Networking

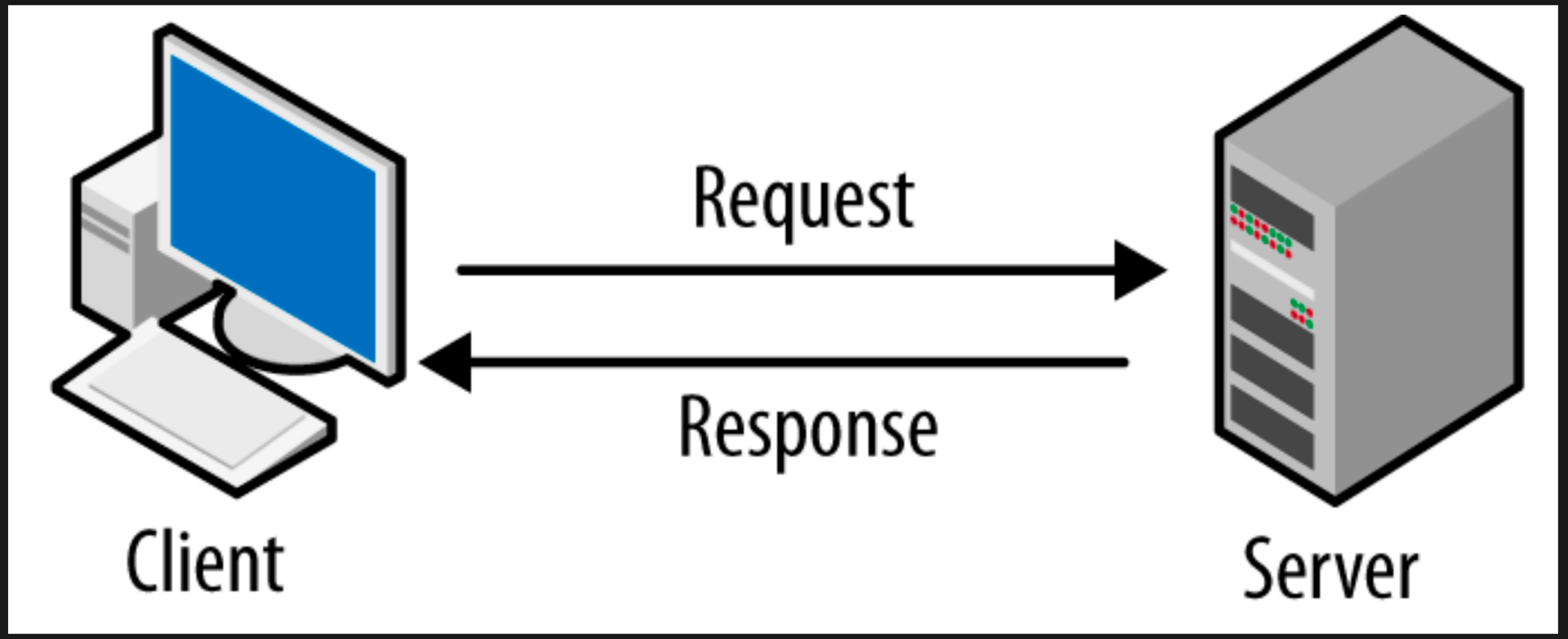
-> Lecture 03

@beknar

El Plan

- Client-Server Architecture // How (almost) all apps on the Internet work?
- REST API // what is API, what is REST, what is HTTP, what is JSON
- NSURLSession // Networking in iOS
- Dependency Management (SPM) // and alternatives
- Alamofire // abstraction over NSURLSession
- Moya // abstraction over Alamofire over NSURLSession

-> Client-Server Architecture


Request
Response
Client
Server

- Client is like the client from real-world, the one who is using some service. Usually referred to as frontend or client-side.
- Server is the one who is serving (giving) the requested stuff. Usually referred to as backend or server-side.
- Server is just a computer in the cloud.
- When client asks from the server it is called request.
- When server gives back to the client it is called response.
- Common types of the clients are: web apps and mobile apps.

Communication between Client & Server

Communication is done by requests and responses:

1. A client sends a request to the server
2. A server receives the request
3. The server runs an application to process the request
4. The server returns an response (output) to the client
5. The client receives the response

-> REST API

What is API?

