SYST35300
Hybrid Mobile App Development

Presentation created for Sheridan College by A. Pak

Agenda

- Ionic UI Components
- Ionic Colors

Ionic UI Components

Ionic offers a library featuring mobile UI components with high native styles, facilitating rapid development of mobile applications that match the desired look and feel.

You can explore a complete list of the UI components, including cards, lists, tabs, and more at this URL: https://ionicframework.com/docs/api.

Additionally, Ionic provides pre-designed icons customized for your applications, accessible at https://ionicons.com.

Ionic UI Components

A typical Ionic page

https://ionicframework.com/docs/api/header
https://ionicframework.com/docs/api/content
https://ionicframework.com/docs/api/footer

```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Ionic UI Components Page
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
    <ion-button [routerLink]="['/xxxxx']">UI Components</ion-button>
</ion-content>

<ion-footer>
  <ion-toolbar>
     <a href="http://www.sheridancollege.ca">Sheridan</a>
  </ion-toolbar>
</ion-footer>
```

Ionic UI Components

To use Ionic UI components in a standalone project, import the required components from **@ionic/angular/standalone** and add them to the imports array in the @Component decorator.
Alternatively, you can import IonicModule to include all Ionic components at once.

Create an Ionic standalone project

```
ionic start w4-inclass3 tabs
```

Use IonicModule

```
ionic serve
```

https://ionicframework.com/docs/api

Selected UI Components:

- ion-action-sheet
- ion-alert
- ion-button
- ion-card
- ion-checkbox
- ion-grid
- ion-img
- ion-input
- ion-item
- ion-list
- ion-radio
- ion-search
- ion-select

Ionic UI Components

Action Sheet

An Action Sheet is a dialog that displays a set of options. It is created by the Action Sheet Controller from an array of buttons, with each button including properties for its text, icon, handler and role.

(https://ionicframework.com/docs/api/action-sheet)

Each button in the action sheet has the following properties:

- text – text of the action to be displayed
- role – 'cancel' | 'destructive' (optional)
- handler – callback after action is selected (optional)
- icon – icon to be displayed

Ionic UI Components

```
<ion-button color="primary" size="small"
            (click)="presentActionSheet()">Open Action Sheet
</ion-button>

<ion-button color="secondary" size="default"
            (click)="presentActionSheet()">Open Action Sheet
</ion-button>

<ion-button color="tertiary" size="large"
            (click)="presentActionSheet()">Open Action Sheet
</ion-button>
```

Ionic UI Components

```
import { ActionSheetController } from '@ionic/angular';

constructor(public actionSheetController: ActionSheetController) {}

async presentActionSheet() {
    const actionSheet = await this.actionSheetController.create({
      header: 'Menu',
      buttons: [ { text: 'Delete', icon: 'trash',
                   role: 'destructive',
                   handler: () => { console.log('Delete'); } },
                 { text: 'Share', icon: 'share',
                   handler: () => { console.log('Share'); } },
                 { text: 'Cancel', icon: 'close', role: 'cancel',
                   handler: () => { console.log('Cancel'); } }
               ]
        });
    await actionSheet.present();
}
```

Ionic UI Components

Alert

An Alert is a dialog that presents users with information or collects information from the user using inputs. An alert appears on top of the app's content, and must be manually dismissed by the user before they can resume interaction with the app. It can also optionally have a header, subHeader and message.

(https://ionicframework.com/docs/api/alert)

Ionic UI Components

```
<ion-button color="primary" size="small"
     (click)="alertAction()">Alert
</ion-button>

import { AlertController } from '@ionic/angular';

constructor(public alertControl: AlertController) {}

async alertAction() {
    const alert = await this.alertControl.create({
      header: 'Alert',
      subHeader: '',
      message: 'This is an Alert Message',
      buttons: [
        { text: 'Cancel',
          handler: () => { console.log('Confirm cancel!'); } },
        { text: 'OK',
          handler: () => { console.log('Confirm OK!'); }
        }]
      });
    await alert.present();
    }
```

Ionic UI Components

Buttons

Buttons provide a clickable element, which can be used in forms, or anywhere that needs simple, standard button functionality. They may display text, icons, or both. Buttons can be styled with several attributes to look a specific way.

