
# lecture 6

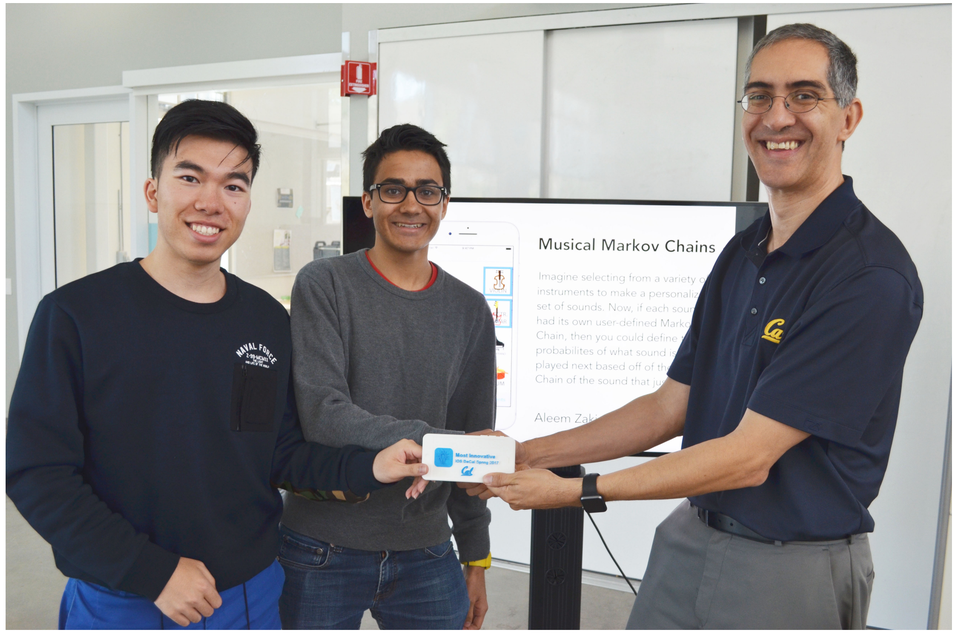
## Closures, Networking, CocoaPods

cs198-001 : spring 2018

# announcements

- no new lab this week - continuation of Pokedex lab + custom app workday (attendance still required)
  - Pokedex due **Wednesday at 11:59 pm**
  - submit on Gradescope ONLY if you do not finish during lab
- hw3 pt 1 (snapchat clone) **due tonight**
- custom app specification released tonight
  - Due **next Monday 3/19 at 11:59pm**

# final presentations




- sometime during dead week
- staff selected top app submissions will be asked to present (prizes!)
- spec released tonight!!

# today's lecture

- Closures
- Networking
- CocoaPods
  - Alamofire

# closures

**Closures** are self-contained blocks of functionality that can be passed around and used in your code.

# closures

**Closures** are self-contained blocks of functionality that can be passed around and used in your code.

They can capture and store references to any constants and variables from the context in which they are defined

- **Global closure functions:** Named, do not capture values
- **Nested closure functions:** Named, capture values from enclosing function

# format of a closure

```
{ (parameters) -> return type in
    statements
}
```

# functions vs closures

```
func isGood(string: String) -> Bool {
    return string == "OK"
}
isGood(string == "OK")
```

# functions vs closures

```
func isGood(string: String) -> Bool {
    return string == "OK"
}
isGood(string == "OK")

let isGoodClosure = {(string: String) ->
Bool in
    return string == "OK"
}
isGoodClosure("OK")
```

In Swift, functions are just a special case of closures!

# global closures

```
let intPow = {(val1: Int, val2: Int) -> Int in
    return Int(pow(Double(val1),
                   Double(val2)))
}

let result = intPow(2, 10)
print(result)
```

# shorthand closures

```
let closure1 = { (i: Int) -> Int in return i * 2 }

// Closures can be type-inferenced, just like functions

let closure2 = { i in return i * 2 }

// If we only have one line, we can omit return

let closure3 = { i in i * 2 }

// We can also specify variables using $[position]
syntax rather than explicitly defining them

let closure4 = { $0 * 2 }
```

# nested closures

```
func makeIncrementer(forIncrement amount:
Int) -> () -> Int {
    var runningTotal = 0
    func incrementer() -> Int {
        runningTotal += amount
        return runningTotal
    }
    return incrementer
}
var incrementer =
makeIncrementer(forIncrement: 5)
```

# nested closures

```
func makeIncrementer(forIncrement amount:
Int) -> () -> Int {
    var runningTotal = 0
    func incrementer() -> Int {
        runningTotal += amount
        return runningTotal
    }
    return incrementer
}
var incrementer =
makeIncrementer(forIncrement: 5)
```

Used frequently in network calls!

