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 in 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.

Memo No.
Date / /

Mo Tu We Th Fr Sa Su

For eg -
```
setTimeout(() => {
    console.log("Hello");
}, 5000);
```

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

NodeJs 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 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

Memo No.
Date / /

Mo Tu We Th Fr Sa Su

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 arguements 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~~. SayHello.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 CommonJs module by default.

Mo Tu We Th Fr Sa Su

Memo No.

Date / /

More recently, you may have seen the ECMA script 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 Th Fr Sa Su

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 chaurasia

Mo Tu We Th Fr Sa Su

Memo No.
Date / /

* npm install – Equivalent to npm i
It will install all node modules

* npm install <package> – Install a package from NPM registry at https://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 unistall <package>
Uninstall a package from your node_modules

Mo Tu We Th Fr Sa Su

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

Mo Tu We Th Fr Sa Su

Memo No.
Date / /

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_module

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

Mo Tu We Th Fr Sa Su

Memo No.
Date / /

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');
```

Mo Tu We Th Fr Sa Su

Date / /

Backend Concepts

* Client Side Architecture

Your Fron-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 URL's, 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

Mo Tu We Th Fr Sa Su

Memo No.
Date / /

- 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 operation	HttpMethod
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 Resourche 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.