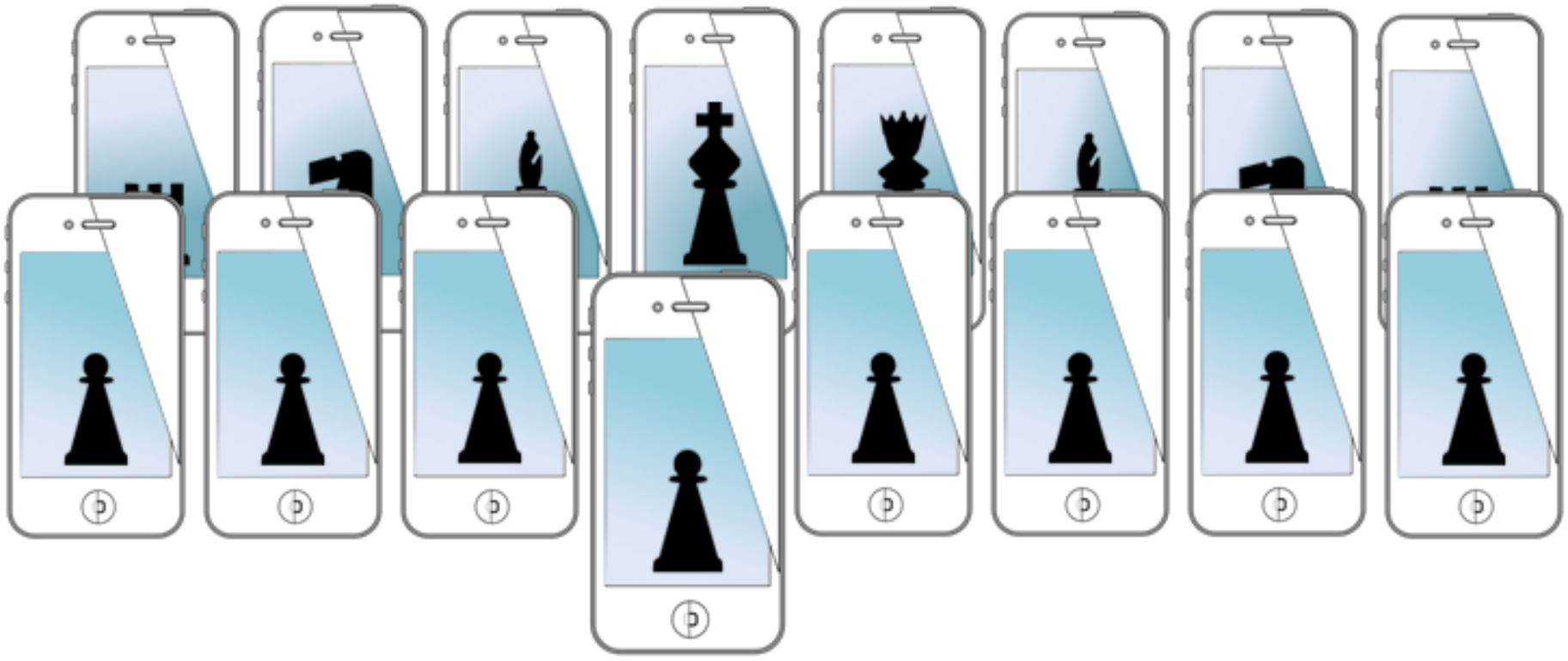
MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

tornado, pymongo, and http requests

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University

course logistics

- lab four due next week
- start to think about the final project proposal

agenda

- *finish tornado (done!)*
- *mongodb (done!)*
- http requests in iOS
- project proposals

working with your web server

- we want to send data to our hosted server!
  - or any server for that matter
- need to form POST and GET requests from iOS
- we will use NSURLSession

