SYST35300
Hybrid Mobile App Development

Presentation created for Sheridan College by A. Pak

Agenda

- Ionic Native APIs

Ionic Native APIs

The Ionic Native API is a collection of plugins and native integrations designed to bring native functionalities to Ionic apps. It supports two plugin sets: Cordova and Capacitor.

Cordova plugins, which have been around longer, provide a wide range of native functionality that works across iOS and Android. While Capacitor, created and maintained by the Ionic team, is the newer cross-platform native runtime with its own set of plugins, it's designed to modernize the process of integrating native functionality in Ionic apps.

In this course, we'll use the Capacitor runtime and its native plugins, as it is the recommended approach for building native functionality in modern Ionic apps.

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

Ionic Native APIs

Installing IDEs and Emulators:

For Android: https://ionicframework.com/docs/developing/android

- **Install Android Studio** https://developer.android.com/studio
- **Select Android SDK using SDK Manager**
- **Create Android Virtual Device using Device Manager**

For iOS: https://ionicframework.com/docs/developing/ios

- **Install Xcode (download from App Store)**
- **May need to install CocoaPods**
  https://guides.cocoapods.org/using/getting-started.html#installation
- **Code signing ...**
  https://developer.apple.com/support/code-signing/

Ionic Native APIs

Developing Ionic Native Applications typically involves the following steps:

1. Generating a new Ionic project.
2. Installing native plugins as needed.
3. Building and running the project using the Capacitor runtime.
4. Debugging the project to identify and fix issues.
5. Publishing the project for deployment.

Ionic Native APIs

1. Generating a New Ionic Project :

```
ionic start <project>
```

2. Installing Native Plugins:

a) Install the Capacitor plugin:

```
npm install @capacitor/XXXX
```

b. After installing the native plugins and wrappers, synchronize the Ionic project with the native environment (both Android and iOS) by copying web assets from the **www** folder to the appropriate platform folder:

```
ionic cap sync
```

Ionic Native APIs

3. Building and Running the Project using the Capacitor Runtime:

For Android: https://ionicframework.com/docs/developing/android

- **`ionic capacitor build android / ios`**
  - Builds the web assets (HTML, JavaScript, CSS) and prepares the native project for execution on an emulator or device.
  - Typically used for the first test run on an emulator/device.
- **`ionic capacitor copy android / ios`**
  - Copies built assets to the native project, updating it without rebuilding.
  - Typically used after modifying web assets without a full rebuild.
- **`ionic capacitor open android / ios`**
  - Opens the native project in Android Studio/Xcode.
  - Typically used for testing /debugging in the IDE.
- **`ionic capacitor run android / ios`**
  - Builds, deploys, and runs the app on an emulator/device directly.
  - Use it to quickly deploy and test the app on an emulator/device.

