IN4MATX 285:
Interactive Technology Studio

Practice: Beyond Web and Mobile

Today’s goals

By the end of today, you should be able to…

- Articulate development practices on:
  - Desktops
  - Smartwatches
  - Virtual/mixed reality headsets
  - Large/public displays
  - Voice assistants/chatbots
- and how they differ from development practices for web and mobile technologies

Thinking beyond web and mobile

Web tools as the standard

- Nearly every platform needs to communicate with a cloud system
- Most platforms need a web browser so people can access sites
- Shared programming language and development environment enables efficient work
- Developers can write once, deploy to many platforms
  - Hopefully customize style and functionality to the device
- Other reasons?

Web is the standard, but not the only

- There are lots of reasons not to implement using web tools
- Further away from the hardware, less performant
- Harder to access device resources
- Further away from platform guidelines
  - Apple Human Interface Guidelines, Material Design
- But if you’re only looking for exposure, it’s the place to start

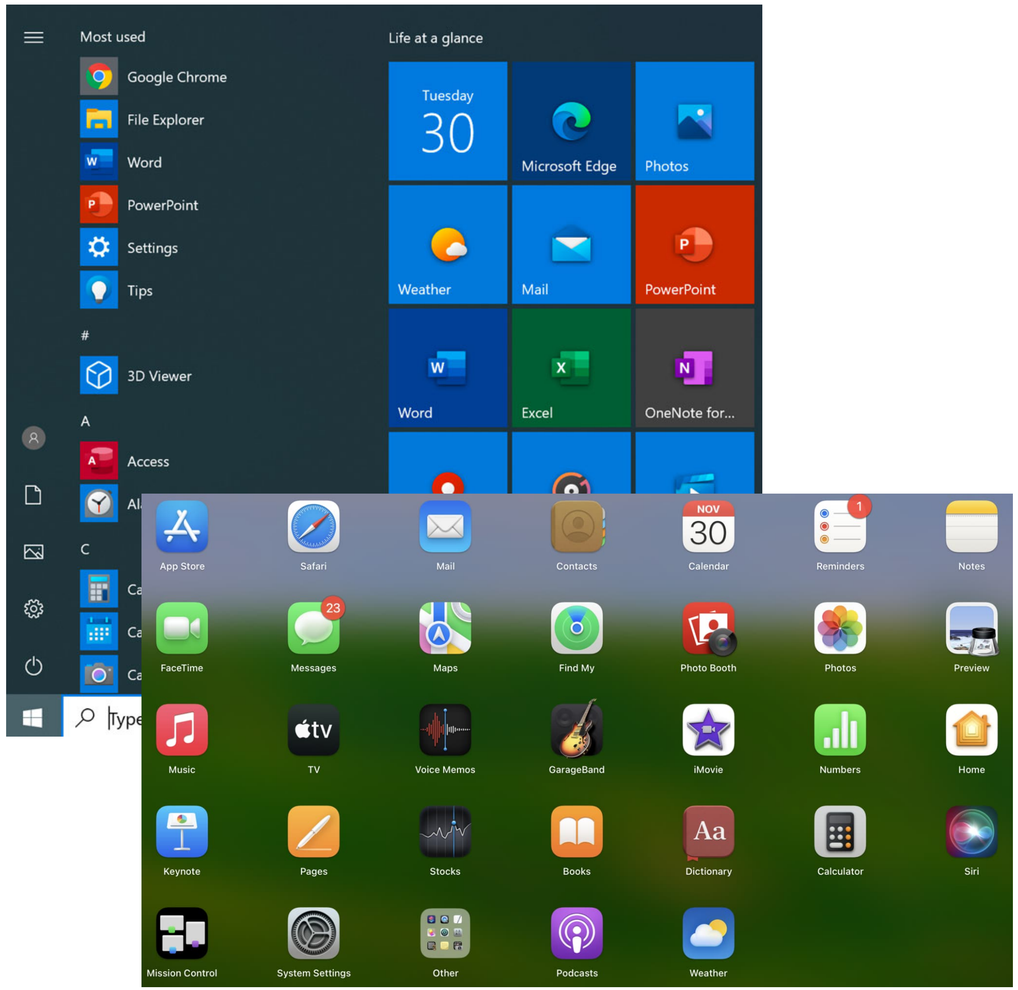
App development across devices

- Desktop apps
- Smartwatch apps
- VR/MR apps
- Voice assistant apps
- Large/public display apps

