

# **IN4MATX 133: User Interface Software**

**Lecture 15:**  
**Device Resources & Sensors**

Professor Daniel A. Epstein  
TA Eunkyung Jo  
TA Lucas de Melo Silva

# Announcements

- A3 grades will be posted soon, probably after class on Wednesday
- Course evaluation form is up
  - <https://evaluations.eee.uci.edu/takeLanding/HYKXET>
- A4 due Friday
  - Office hours today, tomorrow, & Friday

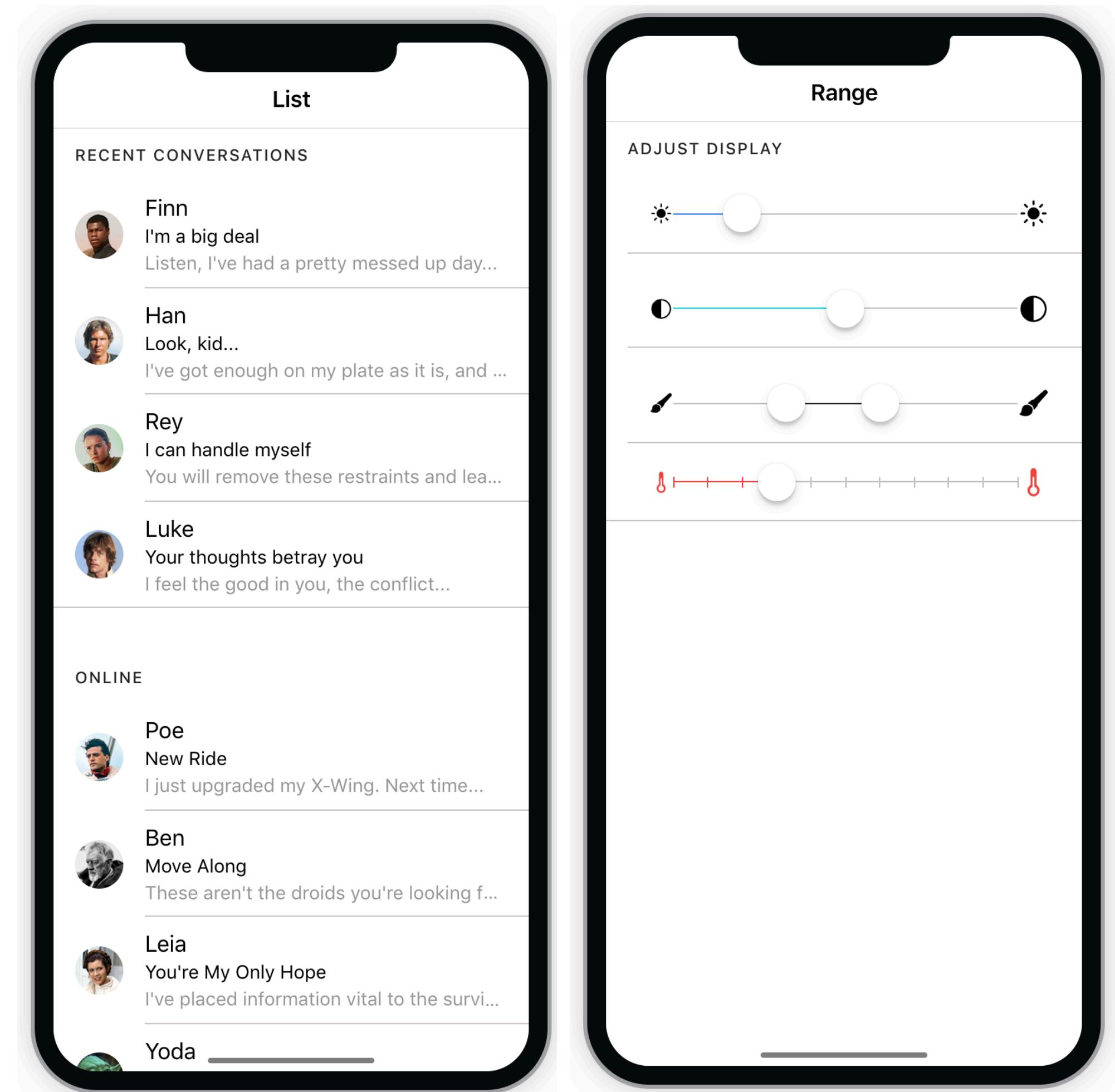
# Today's goals

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

- 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

# Ionic

- 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/>

<https://capacitorjs.com/docs/cordova/using-cordova-plugins>

# 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:

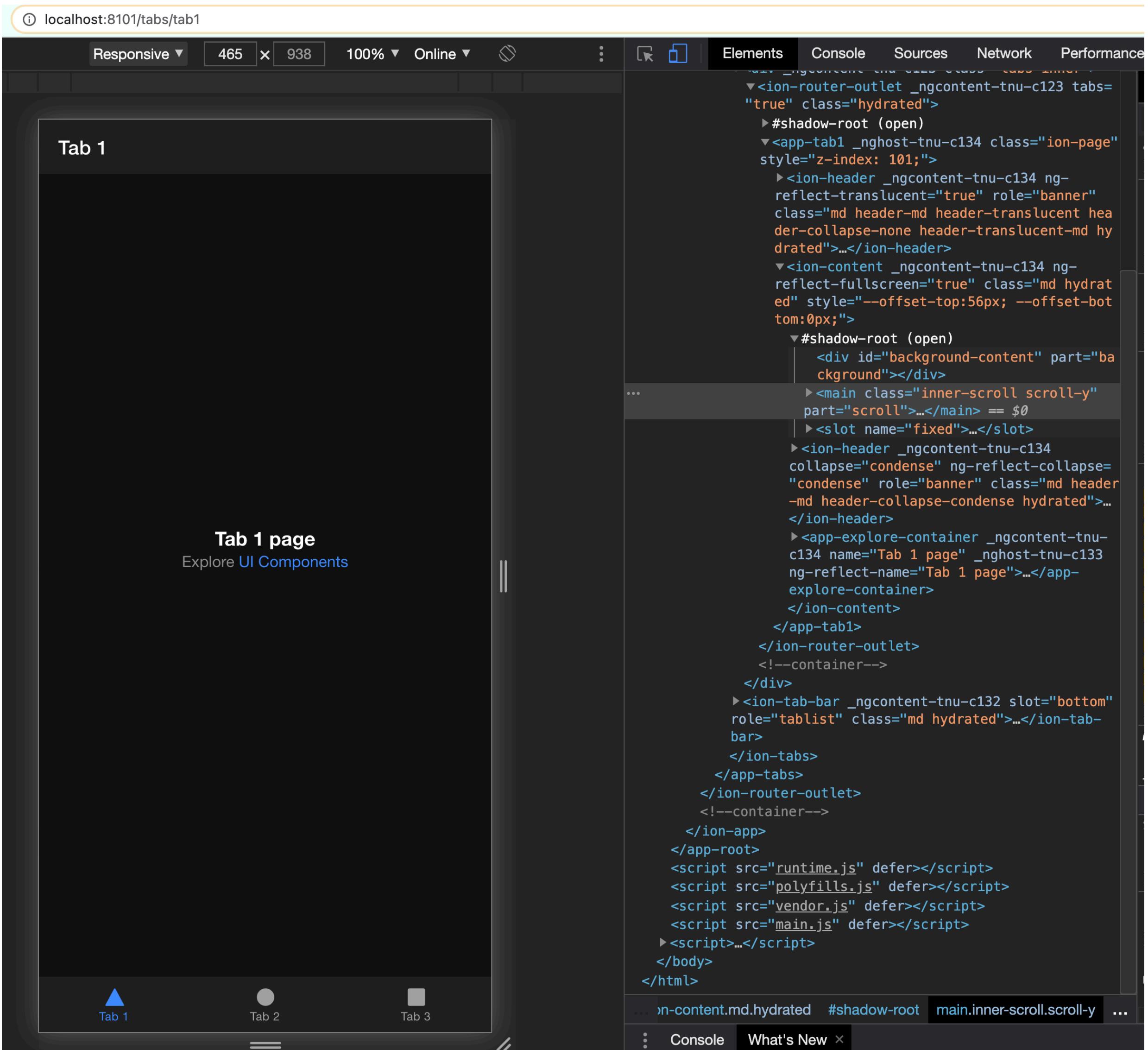
```
$ ionic cap run [android or ios] -l --external
```

# \$Ionic Serve

- Deploys the web project to be accessed in the browser
- Capacitor allows for runtime evaluation of platform and available resources
  - Plugins will try to use native features, but revert to web APIs otherwise

<https://ionicframework.com/docs/cli/commands/serve>

<https://capacitorjs.com/docs/basics/progressive-web-app>



# 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

Platform Name	Description
android	a device running Android
capacitor	a device running Capacitor
cordova	a device running Cordova
desktop	a desktop device
electron	a desktop device running Electron
hybrid	a device running Capacitor or Cordova
ios	a device running iOS
ipad	an iPad device
iphone	an iPhone device
mobile	a mobile device
mobileweb	a web browser running in a mobile device
phablet	a phablet device
pwa	a PWA app
tablet	a tablet device

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

# Capacitor Plugins

- Some (few) are maintained officially
- Others are maintained by the community
- V3 is on the horizon and will solve some known bugs
