



Mo Tu We Th Fr Sa Su

Memo No.

Date

/ /

NODE JS CHEATSHEET

Running NODE JS

node - Run Node REPL In terminal

node -version - print current node version

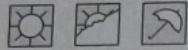
node filename.js - Execute node code

filename.js

Node Js Global Object

In node we have a global object that we can always access

Features that we expect to be available everywhere live in Global Object.



Mo Tu We Th Fr Sa Su

Memo No. _____

Date / /

Foreg - SetTimeout () => {
 console.log ("Hello");
}, 5000);

probably the most famous
global is global.console.log
which we write as console.log

Node Js Module System

In node.js, Each file is
treated as separate module

Modules provide us way of
reusing existing code.



Mo Tu We Th Fr Sa Su

Memo No. _____

Date / /

The require Function.

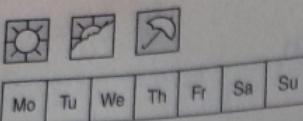
We can re-use existing code by using Node built-in `require()` function.

This function imports code from another module

Eg → `const fs = require('fs');`
`fs.readFileSync('hello')`

Built In Modules

Some modules are built in to Node. These module contains Node specific features



key build in modules | include

- * fs - read and write file
- * path - combines path regardless of which OS you're using
- * process - information about current running process.

Eg → process.argv for arguments passed in or process.env for environment variables

- * http → make request and create http servers
- * https → work with secure HTTP servers using SSL/TLS
- * events → work with event emitter
- * crypto → cryptography tools like encryption and hashing



Mo Tu We Th Fr Sa Su

Memo No.

Date

/ /

Creating Modules

We can create our own module

by exporting a function from
a file and importing
it in another module

// In Src / file Module.js

```
function read (filename) {}  
function write (filename, data) {}
```

```
module.exports = {
```

```
  read,  
  write
```

```
} ;
```

// In Src / file Module. Say Hello.js

```
const { write } = require ('./file Module.js')  
write ('hello.txt', 'Hello World');
```



Mo Tu We Th Fr Sa Su

Memo No. _____

Date / /

Some Node Modules may instead use short hand syntax to export functions

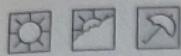
// In src/fileModule.js

```
exports.read = function read(filename){  
}  
exports.write = function write(filename, data)  
{  
}
```

ECMASCRIPT MODULES

The imports above use a syntax known as Common JS (CJS) modules.

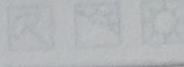
Node treats Javascript code as Common JS module by default.



Mo Tu We Th Fr Sa Su

Memo No.

Date



More recently, you may have seen the ECMAScript module (ESM) syntax.

This is Syntax used in TypeScript

// In src / fileModule.mjs

```
function read(filename) {}
```

```
function write(filename, data) {}
```

```
export { read, write };
```

// In src / sayHello.mjs

```
import { write } from './response.mjs';
write('hello.txt', 'Hello World');
```

We tell node treat Js code as ECMAScript module by using .mjs file extension.