

# EECS 395

Day 27

Nathan Nichols

# Today

---

- Rap session
- Reminders
- Android
- Uncanny Objective-J

# Rap Session

---

- I have talked to some people about webservices, devices, etc.
- Please let me know if I can help

# Reminders

---

- Presentations on Monday and Wednesday
- You will talk for 5ish minutes
- 3ish minutes for questions / discussion
- I will call on people for discussion, etc.
- 5% of your grade is demonstrations
  - Not huge, but be credible
- *Send me your apps before class*

# Monday

---

- Dan
- Jeff
- Eric + Isaac
- David
- Patrick McNally
- Ted
- Jonathan

# Tuesday

---

- Jason
- Ben
- Michelle
- Jake
- Felix
- Patrick Wong
- Greg

# Plotting

---

- Today will be very similar to the web apps lecture
- This is the final lecture day
- It probably won't take the full day

This is the way the world ends  
Not with a bang but a whimper.

# Plotting

---

- Monday & Wednesday are presentations
- Friday is class introspection
- Wednesday is finals!

# Android

---

- I spent yesterday playing with Android and making these slides
- So I'm pretty much an expert

# Android First Impressions

---

- It's very easy to get Hello, World up and running
  - (One command line bit)
- Emulator is a lot better than on iPhone
  - Simulate getting calls, check email, etc.
- Why is Apple so careful to call the iPhone thing a Simulator and not an Emulator?
- Because it's a simulator, not an emulator!

# Simulator Aside

---

- The Intel runs on an ARM processor, not an x86 like every Mac runs
- You can't run code compiled for an ARM processor on an x86 or vice versa
- When you build your app for Simulator, it compiles for x86, and then copies it to the Simulator
- When you build your app for Device, it compiles it for ARM, and then copies it to the Device
- This is one reason you can't run apps from the store in the Simulator

# Simulator Aside

---

- They didn't bother to write a fake Phone call app, etc.
- The Android Emulator, on the other hand, is an actual Emulator
- It's got a little piece in there that says "When the OS executes ARM Process 0x2161, actually do x86 Process 0x17162, and when they ask for ARM 0x172621, do 0x257827"
- (Roughly)
- This means that after writing their Dialer app for the actual device, it works on the emulator for "free"

# Back on Track

---

- For iPhone, you use XCode, Interface Builder, and Instruments
- You guys are all going through your code with Memory Leaks Instrument like a fine-toothed comb, right?
- For Android, you use Eclipse, Eclipse, and I don't think there is an Instruments equivalent
- Strictlier speaking, you download the ADT Eclipse plugin and use that
- Strictliest speaking, the ADT plugin is just a wrapper for a bunch of command line utilities, so you can really use whatever IDE you want

# Eclipse

---

- How many people have used Eclipse?
- It's probably the premier Java IDE
- Originally from IBM
- Written in Java
  - Meta!
- Like every good open source project, it's sprawling and has a big plugin architecture, all this debugging stuff, GUI builder, etc.

# Eclipse

---

- Apple already had XCode/IB for Mac developers
  - Adopted it for iPhone
- Google did not have their own IDE, so they wisely decided to piggyback off of Eclipse
- Added the simulator, Android project templates, etc.
- Let's look at some code
- Remember, Android is more afraid of you than you are of him!

