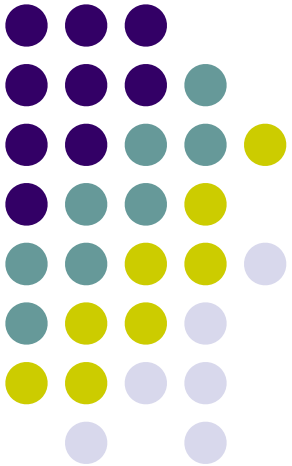
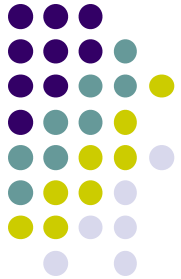
Mobile Programming

Introduction

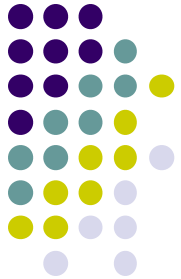


Challenges of Smartphone Programming



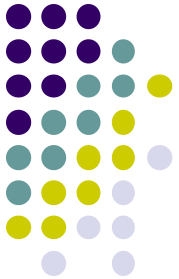
- Screens are small, variety of displays
- Keyboards are small, variety of devices
- Pointing devices are annoying (large fingers vs LCDs)
- CPU speed and memory are limited
- You have to follow the rules of the manufacturer
- Limited data storage
- Network data limitations
- Battery powered
- Device manufacturer limitations

Stakeholders



- Device manufacturers
- Users
- Service Providers
- O/S manufacturers
- Applications / Application Developers
- Government

Mobile Programming Best Practices



- Keep applications simple
  - Limit design to minimum functionality
  - Place each functional component in an Activity

SIMPLE

USER FRIENDLY

Mobile Programming Best Practices (cont'd)

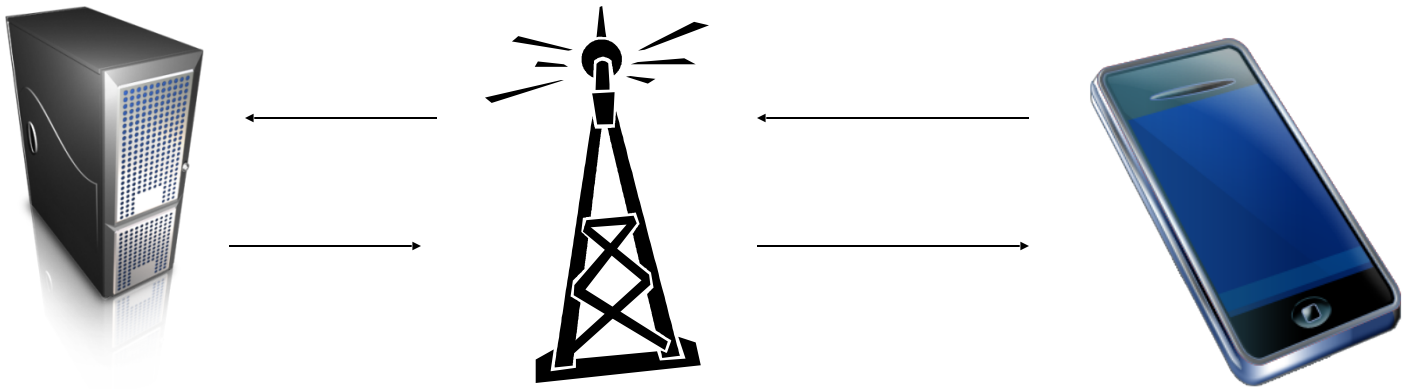
- Limit use of memory
  - Instead of using object types use scalar types
  - Use minimum data type for storing data
  - Manage garbage collection, but how??
    - Allocate an object immediately before the object is used
    - Set all references to objects to null after no longer needing them
    - Always re-use objects instead of creating new ones
    - Reduce throwing exceptions
    - Release all resources after usage (Network, files, db,…)
    - Use local variables as much as possible
    - For more → https://developer.android.com/training/articles/memory.html