- As a result, quality varies immensely
- Features may not work as expected
- Might be better than using Ionic Native and Cordova, these have lagged in updates and compliance with newer platform versions

# Three Official Capacitor Plugins

- Camera
- Local Notification
- Sharing
- (There are others, just three we'll go through)

# Taking a Picture

- To use webcam, install PWA lib:

- install PWA lib

```
$npm install @ionic/pwa-elements
```

- Import in main.ts

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

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

# Taking a Picture

- Import plugins from Capacitor

```
import {
  Plugins,
  CameraResultType,
  CameraPhoto,
  CameraSource,
} from '@capacitor/core';

const { Camera } = Plugins;
```

# Taking a Picture

```
import { Plugins, CameraResultType } from '@capacitor/core';

const { Camera } = Plugins;

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

# Local Notification

## Local Notifications



The Local Notification API provides a way to schedule “local” notifications – notifications that are scheduled and delivered on the device as opposed to “push” notifications sent from a server.

Local Notifications are great for reminding the user about a change in the app since they last visited, providing reminder features, and delivering offline information without the app being in the foreground.

- `schedule(...)`
- `getPending()`
- `registerActionTypes(...)`
- `cancel(...)`
- `isEnabled()`
- `createChannel(...)`
- `deleteChannel(...)`
- `listChannels()`
- `requestPermission()`
- `addListener(...)`
- `addListener(...)`
- `removeAllListeners()`
- `Interfaces`

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

# Local Notification

- Import Plugin in a service or component

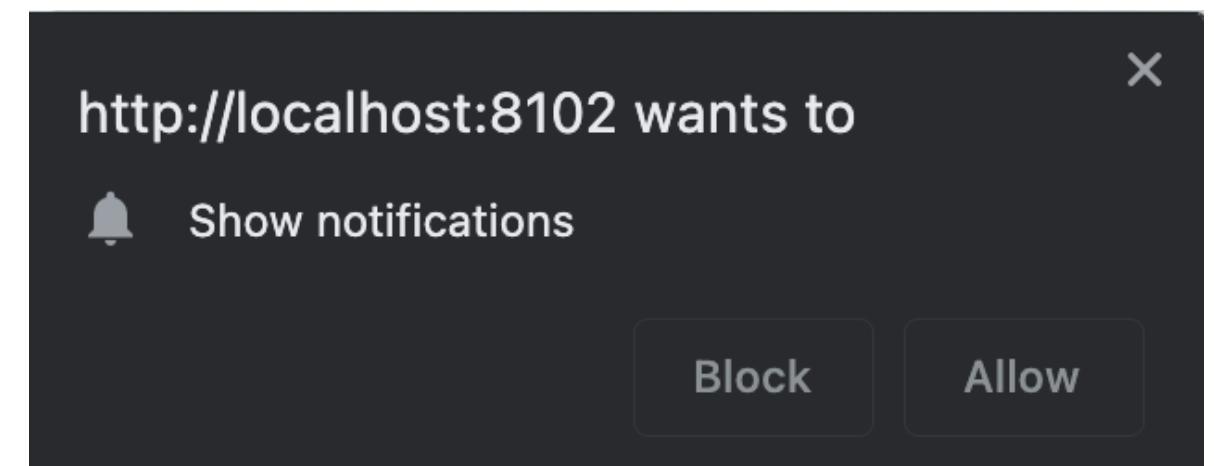
```
import { Plugins } from '@capacitor/core';
```