# why closures?

- **Completion Blocks**

```
func longAction(completion: () -> ()) {
    for index in superBigArray {
        // do some really time consuming stuff
    }
    // notify the caller that longAction is finished
    completion()
}
```

- **Higher Order Functions**

```
let array = [1, 2, 3]
let arrayTimesTwo = array.map{ i in return i * 2 }
```

# networking

# networking and iOS

Networking is acquiring/passing data to/from some URL that exists on the world wide web or a local server

General structure as it relates to iOS

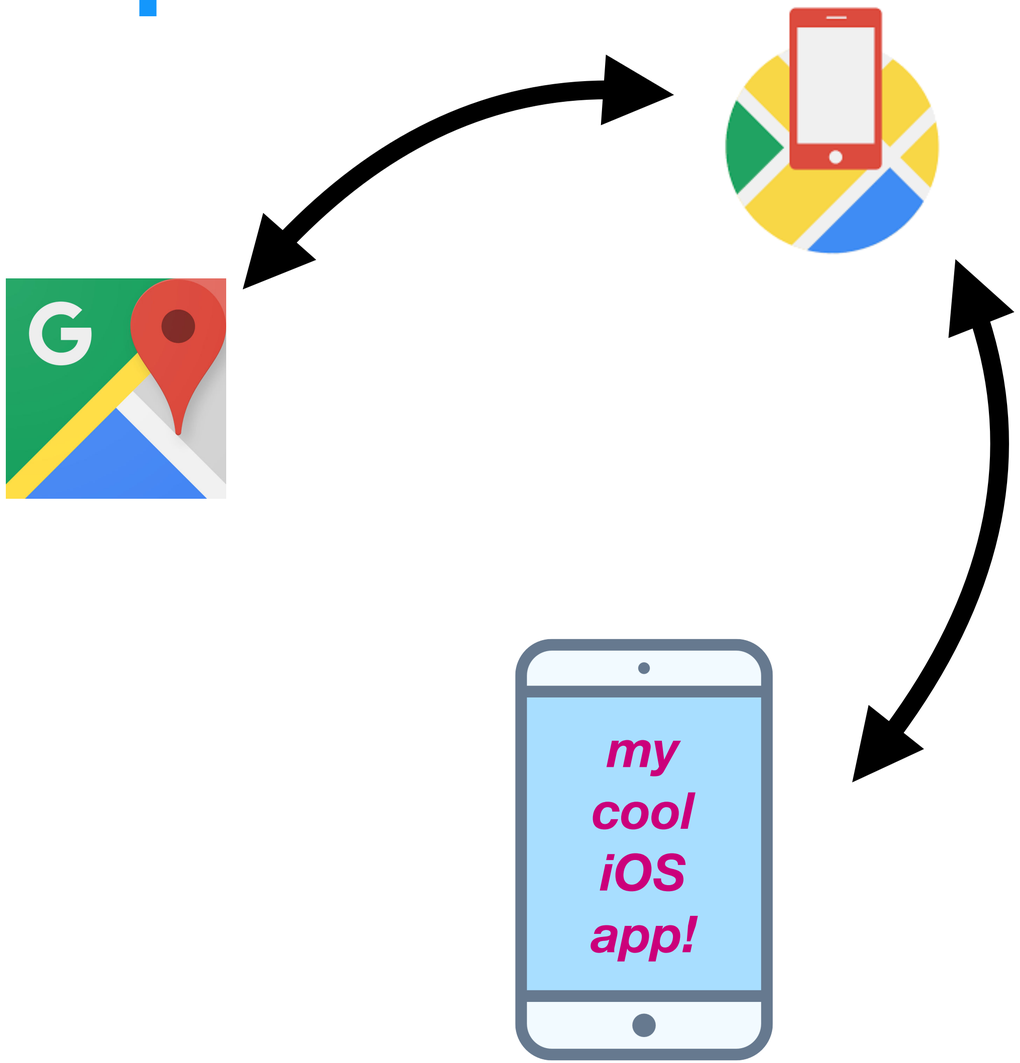
- Recipient Address
- Parameters
- Response

# what is an api?

**application programming interface**

abstraction layer between two software components

used throughout computer science (OS's, hardware, databases)

