

IN4MATX 133: User Interface Software

Lecture:
Mobile Design, Device
Resources & Sensors, Sass

Today's goals

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

- Follow high-level guidelines for developing mobile interfaces
- Find and interpret platform-specific human interface guidelines
- Deploy an Ionic project to test an app on a mobile device
- Access device resources using a Capacitor Plugin
- Describe some of the sensors on modern smartphones
- Describe some ways in which sensors can be used

What makes a good user experience?

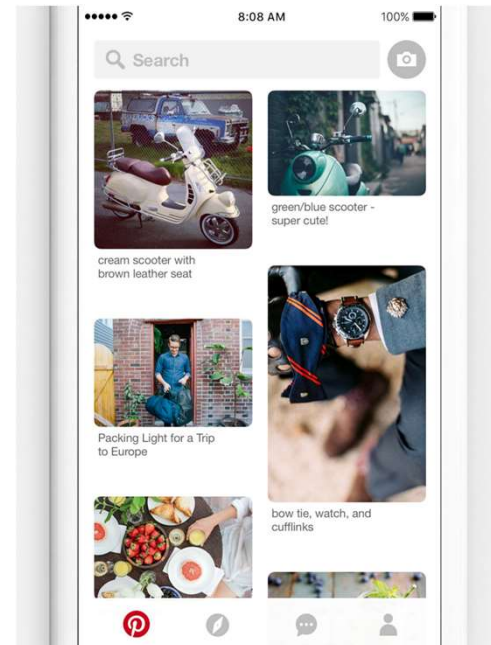
- It's not just the UI
 - The experience begins with the first time you launch an app or go to a website
- There are several components here
 - Initial impression (boot up to app start)
 - User interface
 - Visual design
 - Information presentation
 - The physical device and how it is used with the app

A few principles of mobile design

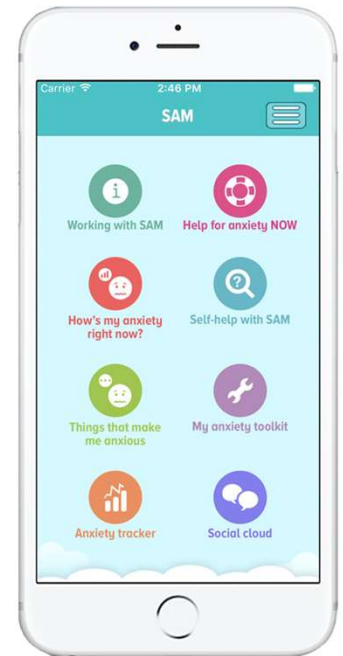
- A useful initial view
- The “uh-oh” button
- Error prevention
- Follow platform conventions

A useful initial view

- Give users clear calls to action
- Put useful content on the homepage
 - Pinterest's images
 - Put more than navigation buttons
- Make it easy to get back to the homepage
 - Bottom navbar, side navigation menu



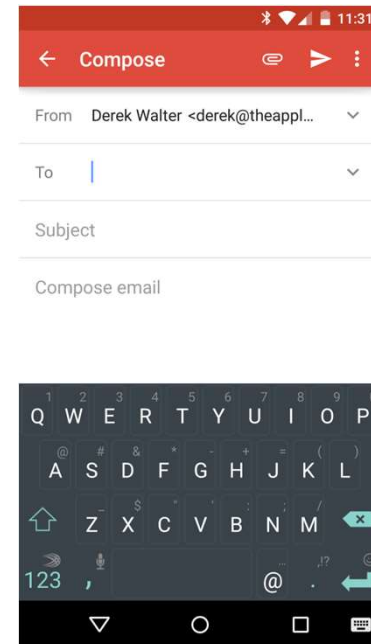
Pinterest



Anxiety management app

The “uh-oh” button

- Functions and buttons are often pressed by mistake
- Undo and redo should be easy
 - Gmail: “undo send”
- Navigating back a page should be easy
 - Breadcrumbs or back buttons (top left)



