

Your Next Week

Saturday March 28	Sunday March 29	Monday March 30	Tuesday March 31
<p>9 AM</p> <ul style="list-style-type: none">— DUE ALL PRE-WORK— DUE Class 04 Code Challenge— DUE Class 04 Lab— DUE Class 05 Reading— Class 05 <p>MIDNIGHT</p> <ul style="list-style-type: none">— DUE Class 05 Learning Journal	<p>MIDNIGHT</p> <ul style="list-style-type: none">— DUE Career: Accountability Partners— DUE Class 04-05 Feedback		<p>6:30 PM</p> <ul style="list-style-type: none">— DUE Class 05 Lab— DUE Class 06 Reading— Class 06A
Wednesday April 01	Thursday April 02	Friday April 03	Saturday April 04
<p>6:30 PM</p> <ul style="list-style-type: none">— Class 06B <p>MIDNIGHT</p> <ul style="list-style-type: none">— DUE Class 06 Learning Journal	<p>6:30 PM</p> <ul style="list-style-type: none">— Co-working		<p>9 AM</p> <ul style="list-style-type: none">— DUE Class 06 Code Challenge— DUE Class 06 Lab— DUE Class 07 Reading— Class 07 <p>MIDNIGHT</p> <ul style="list-style-type: none">— DUE Class 07 Learning Journal

Class 05

DSA: Linked Lists + Module 1 Review

seattle-javascript-401n16

Lab 04 Review

Code Challenge 04

Review

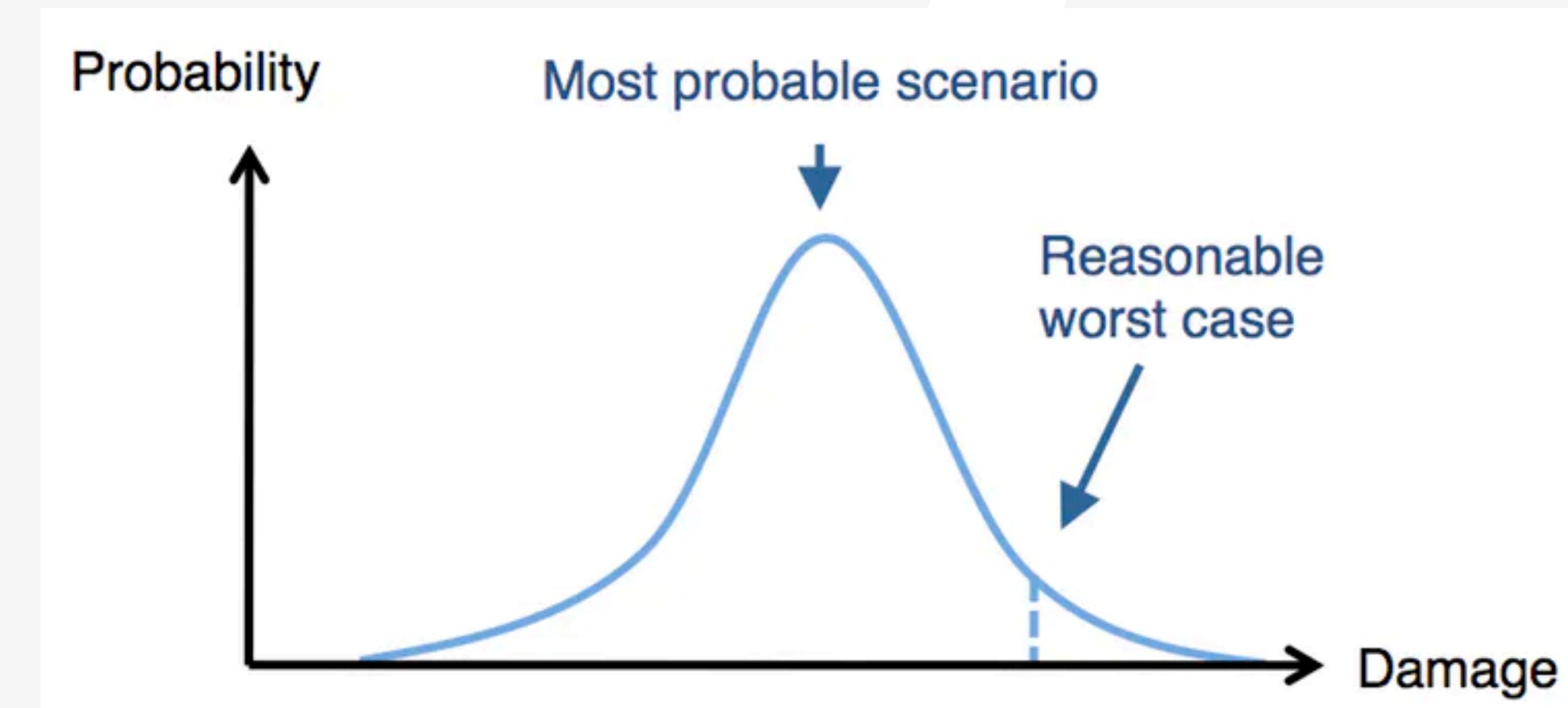
DSA??

