Développement Mobile IONIC

Séquence 3

PLAN

- Les composants:
  - Bouton
  - Liste
  - Select
  - Cards
  - FAB

Les composants

- Les applications ionic sont constituées composants, qui vous permettent de créer rapidement l'interface utilisateur de votre application.
- Ionic est fourni en stock avec un certain nombre de composants, y compris des cartes, des listes et des onglets.
- Ionic propose un ensemble de composants et sous composants pour développer votre application jusqu’à terme.

Les Boutons

- Les boutons fournissent un élément cliquable, qui peut être utilisé dans des formulaires ou n'importe où qui nécessite une fonctionnalité de bouton standard simple.
- Ils peuvent afficher du texte, des icônes ou les deux. Les boutons peuvent être stylisés avec plusieurs attributs pour avoir un aspect spécifique.

Les Boutons

```
<ion-content >
<!-- Default -->
<ion-button>Default</ion-button>
<!-- Anchor -->
<ion-button href="#">Anchor</ion-button>
<!-- Colors -->
<ion-button color="primary">Primary</ion-button>
<ion-button color="secondary">Secondary</ion-button>
<ion-button color="tertiary">Tertiary</ion-button>
<ion-button color="success">Success</ion-button>
<ion-button color="warning">Warning</ion-button>
<ion-button color="danger">Danger</ion-button>
<ion-button color="light">Light</ion-button>
<ion-button color="medium">Medium</ion-button>
<ion-button color="dark">Dark</ion-button>      You, a few seconds ago
<!-- Expand -->
<ion-button expand="full">Full Button</ion-button>
<ion-button expand="block">Block Button</ion-button>
<!-- Round -->
<ion-button shape="round">Round Button</ion-button>
<!-- Fill -->
<ion-button expand="full" fill="outline">Outline + Full</ion-button>
<ion-button expand="block" fill="outline">Outline + Block</ion-button>
<ion-button shape="round" fill="outline">Outline + Round</ion-button>
```

```
<!-- Fill -->
<ion-button expand="full" fill="outline">Outline + Full</ion-button>
<ion-button expand="block" fill="outline">Outline + Block</ion-button>
<ion-button shape="round" fill="outline">Outline + Round</ion-button>
<!-- Icons -->
<ion-button>
  <ion-icon slot="start" name="star"></ion-icon>
  Left Icon
</ion-button>
<ion-button>
  Right Icon
  <ion-icon slot="end" name="star"></ion-icon>
</ion-button>

<ion-button>
  <ion-icon slot="icon-only" name="star"></ion-icon>
</ion-button>

<!-- Sizes -->
<ion-button size="large">Large</ion-button>
<ion-button>Default</ion-button>
<ion-button size="small">Small</ion-button>
</ion-content>
```

Les Boutons

ion-button

CONTENTS

Expand

Fill

Size

Usage

Properties

Events

CSS Custom Properties

Slots



Liste

- Les listes sont composées de plusieurs lignes d'éléments qui peuvent contenir du texte, des boutons, des bascules, des icônes, des miniatures et bien plus encore. Les listes contiennent généralement des éléments avec un contenu de données similaire, comme des images et du texte.

- Les listes prennent en charge plusieurs interactions, notamment en faisant glisser des éléments pour révéler des options, en les faisant glisser pour réorganiser des éléments dans la liste et en supprimant des éléments.

Liste

```
<ion-content >

  <!-- List of Text Items -->
<ion-list>
  <ion-item>
    <ion-label>UVS</ion-label>
  </ion-item>
  <ion-item>
    <ion-label>IDA</ion-label>
  </ion-item>
  <ion-item>
    <ion-label>Mobile</ion-label>
  </ion-item>
  <ion-item>
    <ion-label>Ionc</ion-label>
  </ion-item>
</ion-list>
```

```
<!-- List of Input Items -->
<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>      You, 2 minutes ago • Uncommitte
```

```
<!-- List of Input Items -->
<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>
```