Gmail

Error prevention

- Providing input with small devices is difficult
 - Add in as much assistance as possible to aid with input
- Add input checks
 - How many digits are in that phone number? Credit card number?
- Use appropriate widgets
 - Date/time spinner
 - Sliders

Event name

Location

FROM
Wed, Sep 19, 2012 7:00am

TO
Wed, Sep 19, 2012 8:00am

ALL DAY

(GMT-4:00) Eastern Time *

GUESTS
Guests

Description

REPETITION
One-time event

REMINDEERS
15 minutes Notification

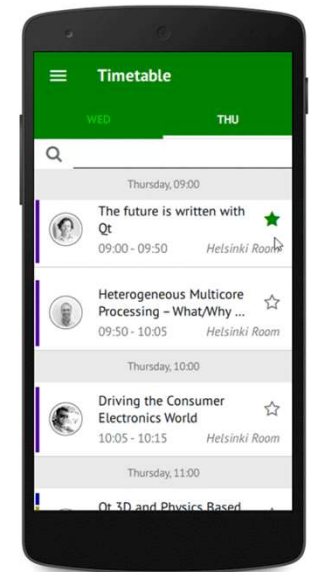
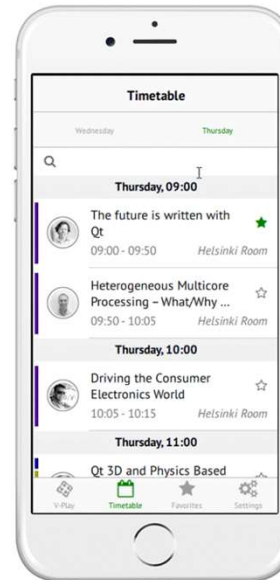
Add reminder

Date Picker Widget

Sep	03	2016
Oct	04	2017
Nov	05	2018

Follow platform conventions

- Users should not have to wonder whether different words, situations, icons, or actions mean the same thing
- Users should not have to remember app-specific navigation



iOS and Android platform conventions: Human Interface Guidelines

Human interface guidelines

- Created by web/mobile platform developers (Google, Apple)
- Key features:
 - Define rules for visual design and style
 - Specify interactions
 - Establish layout techniques
 - Provide consistency across the platform

Human interface guidelines

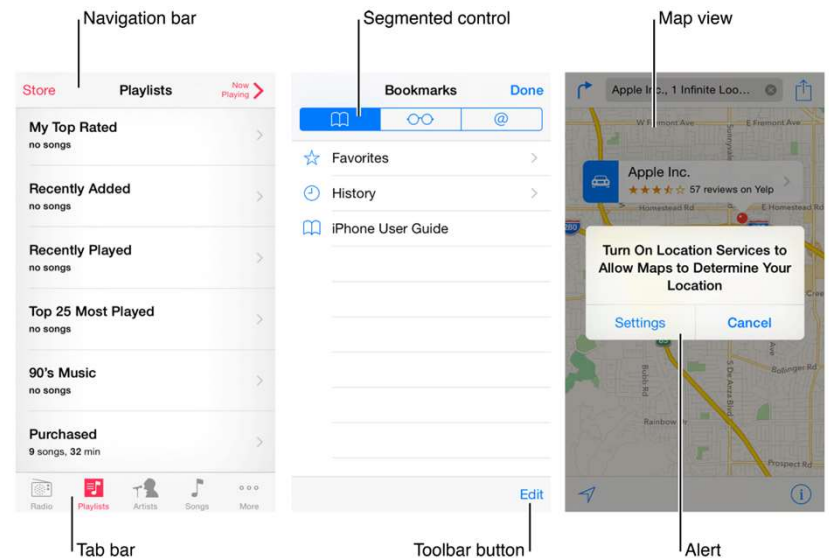
- HIGs are recommendations; you can choose to ignore them
 - The goal is to create an optimal experience for a device or platform
 - These guidelines most often follow best practices

iOS Human Interface Guidelines

- Content over UI
- Use the whole screen
- Single / simple colors
- Borderless buttons and widgets

