Praktikum Entwicklung von Mediensystemen mit iOS

Wintersemester 2012 / 2013

Prof. Heinrich Hußmann, Dr. Alexander De Luca, Fabius Steinberger

Media Informatics munich

17.10.2012

Today

- Organization
- Introduction to iOS programming
- Assignment 1

Organization

- 6 ECTS
- Bachelor: Vertiefendes Thema
- Master: Gruppenpraktikum
- Wednesday 16 - 18, Amalienstr. 17 A107
- http://www.medien.ifi.lmu.de/lehre/ws1213/pem/

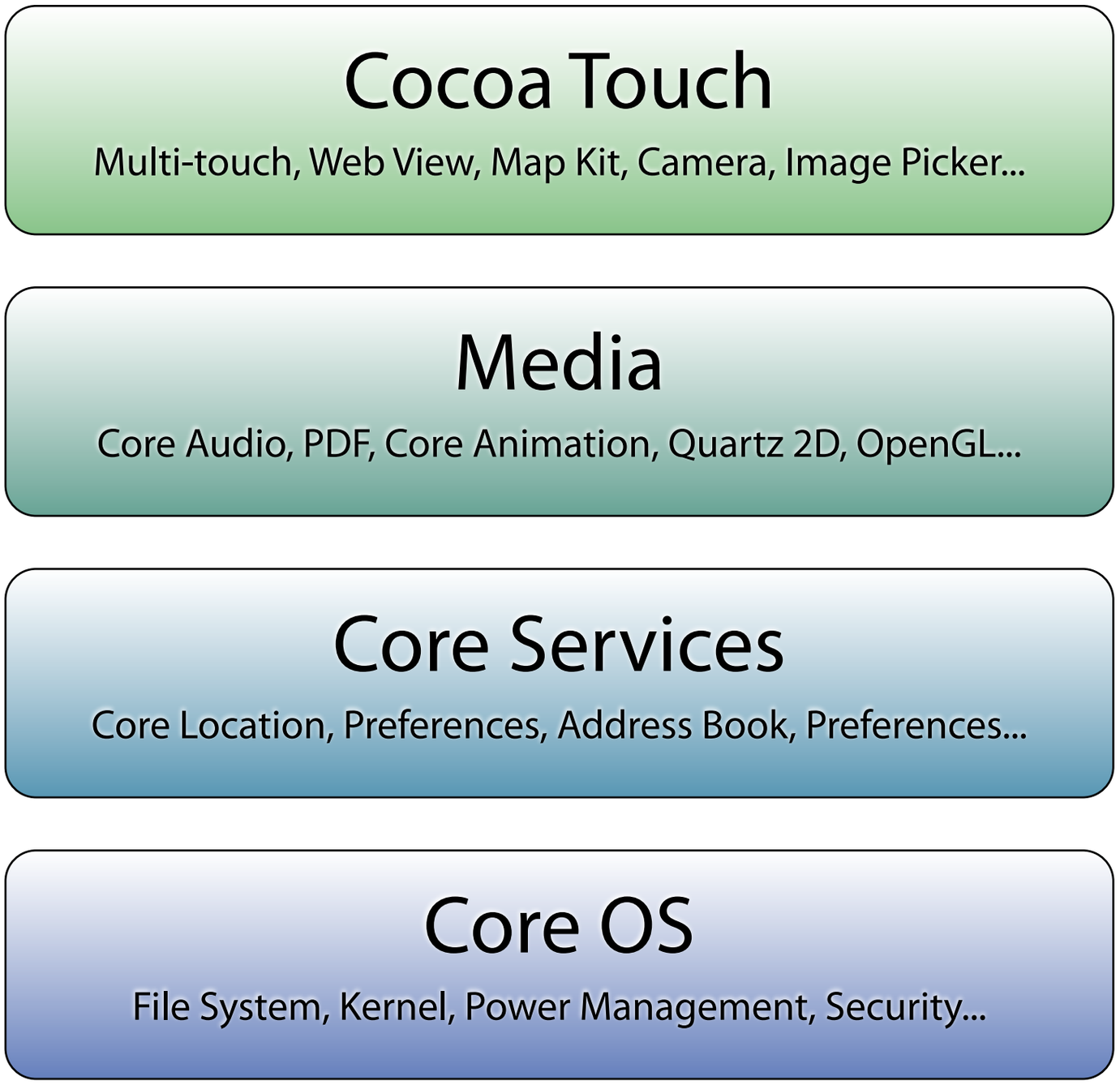
Roadmap

- October, November: weekly lectures and individual assignments
- December, January: app development in teams, milestone meetings and presentations
- February: final presentation (probably 6.2.2013)

iOS

- Mobile operating system by Apple for iPhone, iPad and Apple TV
- Based on Unix, derived from OS X
- 400 million iOS devices sold, 25% share of smartphone market, 65% share of mobile web consumption
- Latest release: iOS 6.0 (September 2012)