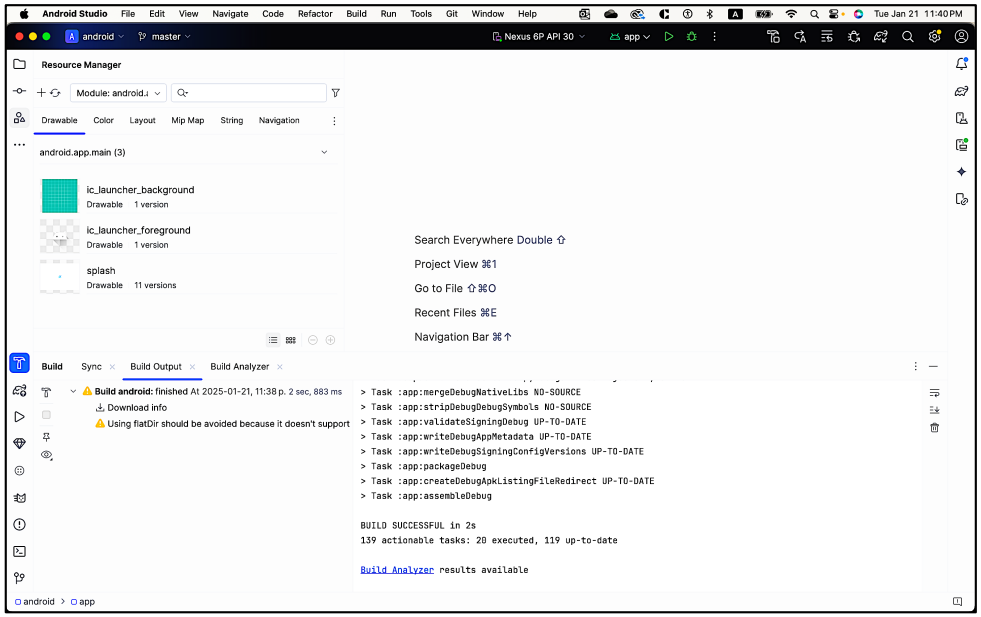
Ionic Native APIs

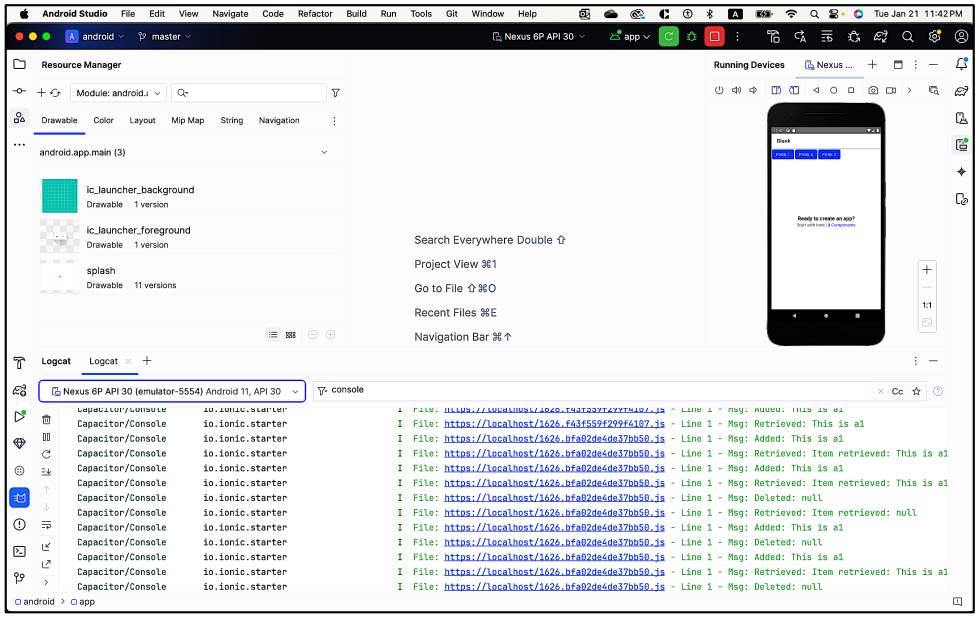
4. Debugging the project

For Android:

Android Studio logs under View → Tool Windows:

- You can select tools such as device manager, running devices, and logcat.





Ionic Native APIs

4. Debugging the project (cont.)

For Android: (cont.)

Chrome offers remote debugging for Ionic applications running in Android Studio:

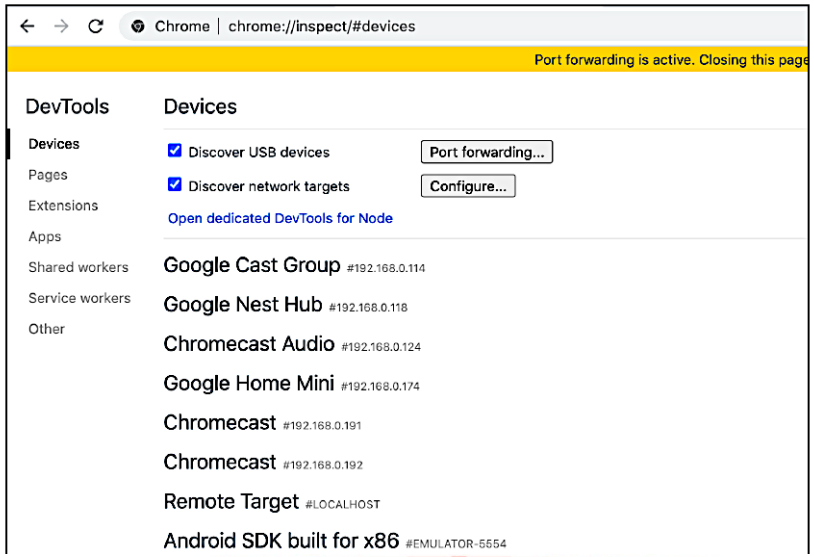
- Launch the Ionic app in Android Studio.
- Open Chrome and navigate to **chrome://inspect/#devices**.
- In the "WebView" section, find and select the app.
- Use Chrome's developer tools to inspect and debug the app.

