

Generating Code from Blueprints



John Papa

PRINCIPAL DEVELOPER ADVOCATE

@john_papa www.johnpapa.net



The Angular CLI generates
code from blueprints



Generating Code from Blueprints



**Blueprints for most common features
e.g. Components, Services, NgModules**

TypeScript blueprints

Options

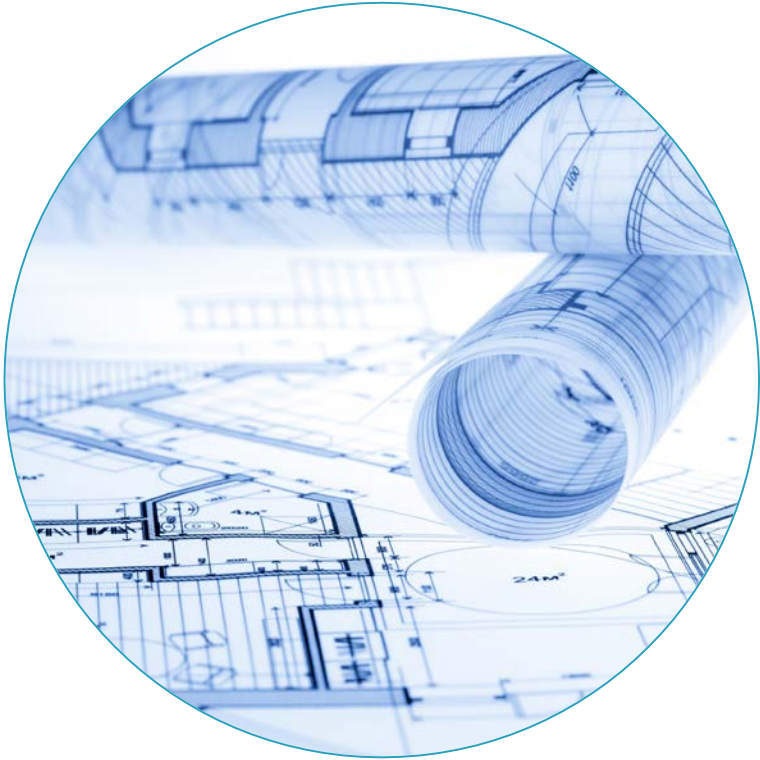
Aliases

Project configuration



The `ng generate <blueprint>` Command





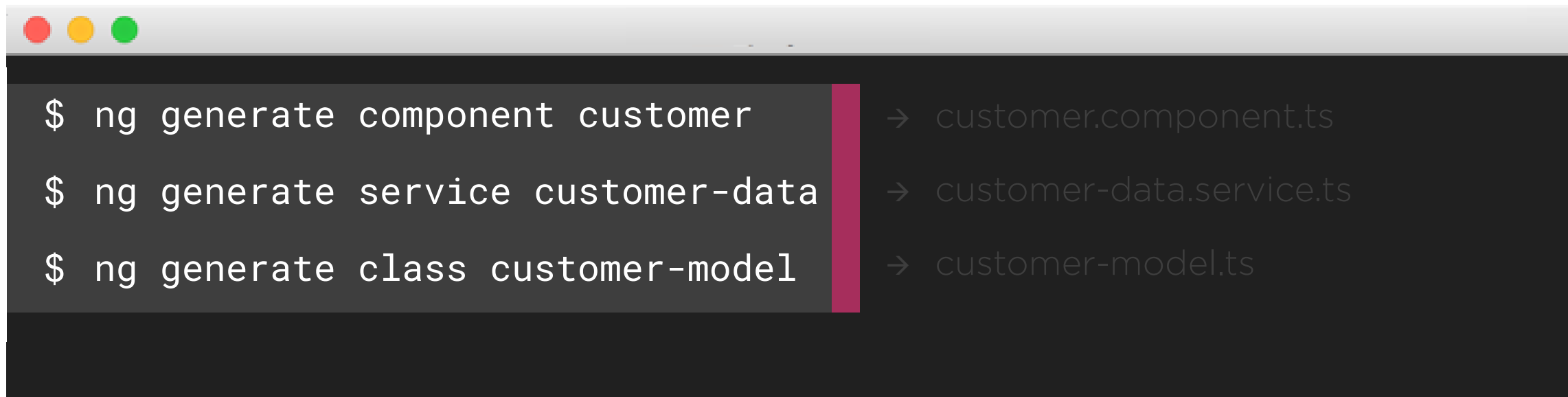
Blueprints

The Angular CLI generates from blueprints

Blueprints define the file(s)