Layers of iOS



User input

- GUI controls: buttons, sliders, switches
- Multi-touch gestures: tap, pinch, rotate, swipe, pan
- Accelerometer: shaking, rotating

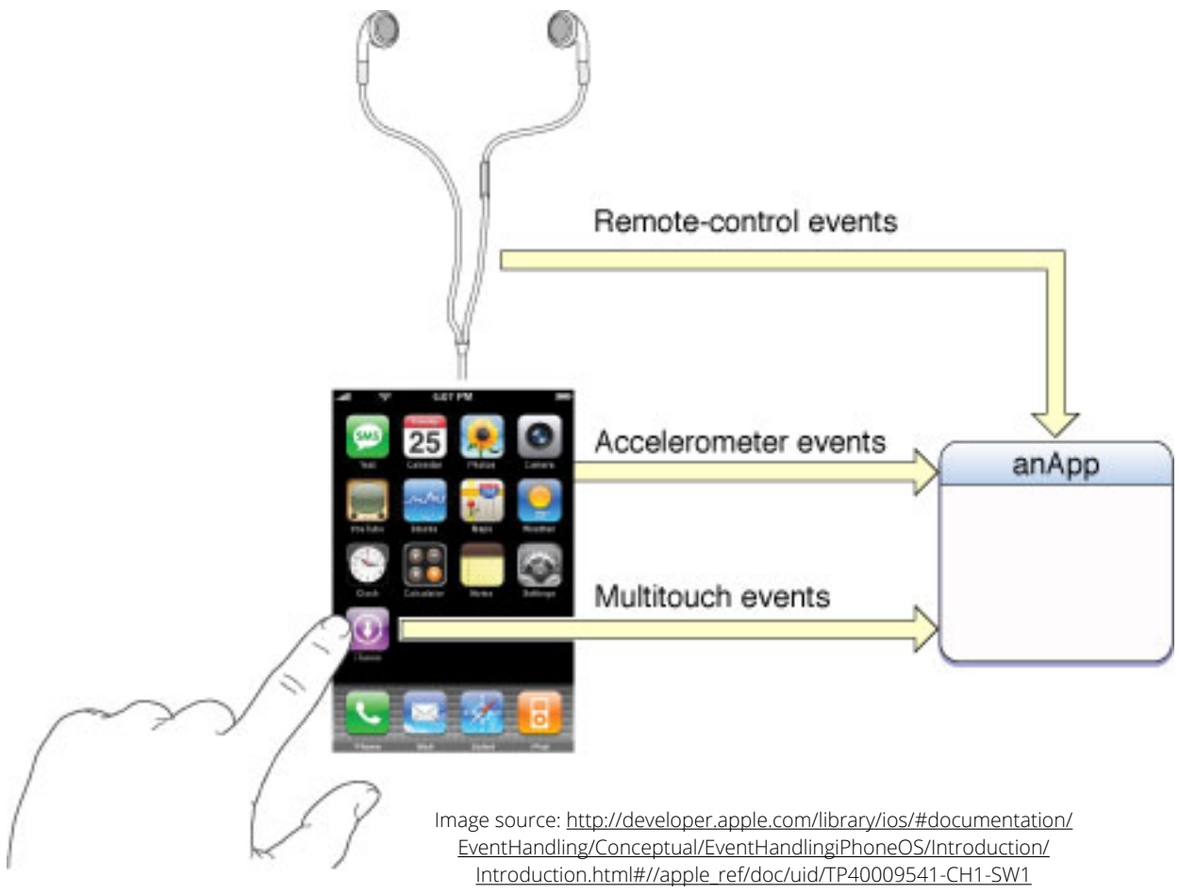


Image source: http://developer.apple.com/library/ios/#documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/Introduction/Introduction.html#//apple_ref/doc/uid/TP40009541-CH1-SW1

iOS Development

Image source: https://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/Introduction.html

