**MERN - CODEHACKS**

MERN (stack name)

M ongoDB (database)

E xpress (node framework)

A ngular (JavaScript/typescript framework)

N ode (web server and compiler)

**Intro to Node**

Node is a JavaScript interpreter that allows us to run JavaScript code free from the browser.

$ node -v (check the version)

$ node (enter JavaScript environment)

$ node your\_file\_name.js (runs file of code written in text editor)

**ECMAScript & ES6**

-ES6 is a superset of ES5. Because ES6 is not a full language in itself, must learn both.

-The vast majority of existing JavaScript is still ES5, including -libraries, legacy code bases, and examples you encounter on the web.

-Many of ES6's most important features are syntactic sugar. Not necessarily needed, but they make the language sweeter to write.

-JavaScript is the language, ECMAScript is the standard.

**Debugging JS**

-When debugging your JS, always confirm that you have **no syntax errors** first. Your code will not run until those syntax errors are fixed.

-When hunting syntax errors, typically the actual problem is **before the line** that errored. Work backwards from there.

-Console.log **everything!** Often times the biggest errors come from faulty assumptions. That variable you thought was a string was actually an array of strings, which can completely change your logic.

-Run your code **early and often**, especially at first. If you're writing 20 or 30 lines of JavaScript at a time before seeing if any of it works, you're coding too much! The stronger you get with JS, the more assumptions you can make, but at first assume nothing!

**What is Scope?**

Global Scope

Every JavaScript application has a global scope. By definition that makes it omnipresent, which means that any variables or functions defined within are always available to every aspect of your application.

Local Scope

Local scope, in contrast, has much narrower program visibility. It is localized to the particular function in which that information is defined.

 let allows for reassignment of our declared variables content and does not require a value upon declaration

const must assign a value at creation and that value may not change for the life of the variable

**Hoisting**

Variable declarations (var) rise to the top of their scope like hot air balloons.

Functions create their own scope and act like cages, preventing declarations from rising out.

Assignments, or = signs, act like anchors. They stay put, no matter how the code is rearranged.

let and const do not get hoisted, and will throw an error if called before they are declared. const will even throw a syntax error if it is called before it is assigned.