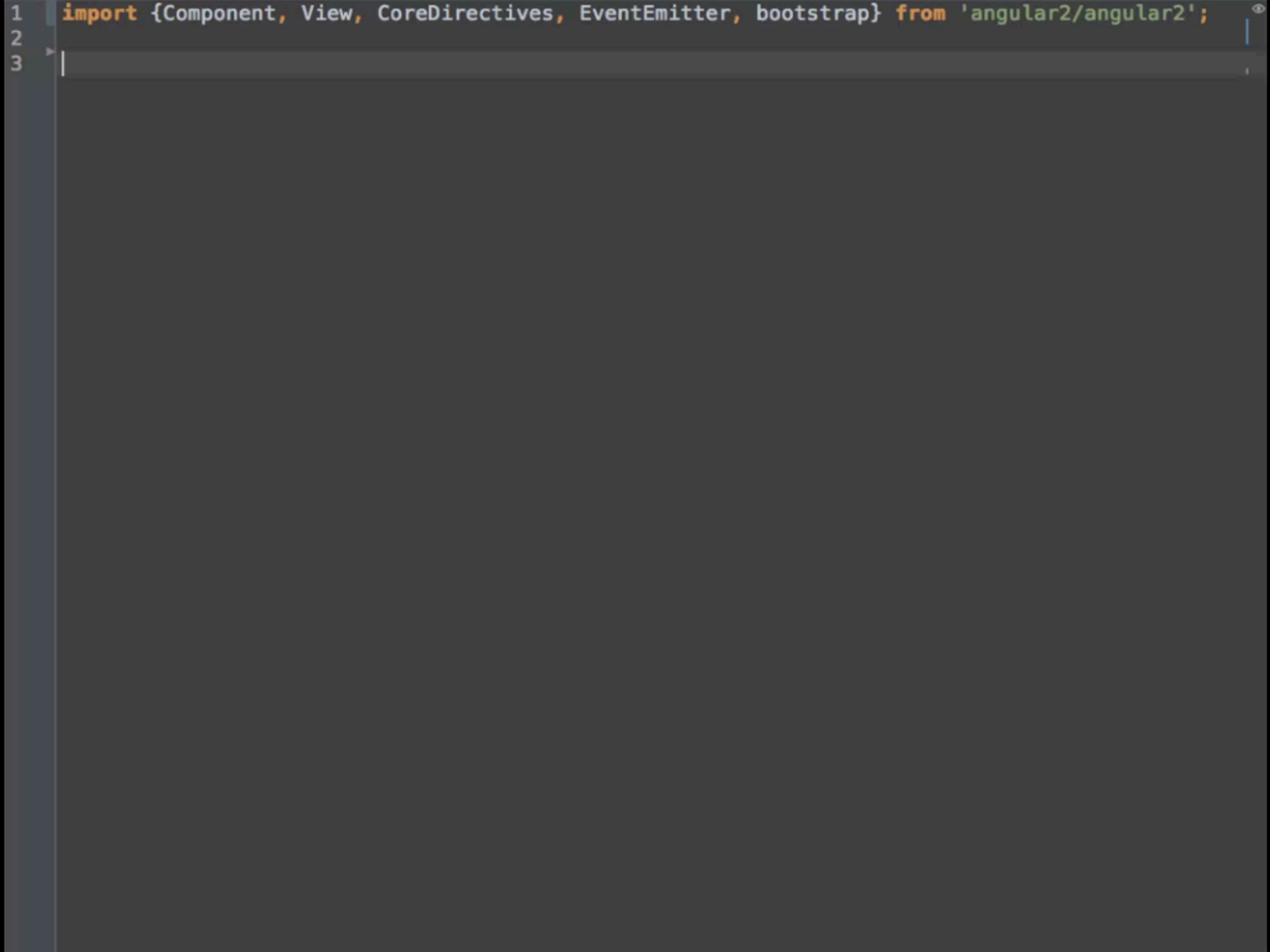
. . . .

