General Assembly
Mobile Development

Lesson 1: Getting Started

Learning Objectives [Write on board each class; keep visible]

- Set course expectations
- Define basic concepts of layout in iOS
- Outline developer workflow and tools on a high level
- Create the first iOS project outlining the development process of an app
- Label Xcode workspace window

Schedule

Time	Topic	Activity	Notes	Assesment
30 min	Intro	Ice breaker and materials	Time allocated to intro and communication about course expectations.	Students will be able to recall class homework and expectations. They will have a good idea of the course progression and final outcome.
30 min	Layout in iOS	Lecture: high level overview of layout in iOS	Students will use autolayout to lay out views in their storyboards, this is their first introduction to its rules	Students are able to describe the rules of basic autolayout, define a constraint, and list possible constraints for simple layout tasks
60 min	Dev workflow & create first project	Lecture/Guided practice	Demonstrate developer flow and tools that will be used in this class. Make sure everyone has their environments set up.	Students will be able to run Xcode, create a new project, add UI elements, and change element

				properties.
40-60 min	Git and GitHub	Demonstration & code along		● Navigate GitHub to class repo ● Students will have their own branch and understand how to push their work to GitHub ● Pull assessments and class resources from master branch
10 min	Review and Q&A			
30 min	*Extra content:* Running on a device	Demonstration		Students will understand the steps for deploying an application to their devices, know what a UDID, provisioning profile, and certificates are.

Introductions
Time: 30 minutes

Topics	Introduce staff, students, and instructors. Set class and homework expectations
Activity Type	Class discussion/lecture
Assessment	Students will be able to recall class homework and expectations. They will have a good idea of the course progression and final outcome.

Instructional Design Notes

- Start the class with an ice breaker activity
    - Recommendation: go around room and indicate
        - Previous experience with mobile or other development environments/languages
        - Why you are taking the course
        - Favorite mobile app
- Set expectations for
    - Study to class time ratios
    - Processes for getting recent class resources and
    - Submitting homework
    - Receiving feedback on homework
- Talk about final deliverable and how far students will get by the end of the course. Distribute **final project assignment** and **rubric** to get students excited about the end goal of the course and communicate skill expectations.
- Deliver **syllabus** and **learning goals** to students. Discuss at a high level how the class will progress.

Layout in iOS
Time: 30 Minutes

Topics	Introduce the students to the concept of autolayout
Description	Students will use autolayout to lay out views in their storyboards, this is their first introduction to its rules
Activity Type	Lecture
Assessment	Students are able to describe the rules of basic autolayout, define a constraint,

	and list possible constraints for simple layout tasks

Instructional Design Notes (20 minutes)

- Introduce interface builder
    - The way we'll be adding elements to the screen
    - Explain to anyone already technical that layout in iOS is very different than CSS or other layout methods
        - And that we won't be covering it in detail until later, so our views might be a little misaligned until then
    - Explain that almost everything that we'll see on screen during the duration of this course will be ***views***
    - There are a lot of different kinds of views, from buttons to labels to much, much more complex things
        - Views can also contain other views
    - How does iOS know where to put views on screen?
        - Autolayout, we'll cover this in a later class

Draw out constraints on whiteboard (10 minutes)

- Draw out a view containing another view on the whiteboard, and note it's frame (X, Y, width, height)
- Note that every view has a position and size on the screen

Dev workflow and create first project

Time: 60 Minutes

Topics	Developer tools and environment check.
Description	Check to make sure everyone has Xcode up and running. While creating the first app talk about the different project templates at a high level. Introduce new project settings as you are building the project. Identify Interface Builder and the Library area.
Activity Type	Lecture/Guided practice
Assessment	Students will be able to run Xcode, create a new project, add UI elements, and change element properties.

Instructional Design Notes (10 minutes)

- Start this activity by making sure that everyone is able to run Xcode 6+ and that their developer accounts are properly connected to Xcode. Instructors and TAs are to walk around and help anyone still struggling.

Code along (30 minutes)

- Go over project options in detail while creating a new project. Give a high level overview on the General options.
- Show Interface Builder and drag a label. Rename the label to “Hello World” and run the application.
- Talk about layout sizes. Demonstrate that if we add an object to a layout other than universal, it only displays for that screen size.
- Emphasize that the interface builder does not specify layout, it’s a graphic tool to add and adjust constraints
- Demonstrate the Attribute inspector by changing attributes of a label/button.
- If there is additional time let students search and add more elements onto their project.
- Run the app and congratulate students on their first app.

Section activity (20 min)

- Create a new project. The view controller should display text that contains a short bio of the students.
- The project should have a button with the text “Goals”.

Link

Mention that we went through the exercise of previewing our applications on the simulator. Apple requires a few more steps to get our apps on our actual devices.

