**Using "declare" Manually**

As you saw in the previous lectures, you can use .d.ts files (indirectly, by installing @types packages OR by creating them manually) to add missing type declarations.

Alternatively, if you maybe just need one or two missing type declarations, you can also declare those missing types yourself in your own .ts files (*note: NOT .d.ts!*).

For example, in a HTML file (e.g., index.html) you may define a global variable like this:

<body>

<script>

var MODE = 'DEFAULT';

</script>

</body>

What if your other code is written in TypeScript and stored in other files but relies on that variable?

For example, you may have code like this in some TypeScript file:

const selectedMode = MODE;

TypeScript will complain about this line since MODE is not defined in the TypeScript file itself. It does not know it. And even if it would know that MODE exists, TypeScript would not know which type of value may be stored by MODE.

That's where a custom type declaration via the declare keyword can come in handy.

You can "convince" TypeScript that MODE exists and stores a value of type string by adjusting your TypeScript code like this:

declare var MODE: string;

const selectedMode = MODE;

declare "tells" TypeScript that a certain "thing" (in this case a variable) exists and is of a certain type.

You can also declare other things - for example the existence and structure of a namespace that may contain various types:

declare namespace D3 {

export interface Selectors {

select: {

(selector: string): Selection;

(element: EventTarget): Selection;

};

}

export interface Event {

x: number;

y: number;

}

export interface Base extends Selectors {

event: Event;

}

}

If you explore the .d.ts files of popular packages like lodash you will find many such declare statements in those files.