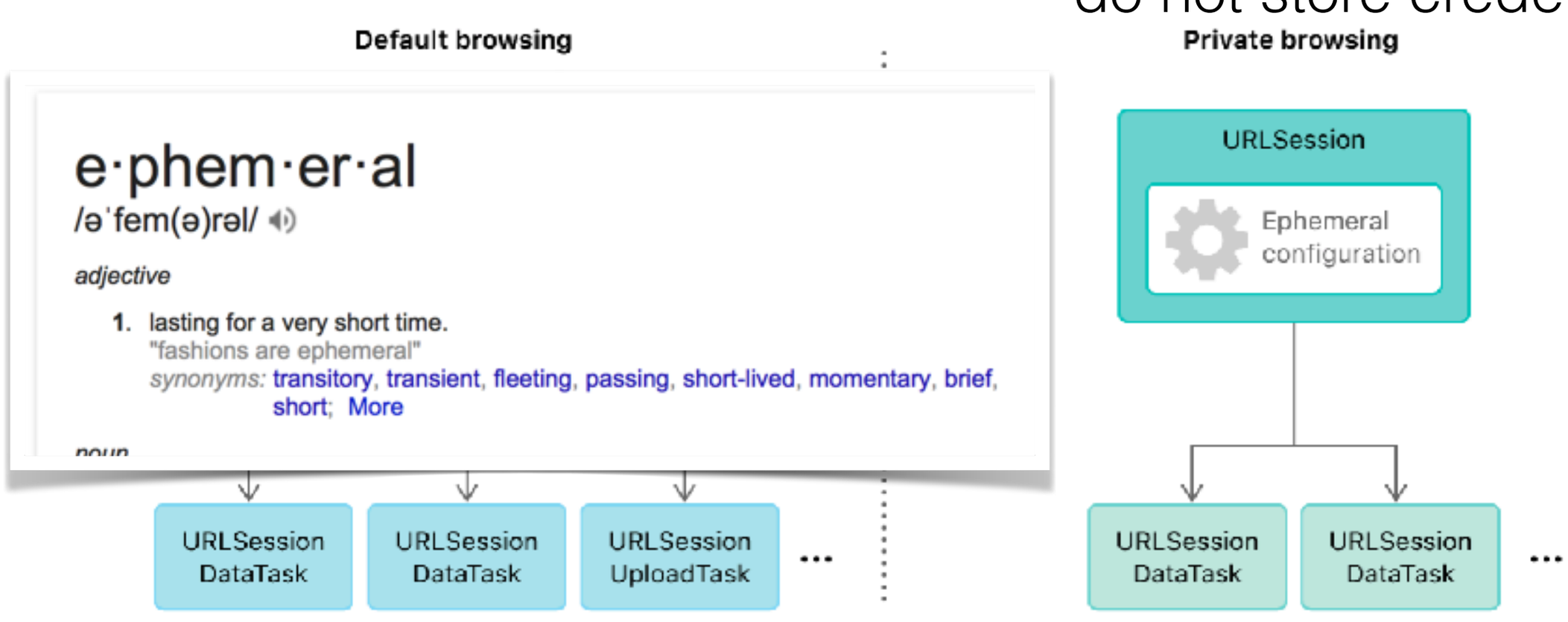
NSURLSession

- proper way to configure a session with a server
- new format starting in iOS7
  - old way was to use `NSURLConnection`
  - before that was to use `sendAsynchronousRequest`
- you may see code for `initWithContentsOfURL`:
  - **never, never, never** use that for networking
- sessions are a huge improvement in iOS
  - and extremely powerful
  - the Stanford course talks about these (check it out)!
  - as promised, we will cover different topics than Stanford course

