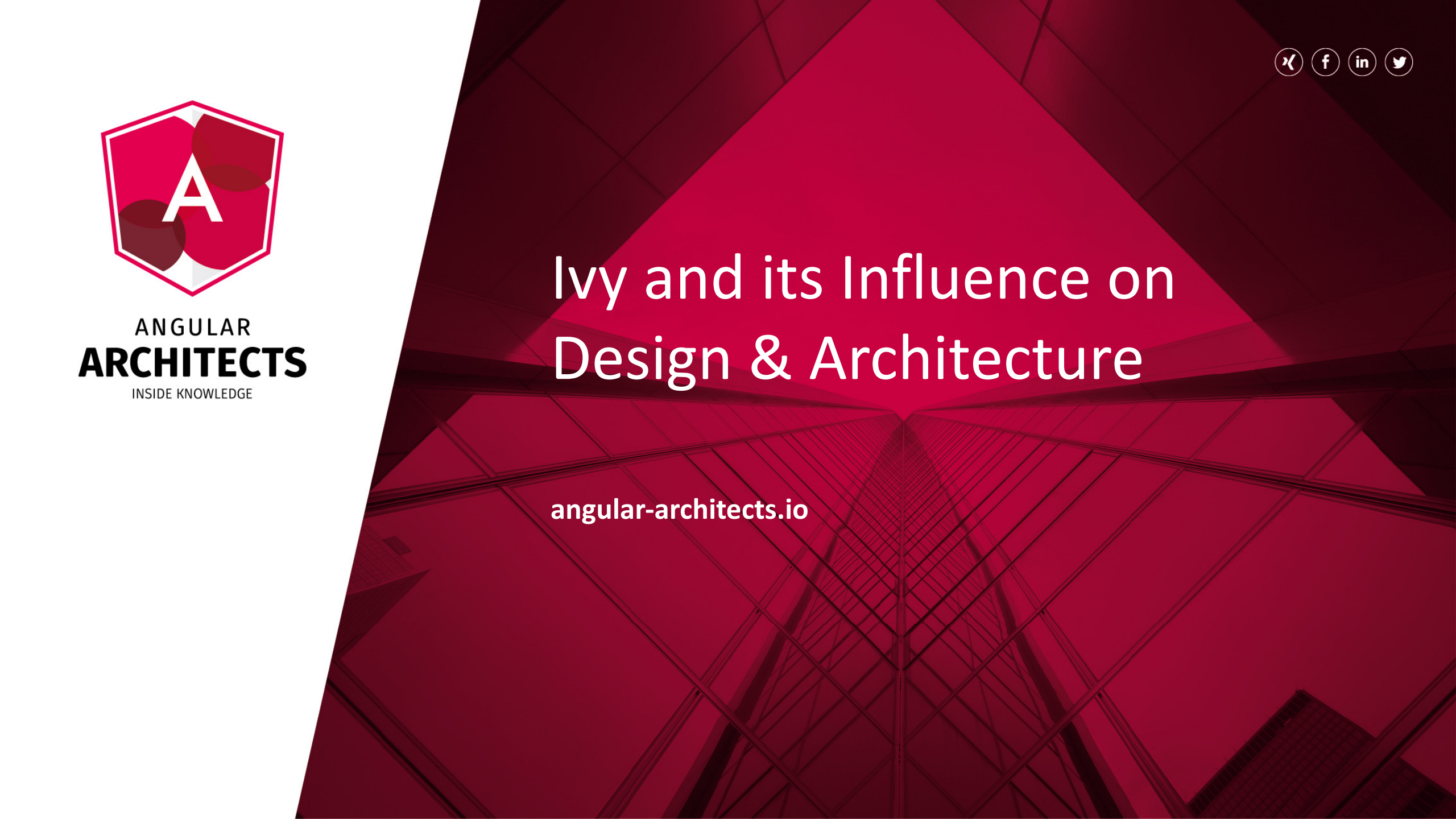


ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Ivy and its Influence on Design & Architecture

angular-architects.io

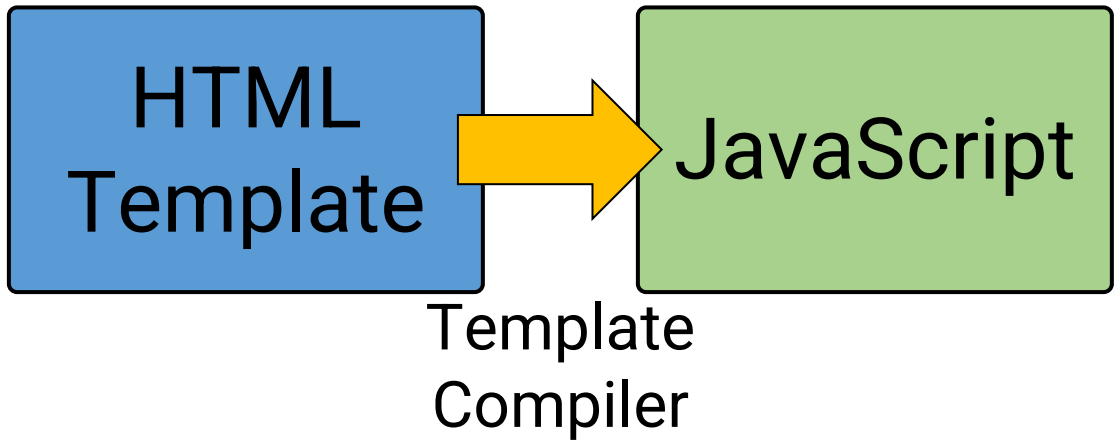
Contents

1)
Ivy Today

2)
Ivy Tomorrow?

