IN4MATX 133: User Interface Software

Lecture:
Ionic Components

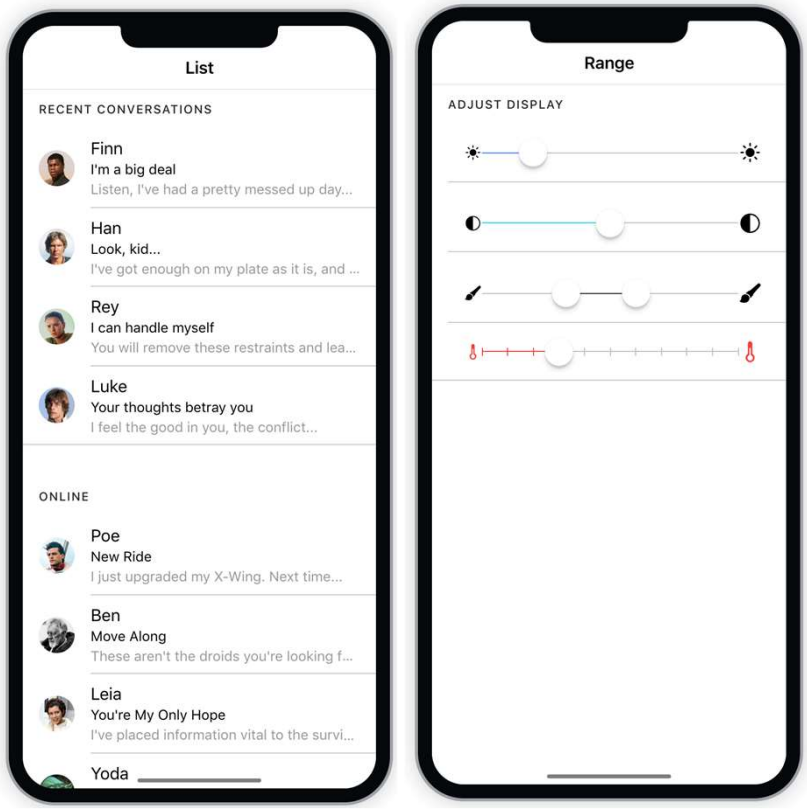
Goals for this lecture

By the end of this lecture, you should be able to…

- Use Ionic Components to make a mobile-friendly app
  - Display structured content with items and lists
  - Style content with colors, icons, and badges
  - Receive user input with inputs and modals
- Use routing to move between pages of your Ionic app

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
- We'll use Ionic 4 in this class



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

Ionic component documentation

- Each component has a lot of potential attributes and properties
- The documentation enumerates many of the options
- This lecture is an **overview**.
    - There are more components than we can reasonably discuss
    - Each component has more options than we can reasonably discuss
- The best way to learn them is to try them out

https://ionicframework.com/docs/api/

Types of Ionic components

- Structural
- Items
- Icons
- Inputs
- Lists
- Modals

Types of Ionic components

- **Structural**
- Items
- Icons
- Inputs
- Lists
- Modals

Structural

- Three structural components:
  - `<ion-content>`: holds the page's main content
  - `<ion-header>`: top bar for title content
  - `<ion-footer>`: bottom bar menu content
- Headers and footers can contain `<ion-toolbar>` with text & buttons
- A few other components can replace header and footer
  - `<ion-tabs>` for a footer with tabs to different pages

Structural

```
<ion-header>
  <ion-toolbar>
    <ion-title>My Navigation Bar</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content padding>
  Content here...
</ion-content>

<ion-footer>
  <ion-toolbar>
    <ion-title>Footer</ion-title>
  </ion-toolbar>
</ion-footer>
```

Types of Ionic components

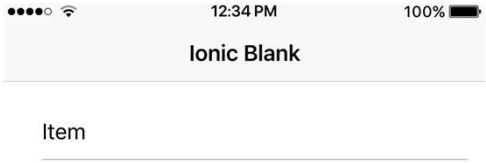
- Structural
- **Items**
- Icons
- Inputs
- Lists
- Modals

Items

- `<ion-item>` is the most basic component
- It's essentially an HTML `<div>`
  - Can hold text, images, and other things
  - Has a css "block" style, so it shows up as a row
- Lots of other components need to be inside of `<ion-item>`
  - For example, `<ion-label>` to put text inside of an `<ion-item>`

