ANGULAR

Angular Workshop

MVPMIX
2019-03-07

Stephen Fluin
@StephenFluin

Agenda

1. Setup Verification
2. Template Syntax
3. Component Composition
4. Services
5. Routing
6. Data
7. Deployment

Slides:

http://bit.ly/angular-mvpmix

Setup Verification

Making sure we all start on the same page

Slides:

http://bit.ly/angular-mvpmix

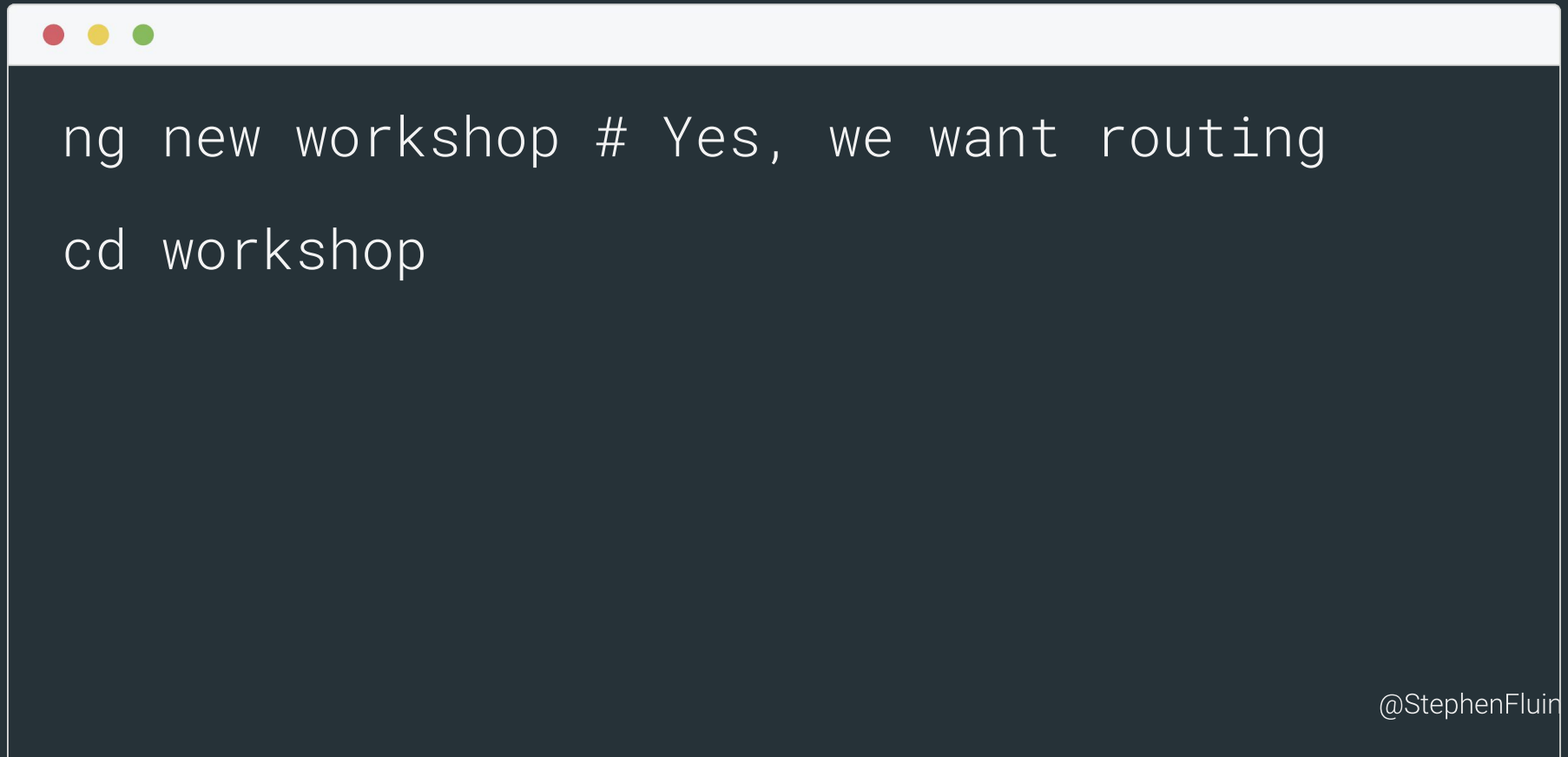
About Package Managers

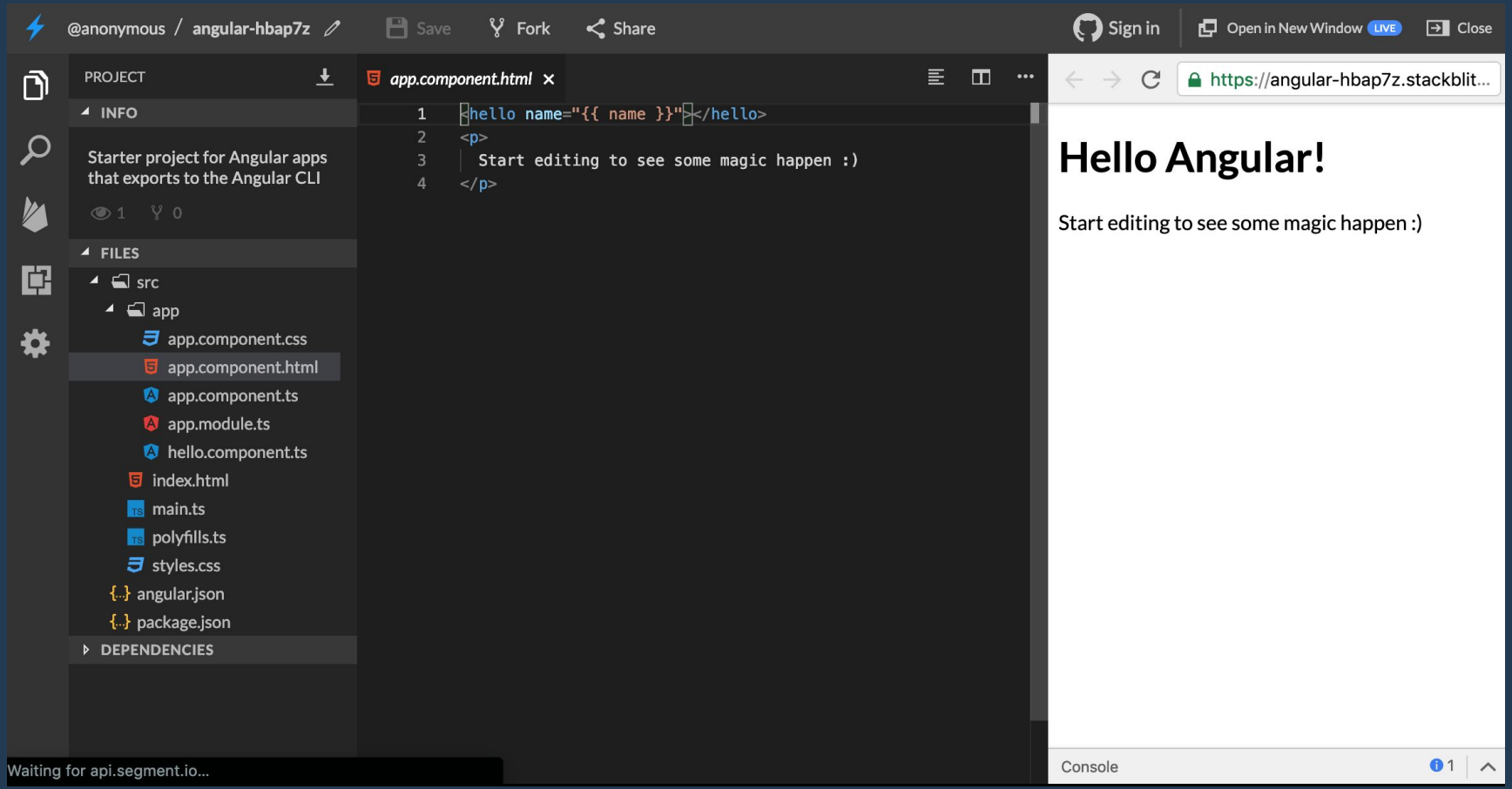
- npm
- yarn

Setup Verification

- node
- npm or yarn
- @angular/cli

Create an application





StackBlitz.com

@StephenFluin

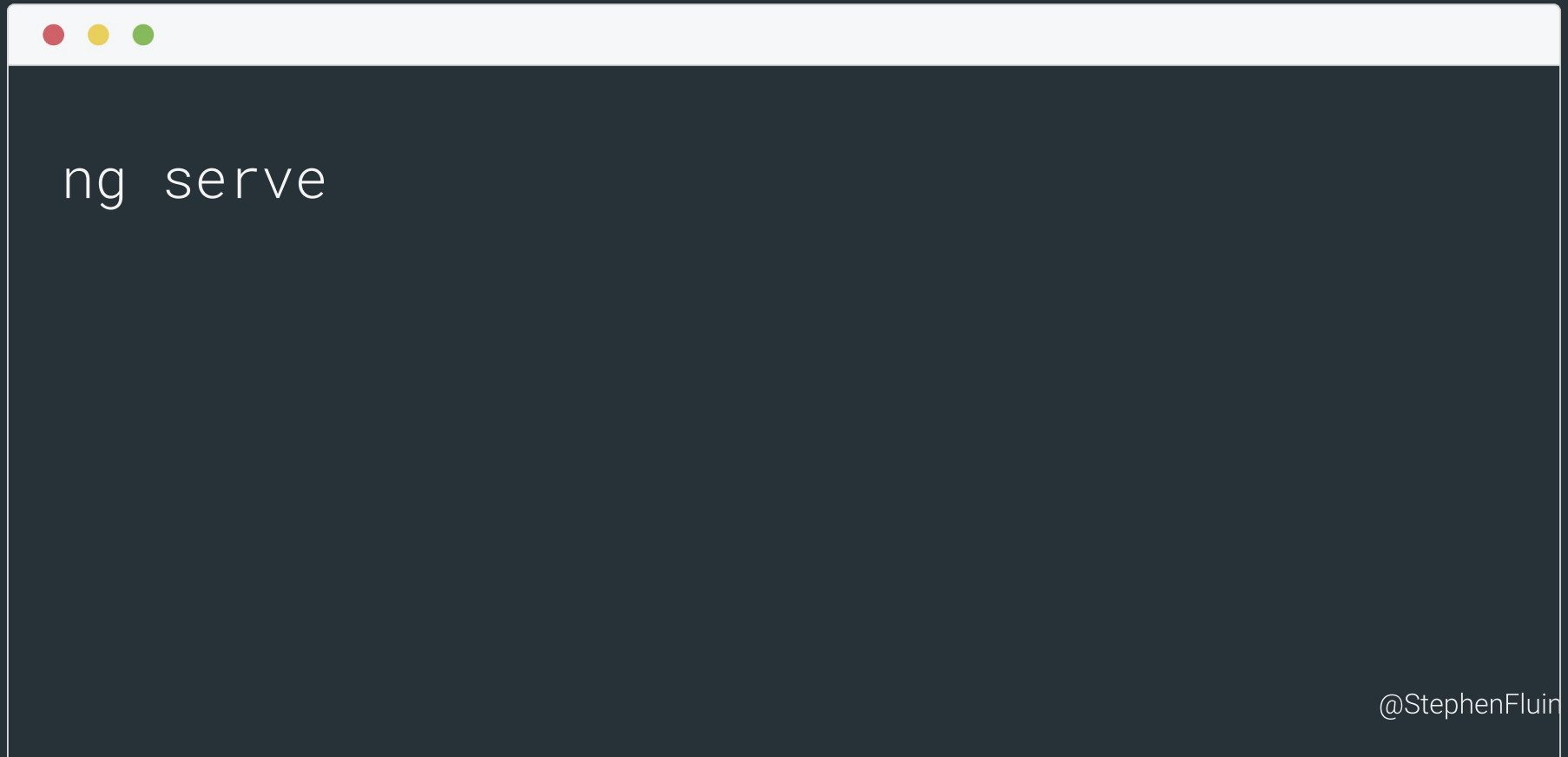
Setup Verification

```
ng version
```

```
stephenfluin-macbookpro:workshop stephenfluin$ ng version
```

```
Angular CLI: 7.3.5
Node: 10.15.1
OS: darwin x64
Angular: 7.2.8
... animations, common, compiler, compiler-cli, core, forms
... language-service, platform-browser, platform-browser-dynamic
... router

Package                                  Version
------------------------------------------------------------
@angular-devkit/architect                0.13.5
@angular-devkit/build-angular            0.13.5
@angular-devkit/build-optimizer          0.13.5
@angular-devkit/build-webpack            0.13.5
@angular-devkit/core                     7.3.5
@angular-devkit/schematics               7.3.5
@angular/cli                             7.3.5
@ngtools/webpack                         7.3.5
@schematics/angular                      7.3.5
@schematics/update                       0.13.5
rxjs                                     6.3.3
typescript                               3.2.4
webpack                                  4.29.0
```

Setup Verification

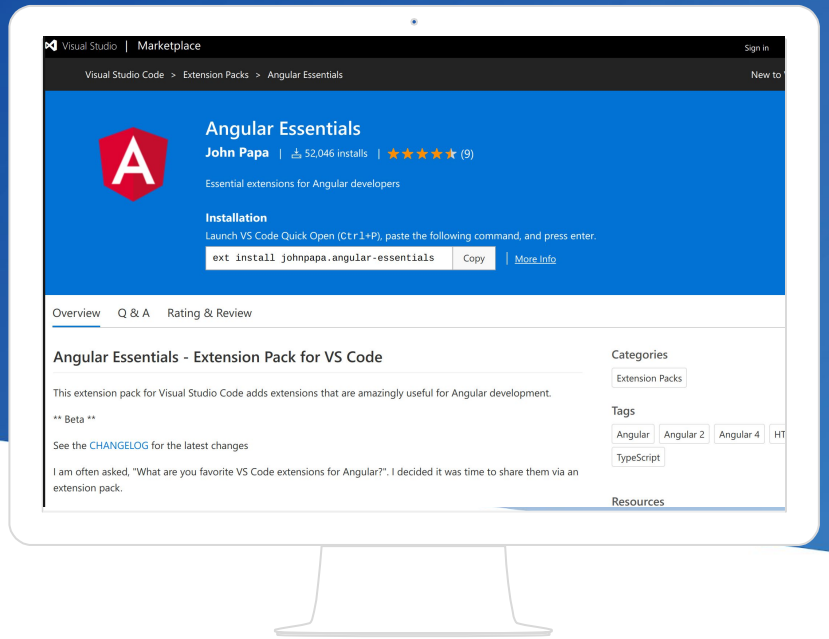


Welcome to app!