- DSA = Data Structures and Algorithms
- **Data structures** define how we store data for easy access, modification, and more!
- **Algorithms** define the most efficient function for doing some process
- Best/most efficient is determined by **Big-O**



Worst Case Says What's Best

- **Big O notation** tells us the worst case scenario
 - “O” = **order/growth rate** of a function
 - What is the longest something will run
 - What is the most space something will take up
- We want to minimize our Big O
 - That's how we know what's efficient



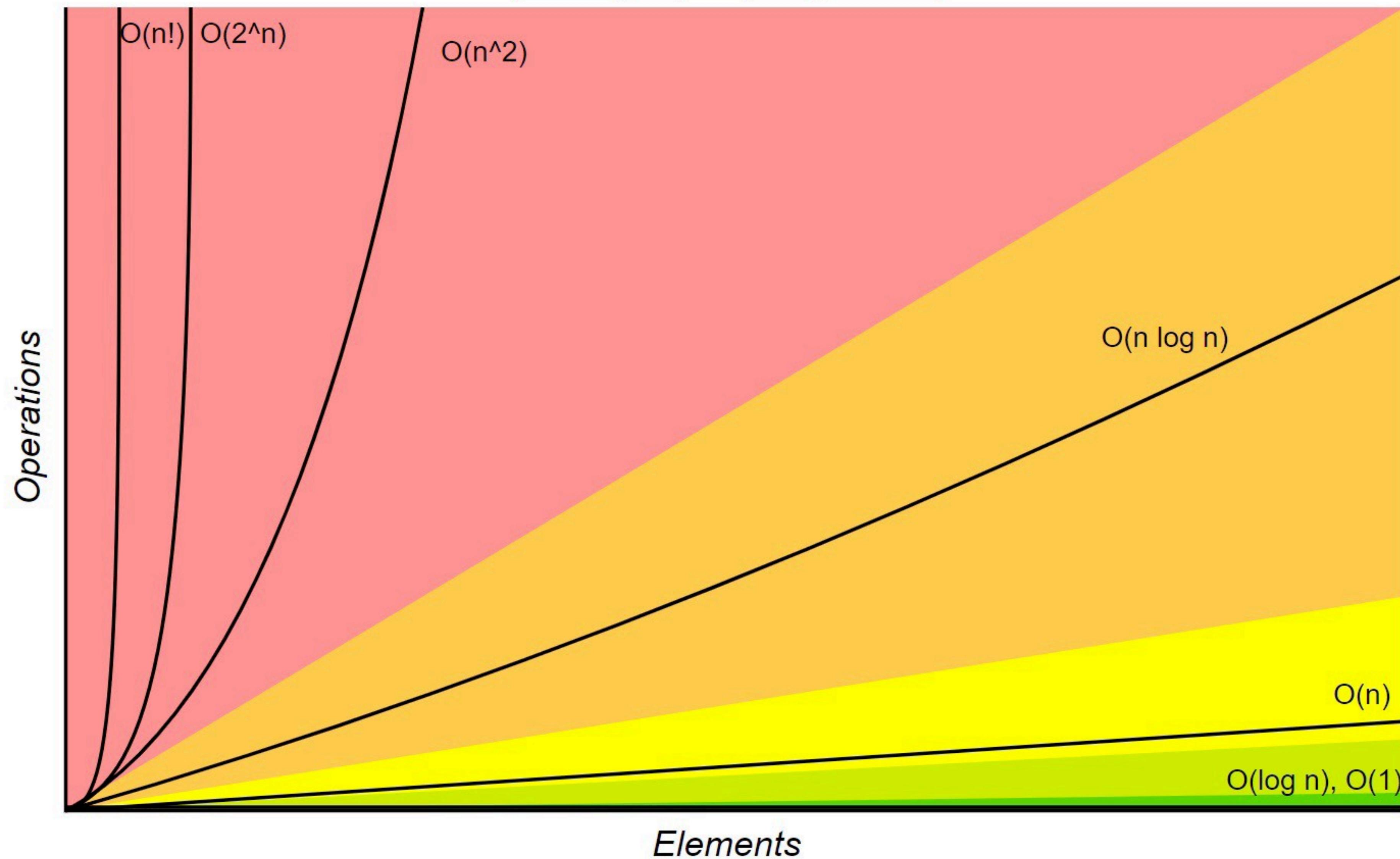
Big-O Is Based On Input Size

- If you had to iterate through an array, how “long” would it take?
 - Array of size 5 >> 5 commands
 - Array of size 100 >> 100 commands
 - Array of size **n** >> n commands
- The size of the input determines the worst case time and space
- $O(n)$ >> if there are **n** items, it will take **n** time or **n** space

$O(1)$		Speed doesn't depend on the size of the dataset
$O(\log n)$		10x the data means 2x more time
$O(n)$		10x the data means 10x more time
$O(n \log n)$		10x the data means about 20x more time
$O(n^2)$		10x the data take 100x more time
$O(2^n)$		The dilithium crystals are breaking up!