Development Environment

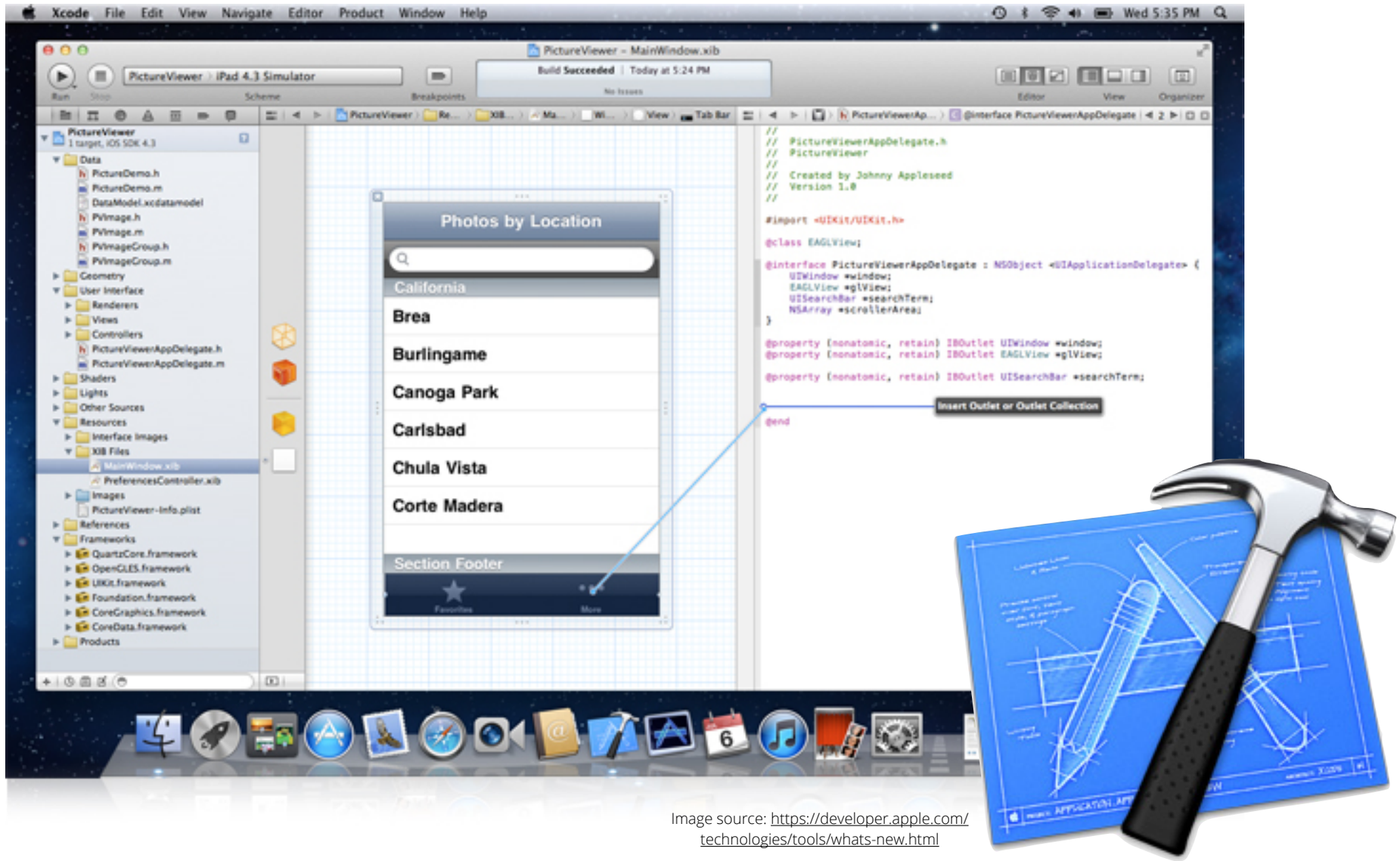


Image source: https://developer.apple.com/technologies/tools/whats-new.html

XCode

XCode

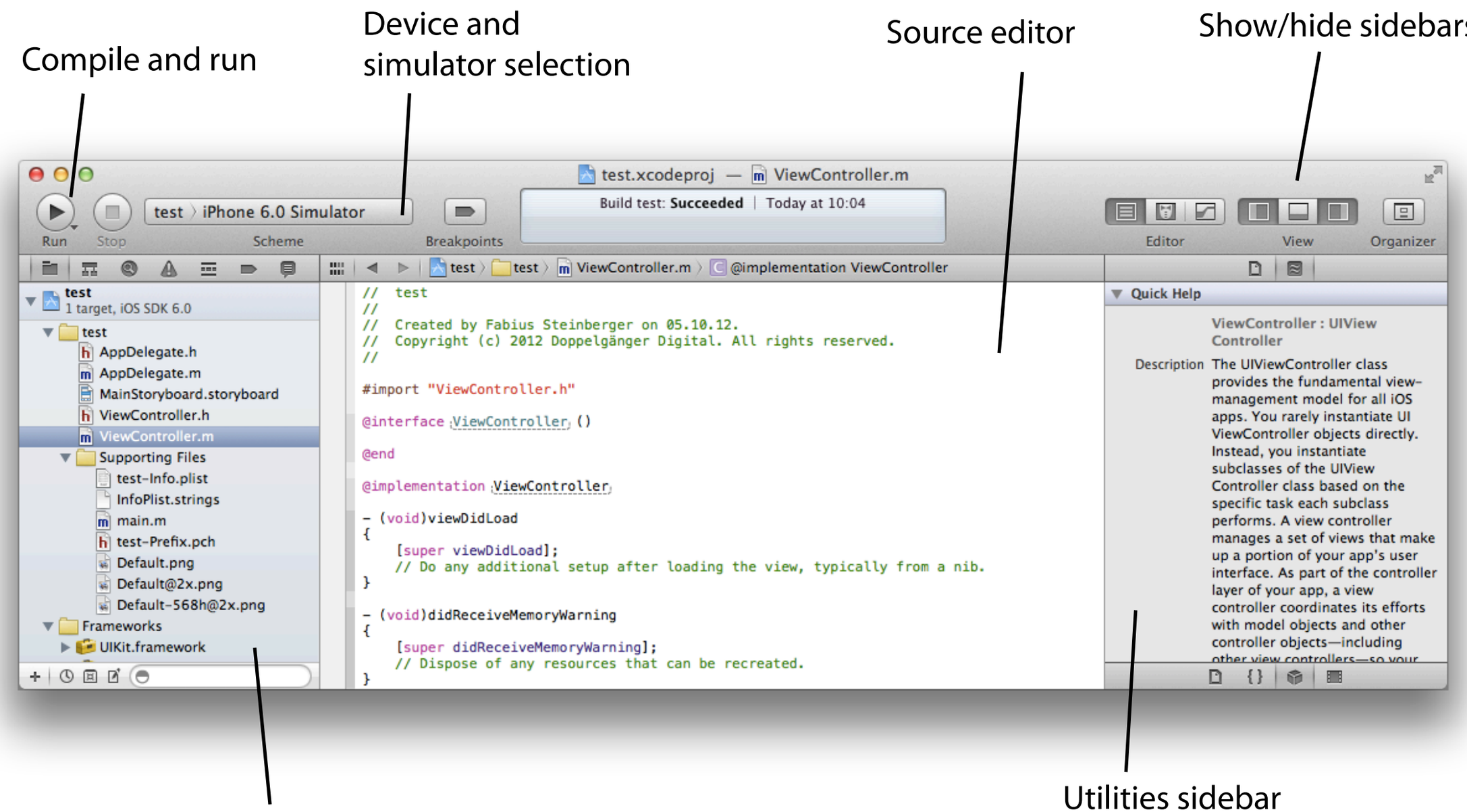
- Source editor: code completion, syntax highlighting, context-sensitive information

