MERN Stack

In [1]:

```
from IPython.display import Image
Image("E:/code/frontend/img/node1.png")
```

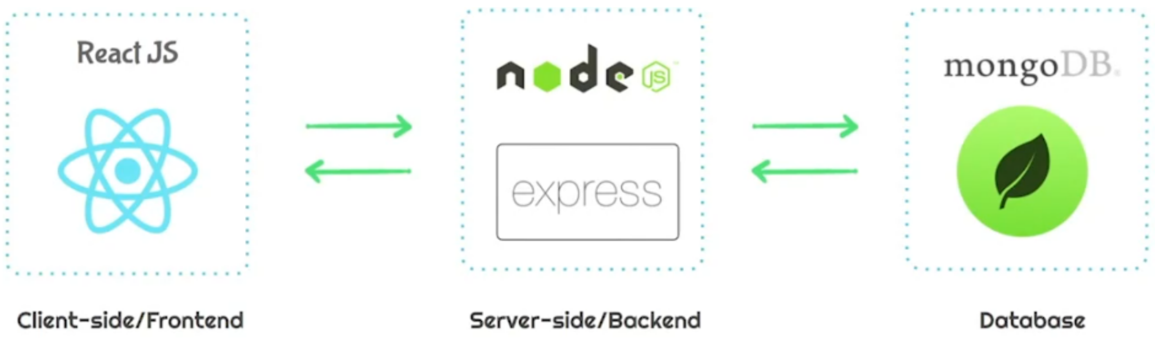
Out[1]:

MERN Stack Application Flow

In [2]:

```
from IPython.display import Image
Image("E:/code/frontend/img/node2.png")
```

Out[2]:



companies using MERN Stack

- facebook
- walmart
- instagram
- oculus
- skype
- Tesla
- pinterest

why mern stack

- great community
- fullstack
- fast learning curve
- open source

IDE(Integrated development Environment)

- it is a software for building applications that makes programming easier
  - VS Code
  - Intellij IDEA
  - Pycharm
  - Atom
  - Eclipse
  - THEIA

Topic 2: Node js

- module exports
- Node Package Manager
- Http Server with Express
- Writing Curd Api's
- Integrating with Databases
- Authentication

Concepts in Focus

- MERN Stack
- Node JS
- Running JavaScript Using Node JS
  - Node REPL (Read-Eval-Print-Loop)
  - Node CLI
- Module
  - Common JS Module Exports
  - Modern JS Module Exports

JavaScript

- In front end , java script runs on the browser
- Browser recieves the JS code and executes the code to update the web applications dynamically

MERN Stack

- MERN stands for MongoDB, Express JS, React JS and Node JS.
- It is a JavaScript Stack that is used for easier & faster deployment of full-stack web applications.

Node JS

- Node JS is a JavaScript environment that executes JavaScript code outside a web browser.

Why Node JS?

- Cross-Platform (Windows, Linux, Mac OS X, etc.)
- Huge number of third-party packages
- Open Source
- Massive Community

How to develop the backend using javascript?

Running JavaScript Using Node JS

- We can run JavaScript using Node JS in 2 ways.
  - Node REPL (Similar to browser console)
  - Node CLI

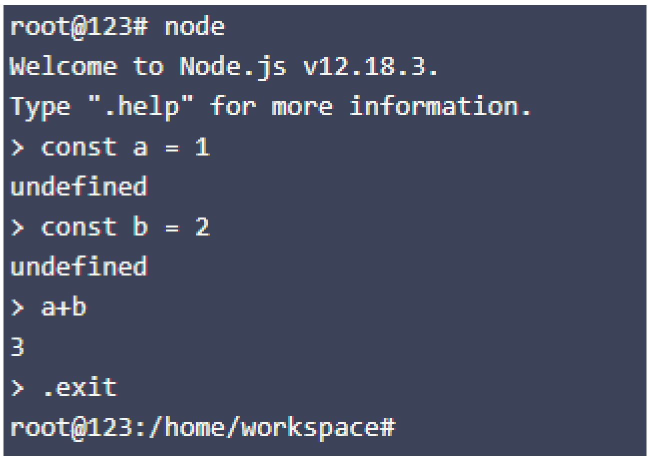
3.1 Node REPL (Read-Eval-Print-Loop)

- The Node REPL is an interactive shell that process Node JS Expressions
- Type node in the terminal and press Enter
- Type .exit and press Enter to exit from the Node REPL.

In [3]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/node5.png")
```

Out[3]:



```
root@123# node
Welcome to Node.js v12.18.3.
Type ".help" for more information.
> const a = 1
undefined
> const b = 2
undefined
> a+b
3
> .exit
root@123:/home/workspace#
```

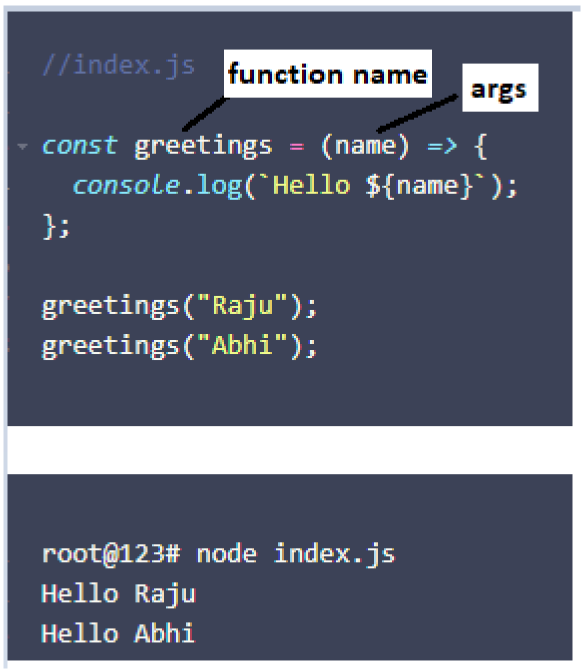
3.2 Node CLI(Command Line Interface)

- We can write JavaScript to a file and can run using Node CLI.

In [4]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/node6.png")
```

Out[4]:



```javascript
//index.js

const greetings = (name) => {
  console.log(`Hello ${name}`);
};

greetings("Raju");
greetings("Abhi");
```

```
root@123# node index.js
Hello Raju
Hello Abhi
```

4. Module

In Node JS, each JavaScript file is treated as a separate module. These are known as the Common JS/Node JS Modules.

