Intro to iOS Development

Patrick Linskey
@plinskey

You

Enterprise Java

iOS development

Me

Patrick Linskey
@plinskey
http://versly.com

Objective C Primer

- Object-oriented extension of C
- Smalltalk-like syntax
- “Full” C compatibility

[object message]

- ***Messages***, not ***methods***
- Braces at the beginning, which is weird.

```
id list = [[NSArray alloc] init];
```

Message Arguments

```
id text = [NSString stringWithFormat:@“Hello, %@!”,
    @“Copenhagen”];
```

- Arguments are embedded in the message signature
- Akin to named parameters, but order is fixed
- Type overloading is unsupported

Legacy C issues

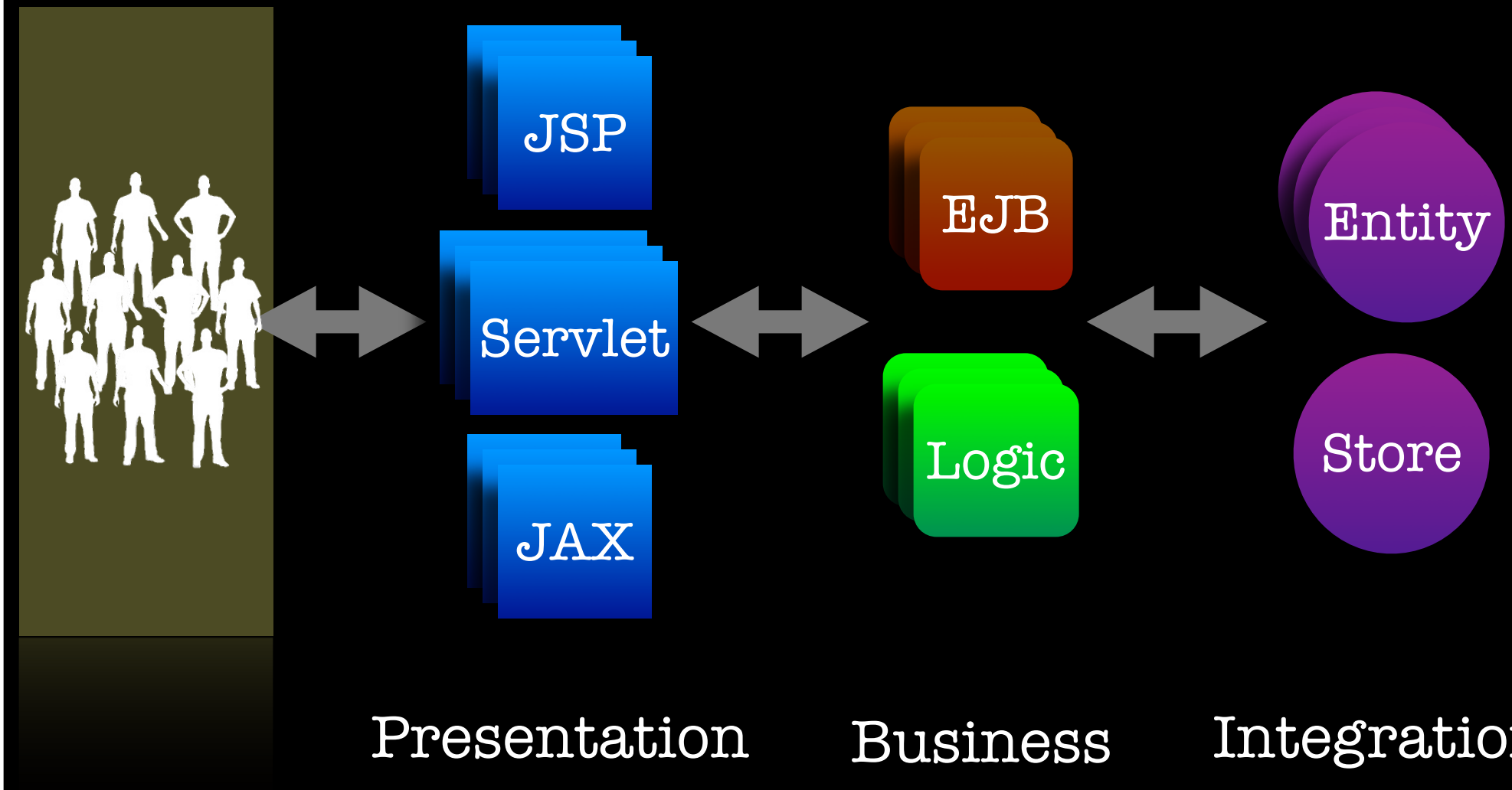
- Header files
- Single-pass compilation
  - Forward-declare private messages, or careful class file definition ordering

@ Directives

- Objective C preprocessor directives
- Strings: @“foo”
- Properties: @property, @synthesize
- Multi-threading: @synchronized
- etc.