Here are some links to help you start:

- Tour of Heroes
- CLI Documentation
- Angular blog

ANGULAR



Angular Essentials

http://bit.ly/vscode-papa
or search in VS code

Angular Console

nrwl.angular-console

nrwl | 16,458 | ★★★★½ | Repository

Angular Console for Visual Studio Code. The user interface app for the Angular CLI

Disable Uninstall

Details Contributions

The UI for the Angular CLI

package.json

```
"prettier": {
  "trailingComma": "es5",
  "tabWidth": 2,
  "singleQuote": true,
  "printWidth": 80
}
```

Component Model

Google Express

Search Google Express

- Shopping in 94560
- Stores
- Departments
  - Household Supplies
  - Home & Garden
  - Grocery
  - Electronics
  - Health & Beauty
  - MORE
- Shopping list
- Quick reorder
- Orders
- Help
- Settings

Shop by department


Household Supplies


Home & Garden

Grocery


Electroni

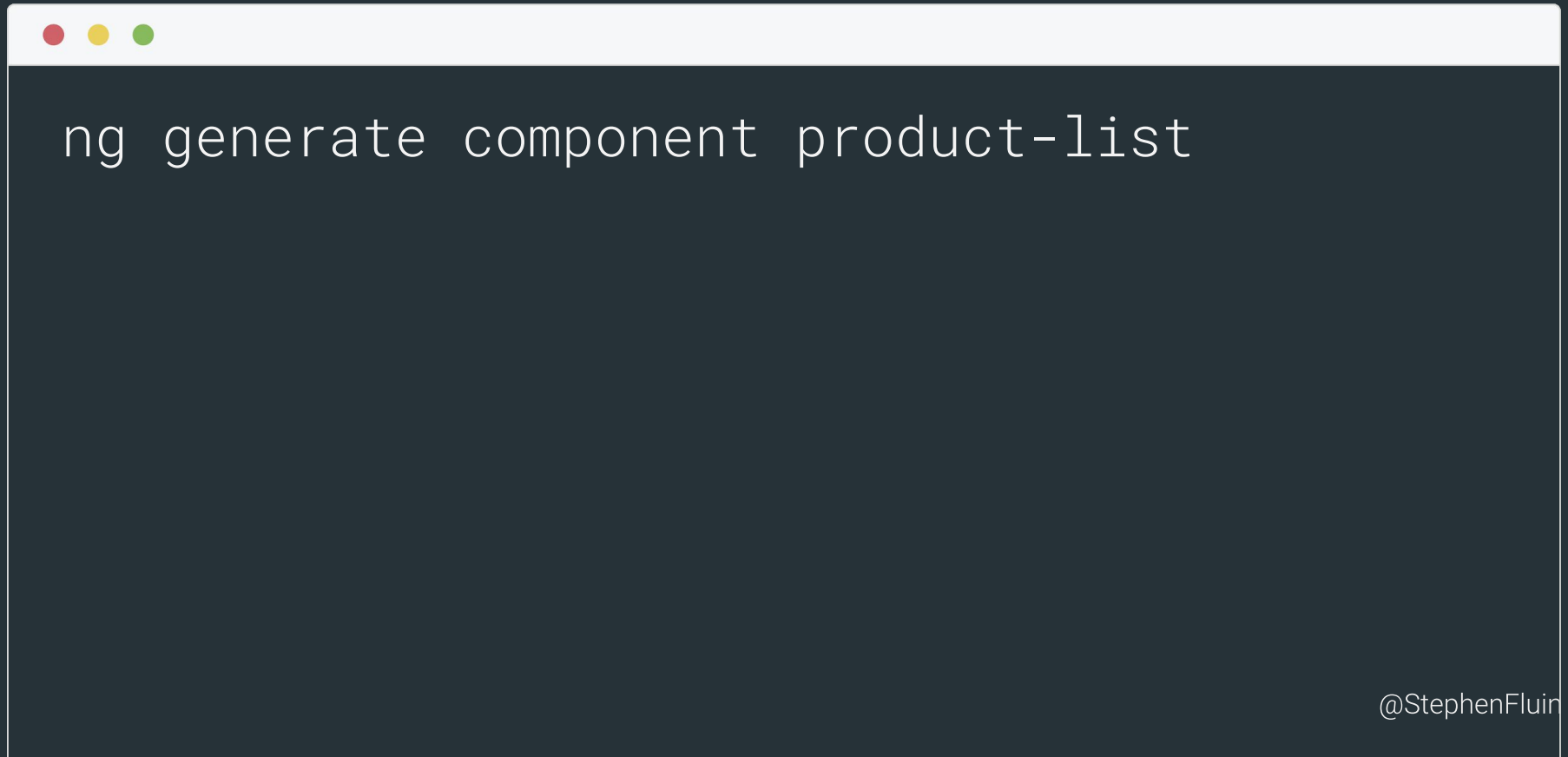
Explore digital cameras

iOS App · Android App

Our App Setup

Create and show a product list

Generate a component



app.component.html

```
<app-product-list></app-product-list>
```

product-list.component.ts

```
export const products = [
 {name: 'Item One',price: '$4.99',description: 'This is a great item for your kids!',coupon: '10',date:
'2019-03-08',},
 {name: 'Item Two',price: '$41.99',description: 'This is a great item for your parents!',date:
'2019-03-08',},
 {name: 'Item Three',price: '$39.99',description: 'This is a great item for yourself!',date: '2019-03-08',},
 {name: 'Item Four',price: '$12.99',description: 'The thing you have always wanted',date: '2019-03-08',},
 {name: 'Item Five',price: '$1.98',description: 'I cannot believe this is so cheap!', date: '2019-03-08'},
];


…
export class ProductListComponent {
 products = products;
…
}
```