Navigation

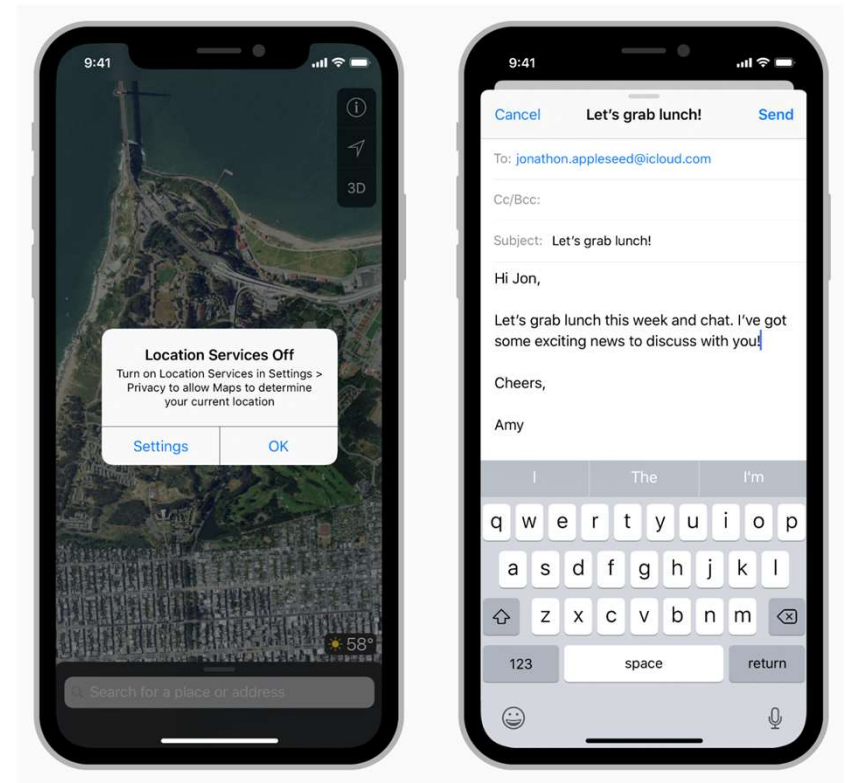
- Should be “natural”
- Use a navigation bar to traverse a hierarchy of data
- Use a tab bar for several peer categories
- Use a new page when that page is an instance of an item for another page



<https://developer.apple.com/design/human-interface-guidelines/ios/app-architecture/navigation/>

Modals

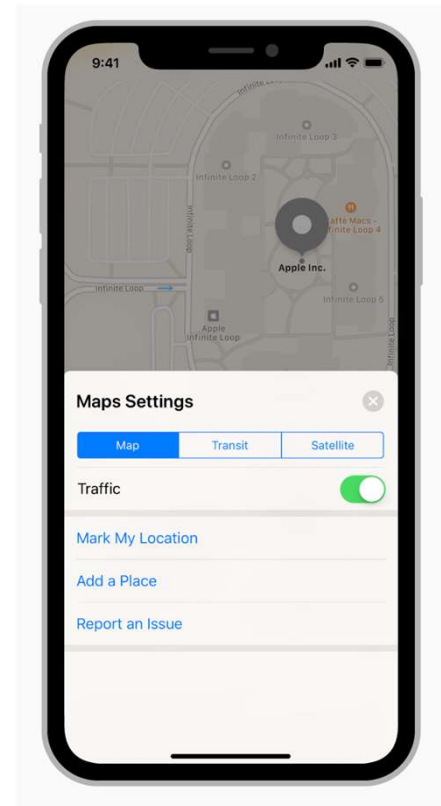
- Grab control of the experience until they are dismissed
- Meant to grab attention for doing one small, specific task
- Make sure the user can back out
- Respect notification wishes
- Use sparingly



<https://developer.apple.com/design/human-interface-guidelines/ios/app-architecture/modality/>

