Day 11

This is CS50 for MBAs. Harvard Busines School. Spring 2015.

Cheng Gong

Table of Contents

Agenda

- Day 11 will be mobile development, day 12 will be online advertising with Dan Armendariz, and day 13 will be about choosing a technology stack, with Patrick Schmid.

Tommy

- First, a short video (embedded in the lecture video) featuring Tommy, a former CS50 TF who now works at Quora. Quora is a question-and-answer website where a community of experts contribute responses and build knowledge. Tommy focuses on mobile development, with the goal of allowing people to easily use all the features of the website from their phones.

Questions

- An article titled "D-Link says sorry for shoddy security and sloppy patching of its routers"[1] was sent in, illustrating the importance of checking users' inputs before executing them as commands.
- "Can you spend a few minutes going more high level again? Website server vs data server. Physically do they look different? Or is one virtual? More of the differences. Computing services vs storage services (seems like computing does both)."

[1] http://betanews.com/2015/04/20/d-link-says-sorry-for-shoddy-security-and-sloppy-patching-of-its-routers/

    # Web servers and databases might physically be the same, with standard sizes to fit into a rack (see rack unit[2]). The differences are actually in the software, where we have a program that is either a web server or a database server running on the machine. Either could be virtual, as both could be in the cloud or on its own physical machine. For a really small website, both the web server and database server software could even run on the same physical machine, since they could talk to each other and not worry about slowing down for lack of resources, though now we literally have a single point of failure for the entire site. As for computing services vs storage services, computing generally requires more CPU and storage generally requires more disk, so the hardware might just be allocated differently to each machine differently. The documentation for them will describe how much of each resource you get.

- What is Docker and why is it being embraced at such a rapid clip?

    # Docker is a software that makes software very portable. For example, an application developed on one virtual machine can easily be run on another, with very little additional configuration, with the help of Docker. From the website[3], "Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in." And developer operations, or devops, would be the process by which software created by developers is actually deployed. In the past, the operations side was more hardware-focused, but with the advent of the cloud, it’s becoming more software-focused.

- Could you explain how Heroku and its add-ons are used? How would you incorporate and pay for an add-on? What makes Heroku easier than AWS?

    # Heroku makes it easier to set up those pieces of software that run web servers or database servers, by pre-configuring and pre-optimizing them for AWS. A developer could do this manually as well, but it takes time and effort that might be better spent elsewhere.

- From the class discussions and my online explorations, I am still not sure that I understand the difference between Redis and PostgreSQL. They both are cloud

[2] https://en.wikipedia.org/wiki/Rack_unit
[3] https://www.docker.com/whatisdocker

storage servers for databases that allow for horizontal and vertical scaling. Is the difference largely that Redis is used for enterprise applications, but PostgreSQL is for "hobby" apps? Thanks!

  # PostgreSQL actually has a longer history than Redis (1996 vs 2009, to be precise), and while PostgreSQL is a more traditional full-featured database system, Redis is object-oriented database program that is also starting to gain features. In general though, the choice to use a certain system over another is based on a number of factors, such as application needs, community support, or developer familiarity.

- On day 9, Professor Malan talked a little bit about oAuth. Many tools for apps require oAuth functionality like Facebook connect. How does oAuth work? What are the security vulnerabilities? What was the difference between oAuth and oAuth 2?

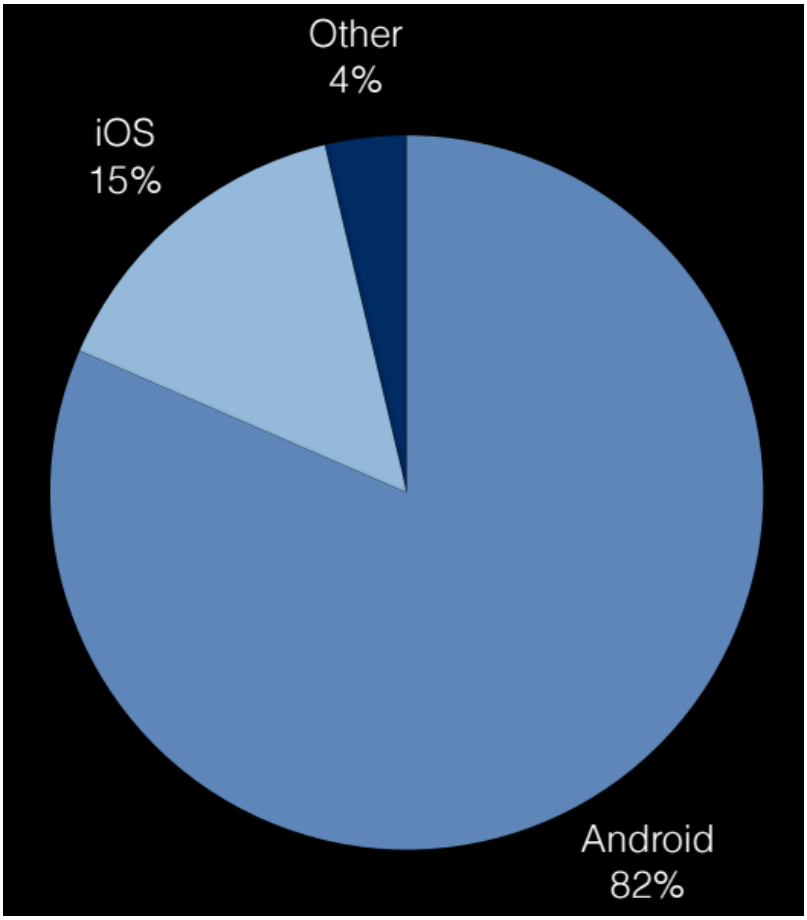
  # OAuth and similar technologies solve the problem of having people log into various websites with one account, without having to pass their usernames and passwords to all the websites (which is insecure). Instead, for Facebook Connect, for example, Facebook's login form is shown as a pop-up, and Facebook, once the user logs in, sends back some sort of secure verification to the other website indicating the user's identity. OAuth 2 includes mobile support.

