Babel

- Babel is a JavaScript compiler.
- Babel does the same and it is called a transpiler that transpiles the code in the ECMAScript version that we want. It has features like presets and plugins, which configure the ECMA version we need to transpile our code.
- BabelJS is a JavaScript transpiler which transpiles new features into old standards.

What is Babel-Transpiler?

Babel-transpiler converts the syntax of modern JavaScript into a form, which can be easily understood by older browsers. For example, arrow function, const, let classes will be converted to function, var, etc. Here the syntax, i.e., the arrow function is converted to a normal function keeping the functionality same in both the cases.

What is Babel-polyfill?

There are new features added in JavaScript like promises, maps and includes. The features can be used on arrays; the same, when used and transpiled using babel will not get converted. In case the new feature is a method or object, we need to use Babel-polyfill along with transpiling to make it work on older browsers.

Here is the list of ECMAScript features available in JavaScript, which can be transpiled and polyfilled −

- Classes
- Decorators
- Const
- Modules
- Destructing
- Default parameters
- Computed property names
- Object rest/spread
- Async functions
- Arrow functions
- Rest parameters
- Spread
- Template Literals

ECMAScript features that can be polyfilled −

- Promises
- Map
- Set
- Symbol
- Weakmap
- Weakset
- includes

Array.from,

Array.of,Array#find,Array.buffer,

Array#findIndex

Object.assign,Object.entries,Object.values

- Difference Between Transpiler & Polyfills

1.Polyfill in general means we are going to write code which should work in the older browsers

For instance, look at this example. We want to use the function `Number.isNaN` in our code, and we write this piece of code before using it.

To fill the gaps, we add the missing functionality which is called a "`polyfill`".

For example, the `filter()` method was introduced in ES5 and is not supported in some of the old browsers. This method accepts a function and returns an array containing only the values of the original array for which the function returns `true`.

```
const arr = [1, 2, 3, 4, 5, 6];

const filtered = arr.filter((e) => e % 2 === 0); // filter outs the even number
```

```
console.log(filtered);

// [2, 4, 6]
```

The polyfill for the filter can be written as follows:

```
Array.prototype.filter = function (callback) {

 // Store the new array

 const result = [];

 for (let i = 0; i < this.length; i++) {

 // call the callback with the current element, index, and context.

 //if it passes the text then add the element in the new array.

   if (callback(this[i], i, this)) {

     result.push(this[i]);

   }

 }

 //return the array

 return result

}
```

1.Transpiling (transforming + compiling) is the process of converting your newer code into an older code equivalent.

A transpiler would analyze our code and rewrite `height ?? 100` into `(height !== undefined && height !== null) ? height : 100`.

```
// before running the transpiler
height = height ?? 100;

// after running the transpiler
height = (height !== undefined && height !== null)
? height : 100;
```

BabelJS - Environment Setup

To work with BabelJS we need following setup −

- NodeJS
- Npm
- Babel-CLI
- Babel-Preset
- IDE for writing code

BabelJS - CLI

Babel Intro

Babel is a free and open-source JavaScript transcompiler

Does it making use of Plugins

That can be used to run codes in older JavaScript engines.

Not does do static typing

Convert babel to old syntax.(Step to follow the set-up).

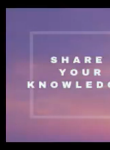
How to use Babel

1)Create a Project
npm –init

2)Install the necessary packages
npm install --save-dev @babel/core @babel/cli @babel/preset-env
npm install @babel/polyfill

3)Create a configuration file
.babelrc

4)Add to the script in package.json
"scripts": {
"babel": "./node_modules/.bin/babel src -d dist"
},