- Interface builder: UI elements library and inspector, split editor to connect UI with code, Storyboards

- Compiler: C, C++, Objective-C
- iOS Simulator: run and test apps on a Mac

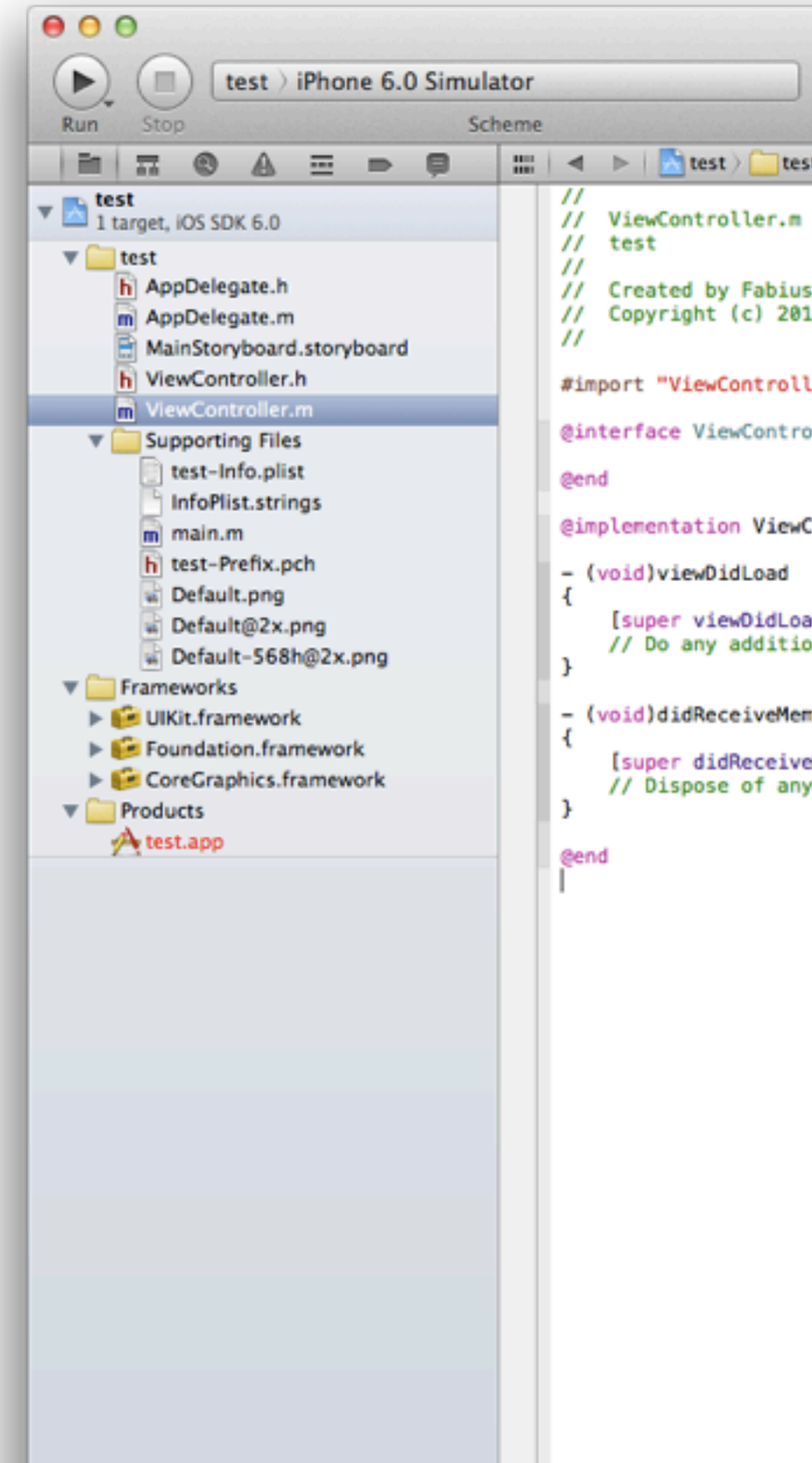
- More: refactoring, version control, debugging, analysis (https://developer.apple.com/technologies/tools/)