Overall memory management

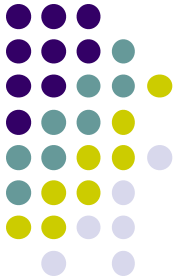
Peak time memory management

Mobile Programming Best Practices (cont'd)

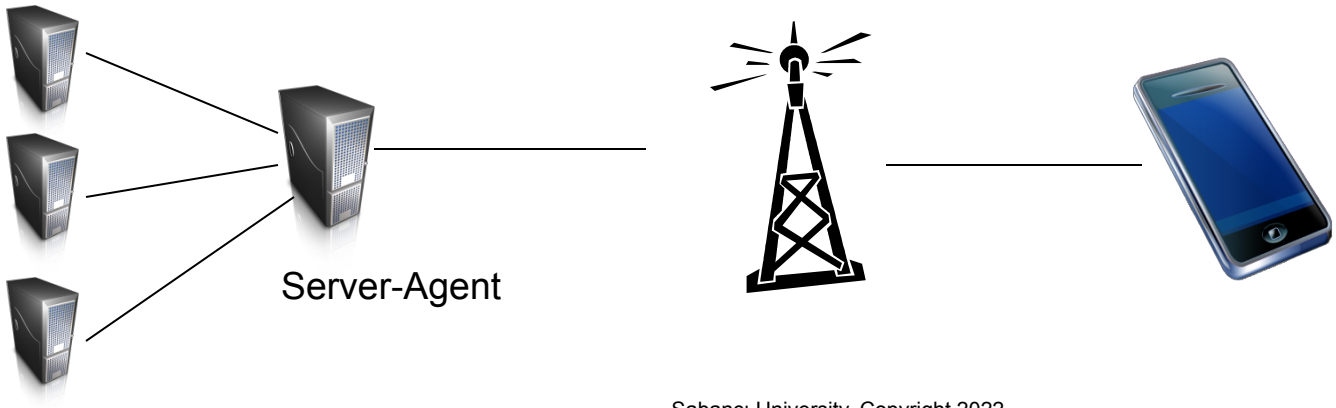
- Off-load computations to the server
  - Perform minimum processing on device
  - Create a client service or web service to support intensive processing
    - Ex: UI on device, data processing on server



Mobile Programming Best Practices (cont'd)



- Manage application's use of network connection
  - Keep transmissions short
  - Keep large amounts of data on a server-agent and request them partially
  - Create a mechanism for recovering from a transmission drop