Big-O Complexity Chart

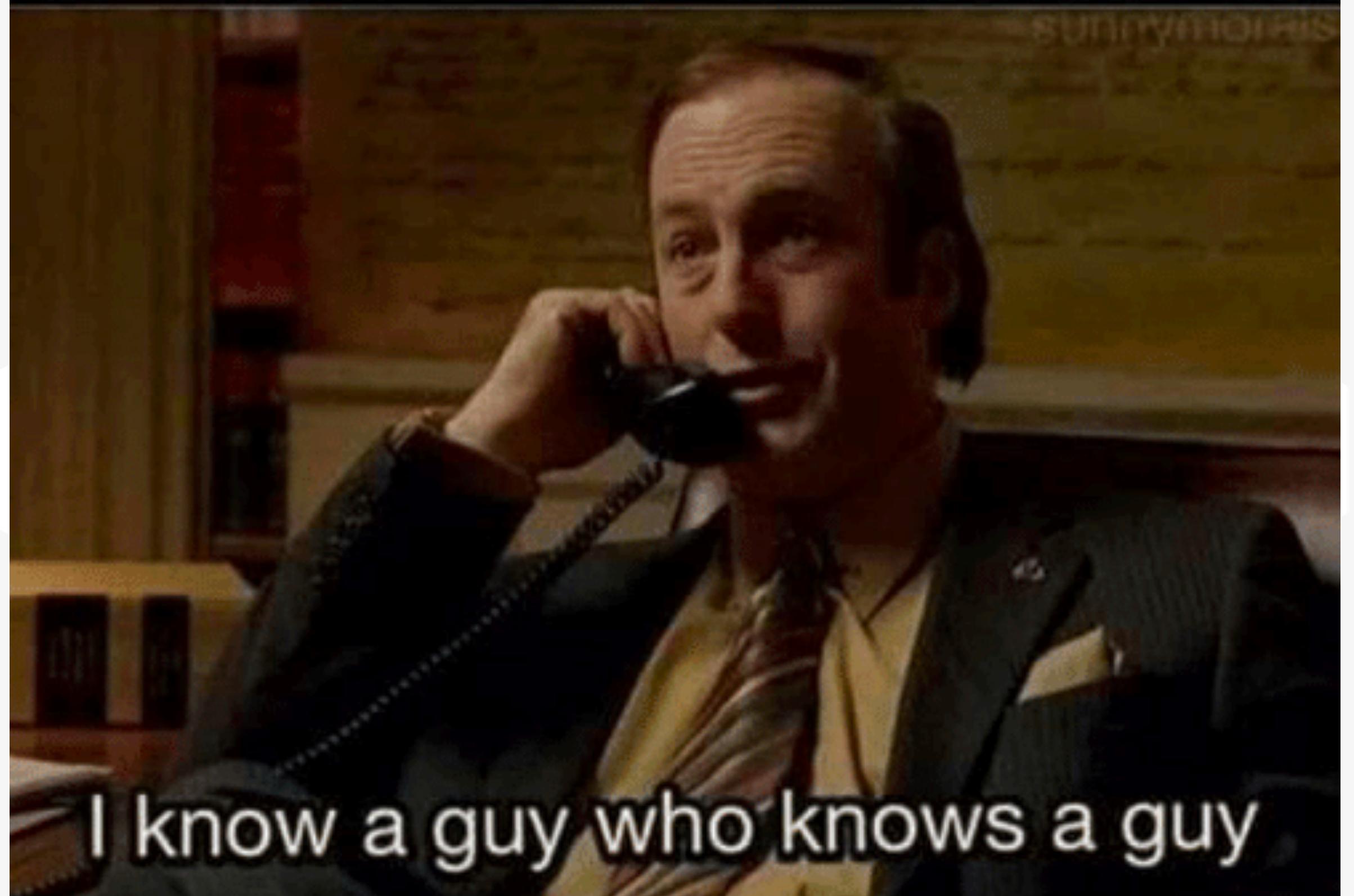
Horrible Bad Fair Good Excellent



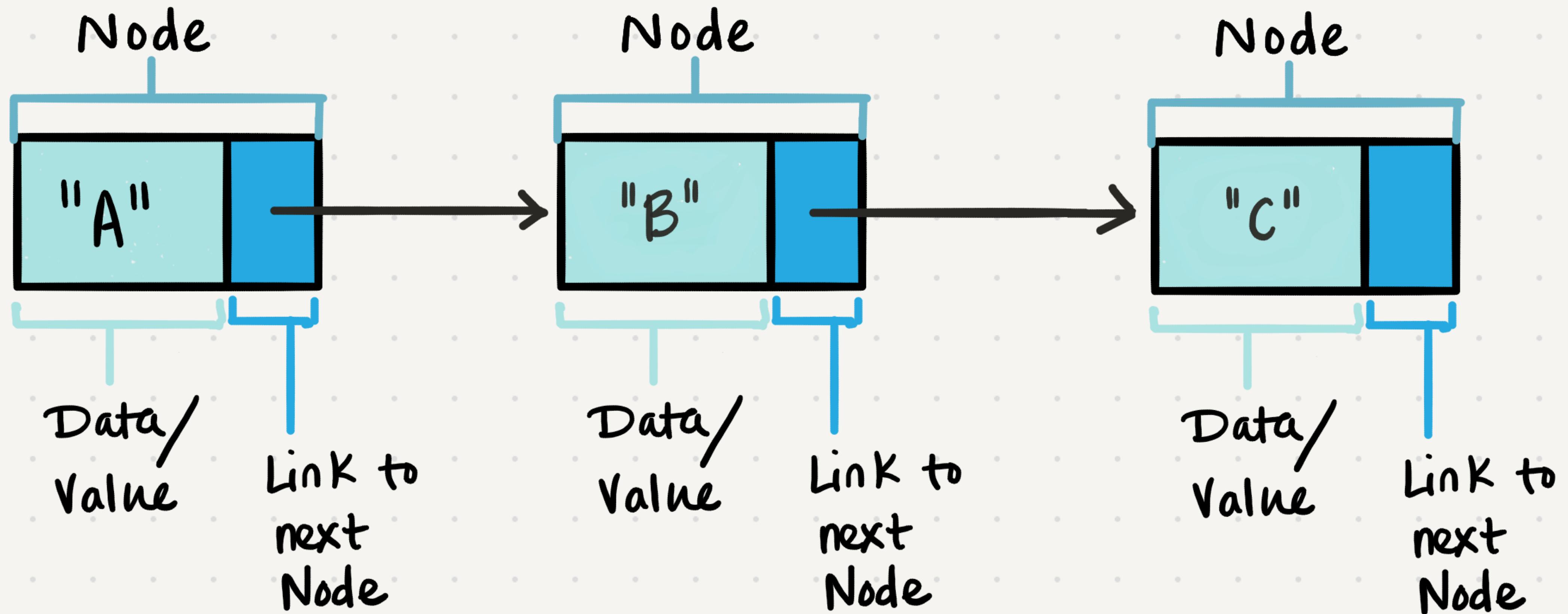
Linked Lists

- Arrays are very rigid in other languages
- Linked Lists are more old-school and flexible
 - A collection of independent **Nodes**
 - Directions of how to get to the next Node
 - No “hopping around” like in arrays