Introduction into Angular 2

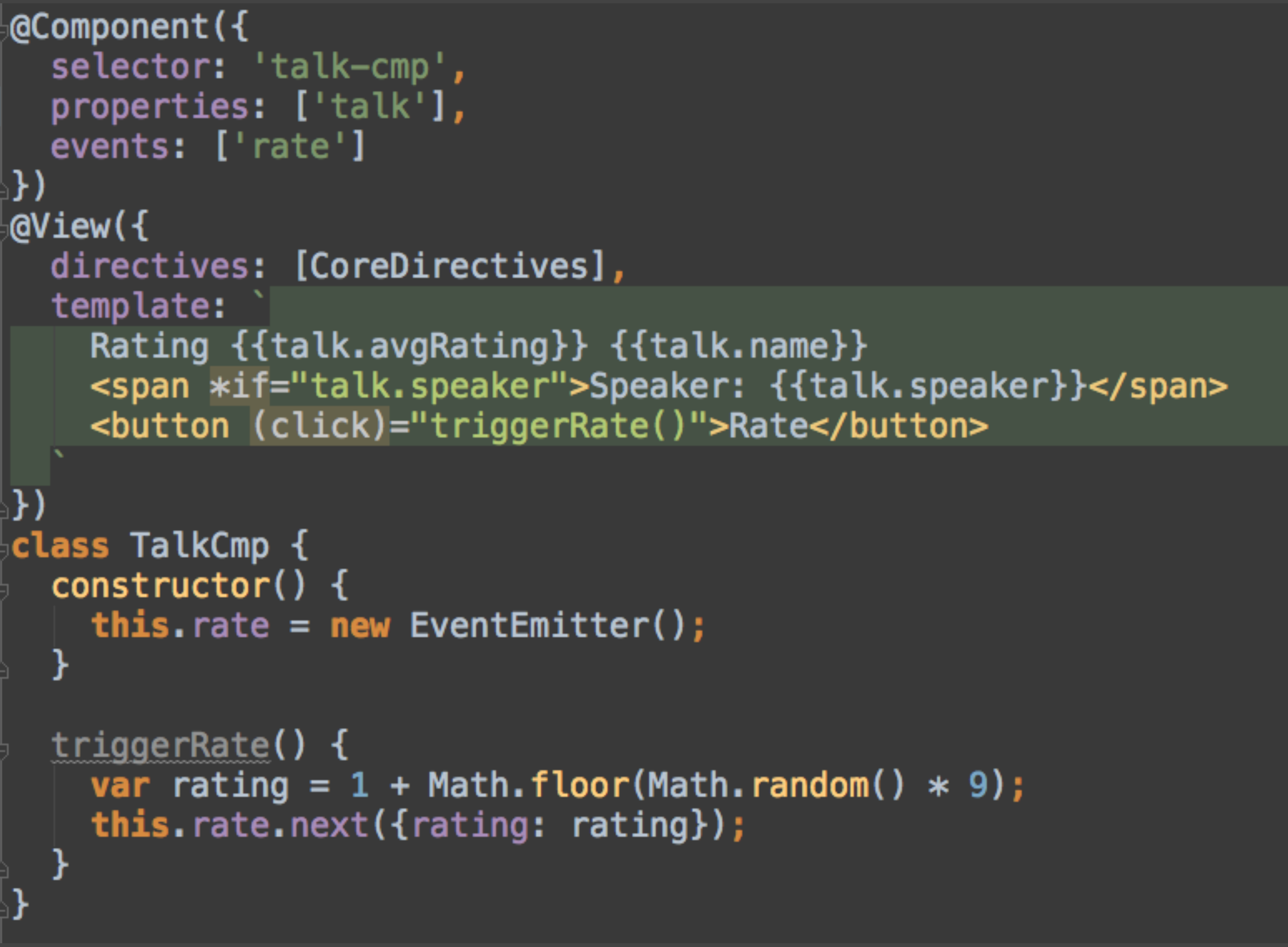
Victor Savkin

Great for Building Large Applications

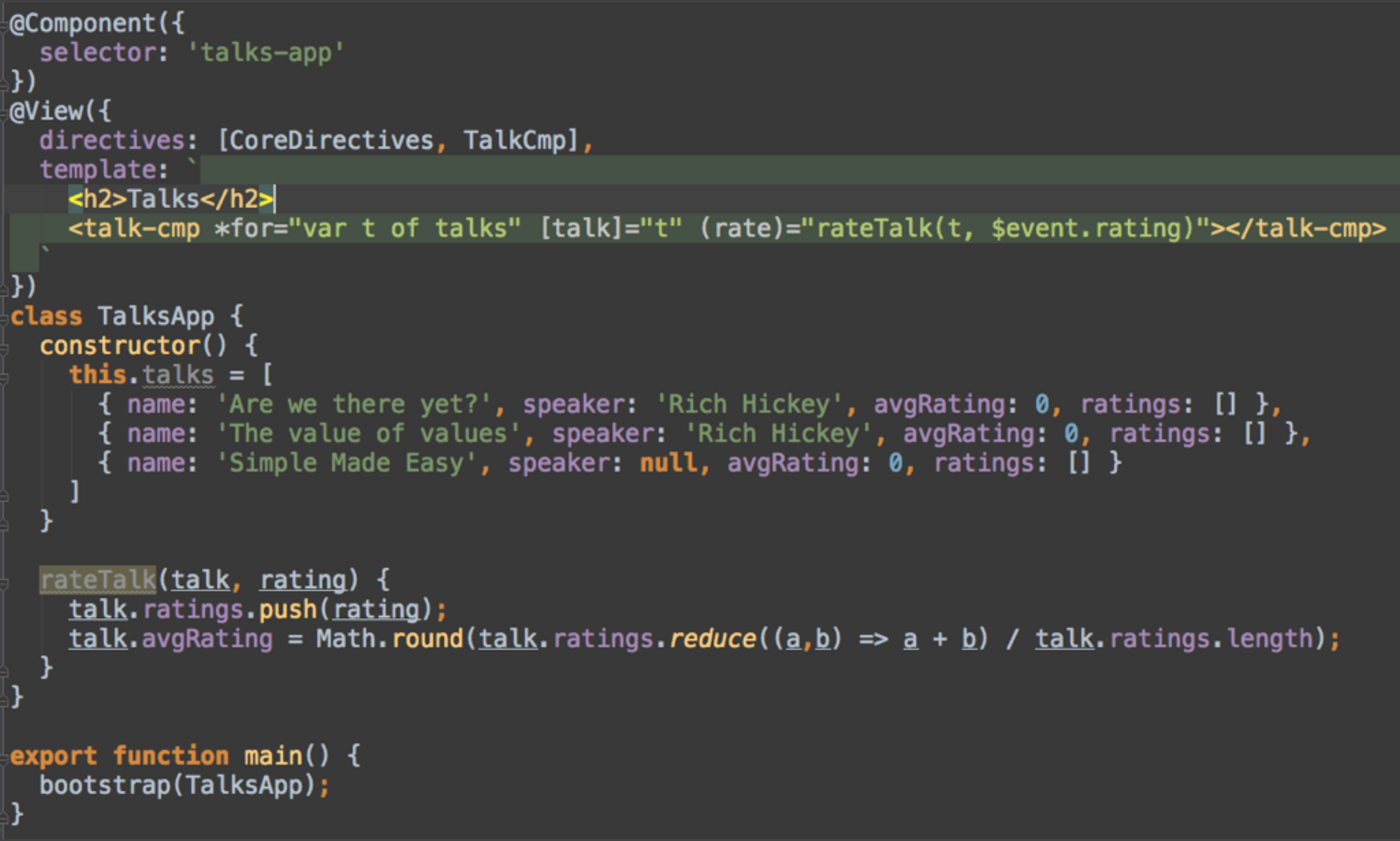
And Easy to Get Started With



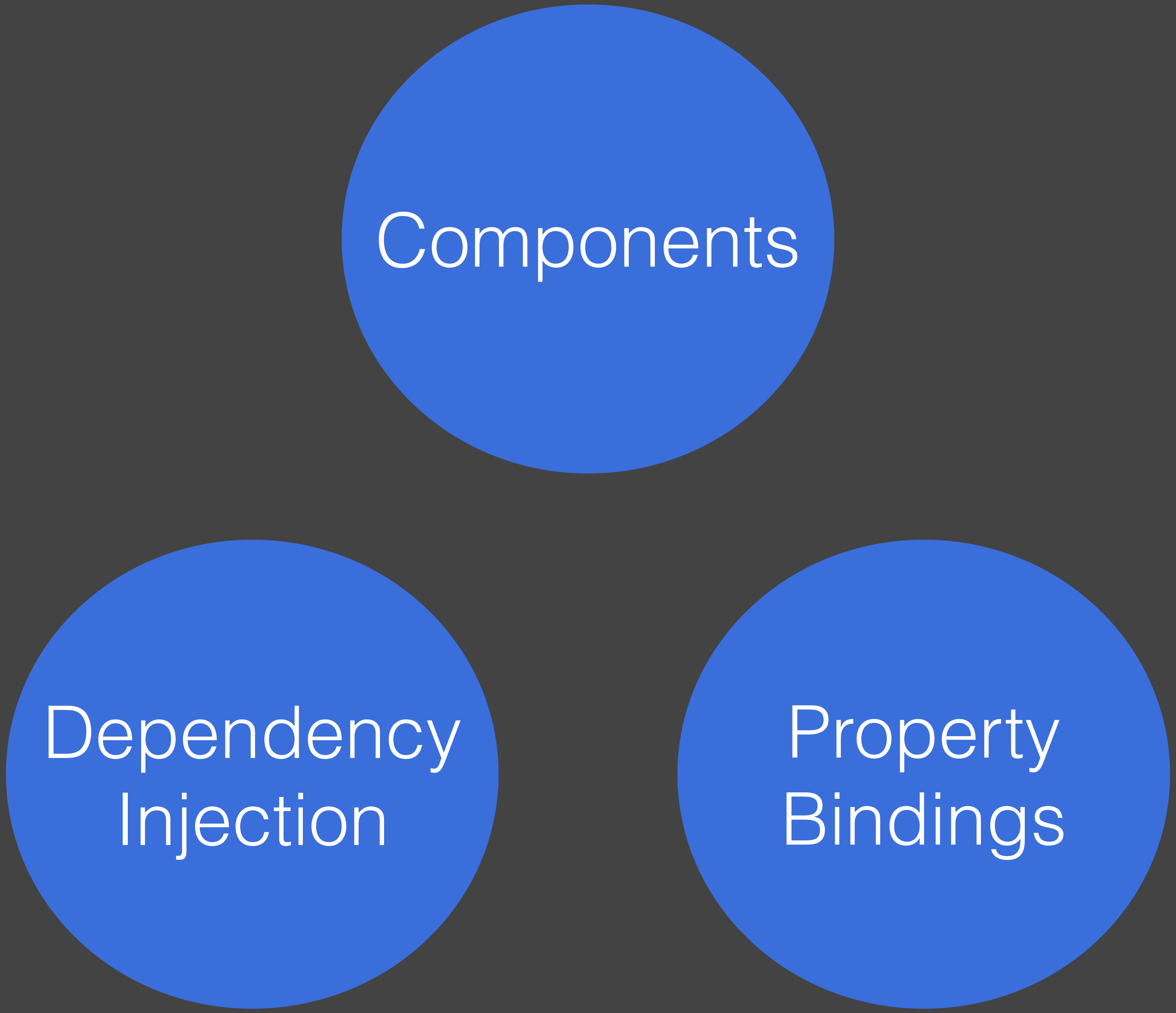
```
import {Component, View, CoreDirectives, EventEmitter, bootstrap} from 'angular2/angular2';

```



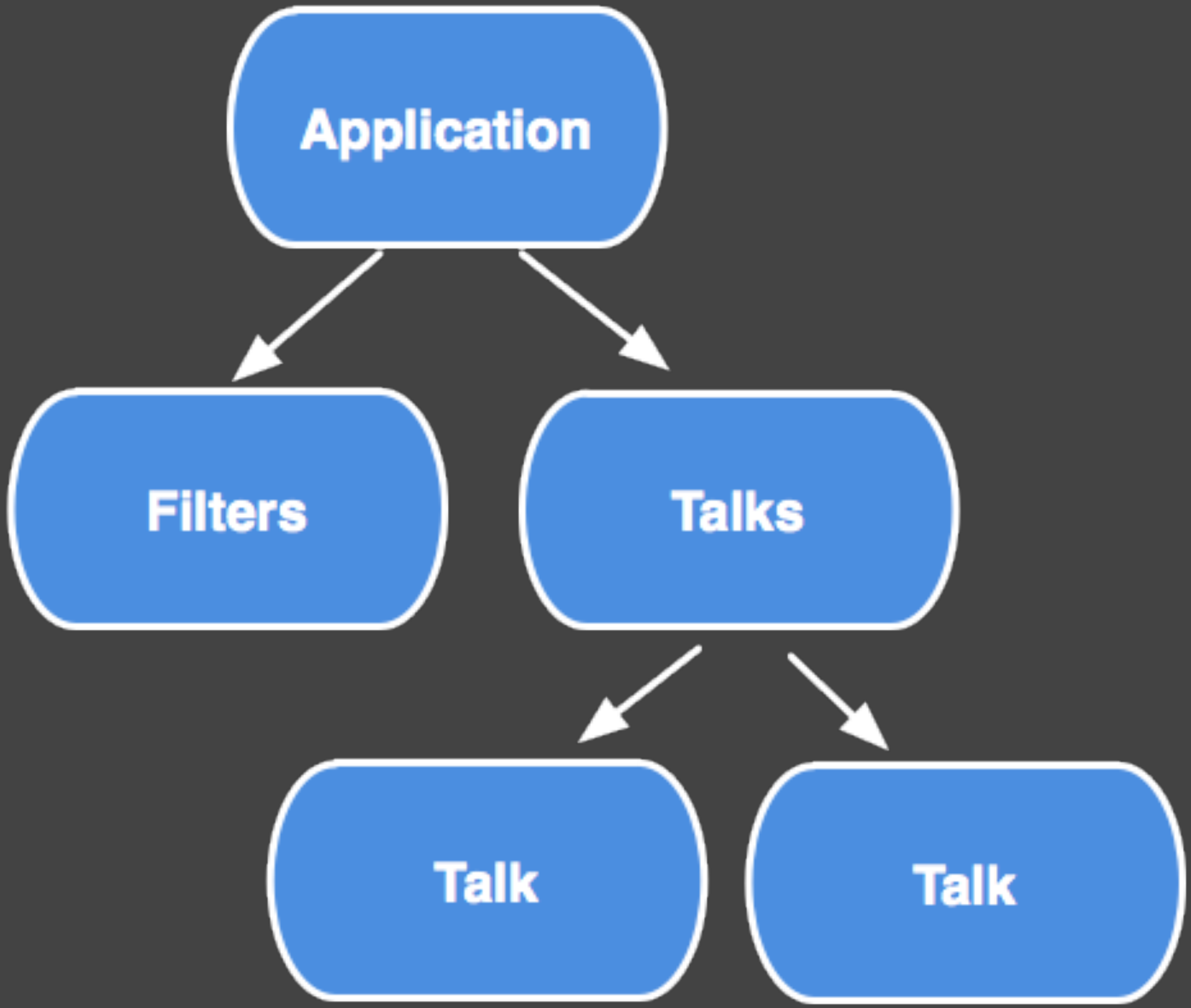
```
@Component({
  selector: 'talk-cmp',
  properties: ['talk'],
  events: ['rate']
})
@View({
  directives: [CoreDirectives],
  template: `
    Rating {{talk.avgRating}} {{talk.name}}
    <span *if="talk.speaker">Speaker: {{talk.speaker}}</span>
    <button (click)="triggerRate()">Rate</button>
  `
})
class TalkCmp {
  constructor() {
    this.rate = new EventEmitter();
  }

  triggerRate() {
    var rating = 1 + Math.floor(Math.random() * 9);