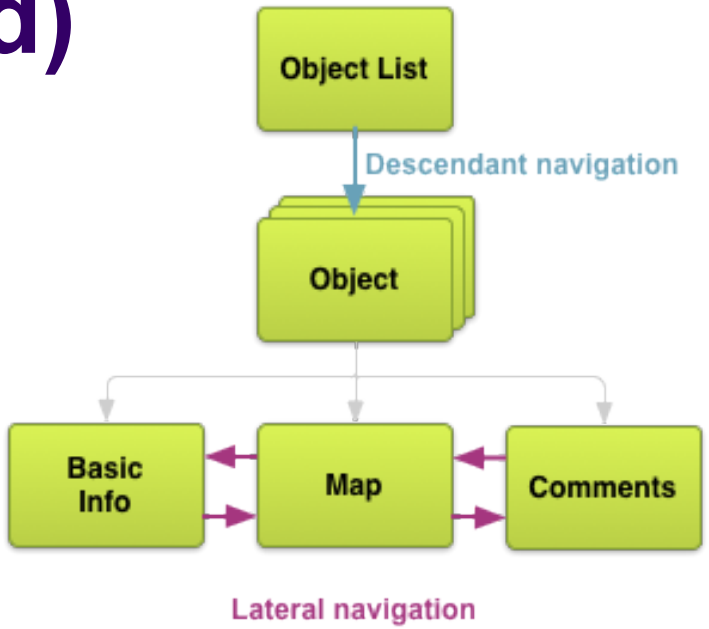


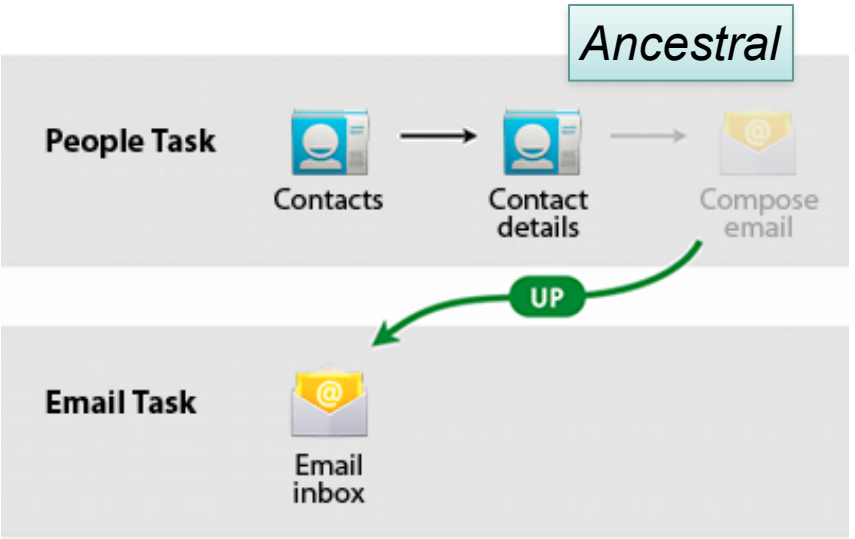
Mobile Programming Best Practices (cont'd)

- Simplify the user interface
  - No standard display for mobile devices! But try to follow ecosystem rules.
  - Use short-cut keys for keyboard input on menus
  - Limit the amount of user input to simple menu selections instead of textboxes
  - Might use images instead of text for selection
  - Might use search function where necessary

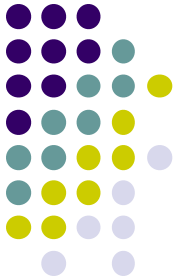
Mobile Programming Best Practices (cont'd)

- Plan effective navigation before coding
  - Descendant and Lateral Navigation
  - Ancestral and Temporal Navigation

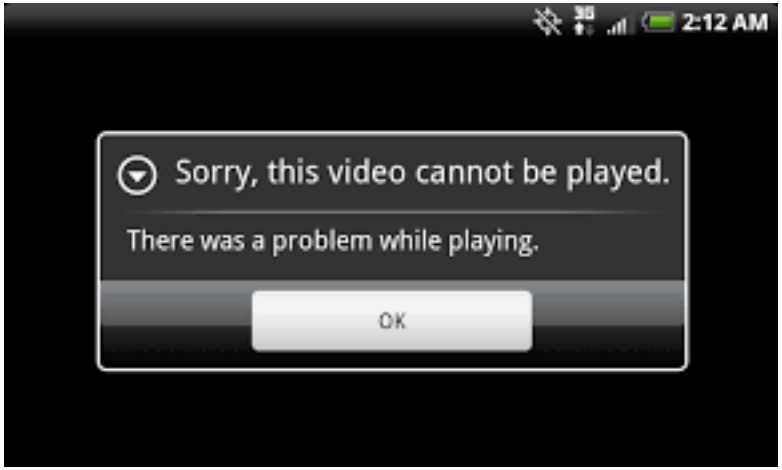




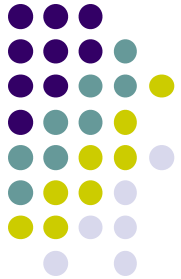
Mobile Programming Best Practices (cont'd)



- Notify the user
  - Display errors in simple and understandable way
  - Let user to display the notification or not
  - Show some progress when a task continues working