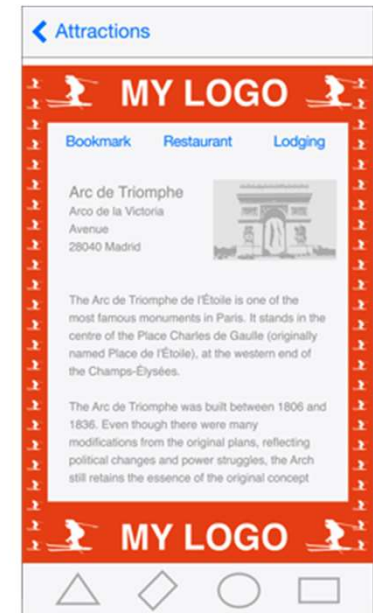
Interactivity

- Use a key color to denote interactive elements
- Denote “active” and “inactive” components differently
- Be aware of colorblindness



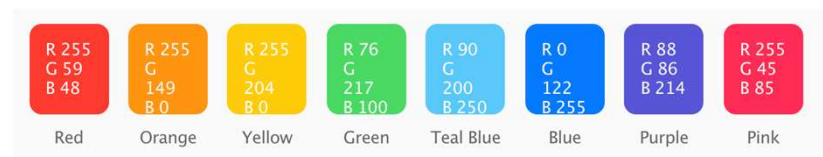
Branding

- It's important to be distinctive...
- But be careful not to pull a user out of the iOS experience
- Your app does not have to look like a default app, but...



Color and Typography

- Colors are great for grabbing attention, but can be overused
- Use complementary colors
 - Palette definers like paletton.com
- Use a single typeface (font), if possible
 - Built-in fonts are just fine
 - Use font size, and color and weight (bold) to highlight information









<https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/color/>

Icons

- A good icon is important
- Keep background simple
- Only use words if they are essential or part of a logo
- Leave your icon out of the interface
- When appropriate, use system icons in the interface itself
 - Use as intended



Icon	Name	Meaning
	Action (Share)	Shows a modal view containing share extensions, action extensions, and tasks, such as Copy, Favorite, or Find, that are useful in the current context.
	Add	Creates a new item.
	Bookmarks	Shows app-specific bookmarks.
	Camera	Takes a photo or video, or shows the Photo Library.
Cancel	Cancel	Closes the current view or ends edit mode without saving changes.
	Compose	Opens a new view in edit mode.
Done	Done	Saves the state and closes the current view, or exits edit mode.
Edit	Edit	Enters edit mode in the current context.
	Fast Forward	Fast-forwards through media playback or slides.

<https://developer.apple.com/design/human-interface-guidelines/ios/icons-and-images/app-icon/>