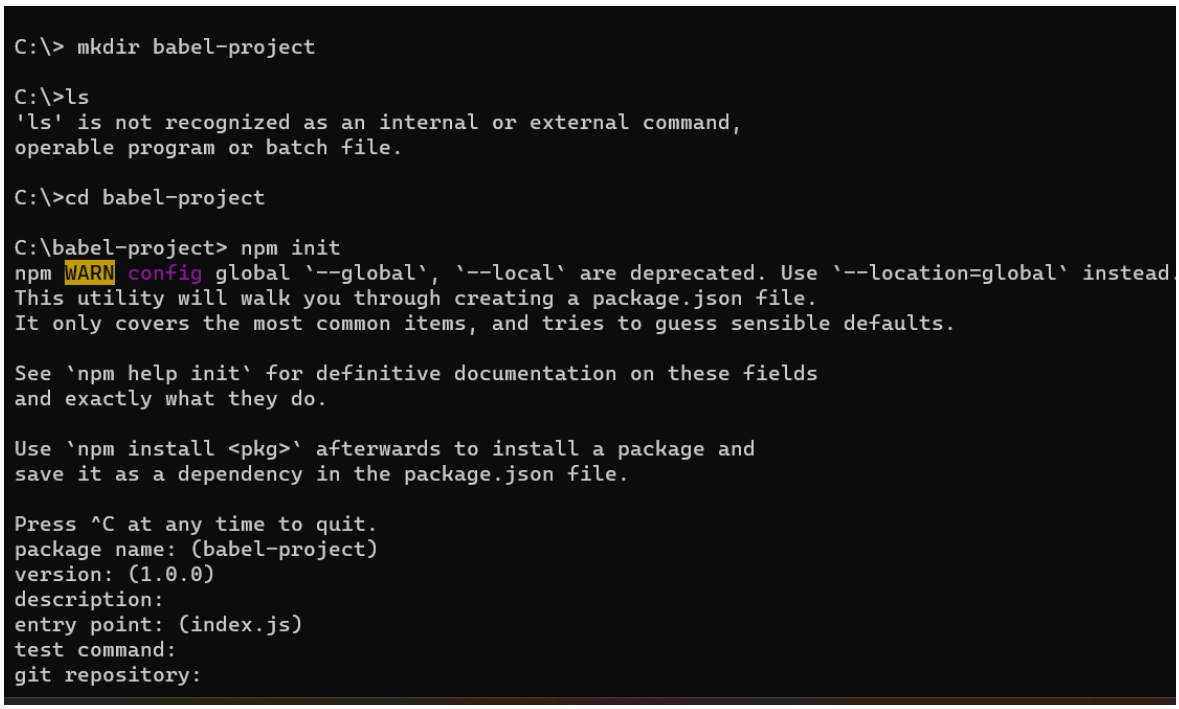


5)Run Babel
npm run babel

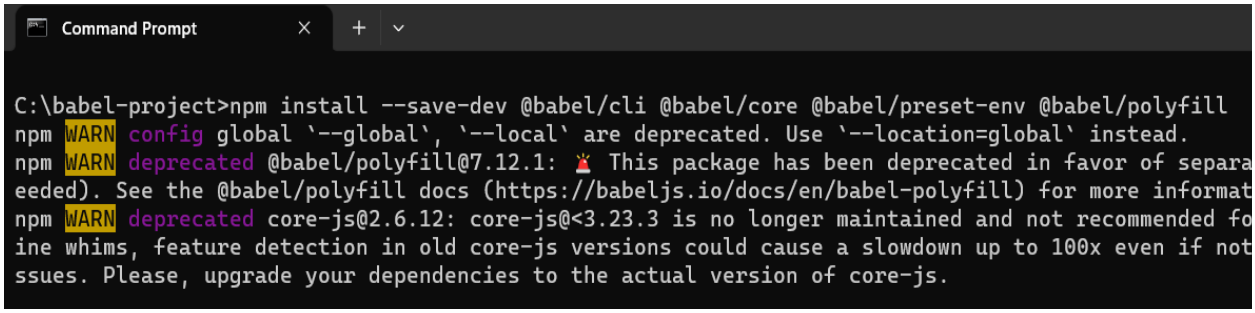
- 1st create a folder in the local machine.

Create a directory wherein you would be working. Here, we have created a directory called *babelproject*.

We have used *npm init* to create the project as shown below –



- Install necessary packages-



```
"devDependencies": {
  "@babel/cli": "^7.21.0",
  "@babel/core": "^7.21.4",
  "@babel/polyfill": "^7.12.1",
  "@babel/preset-env": "^7.21.4"
}
```

@babel/cli

Babel comes with a built-in CLI which can be used to compile files from the command line.

npx babel script.js

npx babel script.js --out-file script-compiled.js

Babel is mainly used to compile JavaScript code, which will have backward compatibility. Now, we will write our code in ES6 -> ES5 or ES7 -> ES5 also ES7->ES6, etc.

@babel/core

Babel Core Compiler.

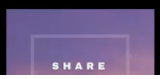
Why @babel/preset-env

Babel by itself cannot do anything. It needs plugins to do Any work.