Desktops

Desktop app development

- What do I mean by a desktop app?
  - A program that runs on your computer rather than your browser
  - Today, many have website equivalents
- Examples: VSCode, Microsoft Word, Chrome



Desktop app development

- Today, you can choose to develop natively or with web tools
- Frameworks like Electron support using HTML/CSS/JS to build desktop apps
- Natively, OS-specific libraries are pretty common
  - But, there are also some libraries for developing jointly across Mac/Windows

https://www.electronjs.org/

Desktop app development

- Tradeoffs between HTML/CSS/JS and native are pretty similar to for mobile
  - Native has better performance, more access to native device features
  - HTML/CSS/JS has less to transfer to/from code for websites
- Types of apps which tend to be built natively:
  - Performance-critical apps: video/image/audio editing, 3d rendering like games/CAD
  - OS utilities: file managers, drivers for printers and other devices

Smartwatches

Smartwatch app design

- Lots to think about on the design side
- How do I minimize interactivity?
- What’s the main use case?
- Am I designing something for the home screen (complication) or a standalone app?
- Do I need a smartwatch view at all?

Smartwatch app development

- Development is more constrained
- Most apps are native, versus leveraging web and mobile technologies
  - WatchKit for iOS, Wear OS for Android
  - (Garmin, Samsung, etc. have all centralized on Android)
- Why?

https://developer.android.com/training/wearables/apps
https://developer.apple.com/documentation/watchkit

Smartwatch app development

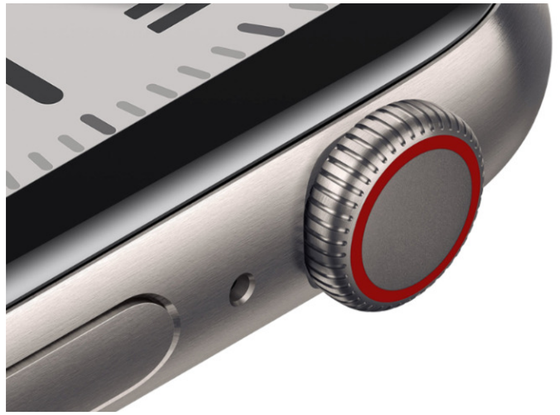
Why native?

- Performance *really* matters
  - Much more limited hardware (memory, processor)
  - Just enough battery to last through the day, but not more
  - Native code tends to be more efficient
- To maintain performance, smartwatches typically don’t have web browsers

Smartwatch app development

Why native?

- Smartwatch apps usually leverage hardware features, which are more accessible with native
  - Interface elements: digital crown
  - Sensors: heart rate, accelerometer, GPS
  - Near-Field Communication (NFC): tap-to-pay



