Course Introduction & Module 1

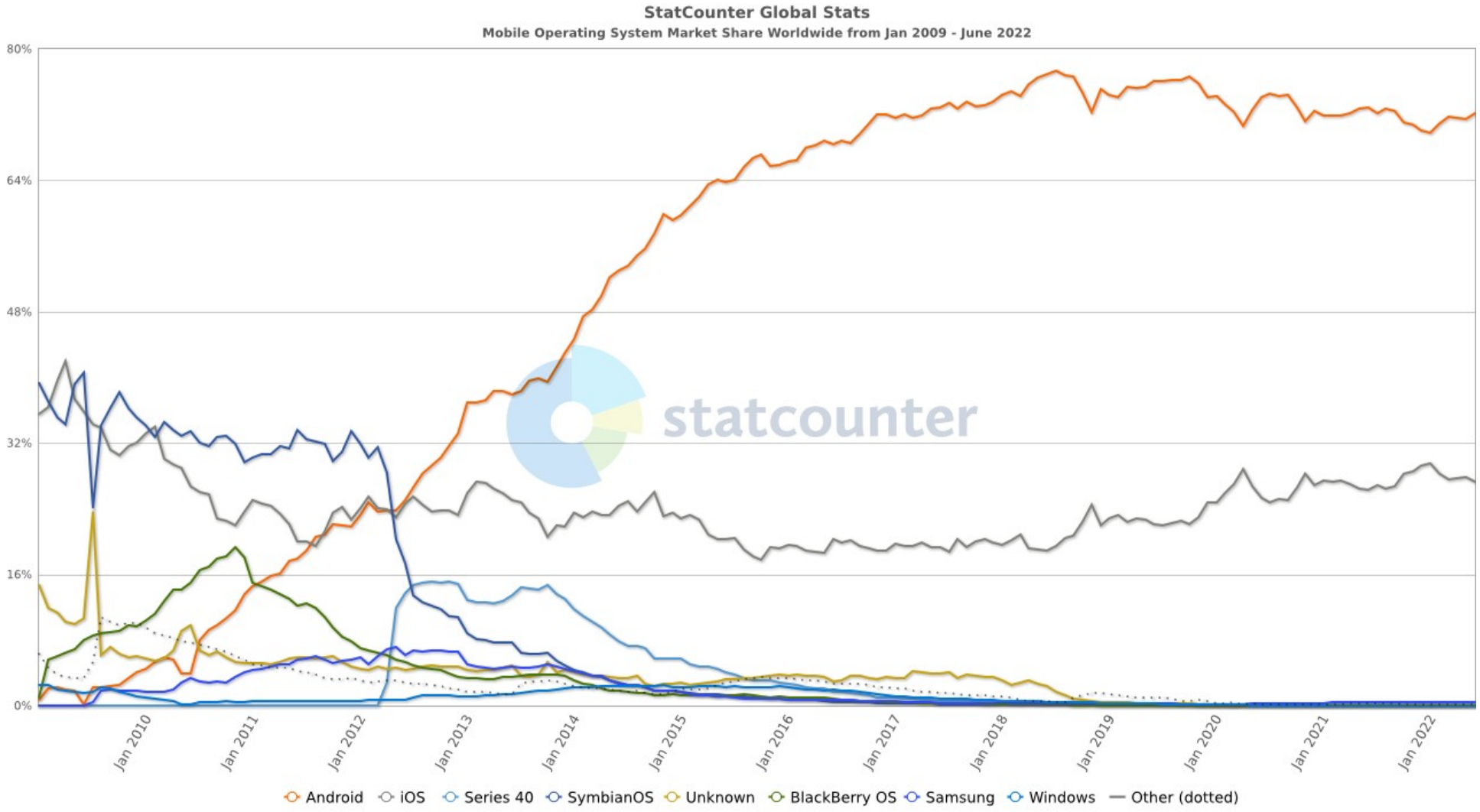
CS 5254

Welcome!

- Questions? Comments? Concerns?
  - **Piazza** is always the best way to contact us regarding anything, at any time
  - Active participation via Piazza is most highly encouraged
    - The goal is for each of us to be a member of a supportive/inclusive learning environment
- About this course:
  - Focused on development of applications for the **Android** platform
    - This is the predominant mobile platform world-wide
      - It also happens to allow substantial flexibility in terms of development environments
    - You don't need an Android device; we'll use an emulator provided by the IDE
  - We'll be using the **Kotlin** language throughout this course
    - An alternative to Java with first-class support by Android starting in 2017
    - Adopted as the preferred language for Android development starting in 2019
- Assistance:
  - Please let us know about anything we can do to help improve your experience in this course
  - Work with Services for Students with Disabilities to request any formal accommodations