For transpiling arrow functions it needs this plugin

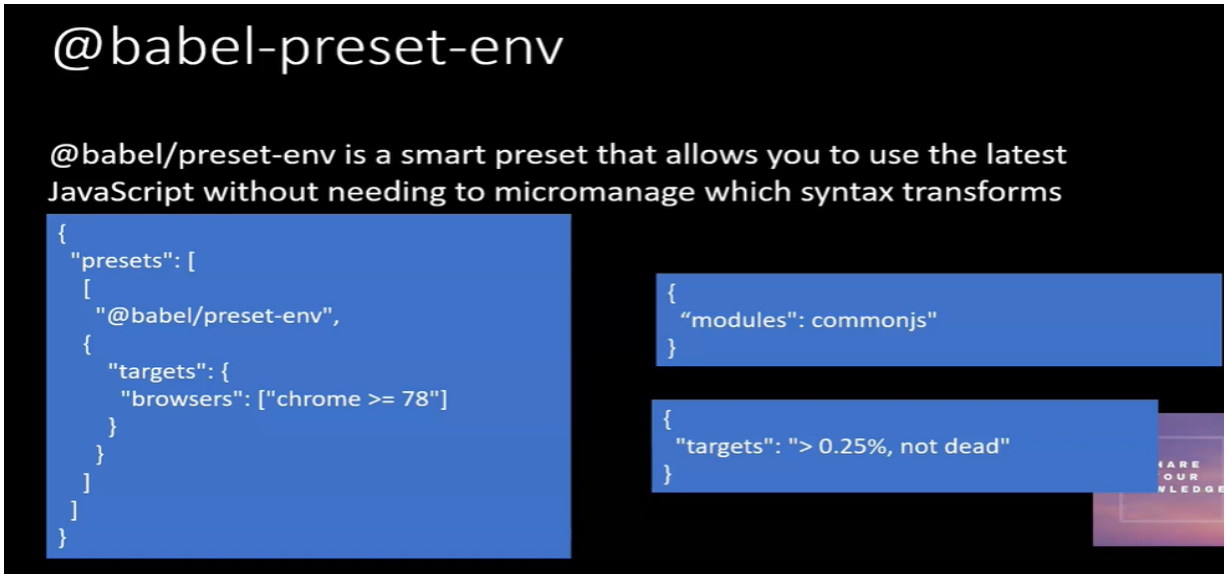
npm install --save-dev @babel/plugin-transform-arrow-functions

Presets - common bundles of plugin

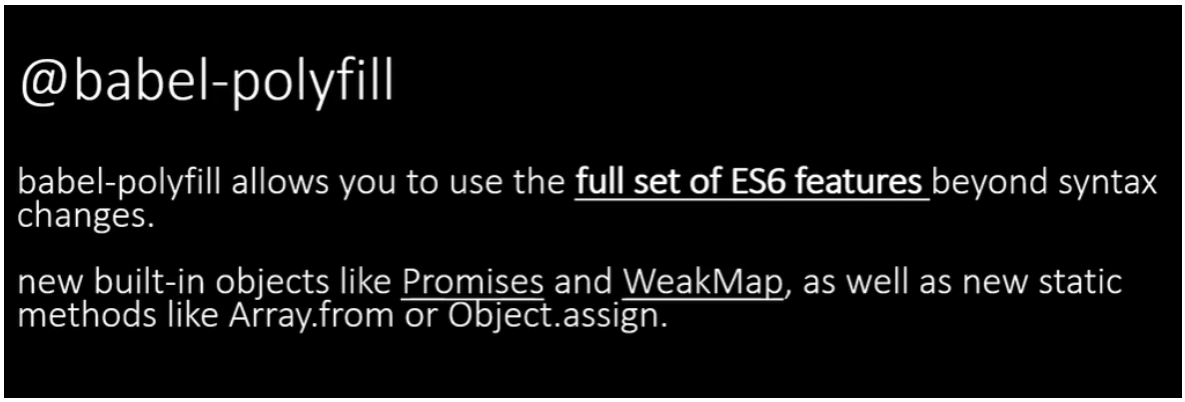


Official Presets

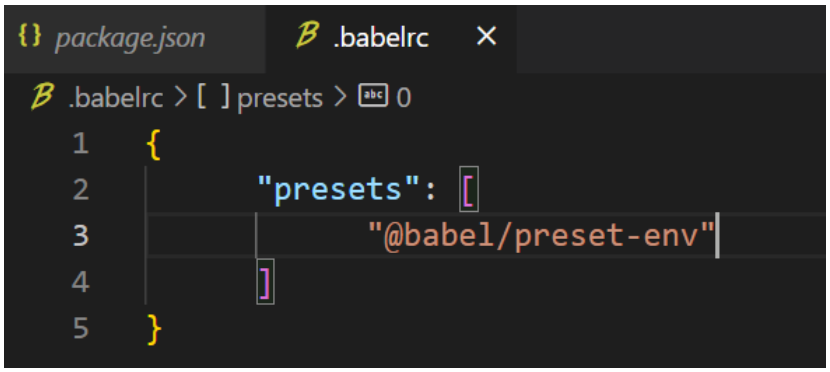
@babel/preset-env
@babel/preset-flow
@babel/preset-react
@babel/preset-typescript