Smartwatch app development

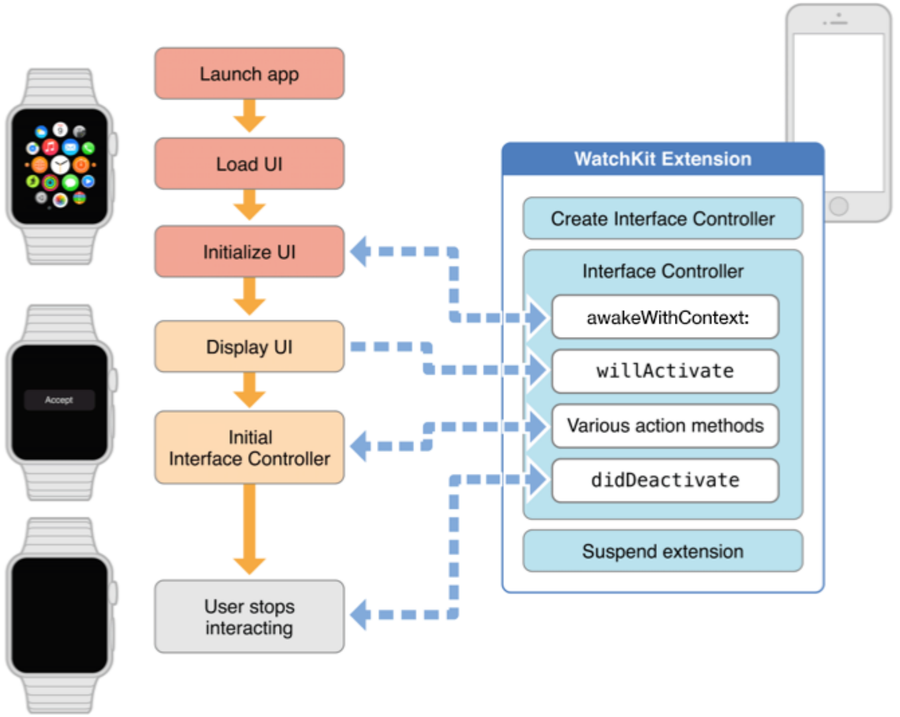
Development in practice

- Uses the native development tools
- Coordinated communication with companion devices
  - Smartwatches are (almost) always paired with a phone
  - Ask the phone to do expensive computation or communicate with servers
  - Example: what meeting do I have next? Let the phone ask Google Calendar’s server

Smartwatch app development

Development in practice

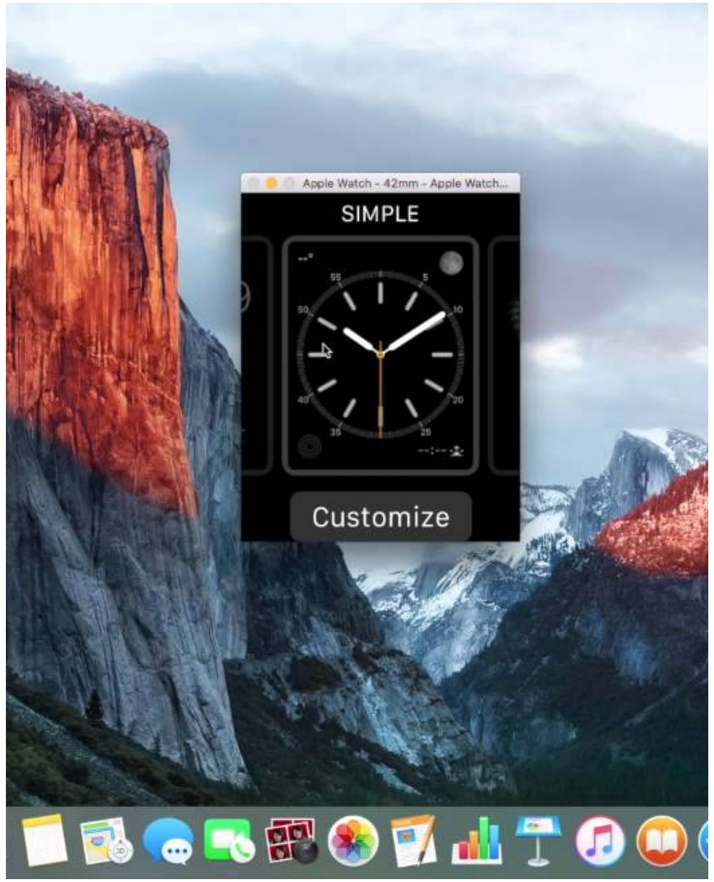
- Need to effectively handle lifecycle events (e.g., start up and shut down)
  - To save battery, smartwatches aggressively shut down apps which aren't being used
  - Example: you probably keep your email open on your phone/browser, but your watch will close out of it
  - Ever had issues with you calendar syncing to your watch? This is why