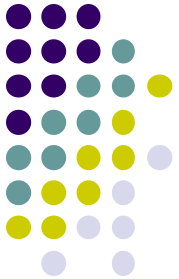


Mobile Programming Best Practices (cont'd)



- Don't concatenate Strings
  - Concatenation increases use of memory and amount of processing
  - Every String concatenation creates a copy of Strings
  - Use equals() for comparison
  - Use StringBuffer or StringBuilder for concatenation

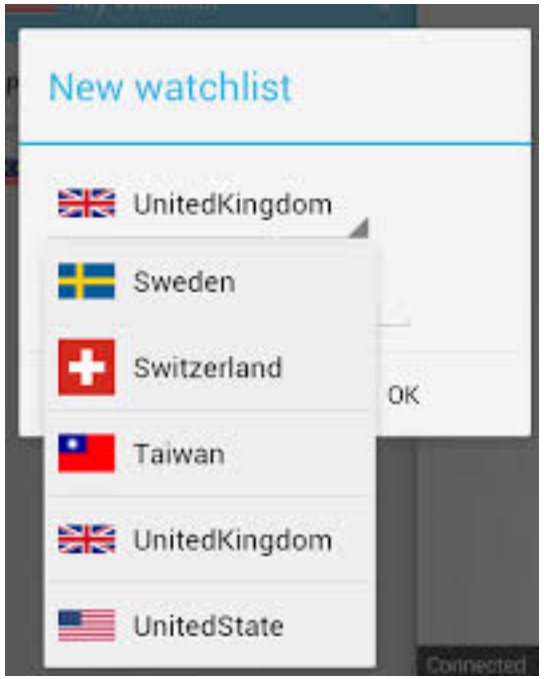
Mobile Programming Best Practices (cont'd)



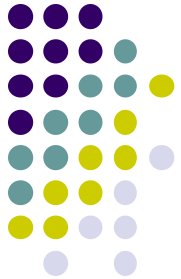
- Avoid synchronization
  - Always use a thread whenever an operation takes longer a tenth of a second
  - Avoid using synchronization unless there is a high likelihood that conflicts among operations will occur

Mobile Programming Best Practices (cont'd)

- Populating drop-down boxes
  - Can load data to drop-downs from a datasource if amount is small
  - Load list dynamically from server when it is long
  - Release list after selection
  - Test for best scenerio!

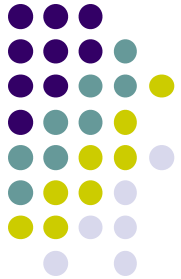


Mobile Programming Best Practices (cont'd)



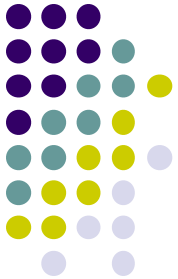
- Dealing with time
  - Devices might not reflect time-zones as they are moving
  - Be careful with time-sensitive data
  - Remind user to adjust date/time
  - Store time-based data always on Greenwich Mean Time (GMT) by using `getTime()` method

Mobile Programming Best Practices (cont'd)



- Automatic data synchronization
  - Build a routine in your app to upload latest data when app is invoked
  - Prompt your user to log in to network if not connected, for usage of latest data
  - Be careful consuming the network!

Mobile Programming Best Practices (cont'd)



- Updating data that has changed
  - Two changes possible:
    - Data changes on device
    - Data changes on server
  - Let user to decide on
    - Incremental updates: data exchange occur whenever data changes, only changed data exchanged. PERFORMANCE ⇩
    - Batch update: Updates a batch of data periodically or on demand, only changed data exchanged. PERFOMANCE ⇧
    - Full update: Should only be used by user selection. For ex restoring data in emergency
      TIME ⇩ PERFORMANCE ⇩

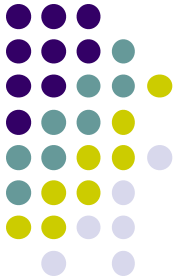
Mobile Programming Best Practices (cont'd)

- Manage possible crashes and long pauses
  - Make your app un-crashable
  - Inform user on long pauses with a dialog
  - Present options for canceling long running operations.