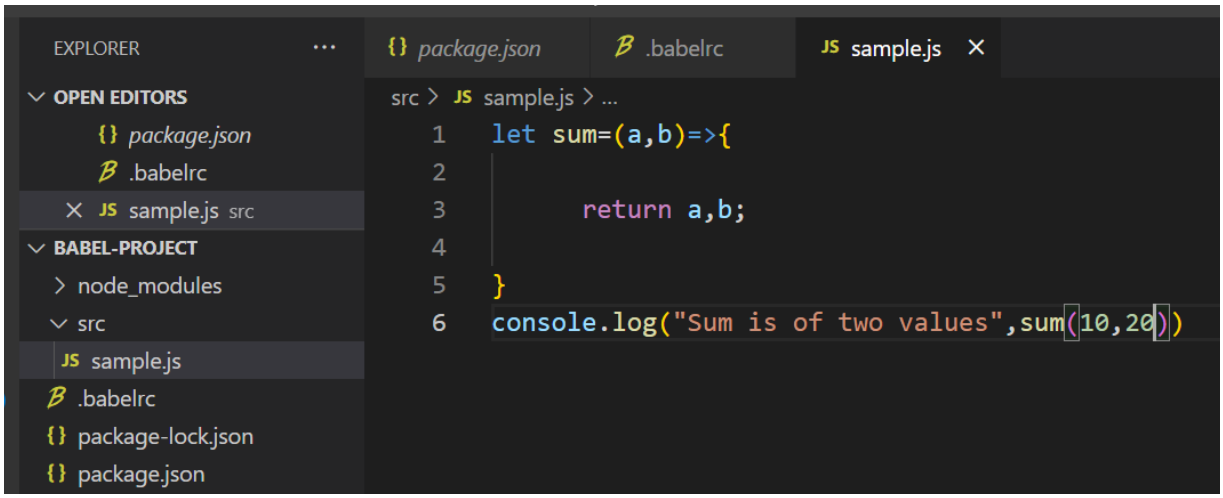
Polyfil,need to be imported into the component.



- Need to create .babelrc file,



To show that babel transpiler.Then create a folder name it is src ,in src create a file.

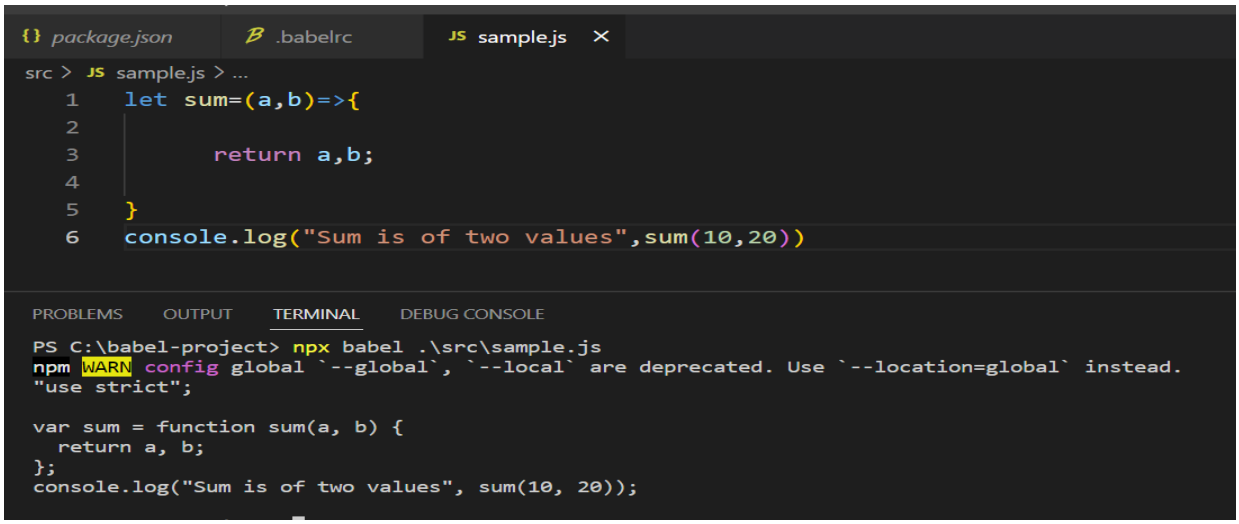


1st run this js file using babel cli->

$npx bable <path of the file>

$npx babel .\src\sample.js

After running cli cmd ,you can see a new arrow function to the old function.