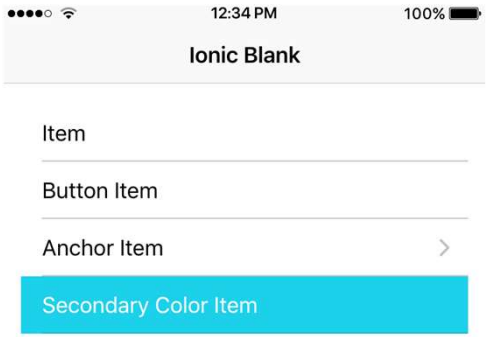
Items

```
 <!-- Default Item -->
<ion-item>
  <ion-label>
     Item
  </ion-label>
</ion-item>
```



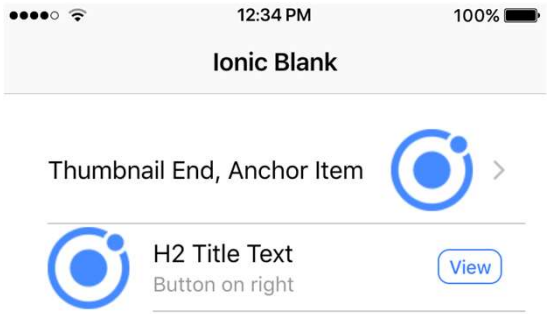
Items

```
 <!-- Default Item -->
<ion-item>
  <ion-label>
    Item
  </ion-label>
</ion-item>

<!-- Item as a Button -->
<ion-item (click)="buttonClick()">
  <ion-label>
    Button Item
  </ion-label>
</ion-item>

<!-- Item as an Anchor -->
<ion-item href="https://www.ionicframework.com">
  <ion-label>
    Anchor Item
  </ion-label>
</ion-item>

<ion-item color="secondary">
  <ion-label>
    Secondary Color Item
  </ion-label>
</ion-item>
```



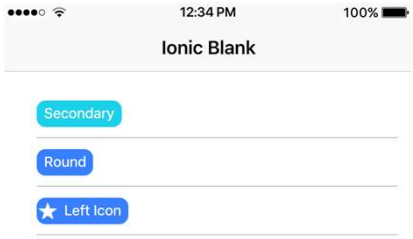
Items

```
<ion-item href="#">
  <ion-label>
    Thumbnail End, Anchor Item
  </ion-label>
  <ion-thumbnail slot="end">
    <img src="assets/icon/favicon.png">
  </ion-thumbnail>
</ion-item>

<ion-item>
  <ion-thumbnail slot="start">
    <img src="assets/icon/favicon.png">
  </ion-thumbnail>
  <ion-label>
    <h2>H2 Title Text</h2>
    <p>Button on right</p>
  </ion-label>
  <ion-button fill="outline" slot="end">View</ion-button>
</ion-item>
```



Items

```
<ion-item>
  <ion-button color="secondary">Secondary</ion-button>
</ion-item>

<ion-item>
  <ion-button shape="round">Round</ion-button>
</ion-item>

<ion-item>
  <ion-button>
    <ion-icon slot="start" name="star"></ion-icon>
    Left Icon
  </ion-button>
</ion-item>
```



Types of Ionic components

- Structural
- Items
- **Icons**
- Inputs
- Lists
- Modals
- Menus

Icons

- Can add labels or fun flavor

```
<ion-item>
  <ion-badge color="primary">11</ion-badge>
</ion-item>
<ion-item>
  <ion-badge color="secondary">22</ion-badge>
</ion-item>
<ion-item>
  <ion-icon name="heart"></ion-icon>
</ion-item>
<ion-item>
  <ion-icon name="moon"></ion-icon>
</ion-item>
<ion-item>
  <ion-badge color="secondary">
    <ion-icon name="moon"></ion-icon>
  </ion-badge>
</ion-item>
```

https://ionicons.com/

Types of Ionic components

- Structural
- Items
- Icons
- **Inputs**
- Lists
- Modals

Inputs

- Ionic provides a lot of common input fields
  - DateTime
  - Checkbox
  - Button
  - Text input
  - …
- For the most part, they should always be in an ion-item
- Bound just as in Angular, with two-way binding on `[(ngModel)]`

Inputs

```
<ion-item>
  <ion-label>Date</ion-label>
  <ion-datetime display-format="MM/DD/YYYY">
  </ion-datetime>
</ion-item>
```

Inputs

```
<ion-item>
  <ion-input required type="text"
  placeholder="First Name"></ion-input>
</ion-item>
```

