CODE with Chris

How to Make iPhone Apps with NO Programming Experience

Beginner Lessons 1-17
Key Takeaways

Lesson 1: Introduction to the Tools and Materials

- You need a Mac in order to install Xcode
- Xcode is the app where we write code in
- You can download Xcode for free in the Mac App Store
- If you want to deploy apps into the App Store, Apple will charge you $99/year

Lesson 2: Playgrounds and Swift

- Playgrounds are a way to test code without creating a new project
- Follow along on your own laptop and try out some of the code statements

Lesson 3: Xcode 8 tutorial and tour

- Xcode has 5 main areas: File navigator, Code Editor, Inspector pane, Debug area and Toolbar
- **File navigator** is where you can see all the files in your project. This is the left side pane.
- Code Editor is where you can write code and work on your user interface. This is the center pane.
- **Inspector pane** is where you can configure properties and look at details of the selected file. It's the right side pane.
- The **Debug area** contains tools to inspect and troubleshoot your code when you encounter errors. It's the bottom pane.
- The **Toolbar** is the strip along the top and it gives you status information, buttons to run and stop your project (left corner), buttons to show/hide certain panes (right corner) and also to select the iOS simulator you want to test your project on.

Lesson 4: The anatomy of an iPhone app

- **View**: the user interface that the user sees.
- **View Controller**: controls the view
- **Model**: manages the data
- The View Controller sits in-between the model and view and facilitates the communication between the two.
- This is called the MVC design pattern and following this will help you write code that is easier to maintain and troubleshoot.

Lesson 5: Your first Swift App

- When you start a new Xcode project, choosing the "Single View Application" template will get you started with a view and view controller.
- Make sure that you choose "Swift" as the language for your project when presented with the options for your new project.
- Your project will start with the following default files:
- **AppDelegate.swift**: this is the entry point for your app. This code file contains a few life cycle events for your app that you can add code to if you want to run some logic during these life cycle events.
- **ViewController.swift**: this is the code file that represents your View Controller. This class manages the view.
- **Main.storyboard**: this file is where you'll see a visual representation of your View. You can add user elements to your view here and customize it.
- **Assets.xcassets**: this is where you'll store your image assets for your project.
- **LaunchScreen.storyboard**: this is where you'll customize your launch screen for your app
- **Info.plist**: contains configuration details for your app
- Follow along with the video to build the Hello World app

Lesson 6: Auto Layout and Stack Views

- Follow the video to work on the user interface for the War card game project
- When you add user elements to the storyboard, you can position or size the element how you want, but that's not how it will appear when you run your app in the iOS simulator.
- Xcode uses a system called Auto Layout to determine the size and position of each element.
- Auto Layout uses **constraints** to determine how to lay things out.
- In the storyboard, there are buttons to open up menus where you can specify constraints for your user elements.

Lesson 7: Stack Views

- Stack Views make building user interfaces much easier
- A Stack View is a container element that can container other elements
- The elements inside of a Stack View are automatically arranged horizontally or vertically

Lesson 8: Completing the UI with Auto Layout and Size Classes

- Follow this lesson on your own computer to practice using Auto Layout constraints and size classes.

Lesson 9: Basic building blocks of Swift programming

- We've talked about the roles of the view, view controller and model and how they work together to form an app.

- Each of those roles are described in what is called a **class**.
- There's a class that represents the view, a class that represents the view controller and a class that represents the model.
- A class contains code that describes the logic and behavior.
- Think of a class like a blueprint that can be used to produce many instances of whatever it is describing. For example, the view controller class can be used to produce many instances of the view controller (if you need more than one).
- Each of these instances is called an **object**.
- When you're coding your project in Xcode, you're simply writing code in a bunch of classes. When your app runs, objects are created from your classes and these objects interact with each other to make your app work.
- A class is organized into **methods** and **properties**.
- A **method** is a block of code with a method name that can be called to execute that block of code.
- A **property** is an attribute of the class that can hold data.
- An Xcode project is made up of classes. A class is made up of methods and properties.
- Watch the video to see the Swift syntax to describe a class, method and property.