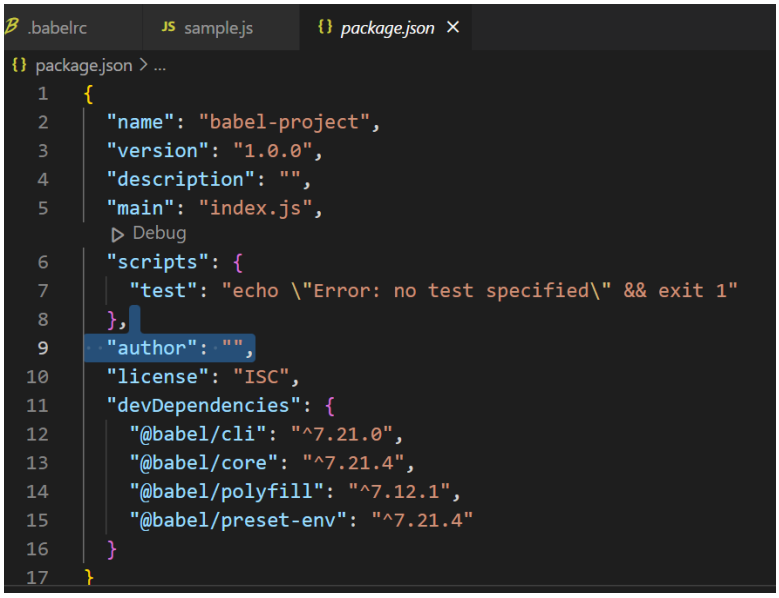


You can also create a file outside,

```
$npx babel .\src\sample.js - -out-file script-compiled.js
```

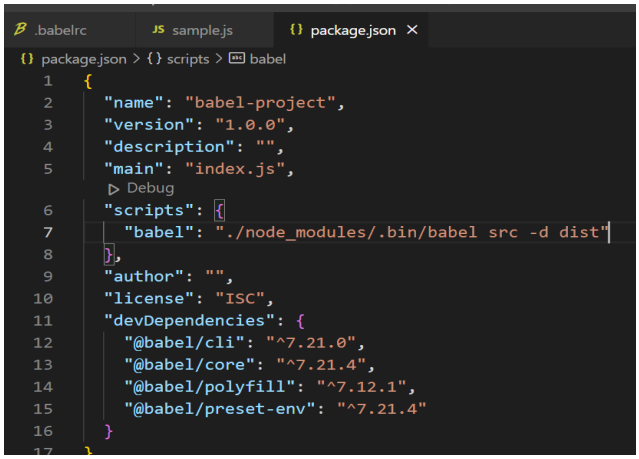
- You can also add script into package.json-

  Previous->

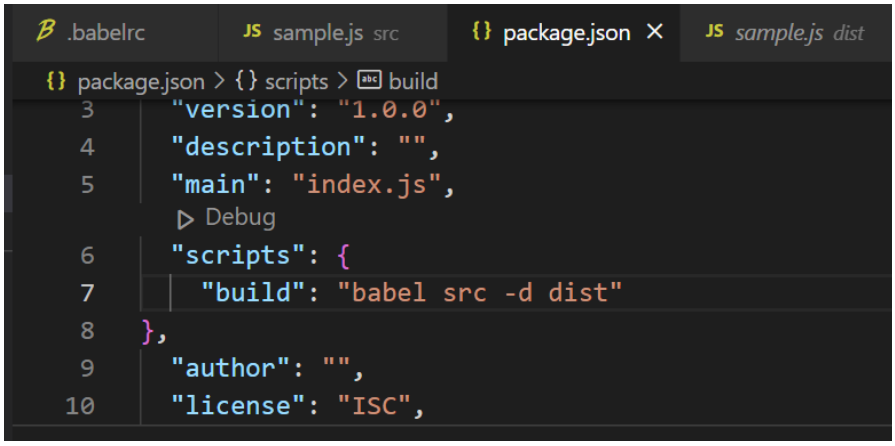


  After adding babel script->

  Older version,you can got an error,



New version,to solve an error,



Add script into package. json->

"babel": "babel src -d dist"

doc-https://babeljs.io/setup#installation

Now this->>

$npm run-script babel

Now,target with chrome version,add optation

Doc- https://babeljs.io/docs/options

ES6 supports chrome >=58.it is es6 syntax.