Smartwatch app development

Development in practice

- Development is often done with simulators/emulators
  - A “mock” version of a device which runs on your desktop
- This often makes debugging a pain
  - Sensors and native resources are hard to emulate/simulate
  - True of other devices, too



Virtual and Mixed Reality

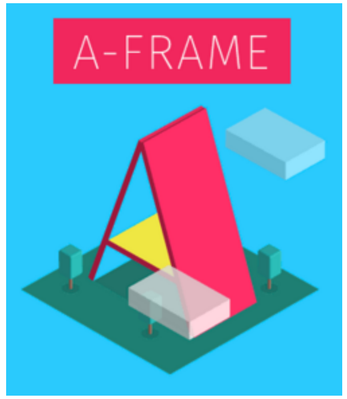
Virtual and Mixed Reality

Defining terms

- Virtual reality: completely replace real environment with a simulated one
  - HTC Vive, Oculus rift
- Mixed reality: overlays 3D graphics into a portion of the real world
  - Microsoft HoloLens, Apple Vision Pro

VR/MR app Development

- More likely to be native development
- But, web tools do exist
- Performance matters a lot
  - Needs to render interface at ~90-120 frames per second to avoid motion sickness (standard is 30-60 frames)



https://aframe.io/

VR/MR app Development

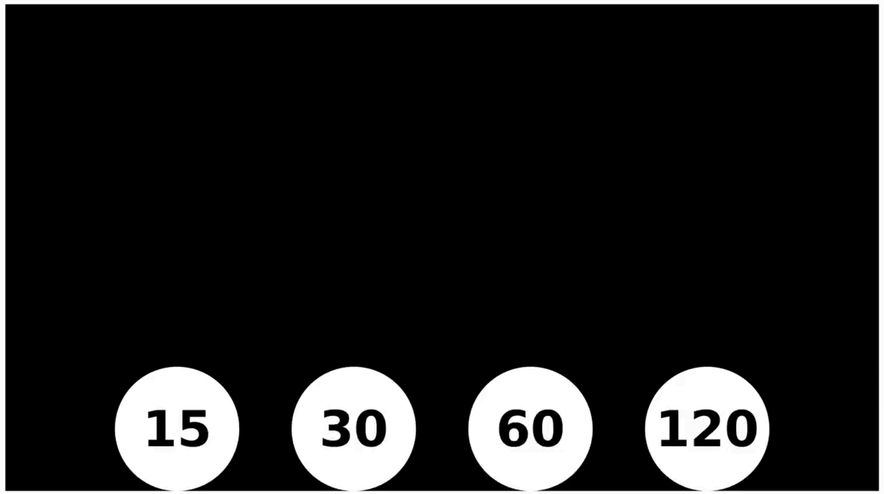
Why native?

- Some similar reasons to desktop/smartwatch
  - Access to native hardware resources (eye/hand tracking, depth sensing)
  - Input mechanisms might be non-standard (controllers, gestures)
- But also some unique ones

VR/MR app Development

Why native?

- Frame rate
  - Interfaces must render quickly to avoid inducing motion sickness
  - 90-120 frames per second (FPS), versus 30-60 on a web/mobile device
  - As best I understand, your vision has less context



https://www.youtube.com/watch?v=pfiHFqnPLZ4

VR/MR app Development

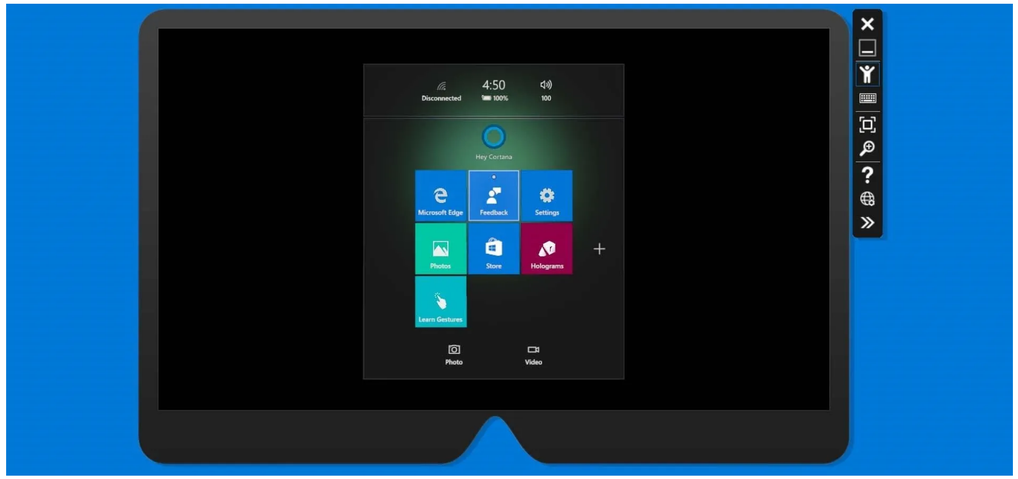
Why native?