Liste

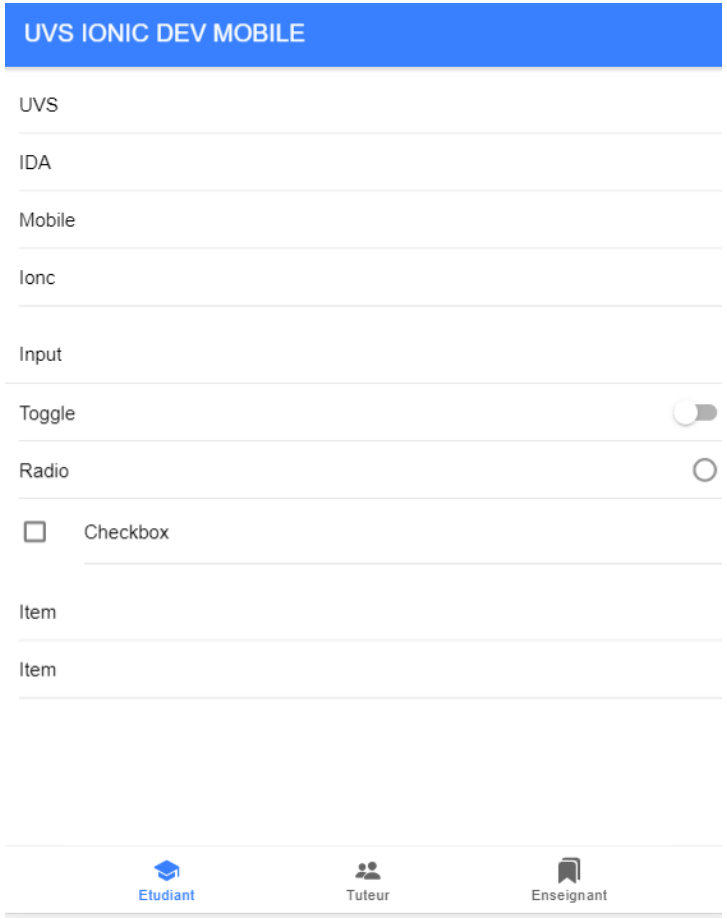
ion-list

CONTENTS

Usage

Properties

Methods



Select

- Les sélections sont des contrôles de formulaire permettant de sélectionner une ou plusieurs options dans un ensemble d'options, similaires à un élément <select> natif. Lorsqu'un utilisateur appuie sur la sélection, une boîte de dialogue apparaît avec toutes les options dans une grande liste facile à sélectionner.

- Une sélection doit être utilisée avec les éléments enfants <ion-select-option>. Si l'option enfant ne reçoit pas d'attribut value, son texte sera utilisé comme valeur.

Single Selection

Par défaut, la sélection permet à l'utilisateur de sélectionner une seule option. L'interface d'alerte présente aux utilisateurs une liste d'options de style bouton radio. L'interface de la feuille d'action ne peut être utilisée qu'avec une seule sélection de valeur. La valeur du composant sélectionné reçoit la valeur de la valeur de l'option sélectionnée.

```
<ion-content >
  <ion-list>
    <ion-list-header>
      <ion-label>
        Single Selection
      </ion-label>
    </ion-list-header>
    <ion-item>
      <ion-label>Genre</ion-label>
      <ion-select placeholder="Select One">
        <ion-select-option value="H">Homme</ion-select-option>
        <ion-select-option value="F">Femme</ion-select-option>
      </ion-select>
    </ion-item>
    <ion-item>
      <ion-label>UVS Filiere</ion-label>
      <ion-select value="brown" okText="Okay" cancelText="Dismiss">
        <ion-select-option value="brown">Dev Web</ion-select-option>
        <ion-select-option value="blonde">Dev Modile</ion-select-option>
        <ion-select-option value="black">Anglais</ion-select-option>
        <ion-select-option value="red">Math</ion-select-option>
      </ion-select>
    </ion-item>
  </ion-list>       You, a few seconds ago • Uncommitted changes
</ion-content>
```