Mobile Programming Best Practices (cont'd)

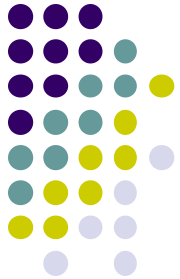
- Optimize battery life
  - When connectivity lost, release your background tasks
  - Location services, data tasks, etc are the ones that consume battery most

Hybrid and Native Apps



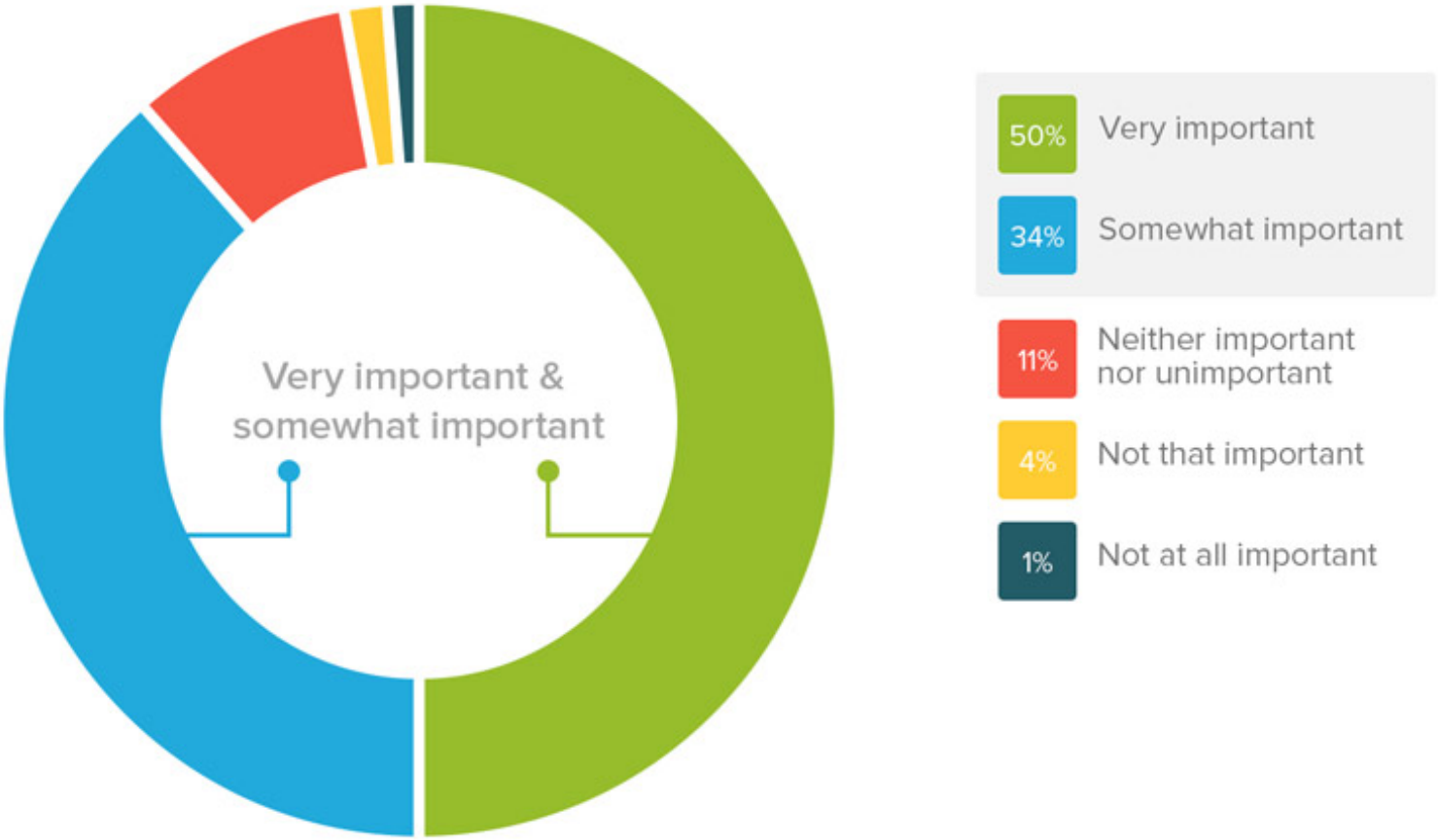
- Hybrid apps
  - Usually written in JS+HTML5
  - Websites packed into a native wrapper
  - Works like a web application
  - Ex. Apache Cordova, Ionic, React…
- Native Apps
  - What we are going to learn!