    this.rate.next({rating: rating});
  }
}
```

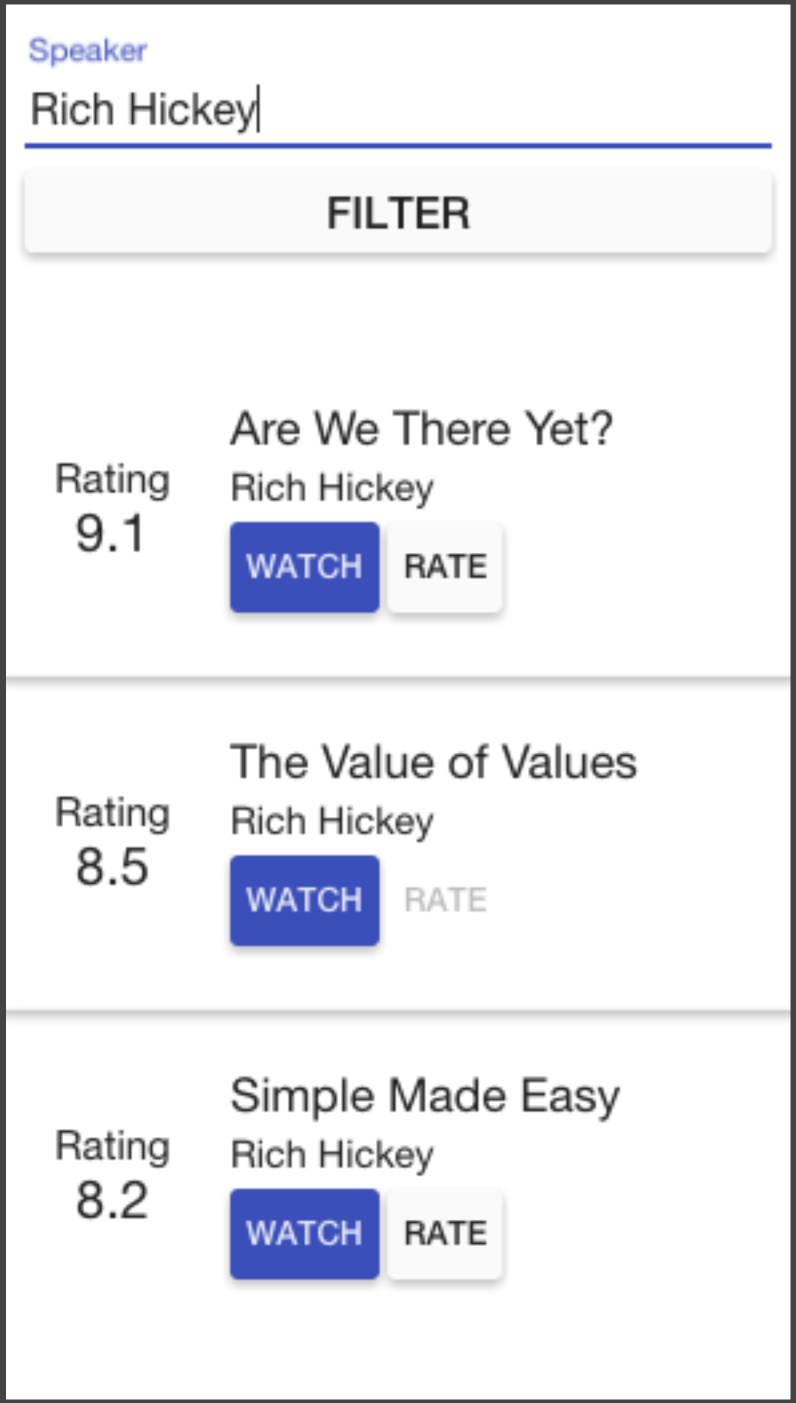


```
@Component({
  selector: 'talks-app'
})
@View({
  directives: [CoreDirectives, TalkCmp],
  template: `
    <h2>Talks</h2>
    <talk-cmp *for="var t of talks" [talk]="t" (rate)="rateTalk(t, $event.rating)"></talk-cmp>
  `
})
class TalksApp {
  constructor() {
    this.talks = [
      { name: 'Are we there yet?', speaker: 'Rich Hickey', avgRating: 0, ratings: [] },
      { name: 'The value of values', speaker: 'Rich Hickey', avgRating: 0, ratings: [] },
      { name: 'Simple Made Easy', speaker: null, avgRating: 0, ratings: [] }
    ]
  }

  rateTalk(talk, rating) {
    talk.ratings.push(rating);
    talk.avgRating = Math.round(talk.ratings.reduce((a,b) => a + b) / talk.ratings.length);
  }
}

export function main() {
  bootstrap(TalksApp);
}
```