Inputs

```
<ion-item>
  <ion-label>Pepperoni</ion-label>
  <ion-checkbox [(ngModel)]="pepperoni"></ion-checkbox>
</ion-item>

<ion-item>
  <ion-label>Sausage</ion-label>
  <ion-checkbox [(ngModel)]="sausage"
disabled="true"></ion-checkbox>
</ion-item>

<ion-item>
  <ion-label>Mushrooms</ion-label>
  <ion-checkbox [(ngModel)]="mushrooms"></ion-checkbox>
</ion-item>
```

Types of Ionic components

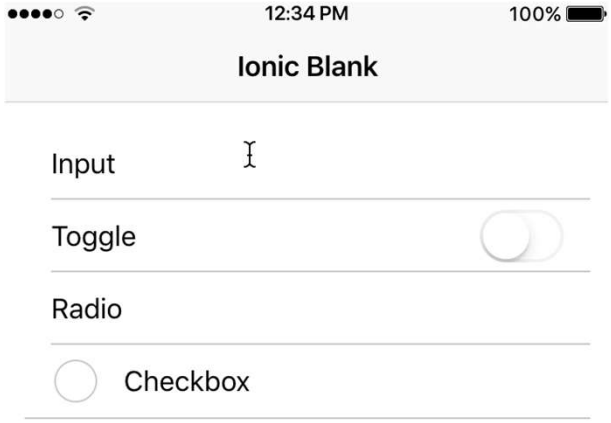
- Structural
- Items
- Icons
- Inputs
- **Lists**
- Modals

Lists

- Display rows of information
- Can provide some structure to items
- Styling lists, rather than items individually, can come in handy

Lists

```
<ion-list>
  <ion-item>
    <ion-label>Input</ion-label>
    <ion-input></ion-input>
  </ion-item>
  <ion-item>
    <ion-label>Toggle</ion-label>
    <ion-toggle slot="end"></ion-toggle>
  </ion-item>
  <ion-item>
    <ion-label>Radio</ion-label>
    <ion-radio slot="end"></ion-radio>
  </ion-item>
  <ion-item>
    <ion-label>Checkbox</ion-label>
    <ion-checkbox slot="start"></ion-checkbox>
  </ion-item>
</ion-list>
```

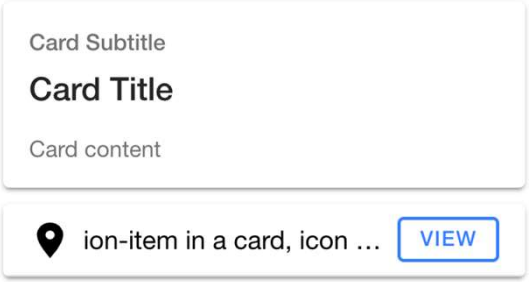


Lists

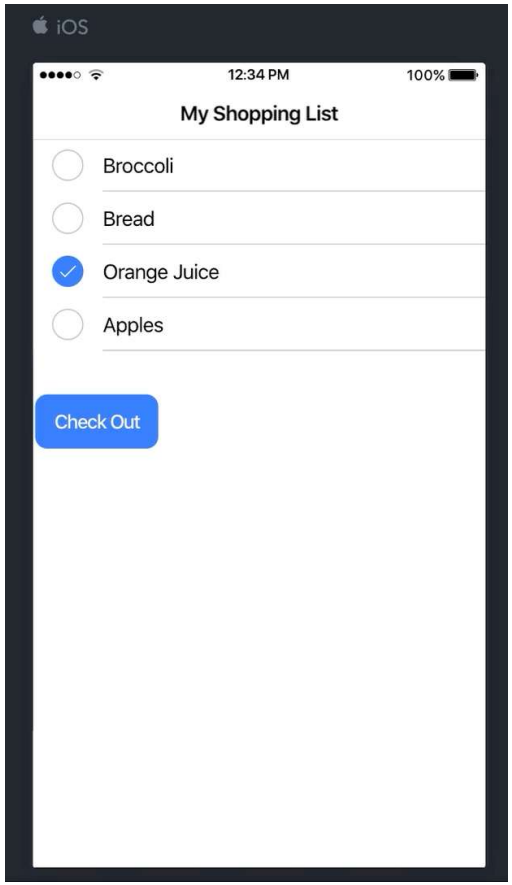
- Lists can contain tags other than `<ion-item>`
- For example, `<ion-card>` provides a “card” layout for presenting information