- It seems Facebook's React Native language looks promising and is relevant to our discussions. Could you perhaps discuss this in class? http://techcrunch.com/2015/04/20/howfacebooks-react-native-will-change-mobile-apps/

  # We'll try to come back to that by the end of class, after our discussion of mobile development.

Mobile Development

- The market share of mobile operating systems look something like this these days:

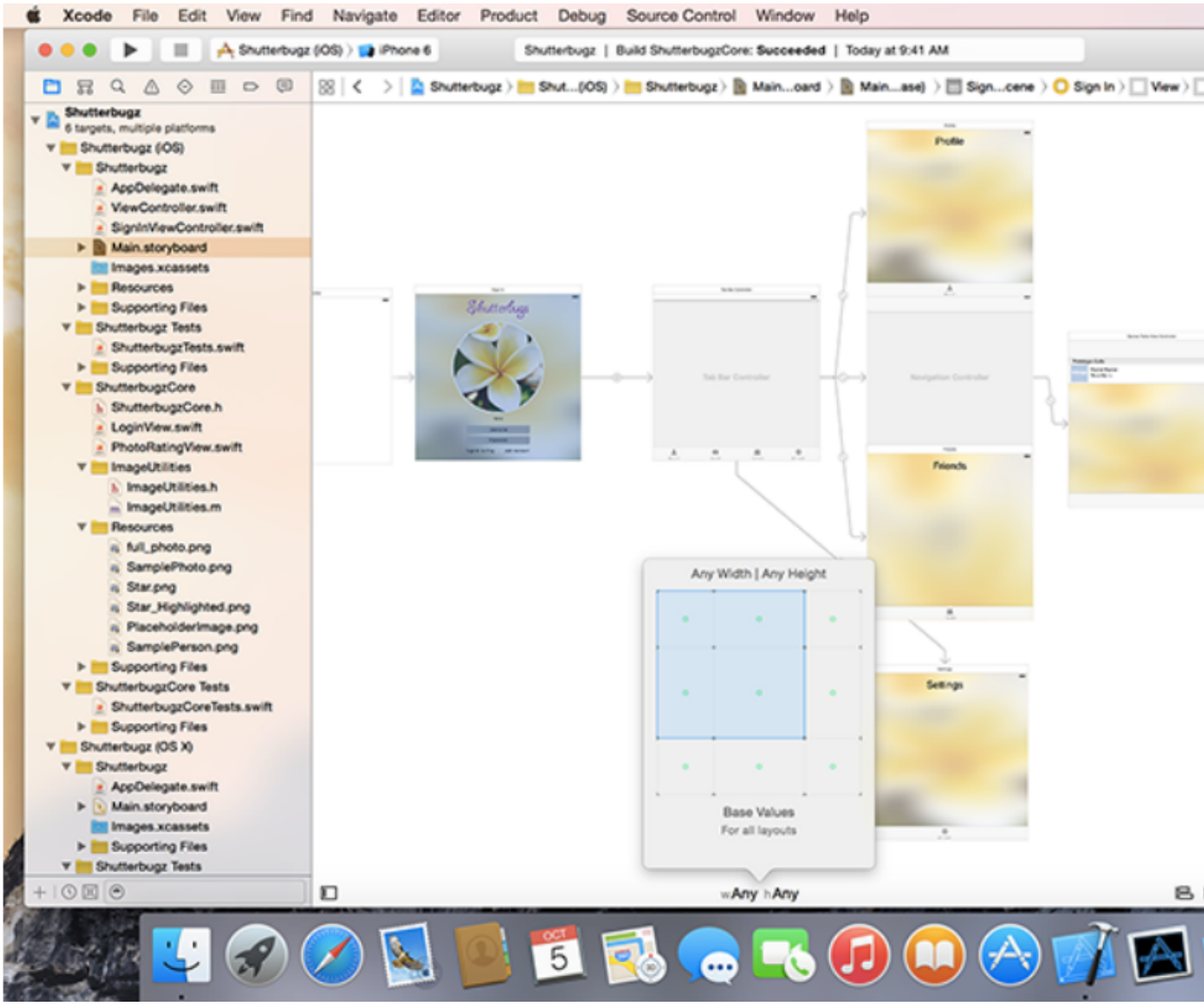


- Of the people in the class, the vast majority have iPhones, so the ratio definitely changes based on demographics and geography.
- The "Other" category includes Windows Phones, Blackberries, and other smaller vendors.
