1) Install prerequisites

1. Install **Node.js (LTS)** from nodejs.org
2. Verify:

```
node -v
npm -v
```

2) Create a project

```
mkdir my-ts-webapp
cd my-ts-webapp
npm init -y
```

3) Add TypeScript (dev dependency)

```
npm i -D typescript @types/node
```

4) Initialize TypeScript config

```
npx tsc --init --rootDir src --outDir public/js --target ES2020 --module
ES2020 --strict --sourceMap
```

Open the generated `tsconfig.json` and ensure these are set (add if missing):

```
{
  "compilerOptions": {
    "rootDir": "src",
    "outDir": "public/js",

    "module": "es2020",
    "target": "es2020",
    "lib": ["es2020", "dom"],

    "sourceMap": true,

    "strict": true,
    "noUncheckedIndexedAccess": true,
    "exactOptionalPropertyTypes": true,

    "isolatedModules": true,
    "skipLibCheck": true
  },
  "include": ["src"]
}
```

5) Add your code & HTML

Create folders:

```
mkdir src public
```

public/index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>TS Web App</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
  </head>
  <body>
    <h1>TypeScript Demo</h1>

    <form id="todo-form">
      <input id="todo-input" placeholder="Add a task" />
      <button type="submit">Add</button>
    </form>

    <ul id="todo-list"></ul>

    <!-- Load the compiled JS (ES module) -->
    <script type="module" src="./js/main.js"></script>
  </body>
</html>
```

src/main.ts

```
// Strongly-typed data model
interface Todo {
  id: number;
  text: string;
  done: boolean;
}

const form = document.getElementById('todo-form') as HTMLFormElement;
const input = document.getElementById('todo-input') as HTMLInputElement;
const list  = document.getElementById('todo-list') as HTMLUListElement;

let todos: Todo[] = [];

function render(items: Todo[]): void {
  list.innerHTML = '';
  for (const t of items) {
    const li = document.createElement('li');
```

```
    li.textContent = `${t.text} ${t.done ? '✅' : ''}`;
    li.tabIndex = 0;
    li.addEventListener('click', () => toggle(t.id));
     // Keyboard accessibility
    li.addEventListener('keydown', (e) => {
      if (e.key === 'Enter' || e.key === ' ') toggle(t.id);
    });
    list.appendChild(li);
  }
}

function add(text: string): void {
  const todo: Todo = { id: Date.now(), text, done: false };
  todos = [todo, ...todos];
  render(todos);
}

function toggle(id: number): void {
  todos = todos.map(t => t.id === id ? { ...t, done: !t.done } : t);
  render(todos);
}

// Small generic util (TypeScript perk!)
function nonEmpty<T extends { length: number }>(v: T): boolean {
  return v.length > 0;
}

form.addEventListener('submit', (e) => {
  e.preventDefault();
  const value = input.value.trim();
  if (nonEmpty(value)) {
    add(value);
    input.value = '';
  } else {
    alert('Please type a task');
  }
});

render(todos);
```

6) Compile TypeScript → JavaScript

One-off build:

```
npx tsc
```

Watch mode (rebuild on save):

```
npx tsc -w
```

7) Preview in the browser (from the command prompt)

Install a tiny static server (dev-only):

```
npm i -D http-server
npx http-server public -p 5173
```

Then open the shown URL (e.g., `http://127.0.0.1:5173`).

Tip: If you prefer, `npx live-server public` also works; it auto-reloads on changes.

Using libraries with types (optional)

```
npm i lodash
npm i -D @types/lodash
```

Then import in `src/main.ts`:

```
import { uniq } from 'lodash';
console.log(uniq([1,1,2,3]));
```

Recompile with `npx tsc` (or keep `npx tsc -w` running).