Google Material Design

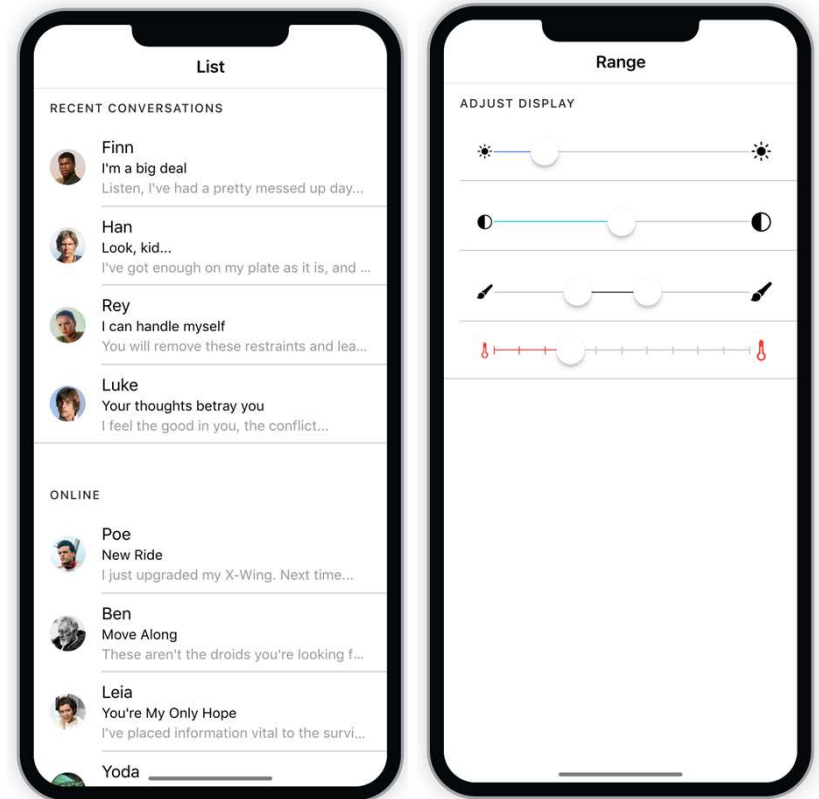
- Philosophy: interface should look like layers on a sheet of paper
 - Have 3D depth and motion
- Follows many of the same patterns as iOS design in terms of interaction
 - Limited use of modals
 - Use color to emphasize content
 - Be subtle with branding
- But, there are differences – study guidelines before deployment!

<https://material.io/design/>

Capacitor Plugins

Ionic components

- Ionic provides Angular-style components for a lot of interface elements common in mobile interfaces
- Lists, buttons, sliders, tabs, modal dialogs, search bars, much more



<https://ionicframework.com/docs/components/>

Ionic + Capacitor

- Capacitor Provides libraries for connecting to device resources in the form of **plugins**
- Possible to use Capacitor alongside ionic native wrapping Cordova plugins
- hundreds of plugins
 - official or community
 - some with known issues

<https://ionicframework.com/docs/native/>



Capacitor Setup

- Adding capacitor to an existing Ionic project:
 - > *cd [project folder]*
 - > *ionic integrations enable capacitor*
- Capacitor builds native “projects” based on web build (folder www)
 - > *ionic build*
- After Ionic builds, use Capacitor to create native projects
 - > *ionic capacitor add [android or ios]*
- After each significant code change, need to update native projects:
 - > *ionic capacitor copy [android or ios]*

Ionic and Capacitor Deployment

- The sync command will both copy and update plugins and dependencies for both Android and iOS. Also, “cap” can be used instead of “capacitor”:

> ionic cap sync

- Commands to open native projects using native IDEs

> ionic cap open [android or ios]

- Live reload keeps native IDE in sync with ionic project and updates deployed emulators:

> ionic cap run [android or ios] -l --external

Platform

- Ionic has an injectible service for detecting what platform(s) the app is running on
- Platforms are not mutually exclusive
 - Mobile, iOS, Android
 - iPad, tablet

<https://ionicframework.com/docs/angular/platform>

Capacitor Plugins

- Some (few) are maintained officially
- Others are maintained by the community
- As a result, quality varies immensely
- Features may not work as expected
- Plugins are abstractions for native resources, so be aware of how each is used on either iOS or Android

Local Storage: Preferences

- Preferences
 - Simple key/value storage for temporary data
 - Should use for caching, not for long term storage
 - User can delete
- Installation
 - `> npm install @capacitor/preferences`

<https://capacitorjs.com/docs/next/apis/preferences>

Local Storage: Preferences

- Add to a component

```
import { Preferences } from '@capacitor/preferences';
```

```
Preferences.get({key:'keyName'}).then((data) => {  
  console.log(data.value);  
});
```

```
Preferences.set({key:'keyName', value:'value'}).then(() => {  
  console.log("set value");  
});
```

Local Storage: Ionic Storage

- Ionic Storage
 - Storage abstraction layer for **permanent** data
 - Key/value storage or more complex (e.g., SQLite)
 - Defaults to IndexedDB and localStorage, depending on availability
 - Both are nosql style databases
- Installation
 - `> npm install @ionic/storage-angular`

<https://github.com/ionic-team/ionic-storage>

https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API

Local Storage: Ionic Storage

- Add to a component

```
import { Storage } from '@ionic/storage-angular';

//in class

this._storage.get({key:'keyName'}).then((data) => {
  console.log(data.value);
});

this._storage.set({key:'keyName', value:'value'}).then(() => {
  console.log("set value");
});
```

Three Official Capacitor Plugins

- Camera
- Local Notification
- Sharing

There are others, but we will just cover these three.