(https://ionicframework.com/docs/api/button)

```
<ion-card>
    <ion-button size="small">Small</ion-button>
    <ion-button>Default</ion-button>
    <ion-button size="large">Large</ion-button>
    <ion-button expand="block">Block</ion-button>
    <ion-button expand="full">Full</ion-button>
    <ion-button [disabled]="true">Disabled</ion-button>
</ion-card>
```

Ionic UI Components

Back Button / Menu Button

The back button navigates back in the app's history upon click. It is smart enough to know what to render based on the mode and when to show based on the navigation stack.

The menu Button component contains an icon and automatically adds functionality to open a menu when clicked.

https://ionicframework.com/docs/api/menu-button
https://ionicframework.com/docs/api/back-button

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="/">
      </ion-back-button>
    </ion-buttons>
    <ion-menu-button slot="start" auto-hide="false">
    </ion-menu-button>
  </ion-toolbar>
</ion-header>
```

Ionic UI Components

Cards

Cards are a standard piece of UI that serves as an entry point to more detailed information. A card can be a single component, but is often made up of some header, title, subtitle, and content. ion-card is broken up into several sub-components to reflect this.

(https://ionicframework.com/docs/api/card)

Ionic UI Components

```
<ion-card color="primary">
    <ion-card-header>
        <ion-card-subtitle>Sub title</ion-card-subtitle>
        <ion-card-title>Card Title</ion-card-title>
    </ion-card-header>
    <ion-card-content>
      <ion-button color="secondary" size="small"
        (click)="presentActionSheet()">Open Action Sheet
      </ion-button>
    </ion-card-content>
</ion-card>
```

Ionic default colors: **primary**, **secondary**, **tertiary**, **success**, **warning**, **danger**, **light**, **medium**, dark.

https://ionicframework.com/docs/theming/basics

Ionic UI Components

Lists

Lists are made up of multiple rows of items which can contain text, buttons, toggles, icons, thumbnails, and much more. Lists generally contain items with similar data content, such as images and text. Lists support several interactions including swiping items to reveal options, dragging to reorder items within the list, and deleting items.

(https://ionicframework.com/docs/api/list)

Ionic UI Components

```
<ion-list>
    <ion-list-header>Top Programming Languages</ion-list-header>
    <ion-item><ion-label>JavaScript</ion-label></ion-item>
    <ion-item><ion-label>Python</ion-label></ion-item>
    <ion-item><ion-label>Java</ion-label></ion-item>
    <ion-item><ion-label>Ruby</ion-label></ion-item>
    <ion-item><ion-label>TypeScript</ion-label></ion-item>
</ion-list>
```

Use the `@for` directive:

```
<ion-list>
    <ion-list-header>Top Programming Languages</ion-list-header>
    @for (item of pgArray; let i=$index; track item) {
        <ion-item><ion-label>{{i}} {{item}}</ion-label></ion-item>
    }
</ion-list>

pgArray = ['JavaScript', 'Python', 'Java', 'Go', 'Elixir',
           'Ruby', 'Kotlin'];
```

Ionic UI Components

Grid

Flexbox for building customer layouts. It is composed of three units — a grid, row(s) and column(s). Columns will expand to fill the row, and will resize to fit additional columns. It is based on a 12 column layout with different breakpoints based on the screen size.

(https://ionicframework.com/docs/api/grid)

```
<ion-grid>
  <ion-row>
    <ion-col>Col 1</ion-col>
    <ion-col>Col 2</ion-col>
  </ion-row>
  <ion-row>
    <ion-col>Col 1</ion-col>
    <ion-col>Col 2</ion-col>
    <ion-col>Col 3</ion-col>
  </ion-row>
</ion-grid>
```