We indicate a name and options





```
$ ng generate component customer      → customer.component.ts
$ ng generate service customer-data  → customer-data.service.ts
$ ng generate class customer-model   → customer-model.ts
```

Generating from Blueprints

The **generate** command will generate code from a blueprint

Syntax: **ng generate <blueprint> <options>**

Use the **--dry-run** option to practice



Components

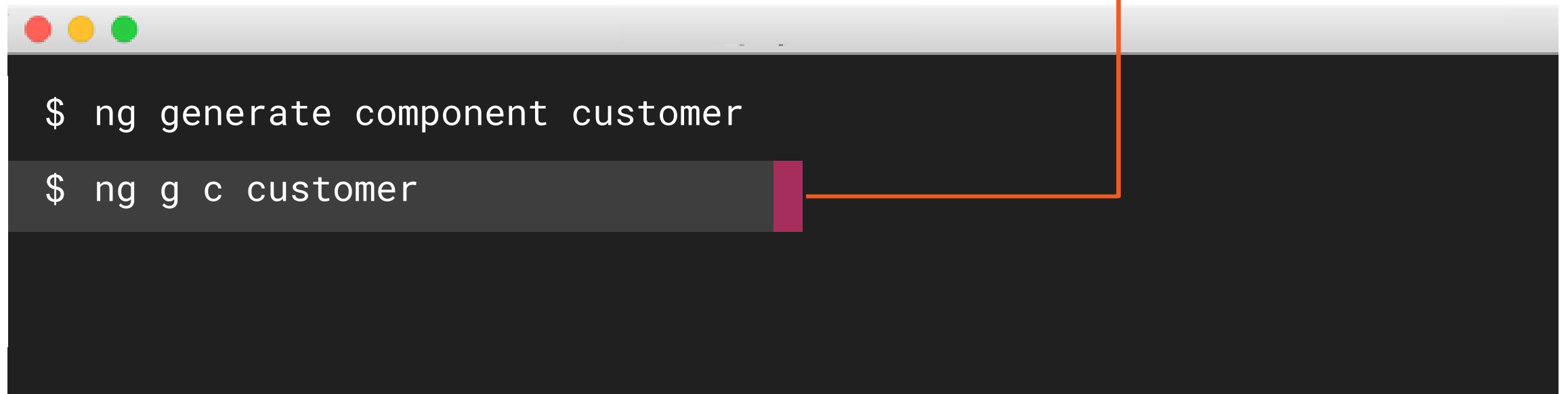


Aliases as Shortcuts

The **generate** command will generate code from a blueprint

Use the **g** alias as a shortcut for **generate**

Use the **c** alias as a shortcut for **component**



```
$ ng generate component customer  
$ ng g c customer
```



Common Component Blueprint Options

Options

Alias

Description

--flat

Should a folder be created?

--inline-template

-t

Will the template be in the .ts file?

--inline-style

-s

Will the style be in the .ts file?

--spec

Generate a spec?

--view-encapsulation

-v

View encapsulation strategy

--change-detection

-c

Change detection strategy

--dry-run

-d

Report the files, don't write them



Common Ways to Generate Components



```
$ ng generate component pet
```

```
$ ng generate component pet --inline-template --inline-style
```



```
$ ng g c pet
```

```
$ ng g c pet -t -s
```