Linked List data structures be like:

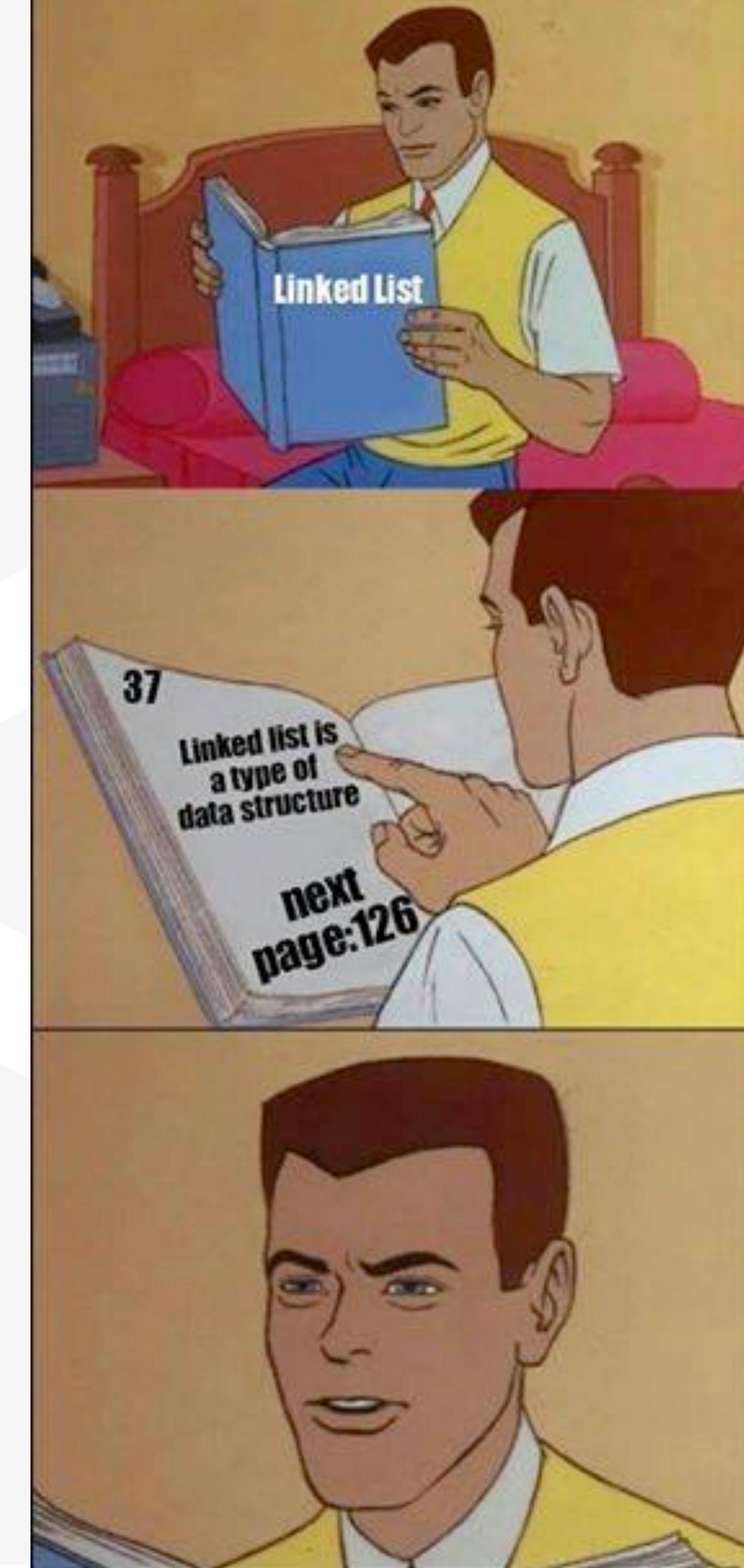


I know a guy who knows a guy



Singly Linked List

- Every Node has two properties:
 - value
 - **next** Node
- Every Singly Linked List keeps track of only the start of the list, called the **head**
- No way to go backwards from a Node, always moving until **next = null**

