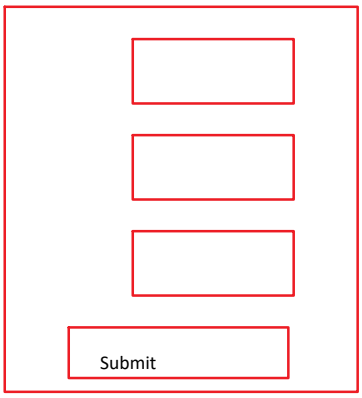
Node JS

Friday, February 14, 2025 9:06 AM



Java script : on client side if we want to apply few validations and if want to check on client side itself (browser to View the output / console also)

Node js is an open server environment which uses the java script on server side / server (Express) (on top of the Browser execution it executes on a server side / we will get the output using console.log)

Node js provided (modules and packages) -- how we can install or remove those packages with the help of npm (node package manager)

It uses the asynchronous programming---

How to fetch apis.

GitHub profile and checkin all your practice code in the site.
I want to see you guys share the projects on github or running on your computer.
Go beyond just your training. Pick the topics from the TOC and do your own learning.

Promise === used as an asynchronous operations --- let p = new Promise((resolve , reject) => { ……})

.then ==== Executes a function after a promise resolves === fetch(url).then(response => response.json())

setTimeOut() --- Runs s function afer a delay ---

Await --- waits for a promise inside an async function === let data = await fetch(url)

Async == declare a function as asynchronous === async function fetchData()
{

}

```
async function fetchUser() {
 console.log("fetching data...");
 let response = await fetch("https://jsonplaceholder.typicode.com/users/1");
 let user = await response.json();
 console.log("User data : ", user);
}
console.log("Initiate the process");
fetchUser();
console.log("process completed");
```

```
//using .then
console.log("Initiate the process");
async function fetchUser() {
 console.log("fetching data...");
 fetch("https://jsonplaceholder.typicode.com/users/1")
  .then((response) => response.json())
  .then((user) => console.log("User Data ...", user));
}
fetchUser();
console.log("process completed");
```

Middleware : have different types

What is middleware _> It is a functions that executes during the request-response cycle . Different types of middleware

1. Application -Level --- app.use() --- applicable on all the routes --- Global(All-routes)
2. Built-in --- app.use(express.json()) -- JSON , static files
3. Route-Level -- app.get('/path', middleware) -- specific routes
4. Third-Party -- ??? -- logging , security -- app.use(third Party)
5. Error-handling --- app.use((err, req , res , next)) Error Responses

Few things to remember

1. Middleware always has (req, res ,next)
2. We have to call next() to pass control to the next middleware
3. Middleware is executed in order of definition .
4. Use express.json() for POST request body parsing

Multithreading : Process and a thread
Multiprocessing and Multitasking

		Microsoft word	SQL	Web browsers
Multiprogramming	You can work with multiple programs (Browser , Word , Excel ,PPt)	When you have also opened excel , ppt	Multiple databases instances are running	Multiple tabs (gmail , youtube , chatgpt)
Multiprocessing (Multiple CPUs)	(Multiple CPU 8CORE) to work with different or multiple task for a better performance	Spell check , grammer check, auto saving of documents in seperate processes , Mutiple files are opened	Multiple transactions , query processing across CPUS	Each tab runs in a separate process
Mutitasking (Multi tasks)	Different task we have opened in one of the program (Spell checker , Printing , writing a document)	Editing , proofing , making comments , spell checker, autosaving of one document	Running multiple queries simultaneously	Streaming a youtube while browsing
Multithreading Multithreads	For executing multiple thread as an process but it is light weight	Multithread , word uses threads for taking user input , and render the text	Each query execution runs on separate threads	Each tab has threads for render the page , network , javascript execution

Foreground Threads : keeps the application running until they complete
Background Threads : Terminate when the main thread ends.