To access one module from another module, we have Module Exports.

- Common JS Module Exports
  - Default Exports
  - Named Exports
- Modern JS Module Exports
  - Default Exports
  - Named Exports

Can we access one module from another module?

- module.exports is a special object included in every JavaScript file in the Node Js Application by default

Common JS Module Exports

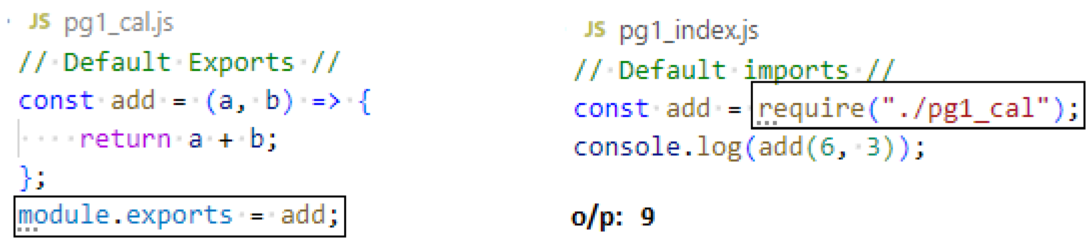
Default Exports

- Exporting Module(Calculator) : The module.exports is a special object included in every JavaScript file in the Node JS application by default.
- Importing Module(Calculator) : To import a module which is the local file, use the require() function with the relative path of the module (file name).

In [5]:

```
from IPython.display import Image
Image("E:/code/frontend/img/node7.png")
```

Out[5]:



pg1_cal.js

```
// Default Exports //
const add = (a, b) => {
    return a + b;
};
module.exports = add;
```

pg1_index.js

```
// Default imports //
const add = require("./pg1_cal");
console.log(add(6, 3));
```

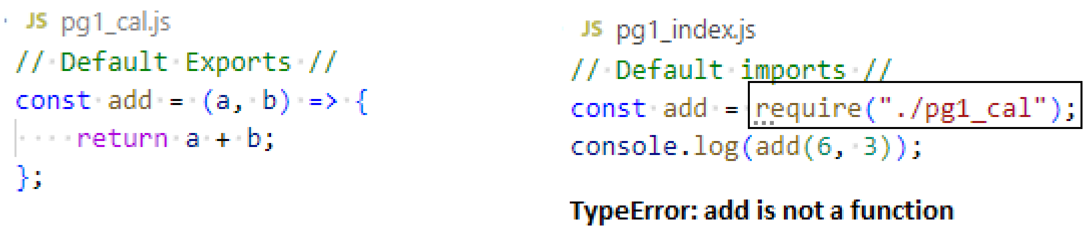
o/p: 9

importing without Exporting module

In [6]:

```
from IPython.display import Image
Image("E:/code/frontend/img/node8.png")
```

Out[6]:



```
pg1_cal.js
// Default Exports //
const add = (a, b) => {
    return a + b;
};
```

```
pg1_index.js
// Default imports //
const add = require("./pg1_cal");
console.log(add(6, 3));

TypeError: add is not a function
```

How can we export multiple functions?

Named Exports

We can have multiple named exports per module.

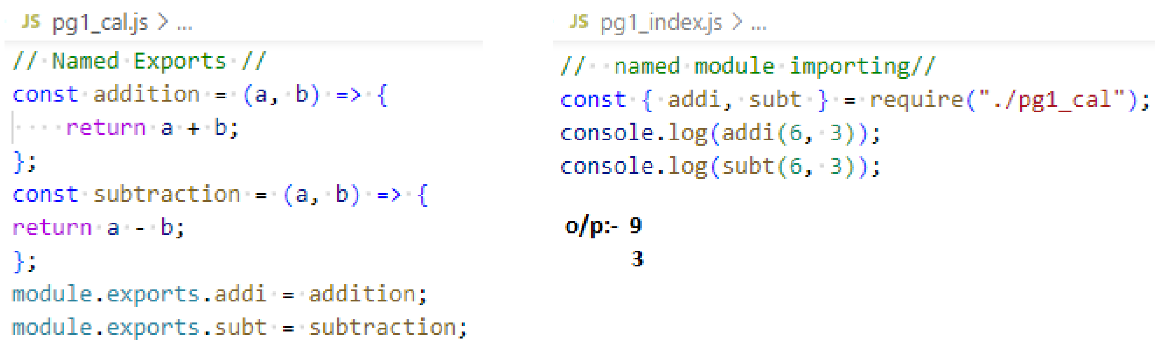
Exporting module

- instead of using module.exports.add we use exports.add

In [7]:

```
from IPython.display import Image
Image("E:/code/frontend/img/node9.png")
```

Out[7]:



```
pg1_cal.js > ...
// Named Exports //
const addition = (a, b) => {
    return a + b;
};
const subtraction = (a, b) => {
return a - b;
};
module.exports.addi = addition;
module.exports.subt = subtraction;
```

```
pg1_index.js > ...
//  named module importing//
const { addi, subt } = require("./pg1_cal");
console.log(addi(6, 3));
console.log(subt(6, 3));

o/p:- 9
      3
```