- API is like a Swift's `protocol`. It describes what some stuff can do, or what can you ask it to do.
- In terms of web services API is used to refer to the list of stuff you can request it to do.
  - Each such thing that you ask API to do – is called endpoint.
- API stands for Application Programming Interface (interface – like Java interface is the same as Swift protocol)

What is REST?

- What is REST?
  - It stands for REpresentational State Transfer.
- What you really need to know for now – is that it is an architectural pattern on API side (backend), and you mostly shouldn’t care.
- Because most of the APIs that you are going to use are REST APIs.
- But good to know that there other ways to build APIs: JSON-RPC, SOAP, GraphQL-based & etc.

What is HTTP?

- Hypertext Transfer Protocol (HTTP) is the foundation of the World Wide Web.
- It is used to transfer information between devices on the network.
- A typical flow over HTTP involves a client making request to a server, which then sends response message.

What's in an HTTP request?

- URL – where?
- HTTP method – how?
- (Optional) HTTP headers – prerequisite information
- (Optional) HTTP body – main message information

What's in an HTTP request method?

- It describes the action that HTTP request is expecting from the server.
- Two of the most common (and widely used) ones are:
  - GET – expects information back in return (e.g. website data)
  - POST – client is submitting information to the web server (such as submitting form information, e.g. a submitted username and password)

What’s in an HTTP request headers?

- HTTP headers contain text information stored in key-value (like dictionary or map) format.
- These headers communicate some core information, like what browser/operating system is being used, what is the language & etc.

What’s in an HTTP request body?

- The body of a request is the part contains the body of information the request is transferring.
- The body of an HTTP contains any information that is being sent to the server, such as username and password in case of authorization form.

What's in an HTTP response?

- HTTP status code
- HTTP response headers
- (Optional) HTTP body

What's an HTTP response status code?

- HTTP status codes are 3-digit codes most often used to indicate whether an HTTP request has been successfully completed.
- Status codes are broken into the following 5 blocks:
  - 1xx Informational
  - 2xx Success
  - 3xx Redirection
  - 4xx Client Error
  - 5xx Server Error
  - "xx" refers to different numbers between 00 and 99

What is JSON?

- JSON stands for JavaScript Object Notation
- JSON is a lightweight format for storing and transporting data
- JSON is “self-describing” and easy to understand
- It is very widely used to pass data between clients and servers.
- JSON Syntax Rules
  - Data is in key/value pairs
  - Data is separated by commas
  - Curly braces hold objects
  - Square brackets hold arrays

-> URLSession

URLSession is a class in iOS that provides an API for making network requests and handling network-related tasks, such as sending HTTP requests and receiving responses. It allows you to interact with web services, download data, upload files, and perform other network operations

