






Ionic

IN4MATX 133 Discussion

Today's Goals

-  Overview of Ionic and Setting Up
-  Ionic Components (Structural, Item, Inputs, Lists)
-  Services
-  Routing
-  Create a Simple Project

Setting up Ionic

Install Ionic

```
npm info ionic          # Check current versions of Ionic
npm install -g @ionic/cli # Install Ionic CLI globally
ionic -v                # Check Ionic version
```

Create Ionic project

```
ionic start [project_name] # Create a new Ionic project
```

Serve Ionic project

```
cd [project_name] # Change directory to project
ionic serve        # Default: localhost:8100
```

Serve Ionic project in a lab

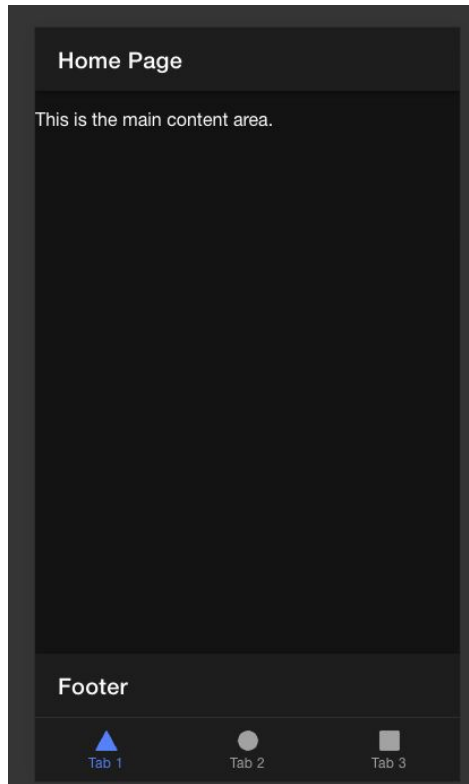
```
cd [project_name] # Change directory to project
npm install @ionic/lab # Install lab in the project (NO -g flag)
ionic serve -l        # Default for lab: localhost:8200
```

Structural Components

```
<ion-header>
  <ion-toolbar>
    <ion-title>Home Page</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <p>This is the main content area.</p>
</ion-content>

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



Components

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

E.g.

ion-item is one of the fundamental UI components in Ionic. It works like an HTML `<div>`, but it automatically **aligns content** and is displayed as a row by default. It is commonly used to wrap text, input fields, buttons, icons, and other elements.

```
<ion-list>
<!-- List items will be inside this -->
</ion-list>
```

- `<ion-list>` is an **Ionic component** used to group multiple list items (`<ion-item>`)
- It acts as a **container** for a structured list of items.
- **Equivalent to HTML's `` or `<div>`**, but styled specifically for mobile apps.

Components

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

```
<ion-item>
  <ion-label>Basic Item</ion-label>
</ion-item>
```

- `<ion-item>` is an **ionic component** that represents **a single row** in a list.
- It **automatically aligns its content** (text, icons, inputs, etc.).
- It is like `` in HTML but with **built-in mobile styling**.
- `<ion-label>` is a **text container** inside an `<ion-item>`.
- It ensures **consistent styling and alignment** across different platforms.
- It can be used to label **inputs, toggles, checkboxes, and more**.

Components

```
<ion-content>
  <!-- Item Components Demo -->
  <ion-list>
    <!-- Basic item -->
    <ion-item>
      <ion-label>Basic Item</ion-label>
    </ion-item>

    <!-- Item with toggle switch -->
    <ion-item>
      <ion-label>Toggle</ion-label>
      <ion-toggle></ion-toggle>
    </ion-item>

    <!-- Item with button -->
    <ion-item>
      <ion-label>Button</ion-label>
      <ion-button id="open-alert">Click Me</ion-button>
    </ion-item>
  </ion-list>

  <!-- Ionic built-in alert (No need for TypeScript function) -->
  <ion-alert
    trigger="open-alert"
    header="Hello!"
    message="You clicked the button!"
    buttons="OK">
  </ion-alert>
</ion-content>
```

Tab 1

This is Tab 1 content.

Basic Item

Toggle



Button

CLICK ME

Ionic Service

An Ionic Service is a reusable module that handles data management, API interactions, device access (camera, storage, GPS, etc.), and global application logic. It works similarly to **Angular Services** and helps keep components clean by separating UI from business logic.