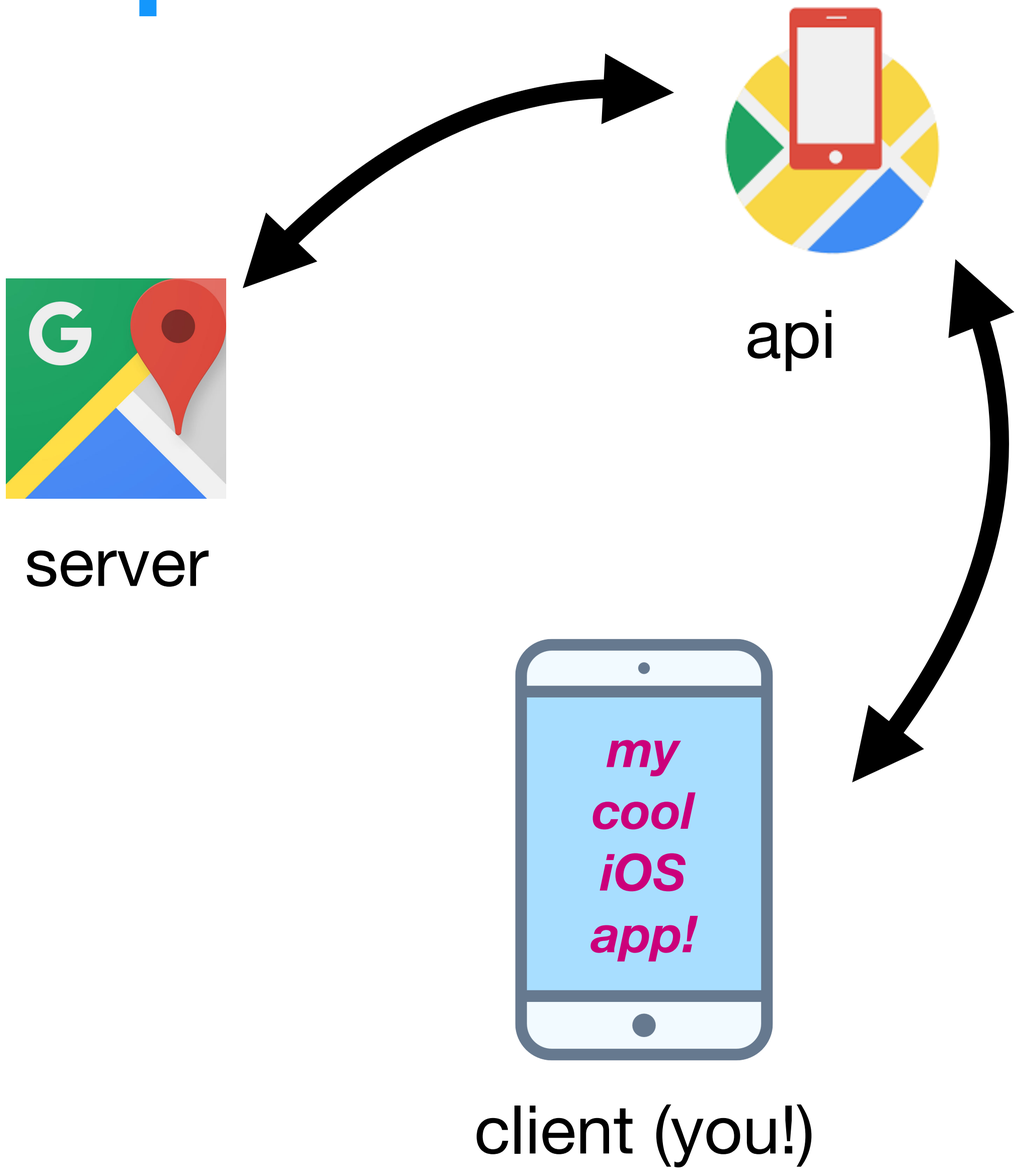


# what is an api?

**application programming interface**

abstraction layer between two software components

used throughout computer science (OS's, hardware, databases)