# Hello, World!

---

- Remember Android apps are all written in Java

# Hello, World!

---

```
public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```

```
- (void)loadView {
    [super loadView];
    UILabel *label = [[UILabel alloc] init];
    [label setText:@"Hello, world!"];
    [self setView:label];
}
```

# Android

---

- So adding controls programmatically is very similar to iPhone
- But it's still usually a bad idea
- How do we lay out controls graphically?

# Android

---

- How do you wire things up?
- I don't think you can, and certainly not with a bizarre and awesome blue line like you get in IB
- You can set IDs and look them up by that

# XML is XML

---

- Hey, what's the X in XIB stand for?
- XML!
- If Interface Builder's files are stored as XML, and Android file's are stored as XML, maybe they look the same?
- Maybe we could even write a converter?

# XML is not XML

---

- Nope, not even close
- Why?
  - Androids XML layout is very similar to HTML
    - “Build a TextView, set its text to ‘Hello, world!’ and put it under this other view.”
  - IB’s XIB are freeze-dried
    - The XIB isn’t an instruction for creating the UILabel, it *is* the UILabel
    - In some deep and nebulous way

# Other UI Controls

---

- [file:///Users/nate/Programming/android-sdk-mac\\_x86-1.5\\_r2/docs/guide/tutorials/views/index.html](file:///Users/nate/Programming/android-sdk-mac_x86-1.5_r2/docs/guide/tutorials/views/index.html)

# Tables in Android

---

- Let's recall how iPhone does tables
  - UITableView, UITableViewCell
  - UITableViewController sits on top of UITableView
  - UITVC's delegate and datasource are in charge of responding to information requests from the tableview

# Tables in Android

---

- Android is actually really similar
- It has a concept of *adapters*
- “An Adapter object acts as a bridge between an AdapterView and the underlying data for that view. The Adapter provides access to the data items. The Adapter is also responsible for making a View for each item in the data set.”
- What does this sound like in the iPhone world?
- The delegate and datasource!

# Tables in Android

---

- Android has `getCount()`, iPhone has – `tableView:numberOfRowsInSection:`
- Android:
  - `getView(int position, View convertView, ViewGroup parent);`
- iPhone:
  - `-cellForRowAtIndexPath:`

# Android Conclusion

---

- All of the design considerations (do one thing well, lots of polish, don't expect user to read documentation, etc.), we've talked about are the same for Android
- A lot of the design patterns (MVC, delegate, event processing, etc.) are the same for Android
- You will be able to pick it up pretty easily, especially because you won't have to learn Objective-C along the way
- If you're interested in doing an Android app, go for it

# Objective-J

---

- <http://www.youtube.com/watch?v=uyFq-oZKwlo>
- <http://280slides.com/>
- What in the world is going on here?

# Objective-J

---

- Normally, web sites work like this:
  - Page is HTML, which is a tree (and is called the DOM)
  - CSS modifies how that HTML is rendered in the browser
  - Javascript modifies the DOM (which changes the look of the page)
  - AJAX communicates with the server without leaving the page
  - The server is running PHP or something, and responds to the AJAX

# Objective-J

---

- This requires knowing at least three different technologies
- Browsers (in)famously render things differently than other browsers
- Why are pages represented in HTML to begin with?
  - It's a markup language
  - Easy to parse, etc.
    - Lisp/Scheme aside
  - Used to make sense

# Objective-J

---

- Building a Powerpoint clone with HTML/JS/CSS is painful and weird
- We're trying to build a desktop application that runs in the browser, p'raps we should use desktop technology and paradigms?
- What's the premier desktop development programming language and frameworks?
- Objective-C and Cocoa, of course!

# Objective-J

---

- Again, this is almost unbelievable
- The people at 280 North have built an Objective-C compiler
  - Compiles to Javascript
  - *Written in Javascript*
- This means you can write Objective-C/J code
- When a user visits your site, Javascript downloads the Obj-J code, compiles it to Javascript, then executes that Javascript
- What good is Objective-C without NSString, UILabel, etc.?

# Cappuccino

---

- So they *also* wrote Cappuccino, which is the Objective-J version of Cocoa
- Labels, buttons, menus, etc.
- You write in Objective-J, and it takes care of the HTML/JS/AJAX/CSS etc. for you
- *You can even use XIBs you built in Interface Builder*
- And what you write is *really* close to Objective-C
- (From Theobroma Cocoa)

```

@implementation ApplicationController : CPObject { }

- (void)applicationDidFinishLaunching:(CPNotification)note
{
    theWindow = [[CPWindow alloc] initWithContentRect:\
                CGRectMakeZero()
                styleMask:CPBorderlessBridgeWindowMask];

    contentView = [theWindow contentView];
    var label = [[CPTextField alloc] initWithFrame:CGRectMakeZero()];
    [label setStringValue:@"Hello World!"];
    [label setFont:[CPFont boldSystemFontOfSize:24.0]];
    [label sizeToFit];
    [label setAutoresizingMask:CPViewMinXMargin | CPViewMaxXMargin];
    [label setFrameOrigin:CGRectMake(100,100)];

    [contentView addSubview:label];

    [theWindow orderFront:self];
}
@end

```