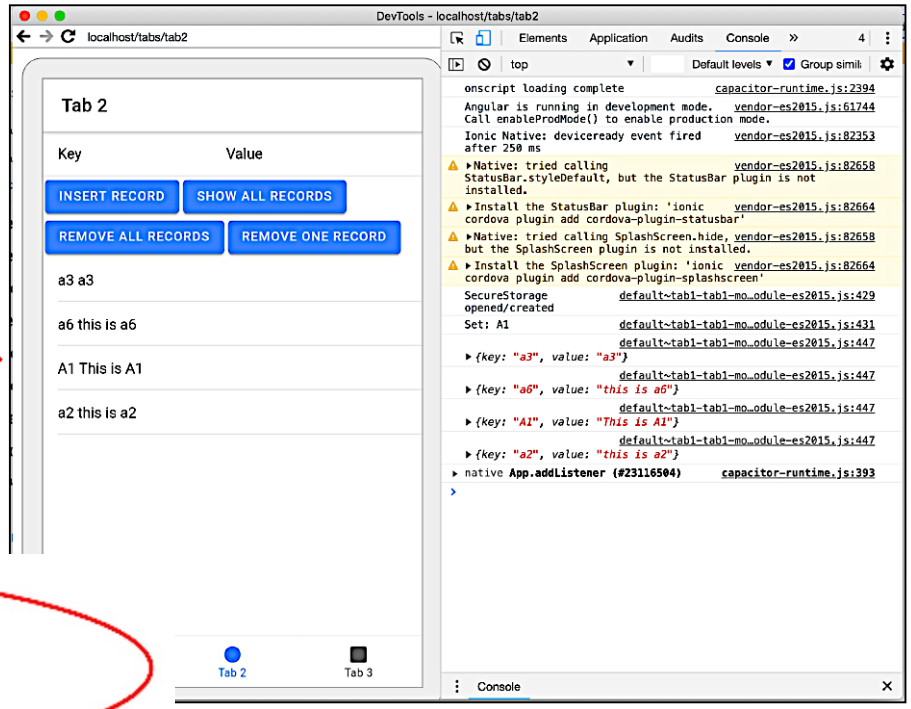
Ionic Native APIs

4. Debugging the project (cont.)

For Android: (cont.)