Aliases makes efficient shortcuts

```
$ ng g c pet -ts
```

We can stack the aliases



Commonly Used Commands

```
$ ng g c pet --flat
```

Don't create a /pet folder

```
--inline-template
```

Put the template in the .ts file

```
--inline-style
```

Put styles in the .ts file

```
--prefix my
```

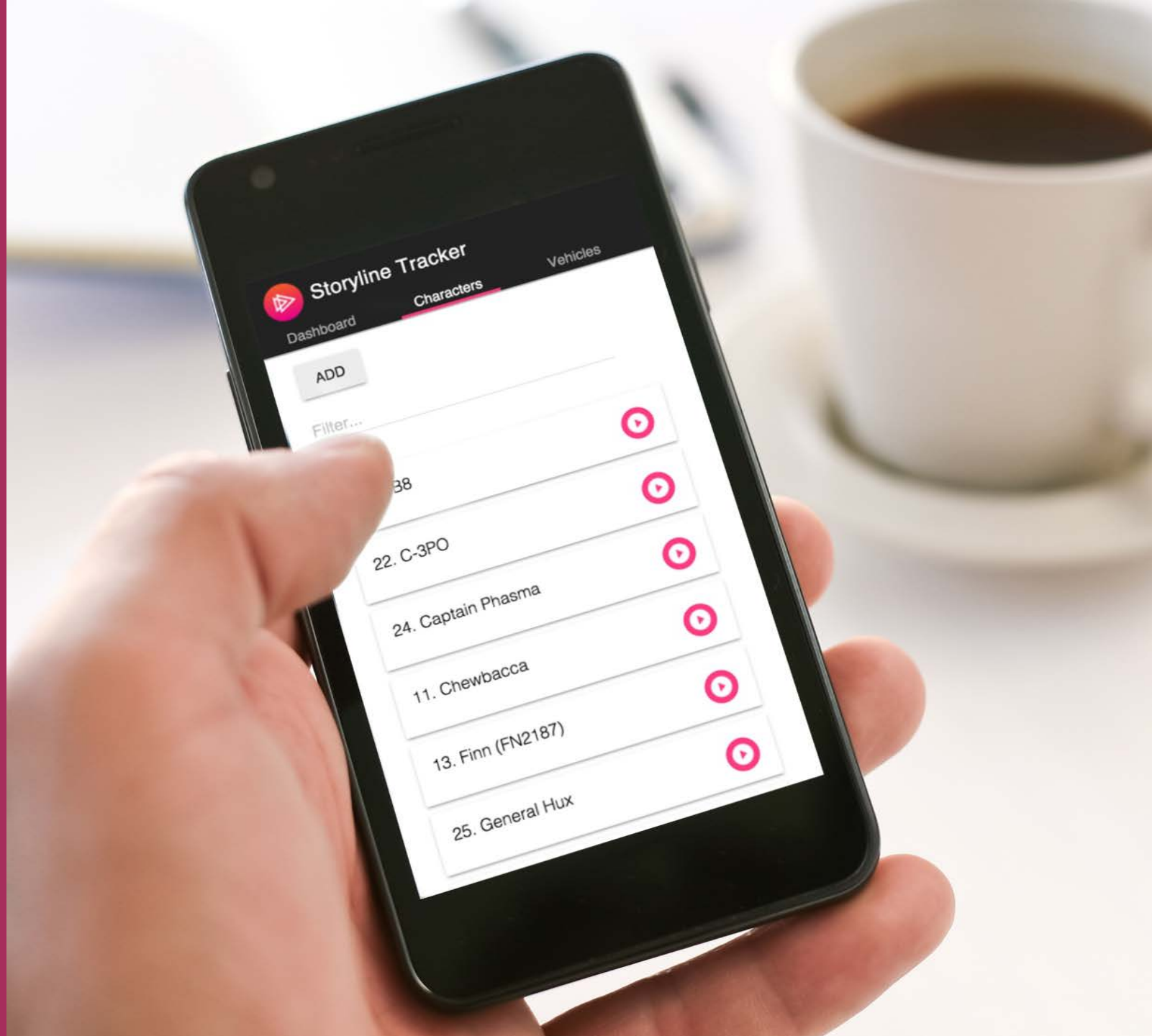
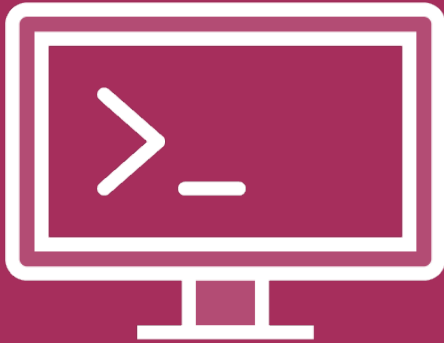
Prefix the component with my-

```
ng g c pet -st --flat --prefix my
```