Multi Selection

En ajoutant l'attribut ***multiple*** au select, les utilisateurs peuvent sélectionner plusieurs options. Lorsque plusieurs options peuvent être sélectionnées, la superposition d'alerte présente aux utilisateurs une liste d'options de style à cocher. La valeur du composant sélectionné reçoit un tableau de toutes les valeurs d'options sélectionnées.

```
<ion-list>
  <ion-list-header>
    <ion-label>
      Multiple Selection
    </ion-label>
  </ion-list-header>        You, a few seconds ago • Uncommitted changes
  <ion-item>
    <ion-label>Toppings</ion-label>
    <ion-select multiple="true" cancelText="Nah" okText="Okay!">
      <ion-select-option value="bacon">Bacon</ion-select-option>
      <ion-select-option value="olives">Black Olives</ion-select-option>
      <ion-select-option value="xcheese">Extra Cheese</ion-select-option>
      <ion-select-option value="peppers">Green Peppers</ion-select-option>
      <ion-select-option value="mushrooms">Mushrooms</ion-select-option>
      <ion-select-option value="onions">Onions</ion-select-option>
      <ion-select-option value="pepperoni">Pepperoni</ion-select-option>
      <ion-select-option value="pineapple">Pineapple</ion-select-option>
      <ion-select-option value="sausage">Sausage</ion-select-option>
      <ion-select-option value="Spinach">Spinach</ion-select-option>
    </ion-select>
  </ion-item>
  <ion-item>
    <ion-label>Pets</ion-label>
    <ion-select multiple="true" [value]="['bird', 'dog']">
      <ion-select-option value="bird">Bird</ion-select-option>
      <ion-select-option value="cat">Cat</ion-select-option>
      <ion-select-option value="dog">Dog</ion-select-option>
      <ion-select-option value="honeybadger">Honey Badger</ion-select-option>
    </ion-select>
  </ion-item>
</ion-list>
```

