Node JS!

Node JS

- NodeJS is an engine for Javascript that runs on its own server.
- NodeJS is built on top of google's v8 javascript engine but has been expanded to be significantly more than that.
- Node is fast, lightweight, and easy to deploy

Node JS: Frameworks

Frameworks: Node is the bedrock upon which many frameworks exist These frameworks built by others are often massive open source projects that then get used by many others. Examples include:

- Express: A powerful web server
- React: A powerful front end framework built around the DOM
- Redux: Another front end framework, built with global data passing in mind.
- Nextjs - a framework to render serverside react apps

Modules!

- When you write code, you clearly do not want to keep all of your work within one single file.
- For the sake of readability / encapsulating common functions, it is best to break your code apart into separate modules.
- Encapsulation of code is key to a good project!

Modules -- Javascript in the browser vs Javascript in Node Server

- Browserscript: Interacts with the DOM/Cookies/Window
- Nodescript: Node works as a server, granting usage to the file system, locally installed packages, environmental control (you can't control what browser the callers are using, but you can control what environment your server's running on).
    - There are ways to get around that when serving up front end pages to the user with a package called babel, which transpiles code to its ES5 compatible javascript. We'll get to

that much later.

Modules -- Javascript in the browser vs Javascript in Node Server

Module Standards:

- Browser Javascript: starting to use the ES Modules standard, letting code use import to import modules.
- Node: Uses CommonJS Module System, which uses the require() syntax to bring in modules.

Due to the fragmentation of who wrote what when, you'll often see two separate ways of importing modules:

Modules - Differences in Importing: ES Modules

```
// ES Module import:
import moduleObject from 'someModule'
```

Modules - Differences in Importing: CommonJS

```
// CommonJS Module Import
const moduleObject = require("someModule")
```

Modules - Exporting

When exporting data, you can export a default function, or multiple functions to be treated as an object:

`someFile.js`

```
export default () => "This is a value returned from a function!"
```

`someImportingFile.js`

```
import whateverNameIWant from './someFile.js'
// const whateverNameIWant = require('./someFile.js')

console.log(whateverNameIWant())
```

Modules - Exporting

- When exporting non-defaults:

`someOtherFile.js`

```
export const someFunction = () => {
  return "values"
}

export const someOtherFunc = () => {
  return 23
}
```

Can be imported as:

```
import * as something from './someOtherFile.js'
// const something = require('someOtherFile.js')

const someValue = something.someFunction()
const someOtherValue = something.someOtherFunc()
```

OR

```
import { someFunction, someOtherFunc } from './someOtherFile.js'
// const { someFunction, someOtherFunc } = require('./someOtherFile.js')

const someValue = someFunction()
const someOtherValue = someOtherFunc()
```

Modules - Exporting

Functions can also be exported as: `defaultExportObject.js`

```
const someFunc = () => {
  return 23
}

const someOtherFunc = () => {
  return [1,2,3,4,5,6]
}

export default {
  someFunc,
  someOtherFunc
}
```

```
import defaultExportObject from './defaultExportObject.js'

const someNumber = defaultExportObject.someFunc()
```