-> Let’s code

-> Dependency Management

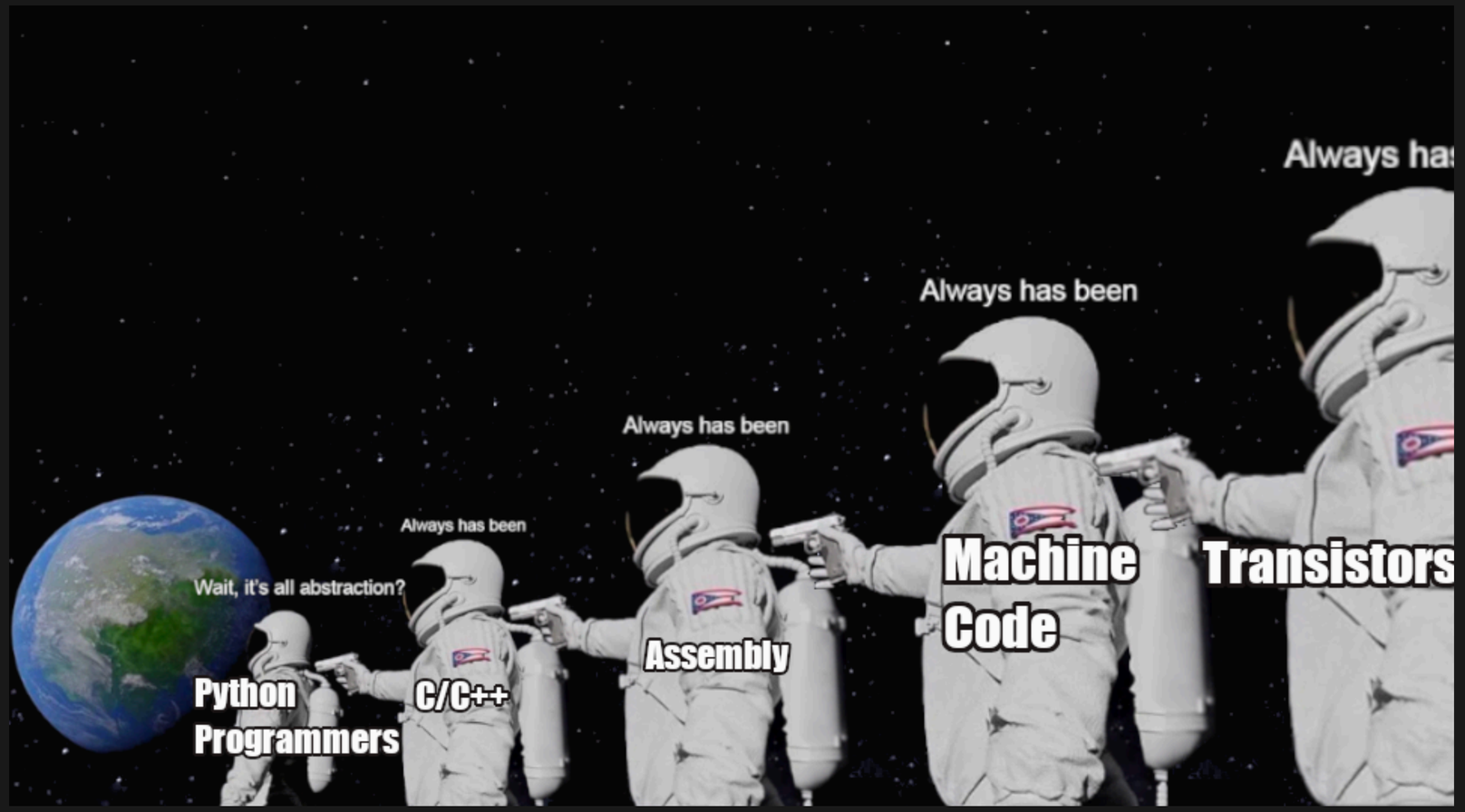
- It is a process of managing external libraries, frameworks, and dependencies that are used in an iOS application.
- It involves handling the inclusion, versioning, and resolution of these dependencies to ensure that the application has access to the required functionality and external code.
- Most popular dependency management tools: CocoaPods, SPM, Carthage

CocoaPods

- Uses a centralized repository of libraries and frameworks
- Developers specify their dependencies in a Podfile, which defines the libraries and their versions required for the project
- CocoaPods handles downloading, installation, integration, conflict resolution of the specified dependencies into the Xcode project

SPM

- Official dependency management tool provided by Apple
- SPM is lightweight and provides tight integration with Swift and Xcode, making it convenient for managing dependencies in pure Swift projects
- SPM uses a Package.swift manifest file to define the package dependencies and their versions
- Not all libraries available in CocoaPods are available in SPM