Multi Selection

ion-select

CONTENTS

Interfaces

Single Selection

Object Value References

Select Buttons

Interface Options

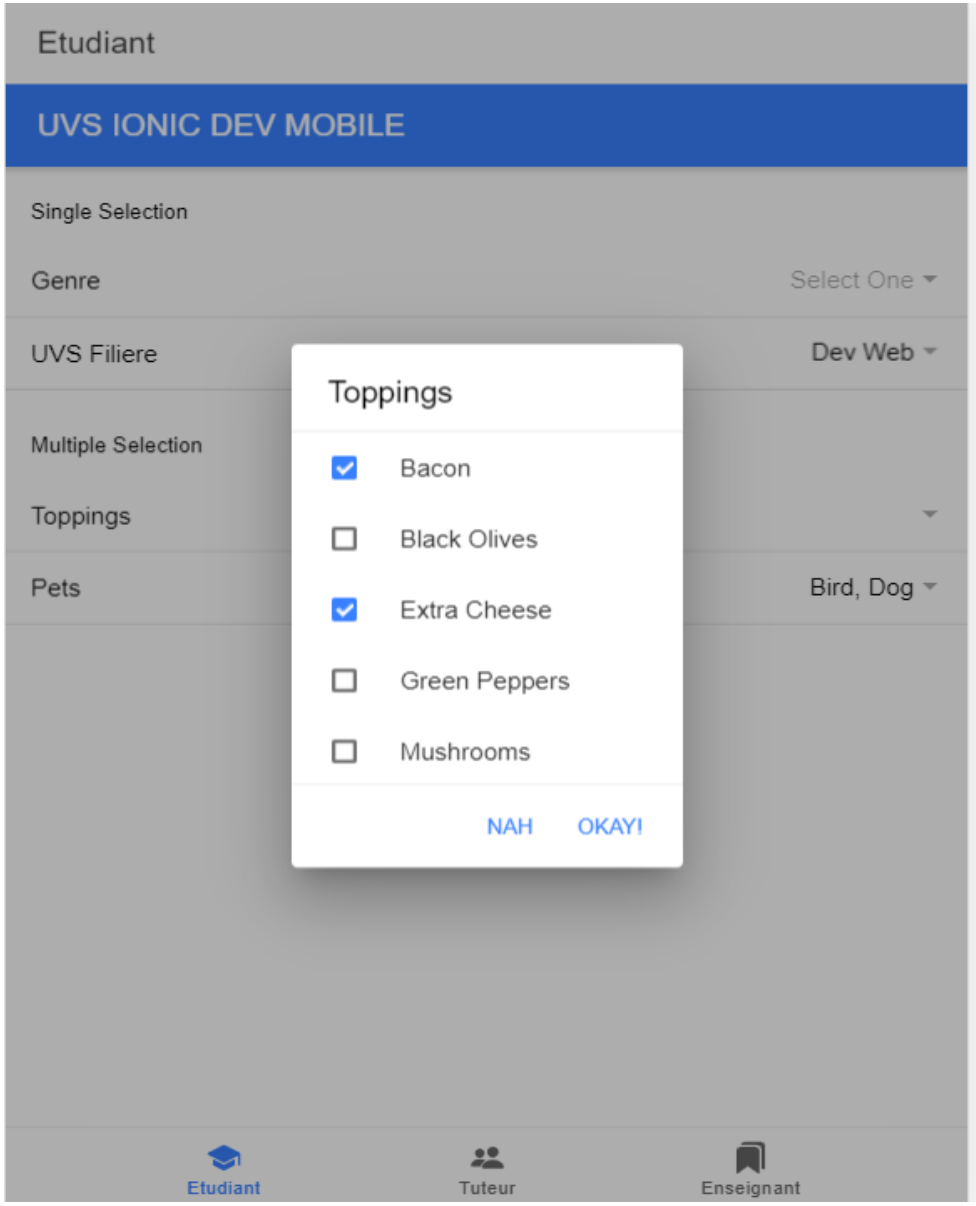
Usage

Properties

Events

Methods

CSS Custom Properties



Card

- Card est un élément standard de l'interface utilisateur qui sert de point d'entrée à des informations plus détaillées. Un card peut être un composant unique, mais elle est souvent composée d'un en-tête, d'un titre, d'un sous-titre et d'un contenu.

```
<ion-header>
  <ion-toolbar color="primary">
    <ion-title>UVS IONIC DEV MOBILE</ion-title>
  </ion-toolbar>
</ion-header>
<ion-content padding>
  <ion-card>
    <img src="assets/imgs/logoUVS.JPG" />
    <ion-card-header text-center>
      Formations en cours
    </ion-card-header>
    <ion-card-content>
      <p>Profession : Ingénieur informatique</p>
      <p>Localisation : IDA UVS</p>
      <a href="https://www.uvs.sn/nos-formations/">Formation UVS </a>
    </ion-card-content>
  </ion-card>
</ion-content>
```