Combine aliases



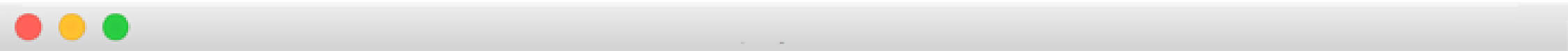
Demo



Out-of-the-box Blueprints

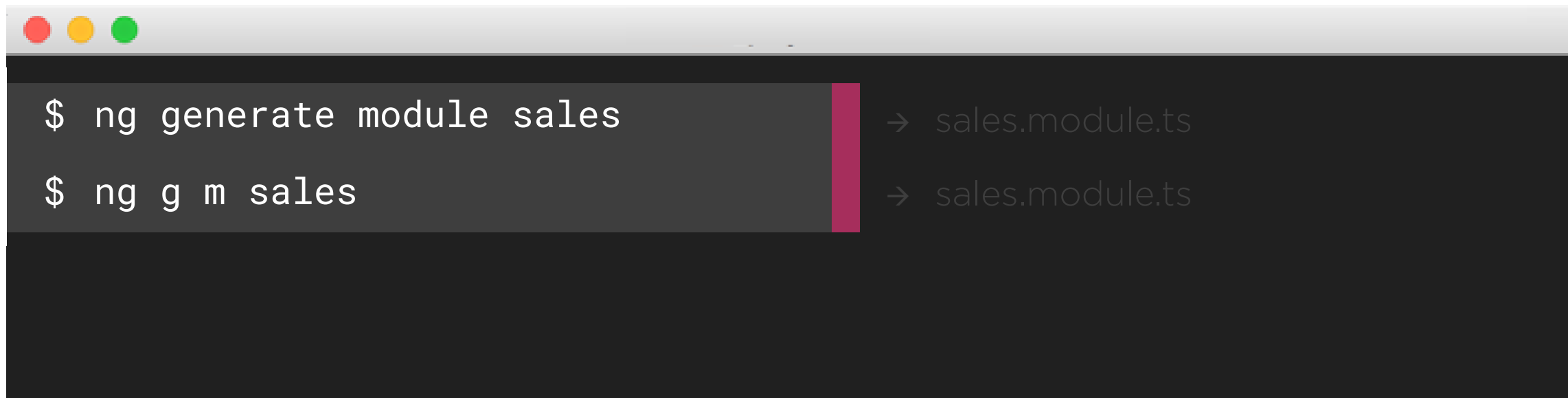


More Blueprints



\$ ng g directive search	Alias d
\$ ng g service customer-data	Alias s
\$ ng g pipe init-caps	Alias p
\$ ng g class customer-model	Alias cl
\$ ng g interface orders	Alias i
\$ ng g enum gender	Alias e



A terminal window with a dark background and a light gray title bar. The title bar has three colored window control buttons (red, yellow, green) on the left. The terminal shows two commands entered: '\$ ng generate module sales' and '\$ ng g m sales'. To the right of each command, the output is shown in a lighter gray font: '→ sales.module.ts'. A vertical pink bar is positioned between the commands and their outputs.

```
$ ng generate module sales → sales.module.ts
$ ng g m sales             → sales.module.ts
```

NgModule BluePrints

Use **ng generate module <options>**

No spec by default

Consider if we want routing



Getting Help

Add the **--help** flag



```
$ ng --help
```

A terminal window with a dark background and a light gray title bar. The title bar has three colored window control buttons (red, yellow, green) on the left. The terminal displays the command `$ ng --help` in white text.

Generating Code from Blueprints



Use the out-of-the-box blueprints

Lean on `ng --help` and the docs

Use aliases for blueprints and options

Practice with `--dry-run`

