

Use cases of Node.js → runtime built on chrome-v8.

- APIs & servers
- Databases
- CLIs
- Build Tools
- Automations
- Basic Scripting
- CPU shopping bots.

\* JS is single threaded.  
(runs event loop).

nvm vs npm

nvm → makes switching between node version easier  
node.js version manager

npm → helps install packages, libraries, plugins, frameworks & applications.

to check the installation location of  
node

which node

REPL

↳ Run evaluate  
print loop

- to check version

node --version

deno → modern runtime for JS & TS  
(node's alternative)

module → reusable chunk of code that has  
its own context.

.mjs is used for ECMAScript<sup>script</sup> module exports.

instead of using .mjs everytime 'just' do some  
changes to package.json.

instead of callback we use promises.

if a func returns promises we can use await.

Error Handling



AA

intro-to-nodejs-v2-site.vercel.app



Decks - AnkiWeb

intro-to-nodejs-v2-site.vercel.app/lesson/07-error-hand...

Welcome

What is Node.js

Installing Node.js

Executing Node

Globals

Modules

File System

**Error Handling**

Packages

CLIs

Servers

Testing

Deployment

## Async Errors

When dealing with callbacks that are used for async operations, the standard pattern is:

```
fs.readFile(filePath, (error, result) => {
  if (error) {
    // do something
  } else {
    // yaaay
  }
})
```

using callback func.

Callbacks accept the `(error, result)` argument signature where error could be `null` if there is no error.

For `promises`, you can continue to use the `.catch()` pattern. Nothing new to see here.

For `async / await` you should use `try / catch`.

```
try {
  const result = await asyncAction()
} catch (e) {
  // handle error
}
```

## Sync Errors

For sync errors, `try / catch` works just fine, just like with async await.

```
try {
  const result = syncAction()
} catch (e) {
  // handle error
}
```

## Catch All

Finally, if you just can't catch those pesky errors for any reason. Maybe some lib is throwing them and you can't catch them. You can use:

```
process.on('uncaughtException', cb)
```

\* be good with error handling.

? Does python have threads??

## Creating local packages & npm

with package.json we convert our app into a package.

don't use npm & yarn in same project.

Now npm is better than yarn.

use node server for small tasks,  
basically CRUD APIs.

express → framework for building server

middleware

↳ (function between what you want  
to do with requests & when it came  
in)

can manipulate, inspect etc. before  
sending it off to other middleware or  
controller.

## Testing with Node.js

→ One of the most common usecases for Node.js is writing tests for Node.js & frontend apps.

### Unit test

Test a little chunk of code,  
(function)

vanilla unit tests can be written with assert.

we use Jest to test JS scripts, it provides better clarity and easier to use.