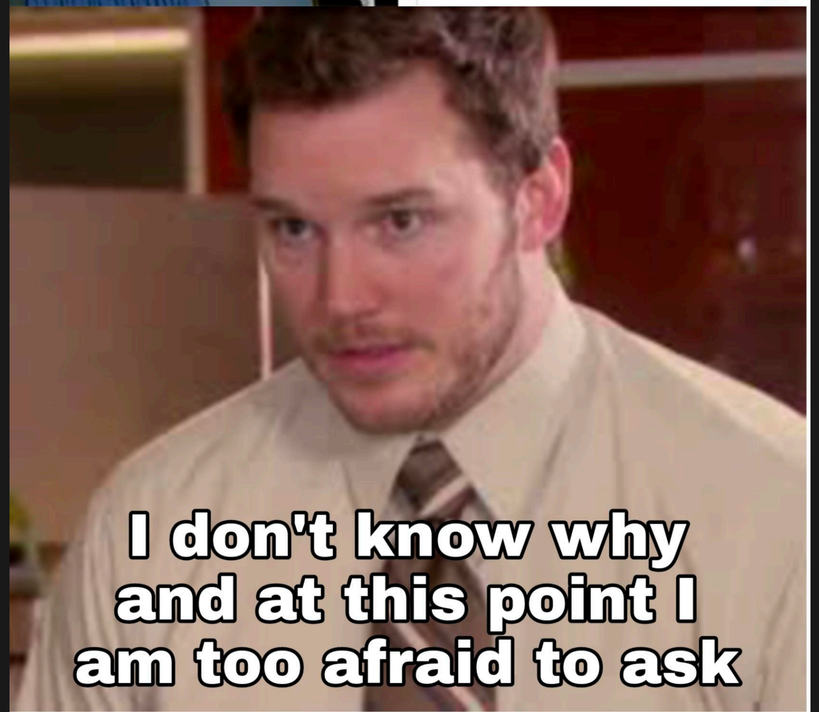

Always has
Always has been
Always has been
Always has been
Wait, it's all abstraction?
Python Programmers
C/C++
Assembly
Machine Code
Transistors

```
public int x;
```

```
private int x;

public int getX()
{
   return x;
}

public void setX(int value)
{
   x = value;
}
```

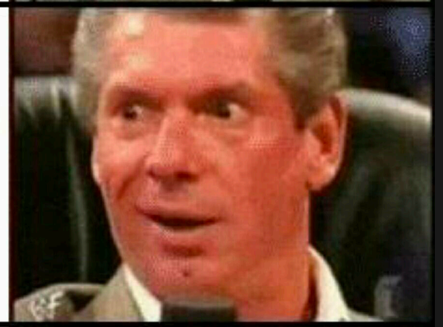


¡n!

```
int ans = 1+1;
```

```
int ans = AddOnes();

public int AddOnes() {
    return 1+1;
}
```

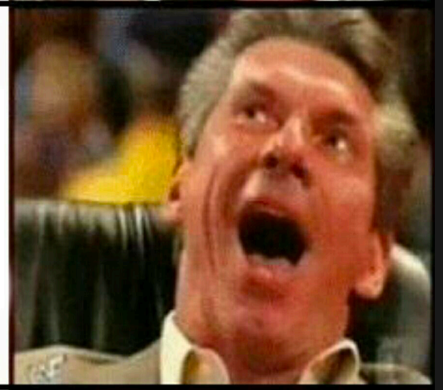


```
int ans = AddOneObjects(new OneObject(), new OneObject());

public int AddOneObjects(OneObject one1, OneObject one2) {
    return one1.Value + one2.Value;
}

public class OneObject {
    public int Value {get; set;}

    public OneObject() {
        this.Value = 1;
    }
}
```