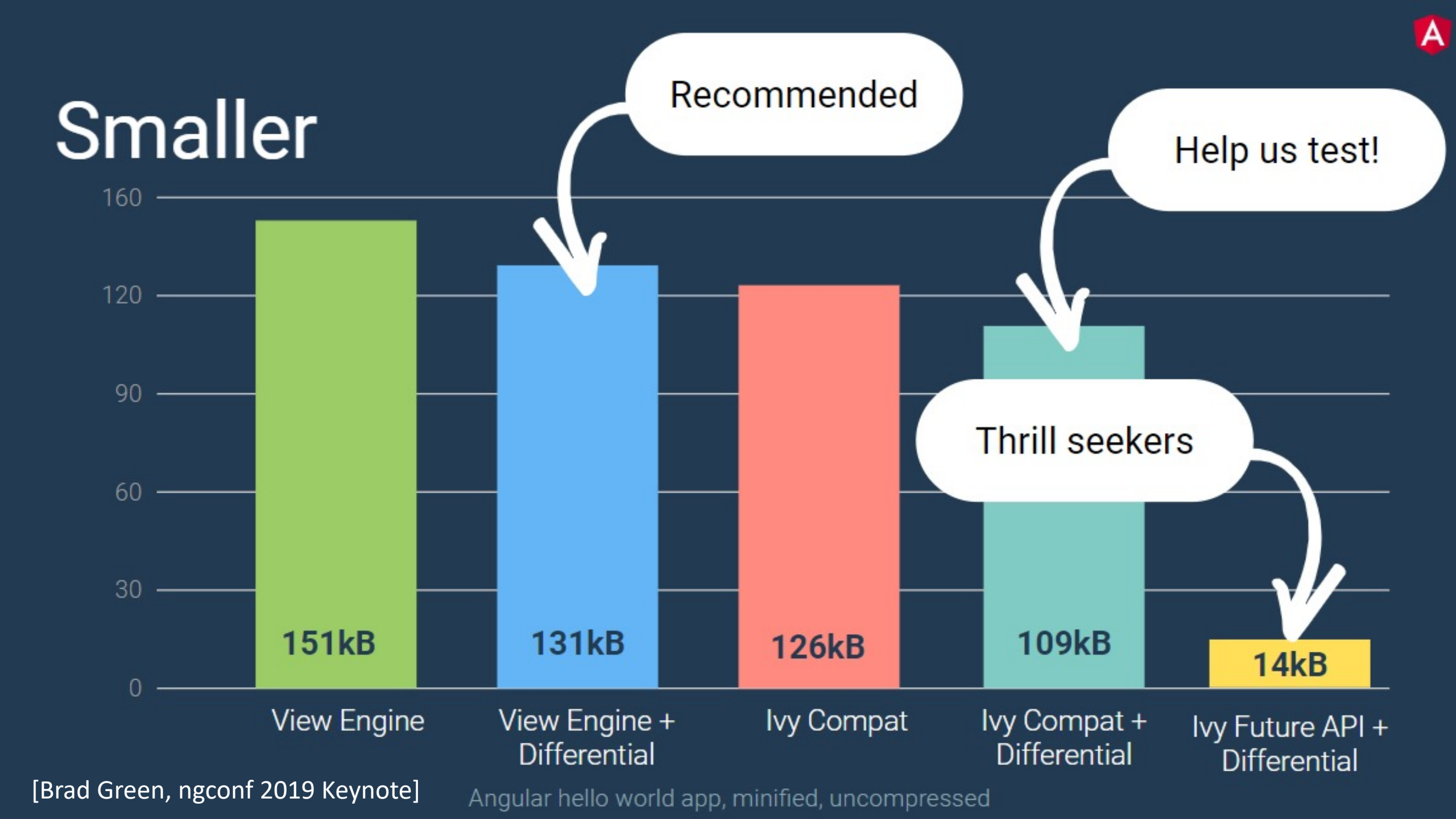
Ivy Today

Angular Compiler



Ivy Compiler leads to smaller Bundles

Up to **40%** less bundle size!



[Brad Green, ngconf 2019 Keynote]

DEMO

Default with Angular 9

```
// Opt out in tsconfig.json
{
    [...]
    "angularCompilerOptions": {
        "enableIvy": false
    }
}
```

Roadmap

Angular 9 (Spring 2020)	• Focus: Backwards compatibility
Angular 11+ (Late Fall 2020)	• New features based upon ivy?

Only Smaller Bundles?

Lazy Components

Lazy Components

```
import('../dashboard-tile/dashboard-tile.component').then(m => {




});
```

Lazy Components

```
import('../dashboard-tile/dashboard-tile.component').then(m => {
    const comp = m.DashboardTileComponent;




});
```

Lazy Components

```
import('../dashboard-tile/dashboard-tile.component').then(m => {
    const comp = m.DashboardTileComponent;

    // Only b/c of compatibility; will not be needed in future!
    const factory = this.cfr.resolveComponentFactory(comp);



});
```

Lazy Components

```
import('../dashboard-tile/dashboard-tile.component').then(m => {
    const comp = m.DashboardTileComponent;

    // Only b/c of compatibility; will not be needed in future!
    const factory = this.cfr.resolveComponentFactory(comp);

    const compRef = this.viewContainer.createComponent(
                                    factory, null, this.injector);
});
```

DEMO


Ivy's architecture has lots of potential for future Angular versions!

Ivy Tomorrow?

