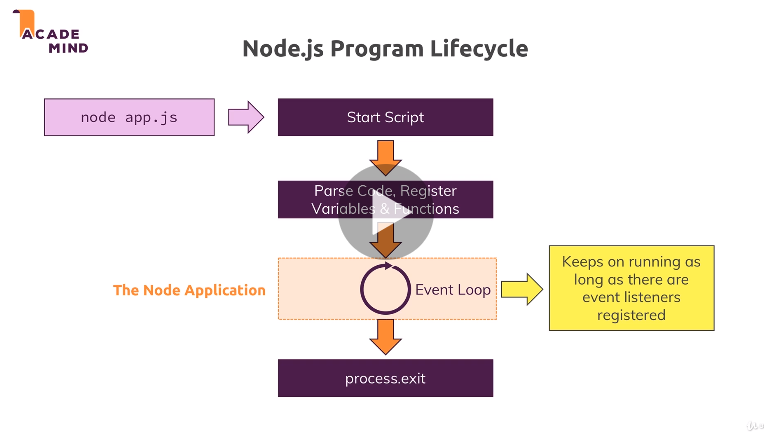


Data encryption - SSL encryption

Spin out a server

Node lifecycle & event



TO Quit node.js server – ctrl + c

Headers – metadata tells what info present in the document

**Request & Response Headers**

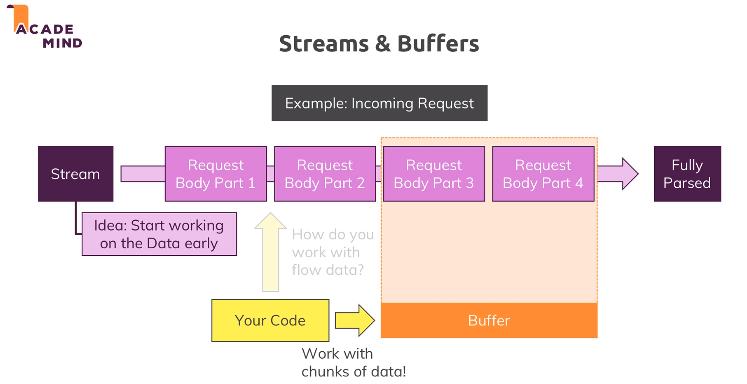
On both requests and responses, Http headers are added to transport metadata from A to B.

The following article provides a great overview of available headers and their role: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

