Project met TypeScript (Frontend)

Doel van het project

In dit project leren jullie hoe ze **TypeScript** gebruiken in een **frontend webproject** zonder frameworks (zoals Angular of React).
De focus ligt op:

- Werken met **TypeScript**
- Begrijpen van het verschil tussen **TypeScript en JavaScript**
- Werken met de **DOM (document object model)**
- Type safety en `tsc` (TypeScript compiler)

Wat ga je leren?

Na het afronden van dit project kunnen jullie:

- Uitleggen wat TypeScript is en waarom het wordt gebruikt
- TypeScript compileren naar JavaScript met `tsc`
- DOM-elementen selecteren en manipuleren
- Events gebruiken (`click`)
- Een eenvoudige takenlijst bouwen in de browser

Gebruikte technologieën

- **HTML5**
- **CSS3**
- **TypeScript**
- **JavaScript (gegenereerd door TypeScript)**
- **Browser (Chrome / Edge / Firefox)**

❗ **Node.js wordt NIET gebruikt om de applicatie uit te voeren**

Projectstructuur

```
ProjectTypescript/
│
├── src/
│   ├── index.html        // HTML-structuur van de pagina
│   ├── index.css         // Styling van de applicatie
│   ├── index.ts          // TypeScript broncode (zelf geschreven)
│   └── index.js          // Gegenereerde JavaScript (door tsc)
│
└── tsconfig.json         // Configuratie voor de TypeScript compiler
```

Beschrijving van de bestanden

◇ index.html

Bevat de structuur van de webpagina:

- Invoerveld voor taken
- Knoppen voor toevoegen en verwijderen
- Een lijst (`ul`) voor taken

De JavaScript wordt gekoppeld via:`<script src="index.js"></script>`

```html
<!DOCTYPE html>

<html lang="en">


<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,

initial-scale=1.0">
```

```html
   <!-- This makes the connection to our compiled ts
code. The 'defer' attribute ensures the script runs
after HTML parsing, preventing blocking and relying on
the fully constructed DOM. -->

   <script src="../dist/app.js" type="module"
defer></script>

   <!-- connect to our stylesheet -->
   <link rel="stylesheet" href="index.css">
   <title>Tasks</title>
</head>

<body>
<h1>Tasks</h1>
<input type="text" id="task-input" placeholder="Add
new task">
<button id="add-button">Add</button>
<ul id="task-list"></ul>
<button id="clear-button">Clear all</button>
<script src="index.js"></script>
</body>
</html>
```

◇ index.css

Verantwoordelijk voor:

- Layout
- Kleuren
- Knoppen en lijstweergave

Dit bestand heeft **geen invloed op de logica** van het programma.

```
*{
margin: 0;
padding: 0;
box-sizing: border-box;
}
body {
font-family: Arial, sans-serif;
text-align: center;
background-color: #f0f0f0;
}
h1 {
    color: #333;
}

#task-input {
padding: 10px;
```

```
width: 60%;
border: 1px solid #ccc;
}
#add-button {
background-color: #007bff;
}
button {
padding: 10px 20px;
margin: 10px;
background-color: #d9534f;
color: white;
border: none;
cursor: pointer;
}
ul {
    list-style: none;
padding: 0;
}
li {

background-color: white;
padding: 10px;
margin: 10px;
border: 1px solid #ccc;
```

```
display: flex;
justify-content: space-between;
align-items: center;
}
```

◇ index.ts

De kern van het project.