- 2D interfaces -> 3D interfaces
  - Lots of math needed
  - Positioning of interface elements
  - Collision detection: what did you click on?
  - Typically not provided in web/mobile libraries, but common in 3D libraries or game engines

https://www.youtube.com/watch?v=pfiHFqnPLZ4

VR/MR app Development

- Emulators are pretty typical of VR/MR development
- You can technically develop *in* VR *for* VR, but this is more in the research phase than practice



Voice Assistants and Chatbots

Voice Assistant and Chatbot Apps

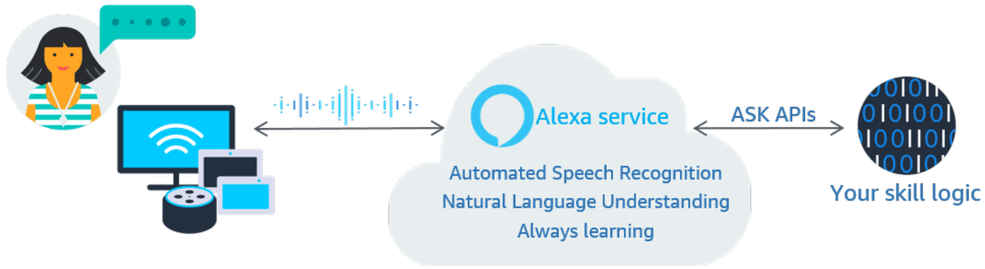
- Voice assistants are not as prominent as they were ~5 years ago, but still some important details
- But, extending open-domain LLM-driven chatbots are fairly similar
- Task-oriented chatbots: Siri, Alexa
- Open-domain chatbots: ChatGPT, Bard

https://developer.amazon.com/en-US/alexa/alexa-skills-kit

https://openai.com/api/

Voice Assistant and Chatbot Apps

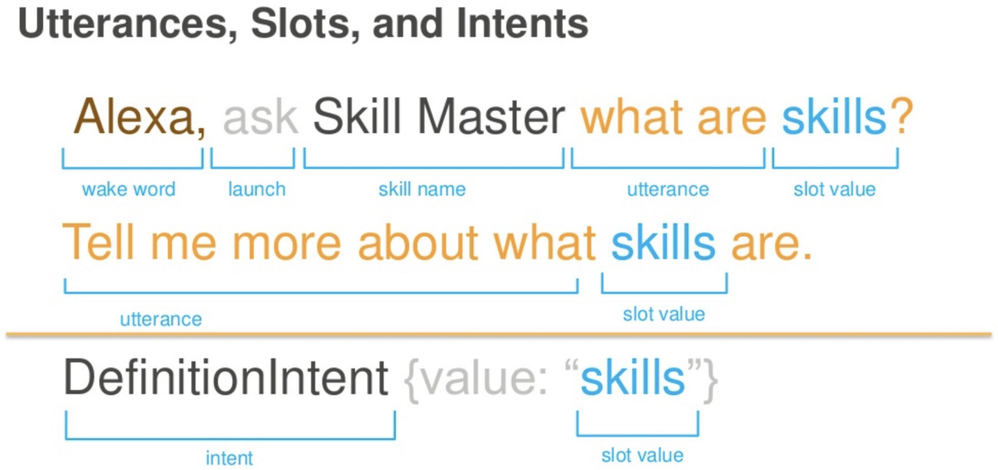
- Extensions often use web languages (JavaScript) for development
  - Technically NodeJS, which is server-side JavaScript, but will look familiar
- No visual interface, no need for HTML/CSS
  - Markup languages versus programming languages




```
while (true) {
  const message = await prompt('You: ');
  if (message) {
    messages.push({ role: 'user', content: message });
    const chatCompletion = await openai.Completion.create({
      model: 'text-davinci-002',
      prompt: messages,
      max_tokens: 1024,
      n: 1,
      stop: ['User:'],
      temperature: 0.8,
    });
    const answer = chatCompletion.choices[0].text;
    console.log(`ChatGPT: ${answer}`);
    messages.push({ role: 'assistant', content: answer });
  }
}
```

