Mobile Application Development

LECTURE 5

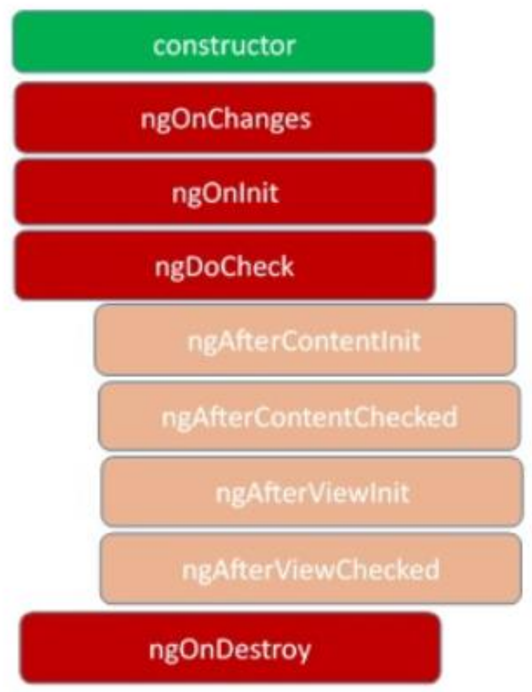
Lifecycle Methods in Angular

- What are lifecycle methods? every component has a life-cycle, a number of different stages it goes through.
- Lifecycle methods are functions that run during various times for a component.
- There are 8 different stages in the component lifecycle in Angular. Every stage is called as lifecycle hook event.

Lifecycle Methods in Angular

- We can use these hook events in different phases of our application to obtain control of the component.
- Angular executes its lifecycle hook methods in a specific order.
- Every framework (React.js , Vue.js) have lifecycle methods. Ionic also has few mobile related lifecycle methods.

Lifecycle Methods in Angular



Since a component is a TypeScript class, every component must have a constructor method. The constructor of the component class executes, first, before the execution of any other lifecycle hook events.

After executing the constructor, Angular executes its lifecycle hook methods in a specific order.

Lifecycle Methods in Angular

- Important lifecycle methods that you will use regularly and should know are ngOnInit, ngOnChanges, ngOnDestroy and ngOnViewInit. Others are not that important but you need to know they exist.

- **ngOnChanges()** - This event executes every time when a value of an @Input() control within the component has been changed.

- **ngOnInit()** - This event initializes after Angular first displays the data-bound properties or when the component has been initialized.

Lifecycle Methods in Angular

- **ngAfterViewInit()** - This method is initialized after Angular initializes the component's view and child views.

- **ngOnDestroy()** - This method is very useful for unsubscribing from the observables and detaching the event handlers to avoid memory leaks. This is executed when ever the component is detroyed/removed from view.

- In React.js, the equivalent to **ngOnInit()** is **componentDidMount()**, in Vue.js, it is **Created()**

“People see in you what you see in yourself.”