Check it-> $npm run build

BabelJS - ES6 Code Execution

ES5 is one of the oldest forms of JavaScript and is supported to run on new and old browsers without any issues. In most of the examples in this tutorial, we have transpiled the code to ES5.

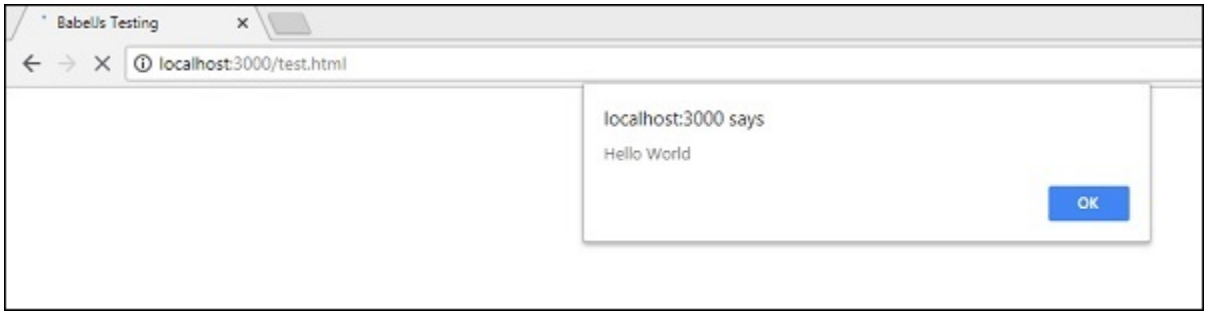
Example> In Src,Create html file,

```
sample1.html U | script-compiled1.js U | script-compiled.js M | index.js U
src > sample1.html > html
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>BabelJs Testing</title>
5      </head>
6      <body>
7          <script type="text/javascript" src="index.js"></script>
8      </body>
9  </html>
```

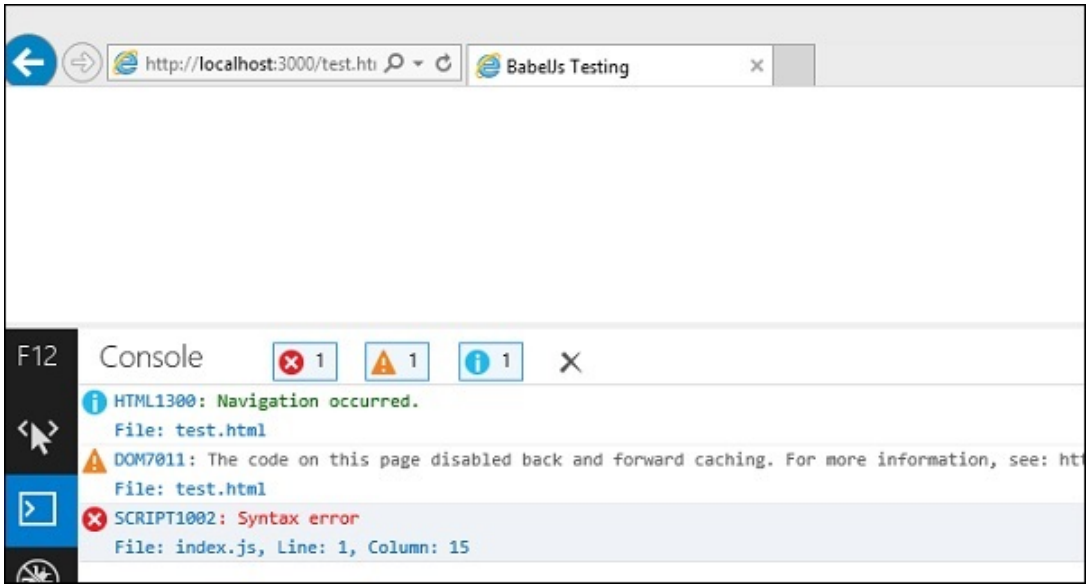
index.js file->

```
sample1.html U | script-compiled1.js U | script-compiled.js M | index.js U
src > index.js > ...
1  var _foo = () => {
2      return "Hello World"
3  };
4
5  alert(_foo());
```

When we run the above html in the Chrome browser, we get the following output –



And when the same HTML is run in Internet Explorer, it generates the following syntax error –



We have used the ES6 Arrow function; the same does not work on all browsers as seen above. To get this working, we have BabelJS to compile the code to ES5 and use it in all browsers.