XCode


Compile and run
Device and simulator selection
Source editor
Show/hide sidebars
test.xcodeproj — ViewController.m
Run
Stop
Scheme
Breakpoints
Build test: Succeeded | Today at 10:04
Editor
View
Organizer
test 1 target, iOS SDK 6.0
AppDelegate.h
AppDelegate.m
MainStoryboard.storyboard
ViewController.h
ViewController.m
Supporting Files
test-Info.plist
InfoPlist.strings
main.m
test-Prefix.pch
Default.png
Default@2x.png
Default-568h@2x.png
Frameworks
UIKit.framework
Quick Help
ViewController : UIView Controller
Description
The UIViewController class provides the fundamental view-management model for all iOS apps. You rarely instantiate UI ViewController objects directly. Instead, you instantiate subclasses of the UIView Controller class based on the specific task each subclass performs. A view controller manages a set of views that make up a portion of your app's user interface. As part of the controller layer of your app, a view controller coordinates its efforts with model objects and other controller objects—including
File navigator
Utilities sidebar (API info)

Contents of an XCode project

- Source code files (.h and .m)
- Interface files (.storyboard and .xib)
- Libraries (.framework)
- Resources, e.g. images (.png)
- App configuration file (Info.plist)



Objective-C

- Language for programming iOS and Mac apps, also used by Apple to create much of OS X, iOS, APIs
- Superset of C
- Object-orientated

Short introduction: https://developer.apple.com/library/mac/#referencelibrary/GettingStarted/Learning_Objective-C_A_Primer/_index.html

Detailed introduction: https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html

Elements of Objective-C

Java	Objective-C
MyClass.java	Header.h Implementation.m
Methods and method calls	Methods and messages
Attributes, setters, getters	Properties, instance variables
Constructor	Initializer
Interface	Protocol
Garbage Collection	Automatic Memory Management (ARC)

Methods

- Definition (in .h):

```
- (void) doSomething;
```

```
- (void) doSomethingWithA: (NSString *) a
andB: (NSString *) b;
```

- Implementation (in .m):

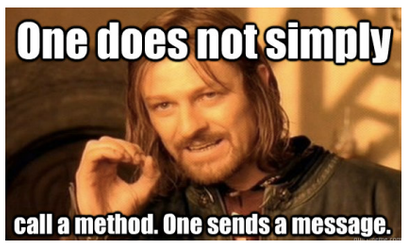
```
- (void) doSomething {
  // do something
}
```

```
- (void) doSomethingWithA: (NSString *) a
andB: (NSString *) b {
    // do something with a and b
}
```

- Method call (“message”) (in .m):

```
[self doSomething];
```

```
NSString* a = @"a";
NSString* b = @"b";
[self doSomethingWithA:a andB:b];
```



Instance Variables (“ivars”)

- Like private attributes in Java
- Definition (in .h): `NSString* _name;`
- Use (in .m):

```
_name = @"Max";
labelText = _name;
```

- You don’t have to use the underscore (_), but it’s good practice. Otherwise you accidentally mix up ivars and properties (see next slide).

Properties

- Auto-creation of an instance variable (private) as well as a getter and setter (public)
- Definition (in .h):

```
@property(strong, nonatomic) NSString *name;
```

`strong/weak:` refers to ownership. Always use strong except for properties that point to a parent.

`nonatomic/atomic:` use nonatomic to avoid multi-threading issues.

- Using getters/setters (in .m):

```
[self setName:@"Max"];
self.name = @"Max";

NSString *labelText = self.name;
labelText = [self name];
```

`self.name:` this syntax does NOT access the variable itself. It's a getter/setter, just like the other syntax.