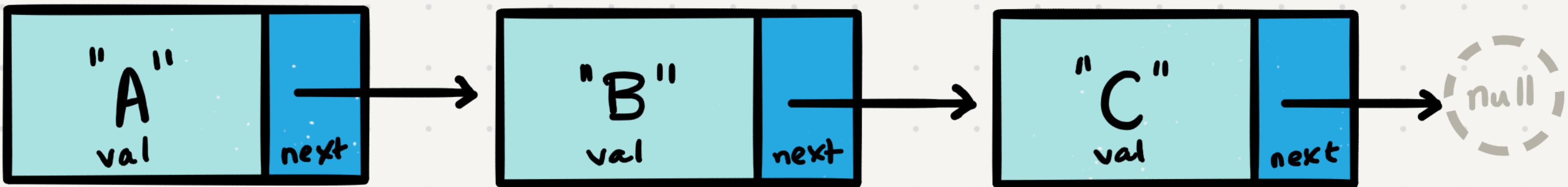


Doubly Linked List

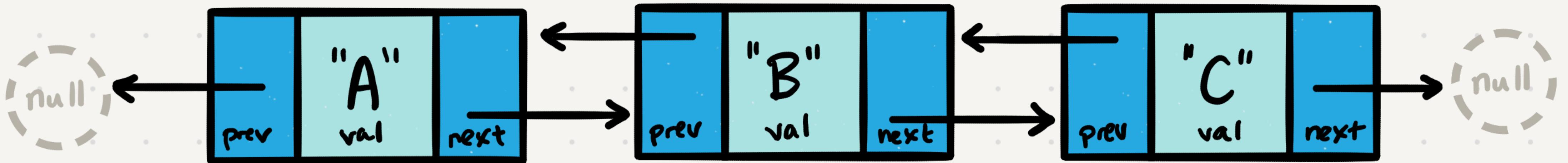
- Every Node has three properties:
 - value
 - **next** Node
 - **prev** Node
- Every Doubly Linked List keeps track of only the start of the list, called the **head**
- You can move forwards or backwards until either **prev=null** or **next = null**



SINGLY LINKED



DOUBLY LINKED



Circular Linked List

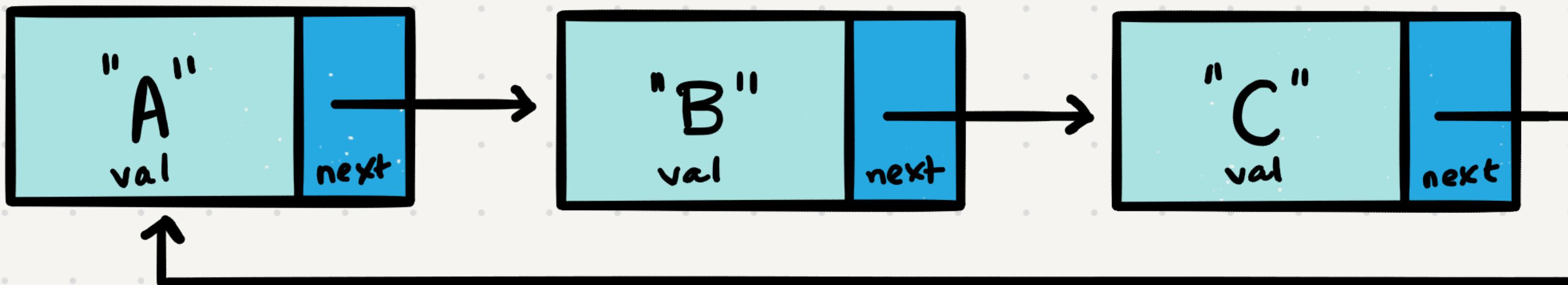
- Just an extension of either a singly or doubly linked list
- One Node points back to a previous Node in the list
 - Creates a cycle
 - You can end up looping the list forever!
 - How could you detect this using code?

Absolutely no one:

Circular linked lists:



CIRCULARLY SINGLY LINKED



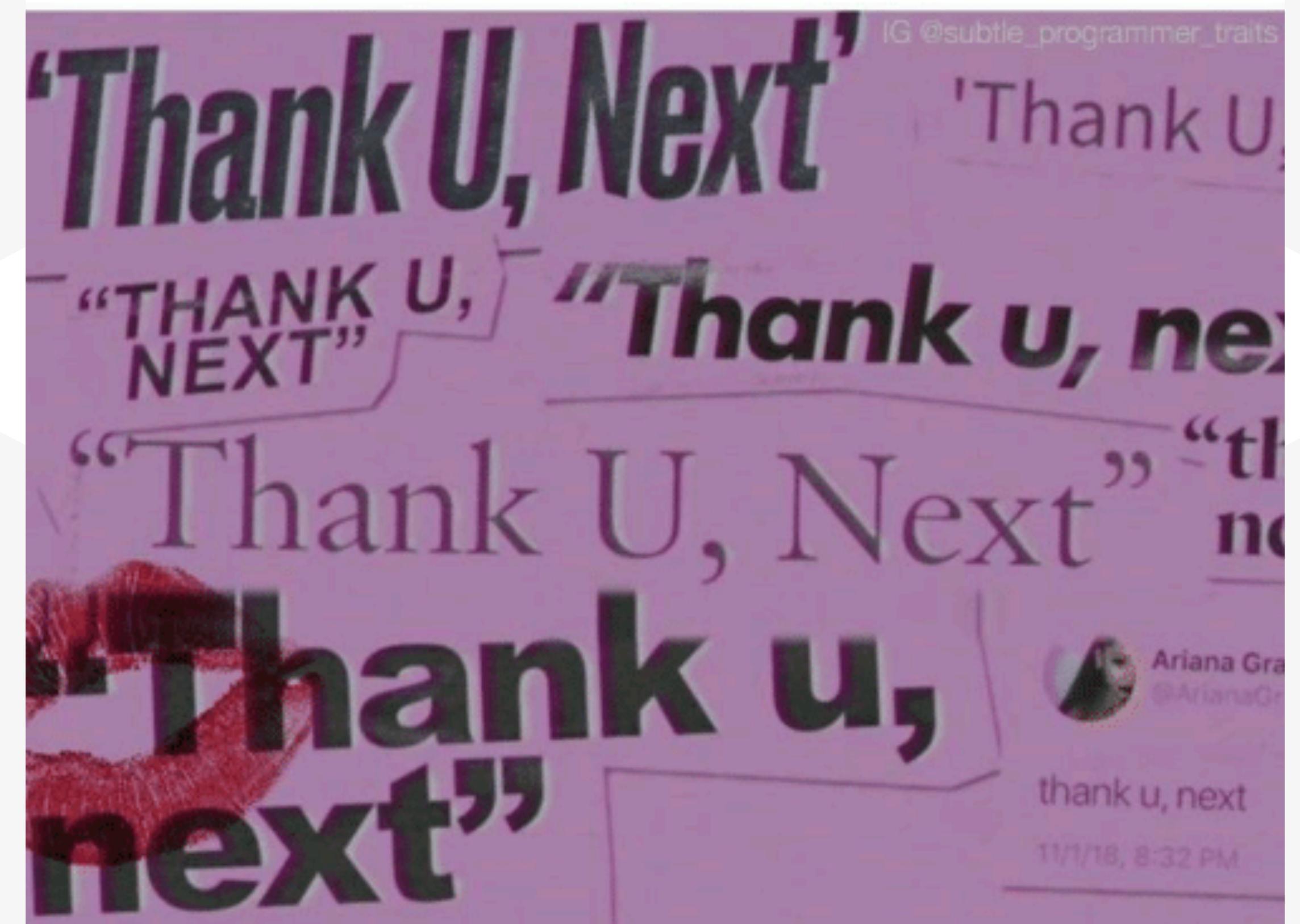
CIRCULARLY DOUBLY LINKED



Big O and Linked Lists

- How long does it take to search for something?
 - What you're looking for could be the last list item (in the worst case)
 - You'd have to visit each Node
 - If there are n Nodes
then.... $O(n)$

when the value in your linked list node isn't the one you're searching for



What About Changing the List?

- To insert something in a Linked List, you either
 - Insert at the beginning
 - Since you know the head, this is an instant command
 - No looping or traversal, so $O(1)$
 - Size of the list doesn't matter!
 - Insert somewhere else
 - Might be the end, in which case $O(n)$

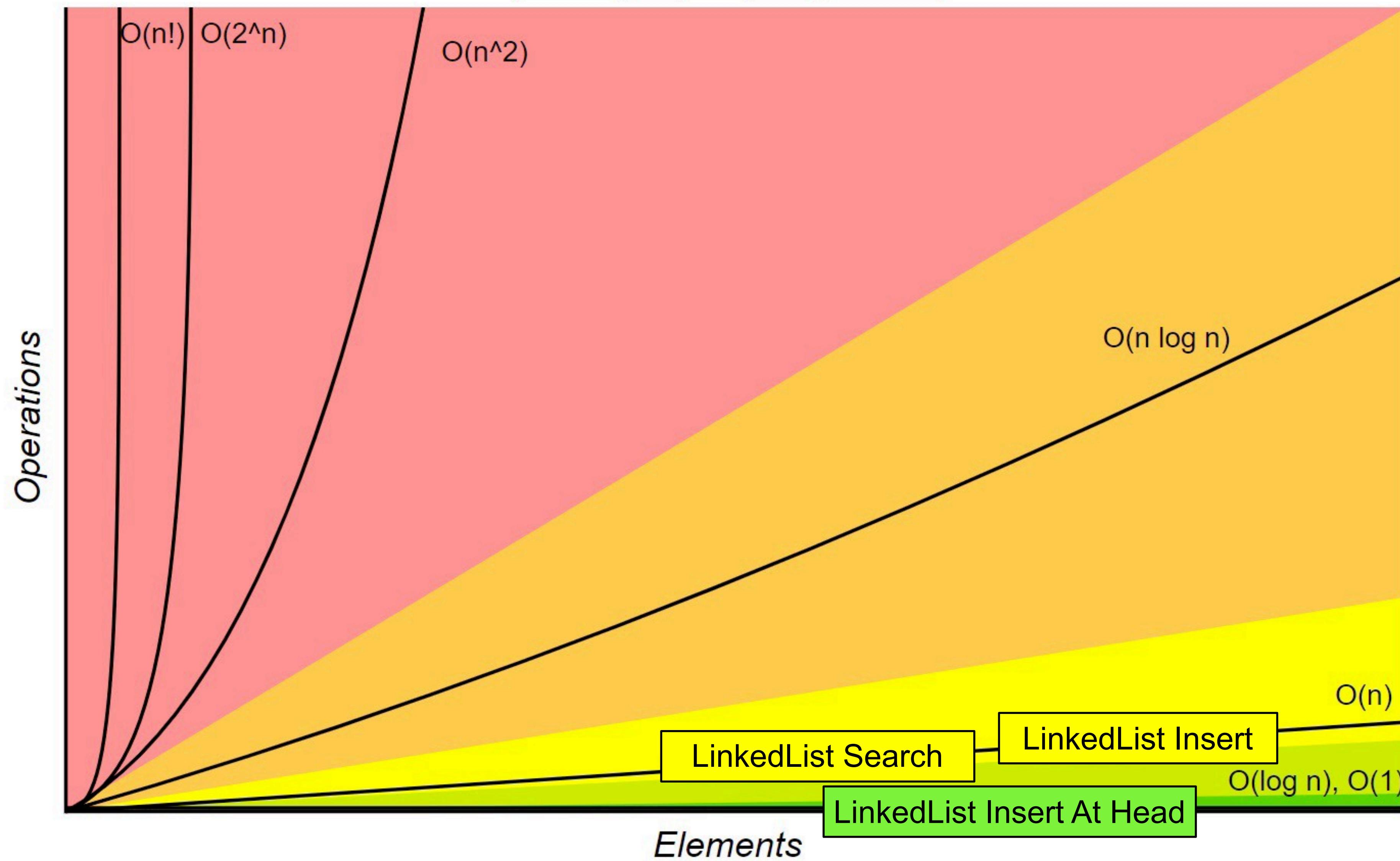
Function: Removes first item from a linked list.