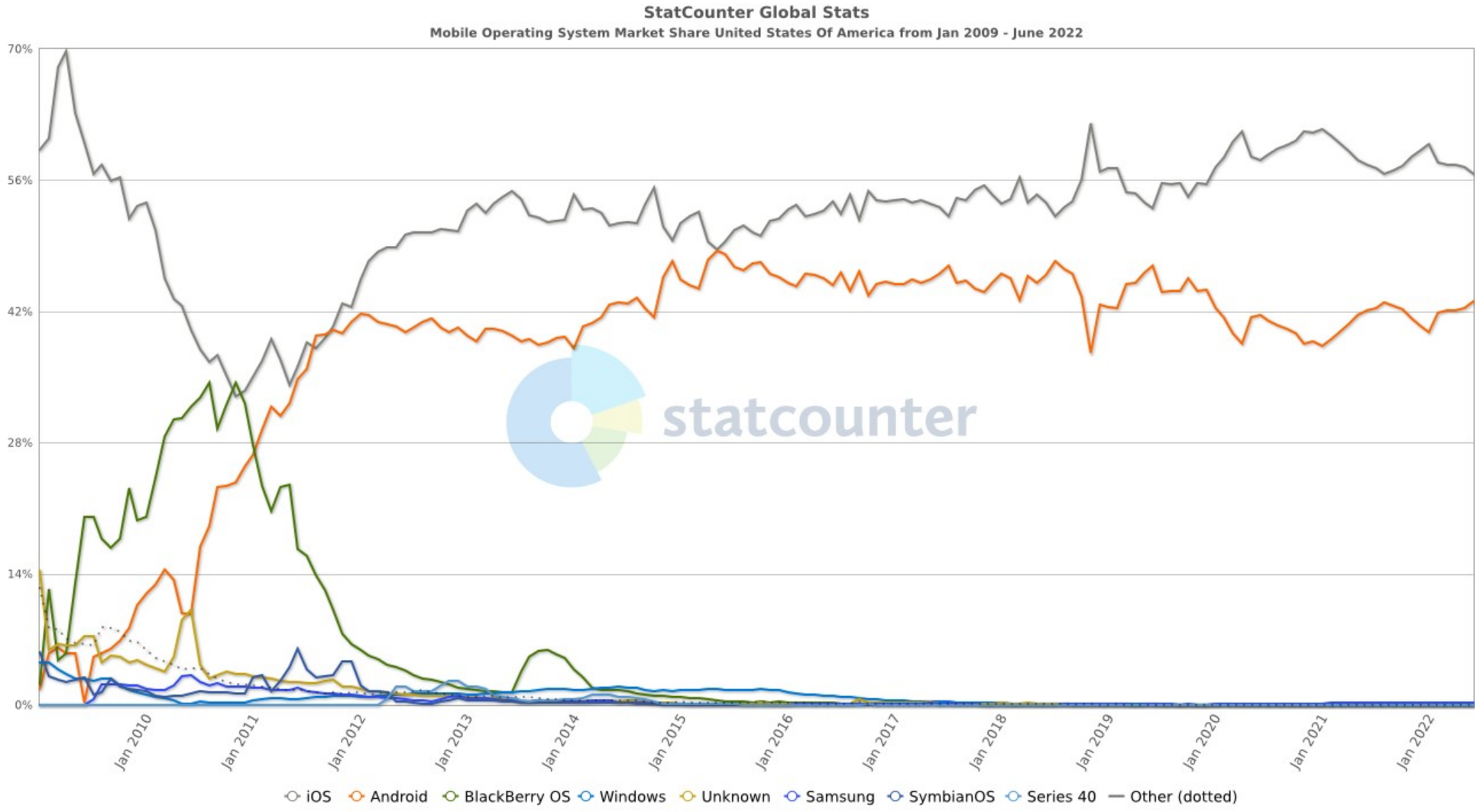
Android worldwide

- Android has remained the dominant mobile platform by a large margin since about 2014
  - Currently trending at around 70% of the market, with iOS in a distant second place



Android in the US

- Trends above don't hold for the US, as iOS has been the market leader since its 2019 introduction
  - However the gap is substantially smaller, with iOS at about 55% to Android at about 45%



About the assignments (all due Mondays at noon ET)

- Readings and Videos – including the Q&A sessions – are ungraded but required
  - You must either attend the Q&A or play the recording at your convenience during the week
  - For readings, decide what depth is appropriate for your background and learning style
    - You may not need to follow along with every textbook example or challenge
  - For recordings, feel free to configure playback for faster or slower speeds as preferred
- Formative Discussions:
  - More than a few simple sentences are expected, but far less than a multi-page essay
    - Try to interact with your classmates if practical, to build upon their contributions
- Formative Quizzes:
  - Timed multiple-choice questions regarding recently covered materials
  - No "trick" questions; nothing is intended to deceive, but please read everything carefully
- Projects:
  - Programming exercises based upon – but more expansive than – the textbook examples
  - Try to allocate extra time for unexpected issues that can arise during the development process
- Exams (Midterm and Final)
  - Summative assessments covering the major topics of the course
  - Same multiple-choice format as the quizzes

What are these module slides?