Modern JS Module Exports

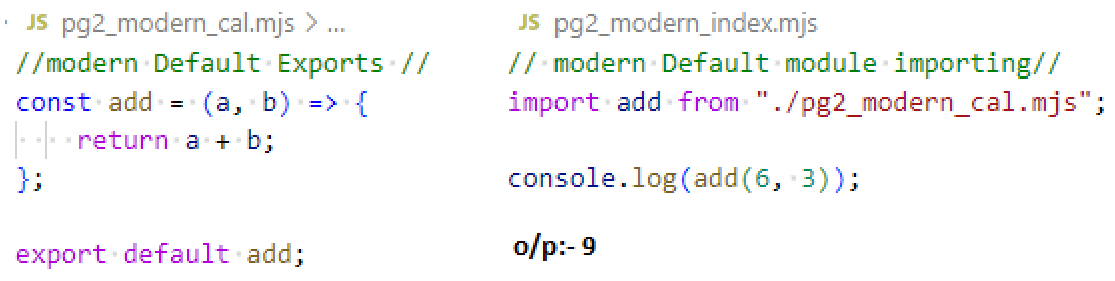
- Modern JS Modules are known as ES6 Modules(ECMA Script Modules).
- Every ES6 module in Node Js should be named with .mjs extension
- The export and import keywords are introduced for exporting and importing one or more members in a module.

Default Exports

In [8]:

```
from IPython.display import Image
Image("E:/code/frontend/img/node10.png")
```

Out[8]:



pg2_modern_cal.mjs > ...

```
//modern Default Exports //
const add = (a, b) => {
    return a + b;
};

export default add;
```

pg2_modern_index.mjs

```
// modern Default module importing//
import add from "./pg2_modern_cal.mjs";

console.log(add(6, 3));
```

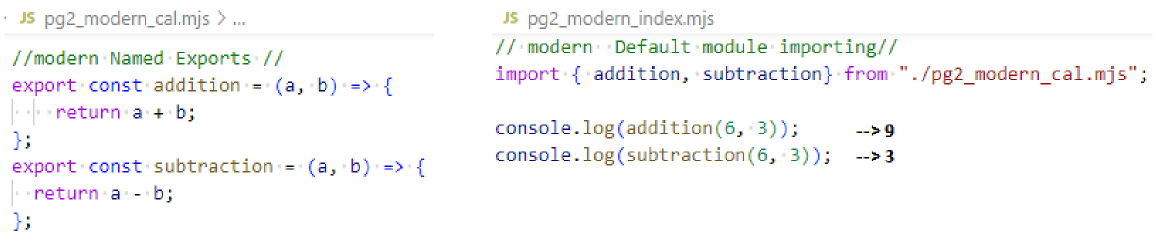
o/p:- 9

Named Exports

In [9]:

```
from IPython.display import Image
Image("E:/code/frontend/img/node11.png")
```

Out[9]:



pg2_modern_cal.mjs > ...

```
//modern Named Exports //
export const addition = (a, b) => {
    return a + b;
};
export const subtraction = (a, b) => {
  return a - b;
};
```

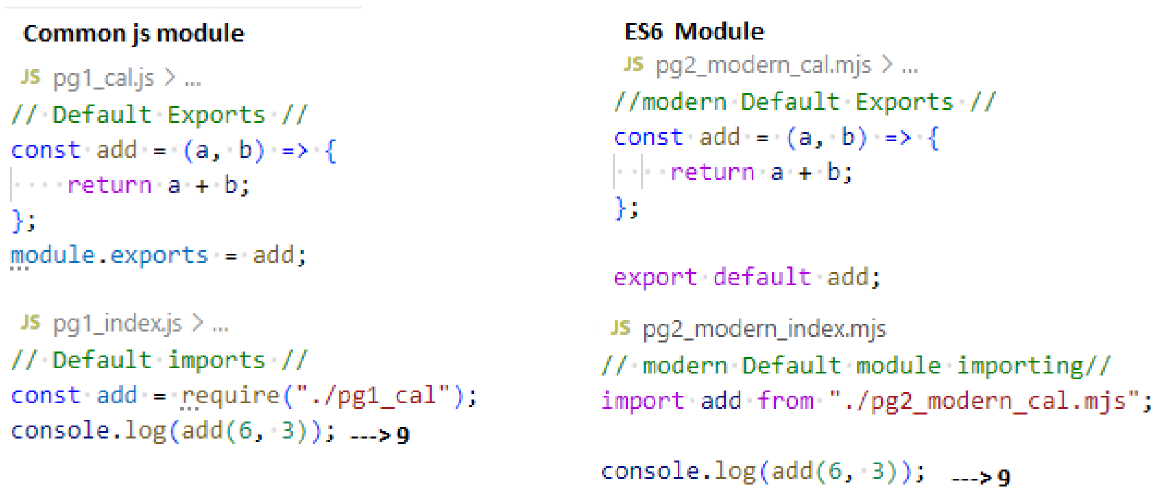
pg2_modern_index.mjs

```
// modern  Default module importing//
import { addition, subtraction} from "./pg2_modern_cal.mjs";

console.log(addition(6, 3));      -->9
console.log(subtraction(6, 3));   -->3
```

In [10]:

```
from IPython.display import Image
Image("E:/code/frontend/img/node12.png")
```

Out[10]:



Common js module

pg1_cal.js > ...

```
// Default Exports //
const add = (a, b) => {
   return a + b;
};
module.exports = add;
```

pg1_index.js > ...

```
// Default imports //
const add = require("./pg1_cal");
console.log(add(6, 3)); --->9
```

ES6 Module

pg2_modern_cal.mjs > ...

```
//modern Default Exports //
const add = (a, b) => {
    return a + b;
};

export default add;
```

pg2_modern_index.mjs

```
// modern Default module importing//
import add from "./pg2_modern_cal.mjs";

console.log(add(6, 3));  --->9
```

