Mad science with the

Angular Compiler

twitter.com/mgechev
github.com/mgechev
blog.mgechev.com

Minko Gechev

mgechev

Functional time traveler.

California, USA

http://blog.mgechev.com/

Organizations

Overview | Repositories 190 | Stars 96 | Followers 2.9k

angular-style-guide

Set of best practices for Angular application development

★ 4.9k 712

angular-seed

Modular starter project for Angular 2 (and beyond) with statically typed build and AoT compilation

TypeScript ★ 3.9k 1.5k

codelyzer

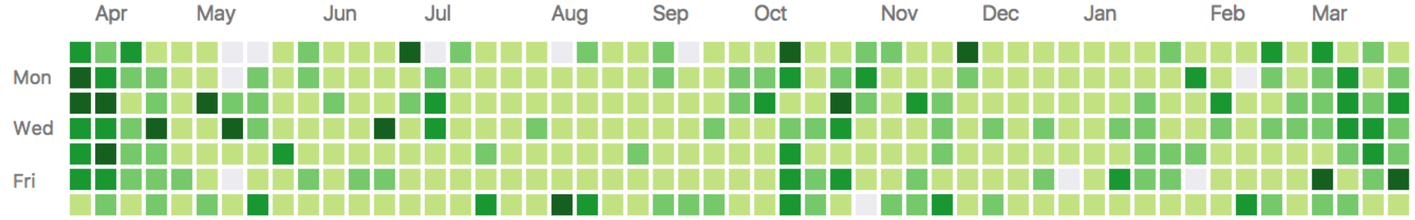
Linting for Angular projects.

TypeScript ★ 1k 91

angular/mobile-toolkit

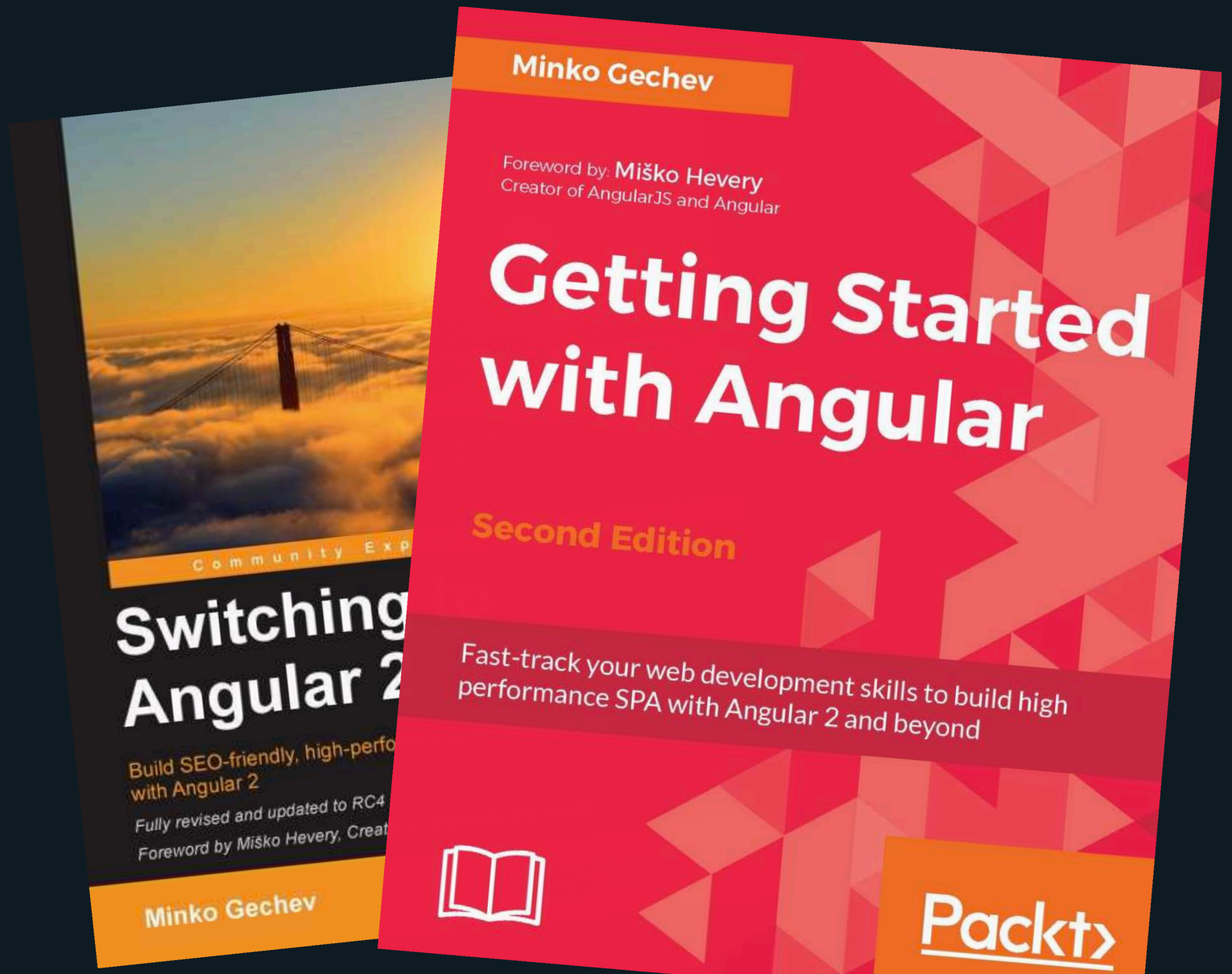
JavaScript implementation of different computer science algorithms.