- The takeaway, though, is that generally mobile developers have to either exclude some part of their target audience, do twice as much work to develop for both Android and iOS platforms, or write an application that works on both platforms, but not optimized for either.
- A native application is a program that's on the App Store (or Play Store) that users can download and install on their device locally.
- For iOS, the languages that these apps must use are either Objective-C or Swift.
- Apple provides an IDE[4] called XCode for devleopers to use, to create apps for iOS and OSX:

[4] https://en.wikipedia.org/wiki/Integrated_development_environment

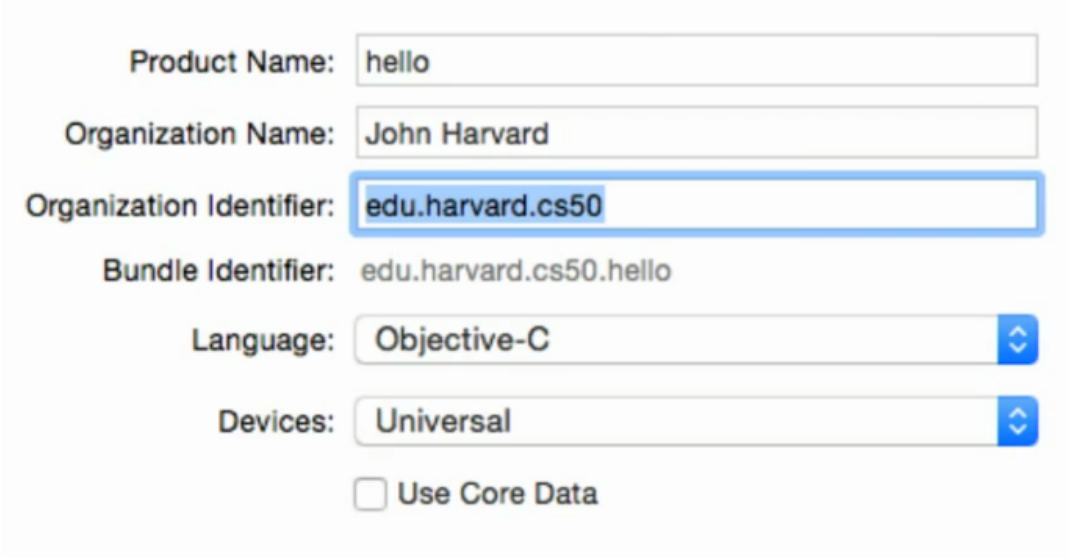
We see a file browser on the left, a preview of the user interface in the middle, and the code for that screen on the right.

- iOS development also involves something called storyboards:



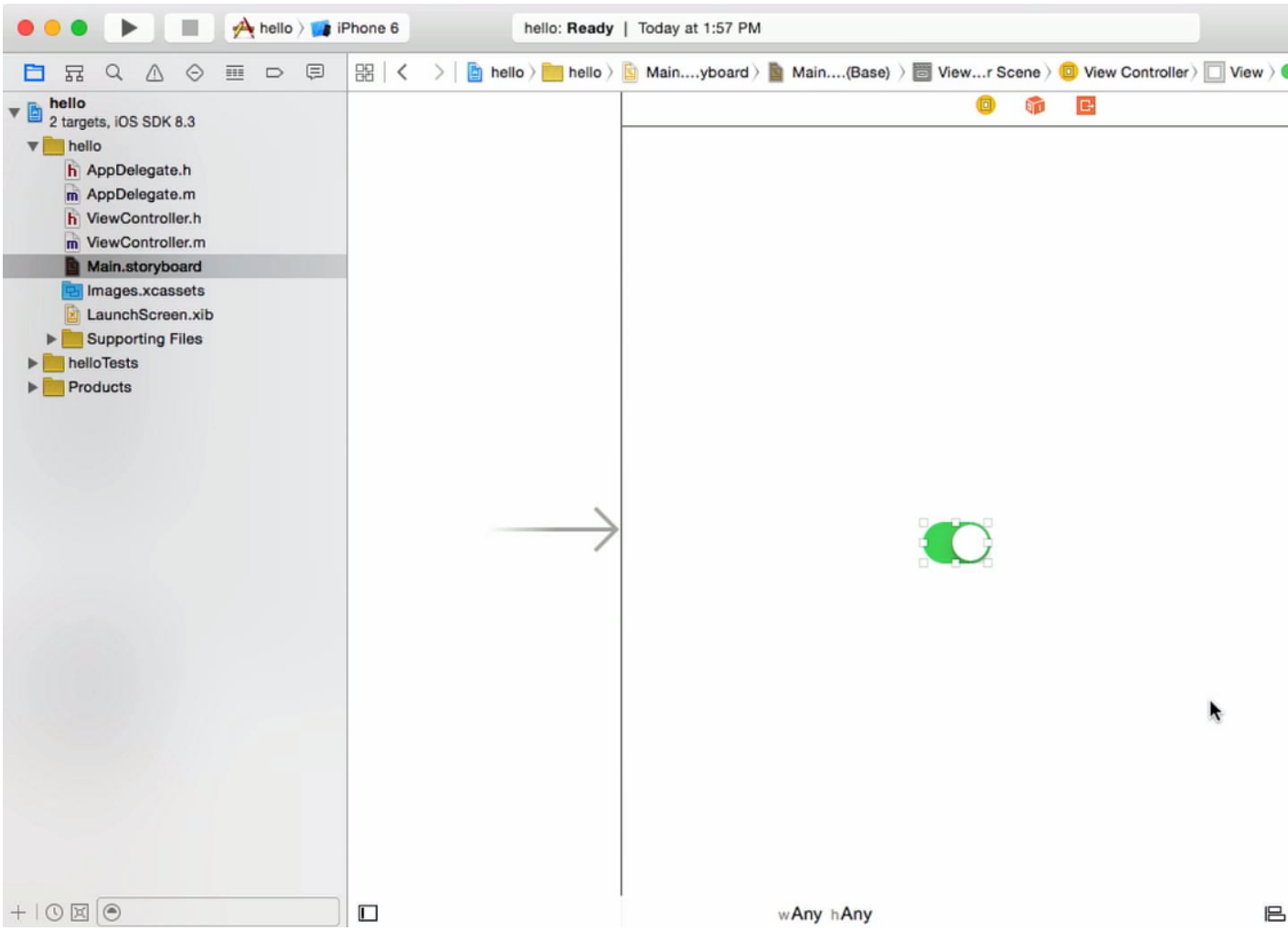
These are the main screens in an app, and the storyboard shows how they are connected by various links.

- David does a quick demo of XCode, where he first creates an app called hello:

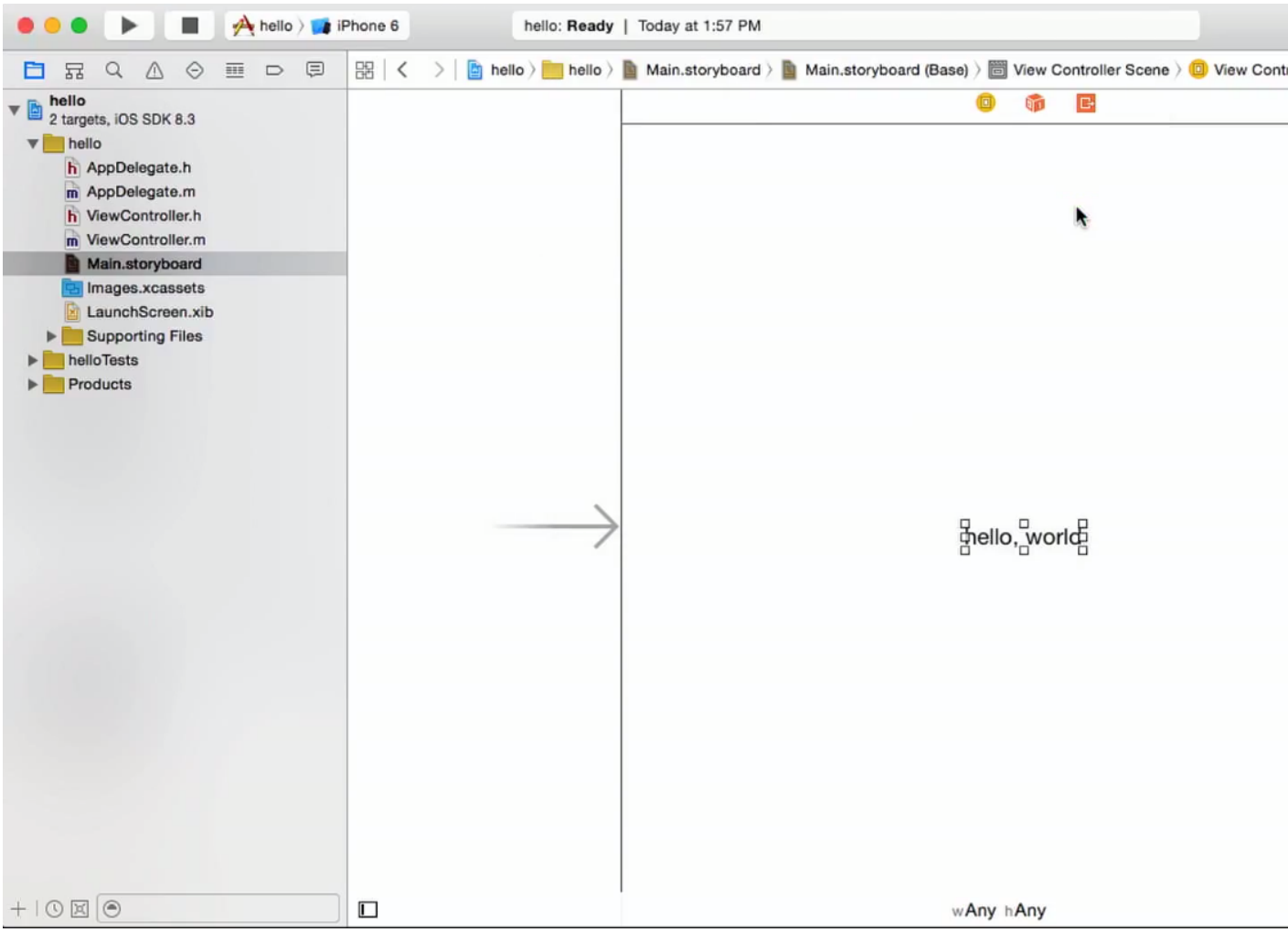


    # The organization identifier, by convention, is the domain name of the organization that created the app, but backwards. Using a domain name keeps organizations from having overlapping names.

- The app has an initially blank storyboard, but we can drag over, say, a switch:



- We can also drag over a label and type some text in:

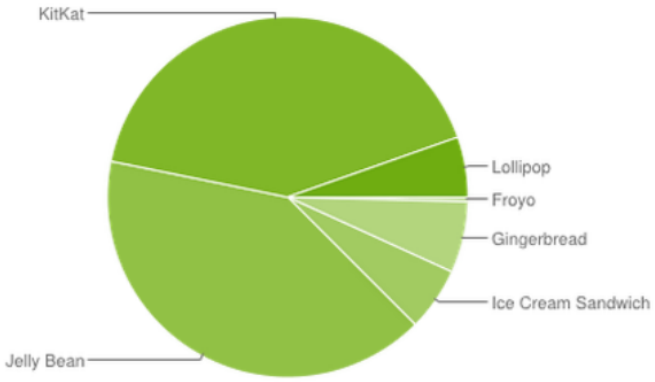


- XCode also comes with a simulator, or a program that simulates an iPhone on your computer that runs the app you're working on.
- As for performance and battery issues, generally developers can optimize their applications with better algorithms or data structures, but at the same time rely on the operating system to make the most use out of the hardware. For example, Apple did not allow background apps in iOS until just a few years ago, since it would negatively affect speed and battery life too much.
- Stanford has a freely available iOS course[5] taught online, with many resources for someone to teach themselves iOS development. Other online education providers have courses as well, but they generally charge for those options.

[5] http://cs193p.stanford.edu/

- Android uses the Java language for native applications. The primary IDE for that is Eclipse[6], and developers also need the Android SDK[7], software development kit, which includes libraries, simulator, and other files.
- There are some freely available tutorials[8] from the official Android site, too.
- The operating system version distribution for Android, however, looks like this:

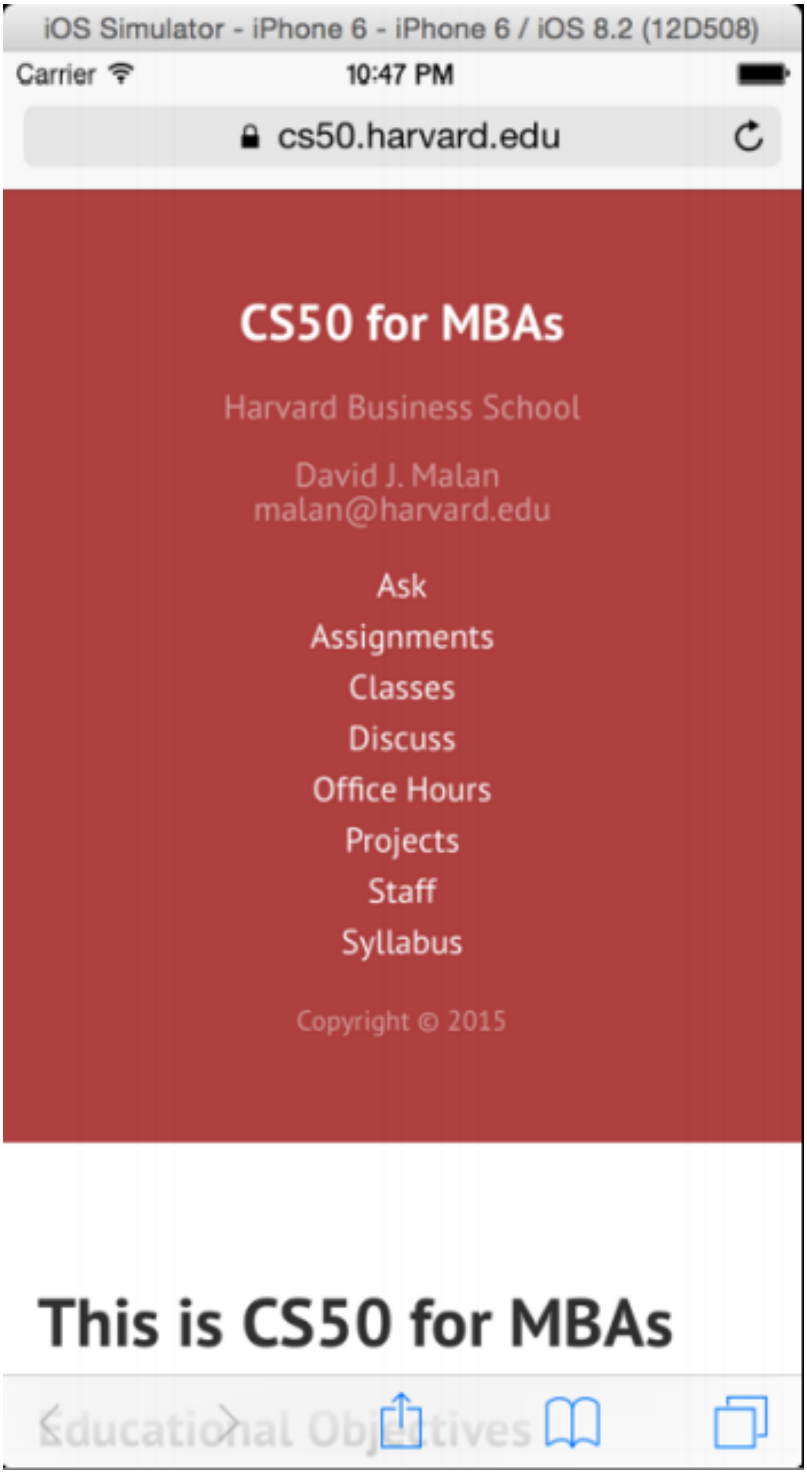
Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	6.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.7%
4.1.x	Jelly Bean	16	16.5%
4.2.x		17	18.6%
4.3		18	5.6%
4.4	KitKat	19	41.4%
5.0	Lollipop	21	5.0%
5.1		22	0.4%



- There are lots of users with older operating systems, implying that their phones are also older and lacking certain features, or perhaps just slower.
- Apple's avoided this by having more forceful automatic updates, as well as controlling the hardware that can run iOS. But bugs in the update process mean that users might end up with bricked, or broken, phones. (In fact, this happened[9] with iOS 9, a few months after this lecture.)
- Dropbox, for example, sends out their app updates to a small percentage of their users at first, and wait a few days for any bug reports or issues before updating all users' apps.
- And backwards compatibility, too, is an issue because not only are newer features unavailable in older software or hardware, older features previously available might be replaced or removed altogether.
- Different screen sizes and resolutions, too, require app developers to be considerate and thoughtful about layout, if they want to support a wide range of devices.