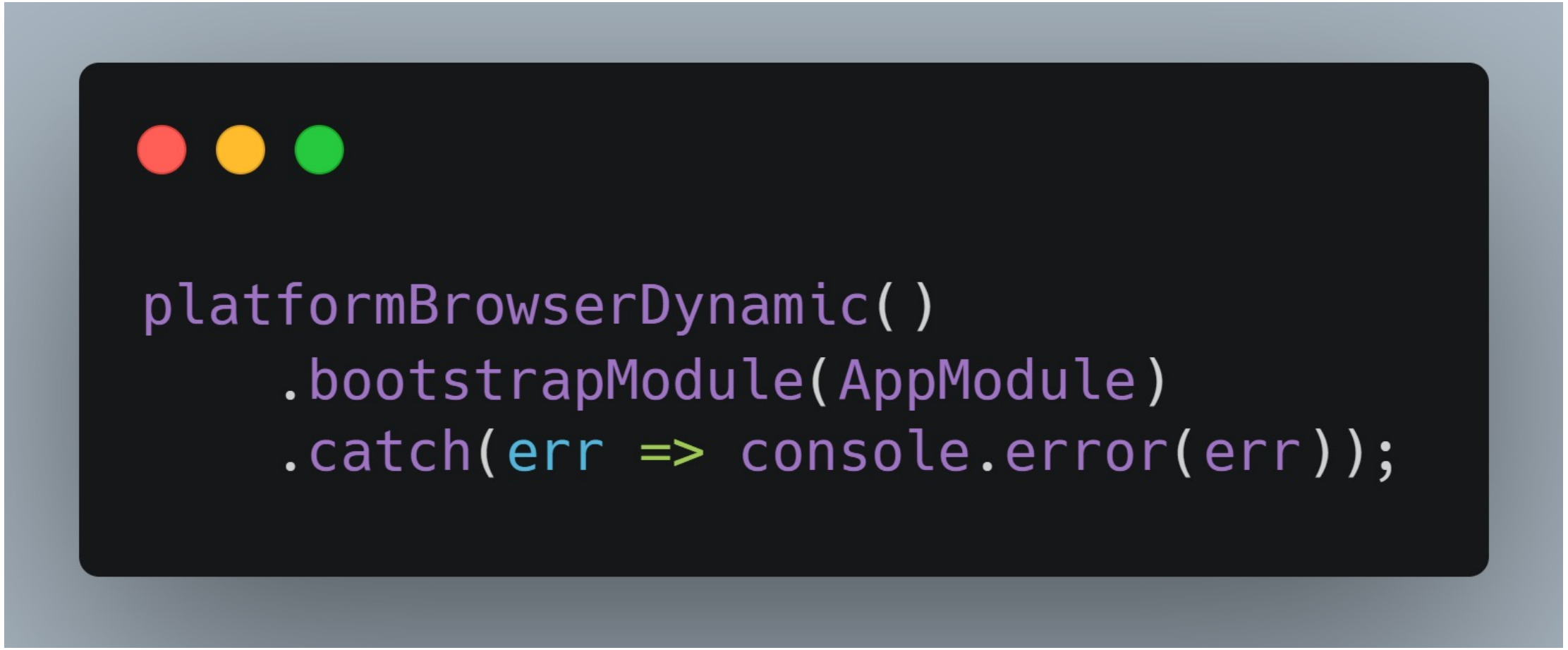
Warning: Private APIs ahead!

Don't use them in production!

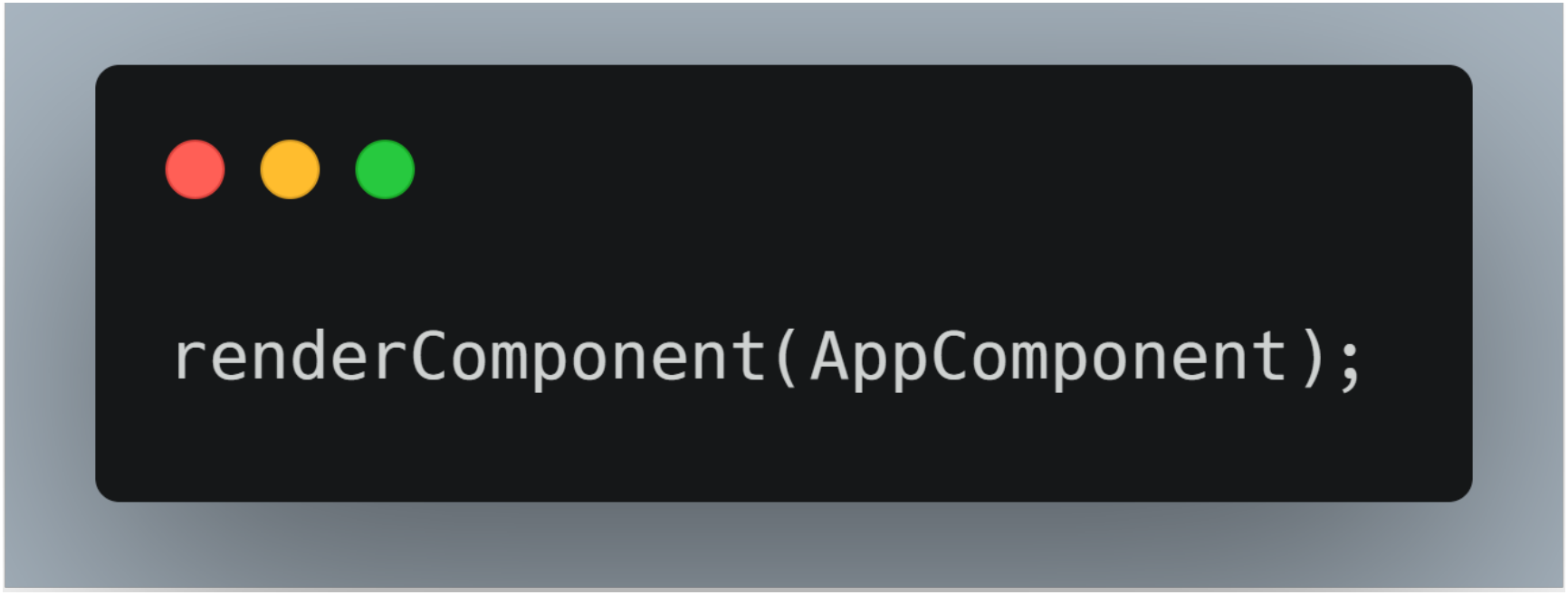
No guarantees those features will ever land in Angular!