TypeScript ★ 982 130



twitter.com/**mgechev**

github.com/**mgechev**



EBGES050

50% off ebook

PBGSA15

15% off book

twitter.com/mgechev

The compiler transforms the source code of an Angular application for achieving higher efficiency.


```
@Component({
  selector: 'header-cmp',
  template: `
    <header>
     {{ title }}
    </header>
  `
})
class HeaderComponent { ... }
```


```
@Component({
  selector: 'header-cmp',
  template: `
    <header>
     {{ title }}
    </header>
  `
})
class HeaderComponent { ... }
```

```
class _View_HeaderComponent extends
  import1.AppView<import0.HeaderComponent> {
    constructor(...) {
      ...
    }
    createInternal(...):import3.AppElement {
      ...
    }
    injectorGetInternal(...):any {
      ...
    }
  }
```

Angular Compiler

Generates...

- Efficient views
- Efficient providers instantiation

Transformation

Is that all we can do?



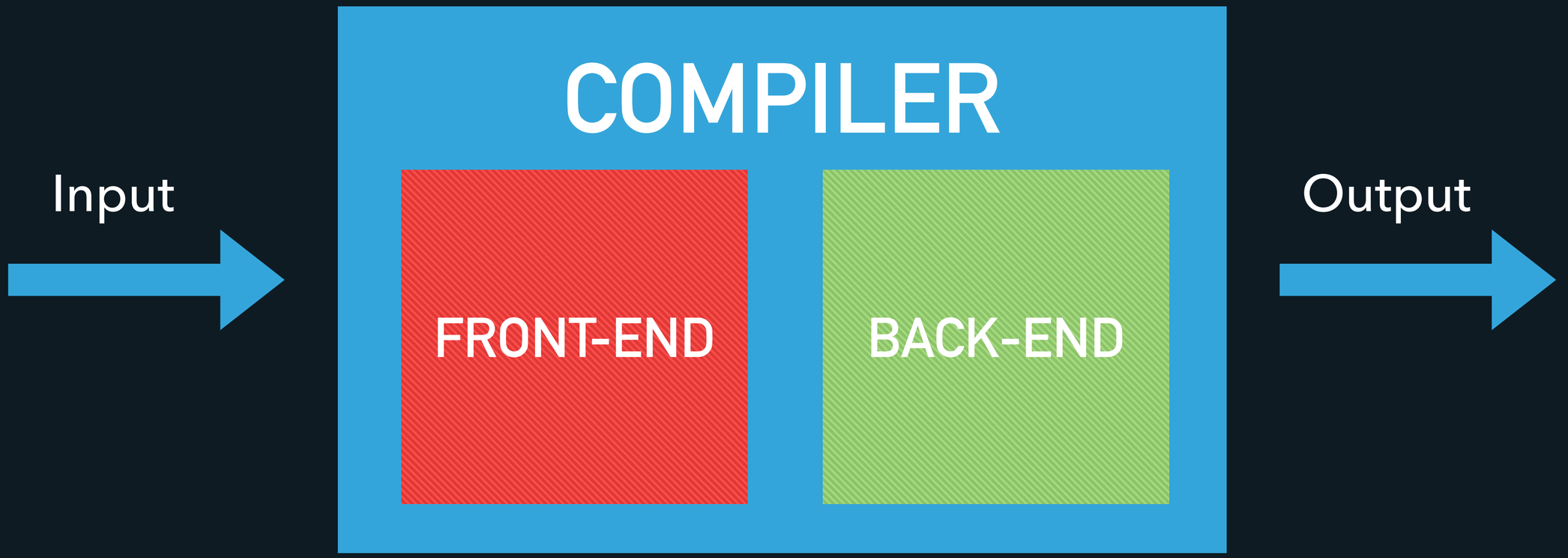

Of Course Not...

- Analyze the source code
- Visualize the source code
- Much more...


How a compiler works?