Constructor vs ngOnInit

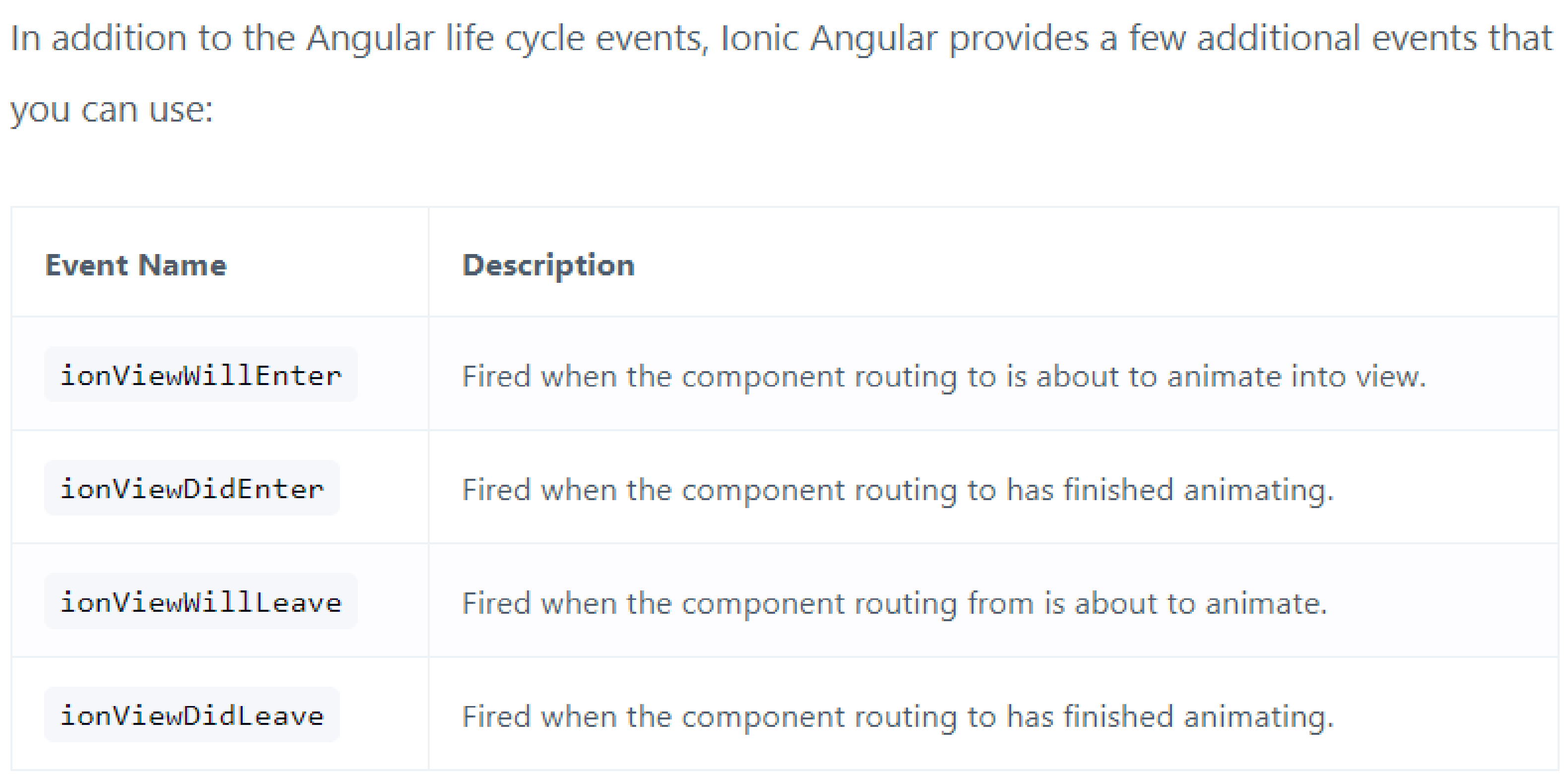
- If we need to inject any dependencies (services) into the component, then the constructor is the best place to inject those dependencies. So you should use **constructor()** to setup **Dependency Injection** and not much else**.**

- **Dependency Injection (DI)** is a software design pattern that deals with how components get hold of their **dependencies**. (we will get back to this topic in lecture 11 when we discuss Services).

Constructor vs ngOnInit

- OK, first of all ngOnInit is part of Angular lifecycle, while constructor is part of ES6 JavaScript class / TypeScript.
- The Javascript engine is what handles the constructor, not Angular. Constructors are not related to Angular, they are a TypeScript feature / ES6 JavaScript class.
- Do any sort of initialization in ngOnInit and not in constructor. Why? your native elements will not be available during the constructor phase. Component input initial values aren't available in constructor, but they are available in ngOnInit.

