05

Para saber mais - ngOnInit x ionViewDidLoad

Nada nos impede de executarmos um código dentro do `constructor` de nossos componentes. Faz até sentido, pois ele é chamado assim que a classe do nosso componente é criada. No entanto, a boa prática é mover todo o código para dentro do método `ngOnInit`. A vantagem disso é que se o nosso componente precisar de algum dado passado através de uma inbound property (`@input`) teremos acesso ao seu valor, algo que não seria possível no construtor. Como não usamos até agora `@input` continuar com a inicialização no construtor não nos causaria problema. Além disso, o Angular disponibiliza uma interface chamada `OnInit` que podemos implementar de modo a garantir em tempo de codificação que a implementação do método `ngOnInit` está correta e também nos dá a possibilidade de utilizar o autocomplete da IDE utilizada.

Entretanto, como estamos num curso de Ionic, a boa prática é lançar mão dos recursos desse framework independentemente do que o Angular nos oferece por baixo dos panos. Por isso, nesse caso, demos preferência à utilização do `ionViewDidLoad`, evento do ciclo de vida do Ionic que é executado sempre que uma página é carregada e que pode ser sobrescrito, como fizemos no curso! A má notícia é que o Ionic não oferece uma interface que dê as mesmas garantias que o Angular dá no caso da interface `OnInit`. No entanto, essa situação pode facilmente ser superada pelos alunos mais atentos e que já fizeram o pré requisito de Angular, ao implementar uma interface que defina os métodos do ciclo de vida do Ionic, assim como feito na prática ao criarmos a interface `NavLifecycles`!

Para mais detalhes sobre o ciclo de vida do Ionic, acesse [esse ótimo post (https://blog.ionicframework.com/navigating-lifecycle-events/)](https://blog.ionicframework.com/navigating-lifecycle-events/) no blog do próprio Ionic!