```
ion-col {
   border: solid 1px blue;
   padding: 10px;
}
```

Ionic UI Components

Image

Img is a tag that will lazily load an image whenever the tag is in the viewport. This is extremely useful when generating a large list as images are only loaded when they're visible. The component uses Intersection Observer internally, which is supported in most modern browser, but falls back to a setTimeout when it is not supported.

(https://ionicframework.com/docs/api/img)

Define the array in the .ts file

```
<ion-list>
   @for (x of items; track x) {
      <ion-item>
        <ion-thumbnail slot="start">
           <ion-img src="x.src" alt="xxxx"></ion-img>
        </ion-thumbnail>
        <ion-label>{{x.text}}</ion-label>
      </ion-item>
   }
</ion-list>
```

Ionic UI Components

Angular Template-Driven Form

Utilizes an input component that wraps around the HTML input element, offering custom styling and added features. While it accepts many of the same properties as the HTML input, it excels on desktop devices and integrates with mobile device keyboards.

Note: **FormsModule** is needed in the component's .ts file.

(https://ionicframework.com/docs/api/input)

Ionic UI Components

Angular Template-Driven Form (cont.)

```
<form #myForm="ngForm" (ngSubmit)=confirm(myForm)>
    <ion-item>
      <ion-label position="fixed">Student ID</ion-label>
      <ion-input required type="text" ngModel name="id">
      </ion-input>
    </ion-item>

    <ion-item>
      <ion-label position="floating">First Name</ion-label>
      <ion-input ngModel name="first"></ion-input>
    </ion-item>

    <ion-item>
      <ion-label position="stacked">Last Name</ion-label>
      <ion-input ngModel name="last"></ion-input>
    </ion-item>
```

Ionic UI Components

Angular Template-Driven Form (cont.)

```
<!-- Radio buttons -->
<ion-radio-group lines="full" ngModel name="radio">
  <ion-list-header>
    <ion-label>Gender</ion-label>
  </ion-list-header>
  <ion-item>
    <ion-label>Male</ion-label>
    <ion-radio slot="start" value="male" checked></ion-radio>
  </ion-item>
  <ion-item>
    <ion-label>Female</ion-label>
    <ion-radio slot="start" value="female"></ion-radio>
  </ion-item>
</ion-radio-group>
```

Ionic UI Components

Angular Template-Driven Form (cont.)

```
    <!-- Checkboxes -->
    <ion-list lines="full">
      <ion-list-header>
        <ion-label>Subjects</ion-label>
      </ion-list-header>
      @for (checkbox of check; let i=$index; track checkbox) {
        <ion-item>
          <ion-label>{{checkbox.label}}</ion-label>
          <ion-checkbox slot="start" ngModel={{checkbox.checked}}
                        name={{checkbox.label}}
                       (ionChange)="onCheckboxChange($event, i)">
          </ion-checkbox>
        </ion-item>
      }
    </ion-list>
    <ion-button type="submit" class="ion-no-margin">
        Create Student Record
    </ion-button>
</form>
```

Ionic UI Components

Angular Template-Driven Form (cont.)

```
import { AlertController } from '@ionic/angular';
import { FormsModule, NgForm } from '@angular/forms';

imports: [..., FormsModule],

constructor(private alertControl: AlertController) { }
id: any;  lName: any;  fName:any; radio:any;
check = [
    {label: "JavaScript", checked: true},
    {label: "Java", checked: false},
    {label: "Python", checked: false},
    {label: "PHP", checked: false},
    {label: "HTML", checked: true}
];
```

Ionic UI Components

Angular Template-Driven Form (cont.)

```
async confirm(form:NgForm) {
   const alert = await this.alertControl.create({
      header: 'New Student Account',
      subHeader: '',
      message: 'Please confirm account creation.',
      buttons: [
        { text: 'Cancel',
          handler: () => { console.log('Confirm cancel!'); } },
        { text: 'OK',
          handler: () => {
                console.log('Confirm OK!');
                this.goCheck(form);
           } }
        ]
   });
   await alert.present();
}
```

Ionic UI Components

Angular Template-Driven Form (cont.)