Components
Dependency Injection
Property Bindings

Components





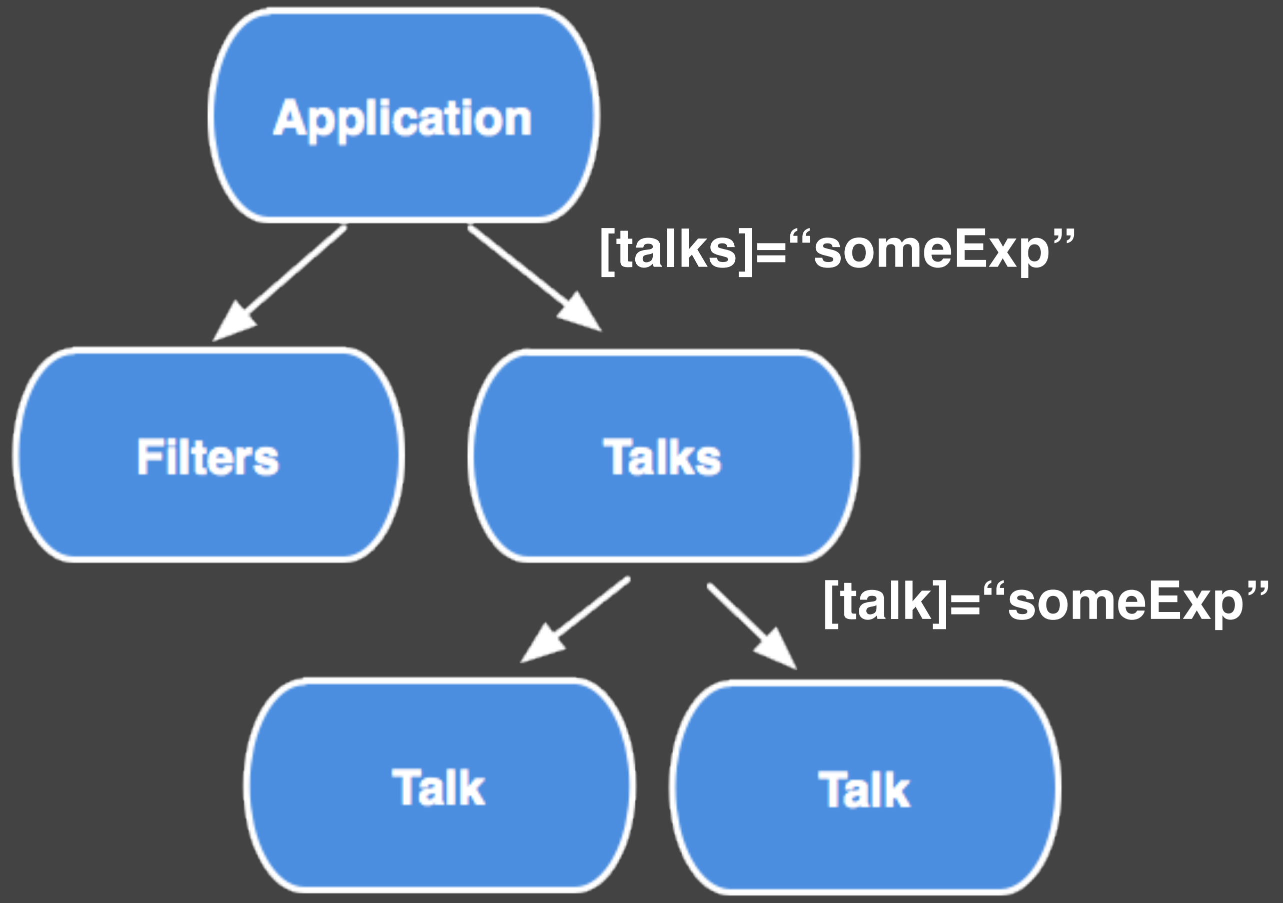
Rating 9.1 — Are We There Yet? — Rich Hickey

Rating 8.5 — The Value of Values — Rich Hickey

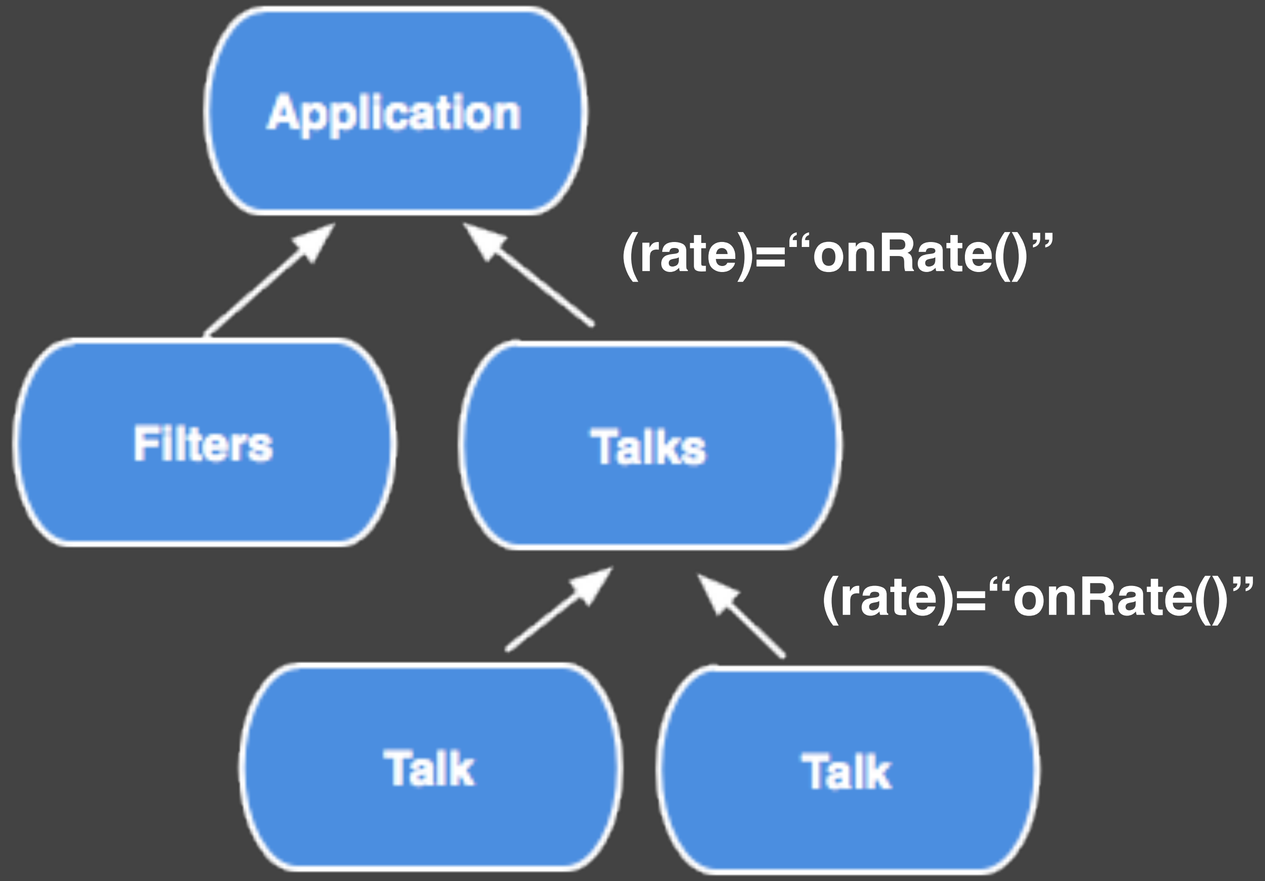
Rating 8.2 — Simple Made Easy — Rich Hickey

```
@Component({
  selector: 'talk-cmp',
  properties: ['talk'],
  events: ['rate'],
  lifecycle: []
})
@View({
  template: `
    {{talk.title}}
    {{talk.speaker}}
    <formatted-rating [rating]="talk.rating"></formatted-rating>
    <watch-button [talk]="talk"></watch-button>
    <rate-button [talk]="talk"></rate-button>
  `

})
class TalkCmp {
  talk: Talk;
  rate: EventEmitter;
  //...
}
```

Parent -> Child



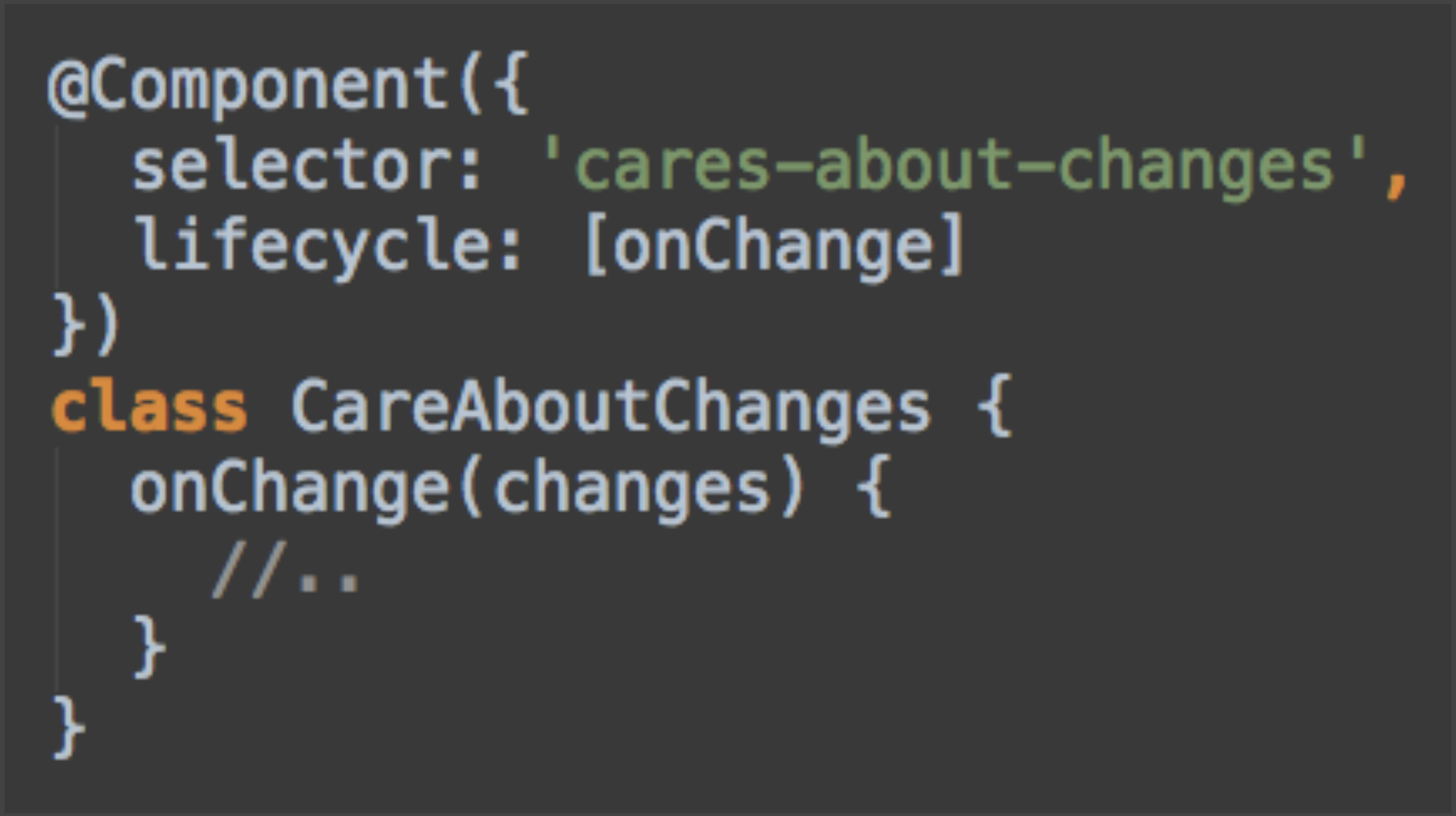
Child -> Parent



```
<talk-cmp [talk]="someExp" (rate)="onRate()"></talk-cmp>
```