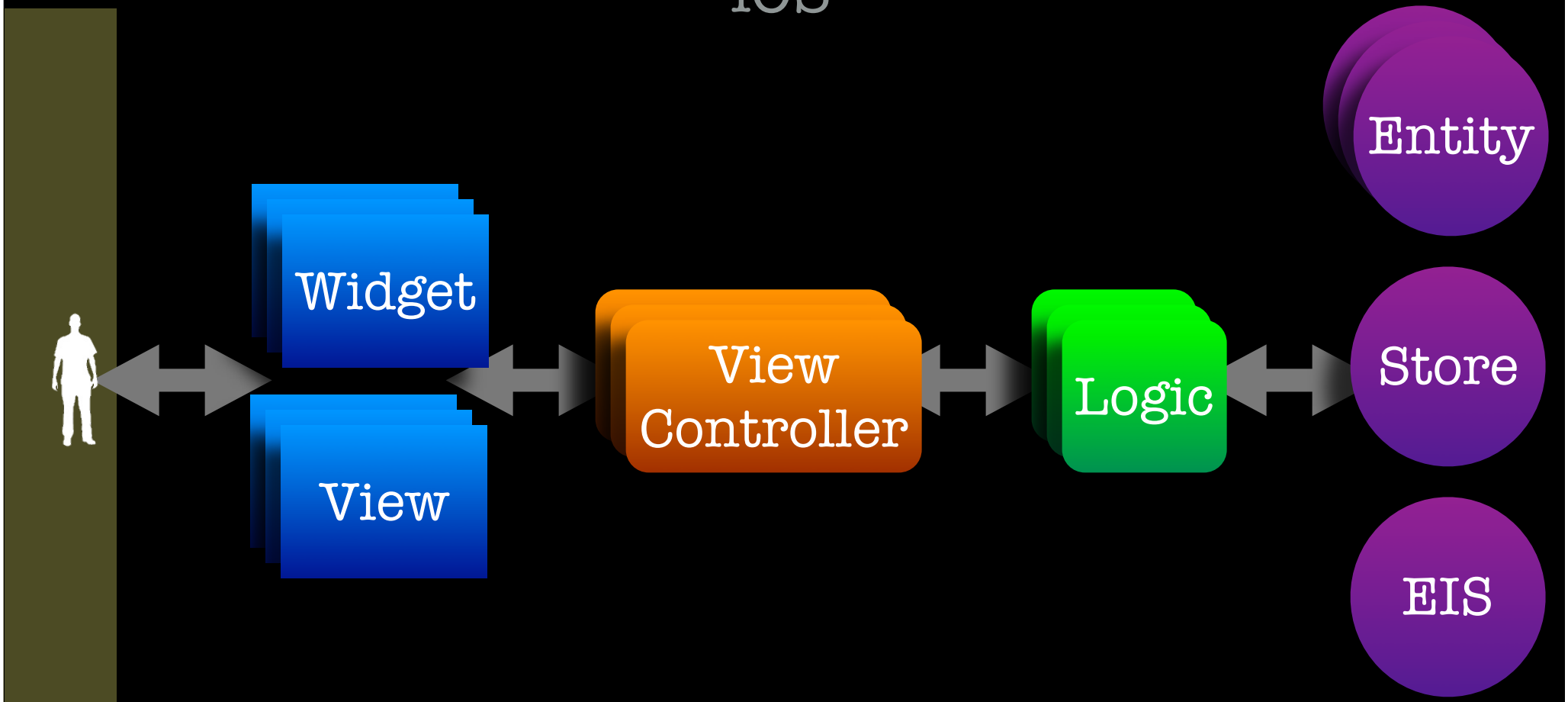
Architecture

JEE



Architecture

iOS



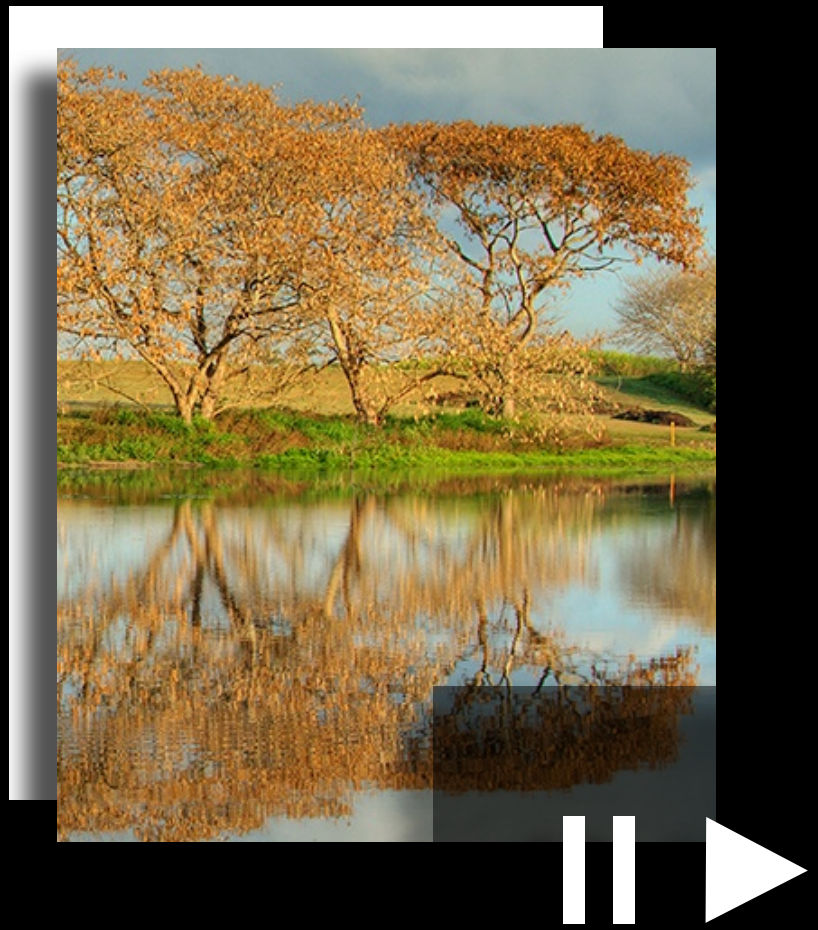
M Model

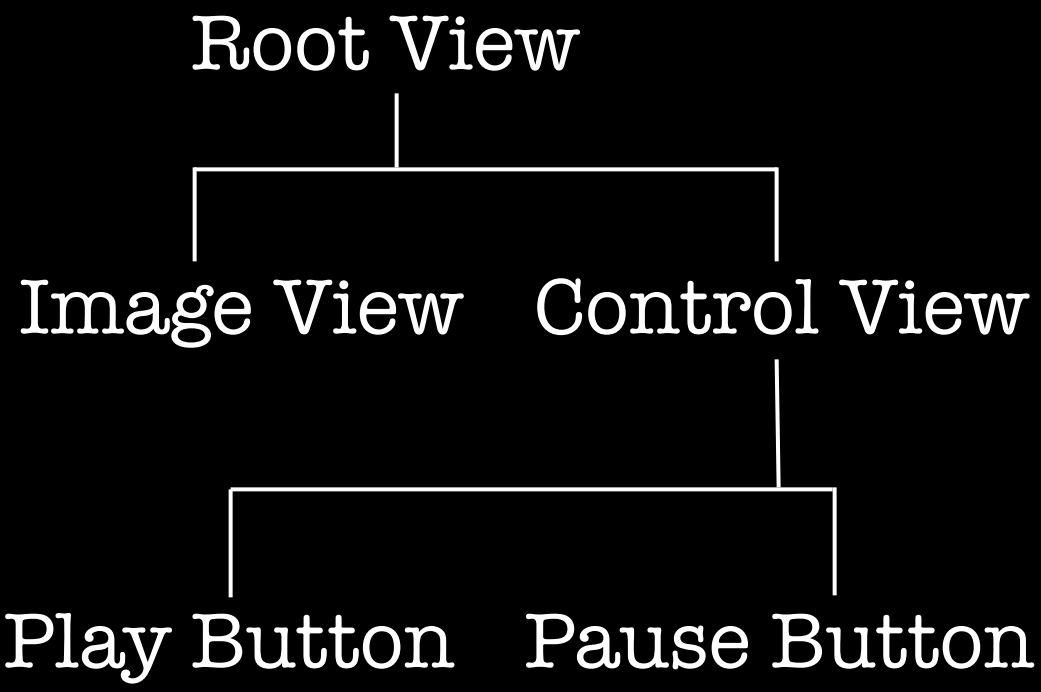
V View