Voice Assistant and Chatbot Apps

- Key structure: *intents* and *slots*
  - User input is mapped to intents, e.g. functions
  - Any parameters are mapped to slots
- “Set a timer for 10 minutes”
  - Intent: “SetTimer” function
  - Slot: “10 minutes”
- Your app might have multiple intents



Voice Assistant and Chatbot Apps

- Platform provides natural language parsing capabilities
  - For voice assistants, audio to text
  - But also parsing out sentence syntax
- You might provide training examples
  - Key phrases, commands that your app should respond to
  - “Set a timer”, “Start a timer”, “Count down”, …

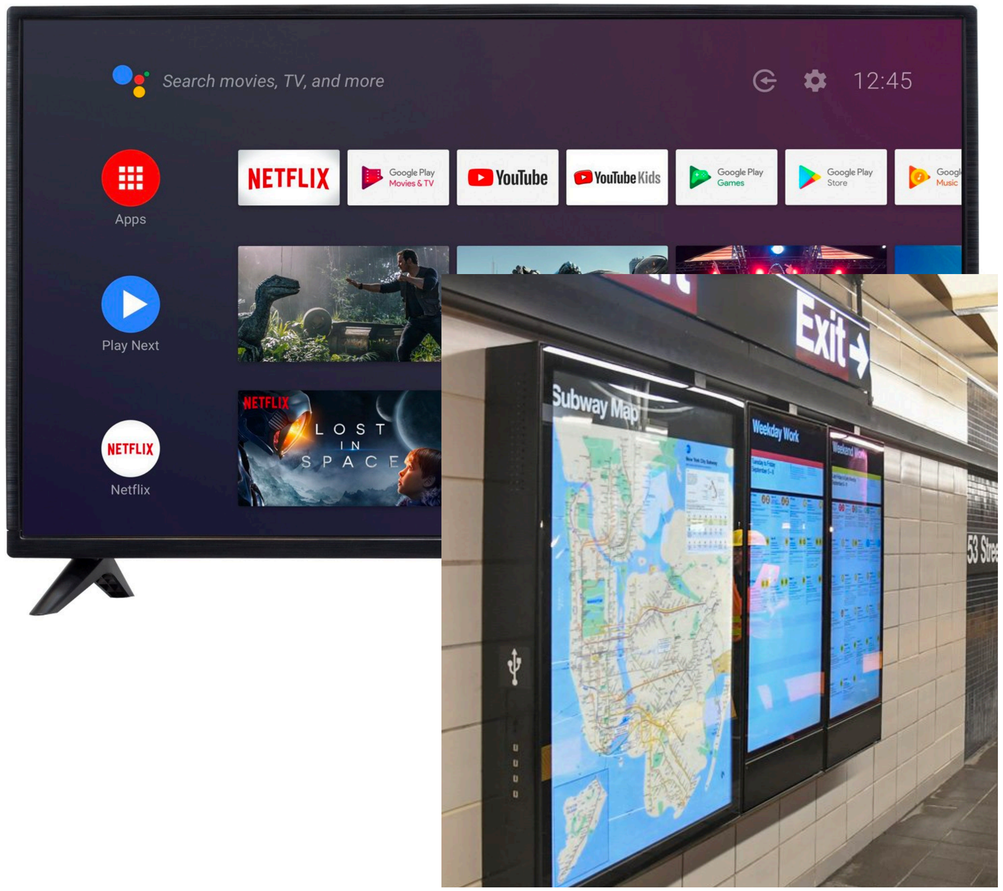
Voice Assistant and Chatbot Apps

- Very easy to emulate, since you can run a chatbot on both desktop and a dedicated device
- You might miss out a little on the situated use, background noise, etc.