```
const { LocalNotifications } = Plugins;
```

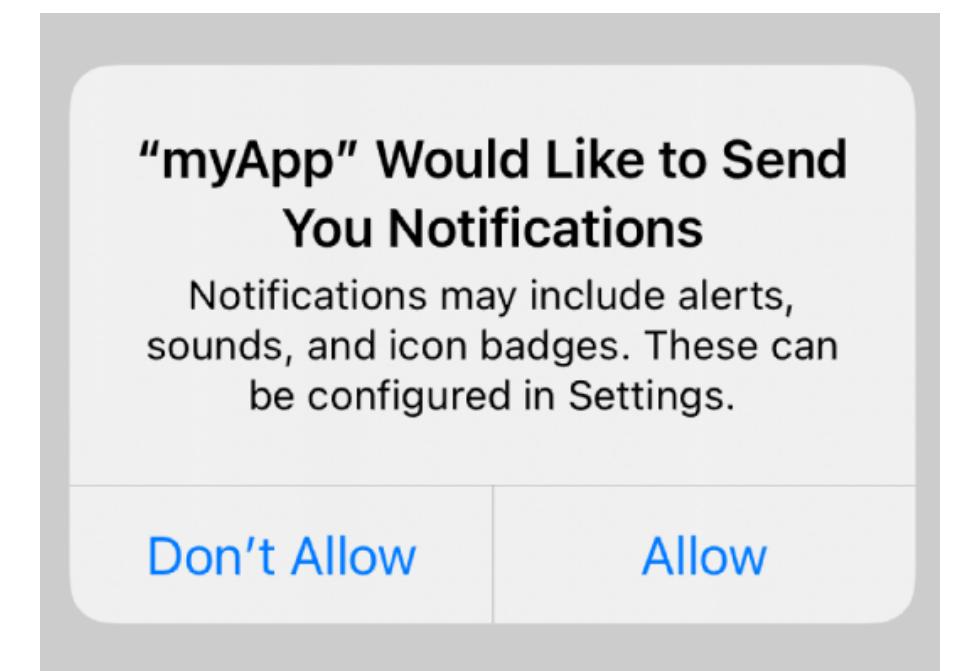
- Prompt user to authorize notifications:

```
async ngOnInit() {
  await LocalNotifications.requestPermission();
}
```

Prompt on Web:



Prompt on iOS:



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

# Local Notification

```
import { Plugins } from '@capacitor/core';
const { LocalNotifications } = Plugins;

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
  - “Support is currently spotty”
  - Can share URLs, Text, Files (e.g., picture)
  - Not available in some desktop browsers (but usually on phone’s browser)

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

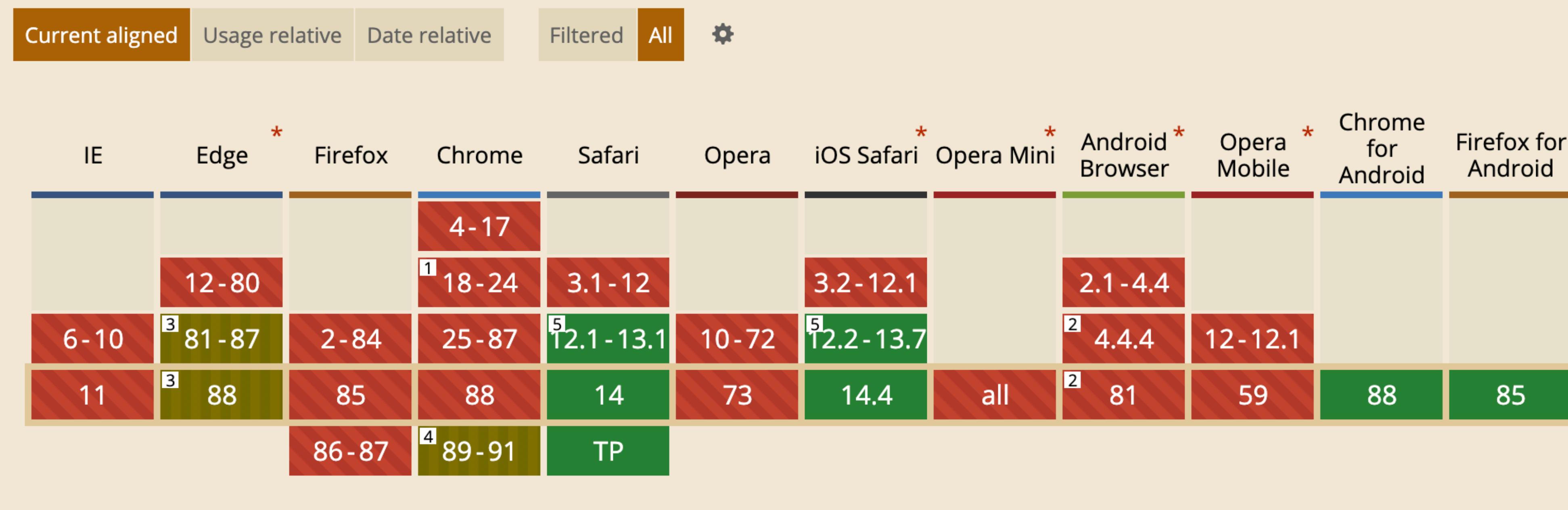
<https://web.dev/web-share/>

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

# Sharing

## Web Share API - WD

A way to allow websites to invoke the native sharing capabilities of the host platform



# Sharing

- Import Plugin

```
import { Plugins } from '@capacitor/core';

const { Share } = Plugins;
```