Setup Steps

- Create a product list component
- Show the product list component
- Add some data

Template Syntax

- Interpolation
- Property Binding
- Event Binding
- *ngIf
- *ngFor

First 5 Template Syntaxes

{{ variableName }} - **Interpolation**

[propertyName]=”variableName” - **Property Binding**

(event)=”doSomething()” - **Event Binding**

*ngIf=”variableName” - ***ngIf**

*ngFor=”let item of list” - ***ngFor**

product-list.component.ts

```
<div *ngFor="let product of products">
  {{product.name}}
</div>
```

Render Product Coupons

```
<div *ngFor="let product of products">
  {{product.name}}
  <div *ngIf="product.coupon">{{product.coupon}} % off</div>
</div>
```

Property Binding

```
<div *ngIf="product.coupon" [title]="product.coupon + '%'">
  {{product.coupon}} % off
</div>
```

Event Binding

```
<div (click)="greatDeal(product)">?</div>
```

Add the event to product-list.component.ts

```
greatDeal(product) {
  alert("What a great deal!");
}
```

Now you try

- *ngFor through all of the items
- {{interpolate}} the information
- *ngIf to conditionally show a coupon
- add [title] binding
- add a (click) event
- optional - add CSS

Component Composition

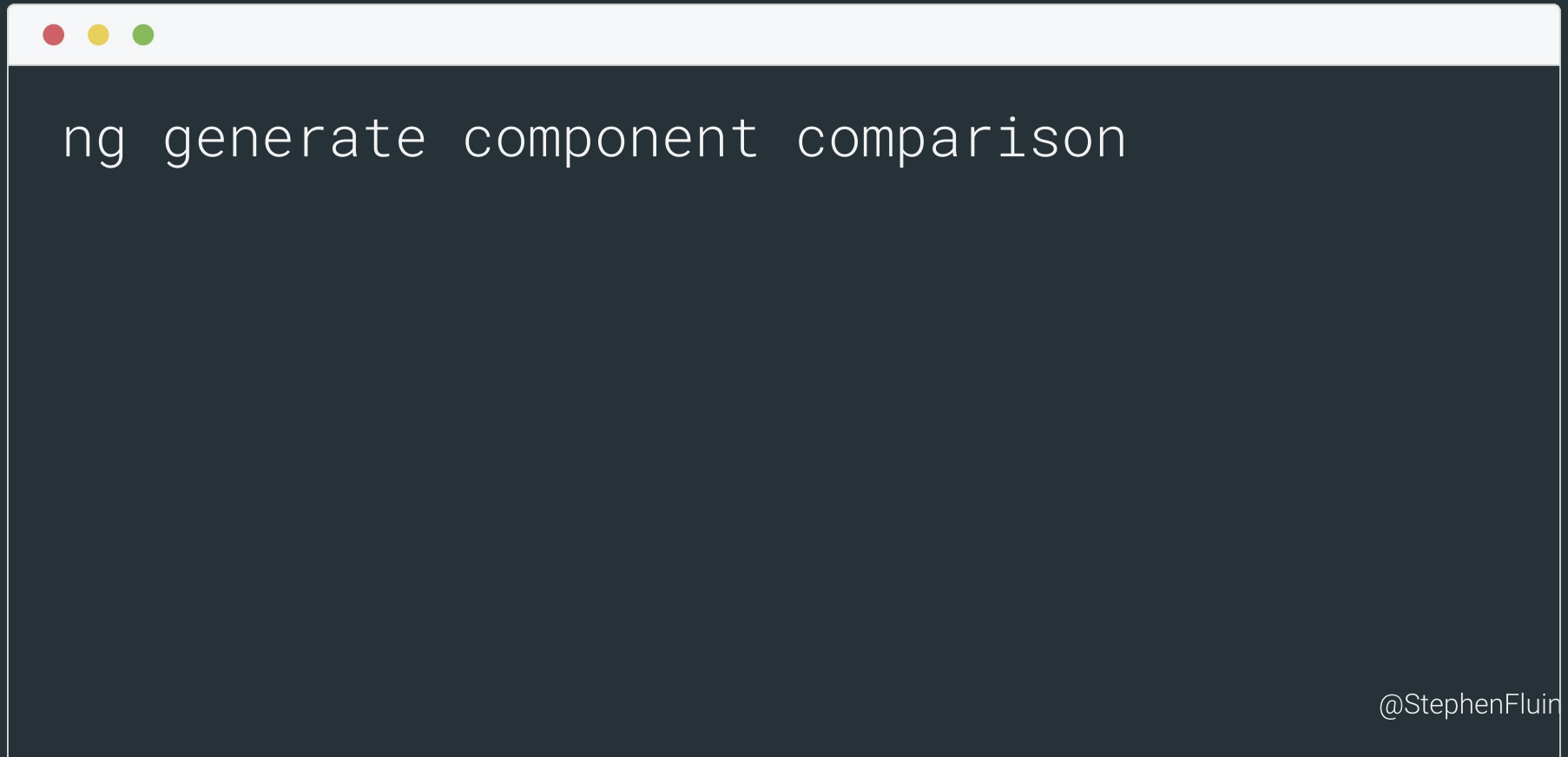
@Input

@Output

Just like DOM

- Properties / Attributes Down
- Events Up

Create a comparison component



Inputs

Take in a property from your parent

comparison.component.ts

```
import { Input, Output, EventEmitter } from '@angular/core';
…

export class ComparisonComponent {
  @Input() product;
…

}
```

comparison.component.html

```
<div *ngIf="product.price == '$41.99'">
  You can save if you buy this in bulk!
</div>
```

product-list.component.html

```
<div *ngFor="let product of products">
  {{product.name}}
  <div *ngIf="product.coupon" [title]="product.coupon + '%'">
    {{product.coupon}} % off
    </div>
    <div (click)="greatDeal(product)">?</div>
    <app-comparison [product]="product"></app-comparison>
</div>
```

Outputs

Emit an arbitrary event your parent can listen to

comparison.component.ts

```
export class ComparisonComponent {
  @Input() product;
  @Output() interest = new EventEmitter();
...
}
```

comparison.component.html

```
<div *ngIf="product.price == '$41.99'" (mouseover)="interest.emit()">
  You can save if you buy this in bulk!
</div>
```

product-list.component.html

```
<div *ngFor="let product of products">
  {{product.name}}
  <div *ngIf="product.coupon" [title]="product.coupon + '%'">
    {{product.coupon}} % off
    </div>
    <div (click)="greatDeal(product)">?</div>
    <app-comparison [product]="product"
      (interest)="logInterest($event)"></app-comparison>
</div>
```

product-list.component.ts

```
logInterest($event) {
  console.log("interest shown!");
}
```

Now you try

- Create a Comparison Component
- Add an @Input()
- Do something with the property
- Add an @Output()
- Listen for events at the parent level and log

Services

Shared logic & data

Add a cart service

```
ng generate service cart
```

Providers

Providers are the point of instantiation

Services can instantiate themselves, or Components or Modules can instantiate them.

Add property and methods

```
import { Injectable } from '@angular/core';

@Injectable()
export class CartService {
  cart = [];

  constructor() { }
  add(product) {
    this.cart.push(product);
  }
}
```

Inject the service in product-list.component.ts

```
import { CartService } from './cart.service.ts';

constructor(public cart: CartService) {
...
}
```

TypeScript Parameter Visibility

none - only visible in this method

private - only visible to this class

public - visible to this class, template, and other classes

Buy button in product-list.component.html

```
<button (click)="cart.add(product)">Buy</button>
```

Now you try

- Create a cart service
- Add a cart array
- Add an add method
- Inject the service
- Wire up a buy button