Taking a Picture

- To use webcam, install PWA lib:
> npm install @ionic/pwa-elements
- To use webcam, install Camera lib:
> npm install @ionic/camera
- Import PWA lib in main.ts

```
import { defineCustomElements } from '@ionic/pwa-elements/loader';
```

```
// Call the element loader after the platform has been bootstrapped  
defineCustomElements(window)
```

<https://capacitorjs.com/docs/web/pwa-elements>

Taking a Picture

- Import plugins from Capacitor to your desired component

```
import {  
  CameraResultType,  
  Camera,  
} from '@capacitor/camera';
```

Taking a Picture

```
import { Camera, CameraResultType } from '@capacitor/camera';

async takePicture() {
  const image = await Camera.getPhoto({
    quality: 90,
    allowEditing: true,
    resultType: CameraResultType.Uri
  });
  // image.webPath will contain a path that can be set as an image src.
  // You can access the original file using image.path, which can be
  // passed to the Filesystem API to read the raw data of the image,
  // if desired (or pass resultType: CameraResultType.Base64 to getPhoto)
  var imageUrl = image.webPath;
  // Can be set to the src of an image now
  imageElement.src = imageUrl;
}
```

<https://capacitorjs.com/docs/apis/camera>

Local Notification

- Goal: send a notification to the phone
- Could be used to remind someone to journal their sleepiness, for example
- To use, install with:
 - `> npm install @capacitor/local-notifications`

<https://capacitorjs.com/docs/apis/local-notifications>

Local Notification

- Import Plugin in a service or component

```
import {LocalNotifications } from '@capacitor/local-notifications';
```

- Prompt user to authorize notifications:

```
await LocalNotifications.requestPermissions();
```

<https://capacitorjs.com/docs/apis/local-notifications>

Local Notification

```
const notifs = await LocalNotifications.schedule({
  notifications: [
    {
      title: "Title",
      body: "Body",
      id: 1,
      schedule: { at: new Date(Date.now() + 1000 * 5)
    },
    sound: null,
    attachments: null,
    actionTypeId: "",
    extra: null
  ]
});
console.log('scheduled notifications', notifs)
```



Can schedule
for the future

<https://capacitorjs.com/docs/apis/local-notifications>

Sharing

- Goal: export data from your app to a social app on the device
- Could be used to share photos to Facebook
- Could be used to share text in a text message
- Uses Web Share API

<https://capacitorjs.com/docs/apis/share>

<https://web.dev/web-share/>

Sharing

- Support is “nuanced”
- Use feature detection rather than assume a particular method is supported
- Check out the compatibility list here:
 - https://developer.mozilla.org/en-US/docs/Web/API/Navigator/share#browser_compatibility

<https://capacitorjs.com/docs/apis/share>

<https://web.dev/web-share/>

Sharing

- Import Plugin

```
import { Share } from '@capacitor/share';
```

- Call Share.share() method with content to be shared:

```
let shareRet = await Share.share({  
  title: 'See cool stuff',  
  text: 'Really awesome thing you need to see right meow',  
  url: 'http://ionicframework.com/',  
  dialogTitle: 'Share with buddies'  
})
```

<https://capacitorjs.com/docs/apis/share>

<https://web.dev/web-share/>

Demo

Plugin Issues

- There are many issues with Capacitor plugins (see link below)
- Only a limited set of functionalities are enabled
- Plugins may be unreliable