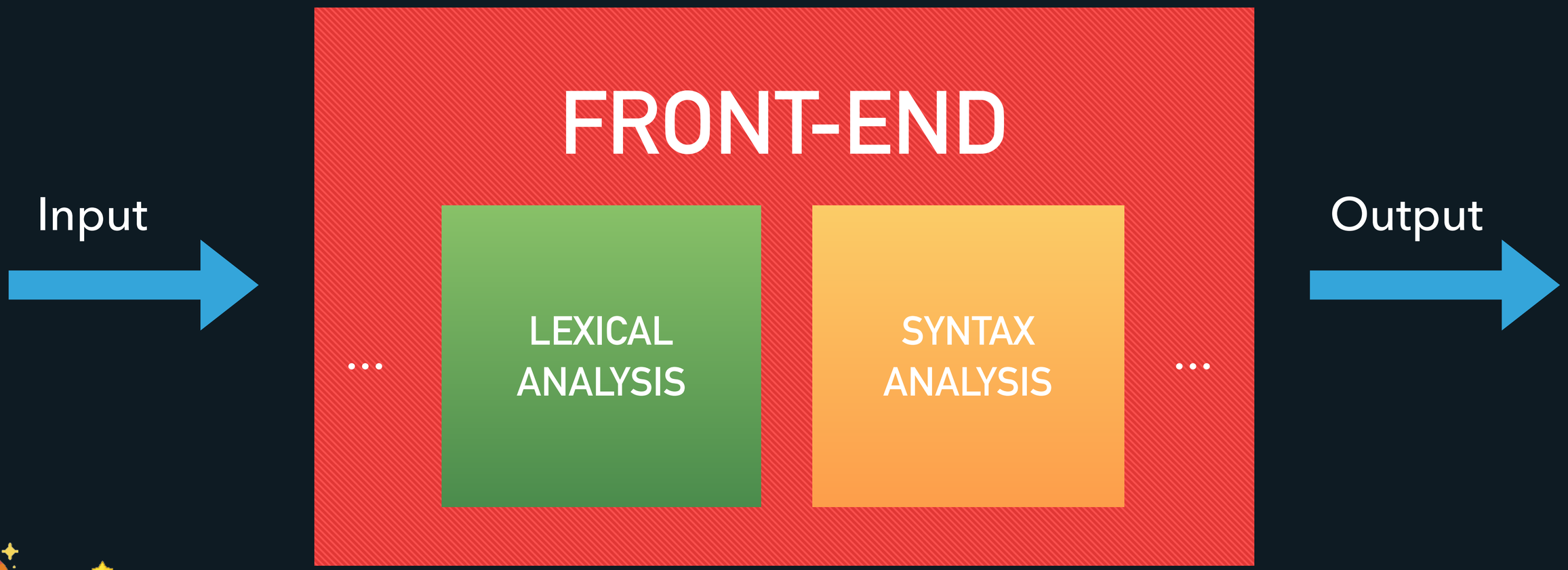

Compiler

Compiler



Compiler

Compiler




Lexical Analysis

```
const product = a * b;
```

Lexical Analysis

```
tokenize('const product = a * b;')
```

Lexical Analysis

```
tokenize('const product = a * b;')
```

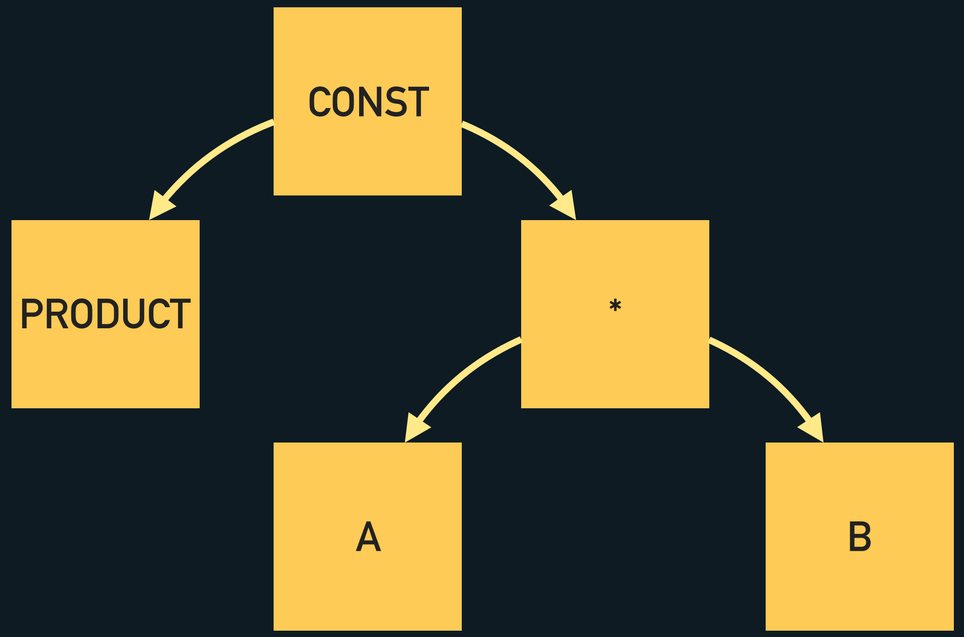
```
[
  { lexeme: 'const', type: 'keyword' },
  { lexeme: 'product', type: 'identifier' },
  { lexeme: '=', type: 'operator' },