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

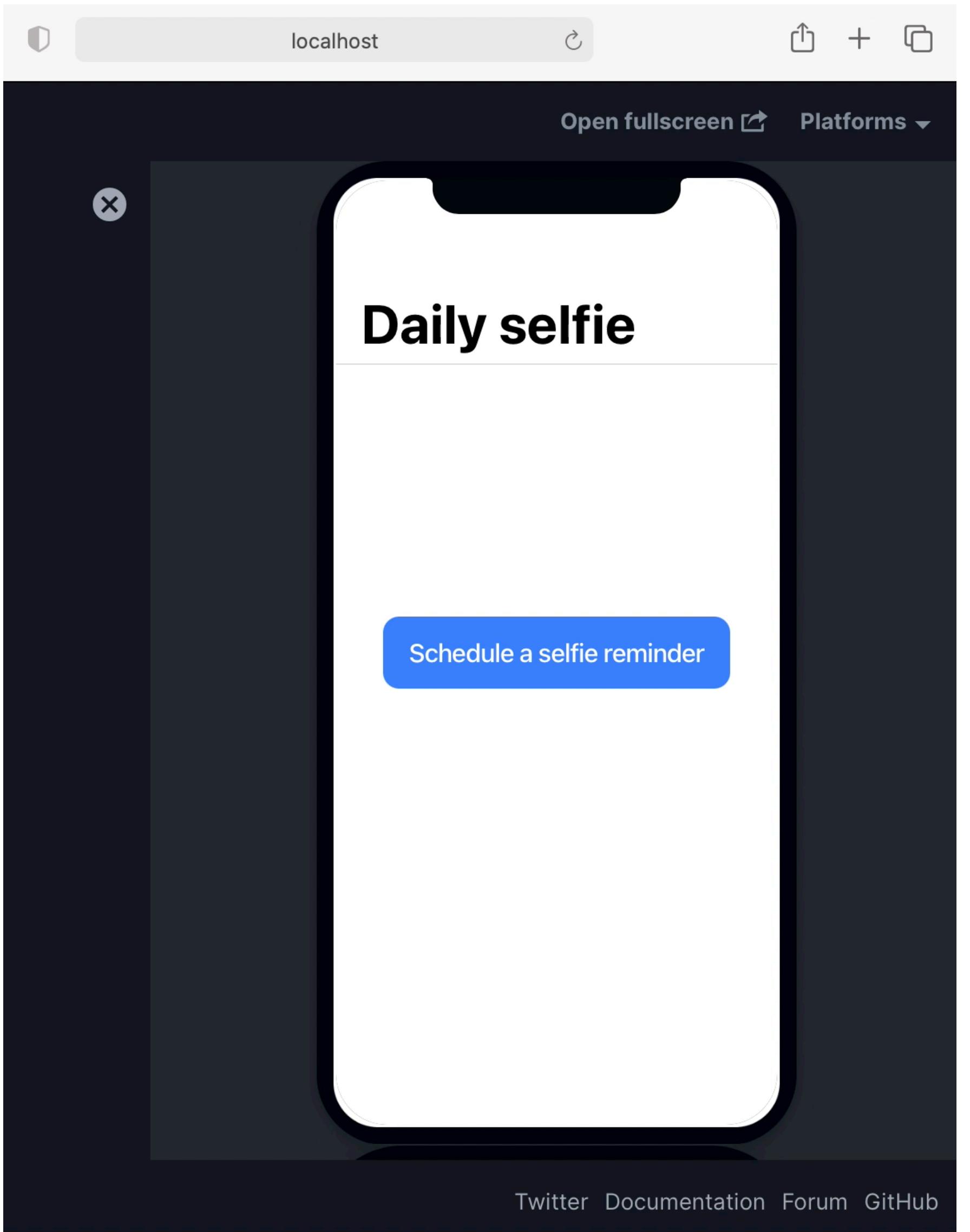
```
import { Plugins } from '@capacitor/core';
const { Share } = Plugins;

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/>

# Plugins



# Plugin Issues

- There are many issues with Capacitor plugins (as with Cordova)
  - For example, the Motion plugin uses only the web API and has some platform permission challenges
- Only a limited set of functionalities are enabled
  - Can't share a file, for instance
- Plugins may be unreliable
  - It's the reality of relying on open-source tools
- Capacitor V3 is on the horizon and seeks to solve some known issues

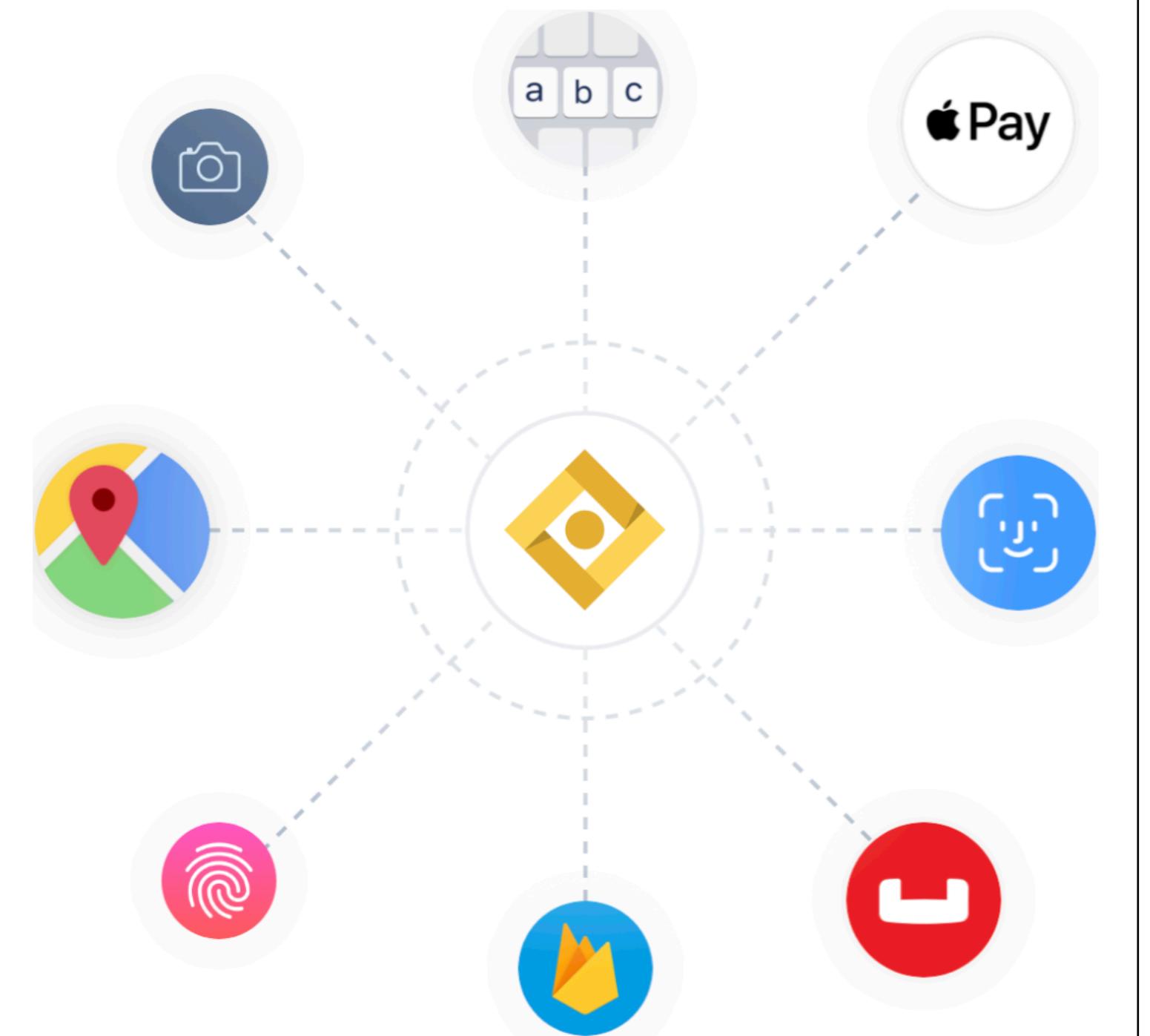
<https://github.com/ionic-team/capacitor/issues>

# 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

## Premier, supported native plugins

Native features maintained by Ionic's team of native experts. Active subscribers get ongoing updates to supported [Capacitor](#) and [Cordova](#) plugins, to keep pace with OS and API changes, and evolving device standards.