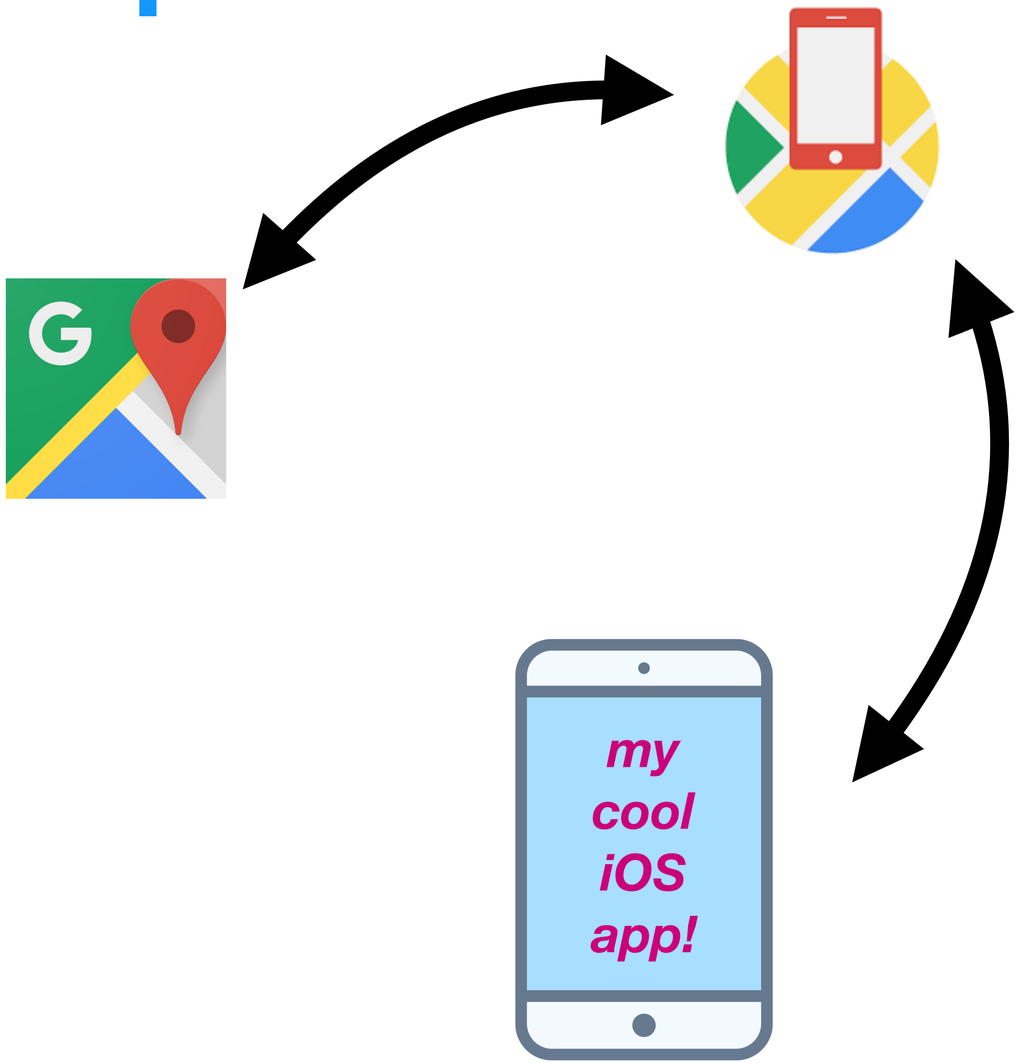


# what is an api?

example: you want to create an map related app!