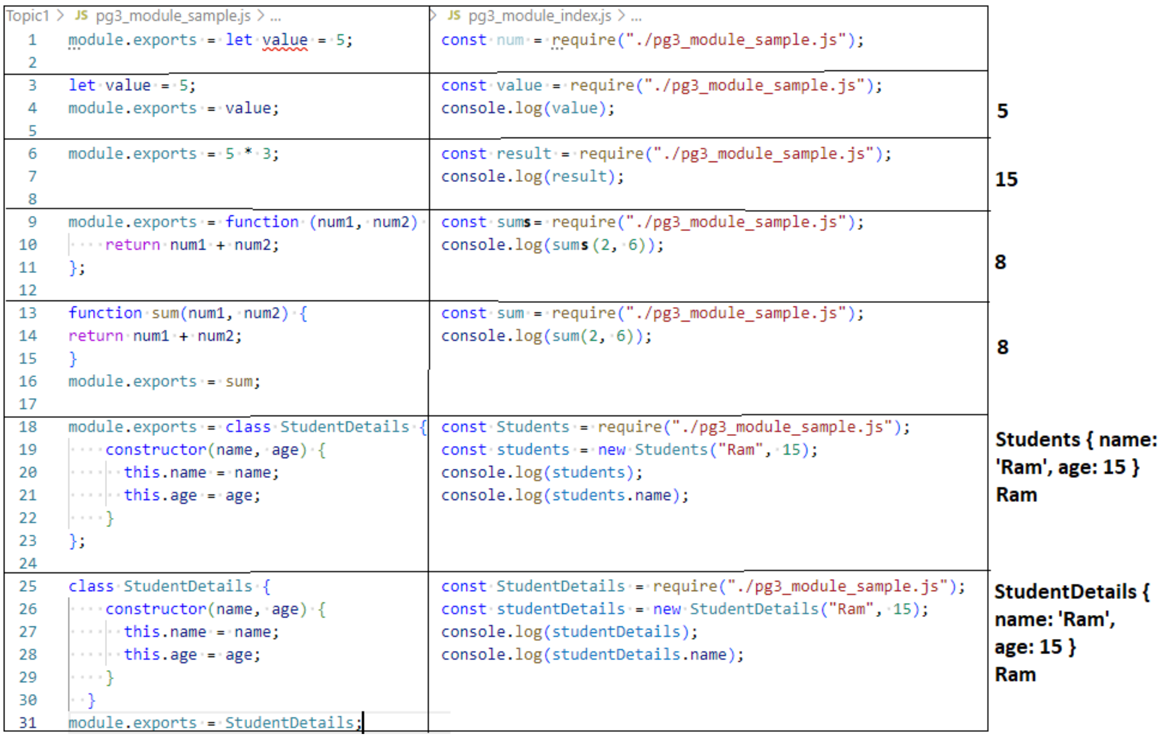
Default Exports : With Default Exports, we can import modules with any name.

- Exporting a variable while defining : We cannot export boolean, number, string, null, undefined, objects, and arrays while defining.
- Exporting a variable after defining : We can export boolean, number, string, null, undefined, objects, and arrays after defining.
- Exporting a value or an expression : We can export a value or an expression directly.
- Exporting a function while defining : We can export a function while defining.
- Exporting a function after defining : We can export a function after defining.
- Exporting a class while defining : We can export a class while defining.
- Exporting a class after defining : We can export a class after defining.

In [11]:

```
from IPython.display import Image
Image("E:/code/frontend/img/node13.png")
```

Out[11]:



Topic1 > pg3_module_sample.js > ...	pg3_module_index.js > ...	
`module.exports = let value = 5;`	`const num = require("./pg3_module_sample.js");`	

`let value = 5;`
`module.exports = value;`
	`const value = require("./pg3_module_sample.js");`
	`console.log(value);`
		5

`module.exports = 5 * 3;`	`const result = require("./pg3_module_sample.js");`
	`console.log(result);`
		15

`module.exports = function (num1, num2) {`
`    return num1 + num2;`
`};`
	`const sums= require("./pg3_module_sample.js");`
	`console.log(sums (2, 6));`
		8

`function sum(num1, num2) {`
`return num1 + num2;`
`}`
`module.exports = sum;`
	`const sum = require("./pg3_module_sample.js");`
	`console.log(sum(2, 6));`
		8

`module.exports = class StudentDetails {`
`    constructor(name, age) {`
`        this.name = name;`
`        this.age = age;`
`    }`
`};`
	`const Students = require("./pg3_module_sample.js");`
	`const students = new Students("Ram", 15);`
	`console.log(students);`
	`console.log(students.name);`
		Students { name: 'Ram', age: 15 }
		Ram

`class StudentDetails {`
`    constructor(name, age) {`
`        this.name = name;`
`        this.age = age;`
`    }`
`  }`
`module.exports = StudentDetails;`
	`const StudentDetails = require("./pg3_module_sample.js");`
	`const studentDetails = new StudentDetails("Ram", 15);`
	`console.log(studentDetails);`
	`console.log(studentDetails.name);`
		StudentDetails { name: 'Ram', age: 15 }
		Ram