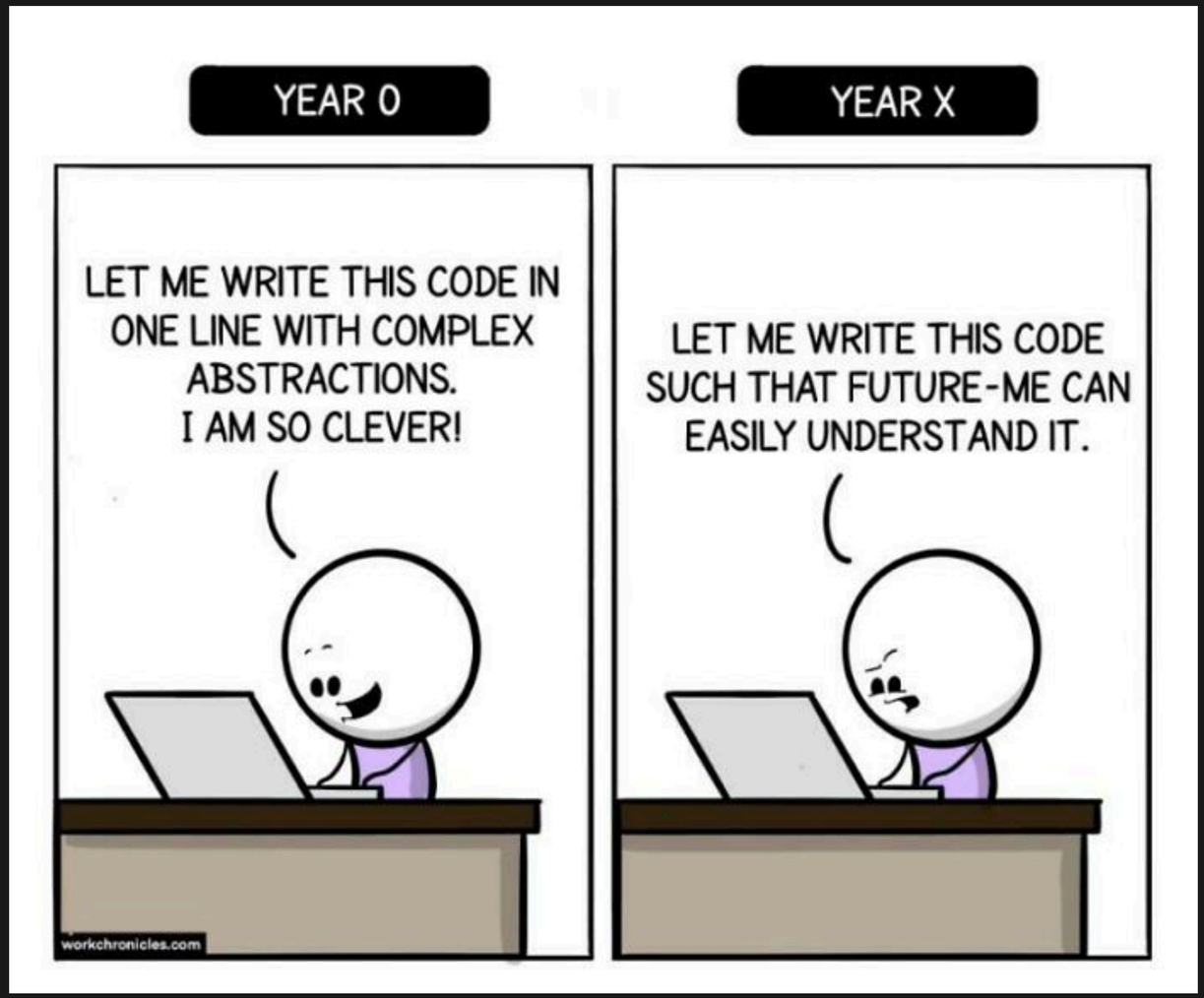
```
int ans = new OneObjectAdderHandler(
    new OneObjectAdderCommand(new OneObjectDTO() {
        One1 = new OneObject(),
        One2 = new OneObject()
    })
).Handle();

public class OneObjectAdderCommand {
    public OneObjectDTO dto {private get; set;}
    public OneObjectAdderCommand(OneObjectDTO dto) {
        this.dto = dto;
    }
}

public class OneObjectDTO {
    public OneObject One1 {get; set;}
    public OneObject One2 {get; set;}
}

public class OneObjectAdderHandler {
    public OneObjectAdderCommand command {private get; set;}
    public OneObjectAdderHandler(OneObjectAdderCommand command) {
        this.command = command;
    }
    public int Handle() {
        return this.command.dto.One1.Value + this.command.dto.One2.Value;
    }
}
```

¡n!


YEAR 0
YEAR X
LET ME WRITE THIS CODE IN ONE LINE WITH COMPLEX ABSTRACTIONS. I AM SO CLEVER!
LET ME WRITE THIS CODE SUCH THAT FUTURE-ME CAN EASILY UNDERSTAND IT.
workchronicles.com