Ionic Page lifecycle

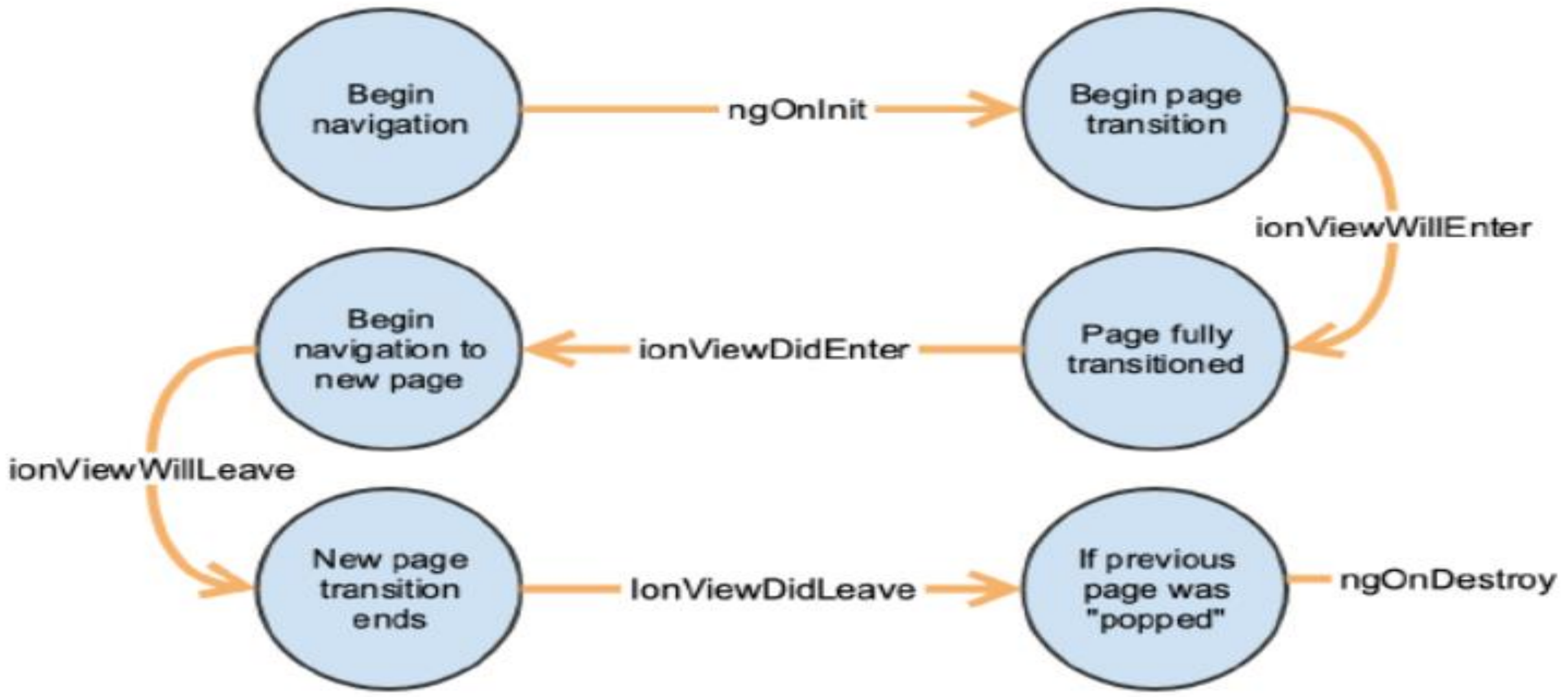


In addition to the Angular life cycle events, Ionic Angular provides a few additional events that you can use:

Event Name	Description
`ionViewWillEnter`	Fired when the component routing to is about to animate into view.
`ionViewDidEnter`	Fired when the component routing to has finished animating.
`ionViewWillLeave`	Fired when the component routing from is about to animate.
`ionViewDidLeave`	Fired when the component routing to has finished animating.

