Date : 22 - 06 - 2020
Morning Session : 9am – 11 PM
By ~ Sundeep Charan Ramkumar Today

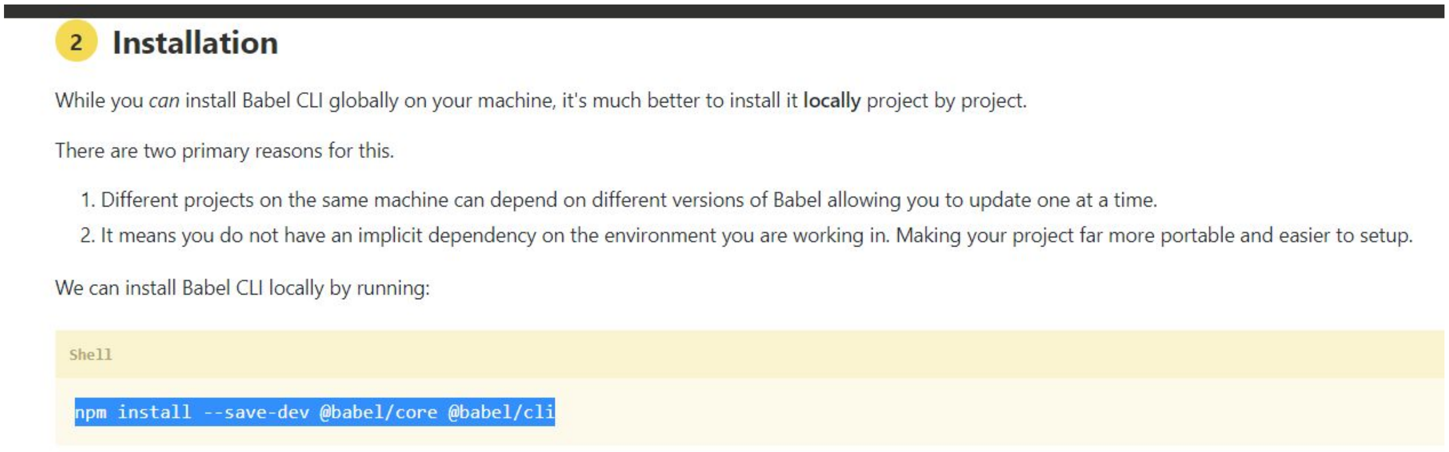
Topics: **ES6 Intro**

ES6 It allows you to write less code and do more. ES6 introduces us to many great features like arrow functions, template strings, class destruction, Modules… and more.

Babel: Babel is a transpiler (translates code in one language to another computer language at the same abstraction level) that can turn our ES6 code into ES5.

https://babeljs.io/setup#installation

Install Babel as Dev Dependency



2 **Installation**

While you *can* install Babel CLI globally on your machine, it's much better to install it **locally** project by project.

There are two primary reasons for this.

1. Different projects on the same machine can depend on different versions of Babel allowing you to update one at a time.
2. It means you do not have an implicit dependency on the environment you are working in. Making your project far more portable and easier to setup.

We can install Babel CLI locally by running:

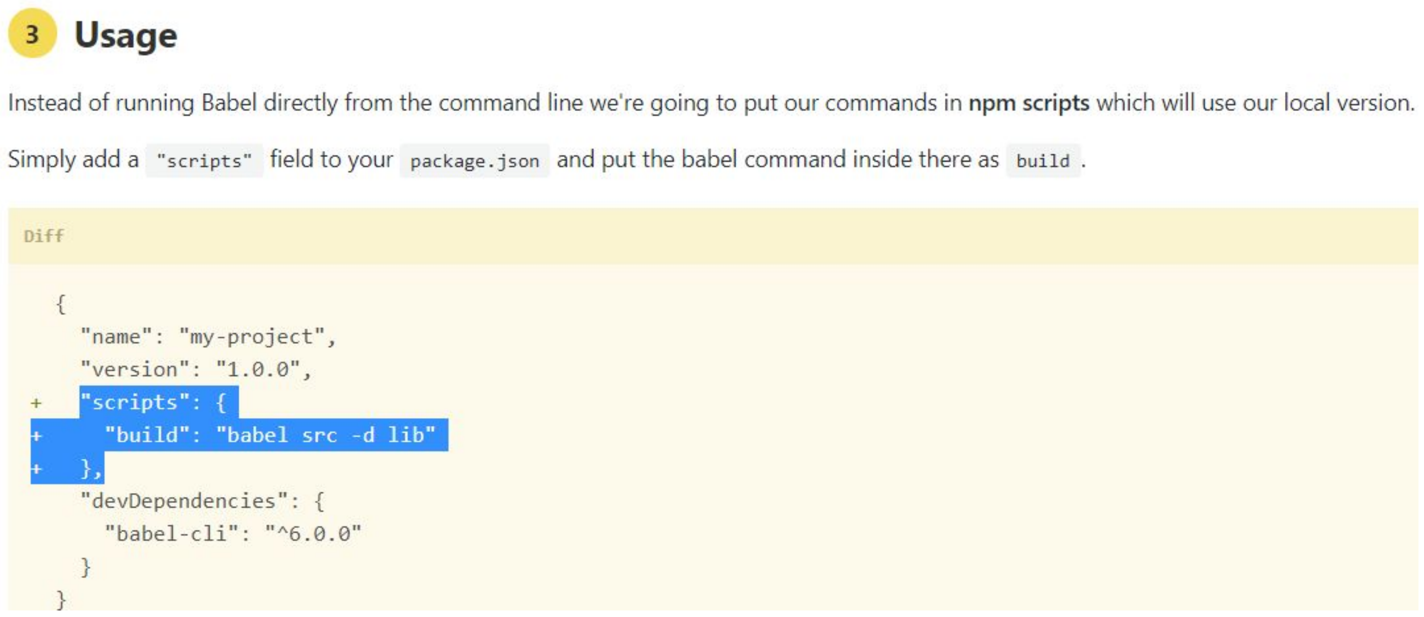
```
Shell
npm install --save-dev @babel/core @babel/cli
```