```
onCheckboxChange(e: any, i:number) {
  this.check[i].checked=e.target.checked;
}

goCheck(form:NgForm) {
   form.value.checkbox=[];
   this.check.forEach( x=> {
      if (x.checked) {
        form.value.checkbox.push(x.label)
      }
   })
   console.log(form.value)
}
```

Ionic UI Components

Angular Reactive Form

Provides direct and explicit access to the form's object model, making it more robust than the template-driven forms. It accepts most of the same properties as the HTML input and perform well on desktop devices and mobile devices.

In this approach, the form model is defined within the component class. The '[formControl]' directive connects the 'FormControl' instances to specific form elements in the view.

Additionally, the 'FormGroup' tracks the value and validity state of a group of 'FormControl' instances.

Validations are directly added to the form control model rather than through attributes in the template.

Note: ReactiveFormsModule is needed in the component's .ts file.

https://angular.io/guide/forms-overview https://angular.io/guide/form-validation

Ionic UI Components

Angular Reactive Form

```
<form [formGroup]="ionicForm" (ngSubmit)="confirm()">
    <ion-item>
      <ion-label position="fixed">Student ID</ion-label>
      <ion-input formControlName="id"  type="text">
      </ion-input>
    </ion-item>

    <ion-item>
      <ion-label position="floating">First Name</ion-label>
      <ion-input formControlName="fName" type="text"></ion-input>
    </ion-item>

    <ion-item>
      <ion-label position="stacked">Last Name</ion-label>
      <ion-input formControlName="lName" name="last"></ion-input>
    </ion-item>
```

Ionic UI Components

Angular Reactive Form (cont.)

```
<!-- Radio buttons -->
<ion-radio-group lines="full" formControlName="radio" >
  <ion-list-header>
    <ion-label>Gender</ion-label>
  </ion-list-header>
  <ion-item>
    <ion-label>Male</ion-label>
    <ion-radio slot="start" value="male" checked></ion-radio>
  </ion-item>
  <ion-item>
    <ion-label>Female</ion-label>
    <ion-radio slot="start" value="female"></ion-radio>
  </ion-item>
</ion-radio-group>
```

Ionic UI Components

Angular Reactive Form (cont.)

```
    <!-- Checkboxes -->
    <ion-list lines="full">
      <ion-list-header>
        <ion-label>Subjects</ion-label>
      </ion-list-header>
      @for (checkbox of check; let i=$index; track checkbox) {
        <ion-item>
          <ion-label>{{checkbox.label}}</ion-label>
          <ion-checkbox slot="start" value="{{checkbox.value}}"
                  checked="{{checkbox.checked}}"
                  (ionChange)="onCheckboxChange($event, i)">
          </ion-checkbox>
        </ion-item>
      }
    </ion-list>
    <ion-row>
      <ion-col><ion-button type="submit" color="danger"
         expand="block">Submit</ion-button></ion-col>
    </ion-row>
</form>
```

Ionic UI Components

Angular Reactive Form (cont.)

```
import { AlertController } from '@ionic/angular';
import { FormGroup, FormBuilder, Validators, ReactiveFormsModule }
from "@angular/forms";

constructor(private alertControl: AlertController,
            private formBuilder: FormBuilder) { }
id: any;  lName: any;  fName:any; radio:any; check!:any[];
ionicForm!: FormGroup;

ngOnInit() {
  this.check = [
    {label: "JavaScript", checked: false, value: "javascript"},
    {label: "Java", checked: false, value: 'java'},
    {label: "Python", checked: false, value: 'python'},
    {label: "PHP", checked: true, value: 'php'},
    {label: "HTML", checked: false, value: 'html'}
    ];
...
```

Ionic UI Components

Angular Reactive Form (cont.)