Card

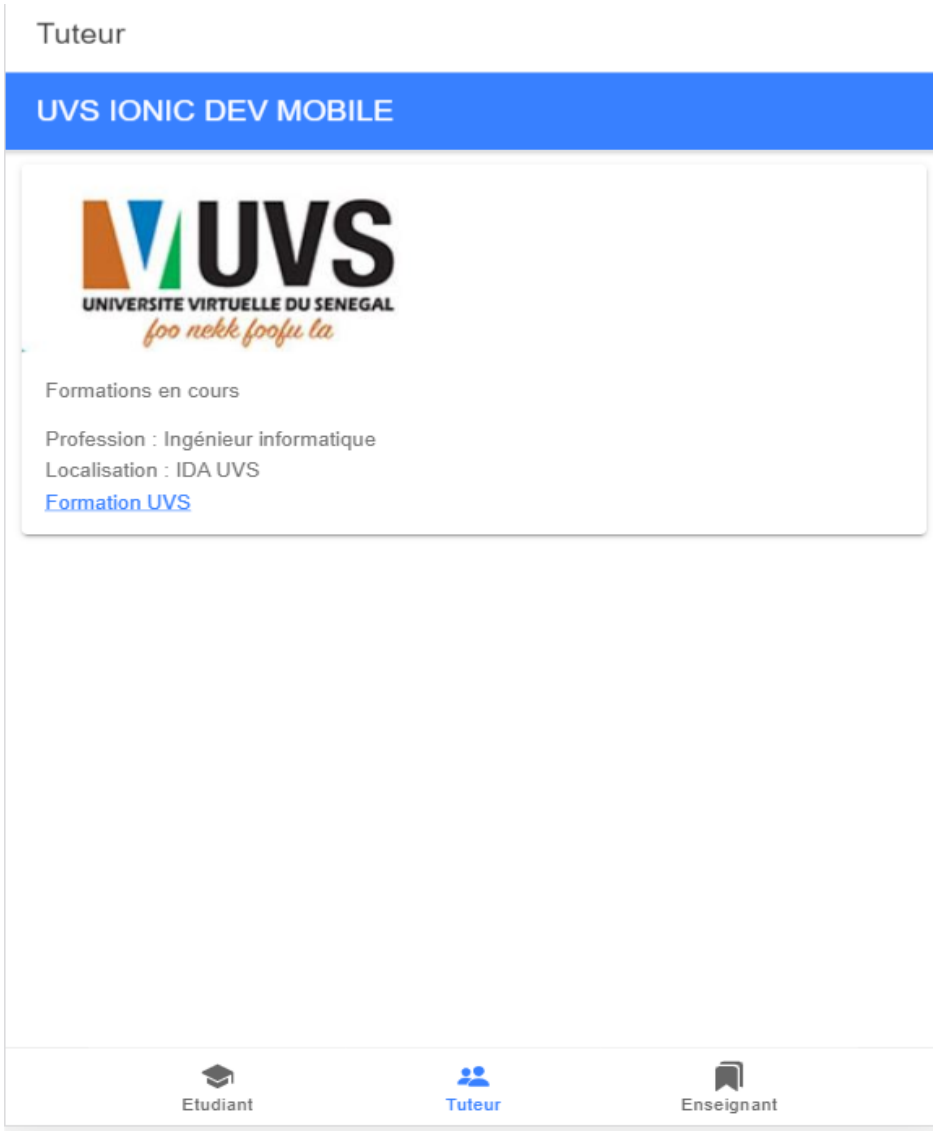
ion-card

CONTENTS

Usage

Properties

CSS Custom Properties



Angular

- Pour ce cours nous allons parler d’angular et Ionic, même si Ionic peut être combiné avec VUJS ou React.
- @ionic/angular combine l'expérience Ionic de base avec les outils et les API qui sont adaptés aux développeurs d’angular.
- Angular possède des outils intégrés pour aider à automatiser les mises à niveau et fournir des commentaires aux développeurs chaque fois que des modifications sont apportées à une API. Cela réduit la friction des mises à jour et maintient l'écosystème dans un état à feuilles persistantes.