-> Alamofire

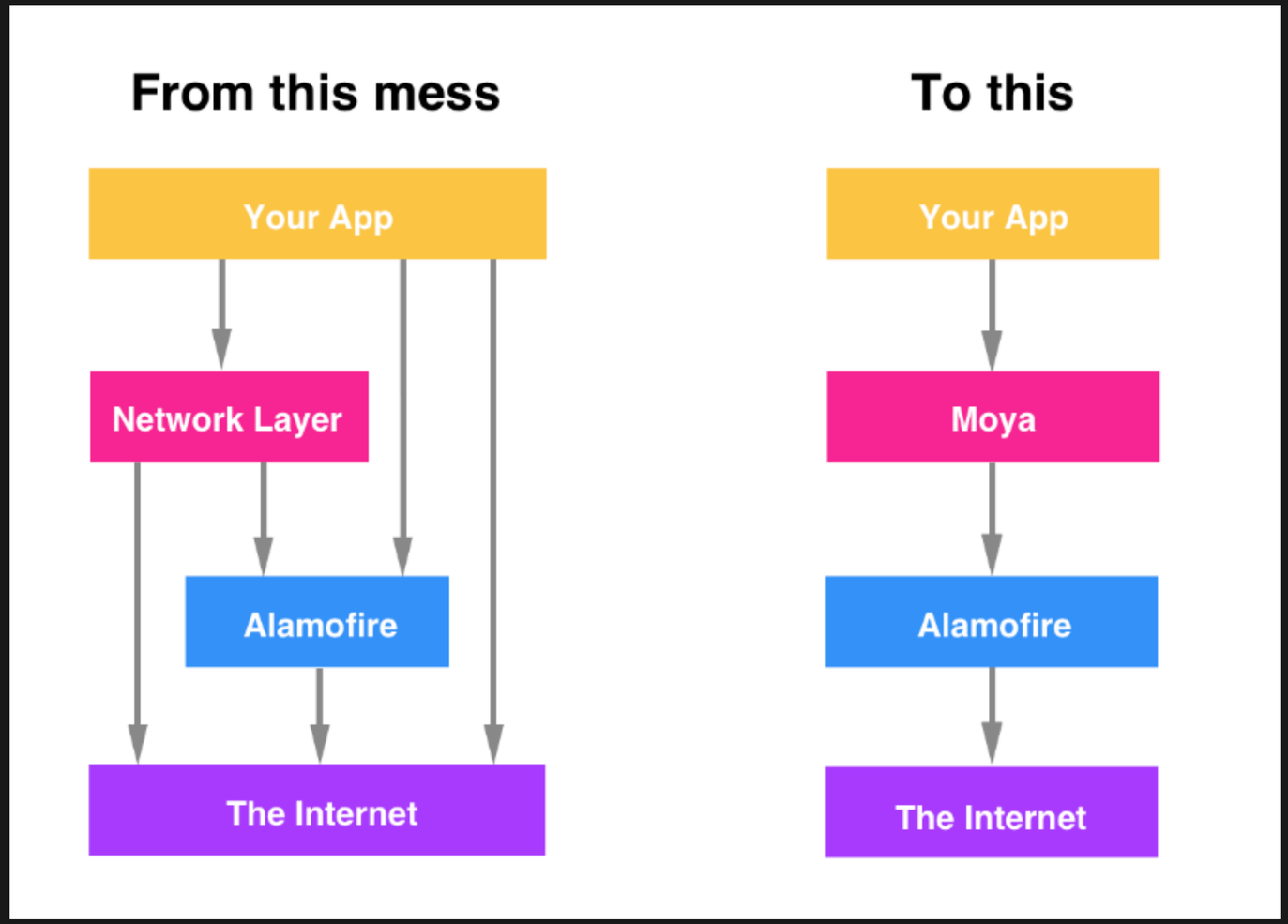
Alamofire is a popular third-party networking library for iOS, written in Swift. It simplifies the process of making network requests, handling responses, and dealing with various networking tasks. It provides a higher-level abstraction over URLSession, offering a more concise and intuitive API for networking operations.


```
AF.request("https://api.example.com/data").responseJSON { response in
    switch response.result {
    case .success(let value):
        // Process the response data
        print("Response: \(value)")
    case .failure(let error):
        print("Error: \(error.localizedDescription)")
    }
}
```

-> Moya

You're a smart developer. You probably use Alamofire to abstract away access to URLSession and all those nasty details you don't really care about. But then, like lots of smart developers, you write ad hoc network abstraction layers. They are probably called "APIManager" or "NetworkModel", and they always end in tears.

¡n!


From this mess
Your App
Network Layer
Alamofire
The Internet
To this
Your App
Moya
Alamofire
The Internet

So the basic idea of Moya is that we want some network abstraction layer that sufficiently encapsulates actually calling Alamofire directly.

-> Let’s code

Q/A

Rakhmet 💚