

Project setup

Keep things together in **modules**. All pages, components, routes, validators, services that go with that module stay with that module. Pages are **container components**, they are children of the root component.

Only **services** get injected into container components, these services will have access to persisted data.

Other components will most likely be **pure components**, they get all their dependencies through @Input directives and emit all changes to their parent through @Output directives.

Storing the **routes** in the module (as a routing module), saves you time tracking it down whenever you need to make changes.

Project setup commands - sample

```
# When setting up a new project think about how
you want it to look. Make a short list of commands
to set up your project. Open the new project in
your IDE, if things do not feel right, adjust your
list and run it again. I have added an example
below.

ng new sample
ng g m core
ng g m core/modules/homepage
ng g c core/modules/homepage/containers/homepage
ng g m core/modules/products
ng g c core/modules/products/containers/product-
view
ng g c core/modules/products/containers/product-
edit
ng g c core/modules/products/containers/product-
add
ng g c core/modules/products/components/product-
form
ng g c core/modules/products/services/product
ng g m core/modules/contact
ng g c core/modules/contact/containers/contact
# Make many more so you get a good feel of how your
decisions will impact the project. Make changes,
delete the project and run you commands again.
```

Life cycle hooks

ngOnCh Respond when Angular (re)sets data-bound input proper-
anges() ties. The method receives a SimpleChanges object of
current and previous property values. Called before
ngOnInit() and whenever one or more data-bound input
properties change.

Life cycle hooks (cont)

ngOnInit() Initialize the directive/component after
Angular first displays the data-bound
properties and sets the directive/compo-
nent's input properties. Called once, after
the first ngOnChanges().

ngDoCheck() Detect and act upon changes that Angular
can't or won't detect on its own. Called
during every change detection run,
immediately after ngOnChanges() and
ngOnInit().

ngAfterContentInit() Respond after Angular projects external
content into the component's view / the
view that a directive is in. Called once after
the first ngDoCheck().

ngAfterViewInit() Respond after Angular initializes the
component's views and child views / the
view that a directive is in. Called once after
the first ngAfterContentChecked().

ngAfterViewChecked() Respond after Angular checks the compon-
ent's views and child views / the view that a
directive is in. Called after the ngAfterVi-
ewInit() and every subsequent ngAfterCo-
ntentChecked().

ngOnDestroy() Cleanup just before Angular destroys the
directive/component. Unsubscribe Observ-
ables and detach event handlers to avoid
memory leaks. Called just before Angular
destroys the directive/component.



By **Robert Broen** (rbroen)
cheatography.com/rbroen/

Not published yet.
Last updated 18th September, 2019.
Page 1 of 1.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>