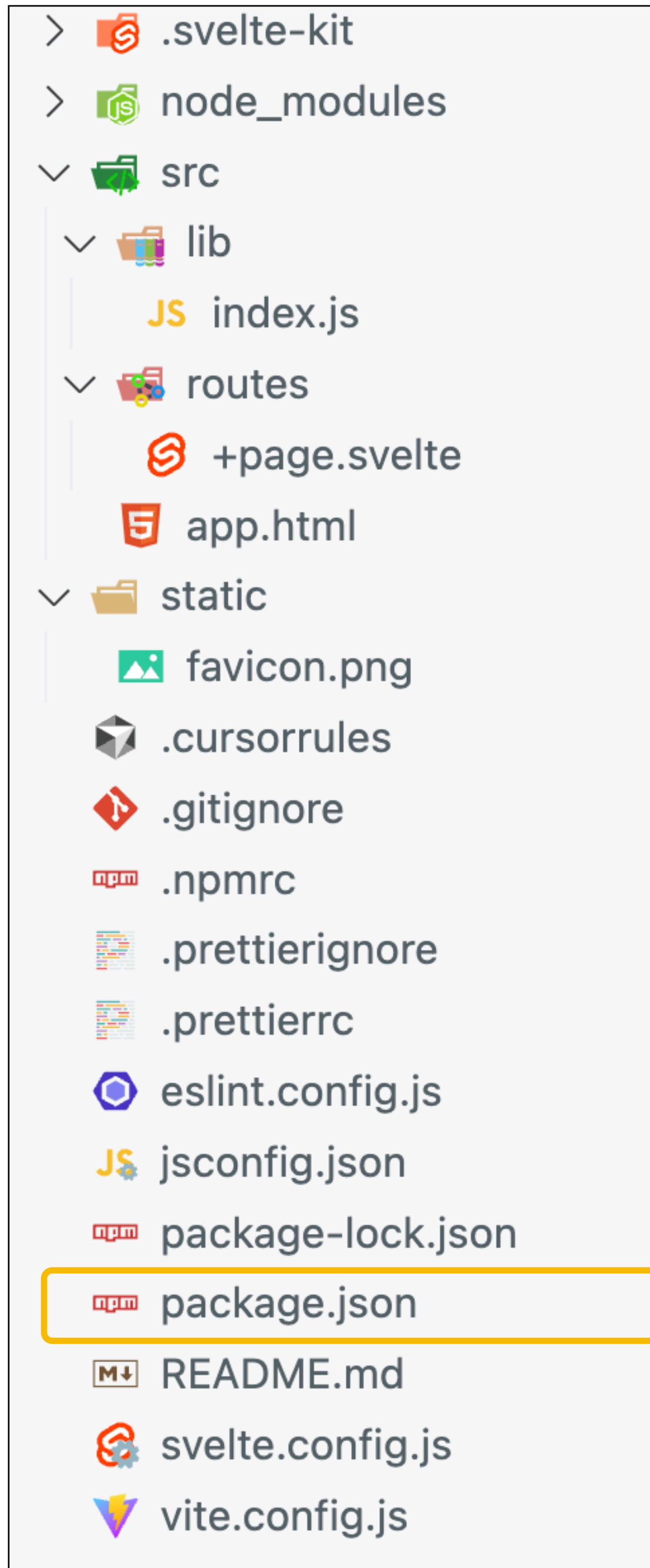
function deleteTodo(id) {
  deleteAllTodos();
  const found = todoItems.findIndex((todo) => todo.id == id);
  const done = todoItems[found];
  todoItems.splice(found, 1);
  renderAllTodos();
  addDone(done);
}

function addDone(doneItem) {
  const table = document.getElementById("done-table");
  const row = table.insertRow(-1);
  const textCell = row.insertCell(0);
  textCell.innerHTML = doneItem.text;
  const dateCell = row.insertCell(1);
  dateCell.innerHTML = doneItem.date;
}

```



# Todo Project Package.json

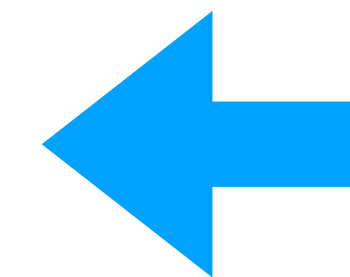


```
{
  "name": "todo",
  "private": true,
  "version": "0.0.1",
  "type": "module",
  "scripts": {
    "dev": "vite dev",
    "build": "vite build",
    "preview": "vite preview",
    "prepare": "svelte-kit sync || echo '",
    "format": "prettier --write .",
    "lint": "prettier --check . && eslint ."
  },
  "devDependencies": {
    "@eslint/compat": "^1.2.5",
    "@eslint/js": "^9.18.0",
    "@sveltejs/adapter-auto": "^4.0.0",
    "@sveltejs/kit": "^2.16.0",
    "@sveltejs/vite-plugin-svelte": "^5.0.0",
    "eslint": "^9.18.0",
    "eslint-config-prettier": "^10.0.1",
    "eslint-plugin-svelte": "^2.46.1",
    "globals": "^15.14.0",
    "prettier": "^3.4.2",
    "prettier-plugin-svelte": "^3.3.3",
    "svelte": "^5.0.0",
    "vite": "^6.0.0"
  },
  "dependencies": {
    "uuid": "^11.1.0"
  }
}
```

npm modules  
used to 'compile'  
the application,  
and bundle it  
into **/.svelte-kit**

*dependencies* are  
also bundled

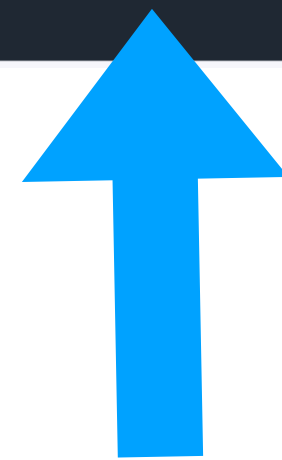
This means you  
can 'install' generic  
npm modules into  
your application





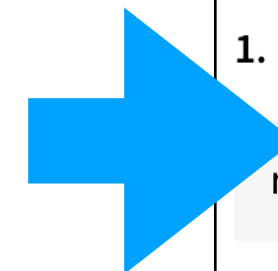
# Uuid function/npm package