```
ionic generate service myService
```

Open `src/app/app.module.ts` and add the service to providers

```
import { MyService } from '../services/my-service.service';
@NgModule({
  providers: [MyService]
})
export class AppModule {}
```

Inject and Use the Service in a Component

```
import { MyService } from '../services/my-service.service';

constructor(private myService: MyService) {}
```


Ionic Routing

Ionic uses Angular's routing system to manage page navigation.

Pages → Full views with nested components

Components → Small reusable UI elements

Ionic stores all routes in `app-routing.module.ts`. This file maps URLs to specific pages.

Ionic provides **two ways** to navigate between pages:

1 View-based (ion-button and routerLink)

```
<ion-button routerLink="/about">Go to About</ion-button>
```

2 Controller-based (Router.navigate())

```
import { Router } from '@angular/router';  
constructor(private router: Router) {}  
goToAbout() {  
  this.router.navigate(['/about']);  
}
```

Ionic Page Lifecycle

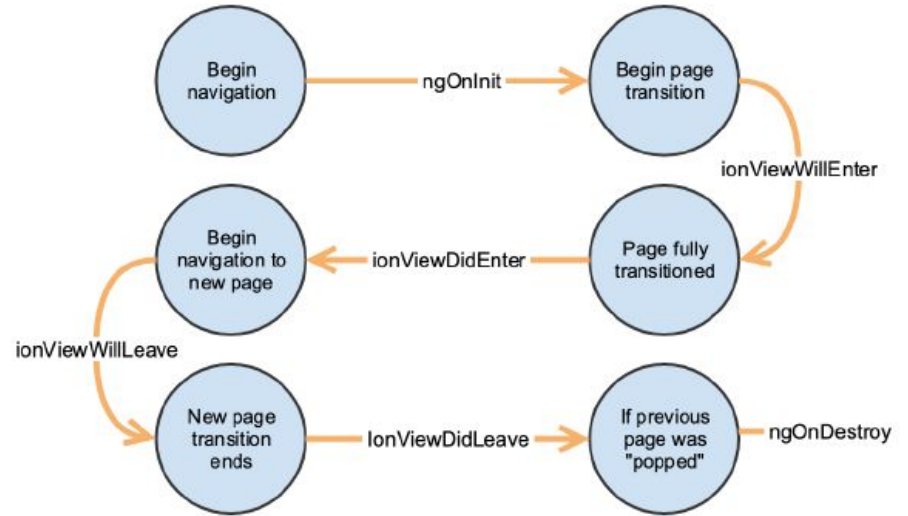
Mostly used are

ngOnInit

- Fired once when a page is loaded

ngOnDestroy

- Fired right before the view is destroyed.



Ionic to do list Demo

- **Add Header, Content Area, and Footer in Home Page**
 - Structure the home page using `<ion-header>`, `<ion-content>`, and `<ion-footer>`.
- **Generate a New Page**
 - Run `ionic generate page list` to create the necessary files.
- **Add Navigation Between Pages**
 - In `home.page.html`, add a button to navigate to the list page.
 - In `home.page.ts`, import `NavController` and define a function to navigate forward.
 - In `list.page.html`, add a back button inside `ion-buttons`.
 - In `list.page.ts`, define a function to navigate back.
- **Add a List in `list.page.ts`**
- **Create a Card-Based List in `list.page.html`**
- **Generate a Service for Managing Groceries**
 - Run `ionic generate service groceries`.
 - Import the service and add it to providers in `app.module.ts`.
- **Inject the Service in Home and Use Input for Adding Items**