Hybrid vs Native



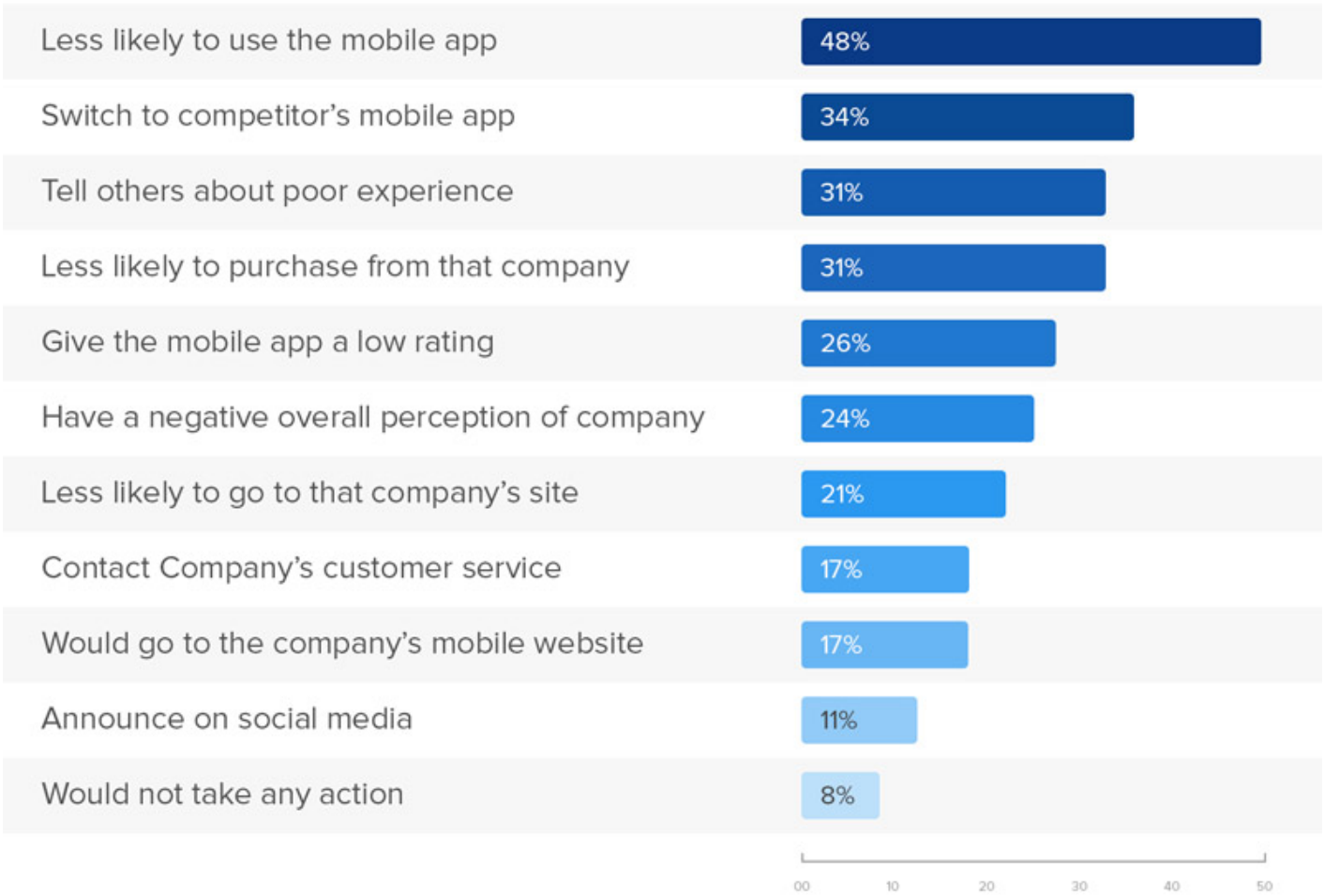
- Nobody has time for bad user experiences, you customers and employees included
  - 79%: would retry an app only once or twice if it failed to work first time
- Hybrid apps are slow in response time and do not look like native (menus, components, etc) – **UI Experience**
- Hybrid app development takes less time for multiple targets. (choose if have less than 6 months for production) - **Performance**
- Native applications have the best performance, highest security and best user experience