Routing requests

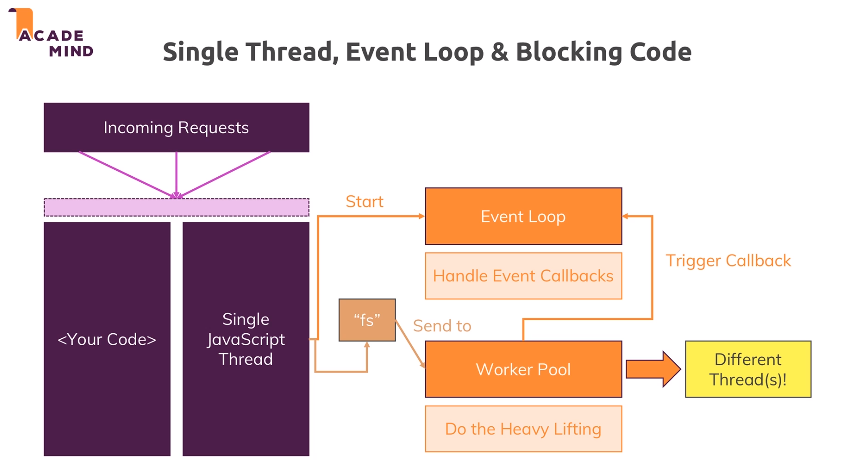
Redirecting requests

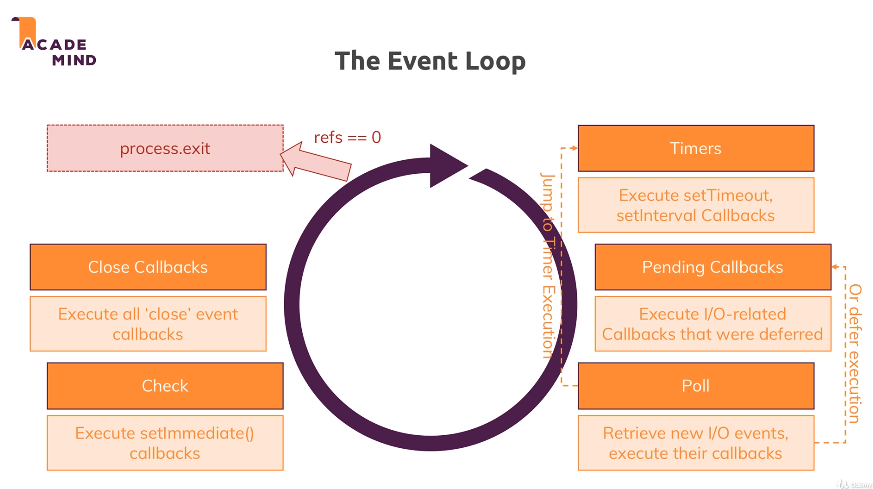
Parsing request bodies

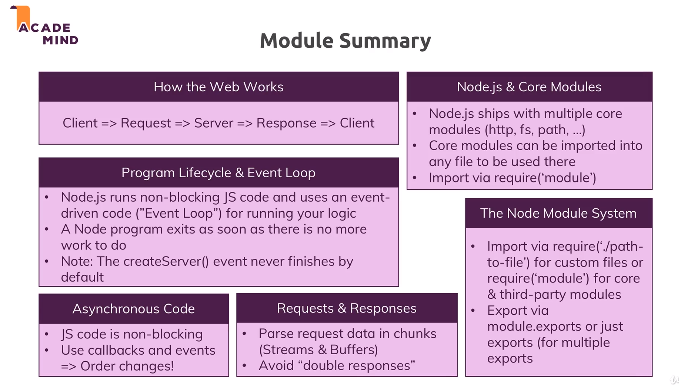


Understanding event driven code execution

Blocking and Non blocking code







**Useful resources:**

Official Node.js Docs: https://nodejs.org/en/docs/guides/

Full Node.js Reference (for all core modules): <https://nodejs.org/dist/latest/docs/api/>

Npm – node package manager

$> Npm init

Creates a package.json file

$> npm install –g npm

// to update npm

// -g – global

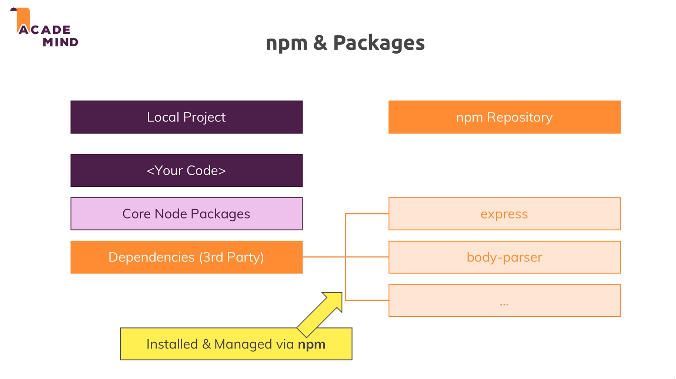
// --save—dev – development

// --save - production

$> npm run <custom-scripts>

$> npm start

$> npm install



Global Features vs Core Modules vs Third-Party Modules

The last lectures contained important concepts about available Node.js features and how to unlock them.

You can basically differentiate between:

* **Global features**: Keywords like const or function but also some global objects like process
* **Core Node.js Modules**: Examples would be the file-system module ("fs"), the path module ("path") or the Http module ("http")
* **Third-party Modules**: Installed via npm install - you can add any kind of feature to your app via this way

**Global features** are **always available**, you don't need to import them into the files where you want to use them.

**Core Node.js Modules** don't need to be installed (**NO npm install** is required) but you **need to import them** when you want to use features exposed by them.