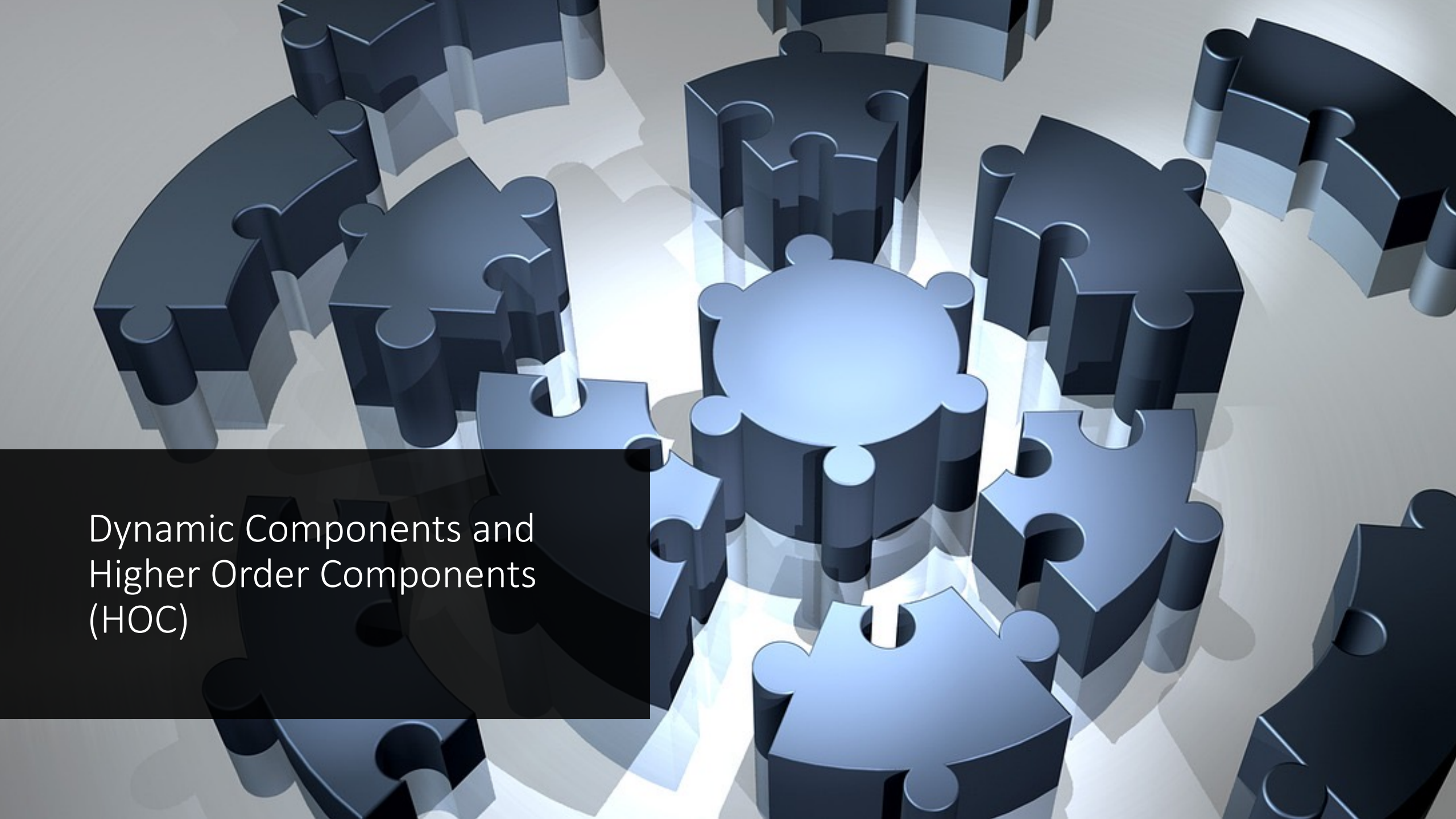
Bootstrapping

Bootstrapping today



Bootstrapping with Ivy

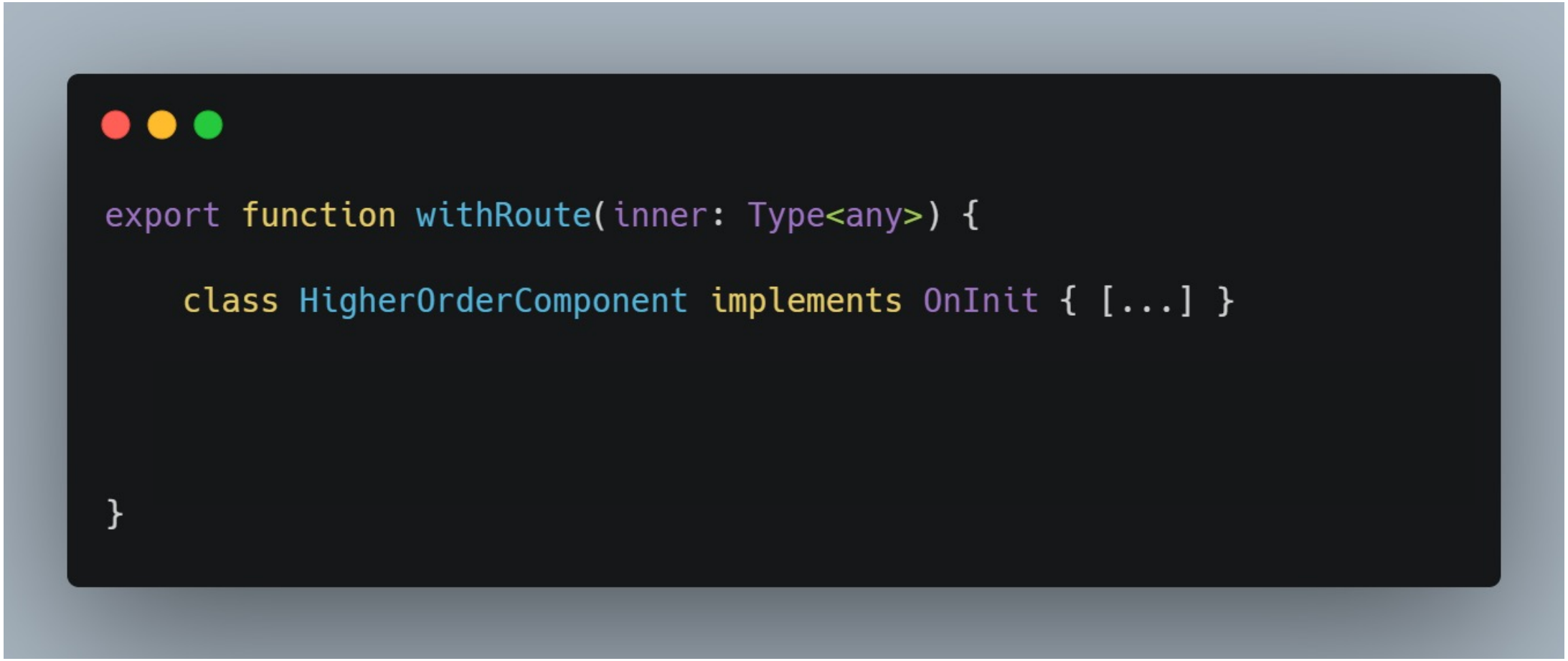




Dynamic Components and Higher Order Components (HOC)