Functionaliteiten:

- Taken toevoegen
- Taken verwijderen
- Alle taken wissen
- Werken met `document.getElementById`
- Type assertions (as `HTMLInputElement`)
- Event listeners (`addEventListener`)

```
// GETTING THE ELEMENTS FROM THE HTML FILE
const taskInput = document.getElementById("task-
input") as HTMLInputElement;
const taskList = document.getElementById("task-list")
as HTMLUListElement;
const addButton = document.getElementById("add-
button");
const clearButton = document.getElementById("clear-
button");
```

```
// THIS CHECK (IF) WE NEED THIS BECAUSE TS IS NOT
HAPPY WITH POSSIBLE NULL VALUES
if (taskInput && addButton && clearButton) {
    addButton.addEventListener("click", () => {
        const taskText = taskInput.value.trim();
            // CREATE LIST ITEM
        if (taskText !== "") {
                const listItem =
document.createElement("li");
                    listItem.textContent = taskText;
                    // DELETE TASK
                    const deleteButton =
document.createElement("button");
                    deleteButton.textContent =
"Delete";

deleteButton.addEventListener("click", () => {
                    taskList.removeChild(listItem);
                });
                listItem.appendChild(deleteButton);
                taskList.appendChild(listItem);
                // CLEAR INPUT FIELD
                taskInput.value = "";
        }
    });
    // CLEARING COMPLETED TASKS
    clearButton.addEventListener("click", () => {
        const completedTasks =
taskList.querySelectorAll("li");
        // LOOP OVER ALL THE TASKS AND DELETE THEM
        completedTasks.forEach((task) => {
            if (task.querySelector("button")) {
                taskList.removeChild(task);
            }
        });
    });
}
```

index.js

- Automatisch gegenereerd door TypeScript
- Wordt uitgevoerd door de browser
- Mag **niet handmatig worden aangepast**

tsconfig.json

Bevat instellingen voor de TypeScript compiler, zoals:

- Target JavaScript versie
- Strikte type-controle (`strict: true`)

```
{
  "compilerOptions": {
    /* Visit https://aka.ms/tsconfig to read more
about this file */

    /* Projects */
    // "incremental": true,
/* Save .tsbuildinfo files to allow for incremental
compilation of projects. */
    // "composite": true,
/* Enable constraints that allow a TypeScript project
to be used with project references. */
    // "tsBuildInfoFile": "./.tsbuildinfo",
/* Specify the path to .tsbuildinfo incremental
compilation file. */
    // "disableSourceOfProjectReferenceRedirect":
true,  /* Disable preferring source files instead of
declaration files when referencing composite projects.
*/
    // "disableSolutionSearching": true,
/* Opt a project out of multi-project reference
checking when editing. */
```

```
    // "disableReferencedProjectLoad": true,
/* Reduce the number of projects loaded automatically
by TypeScript. */

    /* Language and Environment */
    "target": "es2016",
/* Set the JavaScript language version for emitted
JavaScript and include compatible library
declarations. */
    // "lib": [],
/* Specify a set of bundled library declaration files
that describe the target runtime environment. */
    // "jsx": "preserve",
/* Specify what JSX code is generated. */
    // "experimentalDecorators": true,
/* Enable experimental support for legacy experimental
decorators. */
    // "emitDecoratorMetadata": true,
/* Emit design-type metadata for decorated
declarations in source files. */
    // "jsxFactory": "",
/* Specify the JSX factory function used when
targeting React JSX emit, e.g. 'React.createElement'
or 'h'. */
```

```
    // "jsxFragmentFactory": "",
/* Specify the JSX Fragment reference used for
fragments when targeting React JSX emit e.g.
'React.Fragment' or 'Fragment'. */
    // "jsxImportSource": "",
/* Specify module specifier used to import the JSX
factory functions when using 'jsx: react-jsx*'. */
    // "reactNamespace": "",
/* Specify the object invoked for 'createElement'.
This only applies when targeting 'react' JSX emit. */
    // "noLib": true,
/* Disable including any library files, including the
default lib.d.ts. */
    // "useDefineForClassFields": true,
/* Emit ECMAScript-standard-compliant class fields. */
    // "moduleDetection": "auto",
/* Control what method is used to detect module-format
JS files. */

    /* Modules */
    "module": "commonjs",
/* Specify what module code is generated. */
```

```
    // "rootDir": "./",
/* Specify the root folder within your source files.
*/
    // "moduleResolution": "node10",
/* Specify how TypeScript looks up a file from a given
module specifier. */
    // "baseUrl": "./",
/* Specify the base directory to resolve non-relative
module names. */
    // "paths": {},
/* Specify a set of entries that re-map imports to
additional lookup locations. */
    // "rootDirs": [],
/* Allow multiple folders to be treated as one when
resolving modules. */
    // "typeRoots": [],
/* Specify multiple folders that act like
'./node_modules/@types'. */
    // "types": [],
/* Specify type package names to be included without
being referenced in a source file. */
    // "allowUmdGlobalAccess": true,
/* Allow accessing UMD globals from modules. */
```

```
    // "moduleSuffixes": [],
/* List of file name suffixes to search when resolving
a module. */
    // "allowImportingTsExtensions": true,
/* Allow imports to include TypeScript file
extensions. Requires '--moduleResolution bundler' and
either '--noEmit' or '--emitDeclarationOnly' to be
set. */
    // "resolvePackageJsonExports": true,
/* Use the package.json 'exports' field when resolving
package imports. */
    // "resolvePackageJsonImports": true,
/* Use the package.json 'imports' field when resolving
imports. */
    // "customConditions": [],
/* Conditions to set in addition to the resolver-
specific defaults when resolving imports. */
    // "resolveJsonModule": true,
/* Enable importing .json files. */
    // "allowArbitraryExtensions": true,
/* Enable importing files with any extension, provided
a declaration file is present. */
```

```
    // "noResolve": true,
/* Disallow 'import's, 'require's or '<reference>'s
from expanding the number of files TypeScript should
add to a project. */

    /* JavaScript Support */
    // "allowJs": true,
/* Allow JavaScript files to be a part of your
program. Use the 'checkJS' option to get errors from
these files. */
    // "checkJs": true,
/* Enable error reporting in type-checked JavaScript
files. */
    // "maxNodeModuleJsDepth": 1,
/* Specify the maximum folder depth used for checking
JavaScript files from 'node_modules'. Only applicable
with 'allowJs'. */

    /* Emit */
    // "declaration": true,
/* Generate .d.ts files from TypeScript and JavaScript
files in your project. */
    // "declarationMap": true,
/* Create sourcemaps for d.ts files. */
```

```
    // "emitDeclarationOnly": true,
/* Only output d.ts files and not JavaScript files. */
    // "sourceMap": true,
/* Create source map files for emitted JavaScript
files. */
    // "inlineSourceMap": true,
/* Include sourcemap files inside the emitted
JavaScript. */
    // "outFile": "./",
/* Specify a file that bundles all outputs into one
JavaScript file. If 'declaration' is true, also
designates a file that bundles all .d.ts output. */
    // "outDir": "./",
/* Specify an output folder for all emitted files. */
    // "removeComments": true,