https://ionicframework.com/docs/angular/lifecycle

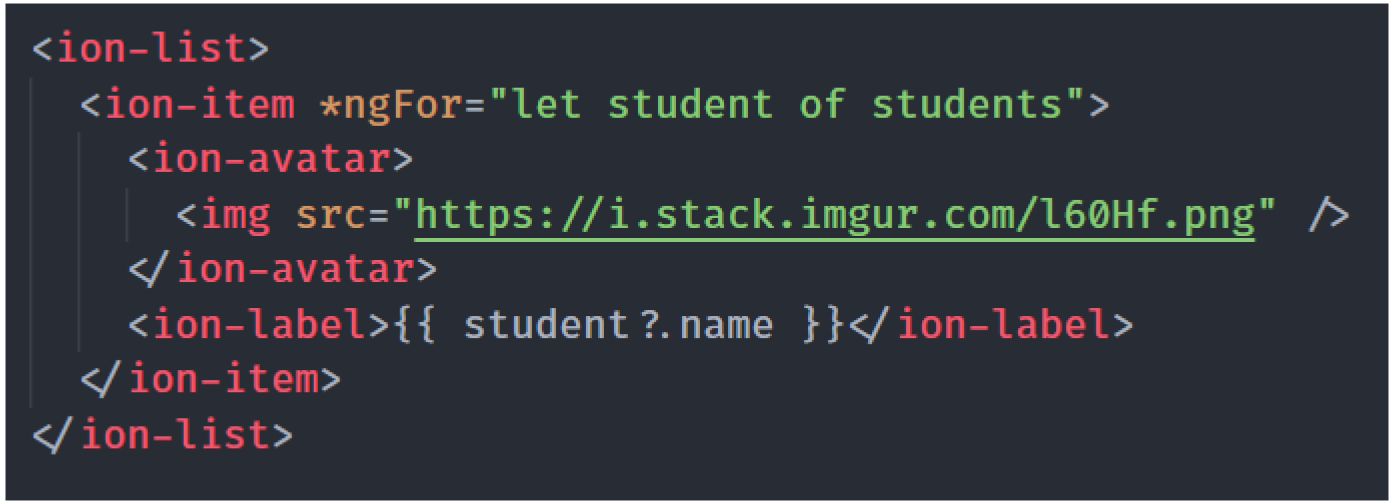
Ionic Page lifecycle



*ngFor usage in Ionic

- *ngFor is a builtin directive that you can use in your Angular templates to iterate through lists

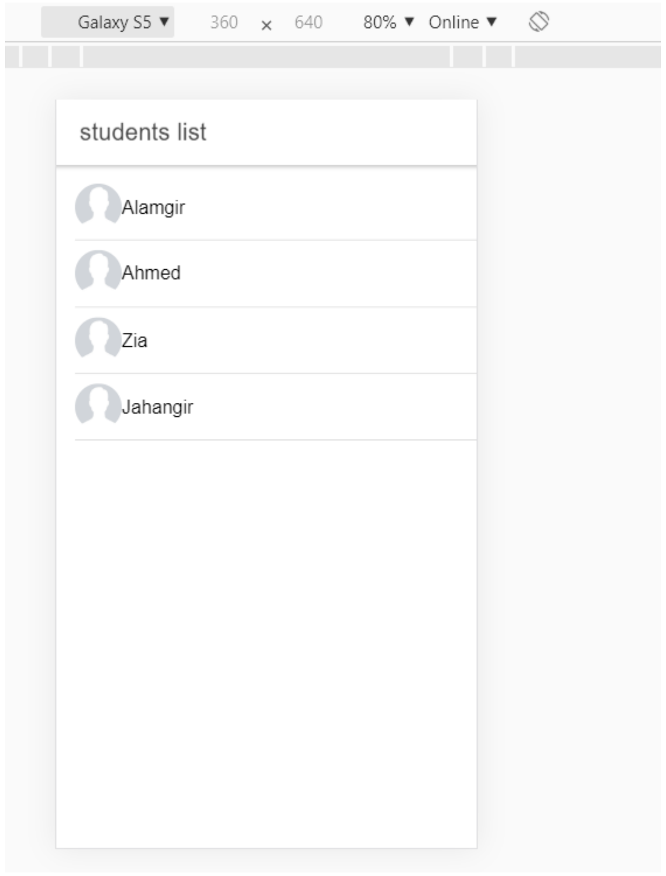
  and arrays of data in order to display it.



```
<ion-list>
  <ion-item *ngFor="let student of students">
    <ion-avatar>
      <img src="https://i.stack.imgur.com/l60Hf.png" />
    </ion-avatar>
    <ion-label>{{ student?.name }}</ion-label>
  </ion-item>
</ion-list>
```