Break

Routing

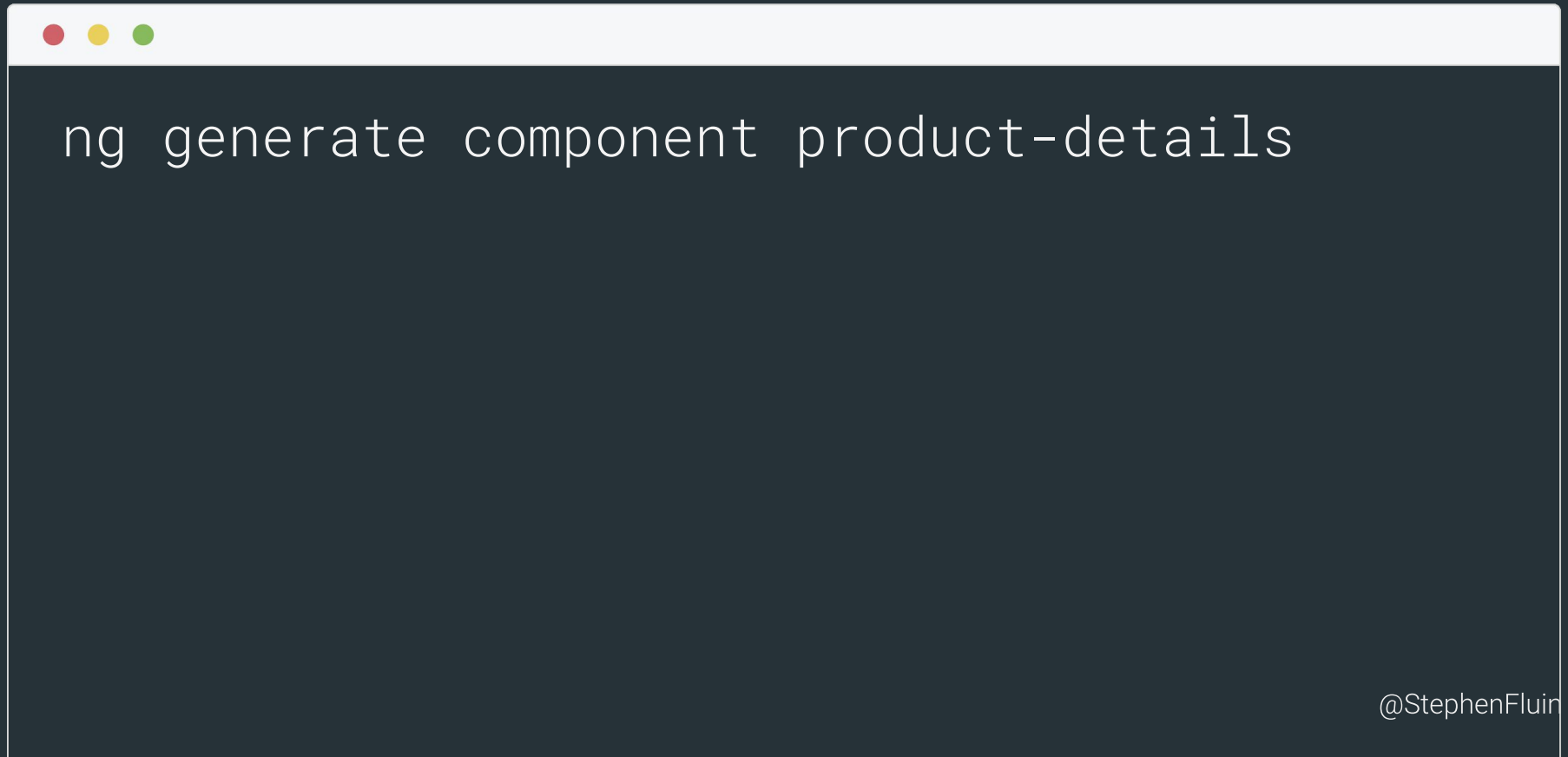
Activated Route

Router Outlets

Routing Examples

```
{ path: '', component: HomePageComponent },
{ path: 'about', component: AboutPageComponent },
{ path: 'products/:productId', component: ProductDetailsComponent },
{ path: '**', component: PageNotFoundComponent }
```

Create Product Details



Inject ActivatedRoute

```
import { ActivatedRoute } from '@angular/router'
...
constructor(route: ActivatedRoute) {

}
```

Route config

Lives in app-routing.module.ts.

Don't forget to import your components!

Add a route for our product details to our route config

```
[
   {path: '', component: ProductListComponent},
   {path: 'products/:id', component: ProductDetailsComponent}
 ]
```

Use the router in app.component.html

```
<router-outlet></router-outlet>
```

Get the Route Parameter

```
import { products } from
'../product-list/product-list.component';

...

  product;

  constructor(route: ActivatedRoute) {
    this.product = products[route.snapshot.params.id]
   }
```

All work the same in templates

Any property on the class will work the same, regardless of source:

- Inputs
- Dependency Injection
- Route Parameters (if saved locally)

Show product details

```
<div>{{product.name}}</div>
<div>{{product.description}}</div>
```

Create Navigation in app.component.ts

```
<div><a routerLink="/">Home</a></div>
<router-outlet></router-outlet>
```

Create Navigation in product-list.component.ts

```
<div *ngFor="let product of products; index as i">
  <a [routerLink]="['/products',i]">...</a>
</div>
```

Now you try

- Create a product details component
- Inject the router
- Setup the route config
- Add links for navigation

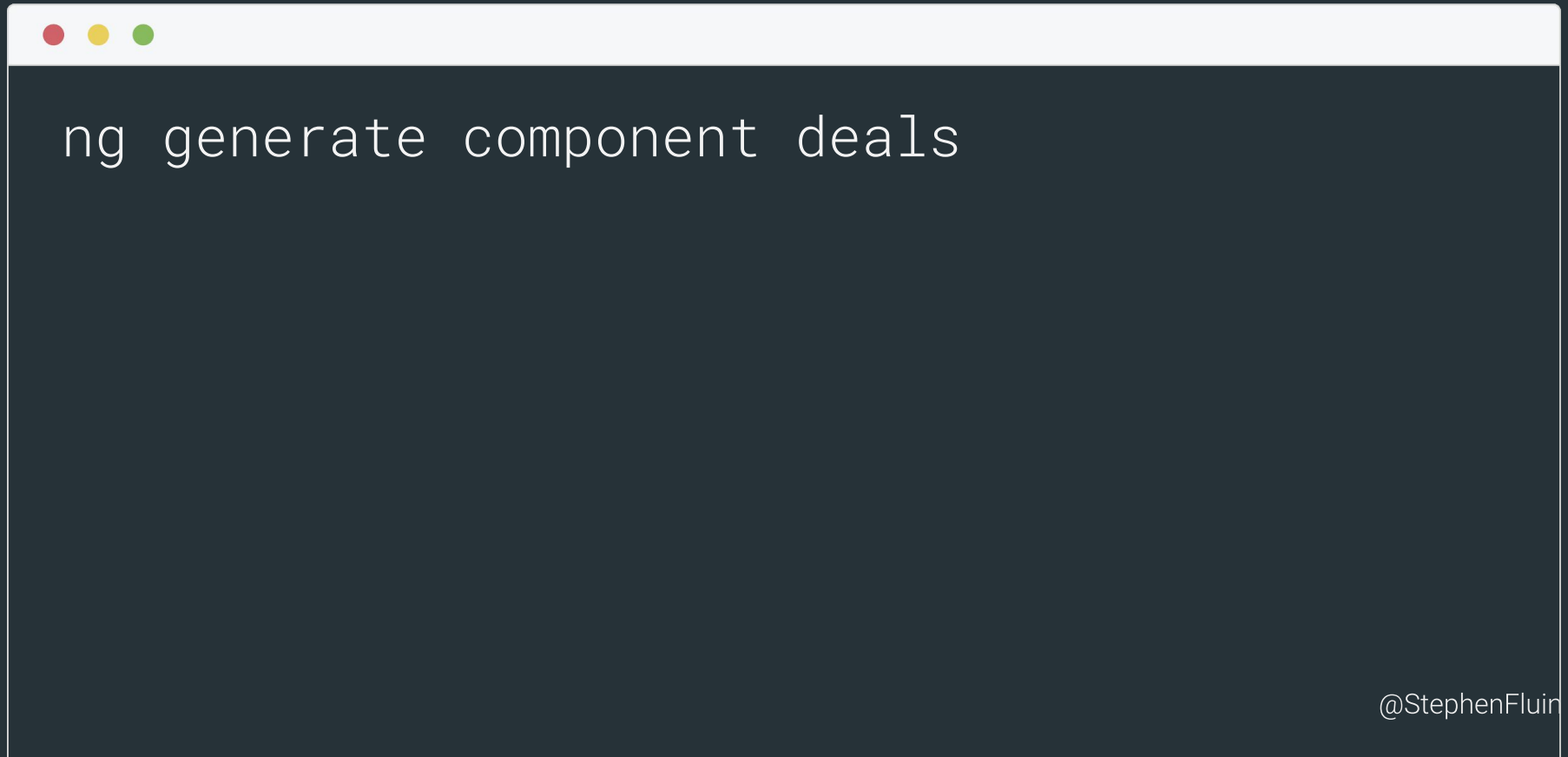
Data

HTTP

/assets/deals.json

```
[
 {"name": "Buy one get one free"},
 {"name": "Prices are lower today for this workshop"}
]
```

Create a deals component



Route to it

Route Config:

```
{path: 'deals', component: DealsComponent}
```

app.component.html:

```
<div><a routerLink="/deals">Deals</a></div>
```

app.module.ts

```
import { HttpClientModule } from '@angular/common/http';

…
imports: [..., HttpClientModule]
…
```

Inject the service in deals.component.ts

```
import { HttpClient } from '@angular/common/http';

constructor(public http: HttpClient) {
...
}
```

Inject the service in deals.component.ts

```
import { HttpClient } from '@angular/common/http';

dealList;
constructor(public http: HttpClient) {
  this.dealList = http.get('/assets/deals.json');
}
```

deals.component.html

```
<div *ngFor="let deal of dealList | async">
  {{deal.name}}
</div>
```

Now you try

- Create a deals.json file and new component
- Import HttpClient into your module
- Inject HttpClient into your component
- Define the stream with `http.get`
- Render the stream with `| async`

Streams are really powerful

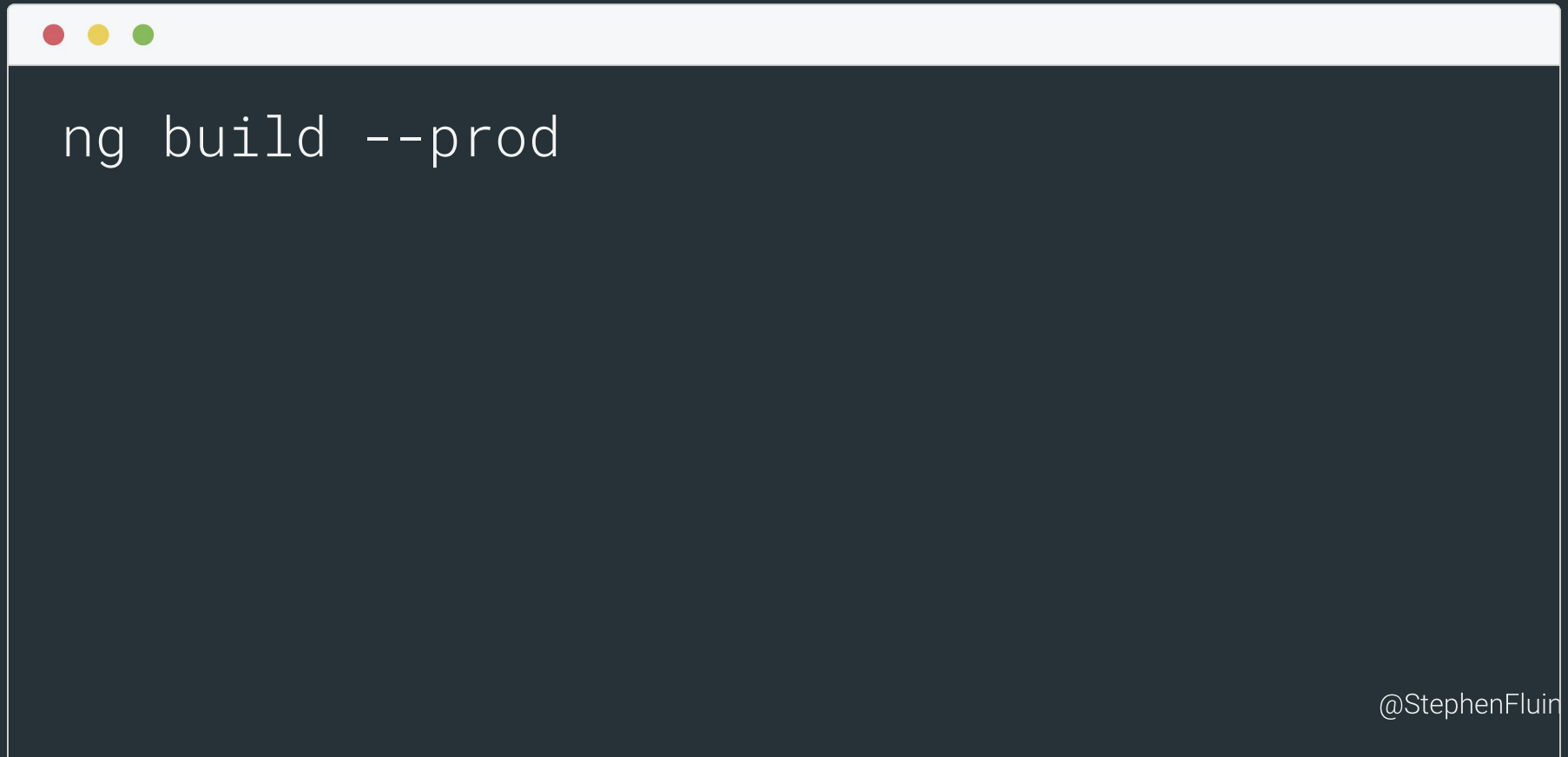
- Automatically Retry
- Cache Results
- Modify the data to meet your needs
- Fetch multiple data from multiple endpoints

