academeez

@angular/universal - SSR

7. angular help with isomorphic code

Isomorphic code helpers

academeez

- Angular provides us injectable services and InjectionToken to help us write our async code
- In this lesson we will go over the common ones.

Location

academeez

- Use this service instead of the browser **location** api.
- usually it’s better to use the **Router**
- will allow you to use **location** and use it in the server side as well

```
export class AppComponent implements OnInit {
  constructor(private _location: Location) {}

  ngOnInit() {
    console.log(this._location.path());
  }
}
```

DOCUMENT

academeez

- **document** is a browser exclusive api and using it on the server side will cause a crash
- Angular provides a document in the server side based on the **domino** package
- You can ask the DI for the DOCUMENT injection token and get an isomorphic document

```
export class AppComponent implements OnInit {
  constructor(@Inject(DOCUMENT) private _document: Document) {}

  ngOnInit() {
    const nav = this._document.getElementById('hello');
    nav.classList.add('foo-bar');
  }
}
```

PLATFORM_ID

academeez

- You can ask angular: “which platform is currently running my code?”
- Angular provides you with an **InjectionToken** called **PLATFORM_ID** which you can use to know which platform currently runs the code
- you can use that token and pass it to the functions **isPlatformBrowser, isPlatformServer**

```
export class AppComponent implements OnInit {
  constructor(@Inject(PLATFORM_ID) private _platformId) {}

  ngOnInit() {
    if (isPlatformBrowser(this._platformId)) {
      console.log('we are running on the browser');
    }
    if (isPlatformServer(this._platformId)) {
      console.log('we are running on the server');
    }
  }
}
```

PLATFORM_ID - warning

academeez

- according to @angular/universal: https://github.com/angular/universal/blob/master/docs/gotchas.md#strategy-2-guards

You may read online and elsewhere that the recommended approach is to use **`isPlatformBrowser`** or **`isPlatformServer`**.
This guidance is **incorrect**. This is because you wind up creating platform-specific code branches in your application code. This not only increases the size of your application unnecessarily, but it also adds complexity that then has to be maintained. By separating code into separate platform-specific modules and implementations, your base code can remain about business logic, and platform-specific exceptions are handled as they should be: on a case-by-case abstraction basis. This can be accomplished using Angular's Dependency Injection (DI) in order to remove the offending code and drop in a replacement at runtime.

- Favor shims instead of branching the code for browser/server
- More about shim in the next lesson

ElementRef

academeez

- An angular class wrapper around a native view element
- Since angular is Platform agnostic the view elements do not have to be DOM
- You can update the view element using ElementRef and it will also be supported in SSR
  - It’s discourage to use ElementRef to update the view element, please use Renderer2

```
export class AppComponent implements AfterViewInit {
  @ViewChild('nav')
  nav: ElementRef<HTMLElement>;

  ngAfterViewInit() {
    this.nav.nativeElement.classList.add('foo-bar');
  }
}
```

Renderer2

academeez

- Using this service you can easily manipulate the view elements while maintaining platform agnostic and SSR
- Better to use this service than injecting the **DOCUMENT** or using **ElementRef**

```
export class AppComponent implements AfterViewInit {
  @ViewChild('nav')
  nav: ElementRef<HTMLElement>;

  constructor(private _renderer: Renderer2) {}

  ngAfterViewInit() {
    this._renderer.addClass(this.nav.nativeElement, 'foo-bar');
  }
}
```

Summary

academeez

- Avoid using browser only API - your code won’t work on Node
- Angular provides you with wrappers around browser only api so you can easily keep you code isomorphic.
- Other than angular wrapper around browser specific API another useful technique we will use to keep our code isomorphic is shimming which we will cover in the next lesson

academeez

Thank You

Next Lesson: 7. shimming