Open vs code ↲

NPM init ↲

`npm install --save-dev @babel/core @babel/cli` ↲

To code traspile

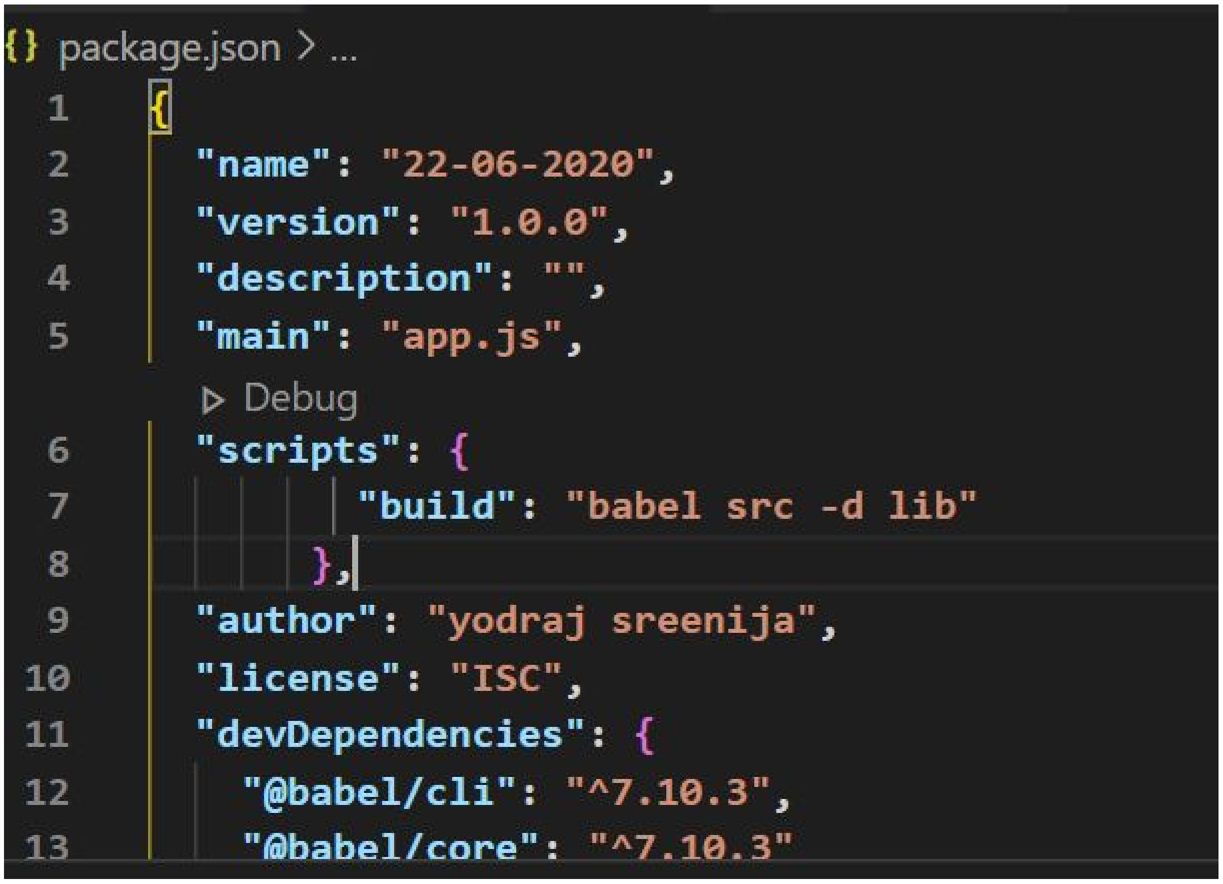


3 Usage

Instead of running Babel directly from the command line we're going to put our commands in **npm scripts** which will use our local version.

Simply add a `"scripts"` field to your `package.json` and put the babel command inside there as `build`.

```diff
  {
    "name": "my-project",
    "version": "1.0.0",
+   "scripts": {
+     "build": "babel src -d lib"
+   },
    "devDependencies": {
      "babel-cli": "^6.0.0"
    }
  }
```

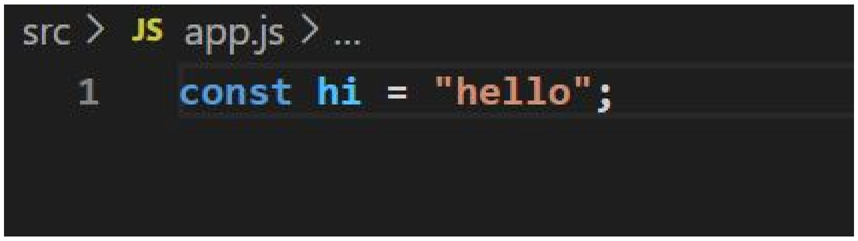


{} package.json > ...

```json
{
  "name": "22-06-2020",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  ▷ Debug
  "scripts": {
        "build": "babel src -d lib"
      },
  "author": "yodraj sreenija",
  "license": "ISC",
  "devDependencies": {
    "@babel/cli": "^7.10.3",
    "@babel/core": "^7.10.3"
```

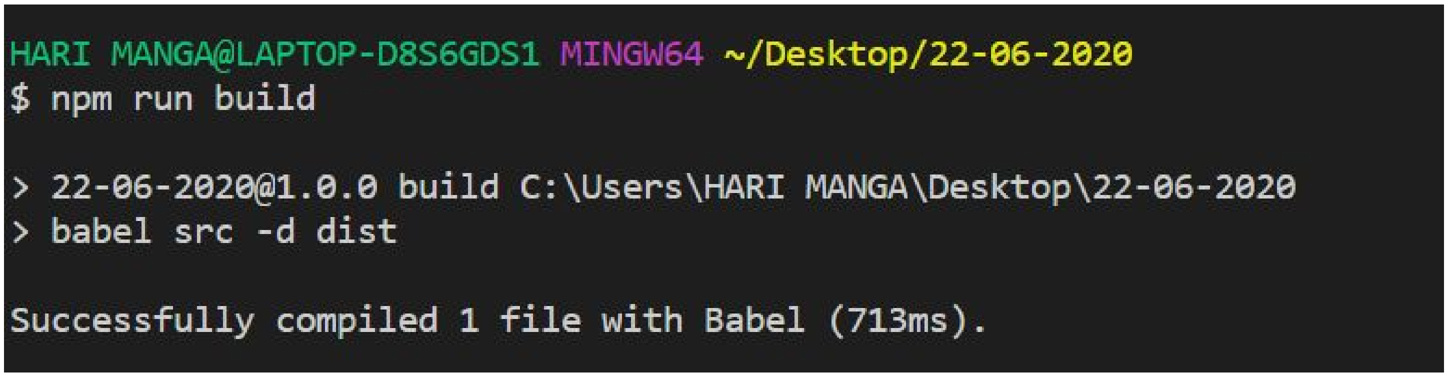
Babel is not global command, so we are using script as inside