Chrome remote debugging capability for Ionic applications





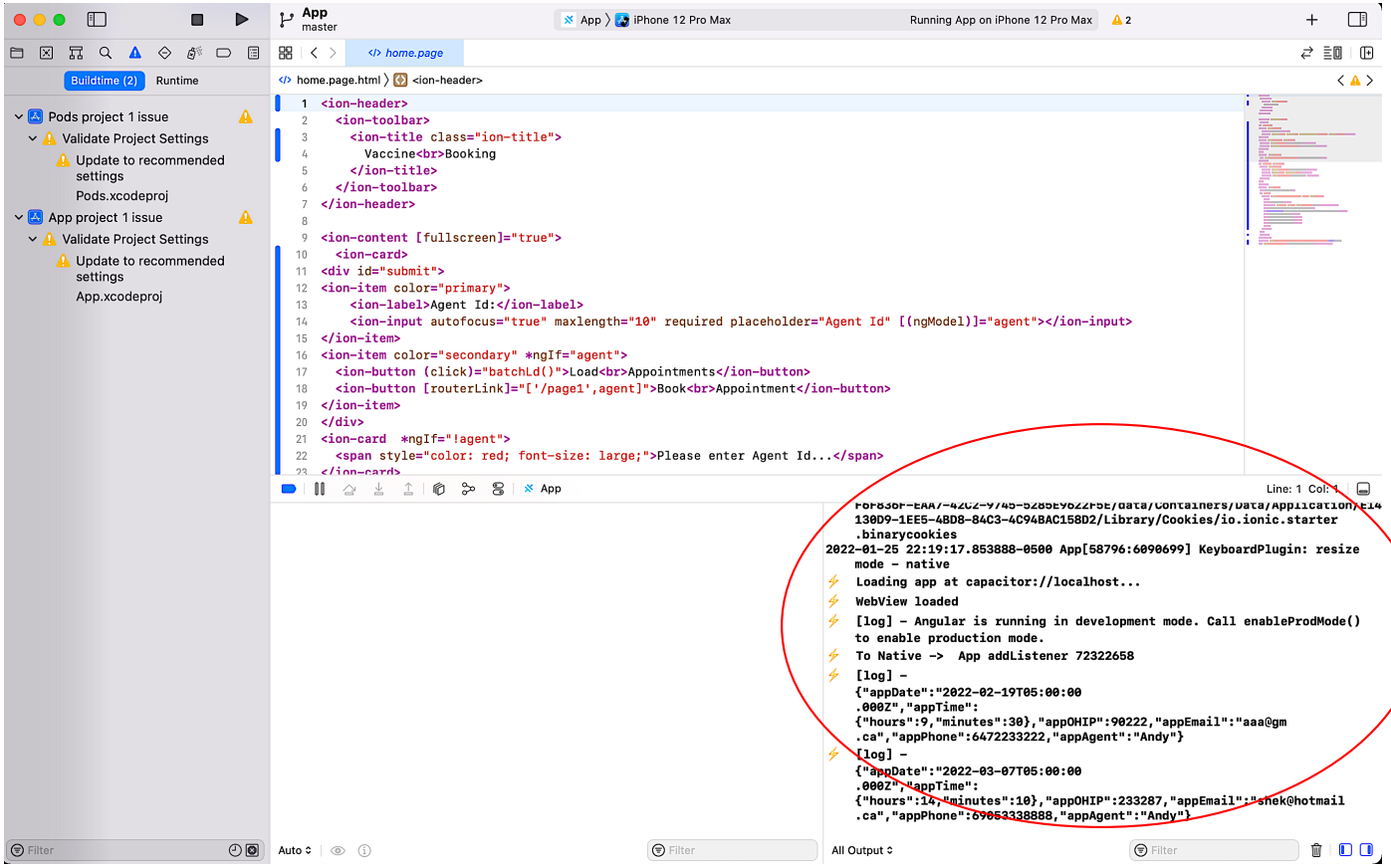
Select inspect

Ionic Native APIs

4. Debugging the project (cont.)

For iOS:

logs can be found in Xcode under view/Debug Area/Active Console



Ionic Native APIs

5. Publishing the project

Publish to App Store (iOS platform)

https://ionicframework.com/docs/deployment/app-store

Publish to Google Play Store (Android platform)

https://ionicframework.com/docs/deployment/play-store

Ionic Native APIs

5. Publishing the project (cont.)

Ionic offers the "**Appflow**" service, which helps developers in building, generating native apps, and deploying updates.

Ionic Appflow services:

- **App Publishing** - Publish directly to the Apple and Google App stores.
- **Live Updates** - Send live updates to users without waiting on app store approval.
- **Native Builds** - Compile native app binaries in the cloud.
- **Automations** - Fully automate your app delivery pipeline.

https://ionic.io/appflow

Ionic Native APIs

Create a standalone Ionic project with blank template

```
ionic start w5-inclass1 blank
```

Selected Capacitor native plugins:

- Action Sheet
- Camera
- Device
- Dialog
- Network
- Status Bar
- Toast

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

Ionic Native APIs

Action Sheet

The Action Sheet API provides access to native Action Sheets, which come up from the bottom of the screen and display actions a user can take.

https://ionicframework.com/docs/native/action-sheet

```
npm install @capacitor/action-sheet

ionic cap sync
```

In home.page.html

```
<ion-button (click)="action()">Open Action Sheet</ion-button>
```

Ionic Native APIs

Action Sheet (cont.)

In home.page.ts

```
import { ActionSheet, ActionSheetButtonStyle } from
        '@capacitor/action-sheet';
import { …, IonButton } from '@ionic/angular/standalone';

imports: […, IonicButton]

async action() {
   const options = [ { title: 'Upload' }, { title: 'Share' },
     {title: 'Remove', style: ActionSheetButtonStyle.Destructive}
   ];
   const result = await ActionSheet.showActions({
        title: 'Photo Options',
        message: 'Select an option to perform',
        options
   });
}
```

Ionic Native APIs

Camera

The Camera API provides the ability to take a photo with the camera or choose an existing one from the photo album.

https://ionicframework.com/docs/native/camera

```
npm install @capacitor/camera

ionic cap sync
```

In home.page.html

```
<ion-button (click)="camera()">Camera</ion-button>
<ion-card>
   <img [src]="imageUrl">
</ion-card>
```

Ionic Native APIs

Camera (cont.)

In home.page.ts

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

imageUrl: string = '';

async camera() {
    const image = await Camera.getPhoto({
        quality: 90,
        allowEditing: true,
        resultType: CameraResultType.Uri
        });
    this.imageUrl = image.webPath || '';
}
```

Ionic Native APIs

Device

The Device API exposes internal information about the device, such as the model and operating system version, along with user information such as unique ids.

https://ionicframework.com/docs/native/device

```
npm install @capacitor/device

ionic cap sync
```

Ionic Native APIs

Device (cont.)

In home.page.html

```
<ion-button (click)="device()">Device</ion-button><br>
<ion-card>
   <ion-card-header>
       <ion-card-subtitle>Device</ion-card-subtitle>
   </ion-card-header>
   <ion-card-content>
       <ul>
          @for (entry of entries; track entry) {
             <li>{{ entry[0] }}: {{ entry[1] }}</li>
          }
       </ul>
       <ul>
          @for (entry of entries2; track entry) {
             <li>{{ entry[0] }}: {{ entry[1] }}</li>
          }
       </ul>
   </ion-card-content>
</ion-card>
```

Ionic Native APIs

Device (cont.)

In home.page.ts