Second Item:



Big-O Complexity Chart

Horrible Bad Fair Good Excellent



Vocab Review

ecosystem



ecosystem

The collection of tools you're using to develop and run your code. Ecosystems (also referred to as environments) can include the type of compiler, IDE, language and infrastructure you're using as a developer for your application.

Node.js



Node.js

An open-source JavaScript runtime environment which allows to run JavaScript code outside of a browser.

Node.js (often called Node) also enables developers to write JavaScript server-side code. Node has unified web development such that JavaScript can be used in every part of application development.

V8 Engine



V8 Engine

Google's high performant open-source program that runs JavaScript code. The V8 Engine is what powers Chrome web browsers.

module



module

A module is a collection of functions or JavaScript code that is defined in a single JavaScript file and then exported for later use. Application code imports various modules and calls the functions or code defined within those modules.

package



package

A file or directory that is described by a `package.json` file.

This `package.json` lists which modules the file/directory needs to install, how to run the application located in the file/directory, and any scripts for development use. Packages usually contain a module(s), and are formatted for easy sharing across multiple developers.

node package manager (npm)

node package manager (npm)

The node package manager allows us to
install packages from external developers and
maintain package versions and updates.

test driven development (TDD)

test driven development (TDD)

A development process where tests for new code is written *before* the code. TDD encourages developers to plan their code and their code's test cases thoroughly before developing. This reduces errors during the development process, and makes it simpler to refine and refactor code.

Jest



Jest

Jest is a JavaScript testing framework that makes it very simple for developers to write and run unit tests for the JavaScript code. Jest relies on the idea of “expecting” an outcome from a code operation. Jest also has the ability to show how much of an application’s code is covered by unit tests.

continuous integration (CI)



continuous integration (CI)

This is a development practice where all code changes go through an automatic process of re-running tests, and then merging new code with existing tested code if no tests are broken. CI helps teams with multiple developers unify their codebase frequently.

unit test



unit test

A unit test is a small, modular test that covers one “unit” of code in an application. The definition of “unit” can be arbitrary or vary from team to team, but it generally refers to independent functions

functional programming



functional programming

A set of guiding principles for software development, where more thought is given to the input and output of functions.

The goal of functional programming is that calling a function anywhere with the same input parameters results in the same output.

pure function



pure function

A pure function is a function where given the same inputs, it always returns the same outputs and has no side effects on the application environment. An example of a side effect is logging to the console.

higher order function



higher order function

A higher order function is any function which takes another function as an argument and/or returns a function. They are often used to abstract or isolate actions within an overarching action.

immutable state



immutable state

Immutable means to not change, and the state of an application usually refers to all of the data an application is maintaining at a certain point in time. Functions which maintain the paradigm of immutable state do not change anything in the application during or after running.

object



object

An object can be a variable, data structure, function or method. Objects, also called *instances*, are a piece of data recorded in memory and accessible by an identifier (such as the variable name or function name).

object-oriented programming (OOP)

object-oriented programming (OOP)

A set of guiding principles where an application is thought of as a collection of objects interacting with one another. The creation of classes helps to solidify complex data collections as a single object.

class



class

A blueprint for an object, specifying how that object should be created, what initial data values it has, and what actions can be done upon this object through class methods. You can think of writing a class as similar to creating a new data type.

prototype



prototype

A prototype is the blueprint of an object.

In JavaScript, everything is based on a generic “Object” class, which is not directly editable by developers. We can change this class blueprint by accessing the prototype of any instance, thereby faking a class specification.

super



super

The `super` keyword refers to the parent class of the current class you're developing. By calling `super()`, you essentially call the parent class constructor.

inheritance



inheritance

The process of basing an object or class upon an existing “parent” object or class.

This allows the “child” to acquire all the variables and methods specified by the “parent”, making code more DRY.

constructor



constructor

A constructor is a function that defines how an object should be created.

Because all JavaScript objects innately inherit from a generic “object” class, we usually don’t have to define much in a constructor other than the initial values of any custom variables.

instance



instance

An instance is a single object created based on a class or data type's specifications. Instances only exist while the application is running, and in our application code we provide variable name references for them so we can instruct how that instance is used.

context



context

A term used to represent the current scope or environment you're operating in.