https://github.com/StephenFluin/2019-03-angular-workshop

Deployment

What good is an app if users can't see it?

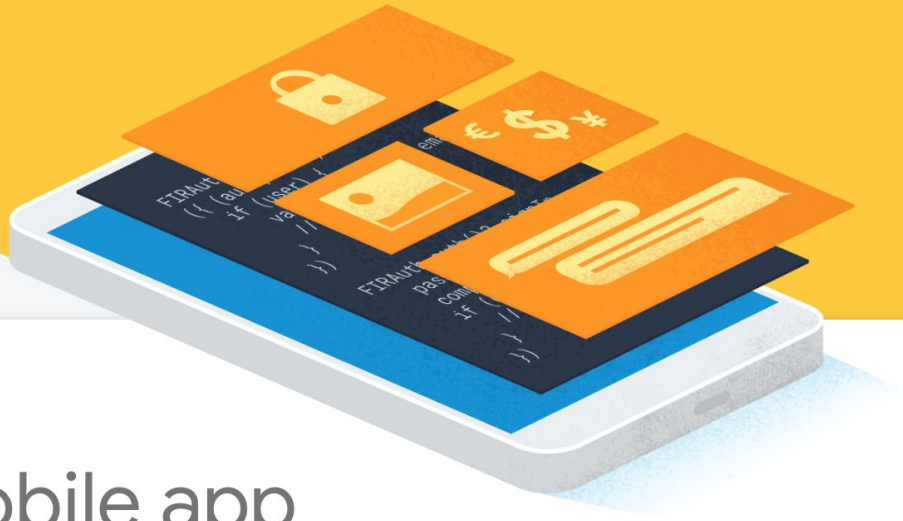
Generating production files



These files are portable

You can host them anywhere

Firebase Products Use Cases Pricing Docs Support Search GO TO CONSOLE



Firebase helps mobile app teams succeed.

GET STARTED WATCH THE VIDEO

Firebase Deploy

```
yarn global add firebase-tools

firebase login

firebase init # dist/workshop

firebase deploy
```

Thank You!

Stephen Fluin
@StephenFluin

Survey: bit.ly/stephen-survey

Advanced Agenda

1. Setup Verification
2. Router Lazy Loading
3. CLI Settings
4. RxJS
5. Service Worker
6. Universal (maybe)

Slides:

http://bit.ly/angular-mvpmix

Router Lazy Loading

Making sure we all start on the same page

Router Lazy Loading

- add routing to our app module
- create a synchronous **home** component
- create a lazy **about** module

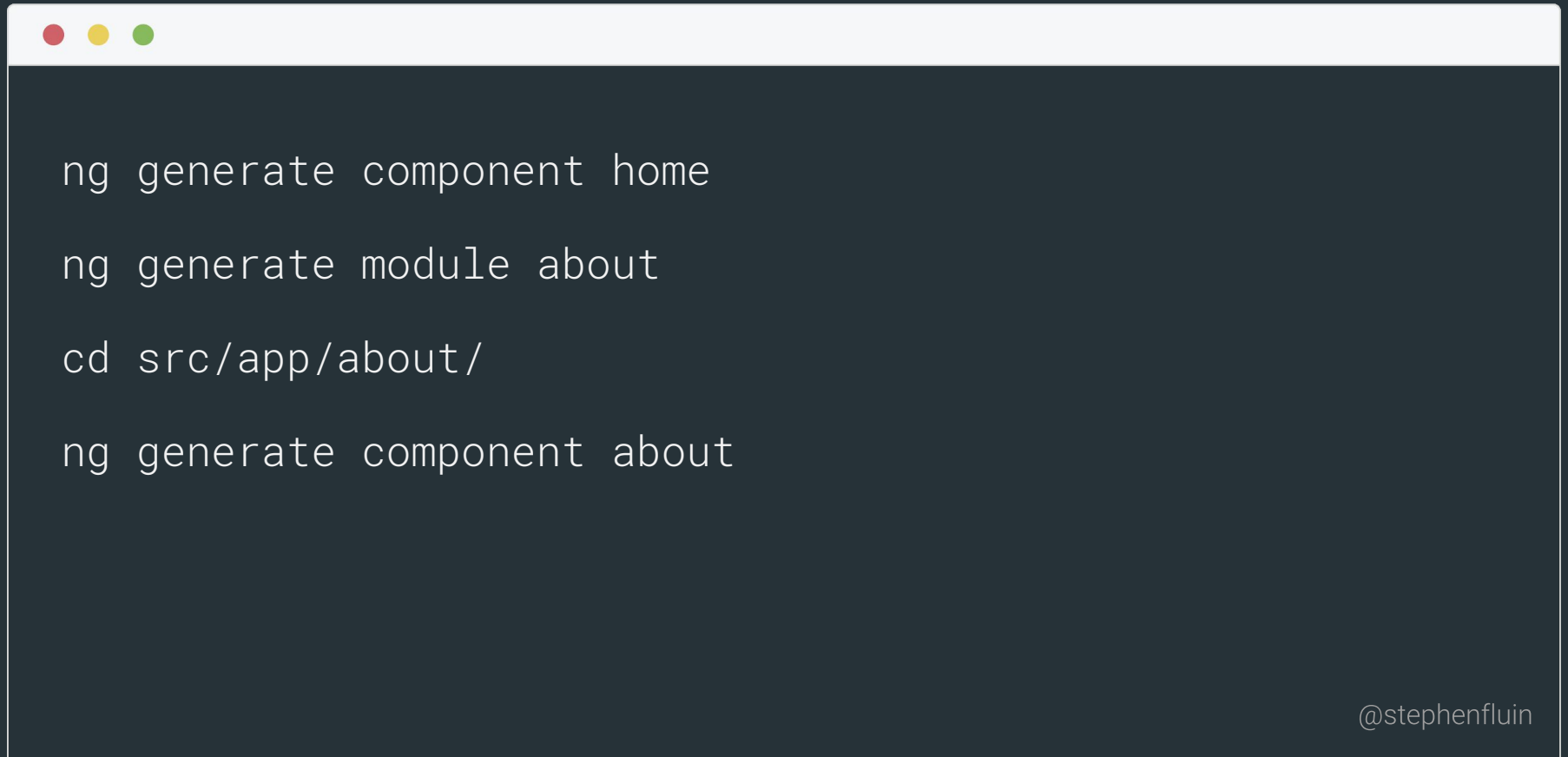
Routing in app.module.ts

```
import { RouterModule } from '@angular/router';

@NgModule({
...
  imports: [
    BrowserModule,
    RouterModule.forRoot([])
  ],
 ...
})
```