```
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);  
  });  
}
```



We have developed our own uuid generation algorithm

We could use this library instead



## uuid

CI passing Browser passing

For the creation of **RFC4122** UUIDs

- **Complete** - Support for RFC4122 version 1, 3, 4, and 5 UUIDs
- **Cross-platform** - Support for ...
  - CommonJS, **ECMAScript Modules** and **CDN builds**
  - Node 8, 10, 12, 14
  - Chrome, Safari, Firefox, Edge, IE 11 browsers
  - Webpack and rollup.js module bundlers
  - **React Native / Expo**
- **Secure** - Cryptographically-strong random values
- **Small** - Zero-dependency, small footprint, plays nice with "tree shaking" packagers
- **CLI** - Includes the **uuid command line** utility

Upgrading from **uuid@3.x** ? Your code is probably okay, but check out **Upgrading From uuid@3.x** for details.

### Quickstart

To create a random UUID...

1. Install

```
npm install uuid
```
2. Create a UUID (ES6 module syntax)

```
import { v4 as uuidv4 } from 'uuid';  
uuidv4(); // => '9b1deb4d-3b7d-4bad-9bdd-2b0d7b3dcb6d'
```

### Install

```
> npm i uuid
```

### Weekly Downloads

50,603,900

Version	License
8.3.2	MIT

Unpacked Size	Total Files
116 kB	66

Issues	Pull Requests
15	3

Homepage  
[github.com/uuidjs/uuid#readme](https://github.com/uuidjs/uuid#readme)

Repository  
[github.com/uuidjs/uuid](https://github.com/uuidjs/uuid)

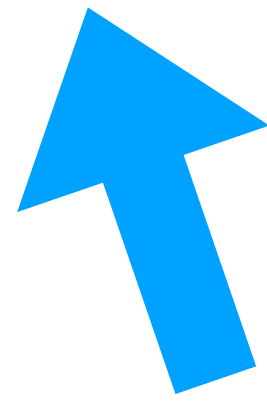
Last publish  
4 months ago

Collaborators

Try on RunKit

<https://www.npmjs.com/package/uuid>

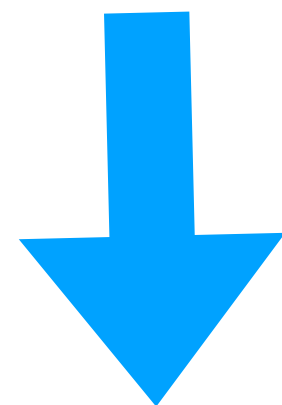
```
npm install uuid
```



Install module in project

Inserts new entry into package

Remove custom implementation,  
replace with import



```
import { v4 as uuidv4 } from 'uuid';
```

```
{
  "name": "todo",
  "private": true,
  "version": "0.0.1",
  "type": "module",
  > Debug
  "scripts": {
    "dev": "vite dev",
    "build": "vite build",
    "preview": "vite preview",
    "prepare": "svelte-kit sync || echo '",
    "format": "prettier --write .",
    "lint": "prettier --check . && eslint ."
  },
  "devDependencies": {
    "@eslint/compat": "^1.2.5",
    "@eslint/js": "^9.18.0",
    "@sveltejs/adapter-auto": "^4.0.0",
    "@sveltejs/kit": "^2.16.0",
    "@sveltejs/vite-plugin-svelte": "^5.0.0",
    "eslint": "^9.18.0",
    "eslint-config-prettier": "^10.0.1",
    "eslint-plugin-svelte": "^2.46.1",
    "globals": "^15.14.0",
    "prettier": "^3.4.2",
    "prettier-plugin-svelte": "^3.3.3",
    "svelte": "^5.0.0",
    "vite": "^6.0.0"
  },
  "dependencies": {
    "uuid": "^11.1.0"
  }
}
```

```
<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>
```



Vast range of npm modules now  
easily accessible



# Svelte vs Vanilla JS

```
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);
}
```

20 lines

```
let todoItems = [];

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 renderAllTodos() {
  for (let i = 0; i < todoItems.length; i++) {
    renderTodo(todoItems[i]);
  }
}

function deleteAllTodos() {
  let table = document.getElementById("todo-table");
  for (let i = 0; i < todoItems.length; i++) {
    table.deleteRow(-1);
  }
}

function renderTodo(todo) {
  const table = document.getElementById("todo-table");
  const row = table.insertRow(-1);
  const textCell = row.insertCell(0);
  textCell.innerHTML = todo.text;
  const dateCell = row.insertCell(1);
  dateCell.innerHTML = todo.date;
  const deleteCell = row.insertCell(2);
  deleteCell.innerHTML = `
```

12

61 lines

# Modules



Using npm modules in  
Svelte