How important is mobile app Performance

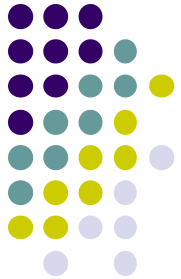


https://info.dynatrace.com/rs/compuware/images/Mobile_App_Survey_Report.pdf

Consumer reaction to poor mobile app experiences

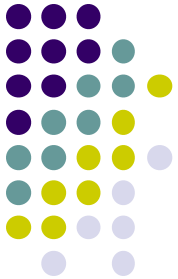


Release Cycles – Hybrid vs Native



- With a hybrid application the user doesn't need to update the app in the app store. (If the update in question is on a page that is loaded from the server.)
- In contrast, for native applications the user needs to update the app to see the changes.
- If the framework is compiled for the target as React Native, then updates depend on the framework provider.

The Differentiators



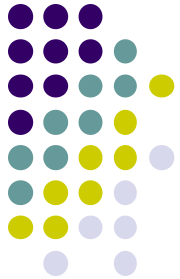
Native

- Best security
- Best in class user experience
- Best performance
- Offline mode

Hybrid

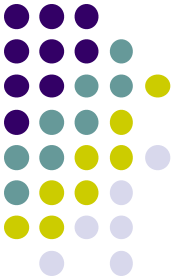
- Portability (one code base, multiple platforms)
- Access to various hardware/software capabilities (through plugins)
- Cheaper origination costs
- Faster (initial) speed to market

Some questions for deciding



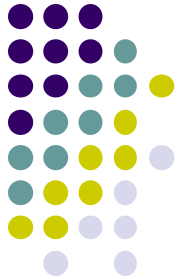
- How much time do you have to market your app (<4 months or greater)
- Do you really need to build an app to work on both platforms right away?
- What does your customers care about? UI Experience? Performance? None?
- What is your mobile app budget?

Learning Mobile Programming



- Mobile programming is related with all! Backend , Frontend, Microcontrollers, TV's, Watches, Refrigerators.
- All development depends on environments: Frameworks provided by stakeholders: Android, IOS, React Native, …
- You need to experience as much environments as you can for success.

Cheatsheet for Learning a Framework



- **What are the programming languages supported?**
  - Java, Kotlin, Swift, Objective C, Javascript, Dart, C#, etc
  - Get experienced on basics!
- **Application Components**
  - Learn about components provided by the framework to display screens, UI components, call threads, access device capability
- **Navigation structure**
  - Learn how to navigate between screens
  - Learn how to pass data between screens
- **Concurrency**
  - Learn how to execute asynchronous tasks, AKA threads, most of operations like downloads take time, so you should always think asynch!
- **Communicating with the Internet**
  - Learn how to make HTTP requests
  - Learn how to process data formats JSON, XML, etc.
- **Learn about the rules of the store**
  - PlayStore, AppStore, Meta Markets