JAVASCRIPT TERMS, TIPS, NOTES

**Lexical Scope**: Scope is how a computer keeps track of all the variables in a program. It refers to the specific environment where a variable is accessible and can be used. JavaScript uses the lexical scoping approach which allows for scopes to be nested and therefore an outer scope encloses (hence closure) an inner scope. There are two types of scope: global scope and local scope. Lexical Scoping defines how variable names are resolved in nested functions: inner functions contain the scope of parent functions even if the parent function has returned. Lexical *this i*s when the *this* in the inner functions have the *this* of the outer functions. Only works with the fat arrow functions. <https://stackoverflow.com/questions/34696686/what-is-lexical-this>

**Closure**:  the combination of a function bundled together (enclosed) with references to its surrounding state (the **lexical environment**). In other words, a closure gives you access to an outer function’s scope from an inner function. In JavaScript, closures are created every time a function is created, at function creation time. Inner functions referring to local variables of its outer function create closures. To use a closure, simply define a function inside another function and expose it. To expose a function, return it or pass it to another function. The inner function will have access to the variables in the outer function scope, even after the outer function has returned.

**How to import and export:**

/------ lib.js ------  
**export** **const** sqrt = **Math**.sqrt;  
**export** **function** **square**(x) {  
 **return** x \* x;  
}  
**export** **function** **diag**(x, y) {  
 **return** sqrt(square(x) + square(y));  
}  
  
//------ main.js ------  
**import** { square, diag } **from** 'lib';  
**console**.log(square(11)); // 121  
**console**.log(diag(4, 3)); // 5

Looping in arrays and objects

for(let hobby of hobbies) //