/* Disable emitting comments. */
    // "noEmit": true,
/* Disable emitting files from a compilation. */
    // "importHelpers": true,
/* Allow importing helper functions from tslib once
per project, instead of including them per-file. */
```

```
    // "importsNotUsedAsValues": "remove",
/* Specify emit/checking behavior for imports that are
only used for types. */
    // "downlevelIteration": true,
/* Emit more compliant, but verbose and less
performant JavaScript for iteration. */
    // "sourceRoot": "",
/* Specify the root path for debuggers to find the
reference source code. */
    // "mapRoot": "",
/* Specify the location where debugger should locate
map files instead of generated locations. */
    // "inlineSources": true,
/* Include source code in the sourcemaps inside the
emitted JavaScript. */
    // "emitBOM": true,
/* Emit a UTF-8 Byte Order Mark (BOM) in the beginning
of output files. */
    // "newLine": "crlf",
/* Set the newline character for emitting files. */
    // "stripInternal": true,
/* Disable emitting declarations that have '@internal'
in their JSDoc comments. */
```

```
    // "noEmitHelpers": true,
/* Disable generating custom helper functions like
'__extends' in compiled output. */
    // "noEmitOnError": true,
/* Disable emitting files if any type checking errors
are reported. */
    // "preserveConstEnums": true,
/* Disable erasing 'const enum' declarations in
generated code. */
    // "declarationDir": "./",
/* Specify the output directory for generated
declaration files. */
    // "preserveValueImports": true,
/* Preserve unused imported values in the JavaScript
output that would otherwise be removed. */

    /* Interop Constraints */
    // "isolatedModules": true,
/* Ensure that each file can be safely transpiled
without relying on other imports. */
    // "verbatimModuleSyntax": true,
/* Do not transform or elide any imports or exports
not marked as type-only, ensuring they are written in
```

```
the output file's format based on the 'module'
setting. */
    // "allowSyntheticDefaultImports": true,
/* Allow 'import x from y' when a module doesn't have
a default export. */
    "esModuleInterop": true,
/* Emit additional JavaScript to ease support for
importing CommonJS modules. This enables
'allowSyntheticDefaultImports' for type compatibility.
*/
    // "preserveSymlinks": true,
/* Disable resolving symlinks to their realpath. This
correlates to the same flag in node. */
    "forceConsistentCasingInFileNames": true,
/* Ensure that casing is correct in imports. */

    /* Type Checking */
    "strict": true,
/* Enable all strict type-checking options. */
    // "noImplicitAny": true,
/* Enable error reporting for expressions and
declarations with an implied 'any' type. */
```

```
    // "strictNullChecks": true,
/* When type checking, take into account 'null' and
'undefined'. */
    // "strictFunctionTypes": true,
/* When assigning functions, check to ensure
parameters and the return values are subtype-
compatible. */
    // "strictBindCallApply": true,
/* Check that the arguments for 'bind', 'call', and
'apply' methods match the original function. */
    // "strictPropertyInitialization": true,
/* Check for class properties that are declared but
not set in the constructor. */
    // "noImplicitThis": true,
/* Enable error reporting when 'this' is given the
type 'any'. */
    // "useUnknownInCatchVariables": true,
/* Default catch clause variables as 'unknown' instead
of 'any'. */
    // "alwaysStrict": true,
/* Ensure 'use strict' is always emitted. */
```

```
    // "noUnusedLocals": true,
/* Enable error reporting when local variables aren't
read. */
    // "noUnusedParameters": true,
/* Raise an error when a function parameter isn't
read. */
    // "exactOptionalPropertyTypes": true,
/* Interpret optional property types as written,
rather than adding 'undefined'. */
    // "noImplicitReturns": true,
/* Enable error reporting for codepaths that do not
explicitly return in a function. */
    // "noFallthroughCasesInSwitch": true,
/* Enable error reporting for fallthrough cases in
switch statements. */
    // "noUncheckedIndexedAccess": true,
/* Add 'undefined' to a type when accessed using an
index. */
    // "noImplicitOverride": true,
/* Ensure overriding members in derived classes are
marked with an override modifier. */
```

```
    // "noPropertyAccessFromIndexSignature": true,
/* Enforces using indexed accessors for keys declared
using an indexed type. */
    // "allowUnusedLabels": true,
/* Disable error reporting for unused labels. */
    // "allowUnreachableCode": true,
/* Disable error reporting for unreachable code. */

    /* Completeness */
    // "skipDefaultLibCheck": true,
/* Skip type checking .d.ts files that are included
with TypeScript. */
    "skipLibCheck": true
/* Skip type checking all .d.ts files. */
  }
}
```

▶️ Hoe start je het project?

1. Compileer TypeScript:

```
tsc
```

2. Open `index.html` in de browser
   (dubbelklik of via “Open with browser”)

❌ **Gebruik GEEN node `index.js`**

Belangrijke regels

- `document` bestaat alleen in de **browser**
- TypeScript (`.ts`) kan niet direct worden uitgevoerd
- Node.js heeft **geen toegang tot de DOM**
- JavaScript (`.js`) wordt uitgevoerd in de browser

Verwachte functionaliteit

- De gebruiker kan een taak invoeren
- Taken verschijnen in een lijst
- Elke taak kan individueel worden verwijderd
- Alle taken kunnen tegelijk worden gewist