C Controller

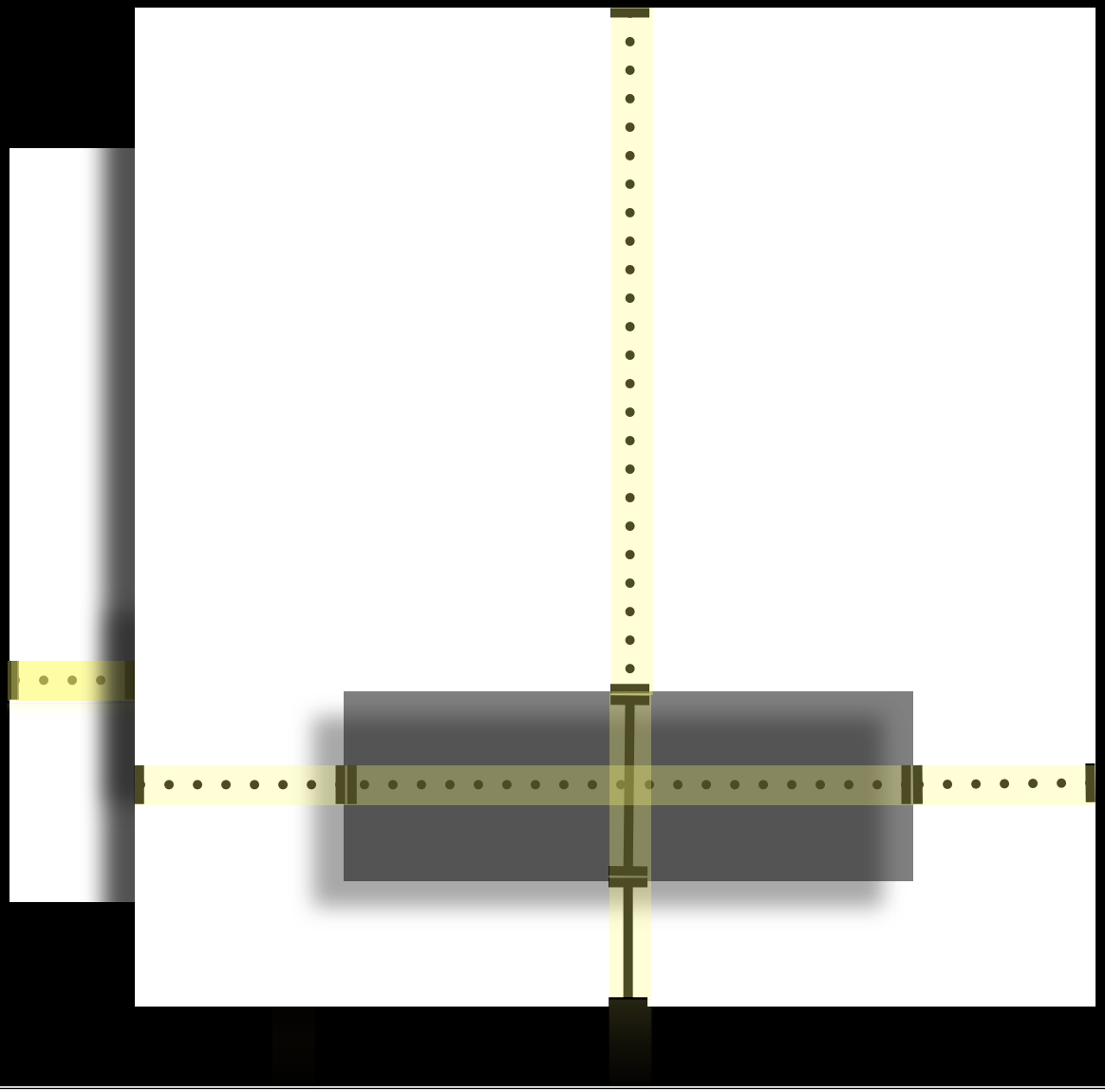
V

View Tree





V Layout

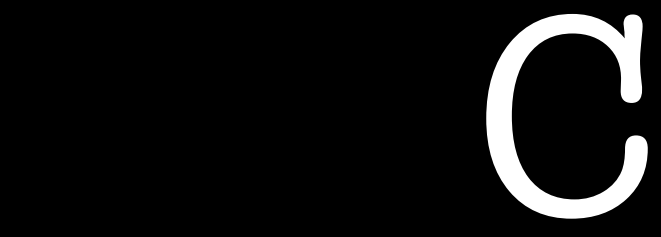


C

ObjC Syntax

```
int width = view.getWidth();