Routing in app.component.html

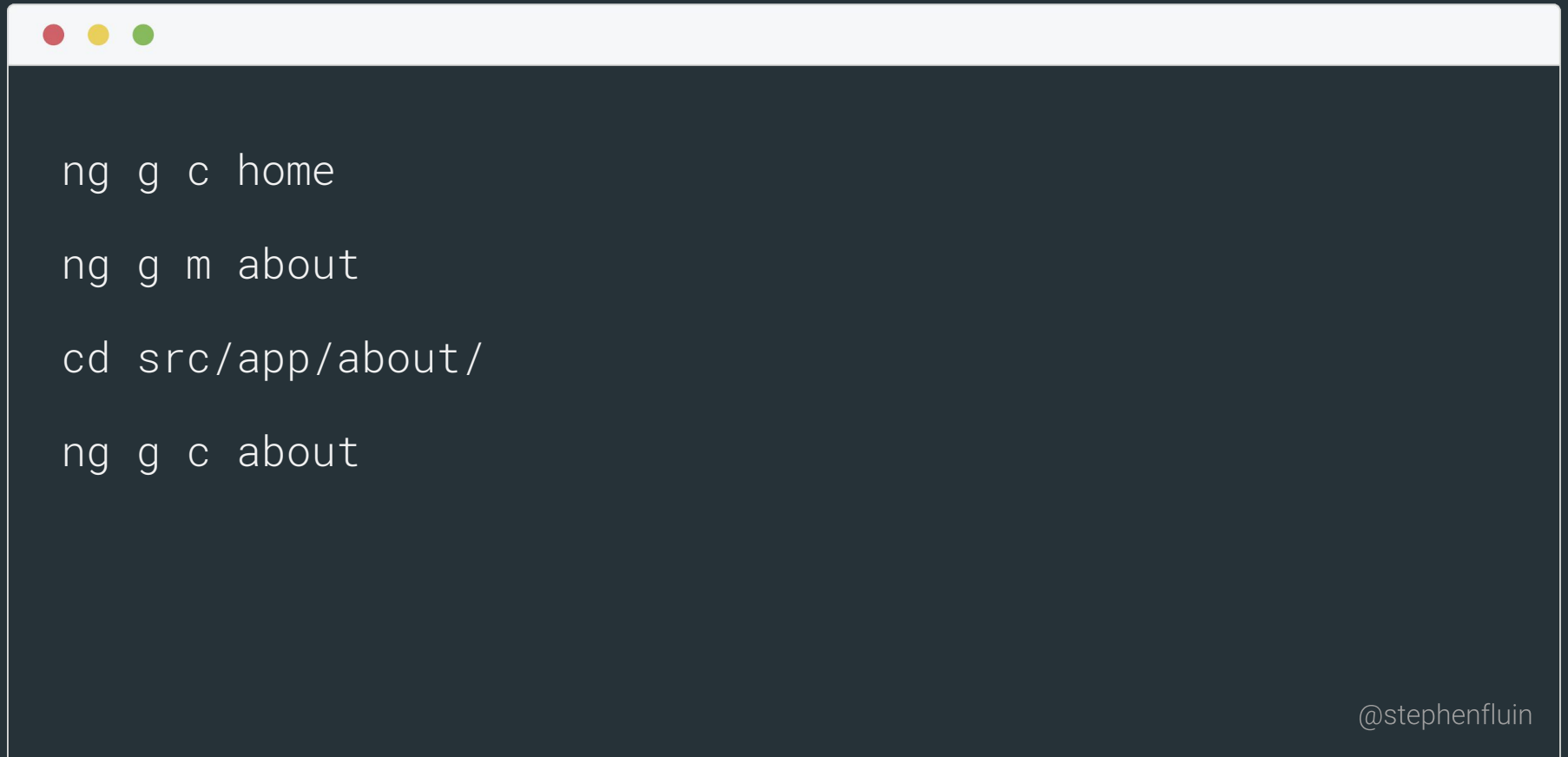
```
<h1>Advanced Angular</h1>
<router-outlet></router-outlet>
```

Home Route & About Module



```
ng generate component home

ng generate module about

cd src/app/about/

ng generate component about
```

Home Route & About Module



Add some links to app.component.html

```
<h1>Advanced Angular</h1>
<div>
  <a routerLink="/">Home</a>
  <a routerLink="/about">About</a>
</div>
<router-outlet></router-outlet>
```

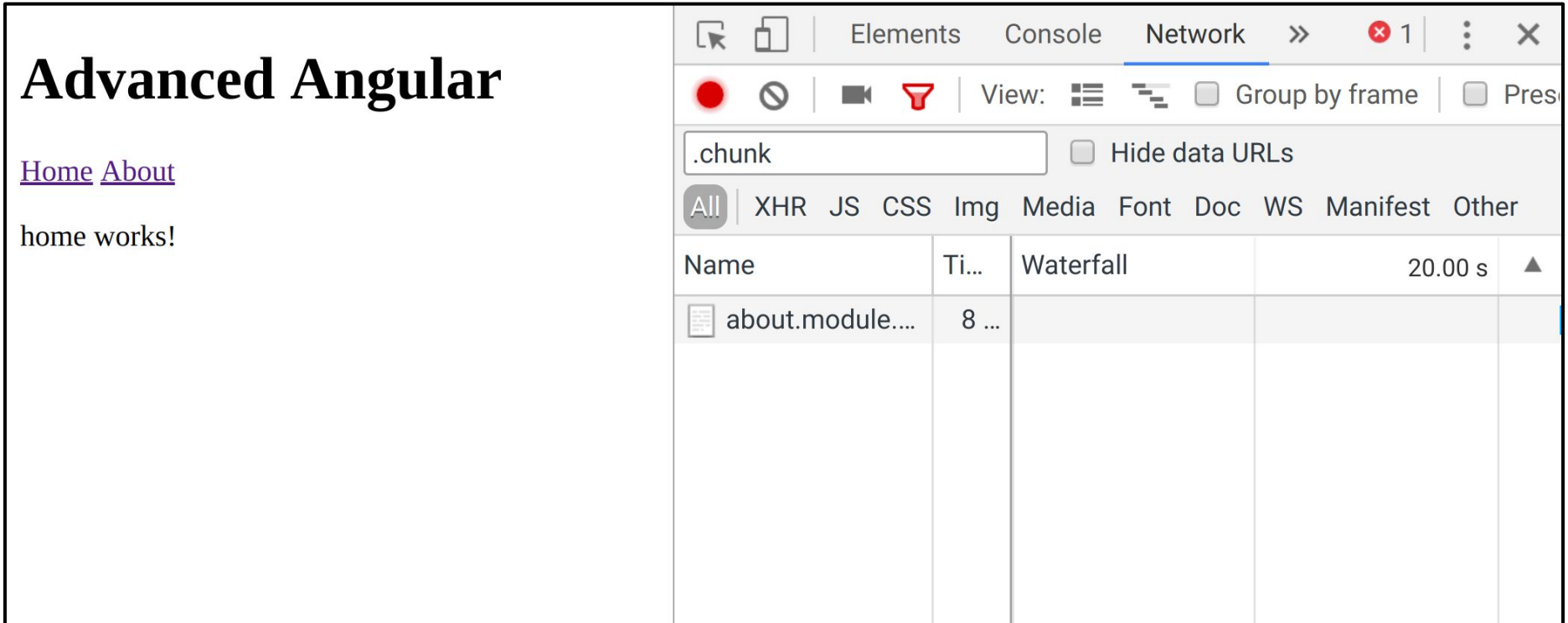
Root Routing

```
RouterModule.forRoot([
  { path: '', component: HomeComponent },
  { path: 'about', loadChildren: './about/about.module#AboutModule },
])
```

Routing in about.module.ts

```
import { RouterModule } from '@angular/router';
@NgModule({
  imports: [
    CommonModule,
    RouterModule.forChild([
      {path: '', component: AboutComponent}
    ])
  ],
  declarations: [AboutComponent]
})
```

Lazy Loading!



Let's Try It

Ask a neighbor, or raise your hand if you get stuck or have questions

At Home: Preload Routes

```
.forRoot[routes, { preloadingStrategy: PreloadAllModules }]
```

At Home: Custom Preload Strategies

```
import { PreloadingStrategy, Route } from '@angular/router';

import { Observable } from 'rxjs/Observable';
import { of } from 'rxjs/observable/of';

export class ConfigBasedStrategy implements PreloadingStrategy {
  preload(route: Route, load: Function): Observable<any> {
    return route.data && route.data.preload ? load() : of(null);
  }
}
```

@stephenfluin

Custom Preload Strategy

```
 RouterModule.forRoot([
...
      {
        path: 'about',
        loadChildren: './about/about.module#AboutModule',
        data: { preload: false },
      },
    ],
    { preloadingStrategy: AppCustomPreloader })
...
providers: [AppCustomPreloader],
```

CLI Settings

This thing is super powerful

CLI Flags

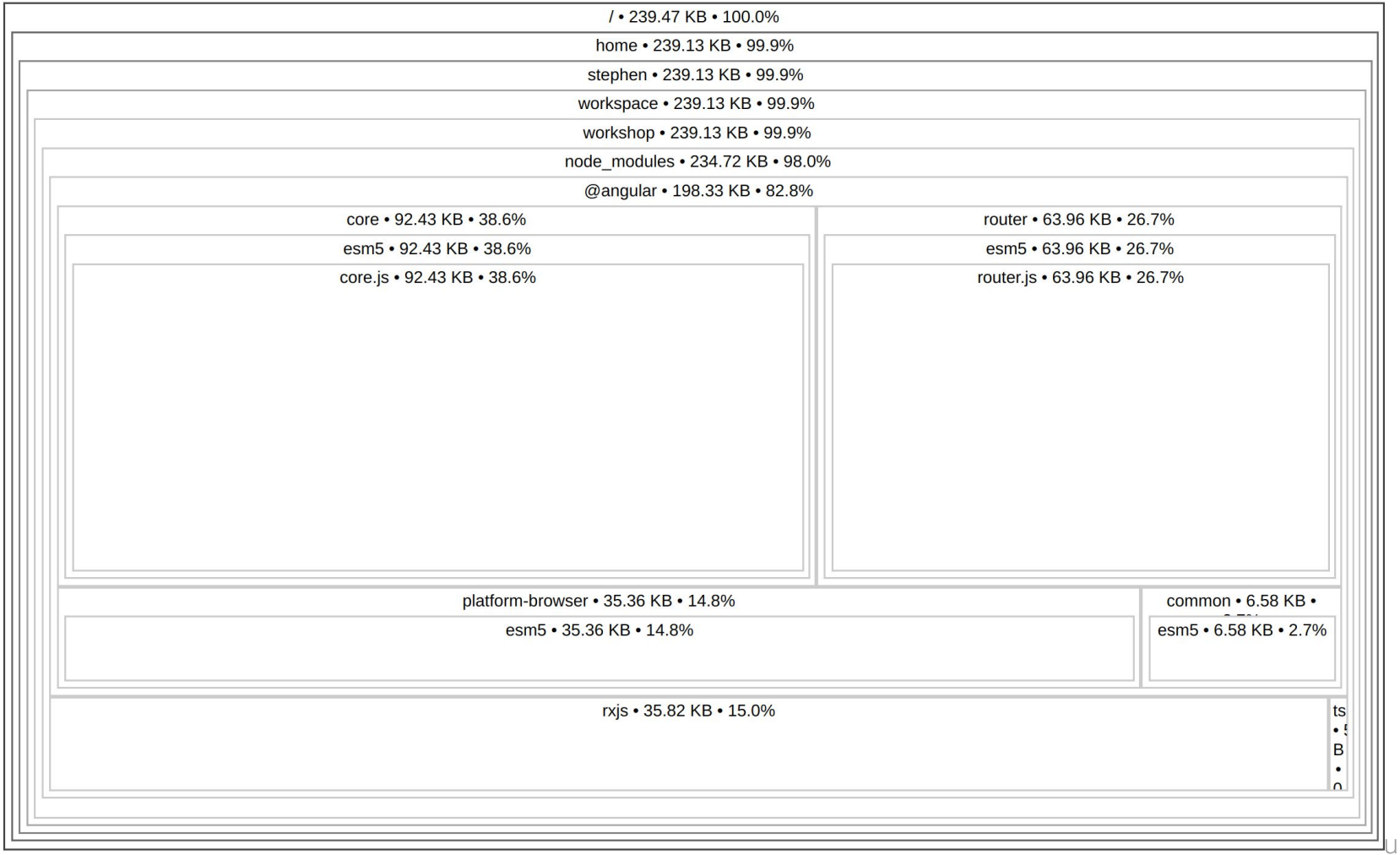
```
ng serve

ng build -prod
```

CLI Flags

```
ng serve --aot

ng build -prod --named-chunks --sourcemaps
```

Source Map Explorer

```
yarn global add source-map-explorer

ng build -prod --named-chunks --sourcemaps

source-map-explorer dist/main*js
```