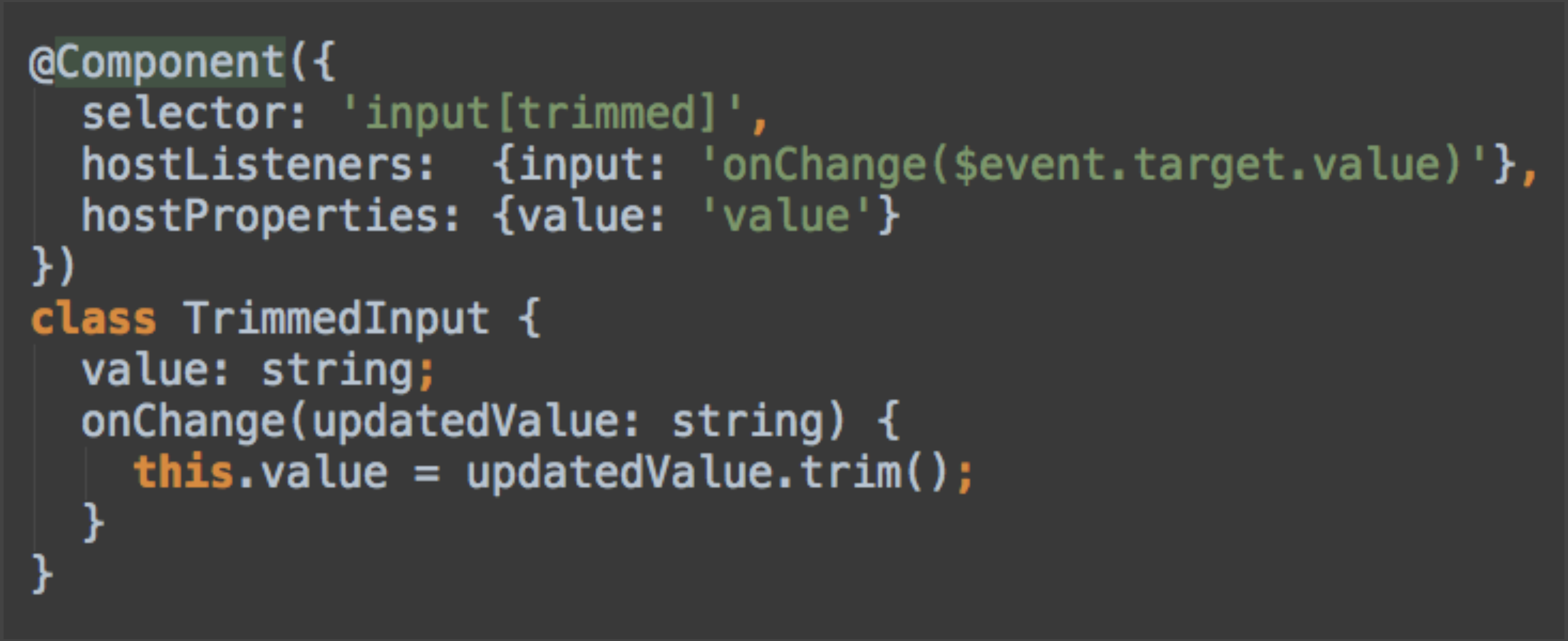
```
@Component({
  selector: 'talk-cmp',
  properties: ['talk'],
  events: ['rate'],
  lifecycle: []
})
@View({
  template: `
    {{talk.title}}
    {{talk.speaker}}
    <formatted-rating [rating]="talk.rating"></formatted-rating>
    <watch-button [talk]="talk"></watch-button>
    <rate-button [talk]="talk"></rate-button>
  `

})
class TalkCmp {
  talk: Talk;
  rate: EventEmitter;
  //...
}
```

```
@Component({
  selector: 'talk-cmp',
  properties: ['talk'],
  events: ['rate'],
  lifecycle: []
})
@View({
  template: `
    {{talk.title}}
    {{talk.speaker}}
    <formatted-rating [rating]="talk.rating"></formatted-rating>
    <watch-button [talk]="talk"></watch-button>
    <rate-button [talk]="talk"></rate-button>
  `
})
class TalkCmp {
  talk: Talk;
  rate: EventEmitter;
  //...
}
```



```
@Component({
  selector: 'cares-about-changes',
  lifecycle: [onChange]
})
class CareAboutChanges {
  onChange(changes) {
    //..
  }
}
```

```
@Component({
  selector: 'conf-app',
  injectables: [ConfAppBackend, Logger]
})
class TalksApp {
  //...
}

class TalksCmp {
  constructor(backend:ConfAppBackend) {
    //...
  }
}
```


```
@Component({
  selector: 'input[trimmed]',
  hostListeners:  {input: 'onChange($event.target.value)'},
  hostProperties: {value: 'value'}
})
class TrimmedInput {
  value: string;
  onChange(updatedValue: string) {
    this.value = updatedValue.trim();
  }
}
```

Components are Self-Describing

Used Directly

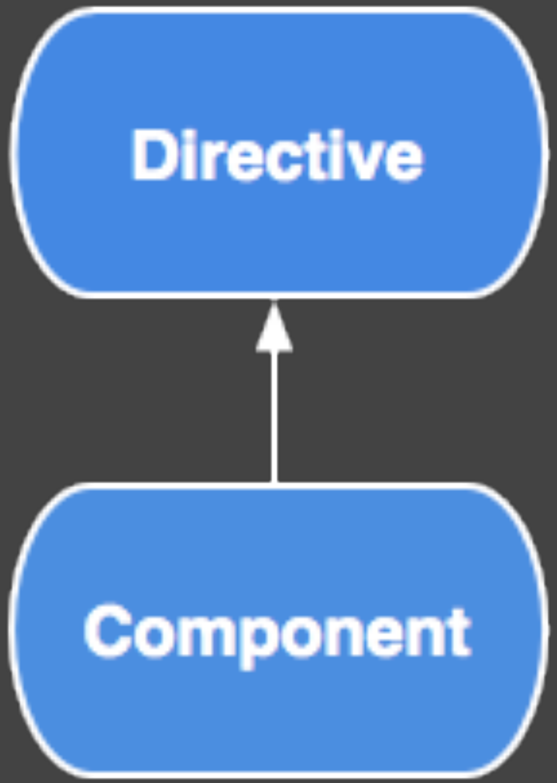
```
<conf-app></conf-app>
```

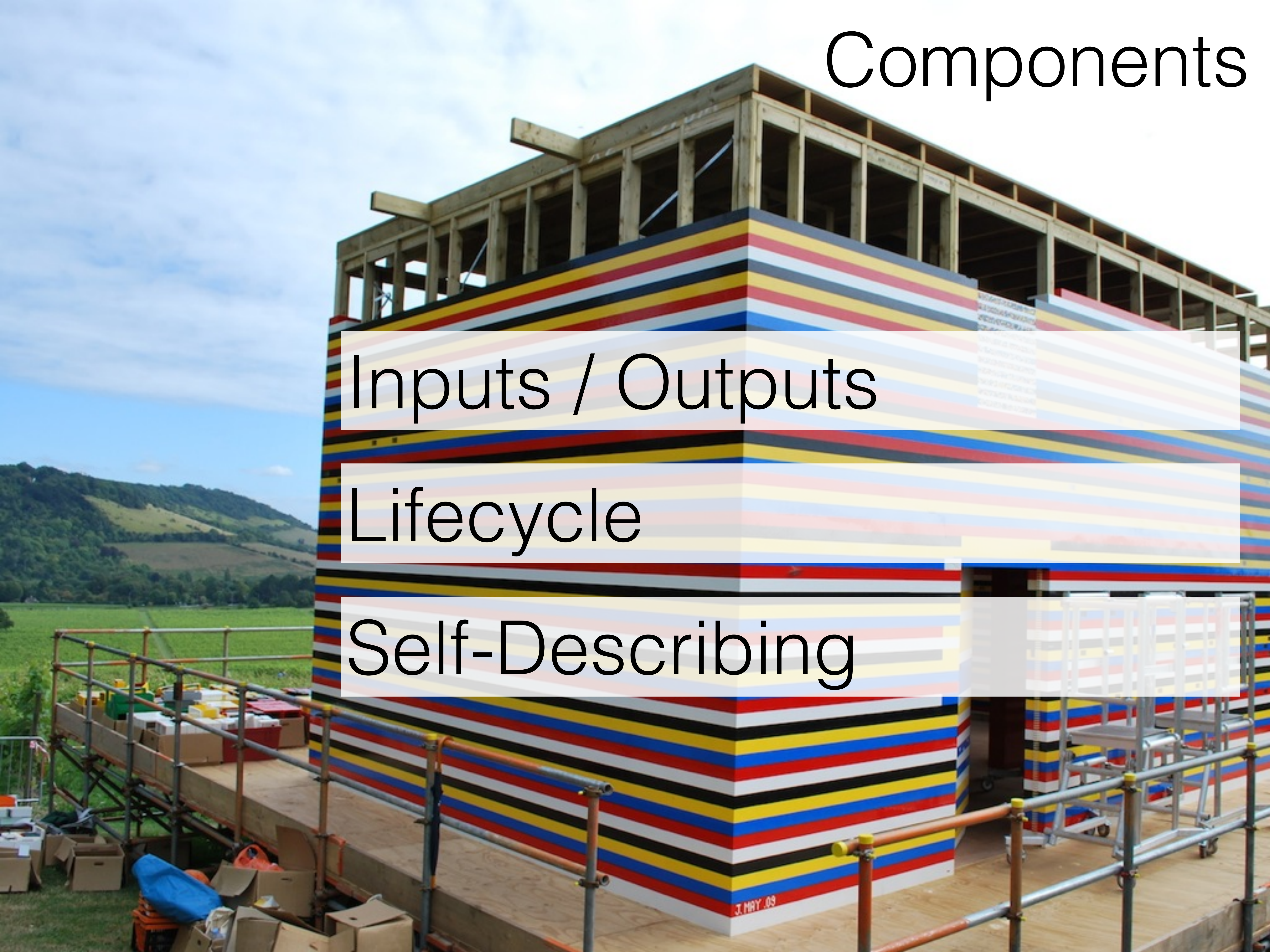
As an Application or Route

```
bootstrap(ConfApp);
```

What Happened to Directives?