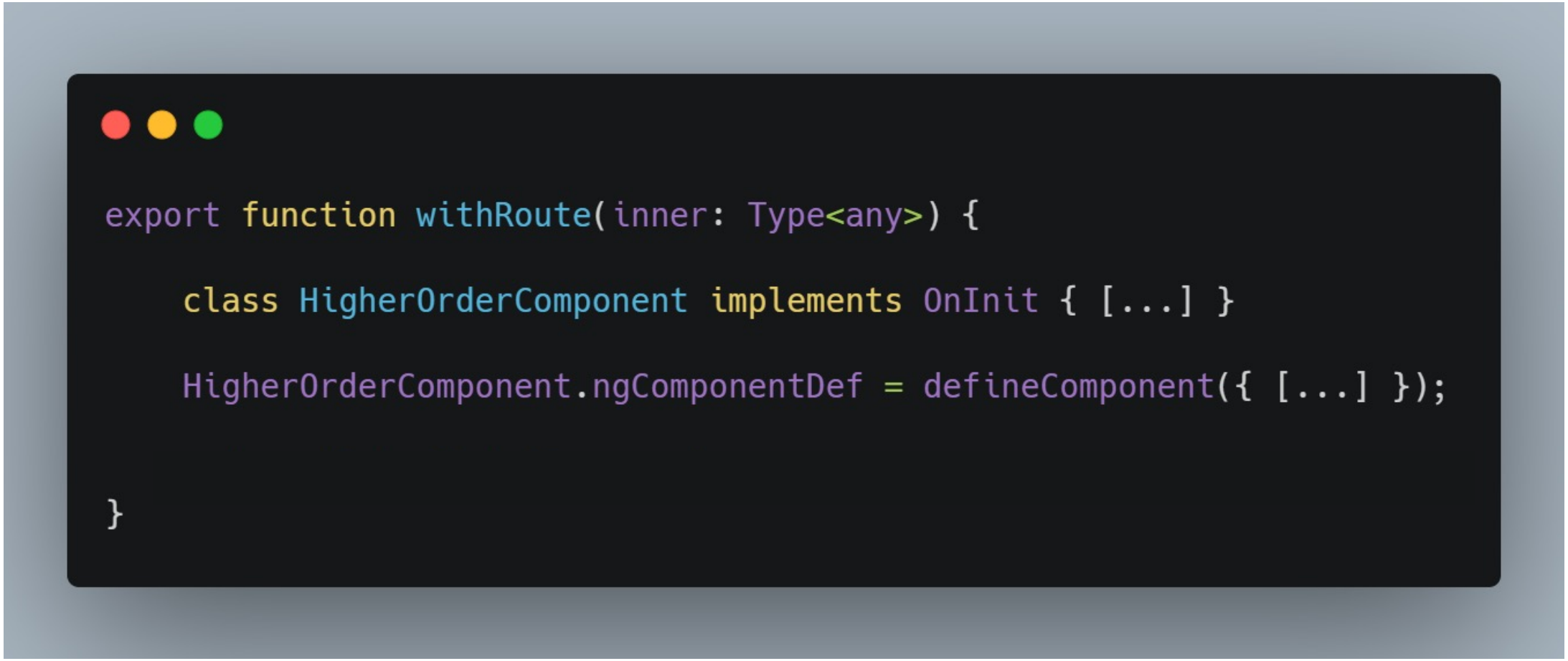
Dynamic and Higher Order Components

Dynamic and Higher Order Components



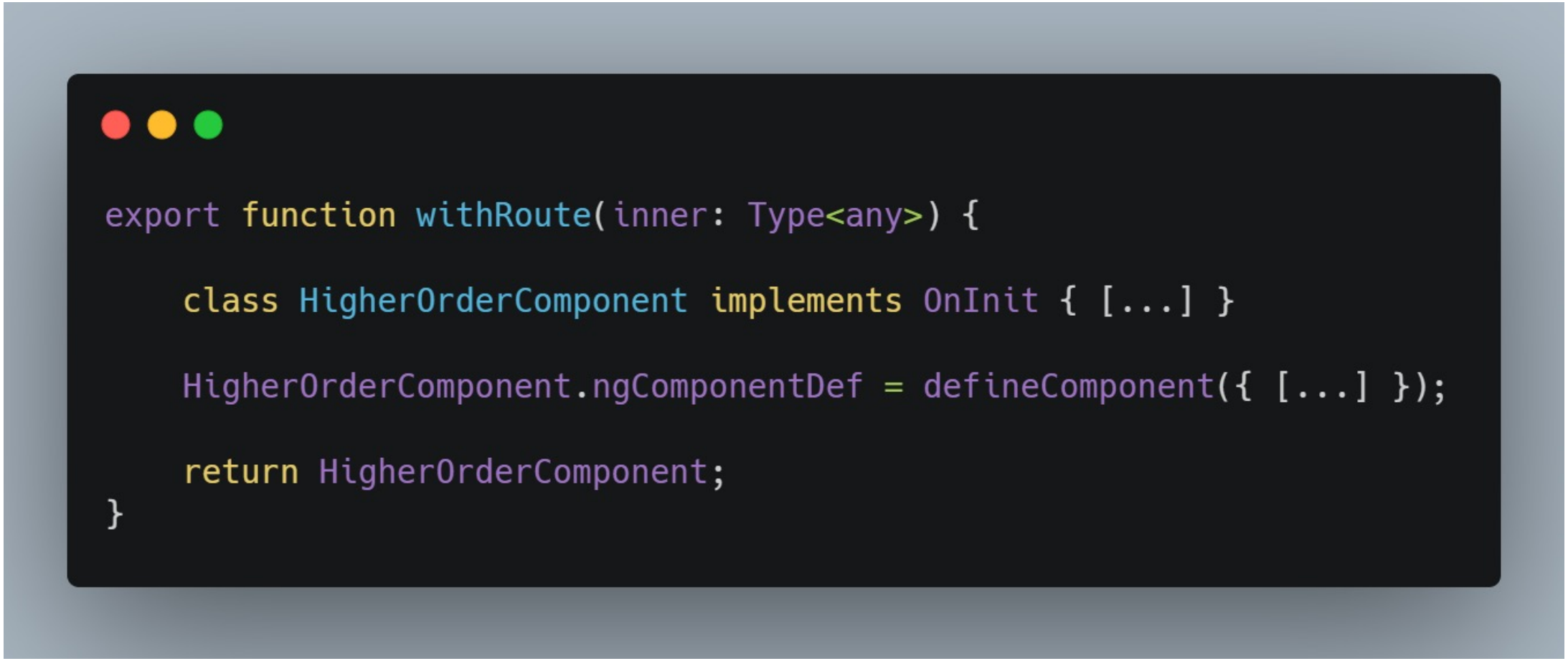
```
export function withRoute(inner: Type<any>) {

    class HigherOrderComponent implements OnInit { [...] }



}
```

Dynamic and Higher Order Components



```
export function withRoute(inner: Type<any>) {

    class HigherOrderComponent implements OnInit { [...] }

    HigherOrderComponent.ngComponentDef = defineComponent({ [...] });


}
```

Dynamic and Higher Order Components



```
export function withRoute(inner: Type<any>) {

    class HigherOrderComponent implements OnInit { [...] }

    HigherOrderComponent.ngComponentDef = defineComponent({ [...] });

    return HigherOrderComponent;
}
```

DEMO

Optional NgModules

Optional NgModules

Optional NgModules

```
const fooDef = FooComponent.ɵcmp;
const barDef = BarComponent.ɵcmp;

fooDef.directives = [
    barDef
];
```

Public-APIs instead of NgModules

```
// index.ts
export const MY_COMPS = [
    BarComponent.ɵcmp,
    FizzComponent.ɵcmp
];
```

Public-APIs instead of NgModules

```
// index.ts
export const MY_COMPS = [
    BarComponent.ɵcmp,
    FizzComponent.ɵcmp
];

// consumer.ts
import { MY_COMPS } from './index';

const def = FooComponent.ɵcmp;
```

Public-APIs instead of NgModules

```
// index.ts
export const MY_COMPS = [
    BarComponent.ɵcmp,
    FizzComponent.ɵcmp
];

// consumer.ts
import { MY_COMPS } from './index';

const def = FooComponent.ɵcmp;

def.directives = [
    ...MY_COMPS
];
```

Minko Gechev

@mgechev

Proposal

```
@Component({
    [...],
    deps: [
        ...BAR_COMPONENTS
    ]
})
export class FooComponent {
    [...]
}
```

DEMO

Zone-less Change Detection

What is Zone.js?