```
  this.ionicForm = this.formBuilder.group({
    id: [123, [Validators.minLength(2),
               Validators.pattern(/^[0-9]+$/),
               Validators.minLength(2)]],
    fName: ['Andy', [Validators.required,Validators.minLength(2)]],
    lName: ['Pak', [Validators.required, Validators.minLength(2)]],
    radio: ['', [Validators.required]],
    checkbox: this.formBuilder.array([])
  })
}  //ngOnInit

onCheckboxChange(e: any, i:number) {
    this.check[i].checked=e.target.checked;
}
```

Ionic UI Components

Angular Reactive Form (cont.)

```
async confirm() {
   if (!this.ionicForm.valid)
      console.log('Please provide all the required values!')
   else {
     const alert = await this.alertControl.create({
      header: 'New Student Account',
      subHeader: '',
      message: 'Please confirm account creation.',
      buttons: [
        { text: 'Cancel',
          handler: () => { console.log('Confirm cancel!'); } },
        { text: 'OK',
          handler: () => {
            console.log('Confirm OK!');
            this.goCheck(); } }
        ]
   });
   await alert.present(); }
}
```

Ionic UI Components

Angular Reactive Form (cont.)

```
goCheck() {
   this.ionicForm.value.checkbox=[];
   this.check.forEach( x=> {
      if (x.checked) {
        this.ionicForm.value.checkbox.push(x.value)
      }
   })
   console.log(this.ionicForm.value)
}
```

Ionic UI Components

Searchbar

Searchbars represent a text field that can be used to search through a collection. They can be displayed inside of a toolbar or the main content.

(https://ionicframework.com/docs/api/searchbar)

```
<ion-searchbar (ionChange)="search($event)"></ion-searchbar>

<ion-list>
   @for (item of items; let i=$index; track item) {
      <ion-item>
        <ion-label (click)="onClick(item)">{{item}}</ion-label>
      </ion-item>
   }
</ion-list>
```

Ionic UI Components

Searchbar (cont.)

```
items = [ 'Audi', 'BMW', 'Honda' ];
allItems = this.items;

search(ev: any) {
  const val = ev.target.value;

  if (val && val.trim() !== '') {
     this.items = this.allItems.filter((item) => {
       return (item.toLowerCase().indexOf(val.toLowerCase()) > -1);
       });
  } else { this.items = this.allItems; }
}

onClick(item:any) {
  console.log(item + ' selected');
}
```

Ionic UI Components

Select

Selects are form controls to select an option, or options, from a set of options, similar to a native <select> element. When a user taps the select, a dialog appears with all of the options in a large, easy to select list. (https://ionicframework.com/docs/api/select)

Note: **FormsModule** is needed in the component's .ts file.

```
<ion-item>
  <ion-label>Breakfast</ion-label>
  <ion-select multiple ok-text="Confirm" cancel-text="Cancel"
              (ionChange)="upDate($event)" [(ngModel)]="item">
    @for (item of breakfast; let i=$index; track item) {
       <ion-select-option value={{item.value}}>{{item.desc}}
       </ion-select-option>
    }
  </ion-select>
</ion-item>
```

Ionic UI Components

Select (cont.)

```
import { …, FormsModule } from '@angular/forms';

imports: […, FormsModule],


item: any;

breakfast = [ {value: 'egg', desc: 'Egg'},
              {value: 'toast', desc: 'Toast'},
              {value: 'bacon', desc: 'Bacon'},
              {value: 'sausage', desc: 'Sausages'},
              {value: 'pancake', desc: 'Pan Cake'} ] ;

upDate(sel: any) {
    console.log(sel.target.value);
}
```

Ionic Colors

Ionic offers nine default colors – **primary**, **secondary**, **tertiary**, **success**, **warning**, **danger**, **light**, **medium**, and dark – that you can use in your project. These colors are made up of different variables like shade, tint, and contrast to enhance your app's design.

You can adjust these default colors or create your own custom ones in the "src/theme/variables.scss" file.

To add new colors:

- Add new CSS variables in the src/theme/variables.scss file in the :root selector
  - :root { --ion-color-favorite: purple; }
- Append new class to the src/global.scss file for the new color
  - .ion-color-favorite { --ion-color-base: var(--ion-color-favorite); }

https://ionicframework.com/docs/theming/colors