one option: create a map application and gather data from scratch

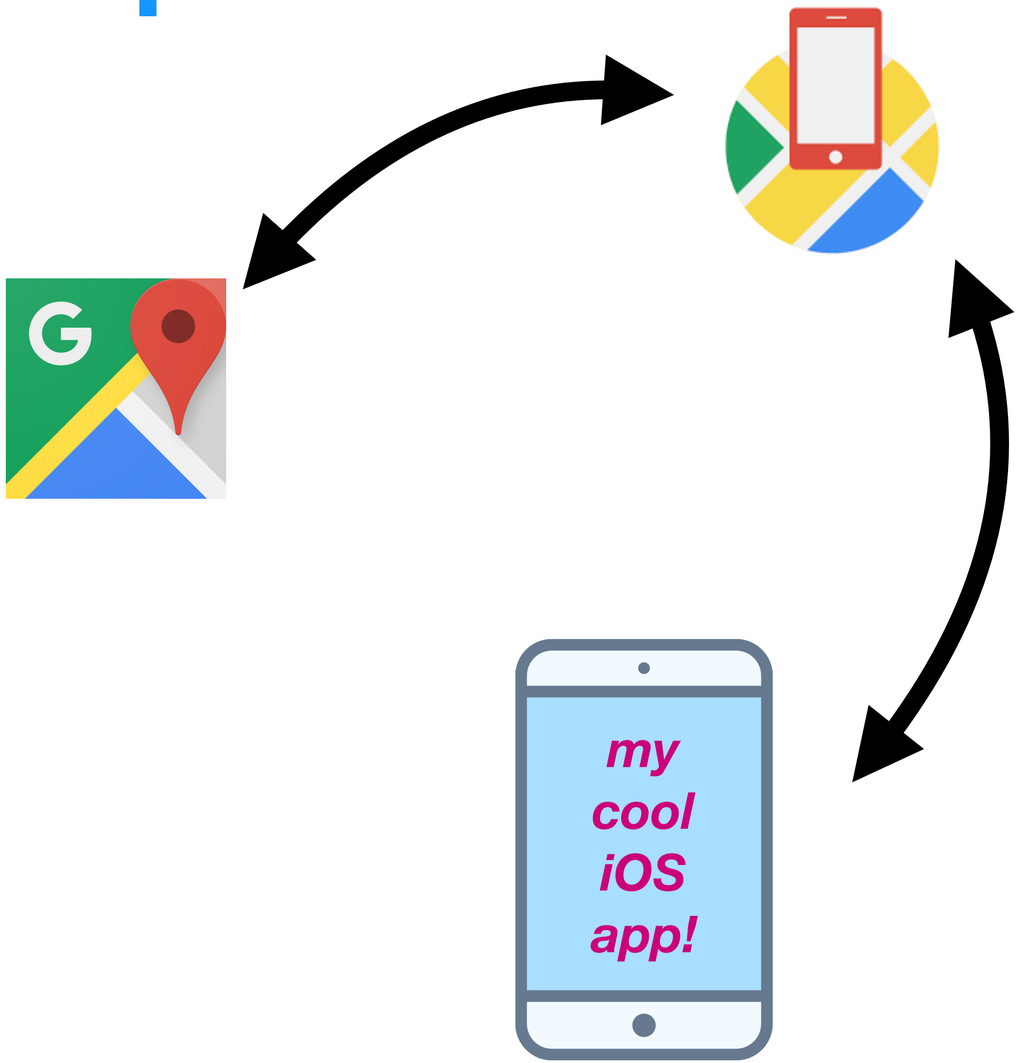
another option: integrate google maps data into your app, using an… api!



# what is an api?

another example:
you've gathered
extensive data and
have created a
database/app using it