Any data, variables or functions available to you in that point in time are all within your current context.

this

this

A keyword used to refer to the current context; the current object, class, or function you're in. The `this` keyword is usually very helpful in blueprint-type code to refer to itself before any instances of it have been created. Each instance can then interpret `this` to mean themselves.

database



database

An organized set of data stored long-term
in an accessible location.

data model



data model

An abstract set of rules or guidelines that categorizes data in an application. Data models define what data should look like, how data relates to one another, and how to access data.

CRUD



CRUD

Create, Read, Update and Delete are the four basic functions for storing and accessing stored data. We use this acronym to keep in mind how we most commonly interact with a database.

schema



schema

A schema is a collection of rules that describes a type of data within our larger data model. Schemas typically define what properties a piece of data should have.

sanitize



sanitize

The process of “cleaning” data to remove any illegal characters, to standardize naming conventions, and to enforce type upon data values.

Structured Query Language (SQL)

Structured Query Language (SQL)

A standard language for interacting with relational databases. SQL relies on “data queries” in the format of “SELECT __ FROM __ WHERE”.

Non SQL (NoSQL)



Non SQL (NoSQL)

Non SQL refers to databases that are non-relational (do not enforce a relation table) and thus do not need to use the SQL language. NoSQL databases can choose what language to use, and they expose custom APIs for communicating with the database within an application.

MongoDB



MongoDB

A cross-platform document-based database that is non-relational. MongoDB uses JSON-like documents that adhere to a certain schema, and arranges these documents into collections.

Mongoose

Mongoose

A package / library for using MongoDB in a Node.js application. Mongoose provides a large suite of features, such as schema validation, API connection to a MongoDB database, and translation between JavaScript code and MongoDB code.

document

document

In MongoDB, each piece of data is stored as a binary json (`.bson`) file on the database server (which can be your local computer if you’re running MongoDB locally!). Because each piece of data is stored as a file, data entries are referred to as “documents”.

record



record

A more generic term for a data entry in a database. For MongoDB, documents and records are synonymous.

Object Relational Mapping (ORM)

Object Relational Mapping (ORM)

A programming technique of converting
data between two incompatible systems.

For example, Mongoose converts
JavaScript data into terms MongoDB can
understand, and vice-versa.

mock



mock

A mock allows you to skip testing some piece of your code while still maintaining a roughly similar input/output path. This way, you can skip calling certain functions and still test your application flow. And by skipping some functions, you can save on time and prevent redundant or unnecessary tests.

middleware



middleware

Code that is used to bridge the gap between applications or tools. Often described as “software glue”, middleware makes it much simpler for two differing systems to communicate with one another. Broadly, middleware can be described as any code running between two other processes.

supergoose



supergoose

A package developed by Code Fellows,
supergoose is used to help set up a mock
database for your tests. This can
otherwise be tedious to do on your own,
and many companies use similar
packages to spin up a fake database for
testing.

pre hook



pre hook

With Mongoose, you can write your own middleware that runs between a Mongoose command being called, and the MongoDB receiving that command. To write this middleware, Mongoose exposes a ‘`pre`’ function, often called the pre hook since it hooks your middleware code up to the correct part of the process.

post hook



post hook

With Mongoose, you can write your own middleware that runs between a MongoDB command being called, and the response to Mongoose on your application. To write this middleware, Mongoose exposes a ‘`post`’ function, often called the post hook since it hooks your middleware code up to the correct part of the process.

in-memory



in-memory

When something is in-memory, it is only accessible while a program is running.

Any in-memory data is not persisted after the application ends.

interface



interface

Like middleware, an interface is a shared space where two systems can exchange information. Interfaces allow two unrelated entities to interact. Interfaces receive a certain kind of input and translate that into the commands needed to get the expected output.

Linked List



Linked List

Like middleware, an interface is a shared space where two systems can exchange information. Interfaces allow two unrelated entities to interact. Interfaces receive a certain kind of input and translate that into the commands needed to get the expected output.

Singly Linked List



Singly Linked List

Like middleware, an interface is a shared space where two systems can exchange information. Interfaces allow two unrelated entities to interact. Interfaces receive a certain kind of input and translate that into the commands needed to get the expected output.

Doubly Linked List



Doubly Linked List

Like middleware, an interface is a shared space where two systems can exchange information. Interfaces allow two unrelated entities to interact. Interfaces receive a certain kind of input and translate that into the commands needed to get the expected output.

Circularly Linked List



Circularly Linked List

Like middleware, an interface is a shared space where two systems can exchange information. Interfaces allow two unrelated entities to interact. Interfaces receive a certain kind of input and translate that into the commands needed to get the expected output.

Big-O



Big-O

Like middleware, an interface is a shared space where two systems can exchange information. Interfaces allow two unrelated entities to interact. Interfaces receive a certain kind of input and translate that into the commands needed to get the expected output.

Node



Node

Like middleware, an interface is a shared space where two systems can exchange information. Interfaces allow two unrelated entities to interact. Interfaces receive a certain kind of input and translate that into the commands needed to get the expected output.

head



head

Like middleware, an interface is a shared space where two systems can exchange information. Interfaces allow two unrelated entities to interact. Interfaces receive a certain kind of input and translate that into the commands needed to get the expected output.

.next



.next

Like middleware, an interface is a shared space where two systems can exchange information. Interfaces allow two unrelated entities to interact. Interfaces receive a certain kind of input and translate that into the commands needed to get the expected output.

.prev



.prev

Like middleware, an interface is a shared space where two systems can exchange information. Interfaces allow two unrelated entities to interact. Interfaces receive a certain kind of input and translate that into the commands needed to get the expected output.

Lab 05 Overview

Code Challenge 05

Overview