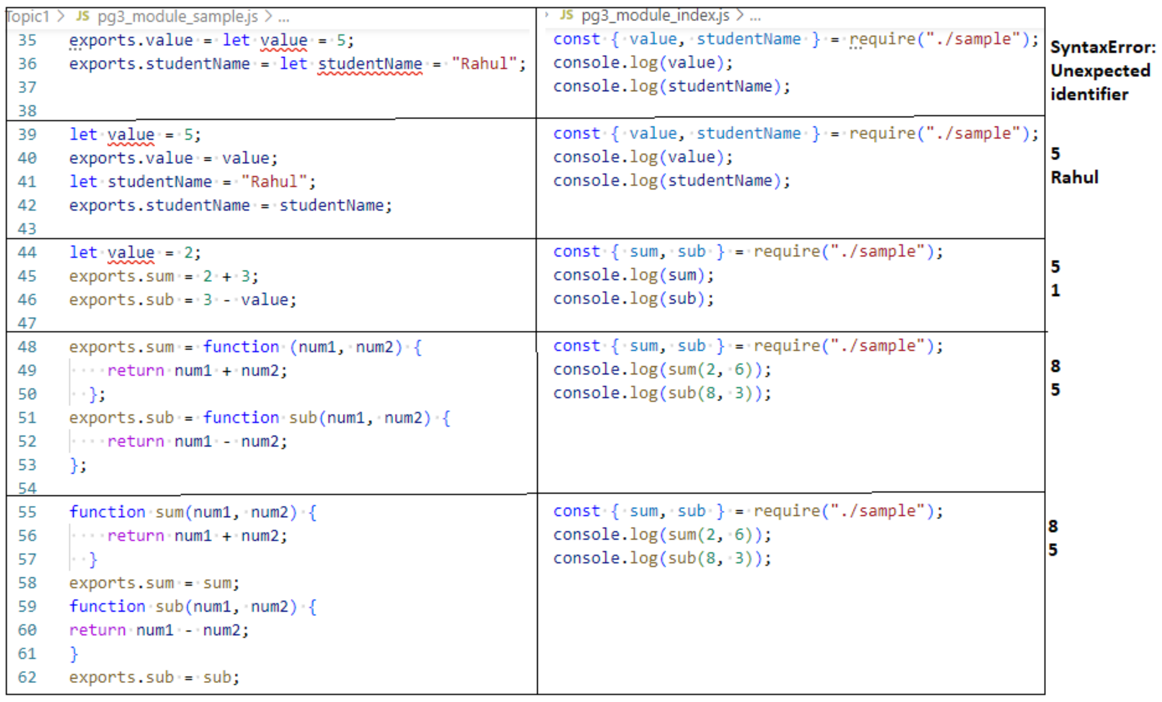
2. Named Exports

- Exporting multiple variables while defining : We cannot export boolean, number, string, null, undefined, objects, and arrays while defining.
- Exporting multiple variables after defining : We can export multiple variables after defining.
- Exporting multiple values and expressions : We can export multiple values and expressions.
- Exporting multiple functions while defining : We can export multiple functions while defining.
- Exporting multiple functions after defining : We can export multiple functions after defining.
- Exporting multiple classes while defining : We can export multiple classes while defining.
- Exporting multiple classes after defining : We can export multiple classes after defining.

In [12]:

```
from IPython.display import Image
Image("E:/code/frontend/img/node14.png")
```

Out[12]:



Topic1 > JS pg3_module_sample.js > ...

JS pg3_module_index.js > ...

```
exports.value = let value = 5;
exports.studentName = let studentName = "Rahul";
```

```
const { value, studentName } = require("./sample");
console.log(value);
console.log(studentName);
```

```
SyntaxError:
Unexpected
identifier
```

```
let value = 5;
exports.value = value;
let studentName = "Rahul";
exports.studentName = studentName;
```

```
const { value, studentName } = require("./sample");
console.log(value);
console.log(studentName);
```

```
5
Rahul
```

```
let value = 2;
exports.sum = 2 + 3;
exports.sub = 3 - value;
```

```
const { sum, sub } = require("./sample");
console.log(sum);
console.log(sub);
```

```
5
1
```

```
exports.sum = function (num1, num2) {
    return num1 + num2;
  };
exports.sub = function sub(num1, num2) {
    return num1 - num2;
};
```

```
const { sum, sub } = require("./sample");
console.log(sum(2, 6));
console.log(sub(8, 3));
```

```
8
5
```

```
function sum(num1, num2) {
    return num1 + num2;
  }
exports.sum = sum;
function sub(num1, num2) {
return num1 - num2;
}
exports.sub = sub;
```

```
const { sum, sub } = require("./sample");
console.log(sum(2, 6));
console.log(sub(8, 3));
```

```
8
5
```

2.6 Exporting multiple classes while defining

We can export multiple classes while defining.

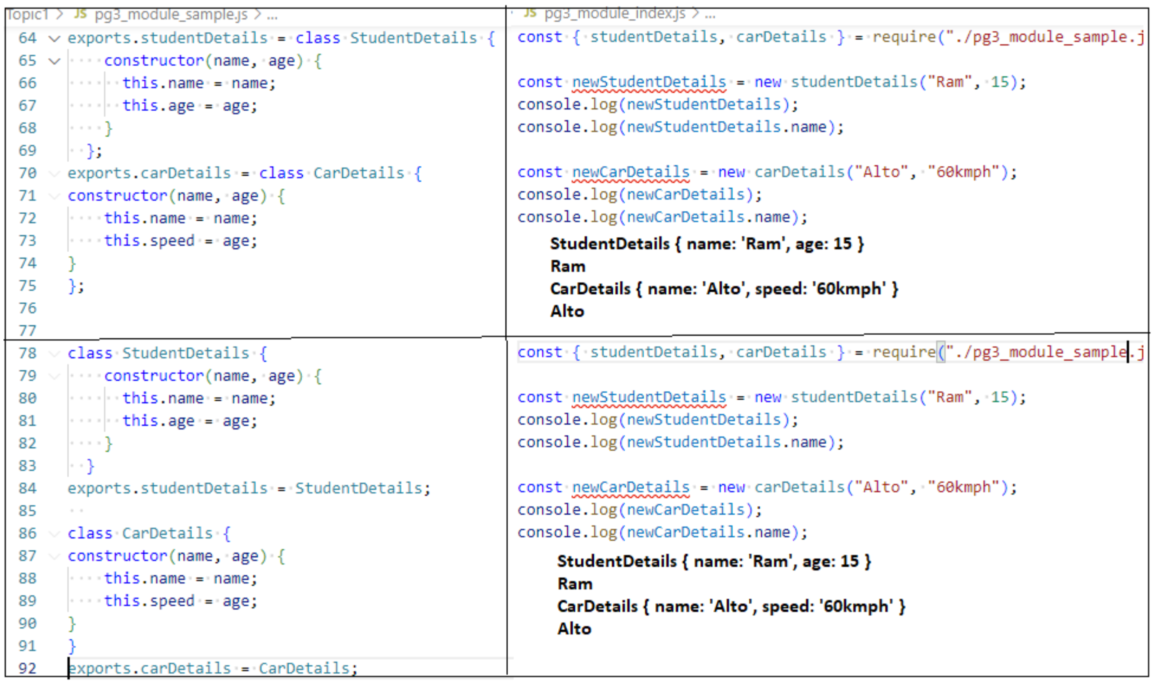
2.7 Exporting multiple classes after defining

