**06 ES6 Modules**

**1) Modules**:

Good authors divide their books into chapters and sections; good programmers divide their programs into modules.

In JavaScript, the word "modules" refers to small units of independent, reusable code. They are the foundation of many JavaScript design patterns and are critically necessary when building any non-trivial JavaScript-based application.

Using modules gives us number of benefits like

1. Increase the maintainability
2. Increase the reusability of code
3. Abstract code

The popular module formats we have

1. AMD (Asynchronous Module Definition) (use in Browser application)
2. CommonJS (use in Node.js)
3. UMD (Universal Module Definition) (use in Browser/Node.js in ES5)
4. ES6 Modules (JavaScript support modules natively)

Here we will learn only "CommonJS" and "ES6 modules".

**2) CommonJS Modules**:

The CommonJS module specification is the standard used in Node.js for working with modules. Modules are very cool, because they let you encapsulate all sorts of functionality, and expose this functionality to other JavaScript files, as libraries

In CommonJS, modules are loaded synchronously, and processed in the order the JavaScript runtime finds them. This system was born with server-side JavaScript in mind, and is not suitable for the client-side (this is why ES Modules were introduced)

A JavaScript file is a module when it exports one or more of the symbols it defines, being them variables, functions, objects:

**Example**: This example is only working in Node.js

circle.js

*//implementation details*

const \_radious = new WeakMap();

*//public Interface*

class Circle {

constructor(radious) {

\_radious.set(*this*, radious);

}

draw() {

console.log("Circle with radious: " + \_radious.get(*this*));

}

}

module.exports = Circle;

index.js:

const Circle = require("./circle");

const c = new Circle(10);

c.draw();

*//Circle with radious: 10*

**Note**:

Everything that declare in a module is by default private.

**3) ES6 Modules**:

Suppose we have a Circle class and a \_radius weak map in a separate module called "circle" and from this module we want to export only the "Circle" class. We want to keep \_radius weak map private. Because it is the part of implementation details.

By default, everything we declare in a module is by default private unless we explicitly export it using the export keyword.

**Example**:

**circle.js**:

*//this is not accessable when we import circle (circle.js)*

const \_radius = new WeakMap();

*//once we import circle (circle.js) Circle is accessable*

export class Circle {

constructor(radius) {

\_radius.set(*this*, radius);

}

draw() {

console.log("Circle with radious: " + \_radius.get(*this*));

}

}

Now from "index.js" we import the "circle" module.

**Index.js**:

import { Circle } from "./circle";

const c = new Circle(10);

c.draw();

*//Uncaught SyntaxError: Unexpected token*

Now if we run "index.js" we will get an error "Uncaught SyntaxError: Unexpected token" Now for solve this problem we can use webpack. But in here we solve the problem in shortcut way. In our "index.html" change the type of script

<script type="module"src="js/index.js"></script>

Now if we run the code, we will get a different error. " GET http://127.0.0.1:5500/03%20Prototypes/js/circle net::ERR\_ABORTED 404 (Not Found) " Here the name of the file is circle.js but we import circle hence this error occurs. For solve this problem we have to add the complete name "circle.js"

**Example**:

import { Circle } from "./circle.js";

const c = new Circle(10);

c.draw();

*//Circle with radious: 10*

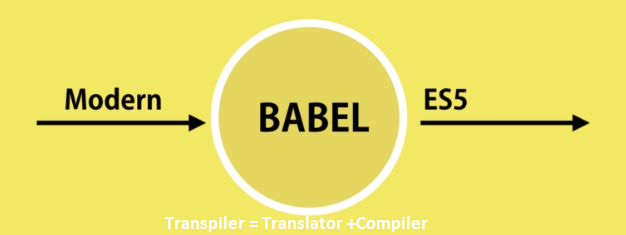
**4) ES6 Tooling**:

Tools are important if we are building browser application. If we are using JavaScript in Node, we don’t be worry about these tools. When we are using modern JavaScript, we need two kinds of tool.

1. Transpiler (Translator +Compiler)
2. Bundler

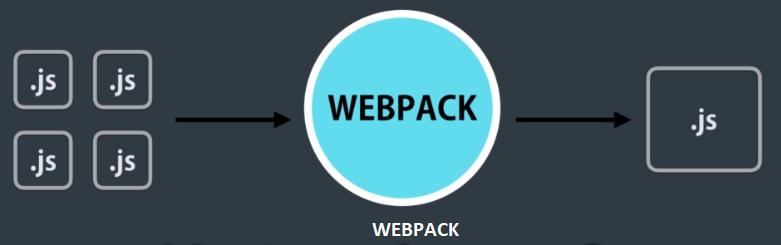
**Transpiler**:

Transpiler is a tool that we give our modern JavaScript code, and it will convert our JavaScript code into code that all browser can understand. For example, BABEL is an example of a very popular transpiler for modern JavaScript.



