In order to solve the problems discussed with module pattern, we have something new that was introduced.

CommonJS & AMD

★ CommonJS

① We can import different JS files using

```
var module1 = require('module1')
var module2 = require('module2')
```

module1 and module2 are the names of the JavaScript files

② To export a function from a file we can do the following

```
module1.exports = {
    function1 : sayHello,
    :
}
```


To import the same

```
var module1 = require('module1').function1
```

So no ~~impo~~ IIFE and global variable collision issues. Just simple 'import, export :)

We have a small issue here
The require statement to import a module is actually synchronous. It can take a lot of time to load right? Bad user experience on the browser

@codewithsimran

So then we had something called browserify that takes a script and converts it into a bundle (even webpack can do things like that).
This bundle file has all the javascript files combined in one single bundle with all the dependencies (whatever we import) and we actually use this bundled file to run things on the browser and since we already have all the dependencies, it wont slow things down

ES6 Modules

@codewithsimran

Finally we have something natively available

To import something

```
import module1 from 'module1';
```

module1 ↓ what we export

'module1' ↓ name of the JS file

To export something

module1.js

```
export function sayHello(){

}
```

@codewithsimran

```
import {sayHello} from 'module1'
```

Finally let's export multiple things

```
export function func1(){
}
export function func2(){
}
export
```

→

```
import {func1, func2}
from 'module1';
```