Definitions and Usages

This lesson is a brief introduction to index signatures with TypeScript.

WE'LL COVER THE FOLLOWING

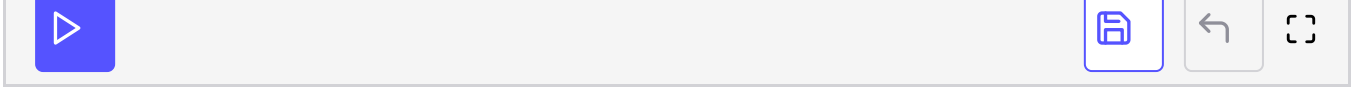
Introduction

JavaScript allows accessing an object's members by using square brackets `[]` with the name of the member between them. It allows for reaching a value dynamically without having to use dot notation. This technique is called accessing by index signature, and it is available to any object for assignment or reading values.

TypeScript brings the game a notch higher with better support for improved control over the manipulation of an index. First, you can only access this via string or number. With JavaScript, you could use an object which would fall back to the `toString` function, for example. This leads to some issues when using an object because `toString` returns `[object Object]`.

In the case of an index map with a number, members can be of any type. The following `example` has `m4` which is an object that contains two primitives but also `m1` as a `string` and `m3` as a `boolean`.

```
interface MyStringDictionaryWithMembers3 {
    [key: number]: string;
    m1: string;
    m2: number;
    m3: boolean;
    m4: { x: string; y: number };
}
```



Configuration

TypeScript (with the option `noImplicitAny` set to true) won't allow you to access a member that isn't defined when working on a strongly typed. This option should be set to true for every new project because it enforces a stronger type of validation.

```
let objectIndex: { x: number; y: string } = { x: 1, y: "2" };
objectIndex["x"] = 2;
objectIndex["y"] = "a string";
// objectIndex[0] = 1; // Doesn't transpile
// objectIndex[{ x: 1 }] = 123; // Doesn't transpile
console.log(objectIndex); // { '0': 1, x: 2, y: '2' }
```