Components are Directives





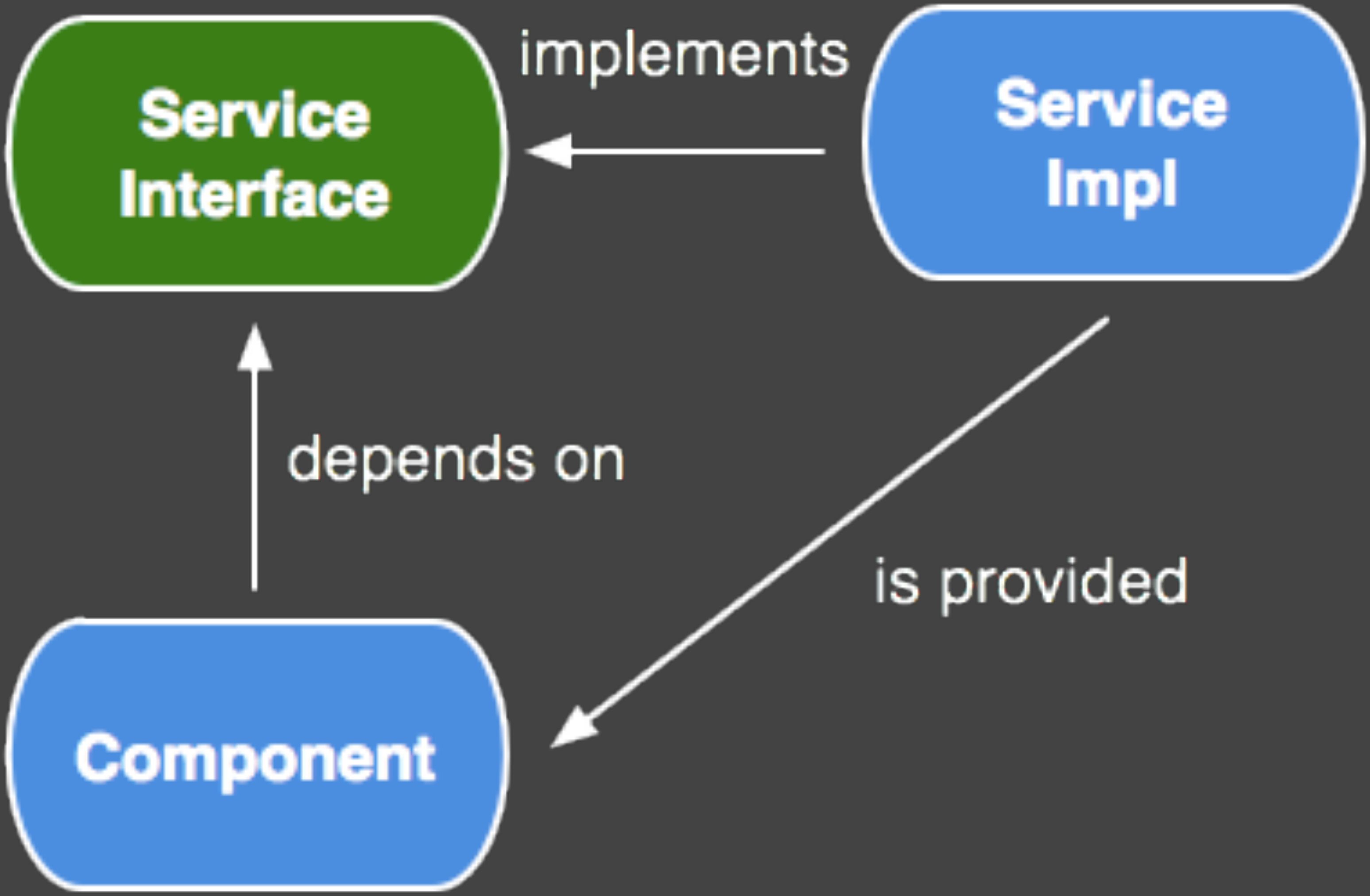
Components

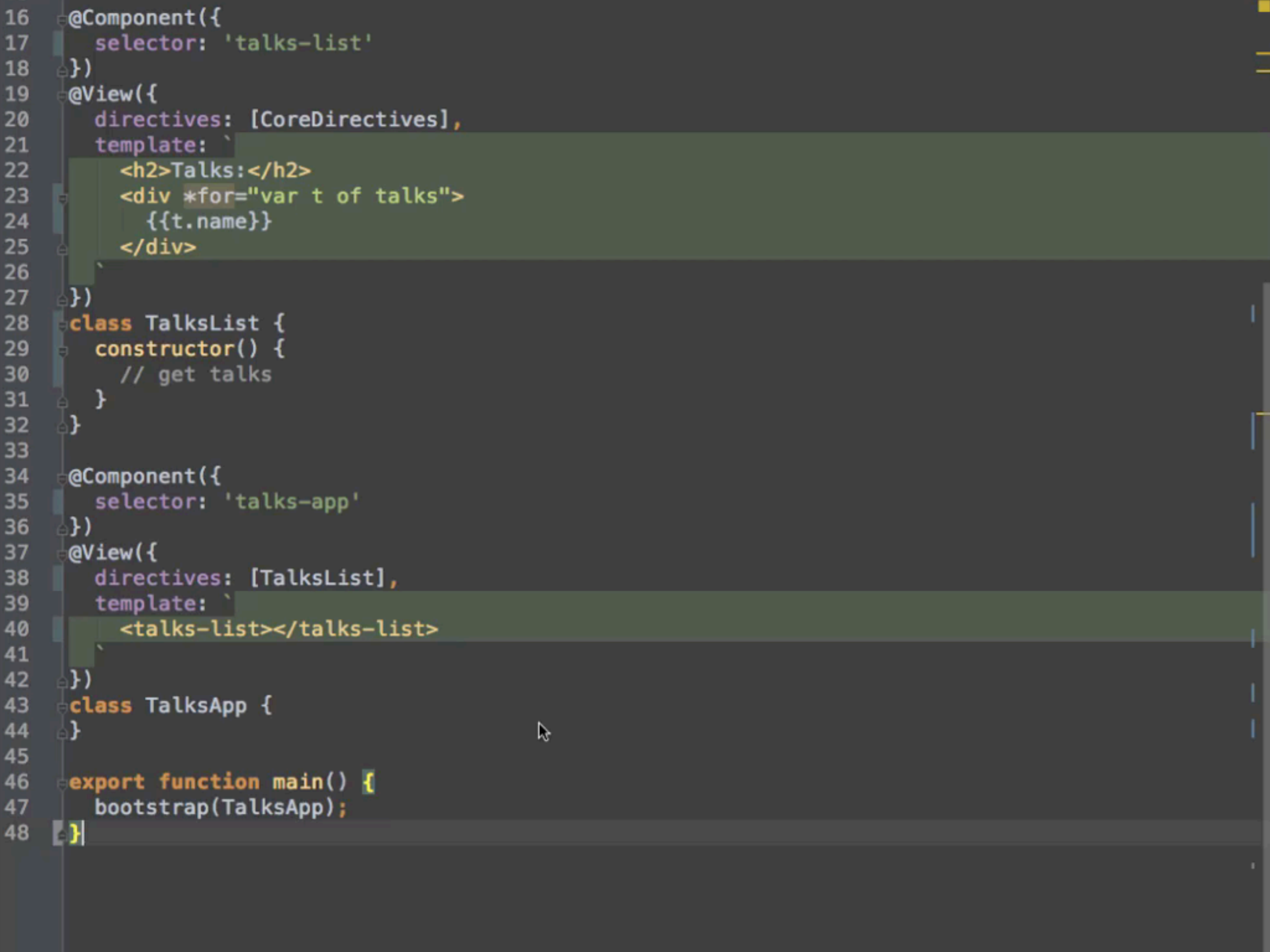
Inputs / Outputs

Lifecycle

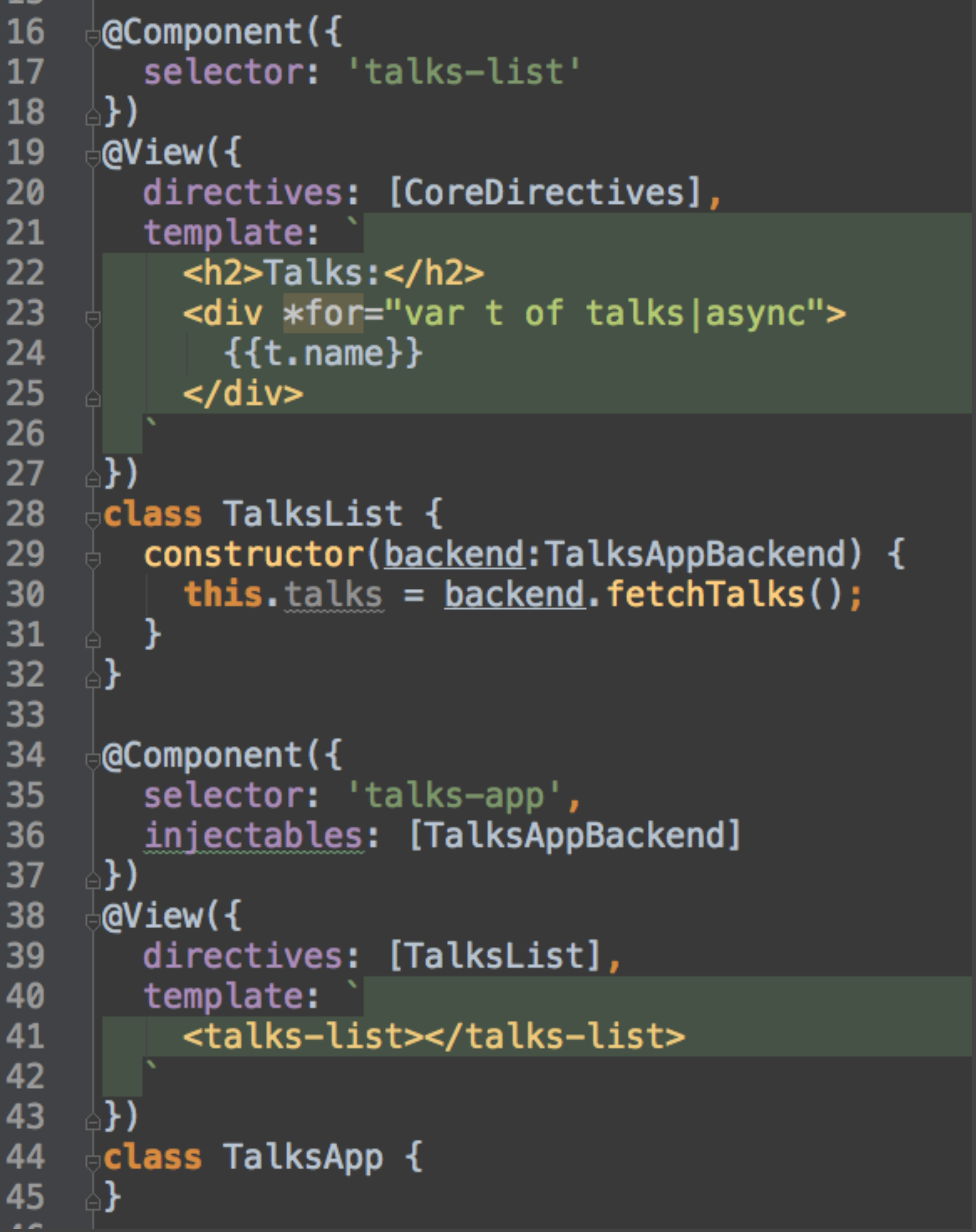
Self-Describing

Dependency Injection


Service Interface
implements
Service Impl
depends on
is provided
Component

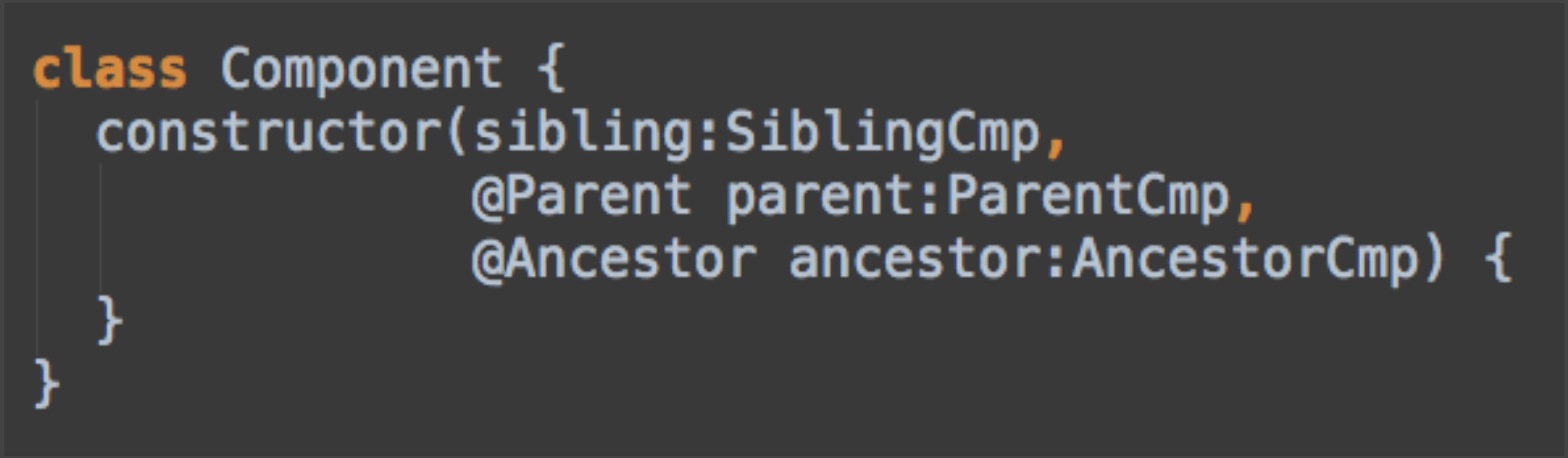


```
@Component({
  selector: 'talks-list'
})
@View({
  directives: [CoreDirectives],
  template: `
    <h2>Talks:</h2>
    <div *for="var t of talks">
      {{t.name}}
    </div>
  `
})
class TalksList {
  constructor() {
    // get talks
  }
}

@Component({
  selector: 'talks-app'
})
@View({
  directives: [TalksList],
  template: `
    <talks-list></talks-list>
  `
})
class TalksApp {
}

export function main() {
  bootstrap(TalksApp);
}
```

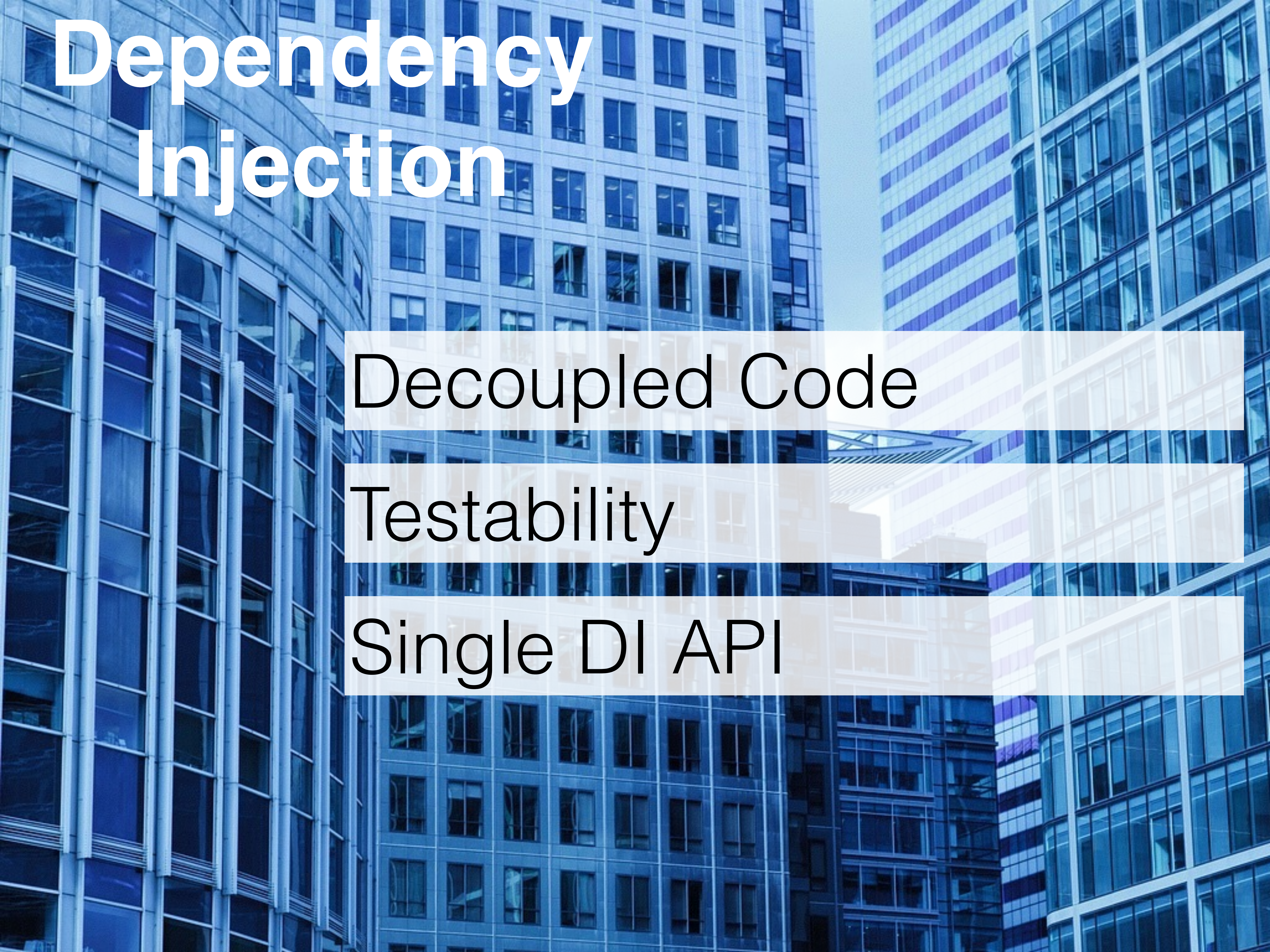


```
@Component({
  selector: 'talks-list'
})
@View({
  directives: [CoreDirectives],
  template: `
    <h2>Talks:</h2>
    <div *for="var t of talks|async">
      {{t.name}}
    </div>
  `
})
class TalksList {
  constructor(backend:TalksAppBackend) {
    this.talks = backend.fetchTalks();
  }
}

@Component({
  selector: 'talks-app',
  injectables: [TalksAppBackend]
})
@View({
  directives: [TalksList],
  template: `
    <talks-list></talks-list>
  `
})
class TalksApp {
}
```

```
class TalksCmp {
  constructor(elRef:ElementRef, backend:ConfAppBackend) {
  }
}
```



```
class Component {
  constructor(sibling:SiblingCmp,
              @Parent parent:ParentCmp,
              @Ancestor ancestor:AncestorCmp) {
  }
}
```



