**TypeScript**

**TypeScript**

Typescript is a typed superset of javacript that compiles to plain javascript.

It will always compile a typescript code into a javascript, even a invalid one.

Objects can be declared by using “Interface”.

**Why we should use typescript?**

TypeScript is a superset of JavaScript which primarily provides optional static typing, classes and interfaces. One of the big benefits is to enable IDEs to provide a richer environment for spotting common errors as you type the code.

To get an idea of what I mean, watch Microsoft's introductory video on the language.

For a large JavaScript project, adopting TypeScript might result in more robust software, while still being deployable where a regular JavaScript application would run.

**Null and Undefined**

By default null and undefined are subtypes of all other types. That means you can assign null and undefined to something like number.

However, when using the --strictNullChecks flag, null and undefined are only assignable to void and their respective types. This helps avoid many common errors.

As a note: we encourage the use of --strictNullChecks when possible, but for the purposes of this handbook, we will assume it is turned off.

**Fat arrow function**

<https://www.youtube.com/watch?v=Fu0HKDNl0rI>

Each optional property denoted by a  “?”

**Rest functions**

The compiler will build an array of the arguments passed in with the name given after the ellipsis (...), allowing you to use it in your function.

Void 0 and undefined is same.

**Interface**

One of TypeScript’s core principles is that type-checking focuses on the shape that values have. This is sometimes called “duck typing” or “structural subtyping”.