Babel = transpiler
Src = source folder
-d = destination
Lib = destination folder



```
src > JS app.js > ...
const hi = "hello";
```

Npm run build ↲



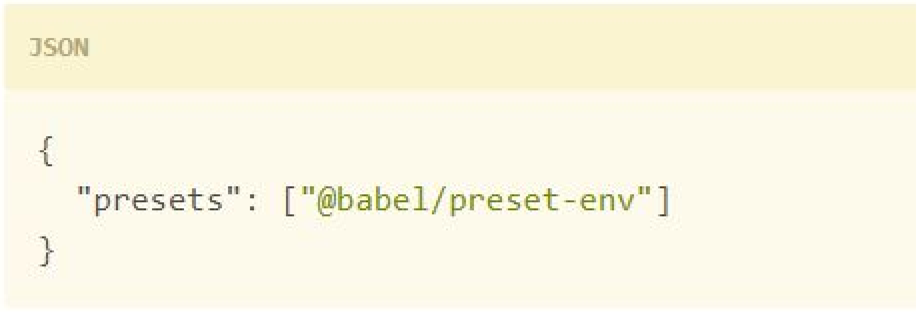
```
HARI MANGA@LAPTOP-D8S6GDS1 MINGW64 ~/Desktop/22-06-2020
$ npm run build

> 22-06-2020@1.0.0 build C:\Users\HARI MANGA\Desktop\22-06-2020
> babel src -d dist

Successfully compiled 1 file with Babel (713ms).
```

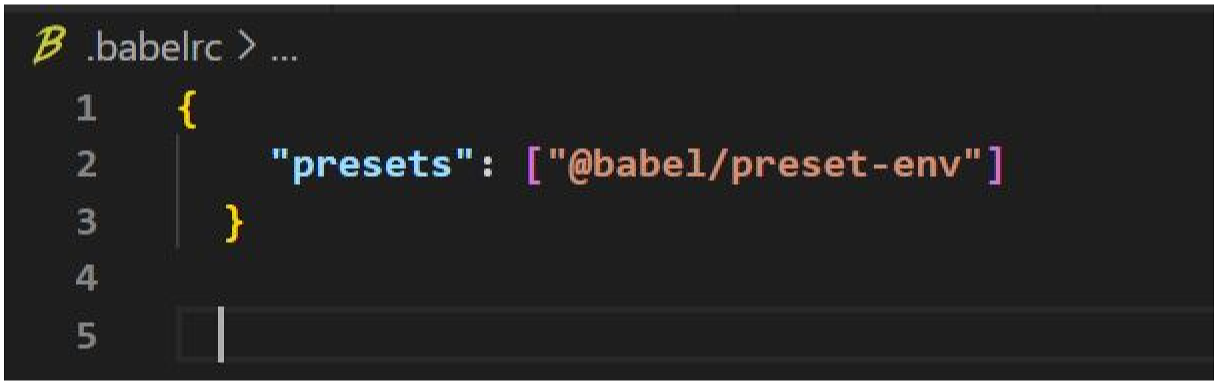
Create a new configuration file **.babelrc**

Copy preset and paste in .babelrc file



```json
JSON

{
  "presets": ["@babel/preset-env"]
}
```



```
.babelrc > ...
{
    "presets": ["@babel/preset-env"]
 }