int height = view.getHeight();
view.setSize(10, 20);
```

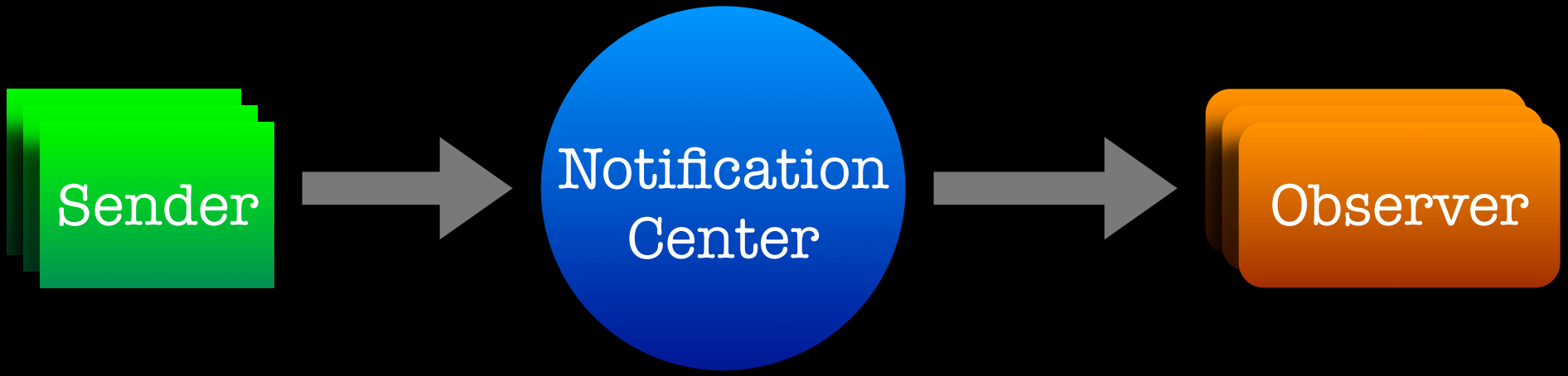
```
int width = [view width];
int height = [view height];
[view setWidth:10 height:20];
```



