<https://projects.100xdevs.com/tracks/ts-hard/ts-hard-1>  
------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**How to know if the api belongs to typescript/javascript:**

If typescript compiler understands the things, i.e., if it’s there in it’s code, then it is specific to typescript. If it is present in index.js files, then it will be belonging to javascript compiler/engine.

**How to create a typescript project and run it**

**Step 1: npm init -y**

The **npm init -y** command is a shortcut for initializing a new Node.js project with default values for the package.json file without having to manually answer prompts.

Here's what each part does:

* **npm init**: This initializes a new Node.js project. It prompts you to provide information about your project, such as its name, version, description, entry point, test command, repository, keywords, author, and license.
* **-y** flag: This flag stands for "yes" and automatically accepts all default values for the prompts, effectively skipping the interactive part of the initialization process. It's useful when you want to quickly create a new project with default settings without being prompted for each detail.

**Step 2: npx tsc –init**

If you install TypeScript locally (**npm install typescript**), you will have a local copy of TypeScript within your project's **node\_modules** directory. You can then use **npx tsc --init** to run the TypeScript compiler (**tsc**) from your local project's **node\_modules/.bin** directory to generate the **tsconfig.json** file.

If you install TypeScript globally (**npm install -g typescript**), you will have **tsc** available globally on your system. In this case, you can simply run **tsc --init** without **npx** from anywhere on your system to generate the **tsconfig.json** file.

Either way, whether you choose to install TypeScript locally or globally depends on your preference and project requirements.

**Step 3: change rootDir in tsconfig.json file to “./src” and outdir set as ‘./dist’**

**Step 4: create folder ‘src’ and file ‘index.ts’ inside it.**

**Step 5: tsc -b**

When you run **tsc -b**, TypeScript looks for **tsconfig.json** files in the current directory and its subdirectories. It then builds each project based on the configuration specified in its respective **tsconfig.json** file.

**Step 6: node dist/index.js**

**‘pick’ api:**

When you are updating the database, and updating only few properties of a model. Instead of giving each property separatelyl as an argument, we can pick them from the model.

**‘partial’ api:**

Partial makes all properties of a type optional, creating a type with the same properties but each marked as optional.

**‘Readonly ’ api:**

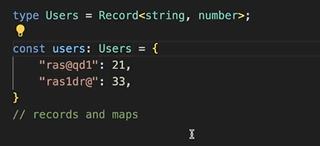
Readonly ensures objects properties can’t be changed after initialization.

**‘Record’ and ‘Map’:**

We can write key maps like this in typescript



This is little bit unclean syntax wise. So, we can write like this using records

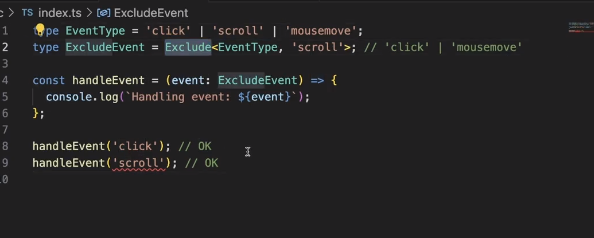


For maps,



**‘Exclude’ api:**

To exclude certain values from given lateral.



**‘Type inference’ in zod:**

In zod, we will write schema for all the variables in the object. And, we also define user object also. Instead of declaring the same variables twice, can we infer variables of object somehow from schema, instead of only checking for the validity.

->Get any info of ‘zod’ on ‘zod.dev’ website.