<https://capacitorjs.com/enterprise>

<https://ionicframework.com/enterprise>

# **Comparing to React Native's Plugins**

# React Native Libraries

- React Native includes a few libraries for accessing device resources
- Examples:
  - CameraRoll
  - AsyncStorage (device storage)
  - Geolocation (GPS)
- The rest are installed through plugins which look similar to Ionic's

<https://reactnative.dev/>

# React Native Plugins

- Used and installed in roughly the same way: install and link
  - `npm install react-native-sensors`
  - `react-native link react-native-sensors`
  - `npm install react-native-notification`
- (notifications require manual linking in Xcode or Android build)
- Notifications plugin only has 7 open issues,  
but also a community-supported plugin

<https://github.com/react-native-sensors/react-native-sensors>

<https://github.com/wix/react-native-notifications>

# Question



prefer

**When might developing a hybrid app be preferable to a native app?**

- A When you need to access a lot of native device resources
- B When you don't need any native device resources
- C When you can use well-maintained libraries to access native resources
- D When you need high performance
- E I don't know, this lecture has scared me away from making hybrid apps

# Thoughts on native resources

- The state of native support is just okay
- You could fork (copy) a broken or incomplete plugin and patch it yourself
- Is this better or worse than having to write the plugin yourself?
- This is the clear downside to building hybrid apps rather than native
  - Device libraries can't be used directly
  - Either need to rely on community libraries or fill in the missing pieces

# Thoughts on native resources

- Hybrid apps and PWAs are already pretty popular for development
- I wouldn't be surprised if it becomes easier to access native resources in the future

# Strengths of hybrid apps

- Can share a codebase between web and mobile