URLSession

- delegate model
- does authentication if you need it!
  - we won't use that though — who would hack our server?
- implements pause / resume, tasks

do not cache
no cookies
do not store credentials



configure a session

```
class ViewController: UIViewController, URLSessionDelegate {

    // MARK: Class Properties
    var session = URLSession()
    let operationQueue = OperationQueue()
```

delegation

will reuse session

```
  //setup NSURLSession (ephemeral)
    let sessionConfig = URLSessionConfiguration.ephemeral

    sessionConfig.timeoutIntervalForRequest = 5.0
    sessionConfig.timeoutIntervalForResource = 8.0
    sessionConfig.httpMaximumConnectionsPerHost = 1

    self.session = URLSession(configuration: sessionConfig,
            delegate: self,
            delegateQueue:self.operationQueue)
```

custom queue

configure a session

- other options:

`ephemeralSessionConfiguration`

`defaultSessionConfiguration`

use global cache, cookies, and credential storage objects

`backgroundSessionConfiguration`

make my session respond to push notifications, launch my app, if needed, handle download completion

configure a task

- tasks are common requests tied to a session
- they give a way to **specify URL** and type of **request**
- we will use a **completion handler** to interpret response from Server
- **larger** downloads allow use of delegates
    - progress indicators, completion indicators

URLSessionDataTask

- common to use for GET requests
- uses blocks for completion

```
dataTaskWithURL:completionHandler:
```

String

String

```
let baseURL = “\(SERVER_URL)/GetRequestURL" + query

let getUrl = URL(string: baseURL)
let request: URLRequest = URLRequest(url: getUrl!)
let dataTask : URLSessionDataTask = self.session.dataTask(with: request,
    completionHandler:{(data, response, error) in
        print("Response:\n%@",response!)
})

dataTask.resume() // start the task
```

must call, or stays suspended

URLDownloadTask

reference slide

- sub-class of URLSessionDataTask
- many delegate methods for getting progress

```
    NSURLSessionDownloadTask *downloadTask = [self.session downloadTaskWithURL:[NSURL
URLWithString:@"someurlfordownloadingimages.com/coolimage"]
          completionHandler:^(NSURL *location, NSURLResponse *response, NSError *error) {
                if(!error)
                {
                    UIImage *img = [UIImage imageWithData:[NSData dataWithContentsOfURL:location]];
                }
            }];
    [downloadTask resume];
```

could use delegate instead of completion handler

```
-(void)URLSession:(NSURLSession *)session
     downloadTask:(NSURLSessionDownloadTask *)downloadTask
didFinishDownloadingToURL:(NSURL *)location

-(void)URLSession:(NSURLSession *)session
     downloadTask:(NSURLSessionDownloadTask *)downloadTask
     didWriteData:(int64_t)bytesWritten
totalBytesWritten:(int64_t)totalBytesWritten
totalBytesExpectedToWrite:(int64_t)totalBytesExpectedToWrite
```

URLSessionDataTask

- common to use for PUT/POST requests
- need to setup HTTP request (default is GET)

`uploadTaskWithRequest:fromData:completionHandler`

```
// create a custom HTTP POST request
let baseURL = "\(SERVER_URL)/PostUrl"
let postUrl = URL(string: "\(baseURL)")
var request = URLRequest(url: postUrl!)

let requestBody:Data? = UIImageJPEGRepresentation(image, 0.25);

 request.httpMethod = "POST"
request.httpBody = requestBody

let postTask : URLSessionDataTask = self.session.dataTask(with: request,
        completionHandler:{(data, response, error) in

 })

 postTask.resume() // start the task
```

could be any data

URLDataTask

reference slide

- can also use this for PUT or POST requests
- highly similar to uploadTask
  - but you don't get the delegate methods for progress

```
// create a custom HTTP POST request
NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:postUrl];
[request setHTTPMethod:@"POST"];

NSData *imageData = UIImageJPEGRepresentation(image, 0.25);
[request setHTTPBody:imageData];

NSURLSessionDataTask *dataTask =
[self.session dataTaskWithRequest:request
        completionHandler:^(NSData *data, NSURLResponse *response, NSError *error) {

         }];
```

