Cheatography

# JS interview Cheat Sheet

by wkerswell via cheatography.com/27961/cs/8223/

## Closures

Function body has access to variables defined outside its scope

Closure is when a function is able to remember and access its lexical scope even when that function is executing outside its lexical scope.

Useful in callbacks. For example passing a value into an ajax success. Var can be defined before the call and still accessed from the success

## Higher order funtions

Functions that accept other functions as their arguments.

EG .map or .filter

Can help to write code quicker with less bugs due to code reuse

## Recursion

When a function calls its self until it doesn't

An example would be when you have a bunch categories form a DB and you want to map all the children into a tree structure.

## Destructuring

Break and object or array into variables

Great for options objects like ajax options

Can be put into the function declaration params with optional values

## Prototypal Inheritance

Objects inherit directly from other objects.

Instances may be composed from many different source objects, allowing for easy selective inheritance and a flat [[Prototype]] delegation hierarchy.

The tight coupling problem, Inflexible hierarchy problem

The Gorilla/banana problem (What you wanted was a banana, but what you got was a gorilla holding the banana, and the entire jungle)

Delegation

## Factory functions

Factories - Functions that create objects and return them.

Better to use than classes for Composition!

Inheritance encourages you to predict the future of your classes (bad) it will most likely change though out the project

## Composition

Composition is simply when a class is composed of other classes; or to say it another way, an instance of an object has references to instances of other objects.

Is better as we dont have to think of all our classes at the start and when we inevitably need to change them we can with ease

Eg A robot dog needs the bark from the dog class but not the sleep.

## Currying

Currying is when a function, instead of taking all arguments at one time, takes the first one and returns a new function that takes the second one and returns a new function which takes the third one, and so forth, until all arguments have been fulfilled.

The idea is that a function can pass through an application and gradually receive the parameters it needs

```
functi on( 'ar g1' )(' arg 2') ( 'a rg3')
```

## Two types of function

Declaration - `function something (){ }`

Hoisted to the global scope

Expression - `var something = funct ion (){}`

Good to use for passing function into other function. EG the ajax success or a .map

## Objects

Objects can be thought of as the main actors(things) in an application

Every component in JavaScript is an Object, including Functions, Strings, and Numbers

We normally use object literals or constr-uctor functions to create objects.


By **wkerswell**

cheatography.com/wkerswell/



Published 24th May, 2016.

Last updated 16th August, 2019.

## Encapsulation

Refers to enclosing all the functionalities of an object within that object so that the object's internal workings (its methods and properties) are hidden from the rest of the application.

This allows us to abstract or localize specific set of functionalities on objects.

A way to do this would be wrap everything in an Immediately-Invoked Function Expression IIFE - a way to implement the module pattern. Allows private methods and data, defining an API for public use.

## OO Javascript

(OOP) refers to using self-contained pieces of code to develop applications

Building applications with objects allows us to adopt some valuable techniques, namely, Inheritance

## Inheritance

refers to an object being able to inherit methods and properties from a parent object

## Redux/Flux

Uses a Uni-directional data flow to keep a Single source of truth

The state of your whole application is stored in an object tree within a single store.

State is read-only. The only way to change the state is to emit an action, an object describing what happened.

State tree is transformed by actions, written with pure reducers.