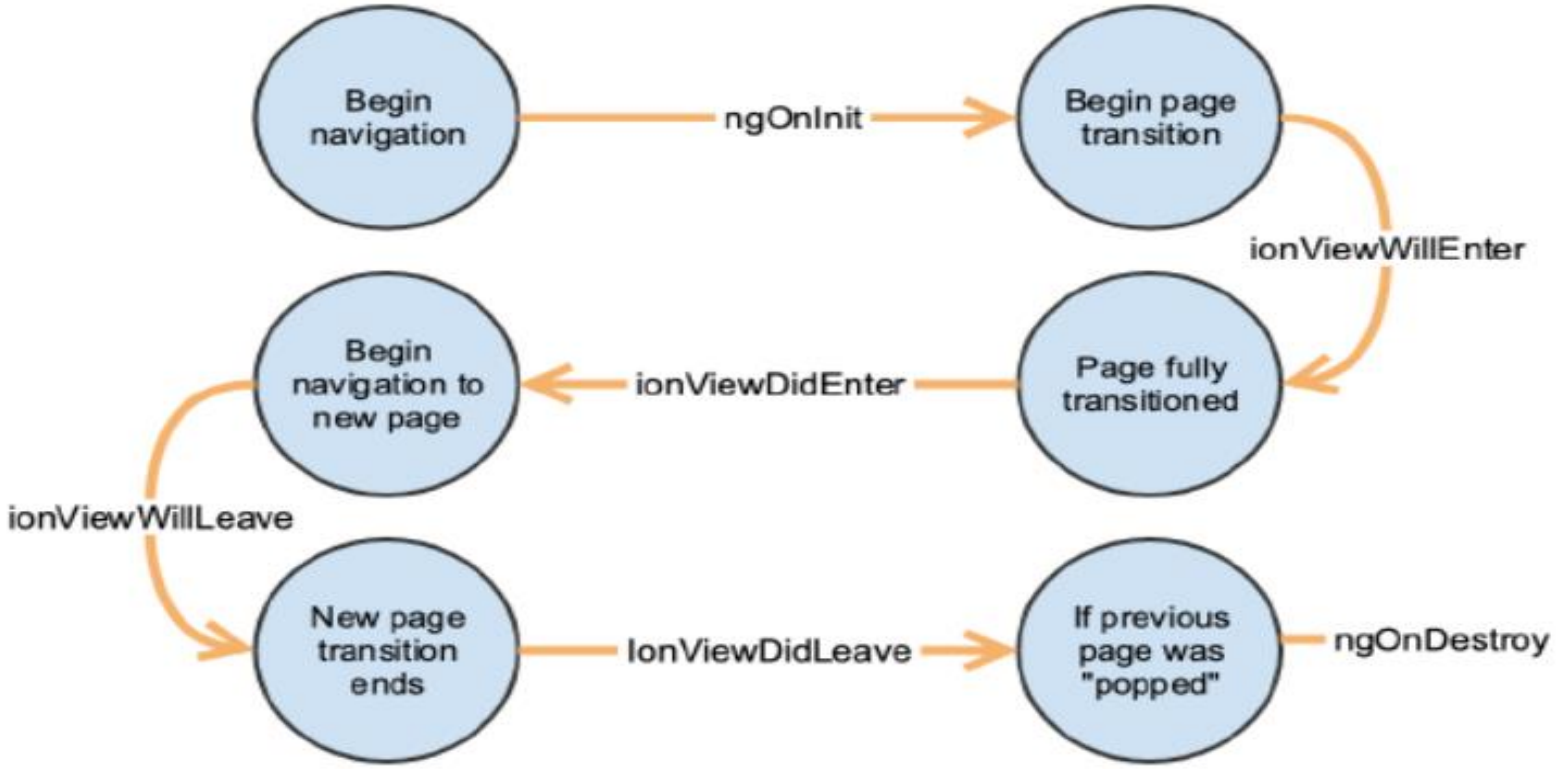
Angular

- Avec Ionic 4+, Le stack angular officielle pour la création d'une application et le routage sont utilisés, de sorte que votre application peut correspondre au reste du grand écosystème angulaire.

- Dans les cas où des fonctionnalités plus avisées sont nécessaires, Ionic fournit ***@ ionic / angular-toolkit***, qui se construit et s'intègre à la ***CLI Angular*** officielle et fournit des fonctionnalités spécifiques aux applications ***@ ionic / angular.***

Cycle de vie d’une Page Ionic

- Ce guide décrit le fonctionnement du cycle de vie des pages dans une application conçue avec Ionic et Angular.



Evènement cycle de vie Angular

- Ionic embrasse les événements du cycle de vie fournis par Angular. Les deux événements d'angular que vous trouverez le plus souvent sont:

Nom de l'événement	La description
`ngOnInit`	Déclenché une fois lors de l'initialisation du composant. Cet événement peut être utilisé pour initialiser des membres locaux et effectuer des appels vers des services qui ne doivent être effectués qu'une seule fois.
`ngOnDestroy`	Tiré juste avant que Angular ne détruise la vue. Utile pour le nettoyage comme la désinscription des observables.

Evènement cycle de vie Ionic

- En plus des événements du cycle de vie d'Angular, Ionic Angular fournit quelques événements supplémentaires que vous pouvez utiliser:

La différence entre ***ionViewWillEnter*** et ***ionViewDidEnter*** est que le premier se déclenche juste après ***ngOnInit*** mais avant le début de la transition de la page, et le second directement après la fin de la transition.

Pour ***ionViewWillLeave*** et ***ionViewDidLeave***, ***ionViewWillLeave*** sont appelés directement avant le début de la transition de la page en cours. Et ***ionViewDidLeave*** n'est appelé qu'après la transition réussie de la nouvelle page (après le ***ionViewDidEnter*** déclenchement des nouvelles pages).

Nom de l'événement	La description
`ionViewWillEnter`	Déclenché lorsque le routage du composant vers est sur le point de s'animer en vue.
`ionViewDidEnter`	Déclenché lorsque le routage du composant vers a terminé l'animation.
`ionViewWillLeave`	Déclenché lorsque le routage du composant est sur le point de s'animer.
`ionViewDidLeave`	Déclenché lorsque le routage du composant vers a terminé l'animation.

Gestion cycle de vie Ionic page

- Ionic a sa compilation, lance le routeur grâce ***<ion-router-outlet />.*** Elle étend Angular ***<router-outlet />*** avec quelques fonctionnalités supplémentaires pour permettre de meilleures expériences pour les appareils mobiles.
- Lorsqu'une application est intégrée <ion-router-outlet />, Ionic traite la navigation un peu différemment. Lorsque vous accédez à une nouvelle page, Ionic conserve l'ancienne page dans le DOM existant, mais la cache à votre vue et transfère la nouvelle page. La raison pour laquelle nous faisons cela est double:
  - 1) Nous pouvons maintenir l'état de l'ancienne page (données à l'écran, position de défilement, etc.)
  - 2) Nous pouvons fournir une transition plus fluide vers la page car elle est déjà là et n'a pas besoin d'être recréée.