Lists

```
<ion-list>
  <ion-card>
    <ion-card-header>
      <ion-card-subtitle>Card Subtitle</ion-card-subtitle>
      <ion-card-title>Card Title</ion-card-title>
    </ion-card-header>

    <ion-card-content>
      Card content
    </ion-card-content>
  </ion-card>

  <ion-card>
    <ion-item>
      <ion-icon name="pin" slot="start"></ion-icon>
      <ion-label>ion-item in a card, icon left, button right</ion-label>
      <ion-button fill="outline" slot="end">View</ion-button>
    </ion-item>
  </ion-card>
</ion-list>
```



Lists, Items, and Inputs



Types of Ionic components

- Structural
- Items
- Icons
- Inputs
- Lists
- **Modals**

Modals

- Intended for quick entry or alerts
- Appear over the app's main content
- Two different styles
  - Modal dialogs
  - Modal pages
- Usually triggered in model or controller (`.ts`) rather than view (`.html`)

Modal dialogs

```
<!--HTML-->
<ion-button expand="full" color="primary" (click)="presentToast()">Send Toast</ion-button>

/*TypeScript*/
import { ToastController } from '@ionic/angular';

export class HomePage {
 /*Inject ToastController*/
 constructor(public toastController: ToastController) {}

 presentToast() {
     this.toastController.create({
       message: 'Hello, world!',
       duration: 2000
     }).then((toast) => {
      toast.present();
     });
   }
}
```

Modal dialogs

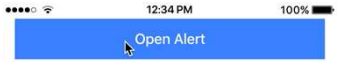
Async/await syntax (same functionality)

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

export class HomePage {
   constructor(public toastController: ToastController)
{}

   presentToast() {
       this.toastController.create({
         message: 'Hello, world!',
         duration: 2000
       }).then((toast) => {
        toast.present();
       });
   }
}
```

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

export class HomePage {
   constructor(public toastController: ToastController)
{}

   async presentToast() {
       var toast = await this.toastController.create({
         message: 'Hello, world!',
         duration: 2000
       });
       toast.present();
   }
}
```

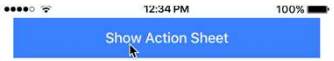
Modal dialogs

```
presentAlert() {
      this.alertController.create({
        header: 'Alert',
        subHeader: 'Subtitle',
        message: 'This is an alert message.',
        buttons: ['OK']
      }).then((alert) => {
        alert.present();
      });
    }
```



Modal dialogs

```
presentActionSheet() {
  this.actionSheetController.create({
    header: 'Albums',
    buttons: [{
      text: 'Delete',
      role: 'destructive',
      icon: 'trash',
      handler: () => {
        console.log('Delete clicked');
      }
    }, {
      text: 'Cancel',
      icon: 'close',
      role: 'cancel',
      handler: () => {
        console.log('Cancel clicked');
      }
    }]
  }).then((actionSheet) => {
   actionSheet.present();
  });
}
```



Modal pages

- Opens up a new page over the current page
  - All pages are components themselves
- Useful for small entry, has more flexibility than dialogs
- Any pages instantiated in model/controller (`.ts`) must be added to the `entryComponents` and the `declarations` in `app.module.ts`
  - Resolves “No component factory found” error

https://stackoverflow.com/questions/51778258/ionic-4-using-modals-via-the-modalcontroller

Modal pages

Create modal page

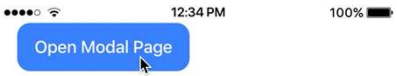
```
import { ModalController } from '@ionic/angular';
import { ModalPage } from '../modal/modal.page';

export class HomePage {
  constructor(public modalController: ModalController) {}

  presentModal() {
    this.modalController.create({
      component: ModalPage,
      componentProps: { name: "IN4MATX 133" }
    }).then((modal) => {
      modal.present();
    });
  }
}
```

Modal pages

Modal page's view & controller

```
<ion-content padding>
  Hello, {{name}}!
  <ion-button (click)="dismiss()">Dismiss</ion-button>
</ion-content>
```