**Bundler**:

Bundler is responsible for combining all our JavaScript files into a single file, which we call a bundle. The most popular bundler is WEBPACK. We give our code to WEBPACK and it minify our code by removing all whitespaces, comment, spaces and so on.



**5) Babel**:

Create a directory named "babel" and go to the directory. Write the following command

"npm init --yes"

This command creates a package.json file in the directory. This is the identification of our application. Now we have to install babel. We have to install three packages.

1. [babel-cli@6.26.0](mailto:bable-cli@6.26.0) (Babel command line interface like npm)
2. [babel-core@6.26.0](mailto:bable-core@6.26.0) (Core of babel, where all the logic of transpiring code is implemented)
3. [babel-preset-env@1.6.1](mailto:babel-preset-env@1.6.1) (understand all the feature in JavaScript starting from ES6)

**npm i babel-cli@6.26.0 babel-core@6.26.0 babel-preset-env@1.6.1 --save-dev**

Here "—save-dev" means development dependency. They are not going to be the part of our application, they are not going to be deploy to the production, they are purely to the development machine.

Now create a file name "index.js" in the directory and write some code. Now we use babel to convert this to code that all browser can understand.

First go to "package.json" and in the script section delete "test" section and add the new script.

"babel" :"babel --presets env index.js -o build/index.js"

Now in the console run

npm run babel

Now in the build folder index.js file is created and we see

"use strict";

var x = 1;

This code is understood by all browser. Here is no let or const keyword. const is replace with var.

But here is a problem we convert only the index.js file if our program contains more file it creates a problem. To resolve this problem, we should use web pack. With webpack we are going to get all our JavaScript files and put them in a bundle.

Before putting them in a bundle we are going to run each file throw babel.

**6) Webpack**:

In our real-world application, we don’t use babel. We use webpack in real world application. Create a directory named "webpack" and go to the directory. In the directory create a "src" folder and inside src create 2 file named

1. circle.js
2. index.js

**circle.js**:

const \_radius = new WeakMap();

export class Circle {

constructor(radius) {

\_radius.set(*this*, radius);

}

draw() {

console.log("Circle with radius " + \_radius.get(*this*));

}

}

**Index.js**:

import { Circle } from "../src/circle.js";

const c = new Circle(10);

c.draw();

Inside webpack create a html file named "index.html" and link the "index.js" file.