/ • 239.47 KB • 100.0%
home • 239.13 KB • 99.9%
stephen • 239.13 KB • 99.9%
workspace • 239.13 KB • 99.9%
workshop • 239.13 KB • 99.9%
node_modules • 234.72 KB • 98.0%
@angular • 198.33 KB • 82.8%
core • 92.43 KB • 38.6%
esm5 • 92.43 KB • 38.6%
core.js • 92.43 KB • 38.6%
router • 63.96 KB • 26.7%
esm5 • 63.96 KB • 26.7%
router.js • 63.96 KB • 26.7%
platform-browser • 35.36 KB • 14.8%
esm5 • 35.36 KB • 14.8%
common • 6.58 KB •
esm5 • 6.58 KB • 2.7%
rxjs • 35.82 KB • 15.0%
ts
• 5
B
•
0

Stephen Fluin
@stephenfluin

Follow

Check out the schema file from the #Angular CLI to see all of things you can control in your angular-cli.json config file: github.com/angular/angula …

```
    "poll": {
      "description": "Enable and define the file watching poll time period (milliseconds).",
      "type": "number"
    },
    "deleteOutputPath": {
      "description": "Delete output path before build.",
      "type": "boolean",
      "default": true
    },
    "preserveSymlinks": {
      "description": "Do not use the real path when resolving modules.",
      "type": "boolean",
      "default": false
    },
    "showCircularDependencies": {
      "description": "Show circular dependency warnings on builds.",
      "type": "boolean",
      "default": true
    },
    "commonChunk": {
      "description": "Use a separate bundle containing code used across multiple bundles.",
      "type": "boolean",
      "default": true
    },
    "namedChunks": {
      "description": "Use file name for lazy loaded chunks.",
      "type": "boolean"
    }
  }
```

3:56 PM - 14 Nov 2017 from Mountain View, CA

Let's Try It

Ask a neighbor, or raise your hand if you get stuck or have questions

At Home: Schematics

```
yarn global add @angular-devkit/schematics @schematics/schematics

./node_modules/.bin/schematics @schematics/schematics:schematic --name mine
```

RxJS

The tip of the iceberg

RxJS Tasks

- Simple HTTP GET request
- A Dynamic HTTP GET request
- switchMap - Combine Data Sources
- startWith - LocalStorage Caching
- shareReplay - Avoiding Multiple Subscriptions

Lettable Operators && HttpClient

1. Lettable Operators (Angular v5 && RxJS 5.5)
2. HttpClient (Angular v4.3)

How Lettable Operators Work

```
import { map, switchMap } from 'rxjs/operators';


observable.pipe(
  switchMap(x => http.get(x.id)),
  map(x => x.items),
)
```

Add HTTP to app.module.ts

```
import { HttpClientModule } from '@angular/common/http';

...

imports: [ HttpClientModule ]

...
```

Add search results to home.component.ts

```
repos;
constructor(http: HttpClient) {
  const path = 'https://api.github.com/search/repositories?q=angular';
  this.repos = http.get<any>(path).pipe(
    map(results => results.items)
  );
}
```

Add to home.component.html

```
<div *ngFor="let repo of repos | async">
  <h2>{{ repo.name }}</h2>
  <p>{{ repo.description }}</p>
</div>
```

Add Search Route

```
{ path: 'search/:term', component: HomeComponent },
```

SwitchMap Search Params

```
constructor(http: HttpClient, route: ActivatedRoute) {
  const path = 'https://api.github.com/search/repositories?q=';
  this.repos = route.params.pipe(
    switchMap( params => {
      const term = params['term'] ? params['term'] : 'angular';
      return http.get<any>(`${path}${term}`);
    }),
    map(results => results.items)
  );
}
```

LocalStorage Caching

```
this.repos.subscribe(list => {
  localStorage['listCache'] = JSON.stringify(list);
});
this.repos = this.repos.pipe(
  startWith(JSON.parse(localStorage['listCache'] || '[]'))
);
```

shareReplay

```
this.repos = route.params.pipe(
  switchMap(params => {
    const term = params['term'] ? params['term'] : 'angular';
    return http.get<any>(`${path}${term}`);
  }),
  map(results => results.items),
  shareReplay(1)
);
```

Let's Try It

Ask a neighbor, or raise your hand if you get stuck or have questions

At Home: Refactor into a Service

- How do you pass the parameter into the service?
- Do you want to cache each search term?
- Where should you shareReplay so that navigating around your app doesn't trigger more HTTP Requests?
- What if you wanted to refresh your HTTP requests every 5 minute?

5 minute break

back at 11:10

Service Worker

More Engaging Applications
aka
Progressive Web Apps

Service Worker

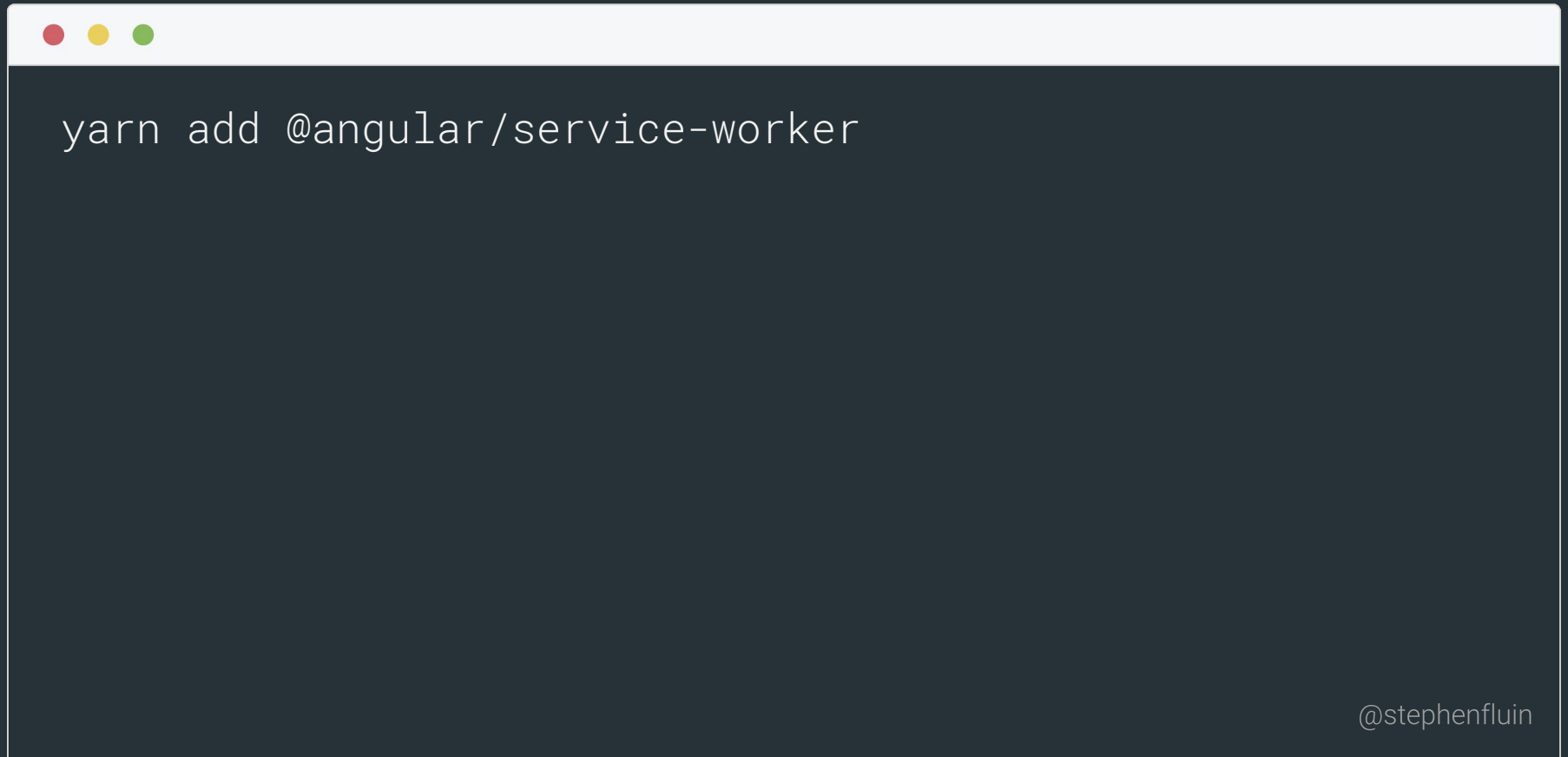
Requires CLI 1.6 (currently beta)

Could have ng new project --service-worker

Service Worker Tasks

1. Install
2. Configure CLI
3. Configure Service Worker
4. Configure Project
5. Register for Push Notifications

Installation



.angular-cli.json

```
"serviceWorker": true
```

New File: src/ngsw-config.json

```
{
  "index": "/index.html",
  "assetGroups": [{
      "name": "app",
      "installMode": "prefetch",
      "resources": {
        "files": ["/favicon.ico", "/index.html"],
        "versionedFiles": ["/*.bundle.css", "/*.bundle.js", "/*.chunk.js"]
      }
  }]
}
```

Add asset prefetch to assetGroups in ngsw-config.json

```
{
  "name": "assets",
  "installMode": "lazy",
  "updateMode": "prefetch",
  "resources": {
    "files": ["/assets/**"]
  }
}
```

app.module.ts

```
ServiceWorkerModule.register('ngsw-worker.js'),
```

app.module.ts

```
environment.production
    ? ServiceWorkerModule.register('ngsw-worker.js')
    : [],
```