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

export class ModalPage implements OnInit {
  @Input() name:string;
  constructor(public modalController:ModalController) { }

  dismiss() {
      this.modalController.dismiss();
  }
}
```



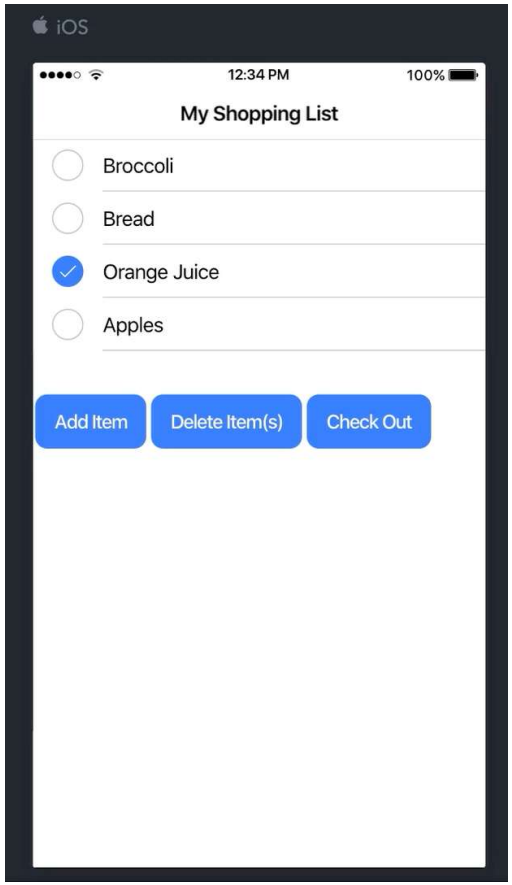
Modal pages

Getting data from modal pages

```
/*modal-page.ts*/
@Input() name:string;

constructor(public
modalController:ModalController) { }

dismiss() {
  this.modalController.dismiss('Hello from
modal!');
}
```

```
/*home-page.ts, creates ModalPage*/
presentModal() {
    this.modalController.create({
      component: ModalPage,
      componentProps: {name: "IN4MATX 133"}
    }).then((modal) => {
      modal.present();
      modal.onDidDismiss().then((data)=>{
          console.log(data);
          // "Hello from modal!"
      })
    });
}
```

Modals


iOS
12:34 PM
100%
My Shopping List
Broccoli
Bread
Orange Juice
Apples
Add Item
Delete Item(s)
Check Out

Routing

Routing

- Like in Angular, `app.routing.module.ts` defines URL routes
- But there's no browser bar in your app…

```
const routes: Routes = [
  { path: '', redirectTo: 'home', pathMatch: 'full' },
  { path: 'home', loadChildren: './home/home.module#HomePageModule' },
  { path: 'modal', loadChildren: './modal/modal.module#ModalPageModule' },
  { path: 'page2', loadChildren: './page2/page2.module#Page2PageModule' }
];
```

Routing

Method 1: defining an `href` attribute

```
<ion-content padding >
  <ion-button href="page2">Go to page 2</ion-button>
</ion-content>
```

Routing

Method 2: using NavController

```
<ion-content padding >
  <ion-button (click)="goToPage2()">Go to page 2</ion-button>
</ion-content>
/*.ts*/
import { NavController } from '@ionic/angular';

export class HomePage {
  constructor(public navCtrl: NavController) {}

  goToPage2() {
    this.navCtrl.navigateForward('/page2');
  }
}
```

- … but supporting undo is important to mobile app design

Routing

Supporting Undo

```
import { NavController } from '@ionic/angular';
export class Page2Page implements OnInit {
  constructor(public navCtrl: NavController) { }
  back() {
    this.navCtrl.back();
  }
}
```

```
<ion-header>
  <ion-toolbar>
  <ion-buttons slot="start">
    <ion-back-button (click)="back()"></ion-back-button>
  </ion-buttons>
    <ion-title>page2</ion-title>
  </ion-toolbar>
</ion-header>
<ion-content padding>
  On page 2
</ion-content>
```

Goals for this lecture

By the end of this lecture, you should be able to…

- Use Ionic Components to make a mobile-friendly app
  - Display structured content with items and lists
  - Style content with colors, icons, and badges
  - Receive user input with inputs and modals
- Use routing to move between pages of your Ionic app

Ionic Setup

Ionic Setup

- `npm install -g ionic`
- `ionic start [projectname]`
- `cd [projectname]`
- `ionic generate [page/component/class] [filename]`

Ionic Serve & Lab

- Run app in your browser with `ionic serve` or `ionic lab`
  - `serve` (left) renders app as it would appear in a browser
  - `lab` (right, recommended) renders iOS and Android views of the app