Large and Public Displays

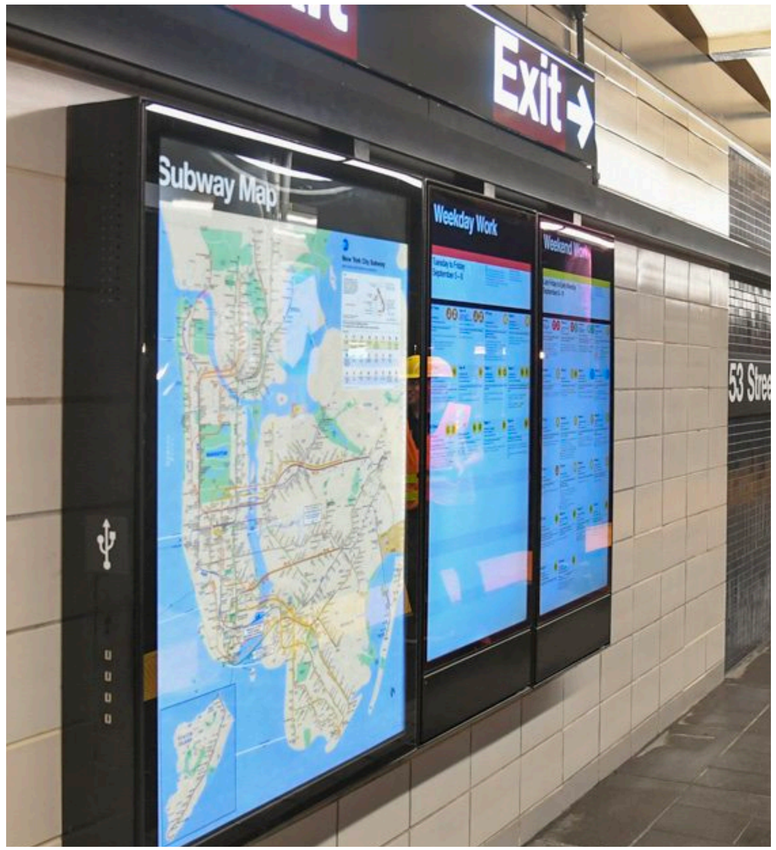
Large and Public Display Apps

- What sorts of large displays are out there?
  - Smart TVs
  - Interactive screens in public spaces
  - Public interactive art



