Greg Tropino

Chapter 3 video responses

Section 1:

1.3) It’s important to know how older versions work because if you need to update someones app that was originally developed you need to know how to read older variations to know what to convert those lines into. It’s important to keep up with new features as your app may not work on a newer version and you will need to figure out how to keep them up to date, or because they may be making your life easier.

Section 2:

2.1) It became the language to learn due to the mac OS and ios being developed in objective c. Since it is built upon that language there is no way to easily remove it.

2.3) You might want to build in one version but deploy in an older because you may be on a beta version of the upcoming update but still want people on older versions to be able to run your program.

Section 3:

3.1) #import <Foundation/Foundation.h>

int main(int argc, const char \* argv[])

{

@autoreleasepool {

int seconds = 60;

int minutes = 60;

int hours = 24;

int days = 365;

int secondsinayear = seconds \* minutes \* hours \* days \* 10;

NSLog(@"There are %i seconds in a year.", secondsinayear);

}

return 0;

}

3.2) int main(int argc, const char \* argv[])

{

@autoreleasepool {

int day = 6; // 0 is sunday and 6 is saturday, 1-5 is mon-fri

if ( (day == 0) || (day == 6)) {

NSLog(@"Have a nice weekend");

} else {

NSLog(@"I hope you're having a good week!");

}

}

return 0;

}

3.3) int main(int argc, const char \* argv[])

{

@autoreleasepool {

int hurricaneCategory = 5;

switch (hurricaneCategory) {

case 0:

NSLog(@"Hurricane, what hurricane?");

break;

case 1:

NSLog(@"The hurricane is a category 1!");

break;

case 2:

NSLog(@"The hurricane is a category 2!");

break;

case 3:

NSLog(@"The hurricane is a category 3, hope you're hidden!");

break;

case 4:

NSLog(@"The hurricane is a category 4, praying wouldn't hurt!");

break;

case 5:

NSLog(@"The hurricane is a category 5, you'll be seeing Dorothy soon!");

break;

}

}

return 0;

}

3.4) 20 seconds

3.5) arithmetic operators, they include the basic mathematical functions like addition, subtraction, division and multiplication. For example: result = a + b;

comparison operators, they check for equality, less than, greater than, or not equal to. For example: if ( a == b )

Modulus operators, used to calculate the remainder, mainly for the purpose of determining whether or not something is odd or even. For example: int remainder = 2003 %4

Increment / Decrement operators, to increase or decrease an integer amount by 1, short hand looks like: a++;

Prefix / postfix operators, this is the computers interpretation of the order of operations for specific commands like how a++ and ++a normally mean the same thing but if you use it in NSLog methods it can do the order in a way you didn’t intend.

Ternary operator, similar to an if statement however it can work with 3 variables instead. It is used to typically replace an if else statement.

3.6) When you are searching your emails for a specific subject name or body of text inside of the email it will keep searching all your emails until its done but will only display the ones that had the text that you were searching for.

3.7) A portion of related code that is combined together and given a name. A function prototype is used to describe the function. The purpose of a function is to perform an actual task where as a function prototype is to somewhat declare that the function actually exists. The first function that is automatically called is the main function, other functions must be called within the main function or be declared outside of it.

Section 4:

4.1) int, float, double, char and BOOL. Apple made the set of classes to make writing certain types of code a lot easier if you were going to try to display a long list of text and dates, as opposed to having to do each character one at a time.

4.4) The scope of a variable is the lifetime of that variable.

4.5) Enum allows you to restrict the values that you could assign to a variable.

4.6) You would define your own data type to make your code a little easier to read by shortening up the lines itself.

4.7) Import will include all associated code with the file listed as if the code was pasted in your current project. Define is available to create a value and give it a specific name, similar to making a variable static and global. If DEBUG is for pulling out specific code if you are NOT in debug mode. You would use import to bring in useful libraries like math functions, you would use define to have variables that need to stay the same (like the amount of pixels per square inch), and use if debug to maybe display information you wouldn’t want normally displayed in the release mode, but something that would be helpful to have displayed if you were in debug mode.

4.8) There is actually no defined string in c or objective c, however it is described in objective c built into the foundation framework listed as follows, NSString \*message = @”Hello”;. NSString is the variable type, the \* tells it to create a space for a NSString variable type, and the variable name is listed after. The = is an assignment operator, and the @ sign is used to tell the compiler that its an object with the variables being listed within quotations followed by a semi colon to end that line of code.

Section 5:

5.1)

Attributes: Height, Weight, Gender, Age

Behaviors: Typing, walking, learning

? Instantiate a single object representing yourself as a member of this class.

5.2) A pointer holds the address that the object is stored, it helps allow for more efficiently modifying and passing of that object as you are not passing the object itself, but the address for the object. A primitive stores the value or a particular data type where as the pointer just stores the location of the object, not the object itself.

5.3) The main difference is that if you start calling multiple methods it displays them in a much more readable manner in objective c where you just pass the arguments without always seeing the method name in most other languages.