- Can save time and effort (sometimes)
- Easily design for various form factors
- Access to some device capabilities

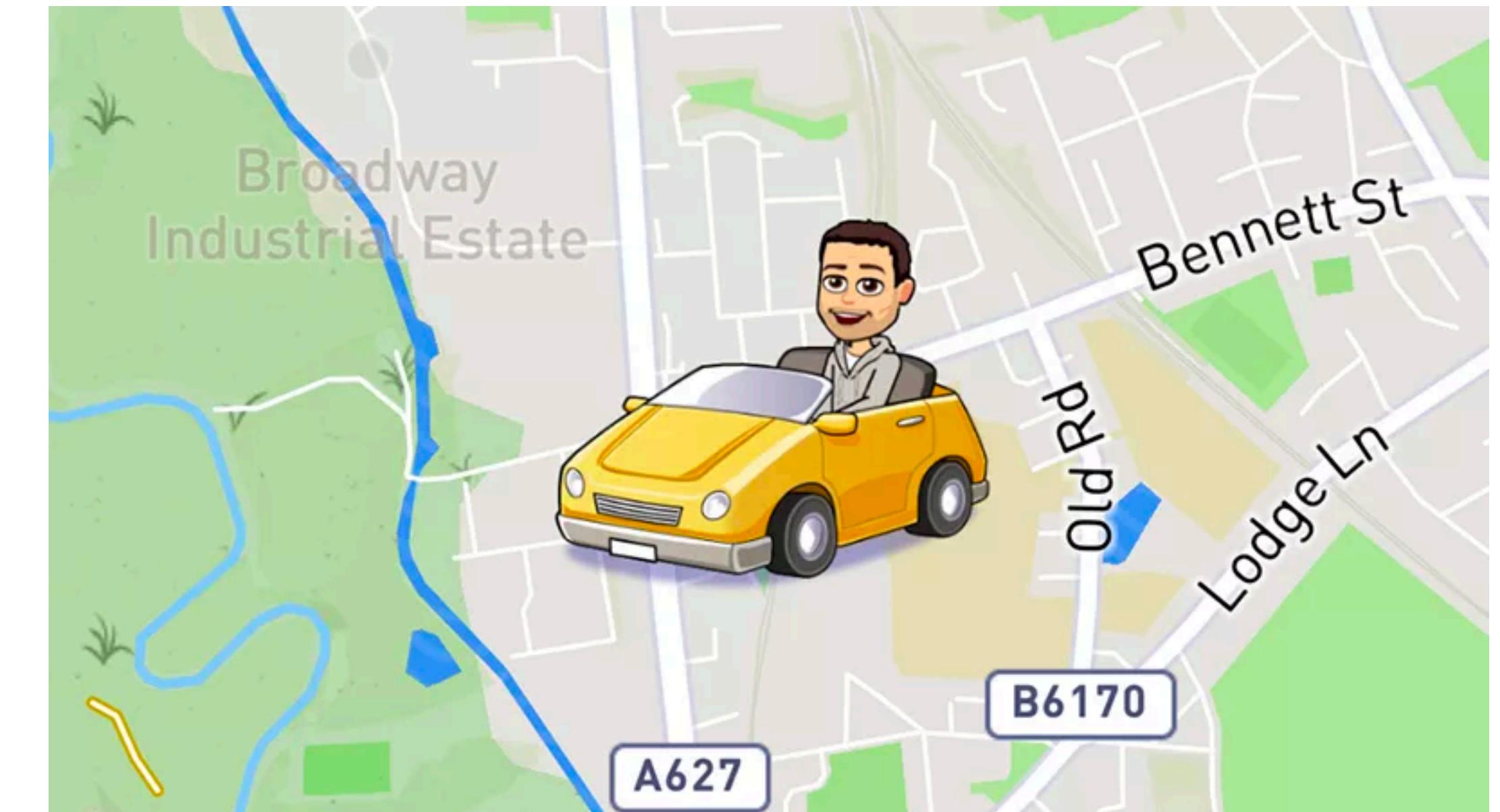
# Weaknesses of hybrid apps

- Performance issues
- Inconsistency with platform
- Limited access to device capabilities

**Modern phones include a lot of sensors**

# Sensors

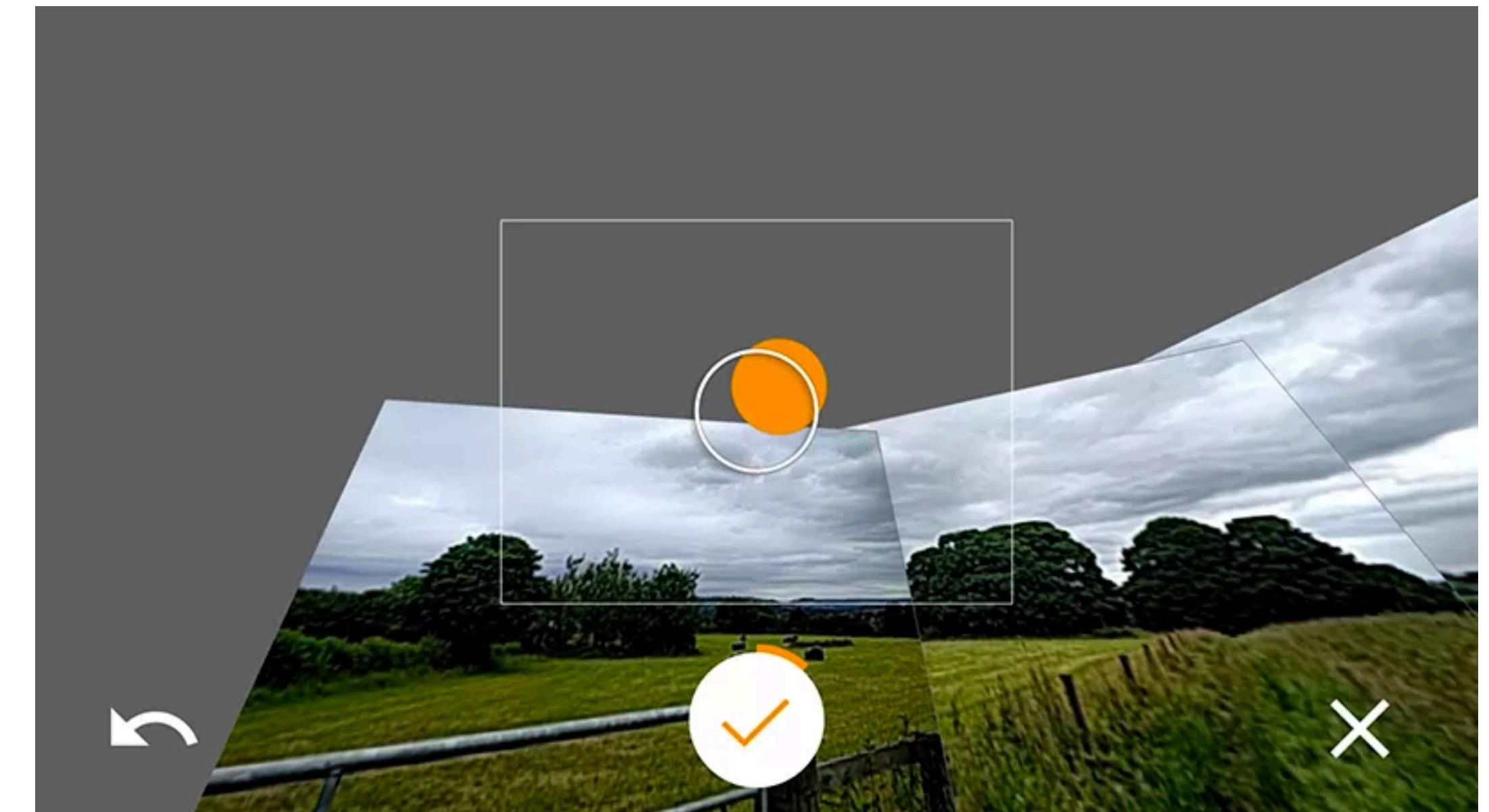
- Accelerometers
  - Axis-based motion sensing
  - Measures acceleration in a particular direction



<https://gizmodo.com/all-the-sensors-in-your-smartphone-and-how-they-work-1797121002>

# Sensors

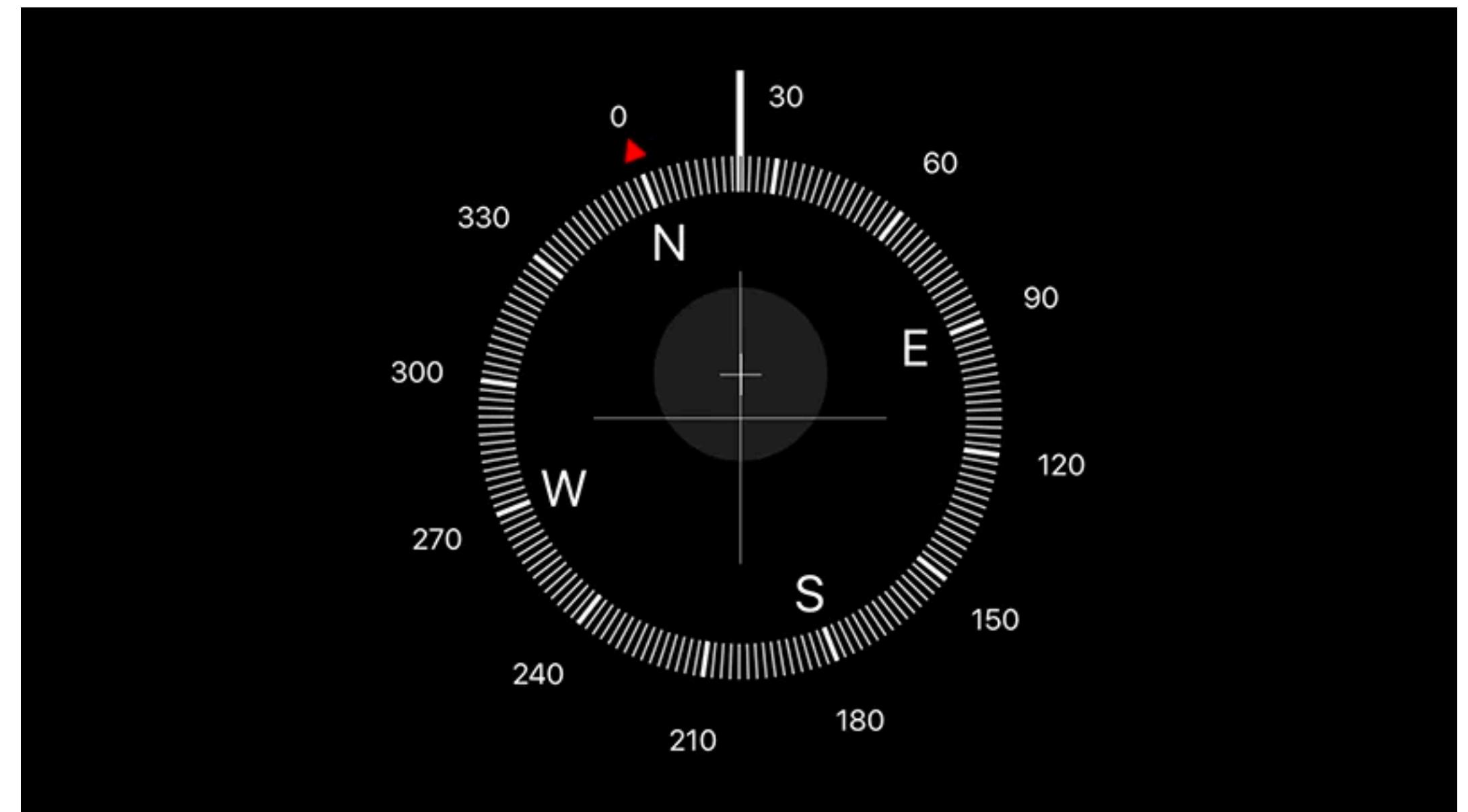
- Gyroscope
  - Measures device orientation
  - Can measure device rotation, where accelerometer cannot



<https://gizmodo.com/all-the-sensors-in-your-smartphone-and-how-they-work-1797121002>

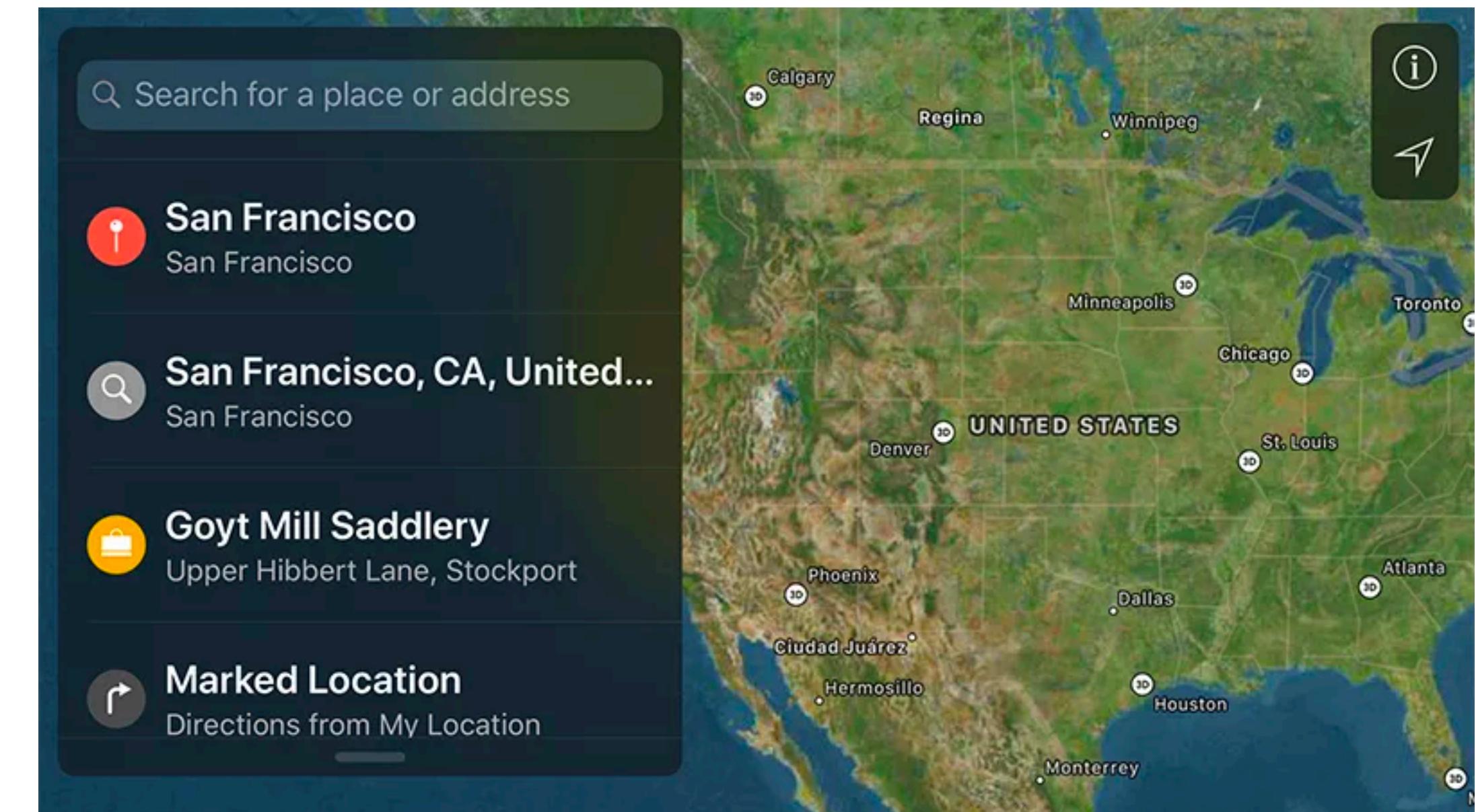
# Sensors

- Magnetometer
  - Identifies cardinal direction
  - Can be used together with Gyroscope to create a compass



# Sensors

- Global Positioning System (GPS)