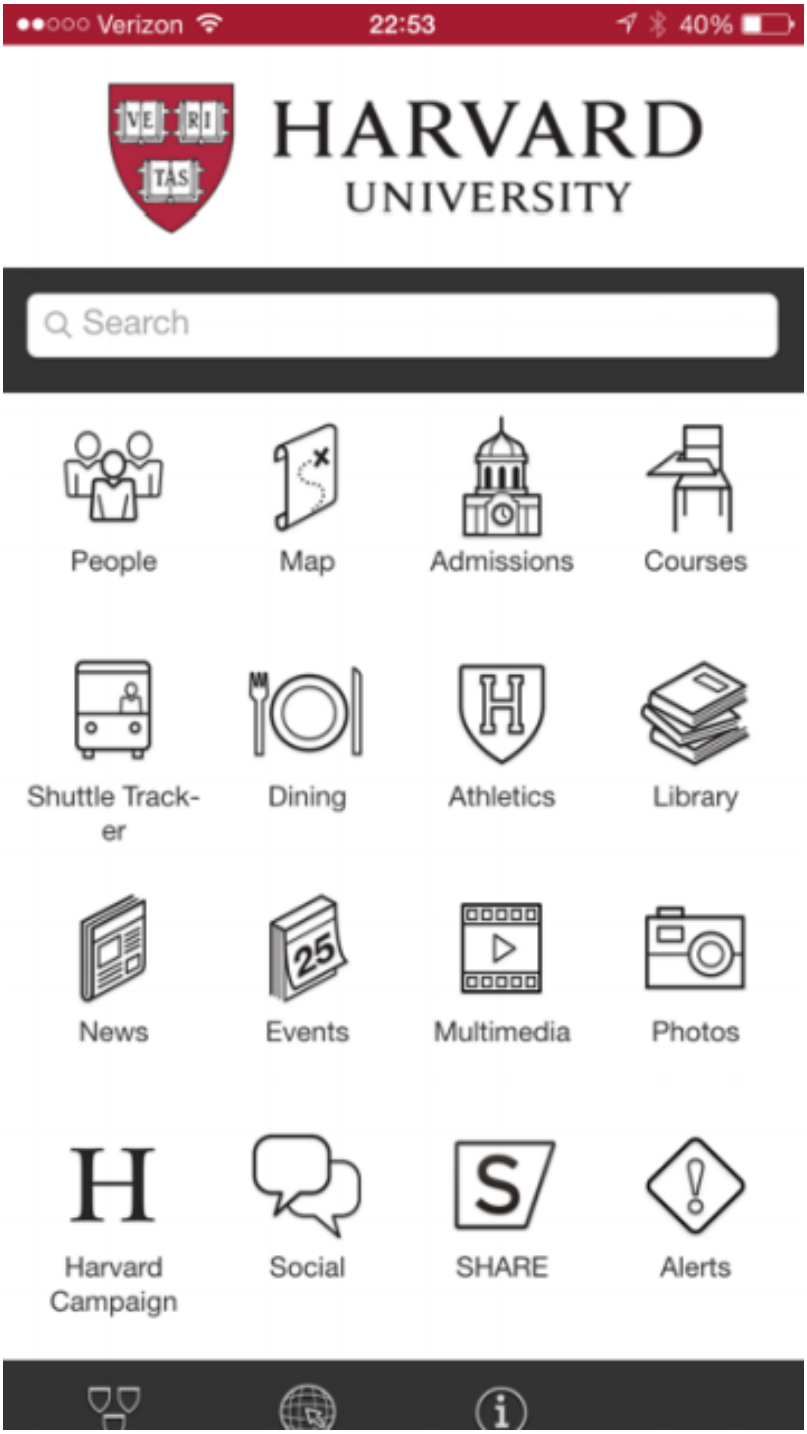
6 https://en.wikipedia.org/wiki/Eclipse_(software)
7 https://en.wikipedia.org/wiki/Android_software_development#Android_SDK
8 http://developer.android.com/training/index.html
9 http://www.reuters.com/article/2015/09/18/us-apple-update-ios-idUSKCN0RI05P20150918

- So instead of writing native applications, companies can also adapt their websites to have a mobile layout:



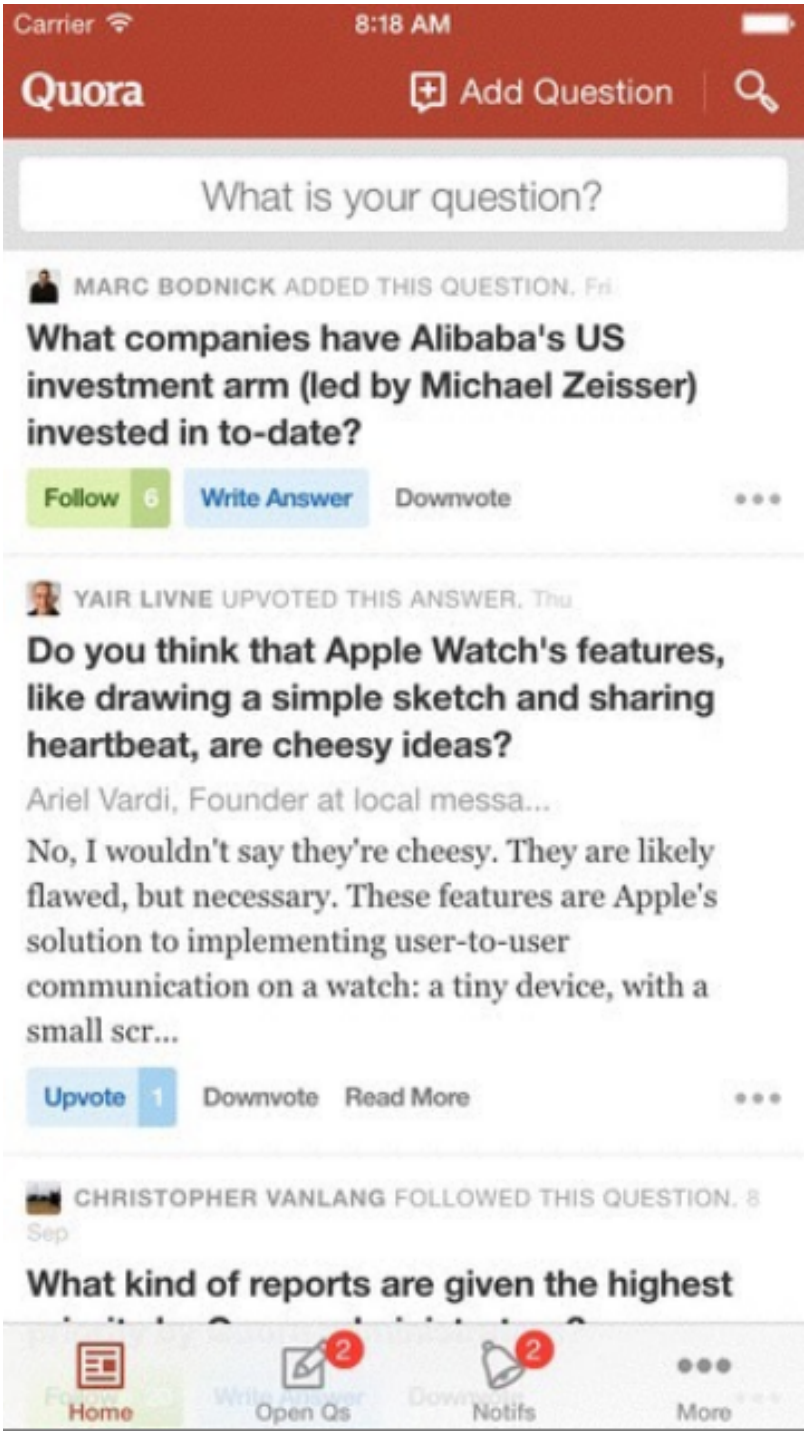
- But websites lack access to features such as location, photos/camera, and notifications.

- The user also needs to have internet access in order to access anything in the app at all, and might also be frustrated if there is a lot of content to download while they are on a slower data connection.
- Using CSS to adapt pre-existing websites for a mobile layout is pretty straightforward.
- The final option, then, is the use of hybrid apps, where the application is written in HTML, CSS, and JavaScript, but installed onto the user’s device so as to have access to all the features it can.
- Using HTML, CSS, and JavaScript as the language means that any smartphone will be supported, since they generally all have browsers.
- Harvard, for example, uses a hybrid app:



- All the main content inside is written with HTML and CSS, and only the buttons below are native, or in Objective-C.

- Apps that look similar to this might be based on some framework or library that has pre-written code for those native features, with a menu bar at the top or bottom and essentially a web browser in the middle that displays most of the content of the app.
- And access to features like the camera or location might be a simple JavaScript call that the content can use, with the help of those libraries.
- Quora, too, has a hybrid app based on some framework:



- The downside is that these apps are generally not as fast, since there is the additional layer of rendering HTML and CSS and running JavaScript on top of these frameworks that use the actual native languages.

- The back-end of the mobile app doesn't matter as much, because APIs can be standard HTTP requests or even provided as a service by companies like Parse[10]. The decision for how to implement that, however, is separate from that of the front-end.
- Some frameworks that developers might use in a hybrid app include:
  # jQuery Mobile
  # PhoneGap
  # React Native
  # Sencha Touch
  # Titanium
  # …
- These frameworks are essentially basic programs, written for each of the various mobile platforms, with blanks for developers to fill in and customize.
- Tommy from Quora returns to talk about the tradeoffs they chose to make for this exact decision.
  # With native apps, all the components like buttons and forms fit in with the rest of the operating system.
  # The developer time for maintaining two platforms, or more, however, increase significantly.
  # With a web approach, changes made to the app will be reflected the next time the app downloads new information, and with a native app, changes take some time as users have to download and update the entire app.
  # In a hybrid app, speed of the application is sacrificed, since the content has to be both downloaded and rendered.
- Given that Quora's main product is a website that's constantly changing as well as the fact that it's a startup, it makes sense for them to choose a hybrid approach rather than writing native apps.
- Day 12 will be about advertising, and Day 13 will be about choosing your entire technology stack. Until then!

[10] https://www.parse.com/