- Using the instance variable (in .m):

```
_name = @"Max";
labelText = _name;
```

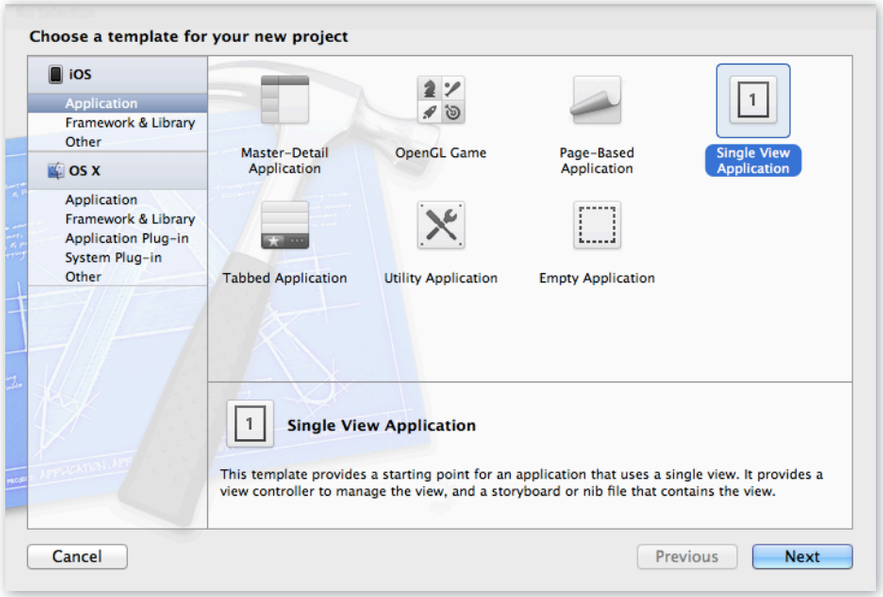
`_name:` Use this instance variable in custom setters/getters and in init-methods only. In any other case, use the getter/setter.

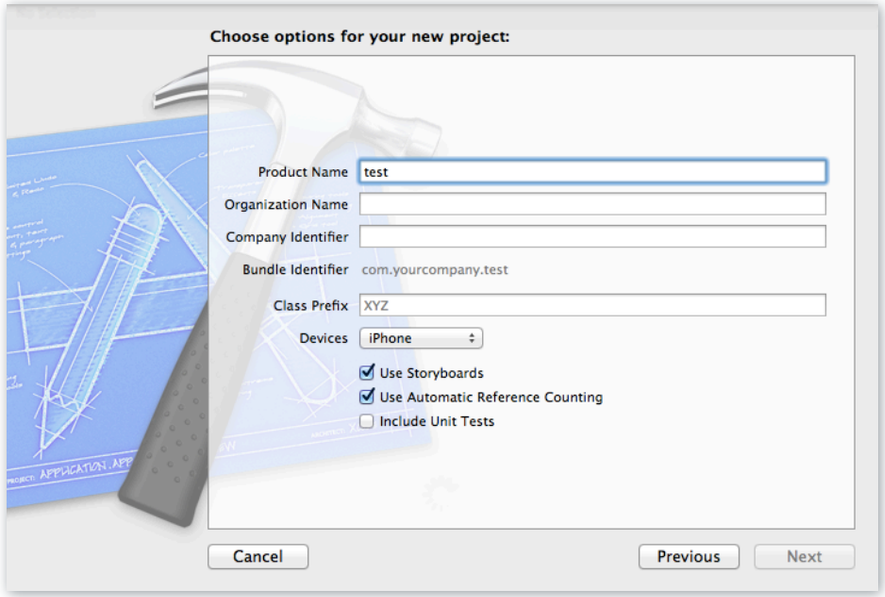
Object Initialization

- Object: `MyClass *myObject = [[MyClass alloc] init];`
- Object with parameter: `MyClass *myObject = [[MyClass alloc] initWithParameter: parameter];`
- String:
```
NSString *hello = @"Hello";
NSString *helloWorld = [NSString stringWithFormat:@"%@ World", hello];
```
- Array:
```
NSArray *colors = @[@"Green", @"Red", @"Yellow"];
NSMutableArray *mutableColors = [@[@"Green", @"Red", @"Yellow"] mutableCopy];
```
- THERE ARE NO NULL POINTER EXCEPTIONS. So if your app compiles but doesn't work properly, make sure your objects aren't `nil`.