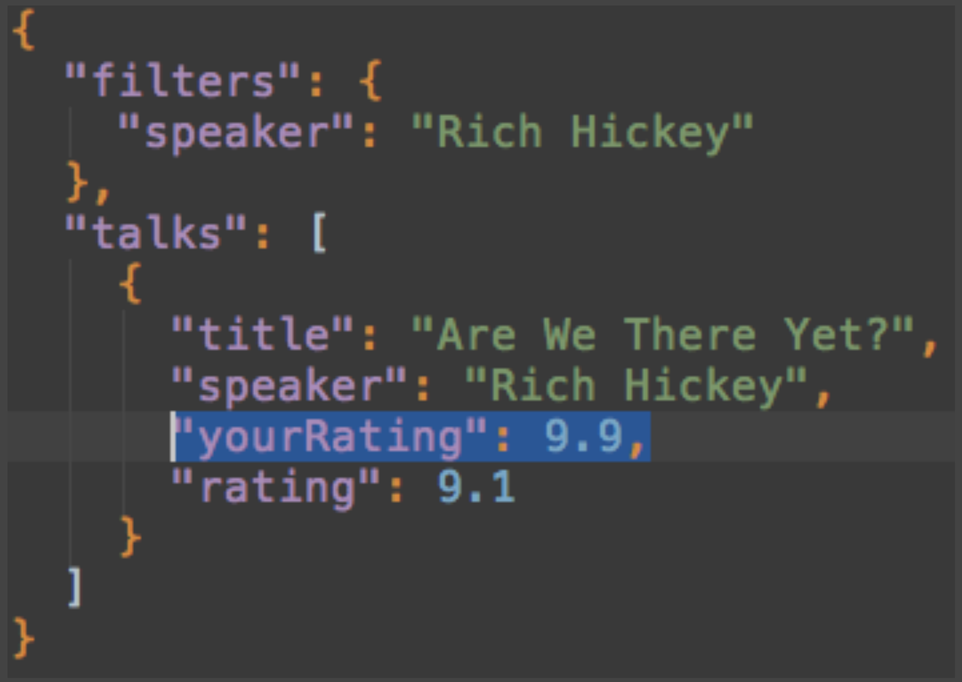
Dependency Injection

Decoupled Code

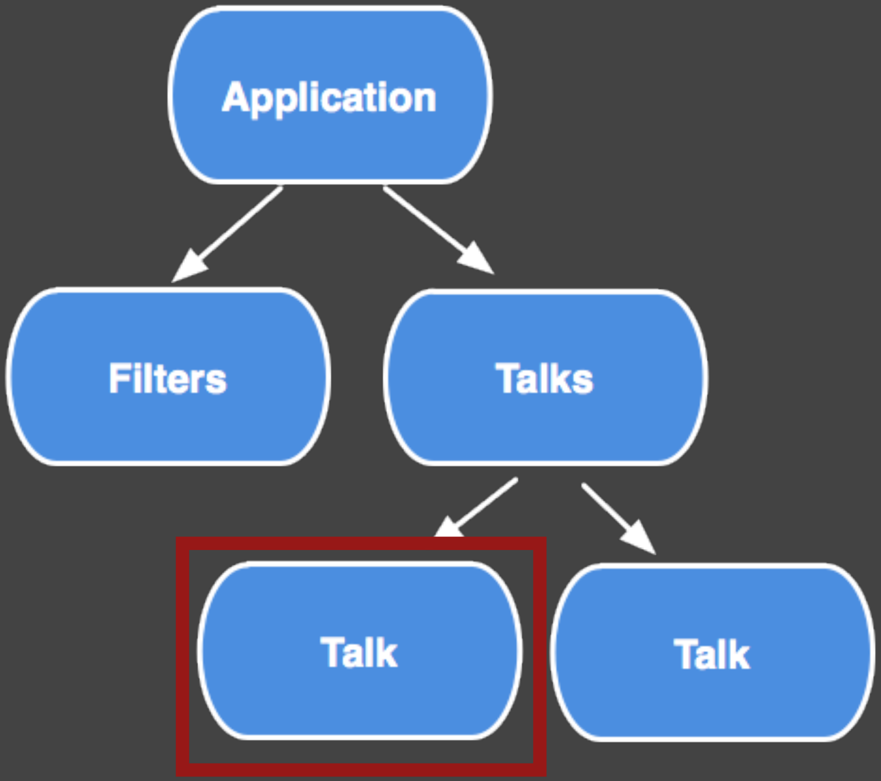
Testability

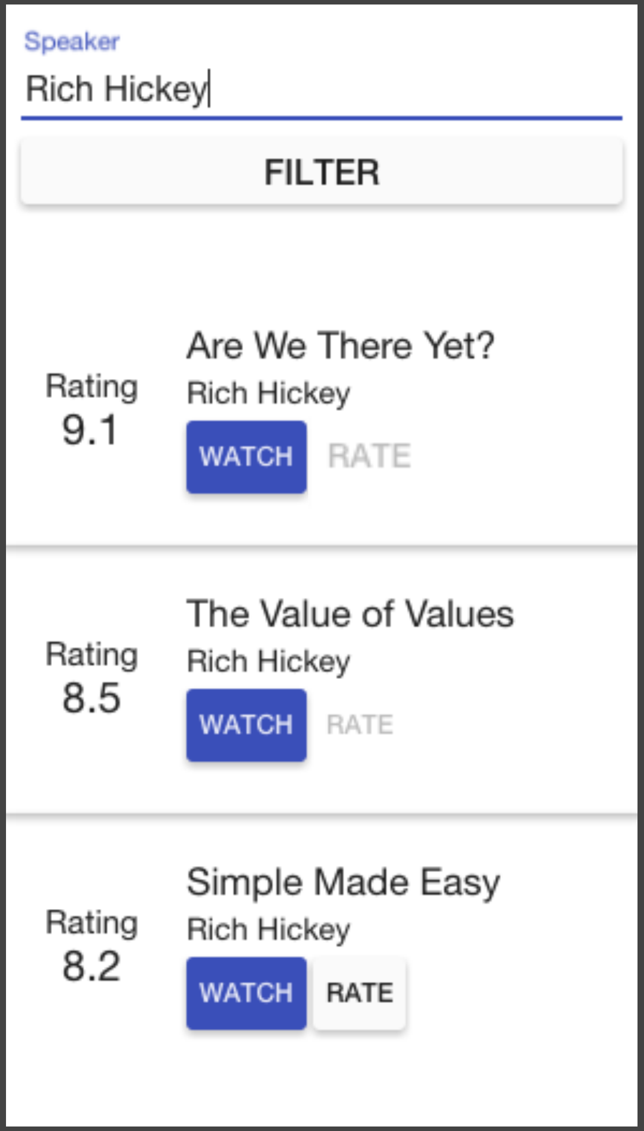
Single DI API

Property Bindings



```
{
  "filters": {
    "speaker": "Rich Hickey"
  },
  "talks": [
    {
      "title": "Are We There Yet?",
      "speaker": "Rich Hickey",
      "yourRating": 9.9,
      "rating": 9.1
    }
  ]
}
```





Speaker

Rich Hickey

FILTER

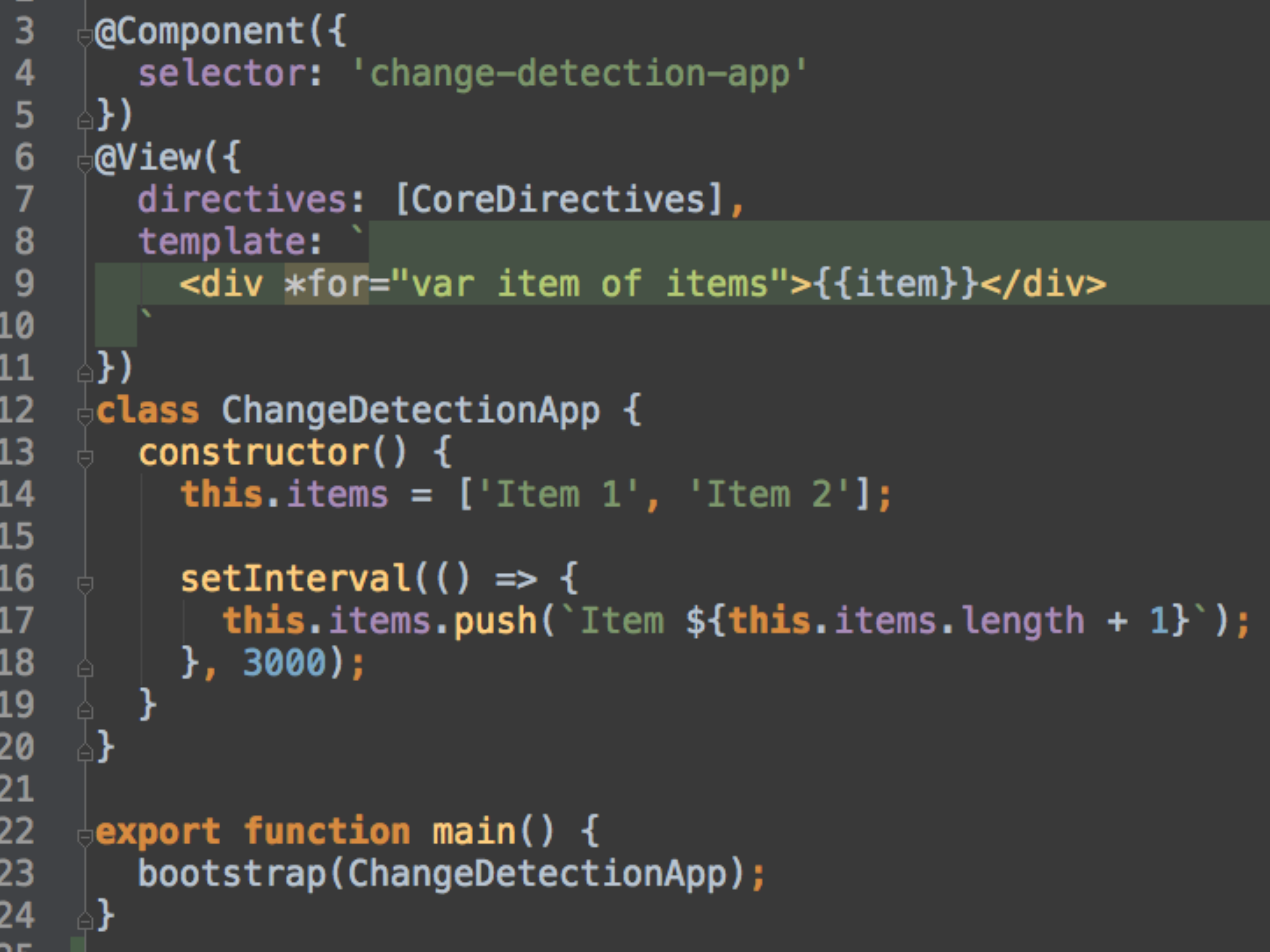
Rating 9.1 — Are We There Yet? — Rich Hickey — WATCH RATE

Rating 8.5 — The Value of Values — Rich Hickey — WATCH RATE

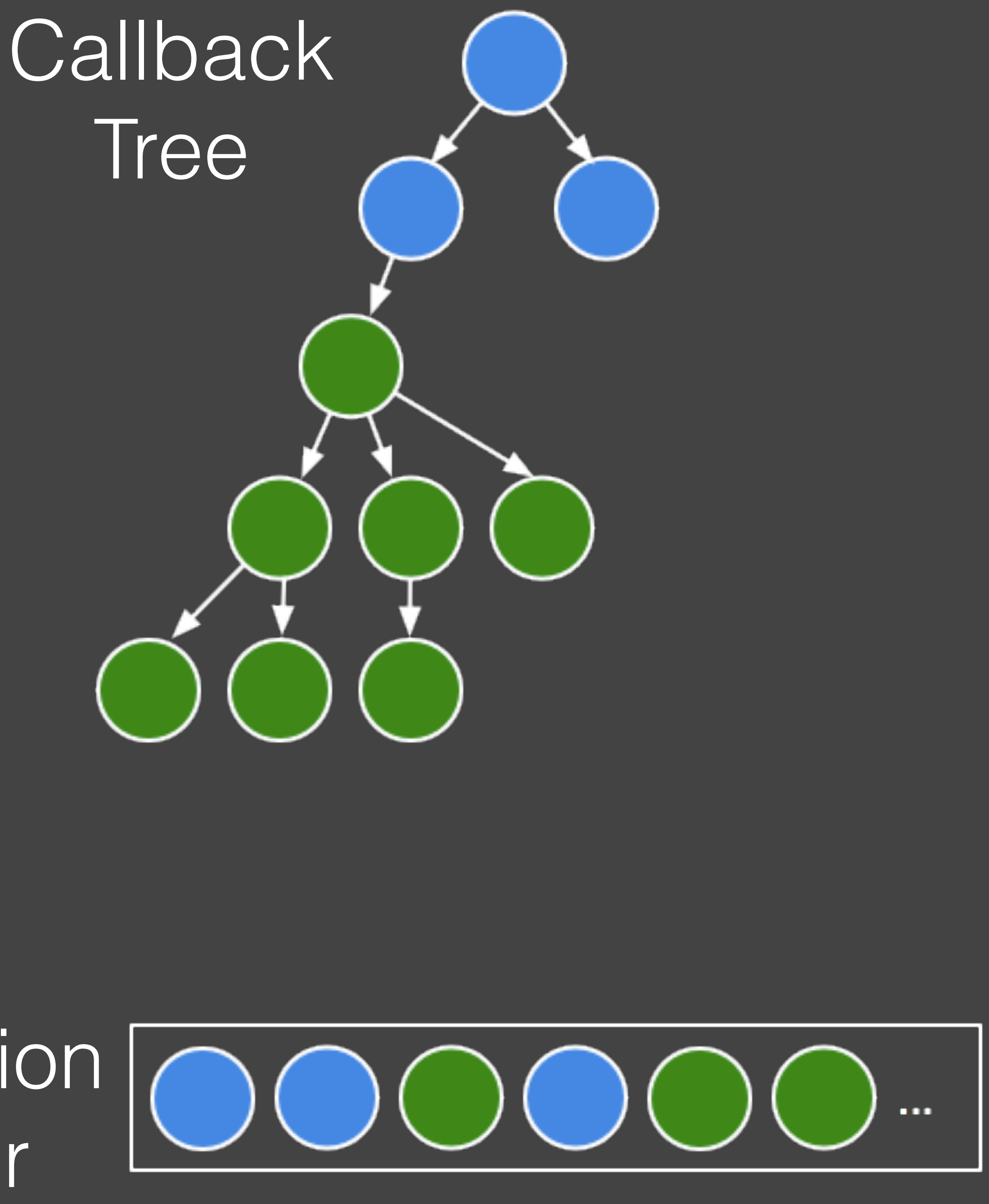
Rating 8.2 — Simple Made Easy — Rich Hickey — WATCH RATE

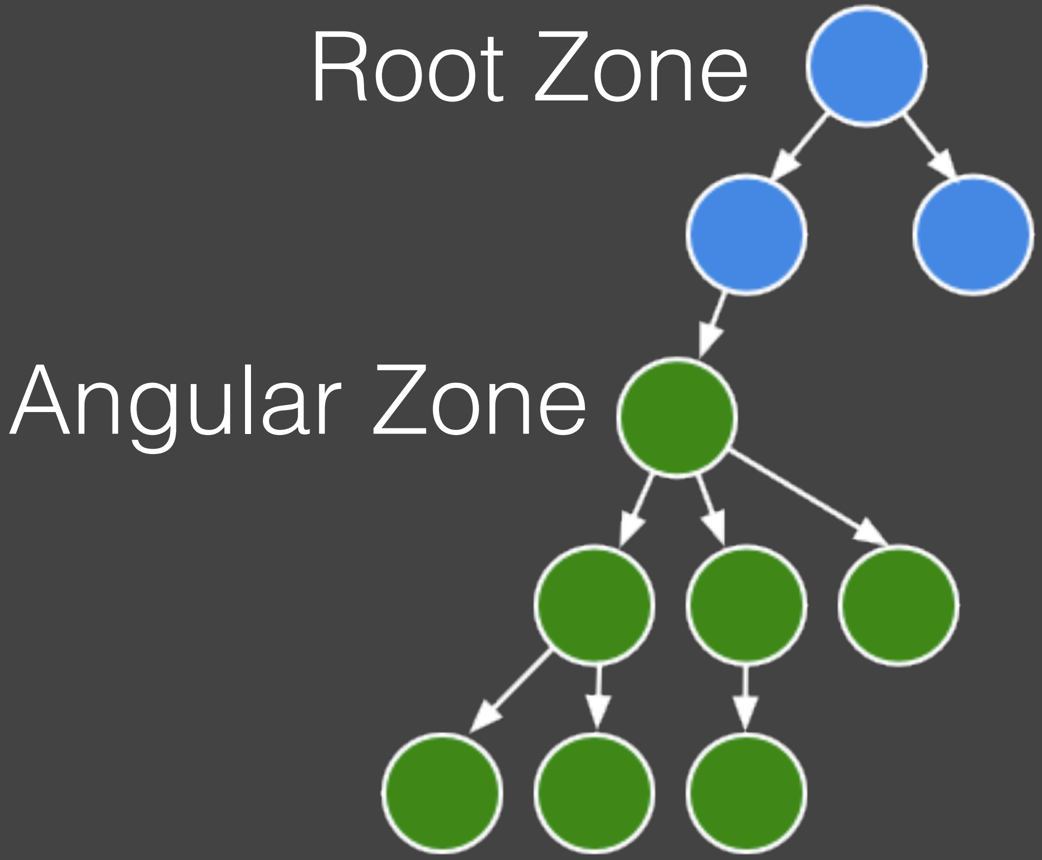
$$\begin{tikzcd}\text{Model} \arrow[r] & \text{Components} \arrow[r] & \text{DOM}\end{tikzcd}$$

```
import {Component, View, bootstrap, EventEmitter, CoreDirectives} from 'angular2/angular2