Listeners vs Actions

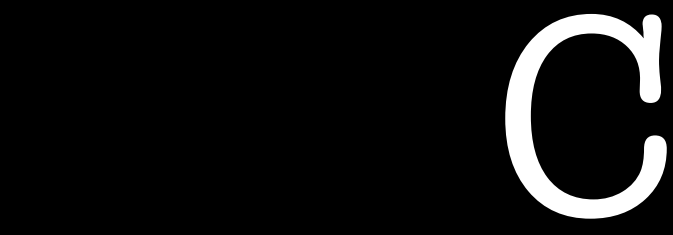
```
button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(Event e) {
        doSomething();
    }
});

[button addTarget:self
    action:@selector(doSomething)
    forControlEvents:TouchUpInside];
```

C Notifications



C Notifications

```
[[NotificationCenter defaultCenter]
    postNotificationName:PersonDidDeleteNotification
    object:person];

                                    . . .

[[NotificationCenter defaultCenter] addObserver:self
    selector:@selector(personDidDelete:)
    name:PersonDidDeleteNotification
    object:nil];

                                    . . .

- (void)personDidDelete:(Notification*)notification {
    Person *person = [notification object];
    ...
}
```



Delegates

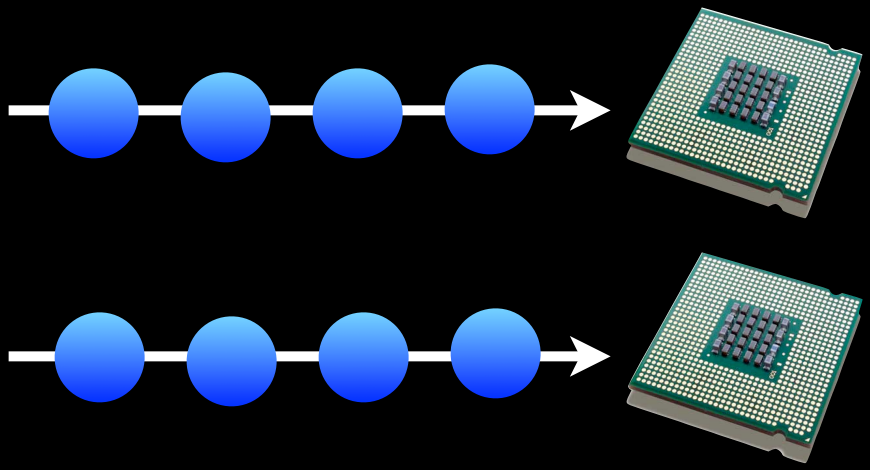
```
@protocol ListViewDelegate

- (int)numberOfItemsInListView:(ListView*)list;
- (ListItem*)listView:(ListView*)list itemAtIndex:(int)i;
...

@optional

- (void)listView:(ListView*)list didSelectItemAtIndex:(int)i;
...

@end
```

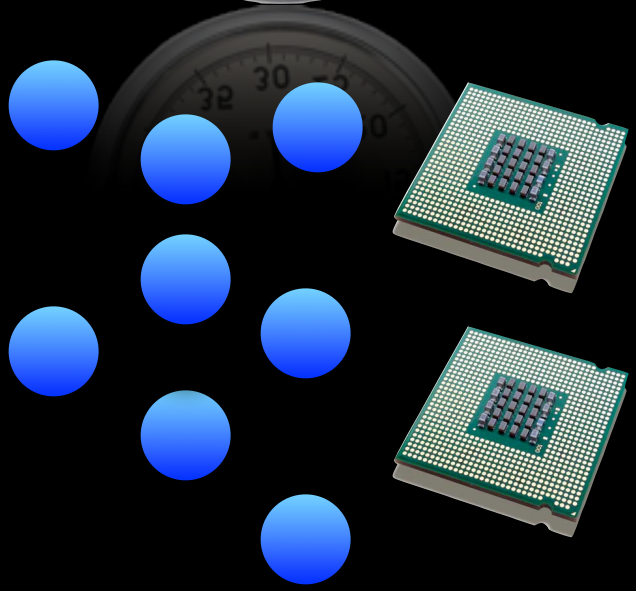
Concurrency

Any sufficiently advanced bug is indistinguishable from a feature.