Service Workers w/ Build

```
# Won't contain a service worker

ng serve


# Will contain a service worker

ng build -prod

cd dist

python -m SimpleHTTPServer
```

Advanced Angular

Home About

angular

One framework. Mobile & desktop.

angular

所有angular学习过程中的代码

angular

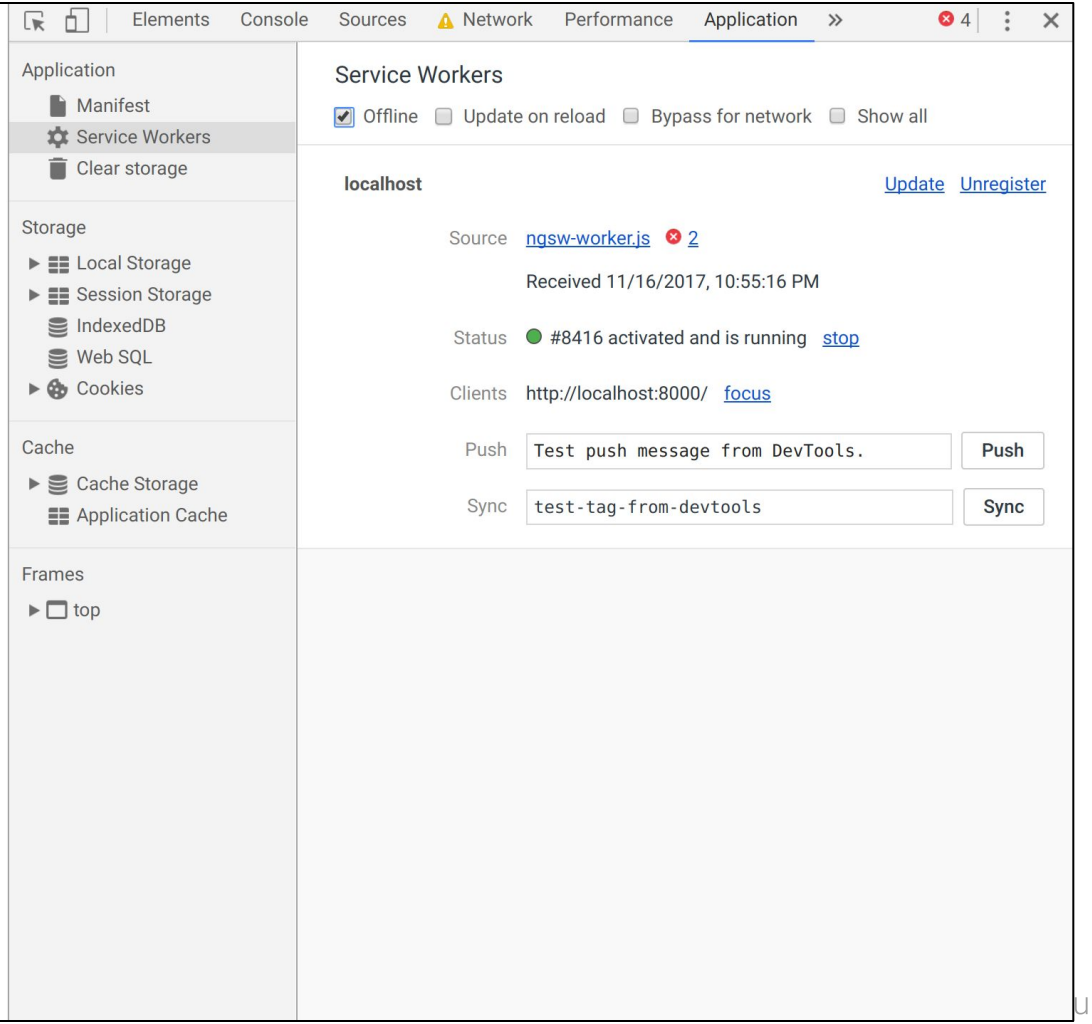
Code to optimize AngularJS for complex pages

angular

Fast and productive web framework provided by Dart

angular

UI-Router for Angular: State-based routing for Angular (v2+)



Let's Try It

Ask a neighbor, or raise your hand if you get stuck or have questions

Push Notifications

1. At build time
    a. Generate server keys
2. At runtime
    a. Register (And ask for permission)
    b. Subscribe to notifications
3. Later
    a. Send notifications

Listen for pushes

```
export class AppModule {
  constructor(injector: Injector) {
    const swPush = injector.get(SwPush, null);
    if (swPush) {
      // swPush.requestSubscription({ serverPublicKey: 'MYKEY' });
      swPush.messages.subscribe(msg => {
        console.log('got a push notification', msg);
      });
    }
  }
}
```

Read More about Push

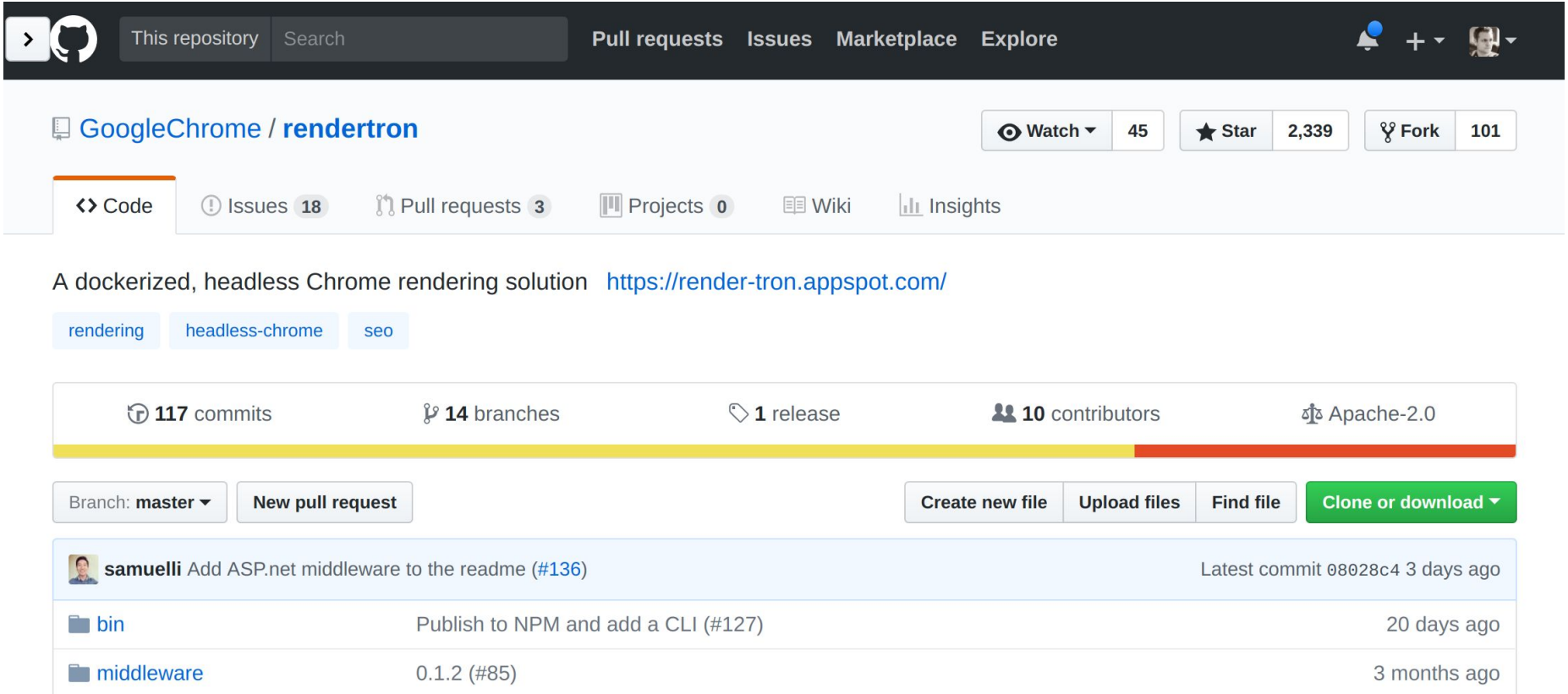
https://developers.google.com

/web/fundamentals/push-notifications/subscribing-a-user

Angular Universal

Machine Parsability + Perceived Performance

Just want Crawlability?



Seamless Transitions w/ Universal

- Generate a CommonJS build
- Render to string method
- Build a server

dependencies

```
yarn add @angular/platform-server
@nguniversal/module-map-ngfactory-loader ts-loader
```

Update app.module.ts's BrowserModule

```
BrowserModule.withServerTransition({appId: 'my-app'}),
```

Create app.server.module.ts

```
import {NgModule} from '@angular/core';
import {ServerModule} from '@angular/platform-server';
import {ModuleMapLoaderModule} from '@nguniversal/module-map-ngfactory-loader';
import {AppModule} from './app.module';
import {AppComponent} from './app.component';
@NgModule({
  imports: [
    AppModule,
    ServerModule,
    ModuleMapLoaderModule // <-- *Important* to have lazy-loaded routes work
  ],
  bootstrap: [AppComponent],
})
export class AppServerModule {}
```

Create tsconfig.server.json

```
{
  "extends": "../tsconfig.json",
  "compilerOptions": {
    "outDir": "../out-tsc/app",
    "baseUrl": "./",
    "module": "commonjs",
    "types": []
  },
  "exclude": [
    "test.ts",
    "**/*.spec.ts"
  ],
  "angularCompilerOptions": {
    "entryModule": "app/app.server.module#AppServerModule"
  }
}
```

Copy your app in .angular-cli.json

```
{
...
      "platform": "server",
      "outDir": "dist-server",
      "main": "main.server.ts",
      "tsconfig": "tsconfig.server.json",
...
}
```

Recap

Better code than ever,
already sitting on your laptop

Some great app strategies

1. Router Lazy Loading
    a. Preload Strategies
2. CLI Settings
    a. Customizing Builds
3. RxJS
    a. Caching
4. Service Worker
5. Universal

Thank you!

STEPHEN FLUIN

Developer Advocate