Let’s make an App!

Tianyu Liu

Why is iOS awesome?

- It’s Objective-C and You already know C!
- VERY robust API (and StackOverflow)
- Huge audience with little effort
- Impress your friends

Objective-C

Syntax

Similar to C

Declaring variables:

```
int i = 0;
int j = 1;
```

Arithmetic:

```
int k = i + j;
```

Similar to C

Loop:

```
for (int i = 0; i <= n; i++) {
    NSLog(@"Hello World!”);
    // NSLog is basically printf
}
```

Condition:

```
if (i == 1) {
    NSLog(@"i is 1");
} else if (i == 2) {
    NSLog(@"i is 2");
} else {
    NSLog(@"i is neither 1 or 2");
}
```

Something weird

Include:

```
#import "cs50.h"
```

Strings:

```
NSString *str = @"Hello World”;
```

Printing to console:

```
NSLog(@"Hello World!”);
```

NSEverything… :

```
NSString, NSArray, NSDictionary…
```

Something weird

~~Function~~ Method declaration:

```
- (void)sayHelloWorld {
    NSLog(@"Hello World!”);
    return;
}

- (int)addInt: (int)intA with:(int)intB{
    return intA + intB;
}

- (returnType)methodName: (parameterType)parameterA
moreMethodName:(parameterType)parameterB
```

Something weird

~~Function~~ Method call:

```
[self sayHelloWorld];
[self addInt:10 with:2];
```

Object-Oriented Programming

What is an object?

- It’s like a struct, but better ;)
- It has methods and properties
- In fact, every view you see in an iPhone App is an object.

Why is it Awesome?

- Much better clarity for complicated logic
- Efficient code reuse
- Clear hierarchical structure
- It’s flexible and fun

Class and Instance

- Class:
  - Template for objects
  - Defines what a object should have
  - Like a struct declaration
- Instance:
  - A specific instance of an object
  - Created based on class declaration

Quick Example

- A Cat

A cat Class

- CatClass:
  property:
    color
    age
    breed

  method:
    chaseMouse
    eatFish
    meow

A cat Object

- CatClass:
  property:
  color
  age
  breed

  method:
  chaseMouse
  meow

- Cat (color = “black”, age = “2”, breed = “Persian”)

- Cat.meow()
  “Hello World! Meow!”

An Objective-C cat

```objc
#import <Foundation/Foundation.h>

@interface Cat : NSObject {
    NSString *color;
    NSString *breed;
    int age;
}

- (void)meow;
- (bool)chaseMouse:(Mouse*)aMouse;

@end
```

```objc
#import <Cat.h>

@implementation Cat

- (void)meow{
    NSLog(@"Hello World!”);
};

@end
```

An Objective-C cat

- Declaration:

```
Cat *myCat = [[Cat alloc] init];
```

- Assign property:

```
myCat.age = 2;
```

- Call method:

```
[myCat meow];
```

Event Driven Programming

Good news, you've done that before

```
google.maps.event.addListener(marker, "click", function(){

});
```

Event-driven

- Functions are called by event
- Much for flexible than ‘one-shot’ programs
- Not a new concept ;)

Event-driven in Objective-C

- Target-Action (like pressing a button)
- Delegation and Protocol
- Notification

Delegation and Protocol

Delegation and Protocol

- Events that Apple gives you :)
- Typically, these events are a bit hard to define by yourself
- A very robust mechanism, can be handled by any class