Write the following command

**Index.html**:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Document</title>

<script type="module" src="src/index.js"></script>

</head>

<body>

</body>

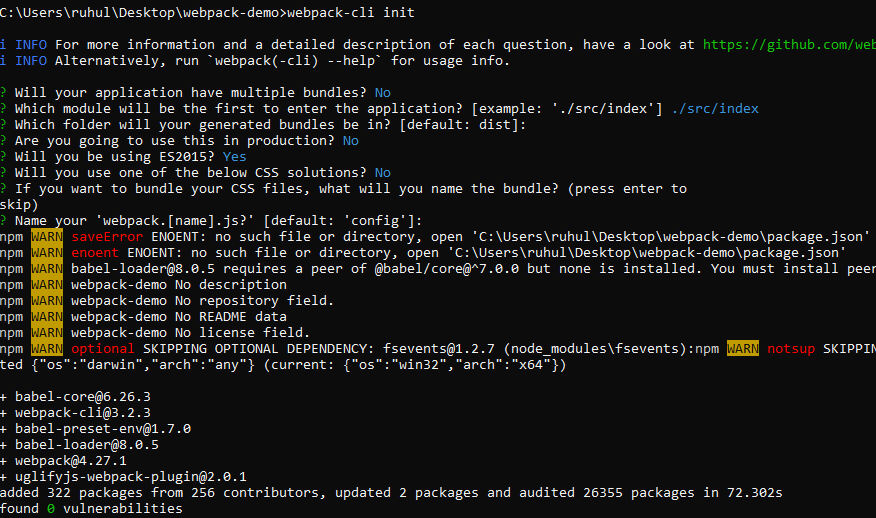
</html>

Now go to our project folder "webpack" and write the following command

npm i webpack-cli@2.0.14

After the installation run the following command

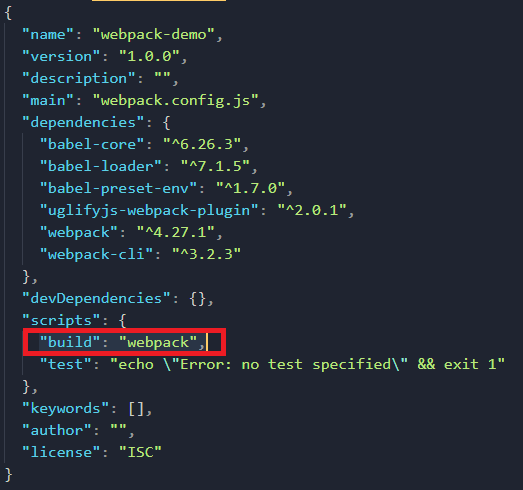
webpack-cli init



Now we have to create "package.json" file. For this run the following command

npm init --yes

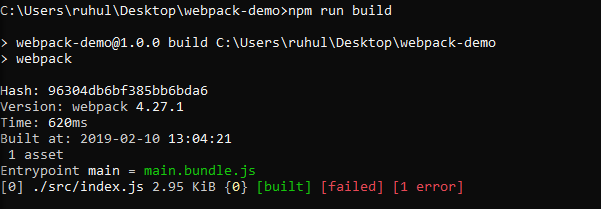
Now "package.json" is create in our directory. Now we have to add "build": "webpack", in this file.



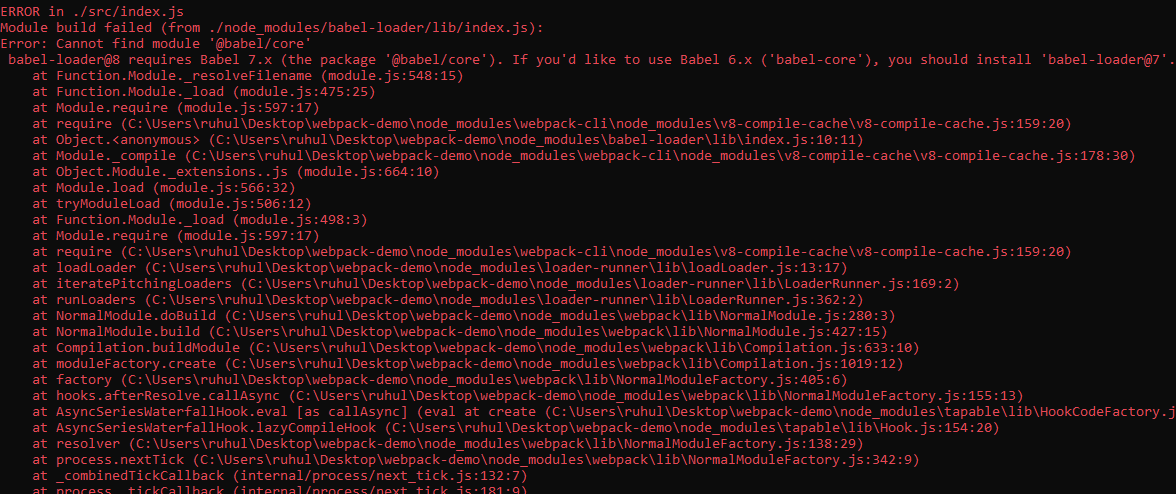
Now our project setup is complete. We have to build our project. For build the project we have to run the following command.

npm run build

But after build command it may gives an error.



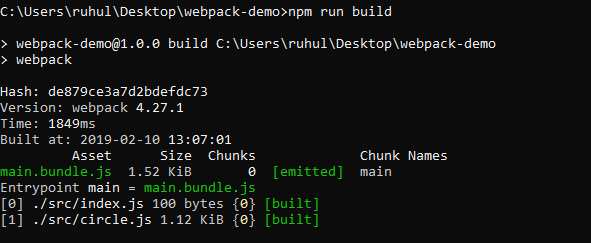
Here is the details of the error.



To solve the error run the following command

npm install babel-loader@7 --save

Now the project builds successfully and create a bundle name "main.bundle.js" file inside dist. folder by using the circle.js and index.js file.



Now go to "index.html" file and do the following change in script tag. Now our module is working and we will see the following output.

*Circle with radius 10*

Now there is a problem in this project. Every time we change our project we have to go in the terminal and build the project. For solve this problem we have to change in the package.json file.

"build": "webpack -w",

Here -w is for watch. Here webpack is watch our project file and every time we create change our file it builds and generate bundle file in dist folder. Now run the build command.

npm run build

Now every time we modify our project webpack rebuild our project automatically.

Now we can change the index.js file as follow.

import { Circle } from "./circle";

06 ES6 Modules