- The intent is to provide insights above and beyond the main course materials
  - Certain topics might have been under-emphasized, missing, or even misleading at times
  - Background and historical context often aids in deeper understanding
  - Reiterating and summarizing key concepts can help synthesize topics from multiple sources
  - Upcoming and ongoing project assignments often require additional materials or concepts
    - Note that an introductory project walk-through is generally done during Q&A sessions
- The goal of this informal format is to provide a nice balance between a video and a document
  - In future weeks, some of these module slides may have associated videos
- We've redesigned the course recently based on latest technology, tool, and textbook updates
  - The slides should help tie together the overall curriculum
  - The ideal is a correct, complete, and consistent set of course resources
- We're very open to your feedback!
  - Please let us know your thoughts about what's working well (and what's not working well)

System requirements

- Here are the minimum requirements for your development environment:
  - System memory:
    - 8 GB RAM
  - Disk Space:
    - 16 GB available
  - CPU:
    - AMD: Ryzen (Zen-based) or better
    - Apple: M1, M2, or better
    - Intel: Skylake (6th generation Core) or better
      - Includes Intel-based: iMac 2015; iMac Pro 2017; MacBook 2016; MacBook Air 2018; MacBook Pro 2016
- Android Studio itself supports systems with older/fewer resources, so...
  - ...the first few weeks you should be fine with a slightly less capable environment
  - However, you will almost certainly begin to encounter issues with Project 2 and beyond

What about Kotlin?

- The extensive official documentation is quite good at https://kotlinlang.org/docs/
  - You might want to try the “Kotlin by Example” tour in the Learning materials section
  - Most of what we’ll do is actually fairly basic, but it will help to become more broadly familiar
- You’re encouraged to explore and practice Kotlin via the playground at https://play.kotlinlang.org/
  - This simple browser-based IDE lets you edit and run Kotlin code in a sandbox environment
- Generally the IDE will provide basic templates for the kinds of classes we’ll develop
  - We’ll most often just override existing functions or add new functions
- Much of the syntax we’ll use will be similar to Java
  - There’s a heavy emphasis on lambdas and other functional programming techniques
  - Very similar to Java 8/9+ lambdas (and most other languages that support lambdas)
- We’ll start slowly and introduce more language features as needed throughout the semester

Kotlin basics from a Java perspective

- Minor but important terminology differences:
  - Methods are called “functions” and they can appear outside of classes
  - Fields are called “properties” and default setter/getter methods are automatically generated
- Syntax differences you’ll probably notice first about Kotlin code:
  - Semicolons are optional (and highly discouraged)
  - Type specification appears after the declaration, separated by a colon
    - Example: `fun countSpaces(name: String, trim: Boolean): Int`
    - Types can often be inferred by the compiler (and should be omitted in these cases)
  - Properties are accessed or assigned via `=` notation rather than getter/setter calls
  - Types are non-null by default; you must append `?` to the type to allow for possible null
    - Use `?.` or `!!.` to force access to a nullable value, or `?:` to specify a fallback value
  - No primitives; everything is an object (although wrapper classes are treated specially)
    - Use `==` for object equivalence comparisons (including comparing to null)
  - Lambda expressions must generally appear within braces: `{ ... }`
    - Default name for a single lambda parameter is `it` (but can be overridden)
- Here are two unofficial but nice quick reference guides that might be helpful:
  - https://blog.kotlin-academy.com/kotlin-cheat-sheet-1137588c75a
  - https://fabiomsr.github.io/from-java-to-kotlin/index.html

Hints and tips for Project 0

- Please confirm that you're definitely using the **5th Edition** of the BNRG textbook
- Try to install Android Studio early to ensure there are no unexpected issues
  - Please ask in Piazza if you encounter any issues
- Just follow along with the textbook for this project
  - Note: Copy-and-paste from a PDF can introduce problems with nesting (mostly in later projects)
    - You still might find it easier to copy code from the O'Reilly web interface instead
  - You may also download solutions for all textbook exercises (minus the challenges)
    - The archive is available at https://www.bignerdranch.com/android-5e-solutions
    - It's still critical to follow along with the textbook to understand how and why it works
- An introduction of Project 0 in Android Studio will be presented during the Q&A session