Responsive and Adaptive Design

CS 442: Mobile App Development

Agenda

– Terminology
– Why?
– Approaches
  – General rules of thumb
  – Tools & techniques

Terminology

"Responsive"
Layout *responds* to changes in viewport (e.g., window or screen) size, orientation, etc.

E.g., if being run on a mobile device that switches between profile/landscape, or if running the same app on a phone, tablet, desktop, etc.

"Adaptive"
UI presentation & behavior *adapt* to differences in feature set of the platform hosting the app.

E.g., running on a mobile device or tablet with touch based input and few or no buttons, vs. on a laptop with precise cursors, keyboard & mice with function keys.

E.g., using a gesture based UI (typical of a mobile OS) vs. a menu-driven UI (typical of a desktop OS).

Why is this important?

The grail: single codebase for all devices & platforms
Ideally, only have to maintain one codebase for different devices and platforms. Win for developers! (is this always true?)

No need to worry about maintaining different versions of the app (what sort of issues might arise from this?)

Can we do it differently?
– Separate codebases
– Higher-level configuration tools (e.g., for selective compilation)

Flutter is a great for building responsive & adaptive apps!

(let's learn how to use the relevant tools)

Techniques

General rules of thumb

– Design for *Smallest* and *Largest* screens
– Obey platform and general UI conventions
– Don't hardcode sizes & positions
– Test, test, test!

I.e., consider the edge cases!

Don't make assumptions based on the most sophisticated users (e.g., who use shortcuts, gestures, etc.)

Prefer relative to absolute size & position specifications (e.g., side panel is 1/3 of screen, vs. xxx pixels).

Note: Flutter's "logical pixels" are a step in the right direction, but is not a panacea.

Don't fight the platform!

Don't try to invent your own UI patterns!

Nothing is a substitute for actually testing your app on the physical devices being targeted (this can get prohibitively expensive).

Tools & Techniques

– Media Queries (`MediaQuery`)
– Breakpoints
– Flexible widgets (e.g., `Flexible`, `Expanded`)
– Auto-rebuilding widgets (e.g., `LayoutBuilder`)

Breakpoints: thresholds for switching between different layouts.