JSON serialization

- parse in tornado

```
import json

class JSONPostHandler(BaseHandler):
   def post(self):
    '''Parse some posted data
    '''
     data = json.loads(self.request.body)
```

- parse in iOS

```
let jsonDictionary: Dictionary =
      try JSONSerialization.jsonObject(with: data!,
       options: JSONSerialization.ReadingOptions.mutableContainers) as! Dictionary
```

the output in both scenarios is a dictionary

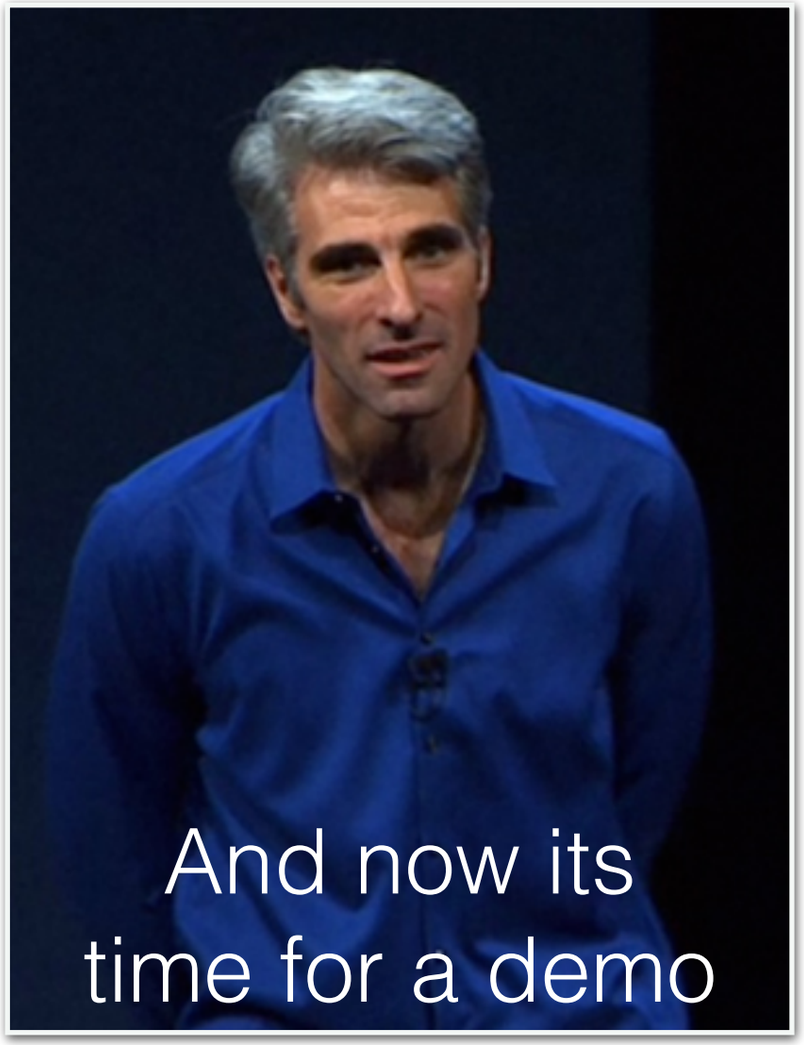
Dictionary

- serialize in iOS

```
let requestBody = try JSONSerialization.data(withJSONObject: jsonUpload,
            options:JSONSerialization.WritingOptions.prettyPrinted)
```

tornado + iOS demo

- send a GET request, handle query in tornado
- do POST with GET-like query
- do POST with JSON in, JSON out



before talking about proposals for next time…

- **next time**: basics of machine learning
  - machine learning as a service
    - install turi-create
- **one week**: **flipped assignment** with our own restful API
  - using ML and networking to do cool things…
  - some code has changed since filming, but mostly the same