5.4) A class method will effect that entire class where as an instance method will only effect that particular object or instance you have.

Section 6:

6.2) A pointer simply tells where an object is located within a block of memory, when you are creating a object or releasing it utilizes the pointer to figure out what block of memory to access or destroy.

6.3) It allocates a spot of memory large enough to fit that object, initialize and then return the address of that memory. You wouldn’t be able to control the allocation and initialization if you just used the new method which is why we are better off avoiding the new method. An objects lifetime is for as long as you don’t release it or when objective c decides to close it.

6.4) The retain count from each object or instance gets added to the “pool” and when the event loop is done it will release each of the retain counts in the entire pool. You should use it when you know someone else will be calling the method.

6.5) New, alloc, retain and copy. You need to remember this to be sure as to what needs to be released and what doesn’t.

6.6) It saves us from having to make the release calls, have dangling pointers that can crash the program or keep track of memory management anywhere as much. The compiler goes through your source code and figures out where the release commands should, or shouldn’t be and automatically places them where they need to be.

6.7) Arc will reclaim memory as it is done with the event loop or when it finds that something is no longer relevant when it has a weak pointer where as garbage collection happens at indeterminable times and can lead to slow downs in the program when it happens unexpectedly. Arc happens during compile time and garbage collection happens during run time. The advantages is one is trying to make your program more efficient to begin with, where as garbage collection attempts to make the running of your program more efficient if running for longer durations.

6.8) You can not use the release or dalloc commands when working with ARC because ARC already performs them.

Section 7:

7.1) The interface (.h) section, it states what methods it has and what properties are available, basically a public listing to everyone, but shows no private functions. The implementation (.m) section holds the actual code of the class and performs the functions showcased in the interface section, kind of like looking under the hood of a car.

Challenge: Create a tweet class for a twitter style app.

7.2)

CHALLENGE: Define what should get passed in and what should get returned by each of your methods in your tweet class above.

7.3) They used to have to declare them in the header file inside a set of curly braces and on top of that they would have to create a accessor method in the header file as well to get and/or set that instance variable. Now we just use the @property in the header file. The only thing that really got obscured is the relationship that you have to maintain, which is as to whether to have a strong reference or a weak reference when coming to objects.

7.4) It is to initialize a new object and make sure it is created in a valid state. You can rely on the standard init method when the initial values for the attributes are not needed to be set at anything, but when you need the values for specific attributes to be set to something at startup, then that’s when a custom initializer may be needed.

7.5) If your object is holding onto a resource you can then have a dealloc method added to the class to make sure its properly closed, but you need to allow ARC to decide as to when to call it.

Section 8:

8.1) You simply need the type (NSString, int, float etc.), the name of the method (with a pointer), and the parameters of the values/objects it will be holding.

NSString \*daysOfWeek[7]= { @”Monday”, @”Tuesday”, @”Wednesday”, @”Thursday”, @”Friday”, @”Saturday”, @”Sunday”};

8.2) Mutable arrays are changeable as to the quantity as the amount of objects that it can contain where as immutable arrays are a fixed quantity once you set them.

NSArray \*myArray = [NSArray arrayWithObjects:@”Monday”, @”Tuesday”, @”Wednesday”, @”Thursday”, @”Friday”, @”Saturday”, @”Sunday”, nil];

NSMutableArray \*myArray = [NSMutableArray arrayWithObjects:@”Monday”, @”Tuesday”, @”Wednesday”, @”Thursday”,

[myArray addObject:@”Friday”, @”Saturday”, @”Sunday”];

8.3)

NSDictionary \*lifeEvents =

[NSDictionary dictionaryWithObjectsAndKeys:

@”Africa”, @”2003”,

@”Cuba”, @”2001”,

@”Mobile Makers”, @”2013”,

@”Buying A House”, @”2009”, nil];

8.4) for (NSString \*events in lifeEvents) {

NSLog(@”A big event was %@ in %@”, events, [lifeEvents objectForKey:events]);

Section 9:

9.1) Add files, remove files, move files, rename files, and see attributes of files.

9.2) The scheme (http, file, ftp etc.), domain (localhost, website etc.), and path. NSURL is faster, can trap errors and used by more classes.

9.3) When you deal with larger programs you may want to keep track of certain things or keep logs, as a result writing a string to disk could help with evaluating information that the program has produced so far, and you may not know when the memory could potentially release the placeholder it had for your information.

9.4) Archiving gives the advantage of being able to encode the data and also reduce the amount of lines of code for some of the larger property lists.

Section 10:

10.1) You can look at the reference page to see all the related methods you are inheriting. You can override a method that you inherited by recreating it in the sub class with the parameters you chose.

10.2) With inheritance you are creating a new subclass, where as with category you are making it seem almost as if that class had the function you wanted all along. You can not add instance variables to categories, only methods.

10.3) They are very useful because they can automate a process (as long as its in the proper format) and perform an action you want without having to write the code yourself.

10.4) It’s useful if you are unsure as to the type of object something will be in advance, however it may not be always completely accurate when applied to varying types of objects.