  - Identify where on the planet you are
  - Navigation in Apple Maps, Google Maps, etc. combines all four sensors



# Sensors

- Proximity sensor: how close/far an object is
  - Switches off your screen when it's in your pocket/backpack/purse
- Ambient light: measures how bright a room is
  - Changes screen brightness to accommodate
- Near field communication (NFC): allows nearby objects to communicate
  - Powers Apple Pay, etc.

# Sensors

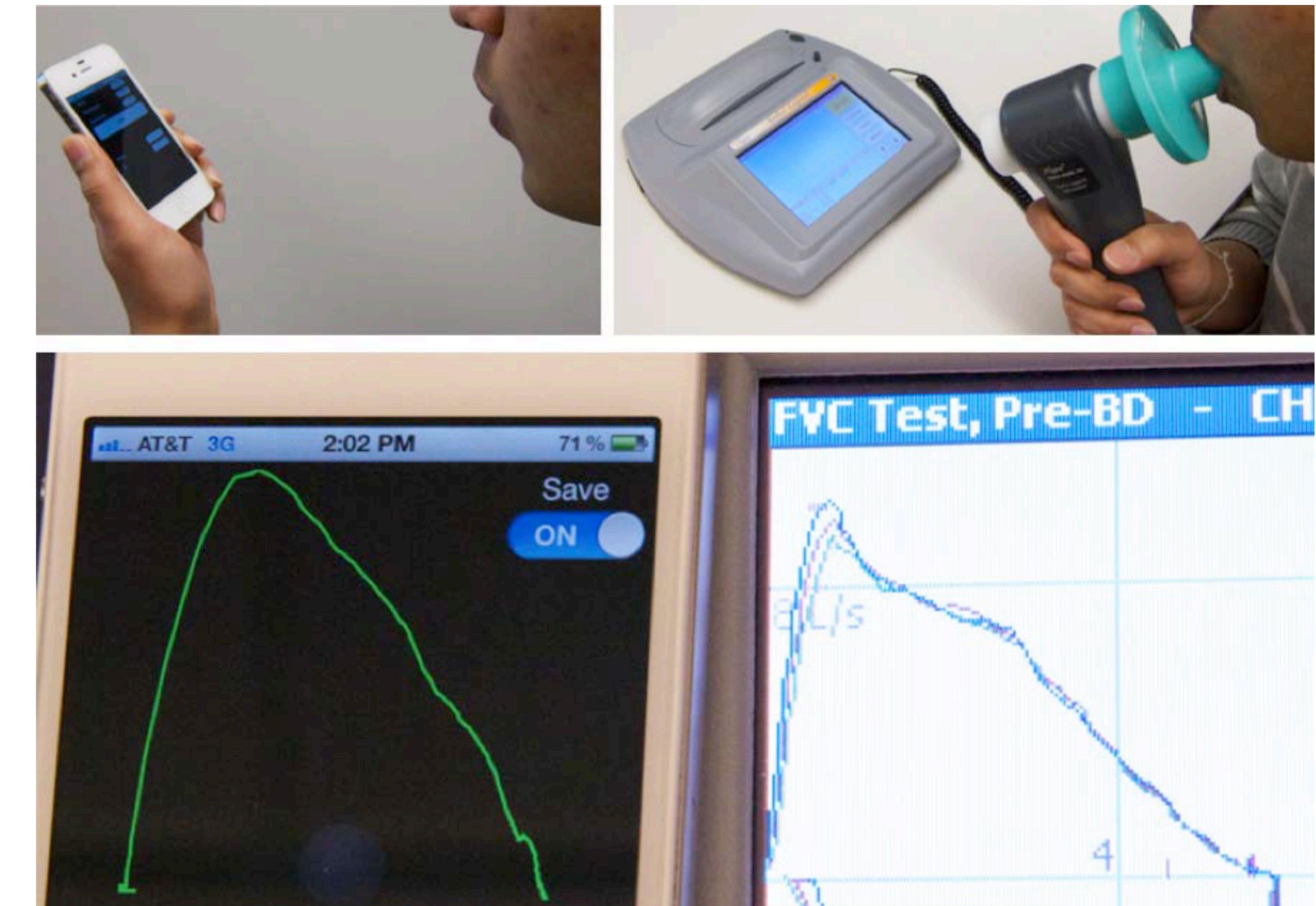
- Sensors can also be re-appropriated
  - Microphone: measure noise, such as for sleep quality
  - Camera: barcode or QR scanner
  - Accelerometer: pedometer
  - Touchscreen: pressure

# **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 Borriello, 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...

- 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

# **IN4MATX 133: User Interface Software**

**Lecture 15:**  
**Device Resources & Sensors**

Professor Daniel A. Epstein  
TA Eunkyung Jo  
TA Lucas de Melo Silva