share this data with
other developers
through your own api

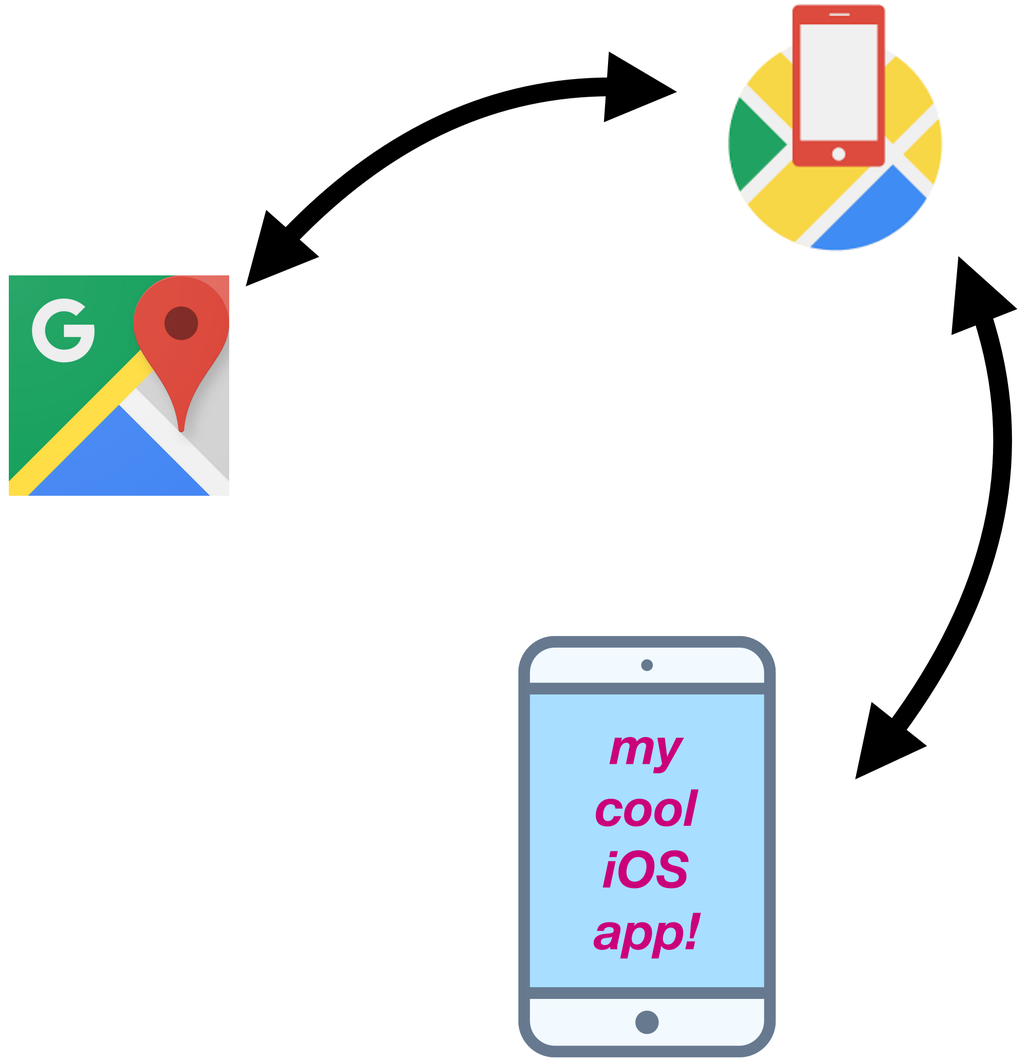


# why are apis used?

save time

hide
implementation
details while
providing
functionality

modularity



# how to use APIs in iOS applications

- **RESTful API calls** (via a URL)
- **iOS SDKs** (quite a few are still in Obj-C but there are often Swift versions!)

Google Places API for iOS

Add up-to-date information about millions of locations for your iOS app.

Google Maps Directions API

Calculate directions between locations using an HTTP request.

# RESTful APIs

# RESTful apis

**re**presentational

**s**tate

**t**ransfer

uses HTTP **requests** to GET, PUT, POST and DELETE data.

requests will return **responses** (XML, YAML, JSON)

# RESTful apis + HTTP methods

http://**maps.google.com**/maps/api/geocode/json?address=berkeley

server

# RESTful apis + HTTP methods

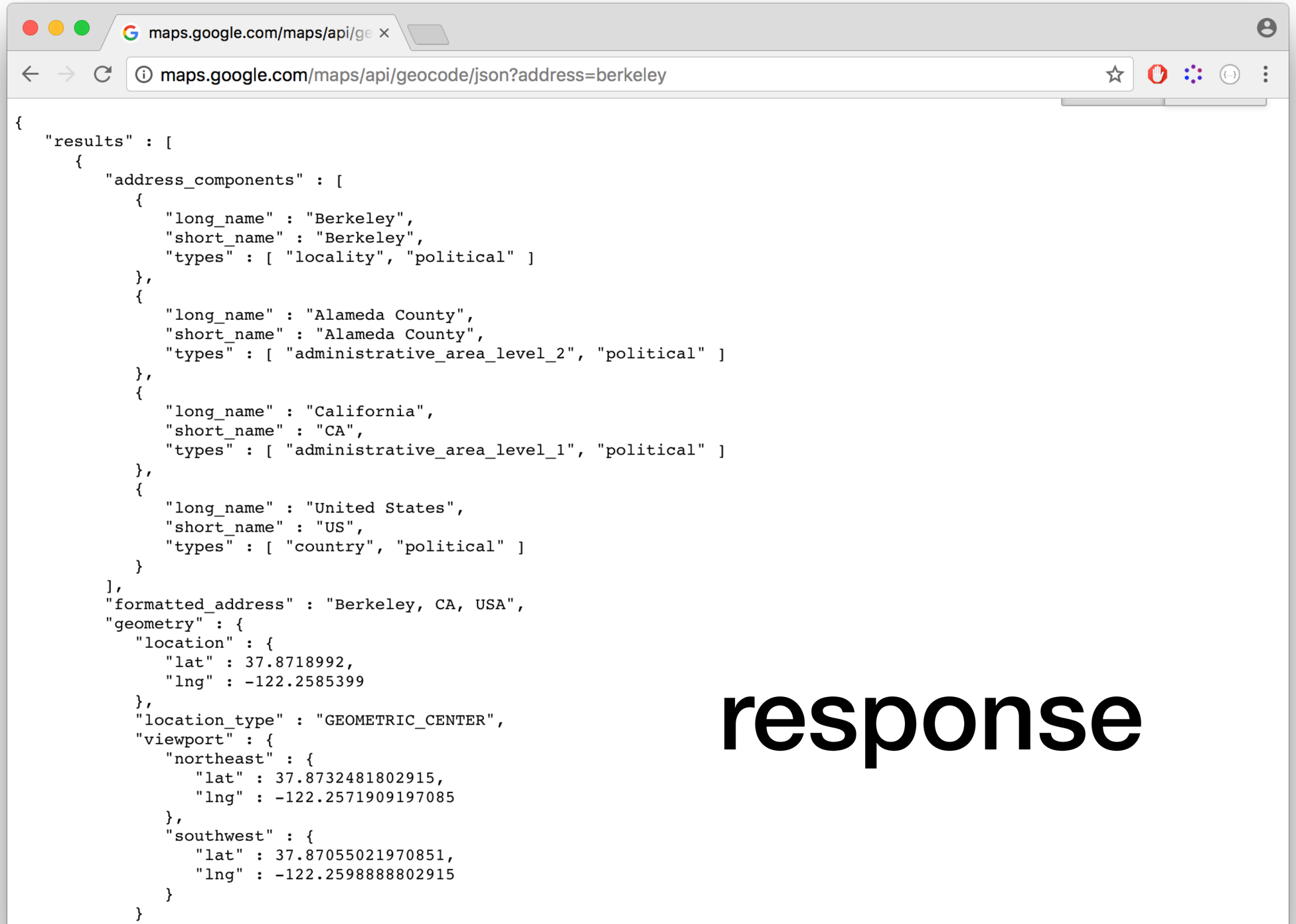
http://maps.google.com/**maps/api/geocode/json**?
address=berkeley