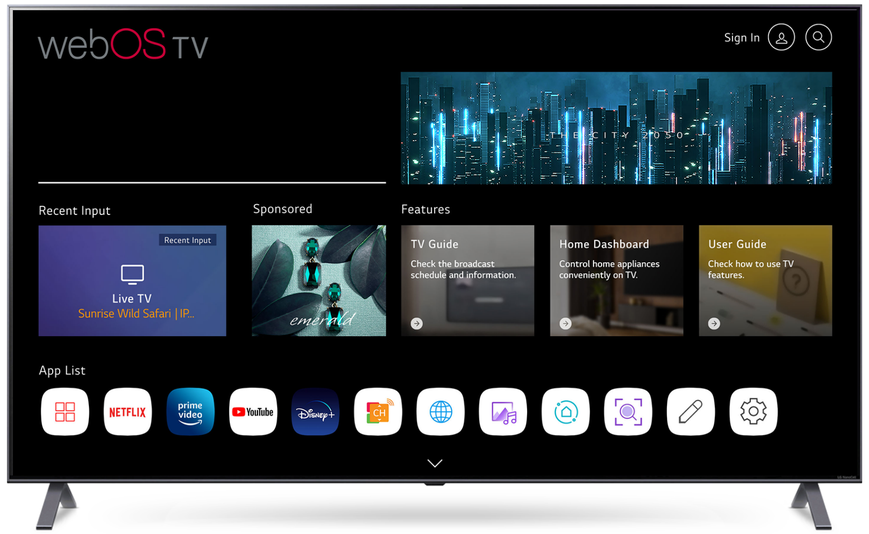
Large and Public Display Apps

- Public displays often don’t have internet access
  - Why? Makes them vulnerable to hacking or losing connectivity
- But, most have access to browsers, even offline
  - So you can still develop with HTML/CSS/JavaScript, just have to be careful with what’s exposed



Large and Public Display Apps

- Coding environments for Smart TVs are rather variable
- Some native platforms exist (Apple, Android), but other TVs leverage web stacks (LG, Samsung)
- Some TVs are limited in hardware, but not all



https://developer.android.com/tv

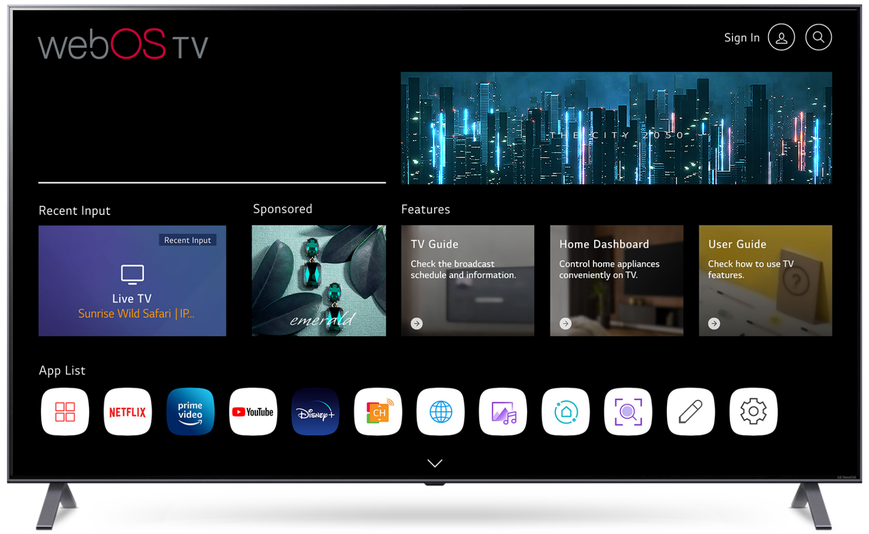
https://developer.apple.com/tvos/

https://www.lg.com/us/webos

https://www.samsung.com/latin_en/tvs/smart-tv/smart-hub-and-apps/

Large and Public Display Apps

- UX development needs for Smart TVs are lighter than other platforms
  - Limited input control (D-pad via controller, maybe some voice-based interaction)
  - Relatively few tasks: search for/ navigate content, play/pause/rewind
  - UX design is important, but development tasks are perhaps more straightforward



Large and Public Display Apps

- Public interactive art, etc. is more likely to be custom
  - Why? Lots of sensing in the environments to identify who is there and what they are doing
  - Non-standard outputs, not necessarily a single rectangular screen or projection
  - Creating a good experience requires performance

Larger reflections

Larger reflections

- HTML/CSS/JavaScript are still used in contexts beyond web and mobile
- But if you need high performance or access to features of the device, their utility is limited
  - Native platforms fill in those gaps
- People still write the code behind other device interfaces on their laptops, which introduces challenges around simulation/emulation
- Useful to ask what device(s) you need to support, and for what

Today’s goals

By the end of today, you should be able to…

- Articulate development practices on:
  - Desktops
  - Smartwatches
  - Virtual/mixed reality headsets
  - Large/public displays
  - Voice assistants/chatbots
- and how they differ from development practices for web and mobile technologies

IN4MATX 285:
Interactive Technology Studio

Practice: Beyond Web and Mobile