```

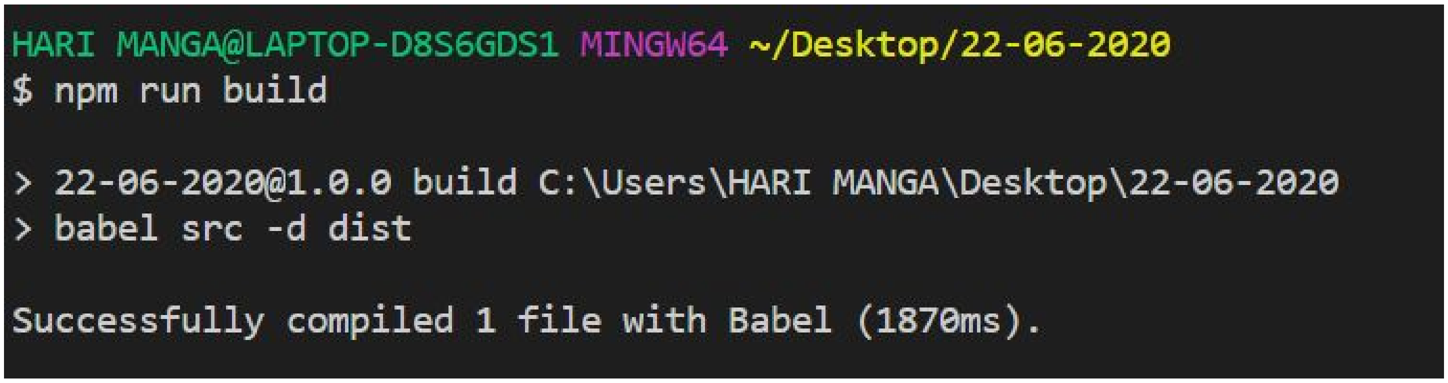
Preset set of configuration bake inside container

https://babeljs.io/docs/en/

https://babeljs.io/docs/en/presets

Npm install @babel/present-env ↲

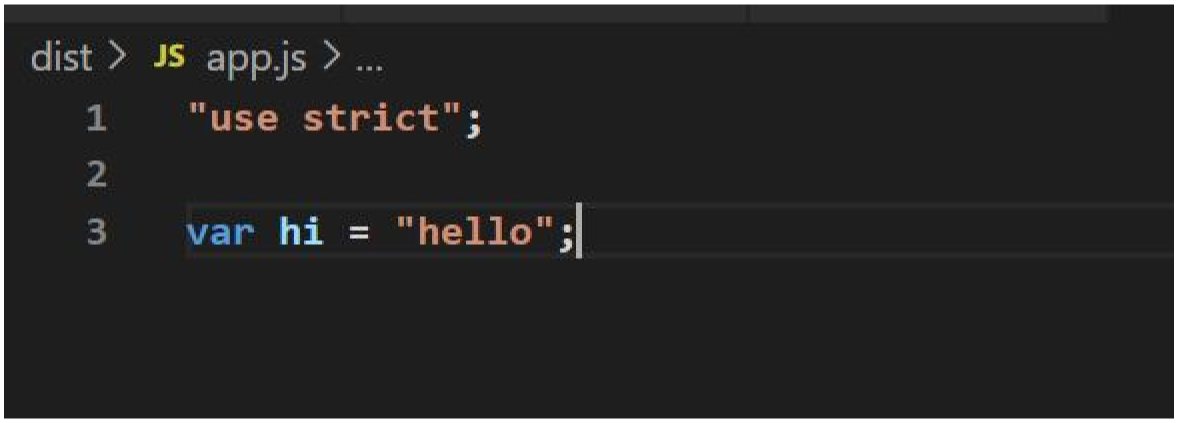
Npm run build ↲



```
HARI MANGA@LAPTOP-D8S6GDS1 MINGW64 ~/Desktop/22-06-2020
$ npm run build

> 22-06-2020@1.0.0 build C:\Users\HARI MANGA\Desktop\22-06-2020
> babel src -d dist

Successfully compiled 1 file with Babel (1870ms).
```



dist > JS app.js > ...

```
"use strict";

var hi = "hello";
```

“Use strict” = Strict mode is implemented to make your program or function follow a strict operating context. It means, the errors which were being ignored by the compiler, will now throw exception messages.

Preset consider as group of plugins