

# Plataforma Node.js

## Módulos em JavaScript e Node.js

# Sistema de Módulos

Node.js suporta dois sistemas de módulos para a Linguagem JavaScript

## CommonJS

```
// Formato da importação  
const fs = require( 'fs')  
const { env } = require('process')
```

```
// Formato da exportação  
const msg = 'Hello'  
module.exports = { msg }
```

## ES6 Modules

```
// Formato da importação  
import fs from 'fs'  
import process, { env } from 'process'
```

```
// Formato da exportação  
export const msg = 'Hello'
```

# Sistema de Módulos

FEATURES	REQUIRE	IMPORT
Syntax	<code>const x = require()</code>	<code>import x from "./"</code>
Used In	CommonJS Modules	ES Modules
Asynchronous	✗	✓
Conditional Usage	✓	✗
Selective Load	✗	✓
Node Support	✓	V 13+
TypeScript Support	✓	✓

# ES Modules (ESM)

A Linguagem JavaScript incorporou o conceito de módulos a partir do ECMAScript 6 (ES2015).

O Node.js, desde a versão 13.2.0, suporta a implementação dos **ES Modules (ESM)**.

Para habilitar ES Modules no lugar do CommonJS, podem ser adotadas as seguintes estratégias:

- Acrescentar extensão **.mjs** aos arquivos
- Incluir o atributo type no **package.json**
- Usar a flag **-input-type=module** em comandos via string

```
$ node app.mjs
```

```
// package.json
{
  "name": "my-app",
  "version": "1.0.0",
  "type": "module",
  // ...
}
```

```
$ node -input-type=module
  -e "import sep from path;
      console.log(sep)"
```

# ES Modules (ESM)

Suporte para módulos de componentes na Linguagem JavaScript

```
// lib/math.js
export function sum(x, y) {
  return x + y;
}
export var pi = 3.141593;
```

```
// app.js
import * as math from "lib/math";
alert("2π = " + math.sum(math.pi, math.pi));
```

```
// otherApp.js
import { sum, pi } from "lib/math";
alert("2π = " + sum(pi, pi));
```


# ES Modules (ESM)

## Tipos de exportação: **Named Exports** | Default Export

Permite exportação de vários componentes independentes por módulo

```
// lib.js
export const sqrt = Math.sqrt;
export function square(x) {
  return x * x;
}

export function diag(x, y) {
  return sqrt(square(x) + square(y));
}
```



```
// main.js
import { square, diag } from 'lib';

console.log(square(11)); // 121
console.log(diag(4, 3)); // 5
```

Fonte: ECMAScript 6 modules: the final syntax - <http://2ality.com/2014/09/es6-modules-final.html>

# ES Modules (ESM)

## Tipos de exportação: Named Exports | **Default Export**

Permitem um único componente a ser exportado pelo módulo como default

```
// myFunc.js  
export default function () { ... };
```

```
// main1.js  
import myFunc from 'myFunc';  
myFunc();
```

```
// MyClass.js  
export default class { ... };
```

```
// main2.js  
import MyClass from 'MyClass';  
let inst = new MyClass();
```

Fonte: ECMAScript 6 modules: the final syntax - <http://2ality.com/2014/09/es6-modules-final.html>

# ES Modules (ESM)

## Formas de Importação de Componentes

```
// Default exports and named exports
import theDefault, { named1, named2 } from 'src/mylib';
import theDefault from 'src/mylib';
import { named1, named2 } from 'src/mylib';

// Renaming: import named1 as myNamed1
import { named1 as myNamed1, named2 } from 'src/mylib';

// Importing the module as an object
// (with one property per named export)
import * as mylib from 'src/mylib';

// Only load the module, don't import anything
import 'src/mylib';
```

Fonte: ECMAScript 6 modules: the final syntax - <http://2ality.com/2014/09/es6-modules-final.html>