



M: Mon

T: Tue

W: Wed

Th: Thu

F: Fri

S: Sat

Su: Sun

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

Node Js Global Object

In Node.js 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

Monday
Date

Memo No.

Date

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

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

Node Js Module System

In nodejs, 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 reuse existing code by using Node built-in `require()` function:

This function imports code from another module.

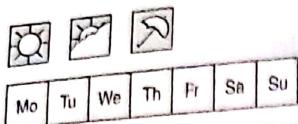
Eg → `const fs = require('fs');`
`fs.readFileSync('hello')`

Built In Modules

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



Scanned with OKEN Scanner



Memo No. _____

Date / /

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

II In Src / fileModule.js

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

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

```
module.exports = {
```

```
    read,
```

```
    write
```

```
}
```

II In Src / fileModule. SayHello.js

```
const { write } = require ('./fileModule')
```

```
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)  
{}  
S 3 (start) b3 (midpoint) S 3 (end) e3 (midpoint)
```

ECMASCRIPT MODULES

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

Node treats Javascript code as common Js module by default.

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.

Mo	Tu	We

Mon
Tue
Wed
Thu
Fri
Sat
Sun

Memo No. _____
Date / /

NODE JS PACKAGES

A package is collection of Node modules along with package.json file describing the package.

NPM COMMANDS

* npm start - executes current Node package defined by package.json -
Defaults to executing node server.js

* npm init - Initialize a fresh package.json file

* npm init -y - Initialize a fresh package.json file , accepting all default options.

- deepa chaurasiya

* `npm install` - Equivalent to `npm i`
It will install all node modules

* `npm install <package>` - Install
a package from NPM registry
at www.npmjs.com.

* `npm install -D <package>`
Install a package ~~globally~~ as
a development dependency

* `npm install -g <package>`
Install package globally

* `npm update <package>`
Update already installed package

* `npm uninstall <package>`
Uninstall a package from
your node_modules



Mo Tu We Th Fr Sa Su

all memo
date

Memo No.

Date / /

- * npm outdated - check for outdated package dependencies
- * npm audit - check for security vulnerabilities in package dependencies
- * npm audit fix - try to fix security vulnerabilities by automatically updating vulnerable packages.

package.json

Most node applications we create includes a package.json file

Which means our node applications are also Node packages

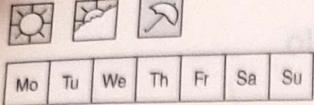
- deepa chaurasia

The package.json file contains

- * Name, version, description, license of current package
- * Scripts to automate tasks like start, test and Install
- * list of dependencies that are required to be installed by current package

node_modules

This folder get installed when you run npm install. The package listed as dependencies in your package.json are downloaded from Npm registry and put in folder node_modules



Mo Tu We Th Fr Sa Su

Memo No.

Date / /

It contains not only direct dependencies.

The entire dependency tree lives in node-modules.

package-lock.json

The package-lock.json is automatically created by NPM to track the exact versions of packages that are installed in node-module

Share your package-lock.json

If you want to ensure everyone is running same exact version of every package

—deepa Chaurasia

Node.js Event Emitter

Node.js provides a built-in module
to work with events

```
const EventEmitter = require('event');
const celebrity = new EventEmitter()
```

```
celebrity.on('success', () => {
```

~~congrats~~

```
    console.log('Success');
```

```
});
```

```
celebrity.emit('success');
```

Backend Concepts

* Client Side Architecture

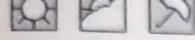
Your Front-end is actually the client.

Your Back-end is usually the server.

In client-server architecture
client gets access to data
(or resource) from server

The client can display data
and interact with this data.

The client and server communicate
with each other using the HTTP
protocol.



Mo Tu We Th Fr Sa Su

Memo No.

Date / /

* API - Short for Application programming Interface

This is set of functions or operations that your backend server supports.

The frontend interacts with backend by only using these operations.

On the web, backend APIs are commonly defined by a list of URLs, corresponding HTTP methods, and queries and parameters.



Mo Tu We Th Fr Sa Su

Memo No.

Date

* CRUD → short for create
Read Update Delete

These are basic operators that every API supports on collection of data.

Your API will usually save these collection of data in a database

* Restful - Restful API's are those that follow certain constraints. These include

- client-server architecture.

Client gets access to resource from server using HTTP protocol

- Stateless Communication -
Each request contains all the information required by server to handle that request.

Every request is separate from other request

- Cacheable - The stateless communication make caching easier.

CRUD HttpMethod
operation

Create Post

Read Get

Update Put

Delete Delete



Mo Tu We Th Fr Sa Su

Memo No.

Date

Node Js Folder Structure

Node servers usually follow
Model View controller pattern
(i.e MVC)

Controllers are grouped together
based on which feature or
collection they are related to.

views are managed by front-end

CROSS-ORIGIN Resources

Sharing

All developer come across cors
error.



Mo	Tu	We	Th	Fr	Sa	Su
----	----	----	----	----	----	----

Memo No. _____

Date / /

Browser follows same Origin policy (SOP) which prevents request being made across different origins.

This is designed to stop malicious servers from stealing information that doesn't belong to them.

CORS allows us to allow or whitelist other origin that we trust, so that we can make request to servers that don't belong to us.