*ngFor usage in Ionic

- Result of *ngFor



students list

Alamgir

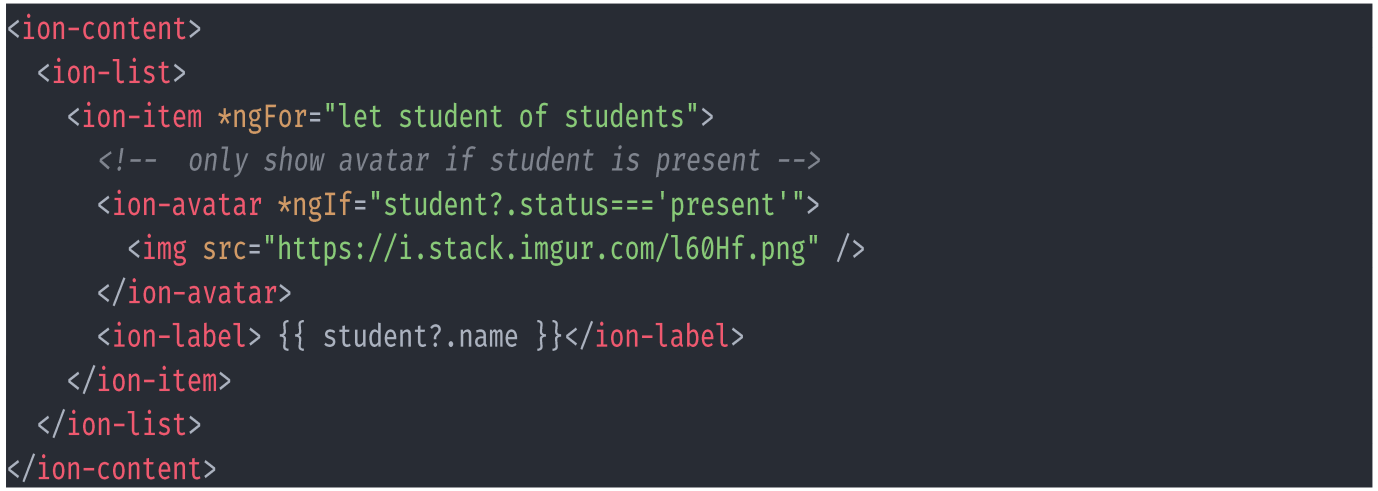
Ahmed

Zia

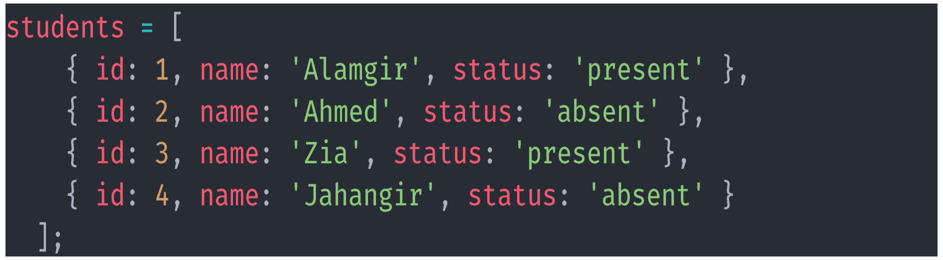
Jahangir

*ngIf usage in Ionic

- *ngIf removed or adds an element to View depending on the condition.



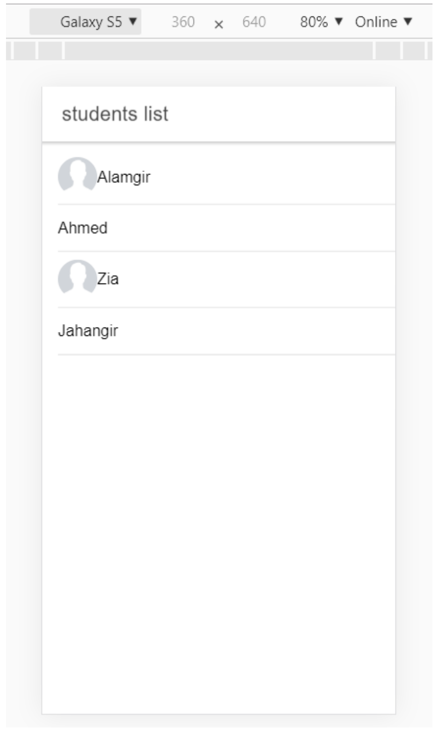
```
<ion-content>
  <ion-list>
    <ion-item *ngFor="let student of students">
      <!--  only show avatar if student is present -->
      <ion-avatar *ngIf="student?.status==='present'">
        <img src="https://i.stack.imgur.com/l60Hf.png" />
      </ion-avatar>
      <ion-label> {{ student?.name }}</ion-label>
    </ion-item>
  </ion-list>
</ion-content>
```



```
students = [
    { id: 1, name: 'Alamgir', status: 'present' },
    { id: 2, name: 'Ahmed', status: 'absent' },
    { id: 3, name: 'Zia', status: 'present' },
    { id: 4, name: 'Jahangir', status: 'absent' }
  ];
```

*ngIf usage in Ionic

- *ngIf result. No Avatar for Ahmed and Zia as they are ‘absent’.



students list

Alamgir

Ahmed

Zia

Jahangir