We can export multiple classes after defining.

In [13]:

```
from IPython.display import Image
Image("E:/code/frontend/img/node15.png")
```

Out[13]:



Topic1 > JS pg3_module_sample.js > ...

```
exports.studentDetails = class StudentDetails {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
  };
exports.carDetails = class CarDetails {
constructor(name, age) {
    this.name = name;
    this.speed = age;
}
};

```

JS pg3_module_index.js > ...

```
const { studentDetails, carDetails } = require("./pg3_module_sample.j

const newStudentDetails = new studentDetails("Ram", 15);
console.log(newStudentDetails);
console.log(newStudentDetails.name);

const newCarDetails = new carDetails("Alto", "60kmph");
console.log(newCarDetails);
console.log(newCarDetails.name);
```

```
StudentDetails { name: 'Ram', age: 15 }
Ram
CarDetails { name: 'Alto', speed: '60kmph' }
Alto
```

```
class StudentDetails {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
  }
exports.studentDetails = StudentDetails;

class CarDetails {
constructor(name, age) {
    this.name = name;
    this.speed = age;
}
}
exports.carDetails = CarDetails;
```

```
const { studentDetails, carDetails } = require("./pg3_module_sample.j

const newStudentDetails = new studentDetails("Ram", 15);
console.log(newStudentDetails);
console.log(newStudentDetails.name);

const newCarDetails = new carDetails("Alto", "60kmph");
console.log(newCarDetails);
console.log(newCarDetails.name);
```

```
StudentDetails { name: 'Ram', age: 15 }
Ram
CarDetails { name: 'Alto', speed: '60kmph' }
Alto
```

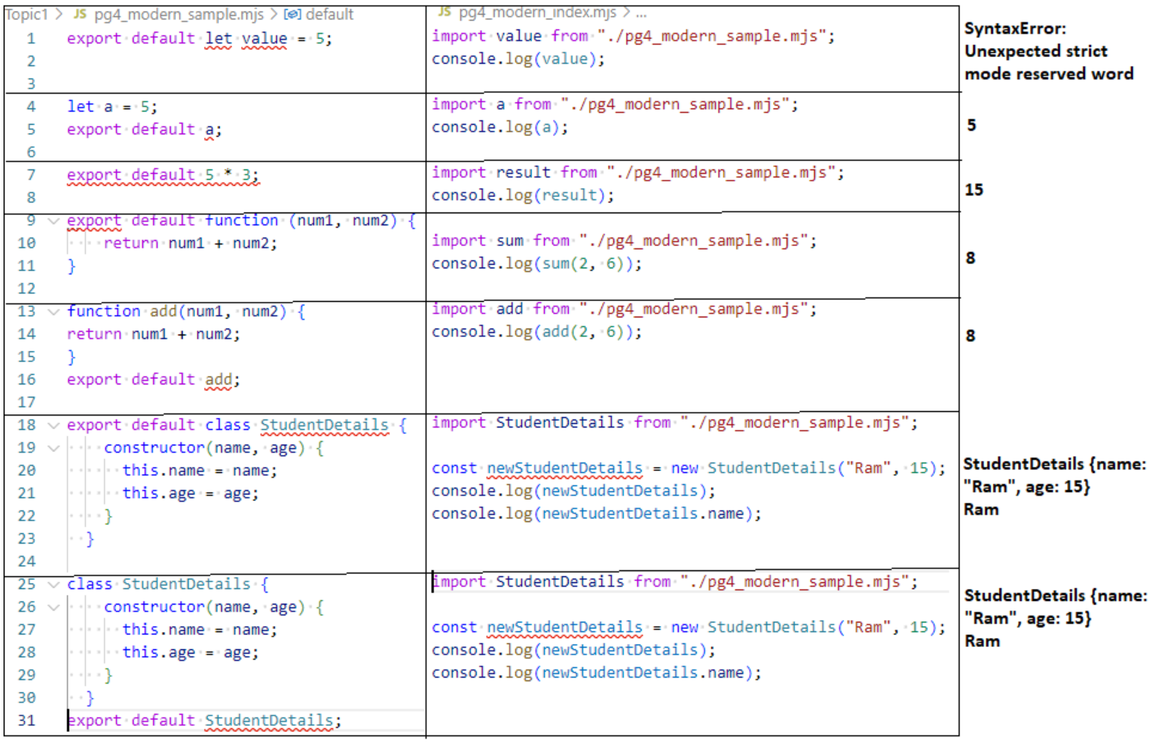
Default Exports

- Exporting a variable while defining
- Exporting a variable after defining
- Exporting a value or an expression
- Exporting a function while defining
- Exporting a function after defining
- Exporting a class while defining
- Exporting a class after defining

In [14]:

```
from IPython.display import Image
Image("E:/code/frontend/img/node16.png")
```

Out[14]:



Topic1 > pg4_modern_sample.mjs > default

pg4_modern_index.mjs > ...

```
export default let value = 5;
```

```
import value from "./pg4_modern_sample.mjs";
console.log(value);
```

```
SyntaxError:
Unexpected strict
mode reserved word
```

```
let a = 5;
export default a;
```

```
import a from "./pg4_modern_sample.mjs";
console.log(a);
```

```
5
```

```
export default 5 * 3;
```

```
import result from "./pg4_modern_sample.mjs";
console.log(result);
```

```
15
```

```
export default function (num1, num2) {
    return num1 + num2;
}
```

```
import sum from "./pg4_modern_sample.mjs";
console.log(sum(2, 6));
```

```
8
```

```
function add(num1, num2) {
return num1 + num2;
}
export default add;
```

```
import add from "./pg4_modern_sample.mjs";
console.log(add(2, 6));
```

```
8
```

```
export default class StudentDetails {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
  }
```

```
import StudentDetails from "./pg4_modern_sample.mjs";

const newStudentDetails = new StudentDetails("Ram", 15);
console.log(newStudentDetails);
console.log(newStudentDetails.name);
```

```
StudentDetails {name:
"Ram", age: 15}
Ram
```

```
class StudentDetails {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
  }
export default StudentDetails;
```

```
import StudentDetails from "./pg4_modern_sample.mjs";

const newStudentDetails = new StudentDetails("Ram", 15);
console.log(newStudentDetails);
console.log(newStudentDetails.name);
```

```
StudentDetails {name:
"Ram", age: 15}
Ram
```

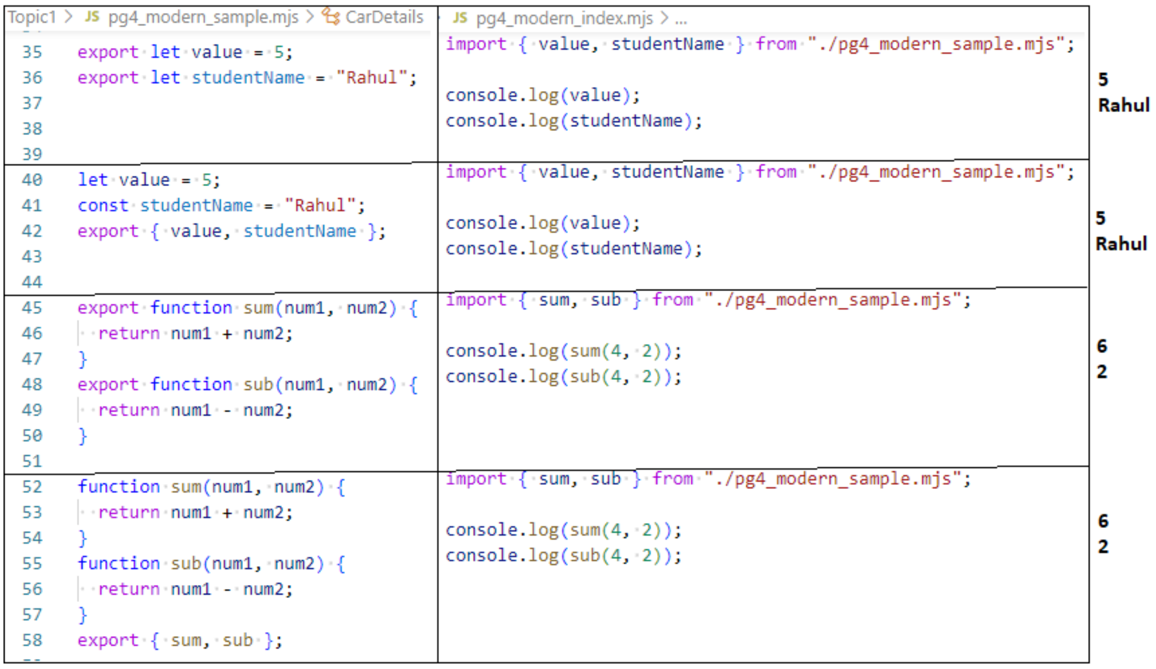
Named Exports

- Exporting multiple variables while defining
- Exporting multiple variables after defining
- Exporting multiple functions while defining
- Exporting multiple functions after defining
- Exporting multiple classes while defining
- Exporting multiple classes after defining

In [15]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/node17.png")
```

Out[15]:



Topic1 > pg4_modern_sample.mjs > CarDetails

```javascript
export let value = 5;
export let studentName = "Rahul";
```

pg4_modern_index.mjs > ...

```javascript
import { value, studentName } from "./pg4_modern_sample.mjs";

console.log(value);
console.log(studentName);
```

```
5
Rahul
```

```javascript
let value = 5;
const studentName = "Rahul";
export { value, studentName };
```

```javascript
import { value, studentName } from "./pg4_modern_sample.mjs";

console.log(value);
console.log(studentName);
```

```
5
Rahul
```

```javascript
export function sum(num1, num2) {
  return num1 + num2;
}
export function sub(num1, num2) {
  return num1 - num2;
}
```

```javascript
import { sum, sub } from "./pg4_modern_sample.mjs";