- Key to automatic change detection in Angular
- Monkey-patches all browser objects
- Finds out when event handlers ran
- Tells Angular to update data bindings

Downsides

- 100+ KB (uncompressed)
  - Web Components?
- Magic
- Zone.js cannot monkey patch native async/ await (ES 2017)

Zone-less Change Detection

```
import { ..., markDirty } from '@angular/core';

[...]

markDirty(this /* <-- Component Instance */);
```

DEMO

Selected Ideas for Automatic Change Detection

Push Pipe

Using observables and a push pipe

```
export class MyComponent {
  price$: Observable<number>;
  [...]
}
```

```
{{ price$ | push }}
```

@ngrx/component (currently prototype)

```
class MoviesComponent extends ReactiveComponent {



}
```

Mike Ryan
@MikeRyanDev

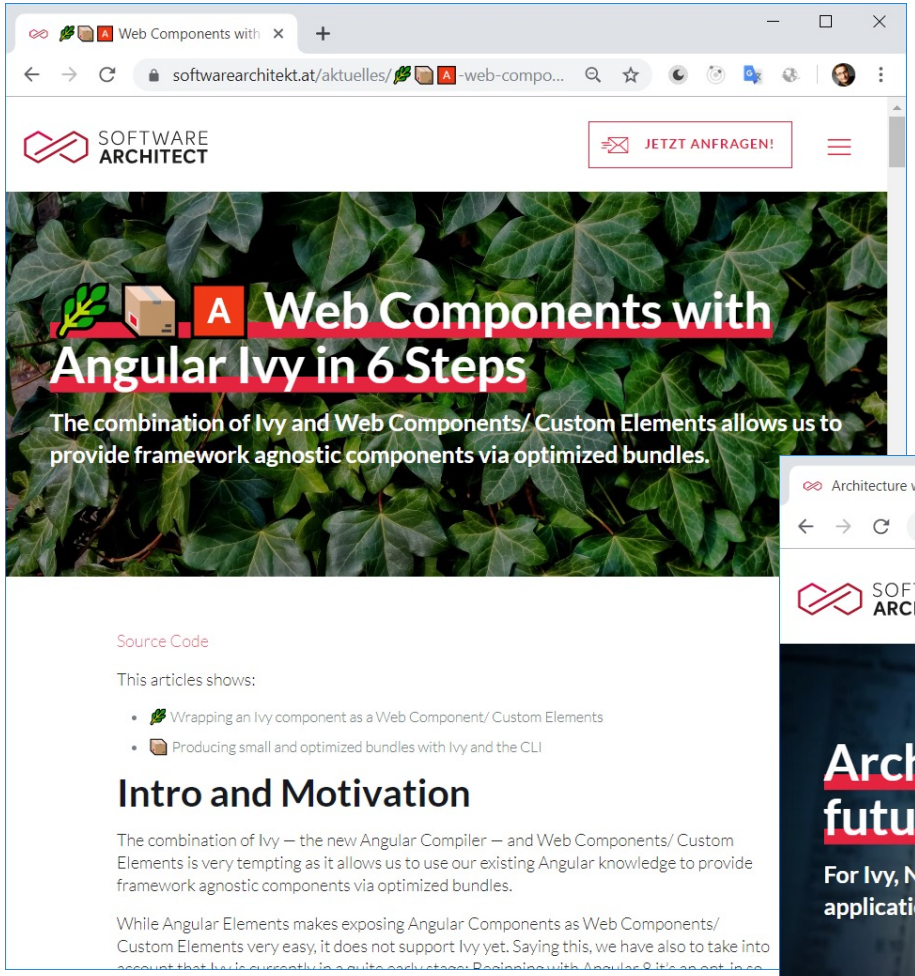
@ngrx/component (currently prototype)

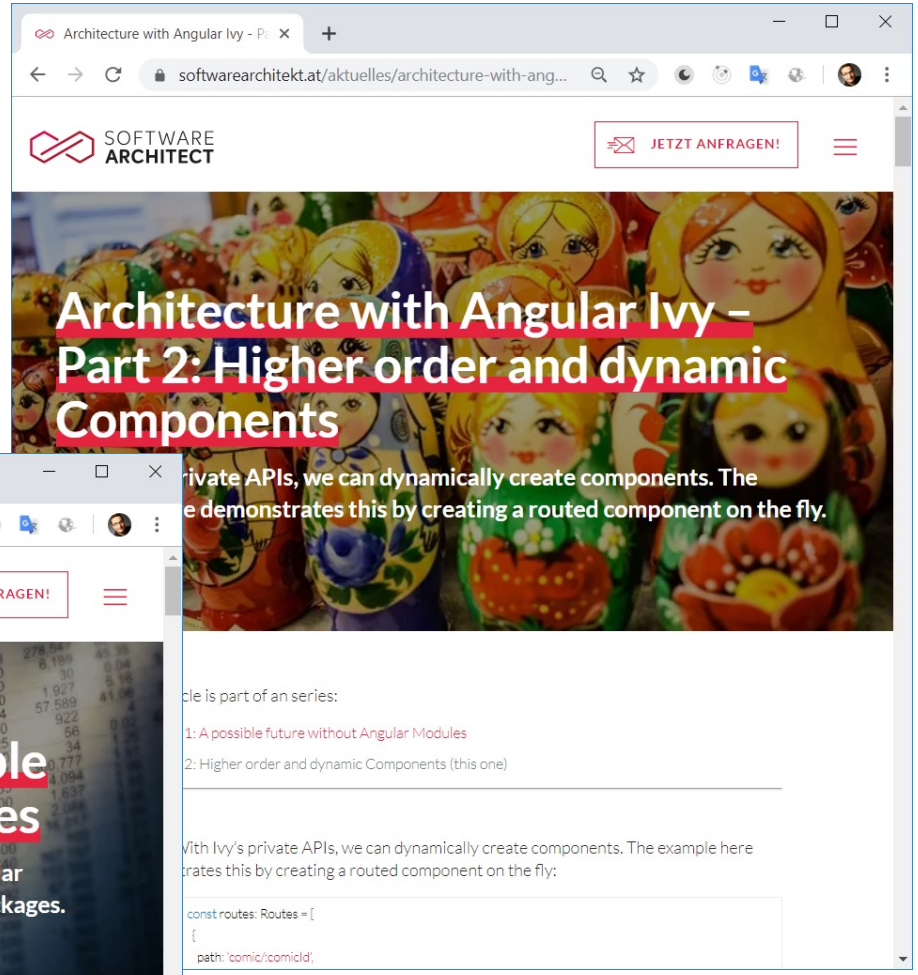
```
class MoviesComponent extends ReactiveComponent {

  state = this.connect({
    movies: this.http
      .get("/api/v1/movies")
      .pipe(map(res => res.data))
  });

  constructor(private htt
}
```