- Bruce Brown

Concurrency



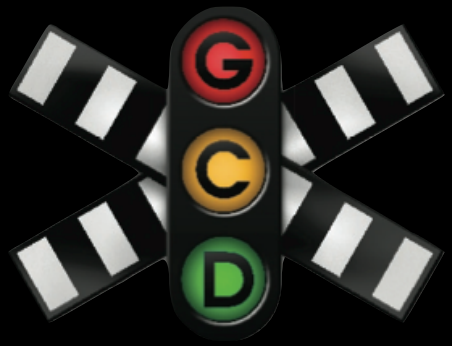


Concurrency

JEE

Concurrency

iOS



Concurrency

iOS

```
Data *data = ...;
[self performSelectorInBackground:@selector(processData:)
                       withObject:data];
[view show];
```

```
- (void)processData:(Data*)data {
   id result = ...;
   [self performSelectorOnMainThread:@selector(display:)
      withObject:result
      waitUntilDone:NO];
}
- (void)display:(id)result {
   [view display:result];
   [view show];
}
```

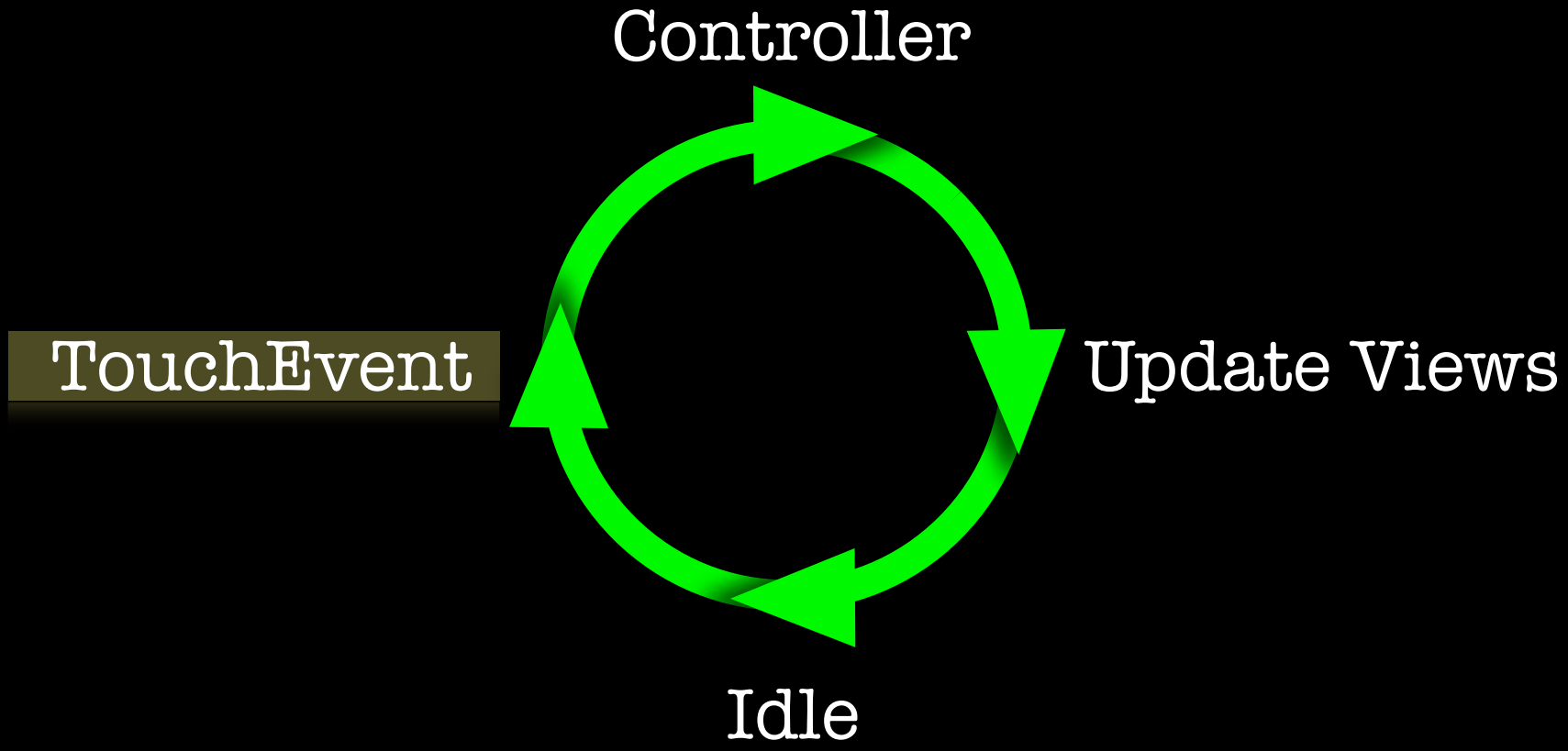
Concurrency

iOS

```
[self performSelector:@selector(hideControls)
   withObject:nil
   afterDelay:3.0];
[busyIndicator show]; . . .
[self processData:data];
[Object cancelPreviousPerformRequestsWithTarget:self
   selector:@selector(hideControls)
   object:nil];
```