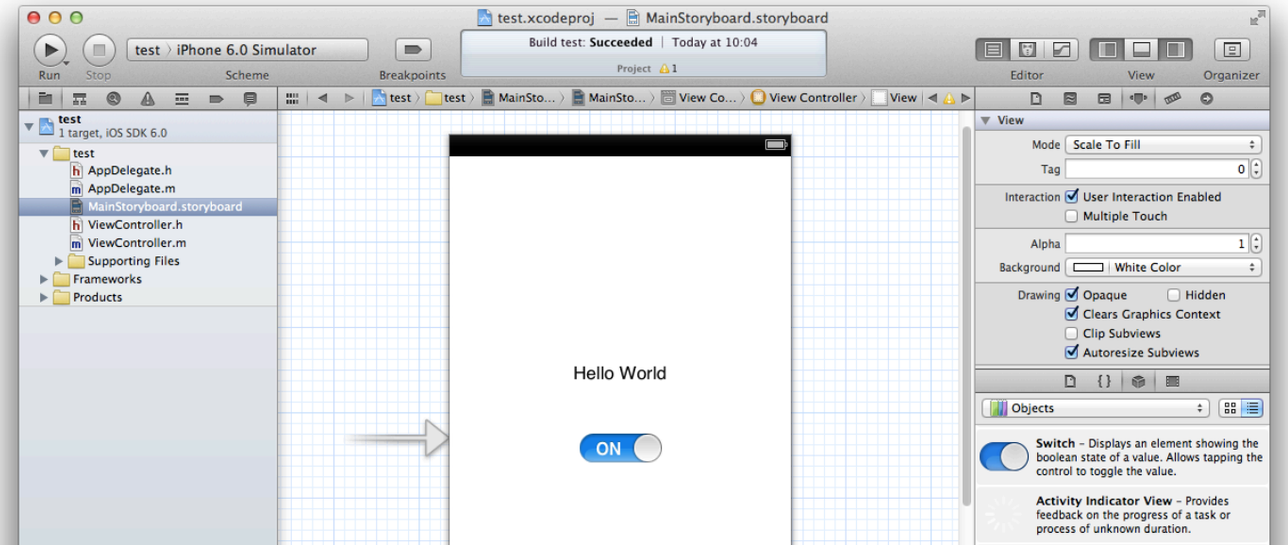
Hello World

- New XCode Project: Single View Application





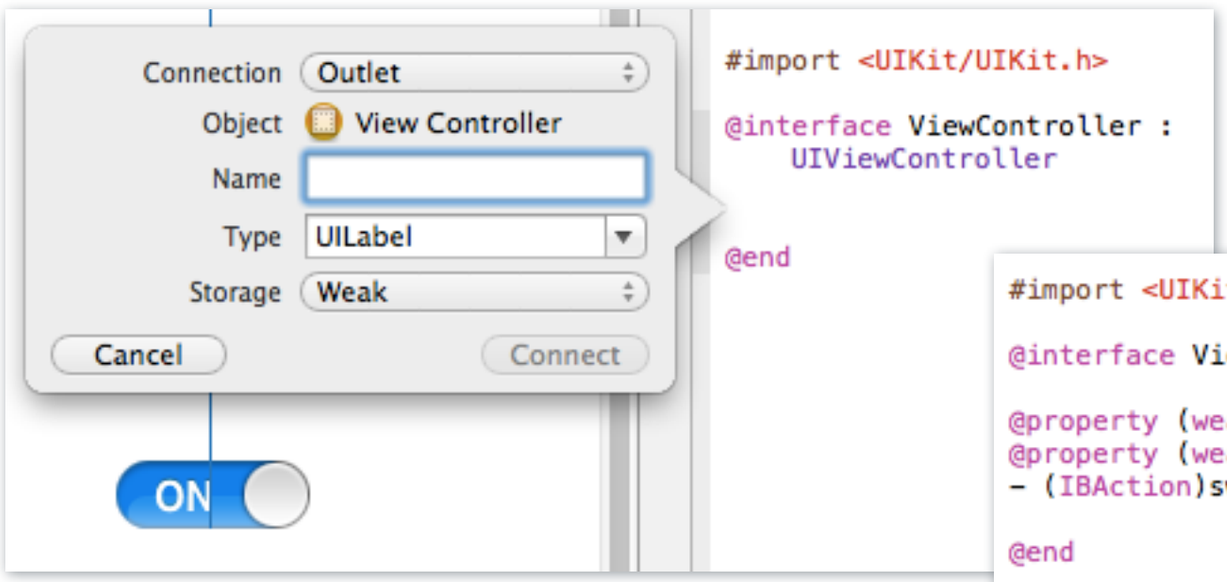
- In the storyboard, drag a text label and a switch onto the screen



Hello World

- Open the assistant editor 

and ctrl-drag the text label into ViewController.h. Enter a name and click Connect. You now have access to the UI element in your code. Do the same for the switch.
- Again, ctrl-drag the switch into the code. This time, select Action instead of Outlet. Enter a name and click Connect. You now have a listener method that is called by the OS when the user changes the value of our switch.




```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

@property (weak, nonatomic) IBOutlet UILabel *myTextLabel;
@property (weak, nonatomic) IBOutlet UISwitch *mySwitch;
- (IBAction)switchChangedValue:(id)sender;

@end
```