resources

# RESTful apis + HTTP methods

http://maps.google.com/maps/api/geocode/json?
**address=berkeley**

parameters


maps.google.com/maps/api/ge
maps.google.com/maps/api/geocode/json?address=berkeley
{
"results" : [
{
"address_components" : [
{
"long_name" : "Berkeley",
"short_name" : "Berkeley",
"types" : [ "locality", "political" ]
},
{
"long_name" : "Alameda County",
"short_name" : "Alameda County",
"types" : [ "administrative_area_level_2", "political" ]
},
{
"long_name" : "California",
"short_name" : "CA",
"types" : [ "administrative_area_level_1", "political" ]
},
{
"long_name" : "United States",
"short_name" : "US",
"types" : [ "country", "political" ]
}
],
"formatted_address" : "Berkeley, CA, USA",
"geometry" : {
"location" : {
"lat" : 37.8718992,
"lng" : -122.2585399
},
"location_type" : "GEOMETRIC_CENTER",
"viewport" : {
"northeast" : {
"lat" : 37.8732481802915,
"lng" : -122.2571909197085
},
"southwest" : {
"lat" : 37.87055021970851,
"lng" : -122.2598888802915
}
}
response

# JSON

**J**ava**s**cript
**O**bject
**N**otation

- easy to send to / from server
- language independent
- readable!

```
{
  "id": "883836255014203",
  "likes": {
    "data": [
      {
        "name": “Soda Hall",
        "id": "213133108715544",
        "created_time": "2017-08-22:"
      },
      {
        "name": “iOS decal",
        "id": "104681762916482",
        "created_time": "2017-08-21"
      },
      {
        "name": “Oski",
        "id": "548276038608761",
        "created_time": “2017-07-17"
...
```

# but how do we do this in iOS?

**making requests** - URLRequest, URLSession (built in), or Alamofire (3rd party)

**parsing** - JSONSerialization (built in) or SwiftyJSON (3rd party)

# URLSession

# URLSession

**Apple's API for downloading content**

**Support various URL schemes**

HTTP, HTTPS, FTP, Data, File

**Pass in a URL**

URL object, allocated from String

# Some Relevant Classes

**URL**

Object that contains URL

**URLRequest**

Contains URL, request method, etc.

**URLResponse**

Contains info for server’s response

# URLSession Workflow

1) **Create URL from a String**

2) **Create URLSession**

3) **Create a URLSessionDataTask**

   Get data from the task and save it

# URLSessionDataTask

- **dataTaskWithURL** - Default HTTP GET
- **dataTaskWithRequest** - Can specify HTTP in the form of a NSURLRequest

# URLSession

```
func loadImage() {
  let url = URL(string:”https://iosdecal.com/img.jpg")

  let session = URLSession.shared

  let task = session.dataTask(with: url!,
                              completionHandler: {
       (data, response, error) -> Void in
       if error == nil {
            let img = UIImage.init(data: data!)
            self.imageView.image = img
       }
    })

  task.resume()
}
```

# JSON parsing in iOS

JSONSerialization using jsonObject

```
jsonObject(with: options:)
```

# JSON parsing in iOS

JSONSerialization using jsonObject

`jsonObject(with: options:)`

```
let data: Data = [received from network request]
let json = try? JSONSerialization.jsonObject(with:
                    data, options: [])

// Do stuff with your JSON

if let names = json["names"] as? [String] {
    print(names)
}
```