Concurrency

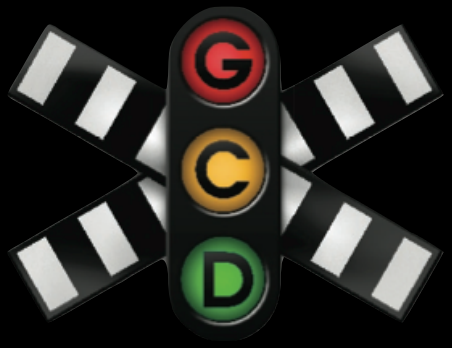
iOS



Concurrency

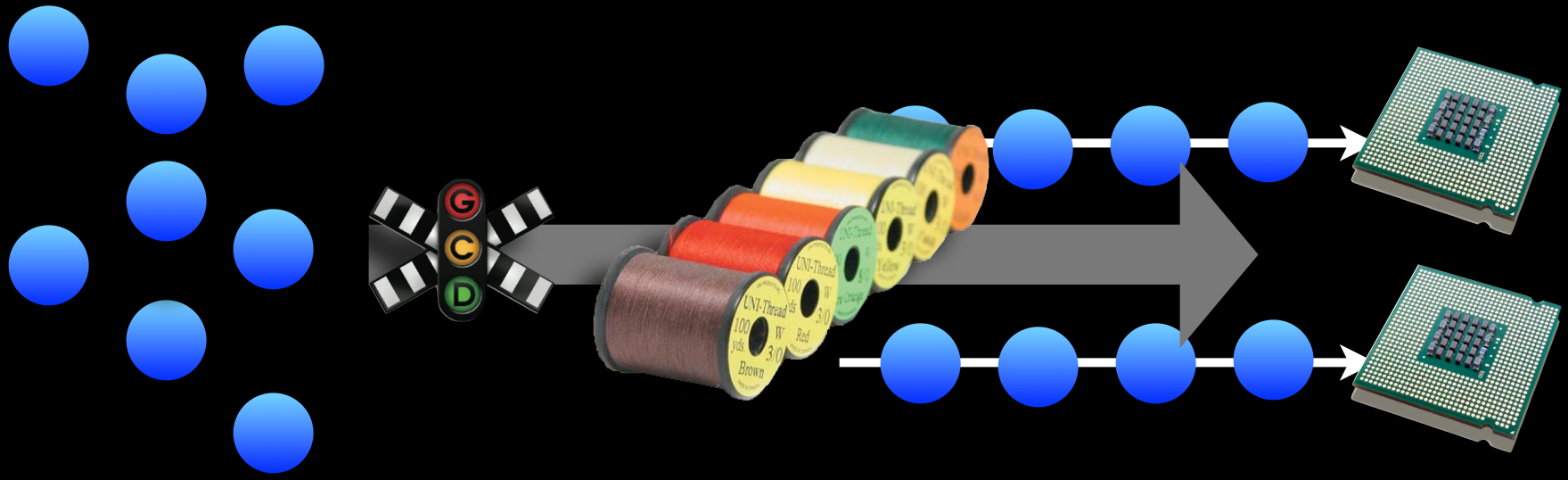
iOS

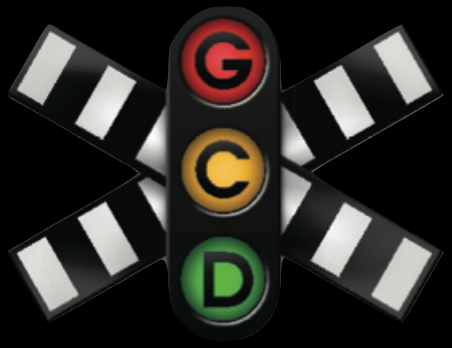
```
[busyIndicator show];
[self performSelector:@selector(processData:)
   withObject:data
   afterDelay:0.0];
```



Concurrency

iOS





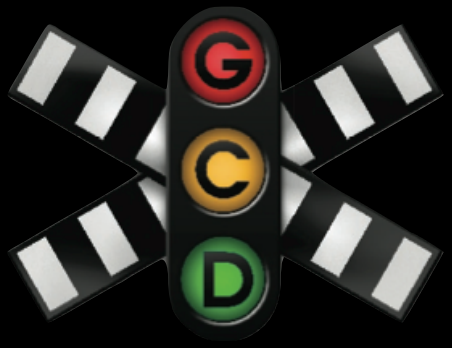
Concurrency

iOS

Queueing Model

C-level API

Blocks

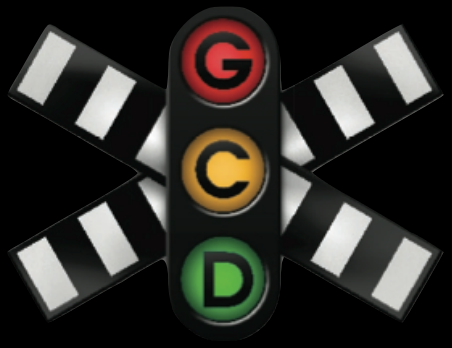


Concurrency

iOS

Blocks

```
float (^myBlock)(int, int) = ^(int x, int y) {
    return x / (float)y;
};

float result = myBlock(1, 2);
```