console.log(sum(4, 2));
console.log(sub(4, 2));
```

```
6
2
```

```javascript
function sum(num1, num2) {
  return num1 + num2;
}
function sub(num1, num2) {
  return num1 - num2;
}
export { sum, sub };
```

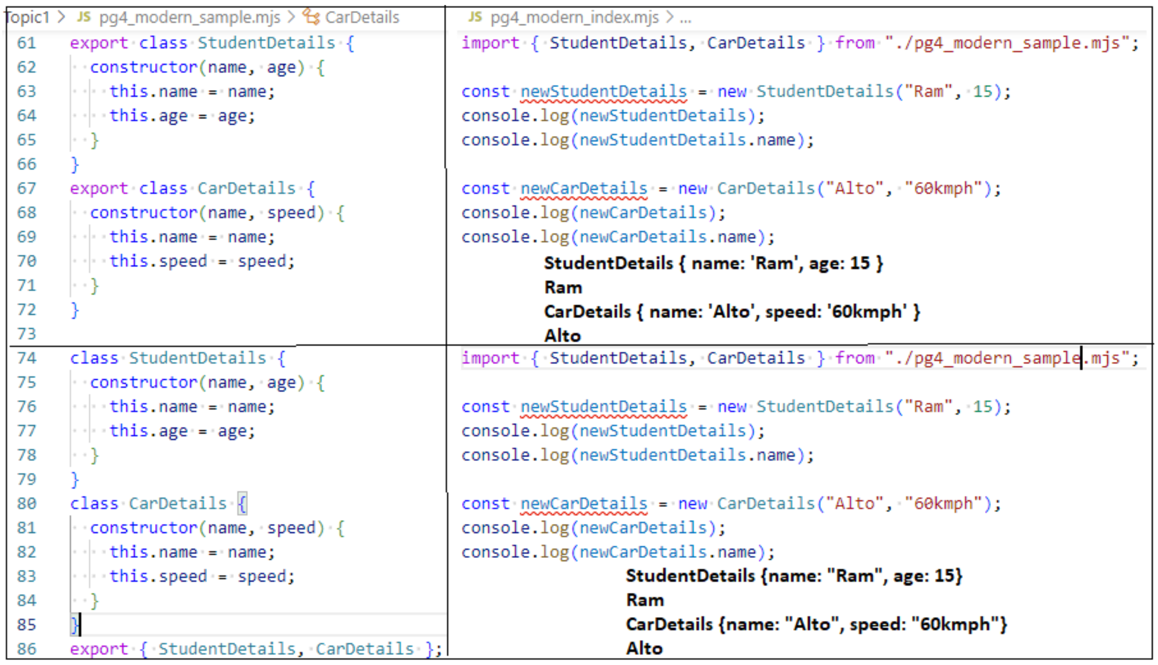
```javascript
import { sum, sub } from "./pg4_modern_sample.mjs";

console.log(sum(4, 2));
console.log(sub(4, 2));
```

```
6
2
```

In [16]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/node18.png")
```

Out[16]:



Topic1 > pg4_modern_sample.mjs > CarDetails

```javascript
export class StudentDetails {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
}
export class CarDetails {
  constructor(name, speed) {
    this.name = name;
    this.speed = speed;
  }
}
```

pg4_modern_index.mjs > ...

```javascript
import { StudentDetails, CarDetails } from "./pg4_modern_sample.mjs";

const newStudentDetails = new StudentDetails("Ram", 15);
console.log(newStudentDetails);
console.log(newStudentDetails.name);

const newCarDetails = new CarDetails("Alto", "60kmph");
console.log(newCarDetails);
console.log(newCarDetails.name);
```

```
StudentDetails { name: 'Ram', age: 15 }
Ram
CarDetails { name: 'Alto', speed: '60kmph' }
Alto
```

```javascript
class StudentDetails {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
}
class CarDetails {
  constructor(name, speed) {
    this.name = name;
    this.speed = speed;
  }
}
export { StudentDetails, CarDetails };
```

```javascript
import { StudentDetails, CarDetails } from "./pg4_modern_sample.mjs";

const newStudentDetails = new StudentDetails("Ram", 15);
console.log(newStudentDetails);
console.log(newStudentDetails.name);

const newCarDetails = new CarDetails("Alto", "60kmph");
console.log(newCarDetails);
console.log(newCarDetails.name);
```

```
StudentDetails {name: "Ram", age: 15}
Ram
CarDetails {name: "Alto", speed: "60kmph"}
Alto
```

Introduction to Node JS | Part 2

- Core Modules - path
- Packages - Third part packages
- NPM CLI - npm install, npm init
- third party packages - date-fns

core modules : The Core Modules are inbuilt in Node JS.

- path : to Handle file paths
- fs : to Handle the file system
- url : to Parse the URL strings

1. Path : The path module provides utilities for working with file and directory paths. It can be accessed using:

index.js

- const path = require("path");
- const filePath = path.join("users", "ravi", "notes.txt");
- console.log(filePath);

output

- root@123# node index.js
- users/ravi/notes.txt

2. Package : A package is a directory with one or more modules grouped.

Node Package Manager (NPM) : repo for third party packages(https://www.npmjs.com/ (https://www.npmjs.com/))

- NPM is the package manager for the Node JS packages with more than one million packages.
- NPM company provides a command line tool which allows you to publish, discover, install, and develop node programs.

project: when we working with large packages, it's a good practice to split our code into multiple modules

- makes code more readable
- easy to maintain
- easy to test

CLI : NPM CLI helps us to setup the Node JS Project to organize various modules and work with third-party packages.

- npm init -y --> Initializes a project and creates a package.json file
- npm install package-name --save --> Installs the third-party packages

3. Steps to create a Node JS Project

- Run the below commands in the terminal.
- Create a new directory/folder : mkdir myapp

- Move into the created folder : cd myapp
- Initialize the project : npm init -y
- it creates package.json file which contians information related to application

4. Third-Party Packages

- The Third-Party Packages are the external Node JS Packages.
- They are developed by Node JS developers and are made available through the Node ecosystem. (https://www.npmjs.com/ (https://www.npmjs.com/))
- search for date-fns --> click on it --> copy the command(npm install date-fns --save)
- date-fns : It is a third-party package for manipulating JavaScript dates in a browser & Node.js.
  - npm install date-fns --save
- addDays : It adds the specified number of days to the given date.

index.js

- const addDays = require("date-fns/addDays");
- const result = addDays(new Date(2021, 0, 11), 10);
- console.log(result);

output

- root@123# node index.js
- 2021-01-21T00:00:00.000Z

Note:

- While creating the Date() object, we have to provide the month index from (0-11), whereas we will get the output considering Jan=1 and Dec=12.