Lesson 10: More Swift concepts and UIKit

- A **subclass** is a class that inherits the properties and methods of another class. This is a great way to create a class that extends the functionality or modifies the functionality of an existing class without changing the code in the original class.
- Watch the video to see the Swift syntax to describe a subclass and a demonstration.
- **UIKit** is a library of pre-coded classes that Apple provides for us to use. These classes provide common functionality that all apps will need.

Lesson 11: Hooking it all up – IBOutlet Properties

- The storyboard contains a visual representation of the view. This is like your view class but represented visually.
- When you run your project, the view in your storyboard gets turned into a view object.
- The View Controller has a property called “view” which references that view object.
- This is how your View Controller has access to the view and do its job (manage the view).
- When you add user interface elements (aka UIElements) to the view in the storyboard, if you want the View Controller to have access to them to control and manage them, you need to create an **IBOutlet property** in your View Controller class and connect that property to reference that UIElement in the storyboard.
- You can do this by going into “Assistant Editor” view where you’ll see the storyboard on your left and view controller on your right.
- Then holding down ctrl and clicking your UIElement and dragging it into your view controller class (see the video for a visual).

Lesson 12: Handling button taps – Methods

- You can define your method to have **parameters**. Parameters are bits of data that are required to be passed into the method when you try to call that method.
- This can be useful if the code or logic you’re performing in the method requires some extra information. Watch the video to see how to specify that a method requires parameters.
- In the storyboard, you can add a button to the view by dragging a Button user interface element from your object library.
- If you want to respond to a tap of the button from the user, you need to create an **IBAction method** in your View Controller that is hooked up to that tap event of that button.

- You can do this in a similar way to creating an IBOutlet property: Go to “Assistant Editor”, hold down ctrl, click on your button and drag it over to the view controller class.
- From the pop up menu, you’ll have to select “IBAction” as the type instead and for the event, select “Touch Up Inside”.
- That will create an IBAction method in your view controller. You can add any code and logic you want inside this method.
- When the user taps the button, this method will be called and executed.

Lesson 13: Adding and displaying images in your app

- Image assets in your app should be produced in multiple sizes due to retina screens and non-retina screens on different devices.
- You’ll need a 1x, 2x and 3x size.
- You simply need to add the appropriate suffix to them like this: myimage.png, myimage@2x.png, myimage@3x.png.
- When you’re creating your image assets, create them at the 3x size and use a tool like Prepo to automatically generate the 2x and 1x versions.
- Drag them into the **Images.xcassets** of your Xcode project to include them.
- Image View elements are used to display images. In your storyboard, you can add an Image View by dragging it into your view from the object library.
- If you select your Image View, in the inspector pane on the right, there’ll be a tab where you can choose the image you want displayed in it (it reads from Images.xcassets).
- In the demo, add a few Image Views to the storyboard, then connect them to our view controller via IBOutlet properties. Then we write code in the button tap IBAction method to programmatically change the displayed image in the Image View when the button is tapped.

Lesson 14: Getting a random number and randomizing the cards

- You can use the arc4random_uniform() function to generate random numbers.
- Watch the video to see how we use this function to display random images in our Image Views.

Lesson 15: Using Arrays to manage the card images

- An **array** is a data structure that helps you manage a collection of data. Each piece of data is stored in a numerical index starting from zero.
- You can access that piece of data by using the array's name and the numerical index it's stored in.
- Arrays can store pieces of text, numbers and even a collection objects.

Lesson 16: Determining a winner – IF statements

- An **IF statement** is a piece of code that allows you to make decisions based on a condition. For example, you can use an IF statement to test a condition and if it evaluates to true, then run this block of code, otherwise, run another block of code.
- Watch the video to see how the Swift syntax looks and how we use the IF statement to determine who has the higher card.

Lesson 17: Styling and keeping score

- The label elements have a "text" property that we assign a string into
- Watch the video to see how we increment the score counter and update the labels

What to do next

1. Send me a message at chris@codewithchris.com and let me know what you think about these beginner lessons!
2. Watch the lessons in their entirety and follow along on your own computer. I'm sure you'll be amazed at how much you can learn in 17 lessons.
3. Thank you for learning with me. I really appreciate it!