Will compile the js file to es5 using babeljs and check again in the browsers->

```
$npx babel .\src\index.js --out-file script-compiled1.js
```

Then you can see another file created name is script-compiled1.js.

If you want to create file into dist,

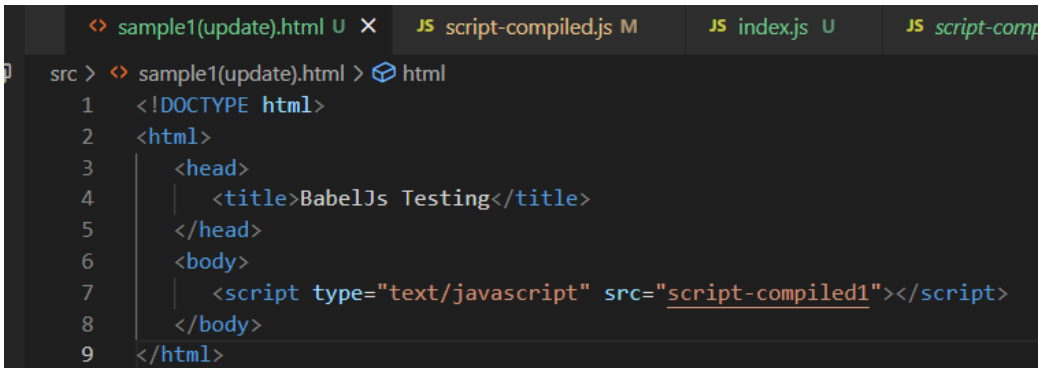
```
$npm run-script build
```

Then ,you can do this code ,in any browser.

Output->

script-compiled.js->

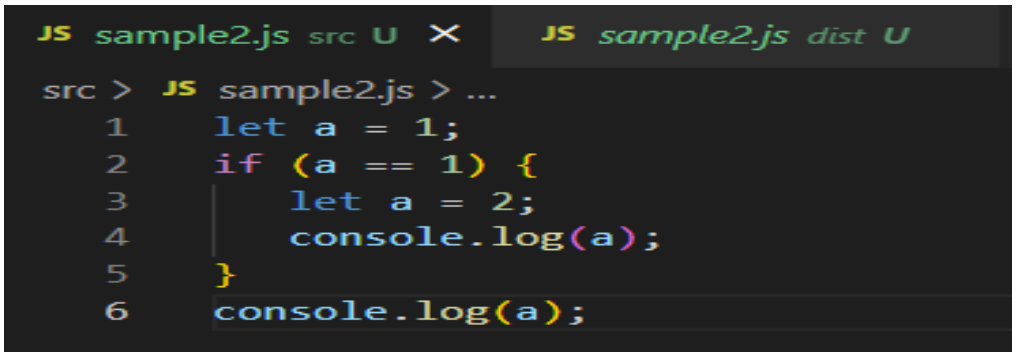
sample1(update).html->



[BabelJS - Transpile ES6 features to ES5]

How to compile the features to ES5 using BabelJS.

sample2.js->



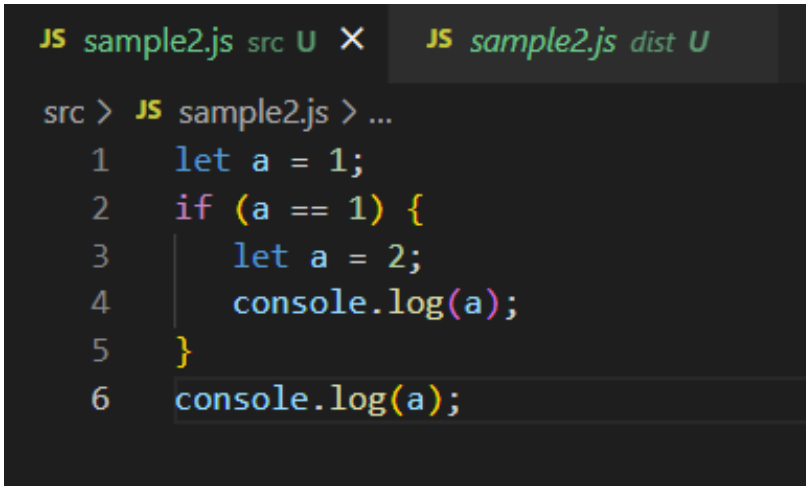
We will now see the code conversion in ES5 using babeljs.