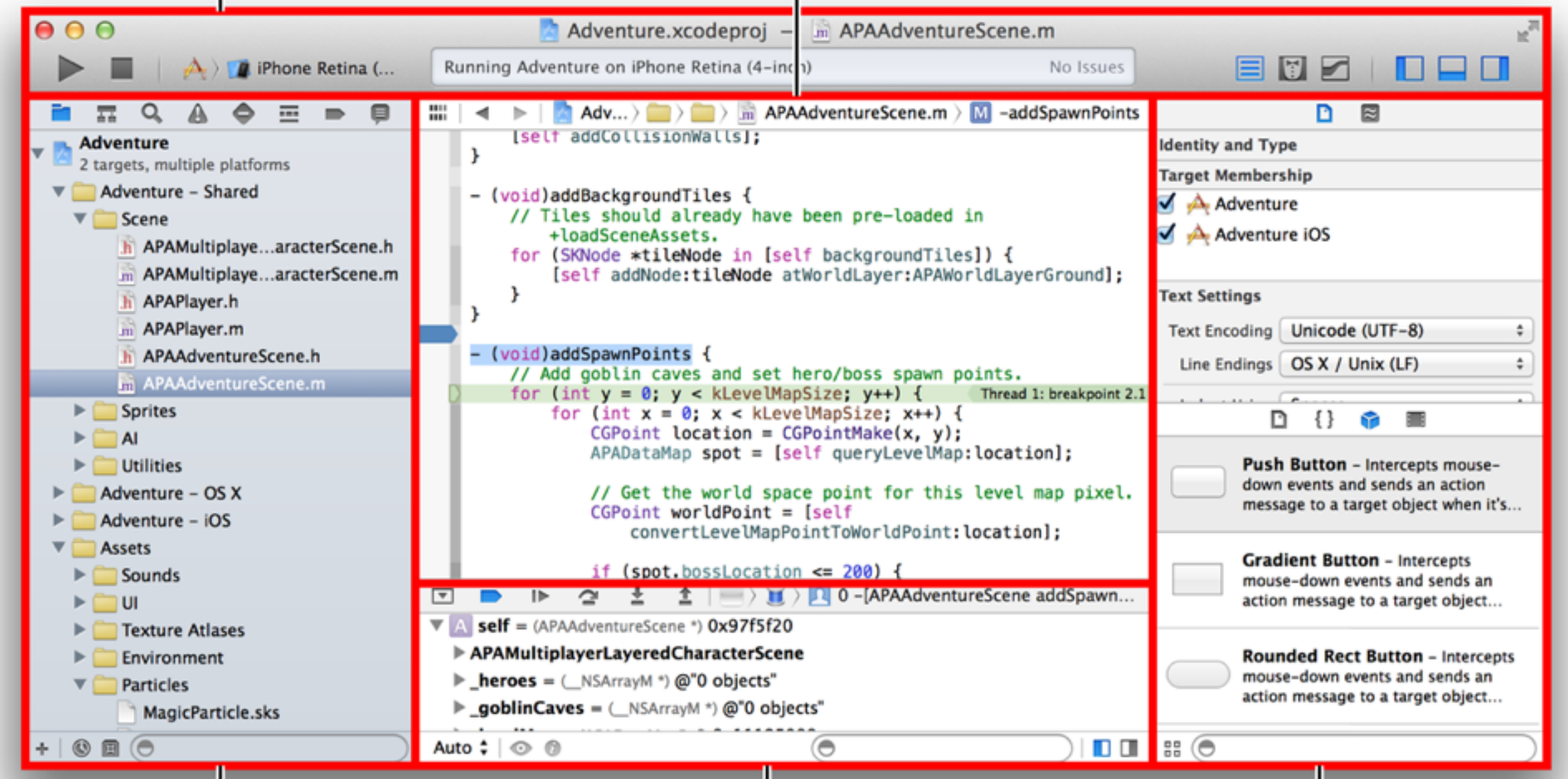
Very quick example

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:
(NSIndexPath *)indexPath{
     NSLog(@“This protocol is called!”);
}
```

Let’s do some Xcode

Toolbar

Editor area



Navigator area

Debug area

Utility area

Awesome Resources

code school

learn by doing

AppCoda

treehouse™

WWDC2012

See, that wasn't so hard. So what is Mr. Higgie doing here you ask? Well, **NSLog** is a C function that handles printing a **NSString** to the console. It's useful for debugging your application. We can create a new **NSString** using the syntax **@""**. **NSLog** can also create formatted strings, like Mr. Higgie is doing above. When we run this line of code, it will print this to the console:

```
> Hello there, Eric.
```

See how it replaces the **%@** with the value from **readersName**? That **%@** is just a placeholder, essentially saying "I'm going to be replaced by a variable that is also passed into **NSLog**". Those placeholders are replaced in the order they are defined. So, if we wanted Mr. Higgie to say both a first and last name, we'd do this:

```
NSString *firstName = @"Eric";
NSString *lastName = @"Allam";

NSLog(@"Hello there, %@ %@.", firstName, lastName);
```

And get this log:

Thank you