Angular Navigation

- Angular Router est l'une des plus importante librairie de l'application Angular. Sans le routage ou la librairie de routage, l'application serait une seule page.
- Avec le routage, on peut créer une application très en liant les différentes pages avec de belles animations.
- Routage simple:

Lorsque notre application se charge, le routeur démarre en lisant l'URL que l'utilisateur essaie de charger. Dans notre exemple, notre route recherche '', qui est essentiellement notre route d'index. Donc le fichier ***LoginComponent*** est chargé.

Ce pattern de correspondance des chemins se poursuit pour chaque entrée que nous avons dans la configuration du routeur

```
import { RouterModule } from '@angular/router';

@NgModule({
  imports: [
  ...
  RouterModule.forRoot([
    { path: '', component: LoginComponent },
    { path: 'detail', component: DetailComponent },
  ])
  ],
})
```

Gestion des Redirections

- Les redirections fonctionnent de la même manière qu'un objet de route typique, mais ne renferment que quelques clés différentes.
- Dans l’exemple suivant, nous recherchons le chemin d'index de notre application. Ensuite, si il est chargé, une redirection es faite vers la ***login route***. La dernière clé de ***pathMatch*** est requise pour indiquer au routeur comment il doit rechercher le chemin.
- L’utilisation de ***full*** permet d’indiquer au routeur que nous devons comparer le chemin complet, même s'il finit par ressembler à quelque chose ***/route1/route2/route3***. Cela signifie que si nous avons comme configuation:

```
{ path: '/route1/route2/route3', redirectTo: 'login', pathMatch: 'full' },
{ path: 'login', component: LoginComponent },
```

Navigation vers les autres Routes

- Pour naviguer vers les autres routes on peut utiliser la directive ***routerLink.*** Soit l'exemple suivant:

```
RouterModule.forRoot([
  { path: '', component: LoginComponent },
  { path: 'detail', component: DetailComponent }
]);
```

- Au niveau de la partie vue on a:

```
<ion-header>
  <ion-toolbar>
    <ion-title>Login</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content class="ion-padding">
  <ion-button [routerLink]="['/detail']">Go to detail</ion-button>
</ion-content>
```

RouterLink fonctionne de manière similaire que ***href ,*** mais la différence est que l'URL est construit sous forme de tableau. Ce qui peut fournir des chemins complexe.

Navigation Par API Router

- On peut par contre aussi naviguer avec les API Router

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';

@Component({
  ...
})
export class LoginComponent {

  constructor(private router: Router){}

  navigate(){
    this.router.navigate(['/detail'])
  }
}
```

Routage avec Lazy Loading

- Dans l’exemple précédent, lorsque le contenu de la page entière est chargé et affiché en une seule fois. Même si cela permet au navigateur ou à l'application de mettre en cache le contenu, rien ne garantit que l'utilisateur verra réellement tout le contenu à afficher.
- C'est donc là que le lazy-loading joue un rôle important, au lieu de charger tout le contenu en bloc, il effectue simplement le chargement des parties demandées. Avec le chargement lazy loading, les pages sont créées avec d'espace réservé qui remplace le contenu actuel lorsque l'utilisateur en a besoin.
- Le lazy-loading semble être un processus compliqué, mais est en fait très simple.

Routage avec Lazy Loading

```
import { RouterModule } from '@angular/router';

@NgModule({
  imports: [
  ...
  RouterModule.forRoot([
    { path: '', redirectTo: 'login', pathMatch: 'full' },
    { path: 'login', loadChildren: './login/login.module#LoginModule' },
    { path: 'detail', loadChildren: './detail/detail.module#DetailModule' }
  ])
  ],
})
```

```
...
import { RouterModule } from '@angular/router';
import { LoginComponent } from './login.component';

@NgModule({
  imports: [
  ...
  RouterModule.forChild([
    { path: '', component: LoginComponent },
  ])
  ],
})
```

FIN DE LA SEQUENCE 3

References

- https://ionicframework.com/docs/
- https://openclassrooms.com/fr/courses/5098931-developpez-une-application-mobile-multiplateforme-avec-ionic-3
- https://ionic.mobiletuto.com/v3/chap10/chap10-5.html
- https://ionicframework.com/docs/components
- https://ionicacademy.com/ionic-routing-navigation/