Common JS

In Node JS there are two ways to create modules.

1. Common JS
    - Common JS is a module system that is used to create modules.
    - Common JS has pretty high usage.
    - Comes pre - installed in Node JS.
2. ES6 import syntax
    - Recently being promoted a lot.
    - Reasons - Browser support, and better specification, and nice looking syntax as well.
    - Supports a bunch of new features.

We need to make use of the work ***require***.

Require is a function that is used to load modules / files inside your system.

E.g:-

```
touch util.js
```

Then will create a function in the *util.js* ->

We have created a function in *util.js* named addNumbers which will return the addition of two numbers.

```
function addNumbers(a, b) {
    return a + b;
}
```

To export something from Common JS we use ->

- `module.exports.<something>`. Here we attached **something** to module.exports
- Another way

```
module.exports = {
    createFile,
    addTwoNumbers,
};
//  This will export 2 functions createFile and addTwoNumbers to a
different file.
```

For exporting addNumbers function, we use the command ->

```
module.exports.addNumbers = addNumbers
```

Import A File Or Functions Of A File In Another File

- To import a file present in the directory we use the syntax.

```
const util = require('./util');
```

For this type of import statement we can import the functions exported by util by ->

```
// Call ->
console.log(util.addNumbers(3, 7));
```

This will give us the output 10.

- 2nd way to import

```
const { createFile, addTwoNumbers } = require('./util');
```

For this type of import we can directly use the name of the functions to import them.

E.g:-

```
// Call ->
console.log(addTwoNumbers(1, 2));
createFile('./Vid - 2/test.txt', 'Hello World!');
```

Async and Sync Difference

1. Asynchronous Function
    - If we have a written the code above to create a file, and then try to read it afterwards, we might fail.
    - Better for writing production ready code.
    - Consumes less resources. Better Performance.
    - For a request Node can work on other things, while the OS is waiting for the network requests.
2. Synchronous Function
    - Blocking the program forward until the line is compiled successfully.
    - Not good for writing production code.
    - Consumes more resources.
    - For a requests, both Node and OS will wait for the request to be completed, only then the next line will get executed.

FS Module

It is a module which allows us to access and interact with the file system.

It allows to create, remove, edit, append to files.

- Write in Directory - **fs.writeFile**

```
// 1st type
fs.writeFile('location of the file', 'data to be inputted in the file');
// 2nd type
fs.writeFileSync('location of the file', 'data to be inputted in the
file');
// For the location of the file we need to input perfect location where
the file is to be placed.
```

- Read Directory - **fs.readDir**

```
console.log(fs.readDirSync('.'));
// This will read the current directory and provide us with all the files
present in the folder.
```

- Read File - **fs.readFileSync**

```
console.log(fs.readFileSync('index.js'));
//  This will give a output buffer which contains the content of the file.
console.log(fs.readFileSync('index.js', 'utf8'));
// This will give the output as text since our text is encoded in UTF8 it
will decode it and give us the output.
```

ECMAScript

.mjs - Module JavaScript File