Example:

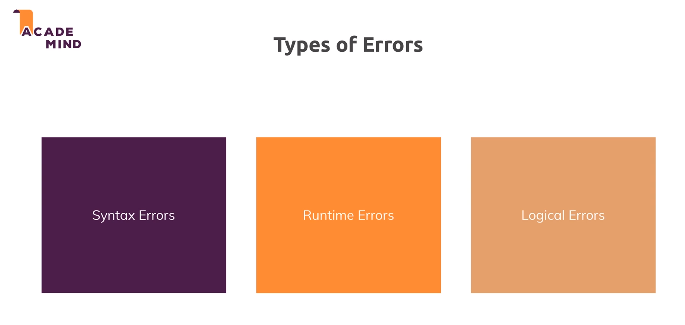
const fs = require('fs');

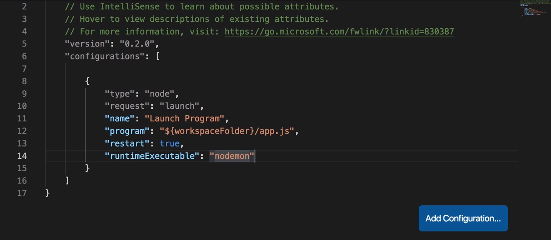
You can now use the fs object exported by the "fs" module.

**Third-party Modules** **need to be installed**(via npm install in the project folder) **AND imported**.

Example (which you don't need to understand yet - we'll cover this later in the course):

1. // In terminal/ command prompt
2. npm install --save express-session
3. // In code file (e.g. app.js)
4. const sessions = require('express-session');

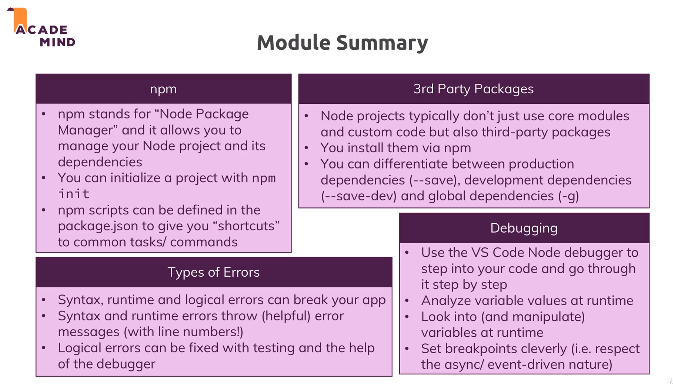




Debugging Node.js in Visual Studio Code

Want to dive super-deep into the latest debugging capabilities Visual Studio Code gives you (for Node.js apps)?

This article will be very helpful: <https://code.visualstudio.com/docs/nodejs/nodejs-debugging>



Attached, you find the source code for this section.

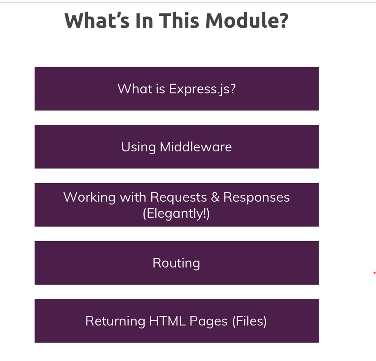
When using my source code, make sure to run npm install in the extracted folder!

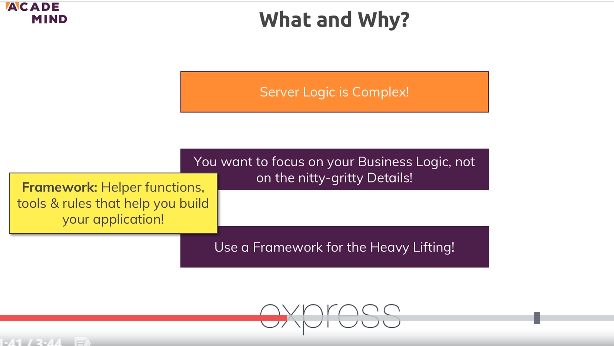
Useful resources:

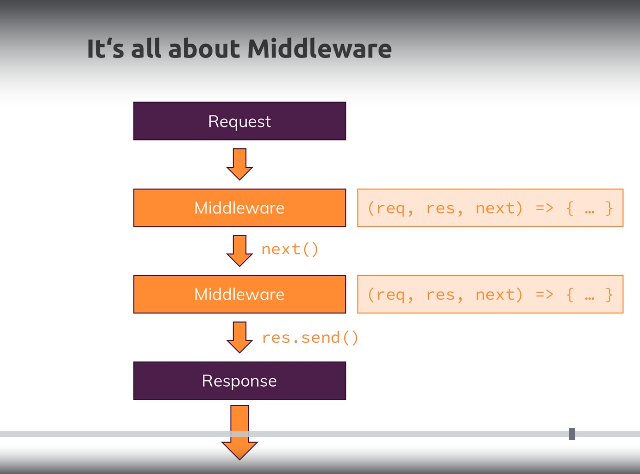
* More on debugging Node.js: <https://nodejs.org/en/docs/guides/debugging-getting-started/>
* Debugging Node in Visual Studio Code: <https://code.visualstudio.com/docs/nodejs/nodejs-debugging>

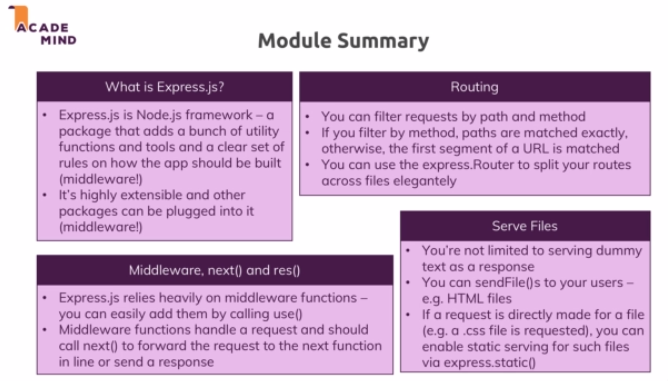
Resources for this lecture

* [01-understanding-npm-scripts.zip](javascript:void(0))
* [02-using-nodemon-for-autorestarts.zip](javascript:void(0))
* [03-finished.zip](javascript:void(0))









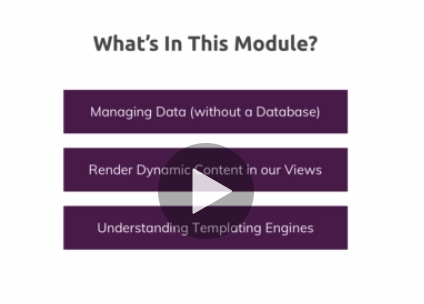
Useful Resources & Links

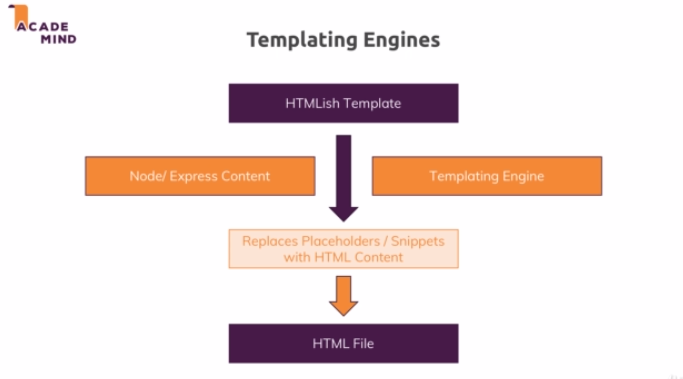
Attached, you find the source code for this section.

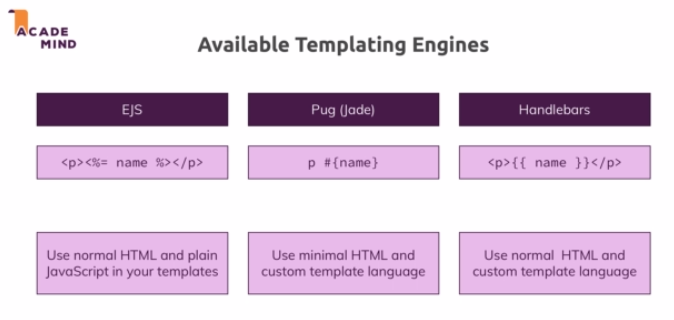
When using my source code, make sure to run npm install in the extracted folder!

Useful resources:

* Express.js Official Docs: <https://expressjs.com/en/starter/installing.html>







Official Pug Docs

Want to learn more about Pug? Check out their official docs: <https://pugjs.org/api/getting-started.html>

$> npm install --save [express-handlebars@3.0](mailto:express-handlebars@3.0)