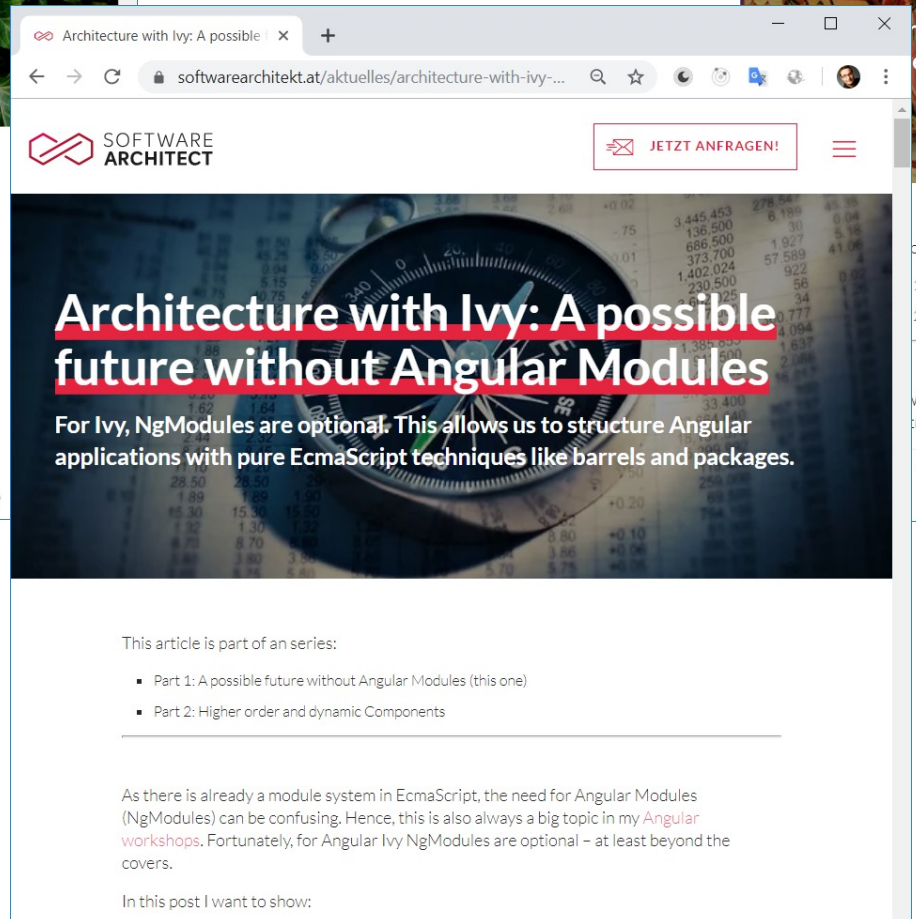
```
{{ state.movies.length }}
```

Mike Ryan

@MikeRyanDev


Web Components with
softwarearchitekt.at/aktuelles/
SOFTWARE ARCHITECT
JETZT ANFRAGEN!
Web Components with Angular Ivy in 6 Steps
The combination of Ivy and Web Components/ Custom Elements allows us to provide framework agnostic components via optimized bundles.
Source Code
This articles shows:
Wrapping an Ivy component as a Web Component/ Custom Elements
Producing small and optimized bundles with Ivy and the CLI
Intro and Motivation
The combination of Ivy — the new Angular Compiler — and Web Components/ Custom Elements is very tempting as it allows us to use our existing Angular knowledge to provide framework agnostic components via optimized bundles.
While Angular Elements makes exposing Angular Components as Web Components/ Custom Elements very easy, it does not support Ivy yet. Saying this, we have also to take into


Architecture with Angular Ivy - P
softwarearchitekt.at/aktuelles/architecture-with-ang...
SOFTWARE ARCHITECT
JETZT ANFRAGEN!
Architecture with Angular Ivy – Part 2: Higher order and dynamic Components
rivate APIs, we can dynamically create components. The
e demonstrates this by creating a routed component on the fly.
cle is part of an series:
1: A possible future without Angular Modules
2: Higher order and dynamic Components (this one)
Vith Ivy's private APIs, we can dynamically create components. The example here
rates this by creating a routed component on the fly:
const routes: Routes = [
{
path: 'comic/:comicId',


Architecture with Ivy: A possible
softwarearchitekt.at/aktuelles/architecture-with-ivy-...
SOFTWARE ARCHITECT
JETZT ANFRAGEN!
Architecture with Ivy: A possible future without Angular Modules
For Ivy, NgModules are optional. This allows us to structure Angular applications with pure EcmaScript techniques like barrels and packages.
This article is part of an series:
Part 1: A possible future without Angular Modules (this one)
Part 2: Higher order and dynamic Components
As there is already a module system in EcmaScript, the need for Angular Modules (NgModules) can be confusing. Hence, this is also always a big topic in my Angular workshops. Fortunately, for Angular Ivy NgModules are optional – at least beyond the covers.
In this post I want to show:

Conclusion

Short term: Smaller bundles	Long term: Huge potential	Lazy components
HOC	Optional NgModules	Zone-less change detection


"Your future hasn't been written yet!"
Emmet Brown, PhD