<https://github.com/ionic-team/capacitor/issues>

Premier Plugins

- The company behind Ionic maintains a set of plugins
 - Ionic's team is behind Capacitor
- They are presumably more reliable, but this comes at a cost

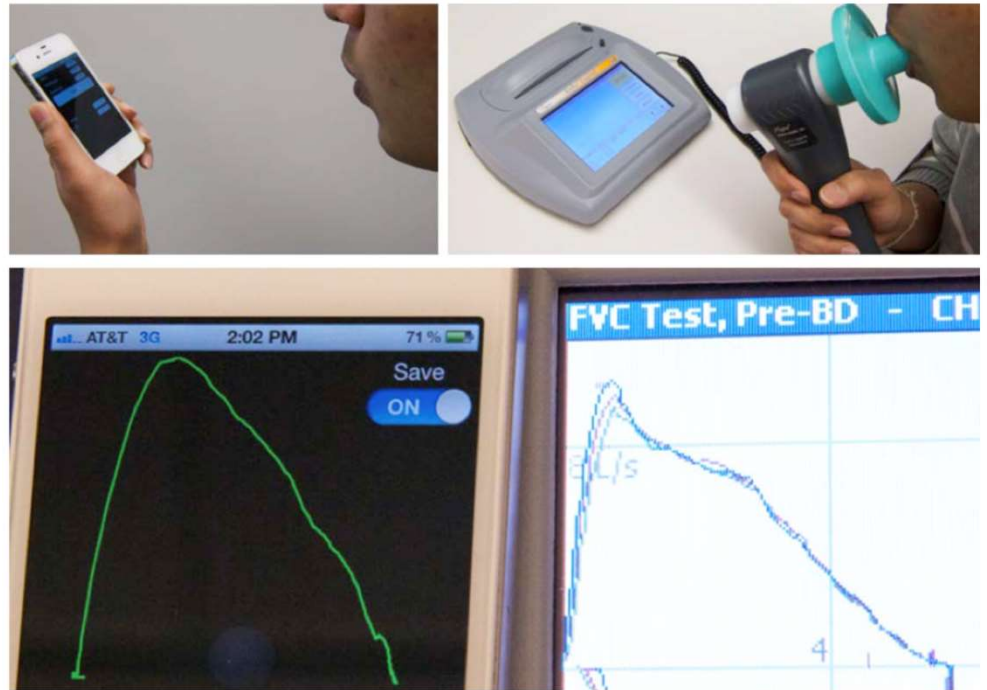
<https://capacitorjs.com/enterprise>

Appropriating Sensors in Research

Appropriating Sensors in Research

SpiroSmart

- Lung function (asthma/blockage) via a microphone



<https://dl.acm.org/citation.cfm?id=2370261>

Eric C. Larson, Mayank Goel, Gaetano Boriello, Sonya Heltshe, Margaret Rosenfeld, Shwetak N. Patel.

SpiroSmart: Using a Microphone to Measure Lung Function on a Mobile Phone. UbiComp 2012

Appropriating Sensors in Research

BiliCam

- Jaundice in newborns via camera and a calibration card



<https://dl.acm.org/citation.cfm?id=2632076>

Lilian de Greef, Mayank Goel, Min Joon Seo, Eric C. Larson, James W. Stout, James A. Taylor, Shwetak N. Patel.

BiliCam: Using Mobile Phones to Monitor Newborn Jaundice. UbiComp 2014

Appropriating Sensors in Research

Why?

- Medical devices are expensive and inaccessible
- Phones are widely available
 - ~40% of the world owns a smartphone today
 - Can enable these tests in lower-resource countries or counties
 - Can enable at-home tests and continuous monitoring
- Regulation is a separate and important issue

Today's goals

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

- Follow high-level guidelines for developing mobile interfaces
- Find and interpret platform-specific human interface guidelines
- Deploy an Ionic project to test an app on a mobile device
- Access device resources using a Capacitor Plugin
- Describe some of the sensors on modern smartphones
- Describe some ways in which sensors can be used