```

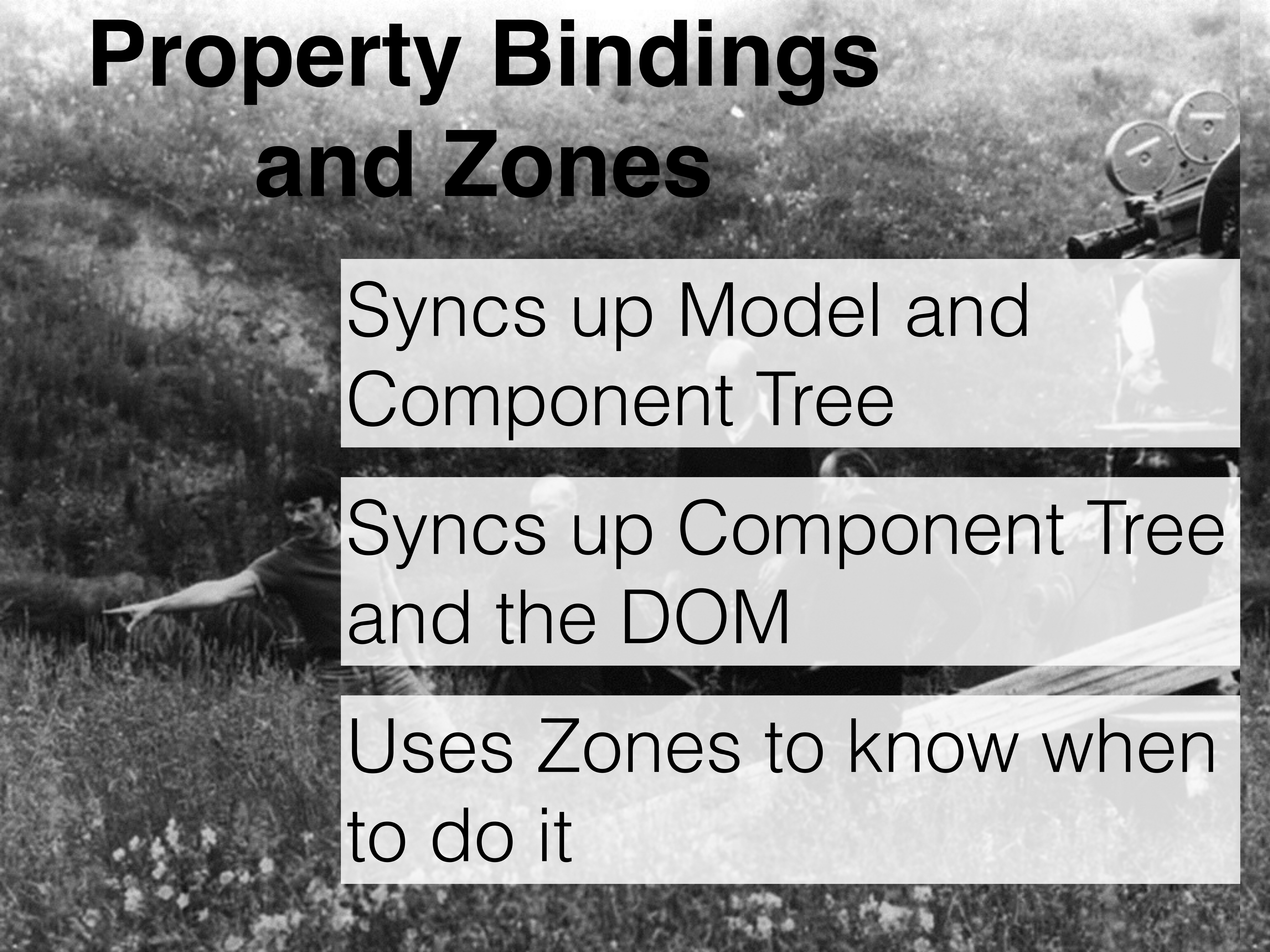


```
@Component({
  selector: 'change-detection-app'
})
@View({
  directives: [CoreDirectives],
  template: `
    <div *for="var item of items">{{item}}</div>
  `
})
class ChangeDetectionApp {
  constructor() {
    this.items = ['Item 1', 'Item 2'];

    setInterval(() => {
      this.items.push(`Item ${this.items.length + 1}`);
    }, 3000);
  }
}

export function main() {
  bootstrap(ChangeDetectionApp);
}
```


Callback
Tree
Execution
Order
...


Root Zone
Angular Zone

No More scope.$apply



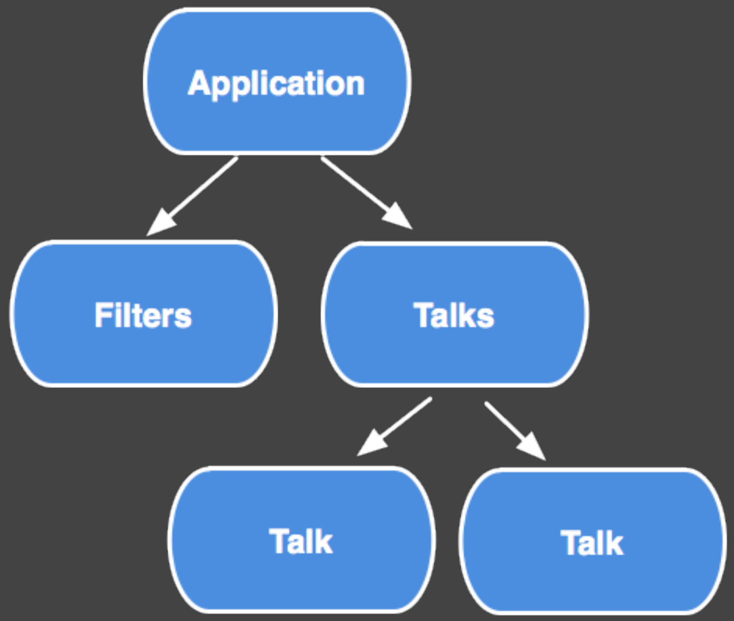
Property Bindings and Zones

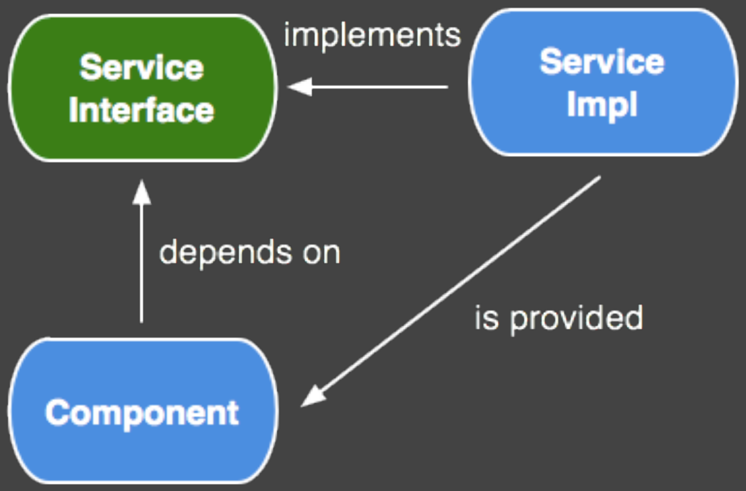
Syncs up Model and Component Tree

Syncs up Component Tree and the DOM

Uses Zones to know when to do it

Summary

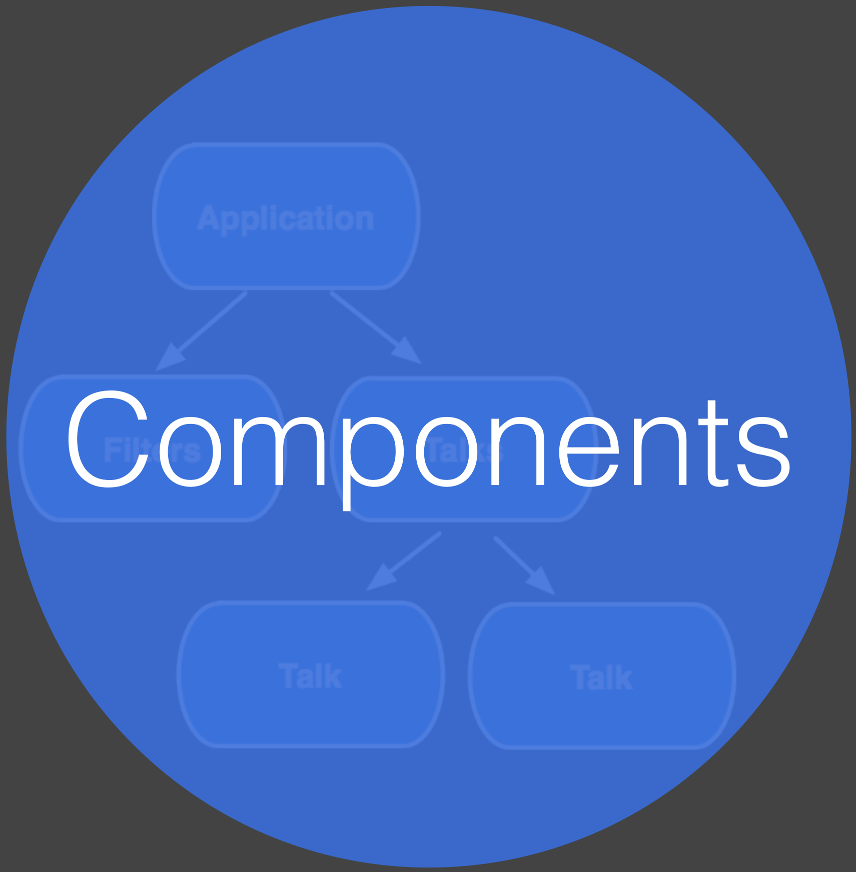




$$\begin{tikzcd}\text{Model} \arrow[d] \\ \text{Components} \arrow[d] \\ \text{DOM}\end{tikzcd}$$

- Form Handling
- Data Access
- Router
- Animations

- Material Components
- Unit Testing
- E2E Testing



Components

Dependency Injection

Property Bindings & Zones

Learn More

Angular 2 Step by Step Guide
https://angular.io/docs/js/latest/guide/

Brian Ford on Zones
https://www.youtube.com/watch?v=3IqtmUscE_U

Victor Savkin's Blog
http://victorsavkin.com

Thank You!

@victorsavkin