Let us run the following command to convert the code –

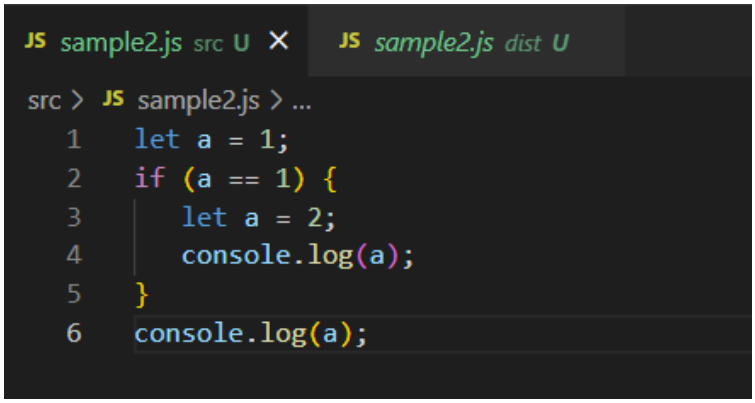
```
$npx babel .\src\sample2.js --out-file script-compiled2.js
```

```
$npx bable <filename/path> --out-file <js file name>
```

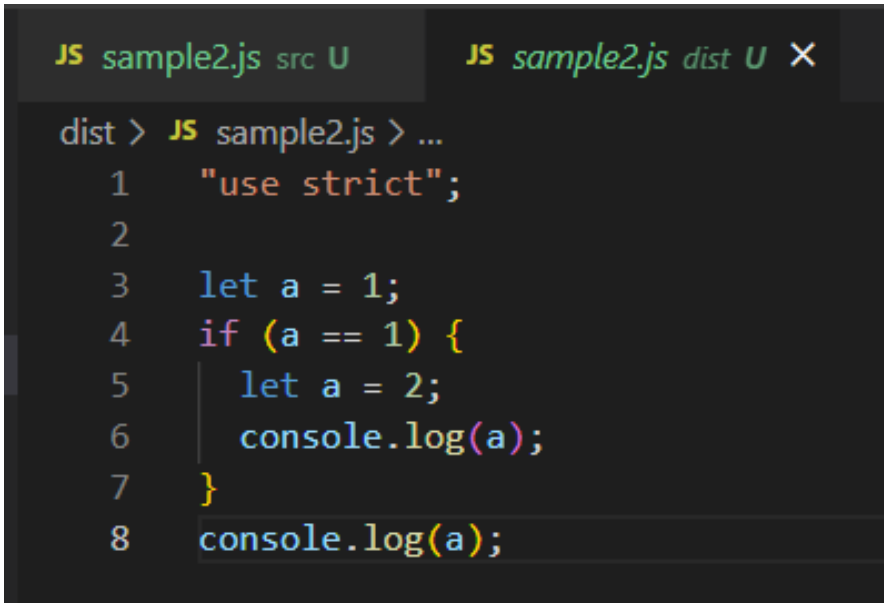
The output from es6 to es5 for the let keyword is as follows –



Let using ES6->



Transpiled using babel to ES5->



Classes->

ES6 comes with the new Classes feature. Classes are similar to the prototype based inheritance available in ES5.The class keyword is used to define the class. Classes are like special functions and have similarities like function expression. It has a constructor, which is called inside the class.

class.js->

```
JS class.js U X   sample1.html   class.html U
src > JS class.js > ...
1   class Person {
2       constructor(fname, lname, age, address) {
3           this.fname = fname;
4           this.lname = lname;
5           this.age = age;
6           this.address = address;
7       }
8
9       get fullname() {
10          return this.fname +"-"+this.lname;
11      }
12  }
13  var a = new Person("Siya", "Kapoor", "15", "Mumbai");
14  var persondet = a.fullname;
15  console.log(persondet)
```

Class.html->

```
JS class.js U   sample1.html   class.html U X
src > class.html > html > body > script
1   <!DOCTYPE html>
2   <html>
3       <head>
4           <title>BabelJs Testing</title>
5       </head>
6       <body>
7           <script type="text/javascript" src="class.js"></script>
8       </body>
9   </html>
```

Command->

```
$npx babel .\src\class.js --out-file script-compiled3.js
```

To import it in the dist folder.

```
$ npm run-script build
```