```
import { Device } from '@capacitor/device';
import { CommonModule } from '@angular/common';
import { …, IonCard, IonCardHeader, IonCardContent,
          IonCardSubtitle } from '@ionic/angular/standalone';

imports: […, IonCard, IonCardContent, IonCardHeader,
          IonCardSubtitle, CommonModule, ],

entries!:any; entries2!:any;

async device(){
    try { const info = await Device.getInfo();
          this.entries = Object.entries(info);
          const info2 = await Device.getBatteryInfo?.();
          this.entries2 = Object.entries(info2);
          }
    catch(err) { console.log(err) }
}
```

Ionic Native APIs

Dialog

The Dialog API provides methods for triggering native dialog windows for alerts, confirmations, and input prompts

https://ionicframework.com/docs/native/dialog

```
npm install @capacitor/dialog

ionic cap sync
```

In home.page.html

```
<ion-button (click)="dialog()">Dialog</ion-button>
```

Ionic Native APIs

Dialog (cont.)

In home.page.ts

```
import { Dialog } from '@capacitor/dialog';

async dialog() {
     await Dialog.alert({
         title: 'Stop', message: 'this is an error',
     });
     const { value: confirmed } = await Dialog.confirm({
         title: 'Confirm',
         message: `Are you sure?`,
     });
     const { value:name, cancelled } = await Dialog.prompt({
         title: 'Hello',
         message: `What's your name?`,
     });
     console.log('Confirmed:', confirmed);
     console.log('Name:', name, 'Cancelled:', cancelled);
};
```

Ionic Native APIs

Network

The Network API provides network and connectivity information.

https://ionicframework.com/docs/native/network

```
npm install @capacitor/network
```

```
ionic cap sync
```

Ionic Native APIs

Network (cont.)

In home.page.html

```
<ion-button (click)="network()">Network</ion-button><br>
<ion-card>
  <ion-card-header>
    <ion-card-subtitle>Network</ion-card-subtitle>
  </ion-card-header>
  <ion-card-content>
    <ul>
        @for (entry of netStatus; track entry) {
          <li>{{ entry[0] }}: {{ entry[1] }}</li>
        }
    </ul>
  </ion-card-content>
</ion-card>
```

Ionic Native APIs

Network (cont.)

In home.page.ts

```
import { Network } from '@capacitor/network';

netStatus!:any;
network() {
   Network.addListener('networkStatusChange', status => {
        console.log('Network status changed', status);
   });
   const logCurrentNetworkStatus = async () => {
        const info= await Network.getStatus()
        this.netStatus = Object.entries(info)
        console.log('Network status:', this.netStatus);
   };
   logCurrentNetworkStatus();
}
```

Ionic Native APIs

Status Bar

The StatusBar API Provides methods for configuring the style of the Status Bar, along with showing or hiding it.

https://ionicframework.com/docs/native/status-bar

```
npm install @capacitor/status-bar

ionic cap sync
```

In home.page.html

```
<ion-button (click)="statusHide()">Status Bar Hide</ion-button>

<ion-button (click)="statusShow('#ADD8E6')">Status Bar Show
</ion-button>
```

Ionic Native APIs

Status Bar (cont.)

In home.page.ts

```
import { StatusBar, Style } from '@capacitor/status-bar';
import { Platform } from '@ionic/angular';

constructor(private platform: Platform) {}

async ngOnInit() {
   this.statusShow('#ffff00');
}

async statusHide() { await StatusBar.hide(); }

async statusShow(color:string) {
   await this.platform.ready(); // Ensure platform is ready
   await StatusBar.setOverlaysWebView({ overlay: false });
   await StatusBar.setBackgroundColor({ color: color });
   await StatusBar.setStyle({style:Style.Light }); // Optional:
   await StatusBar.show();
}
```

Ionic Native APIs

Toast

The Toast API provides a notification pop up for displaying important information to a user. Just like real toast!

https://ionicframework.com/docs/native/toast

```
npm install @capacitor/toast

ionic cap sync
```

In home.page.html

```
<ion-button (click)="toast()">Toast</ion-button>
```

Ionic Native APIs

Toast (cont.)

In home.page.ts

```
import { Toast } from '@capacitor/toast';

async toast() {
   await Toast.show({
        text: 'Hello!',
   });
};
```