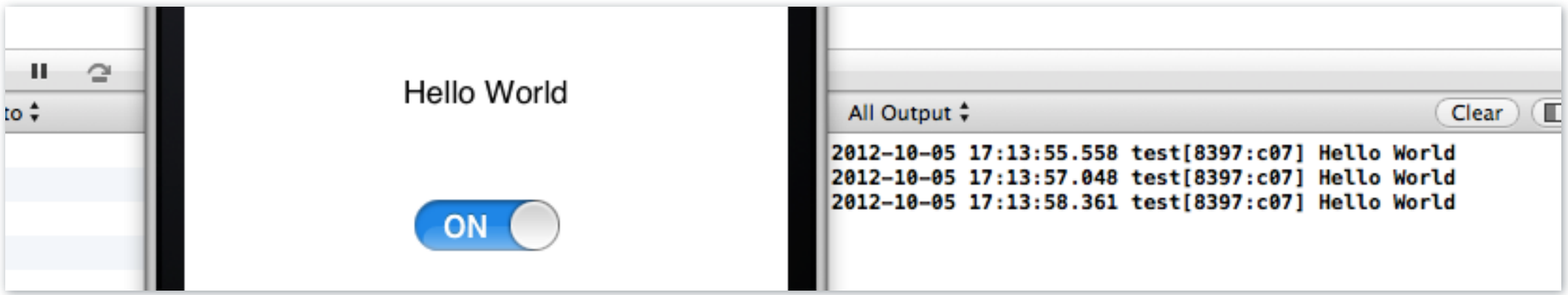
Hello World

- Close the assistant editor and go to ViewController.m. Complete the IBAction method:

```
- (IBAction)switchChangedValue:(id)sender {

    if (self.mySwitch.on) {
         self.myTextLabel.text = @"Hello World";
         NSLog(@"Hello World");
     } else {
         self.myTextLabel.text = @"";
     }
}
```

- Open the debug area and run the code.



UIViewController

- One of the most important classes in iOS programming
- You have to subclass UIViewController when creating a new screen
- Provides methods for managing the view hierarchy throughout its life cycle and for reacting to events (also great for debugging), e.g.

```
– viewDidLoad:
– viewWillAppear:
– viewDidAppear:
– viewWillDisappear:
– viewDidDisappear:
- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
duration:(NSTimeInterval)duration;
```

- For more see http://developer.apple.com/library/ios/#documentation/uikit/reference/UIViewController_Class/Reference/Reference.html

App Delegate

- Every app must have an App Delegate.
- Provides methods for managing the app throughout its life cycle (also great for debugging), e.g.

```
- application:didFinishLaunchingWithOptions:
– applicationDidBecomeActive:
– applicationDidEnterBackground:
– applicationWillEnterForeground:
– applicationWillTerminate:
```

- For more see: http://developer.apple.com/library/ios/#documentation/uikit/reference/UIApplicationDelegate_Protocol/Reference/Reference.html
- There are lots of protocols (often named Delegate), e.g. for managing the keyboard, table views, date pickers.

Top 3 Resources

1

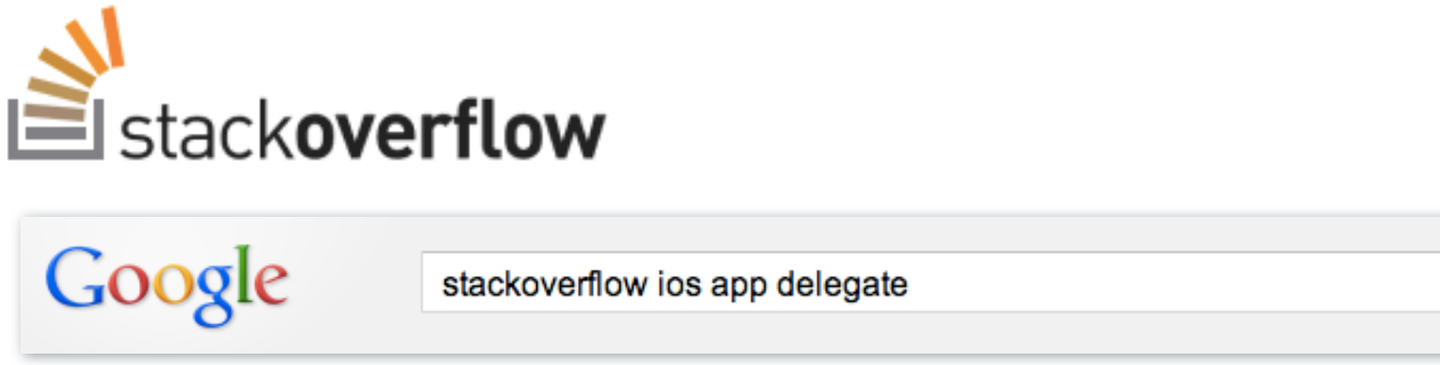
or https://developer.apple.com/library/ios

2

RAYWENDERLICH
Tutorials for iPhone / iOS Developers and Gamers

http://www.raywenderlich.com/tutorials

3



Assignment 1

- Individual assignment
- Get to know XCode and Objective-C
- Due next Wednesday 12:00, upload to Uniworx

- Questions?