Cheat Sheet – Forms

More about Angular 2 Forms

The Idea behind the new Forms Module:
https://docs.google.com/document/u/1/d/1RIezQqE4aEhBRmArIAS1mRIZtWFf6JxN_7B4meyWK0Y/pub
Official Documentation: https://angular.io/docs/ts/latest/guide/forms.html

Template-driven vs Data-driven

Angular 2 forms can be created via two ways: By using the template-driven approach or the data-driven approach.
In the first approach (template-driven), Angular 2 will automatically create a form with which it works by identifying the <form> tag.
You will have to assign controls to this form by adding the ngModel directive to the HTML elements which should be form inputs/ controls.

In the data-driven approach, you create the form on your own (in the component body) and then assign it to the HTML code by using formGroup for the form and formControlName for the individual controls. These directives will sync your template form (HTML code) and the form created in the component body.

Available Directives

ngModel	Registers controls in the template-driven approach. Exports controls via ngModel (#ctrl="ngModel") and may also be used with property binding to provide default values or two-way binding.
ngModelGroup	Allows you to group multiple controls together in the template-driven approach.
formGroup	Synchronizes a HTML form with your form created in code (data-driven approach). Prevents Angular 2 from creating the form automatically (which would be the template-driven approach).
formGroupName	Allows you to sync a FormGroup with a group of Inputs in your HTML code.
formControl	Syncs a FormControl in your code with an Input in your HTML code.
formControlName	Syncs a FormControl inside a FormGroup (e.g. the overall Form) with an Input in that FormGroup in the HTML code.

formArrayName	Syncs a FormArray in a FormGroup with an Array of Inputs in your HTML code.
ngSubmit	Allows you to execute code whenever the Form gets submitted.

Custom Validators

You may not only use built-in validators (overview: https://angular.io/docs/ts/latest/api/common/index/Validators-class.html) but also build your own ones.
You may create functions which can be passed as validators inside your component body, in static classes or as global functions.
Make sure to only pass a reference to the function and not execute it instead.

Useful Resources

Official Guide: https://angular.io/docs/ts/latest/guide/forms.html