Introduce git and github

Time: 40-60 Minutes

Topics	Version control and Github
Description	Introduce class to GitHub and git and set up their version control environments. Make sure you lead with GitHub so they are clear on the purpose before they get more intimate with tool.
Activity Type	Demonstration & code-along
Assessment	● Navigate GitHub to class repo ● Students will have their own branch and understand how to push their work to GitHub ● Pull assessments and class resources from master branch

Instructional Design Notes

GitHub:

- The recommended way to start the lesson is by introducing GitHub first, as opposed to git. It will be easier to talk about a social network for sharing files, than talk version control and its given benefits and flows.
- **Mention that this tool will be very important in our class for pulling homework, slides, submitting homework, and getting feedback from instructors on homework.**
- Introduce GitHub as an essential tool for backing up, sharing, and collaborating code for developers, however it can be used by any profession that works with computer files. It lets you collaborate on work and revert to any place in your saved history. Furthermore it lets you branch out your work by functionality or role.
- Compare the branching out model as a parallel universe to the master path.
- GitHub lets you also create conversation about every file change or topic in your project.
- Briefly mention the pricing model for GitHub. Mention that it's free to have public repositories and paid if you want to make your repositories private. Because of this free model there is a great amount of code available on GitHub that you have access to for free. What's even cooler is that you can also contribute into your favorite projects without any real barrier to entry (if your contribution is legitimate of course).

Git:

- Link: Introduce git as the pipes that run GitHub. Mention that git is used primarily as a version control tool, and it's the tool that we use to communicate to GitHub.
- The primary method that developers use to communicate with git is through the command line, however for the purposes of this class we are going to use the GitHub desktop tool.

What we are using GitHub [tool] for:

- Pulling class notes (slides) and resources
- Pulling assignments
- Turning in work/assignments
- Collaborating with each other (much later on in the class after we get more comfortable)
- Backing up your work

Demonstration and code along

Goal: explain how it works and get everyone to pull class repository

- Start by demonstrating GitHub and navigating to our class repository
- Highlight folders they will pull from:
    - Lessons
    - Resources
    - Assessments

- Highlight folders they will push code to:
    - Assessments
    - Class work
    - Notes (optional)
- Mention that every student will have their own branch that they will push their work to
    - Note: GitHub for Mac does not allow to sync forked repository with upstream. If this feature was available the first alternate option below would be preferred.
    - Alternate option 1: get all students to fork the class repo and work on their own forks.
    - Alternate option 2: each student creates their own repo to submit their files to and has a totally separate repo where they only pull class work from.
- Very important points to note at this time and while coding along:
    - Git will watch for changes for both files and folders under its root repository (may have to explain what root is for very non technical students). **Therefore, all work they created that students want to turn in and back up should be saved in these folders.**
    - **Only make changes to anything while you have your own repository selected**
    - **Only add your files to the Assessments and Class work (and notes if you include in your folder structure) folders**
    - Explain that it's important that they only work in their branches and the assigned folders because the syncing mechanism works like a portal in a parallel universe. When you sync from another universe you open a portal to duplicate all of their things, however if you've edited those same things, there will be clashing (merge conflicts).
- Setup Sourcetree
    - Download and add GitHub accounts
    - Go to class repo on GitHub and copy URL
    - Navigate to branches: explain that this is where they will sync class work on, and where they'll be able to view everyone else's branches.
    - Have them add their own branch by selecting the + sign on the master branch. The branch should be named FirstnameLasname (camel case). Have them publish the branch.
    - Demonstrate adding a new file by adding their first Assessment.
        - **Demonstrate "Commit" (mention that they should commit very often with clear messages) and sync.**
    - Demonstrate how they should sync up your changes by doing a **Sync** and then merging the master branch into their branch. *****Make sure they are very clear here, as they could easily do it the other way around.**
    - Make sure everyone has their own branch selected and have them

    add the new project they created into class work. Highlight that this should show up in the “Changes” tab. They should commit and sync up the repository. Have the class navigate back to GitHub and see their changes live.
    - Repeat this again in the Lessons folder where they can pull this lesson’s slides.

Check for Understanding
Do a Q&A with the class. Ask the following questions:

- What is a branch?
- What is Sourcetree?
- How do you pull changes from GitHub? How do you merge?
- Why do we not want to make changes to the main branch?

Review and Q&A

Time: 10 Minutes

- How often do students submit assessments?
- What is autolayout?
- What is interface builder? Xcode?
- What is a constraint?
- *Optional:* What are requirements for deploying to device?

Check for Understanding (10 min)

- Instructors to discuss questions out loud with students. Instructors and TAs to call on random students to explain each answer.

Extra material if session is running short:

Running iOS on simulator and device

Time: 30 Minutes

Topics	Deployment process
Description	Demonstrate the process for getting applications into device and publishing to App store (although mostly focused at device deployment). Mention that this is important to note, even if students aren’t iOS dev program members.
Activity Type	Demonstration
Assessment	Students will understand the steps for deploying an application to their devices.

Instructional Design Notes

- Developing and running app on simulator is free.
- Deploying app to device and publish app to app store requires users to be part of the developer program ($99/year).
- Add device to Xcode (Window->Devices).
- Show class developer account portal.
- Explain that Apple requires a certificate to be able to deploy to device/publish to App store (this is their way of confirming that you are an Apple approved party). Show the need for this by displaying the error message under the General tab. Add developer account to Xcode.
- Define provisioning profiles.
- Explain where to get UDID number and demonstrate how it applies to the above workflow.