# accessing values from json

```
// Example JSON with object root:
/*
   {
     "someKey": 42.0,
     "anotherKey": {
       "someNestedKey": true
     }
   }
*/
if let dictionary = json as? [String: Any] {
   if let number = dictionary["someKey"] as? Double {
     // access individual value in dictionary
   }
}
```

example link

# parsing via initializers

```
struct Dog {
  init?(json: [String: Any]) {
    guard let name = json["name"] as? String,
      let friends = json["friends"] as? [String],
      let bestFriendName = friends[0],
      let secondBestFriendName = friends[1],
      let meals = json[“meals"] as? [String: Any],
    else {
      return nil
}
```

We can create model objects from JSON!

# CocoaPods

# What are CocoaPods

A dependency manager for Cocoa Projects

CocoaPods are Swift and Objective-C classes that other people write for you that you can use in your project:

- Makes life more **efficient** and **easier**

# What are CocoaPods

A dependency manager for Cocoa Projects

# Timepiece

## Adding A Year To the Current Date:

Without Timepiece:

```
let calendar = NSCalendar.currentCalendar()
let newDate = calendar.dateByAddingUnit(.Year, value:
1, toDate: NSDate(), options:
NSCalendarOptions.MatchNextTime)
```

With Timepiece

```
let newDate = now + 1.year
```

# How to use CocoaPods

# Install CocoaPods

```
sudo gem install cocoapods
```

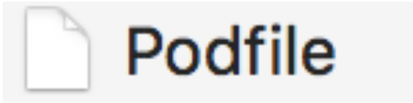
Or if you prefer Homebrew:

```
brew install cocoapods
```

# Make a Podfile

1. Just named Podfile with no extension
2. Format

Hint: `pod init` does this automatically!



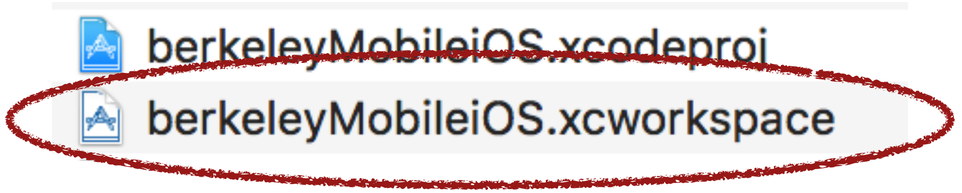

```
platform :ios, '8.0'
use_frameworks!

target 'MyApp' do
  pod 'Timepiece' '~> 1.0.2'
end
```

# Update Dependencies

Once you have your podfile,

```
pod install
```

# Open .xcworkspace



# Check In!

# Alamofire

# **Alamofire** : Networking in Swift

HTTP networking library written in Swift

Simplifies common networking tasks

- Request/Response methods
- JSON serialization
- Authentication

GitHub link

# **Alamofire** : Requests

**.request**: HTTP requests
**.upload**: Upload large files
**.download**: Download large files or resume a download already in progress.

# **Alamofire** : Request types

```
Alamofire.request("https://httpbin.org/get")
    // default is GET

Alamofire.request("https://httpbin.org/post",
method: .post)
```

# **Alamofire** : Response Handlers

```
// Response Data Handler - Serialized into Data
func responseData(queue: DispatchQueue?,
completionHandler: @escaping (DataResponse<Data>)
-> Void) -> Self


// Response JSON Handler - Serialized into Any
func responseJSON(queue: DispatchQueue?,
completionHandler: @escaping (DataResponse<Any>)
-> Void) -> Self
```

# **Alamofire** : Response Validation

```
Alamofire.request("https://httpbin.org/get”)
    .validate().responseJSON { response in
        switch response.result {
            case .success:
                print("Validation Successful")
            case .failure(let error):
                print(error)
        }
}
```

# **Alamofire** : Parameters

```
Alamofire.request("https://httpbin.org/post",
        method: .post,
        parameters: parameters)

Alamofire.request("https://httpbin.org/post",
        method: .post,
        parameters: parameters,
        encoding: URLEncoding.default)
```

# **Alamofire** : Authentication

```
let user = "user"
let password = "password"

Alamofire.request(“https://
    httpbin.org/basic-auth/\(user)/\
    (password)")
     .authenticate(user: user,
                   password: password)
     .responseJSON { response in
   debugPrint(response)
}
```