project proposal

Final Project Proposal

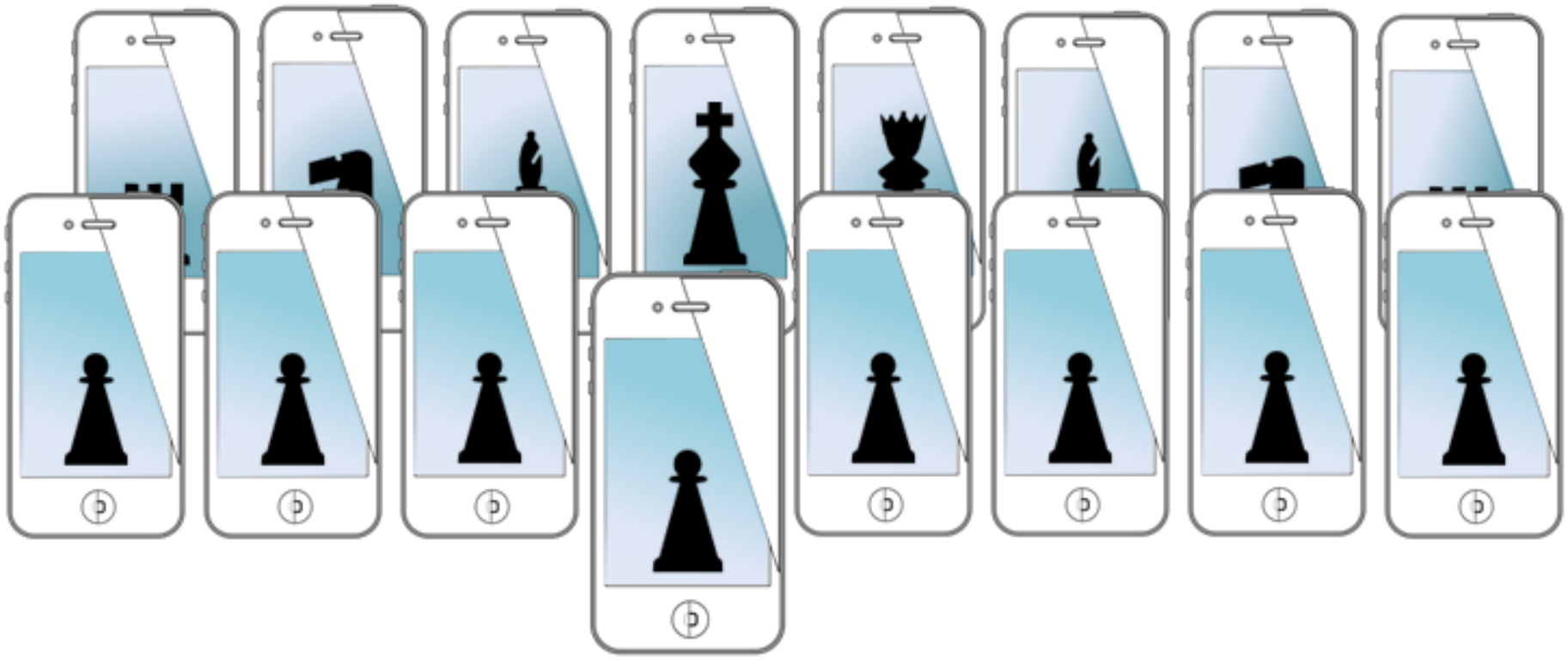
All final projects should be approved by the instructor via the final project proposal. This is a description explaining the overall idea, which labs the project builds from, and a list of four or more design constraints that the design will meet. Turn in the project proposal via canvas or talk it over with the instructor.

If your group is opting into the "mother of all demos" (see explanation below) your proposal must specify this (you can opt out later, but you cannot opt in). Note that MOD projects should have more difficult constraints. The instructor may tighten constraints for those that wish to opt into the MOD.

project proposals

- head on over to the another keynote

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

tornado, pymongo, and http requests

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University