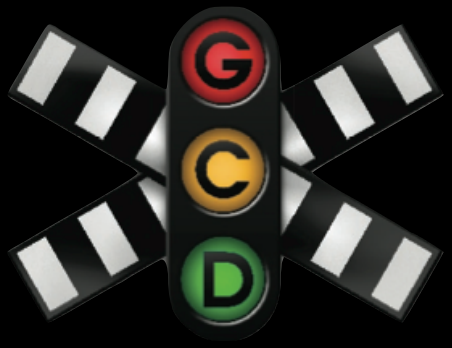


Concurrency

iOS

Blocks

```
int y = 2;

float (^myBlock)(int) = ^(int x) {
    return x / (float)y;
};

float result = myBlock(1);
```



Concurrency

iOS

Blocks

```
(float (^)(int)) divideByY(int y) {
    return ^(int x) {
        return x / (float)y;
    };
}

float (^divideBy2)(int) = divideByY(2);
float result = divideBy2(1);
```

Concurrency

iOS

```
Data *data = ...;
id result = [self processData:data];
[view display:result];
[view show];
```

Concurrency

iOS

```
Data *data = ...;
dispatch_async(dispatch_get_global_queue(0,0), ^{
    id result = [self processData:data];
    dispatch_async(dispatch_get_main_queue(), ^{
        [view display:result];
        [view show];
    });
});
```

Concurrency

iOS

```
Cache *cache = ...;
dispatch_queue_t cacheQ = dispatch_queue_create(
   “com.xyz.cache”, NULL);
                                        . . .
Record *record = ...;
dispatch_async(cacheQ, ^{
   [cache addRecord:record];
});
                                        . . .
__block Record *result;
dispatch_sync(cacheQ, ^{
   result = [cache recordForKey:key];
});
```

Memory

- No garbage collection!
- Reference counting instead
  - ObjC for Mac OS has GC. iOS will too (soon? http://bit.ly/95kn6f)
- Low-memory notifications

Memory

alloc = create 

retain = increase reference count 

release = decrease reference count 

autorelease = release later 

Memory

1. **[retain]** what you need later
2. **[release]** what you alloc or retain
3. **[autorelease]** what you return

Memory

```
@interface Person {
    String *_firstName;
    String *_lastName;
}
...
@property String *firstName;
@property String *lastName;
- (String *)fullName;
...
@end
```

Memory

```
- (void)setFirstName:(String*)newFirst {
    [newFirst retain];
    [_firstName release];
    _firstName = newFirst;
}
```

Memory

```
[person setFirstName [person getFirstName]];

- (void)setFirstName:(String*)newFirst {
    [_firstName release];
    [newFirst retain];
     _firstName = newFirst;
}
```

Memory

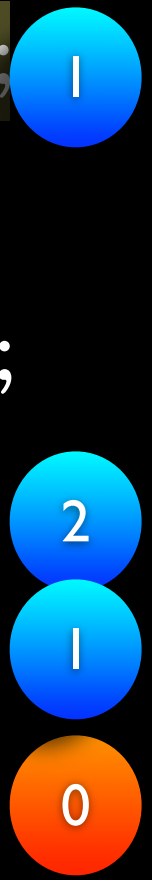
```
- (String *)fullName {
    String *full = [[String alloc] initWithFormat:
        “%s %s”, _firstName, _lastName];
    return [full autorelease];
}
```

Memory

```
- (void)dealloc {
    [_firstName release];
    [_lastName release];
    [super dealloc];
}
```

Memory

```
Person *person = [[Person alloc] init];
person.firstName = “Patrick”;
person.lastName = “Linskey”;
String *fullName = [person fullName];
Log(fullName);
[list add:person];
[person release];

                    . . .

[list remove:person];
```



Memory

```
[[NotificationCenter defaultCenter] addObserver:self
    selector:@selector(memoryWarning:)
    name:MemoryWarningNotification
    object:nil];

                                    ...
- (void)memoryWarning:(Notification*)notification {
    [cache clear];
    ...
}
```

Memory

```
- (void)viewDidUnload {
    [view release];
    view = nil;
    ...
    [super viewDidUnload];
}
```

Testing

- Xcode has not been a shining beacon in the darkness here
- Ships with SenTest and OCUnit
  - Integration is ***much*** better than in old Xcodes (<= 3)
- GHUnit* is also popular
  - Basically mandatory for old Xcodes

* https://github.com/gabriel/gh-unit

The App Store

- Walled Garden
- Approval latency is unpredictable
- Fast-delivery-cycle teams will find this frustrating

Beta Distribution

- Can distribute beta builds to a limited number of iOS devices*
- Consider TestFlight

  https://testflightapp.com/

* 100 - 500, depending on your specifics

Questions?

plinskey@gmail.com

@plinskey

Thanks!