  ...
]
```

Syntax Analysis

```
parse(tokenize('...'))
```

Syntax Analysis

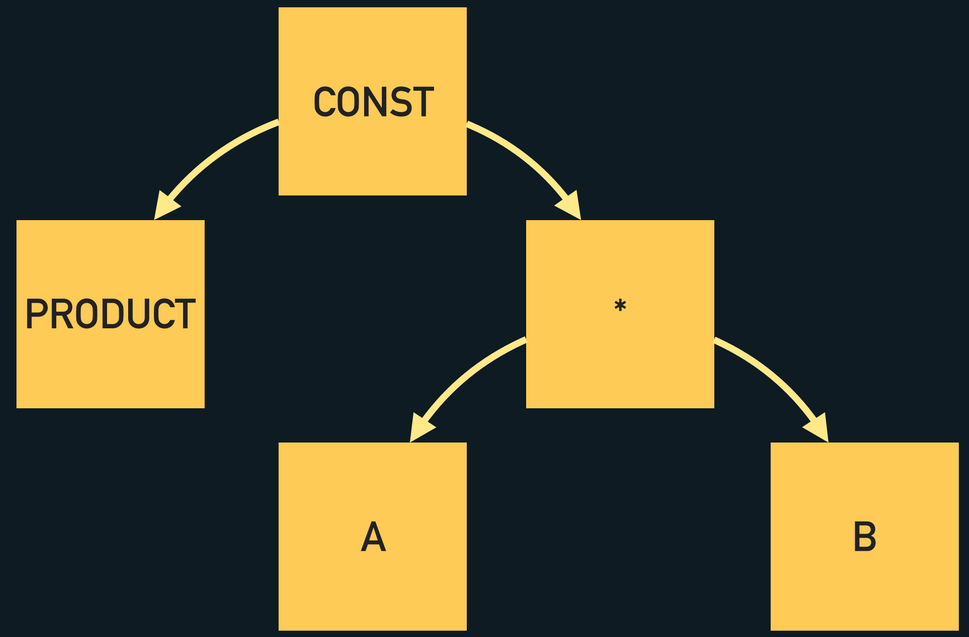
```
parse(tokenize('...'))
```



Syntax Analysis

```
parse(tokenize('...'))
```

AST




Static Analysis


twitter.com/mgechev



Static program analysis is the analysis of computer software that is performed without actually executing programs.

Wikipedia


github.com/mgechev

codelyzer

twitter.com/mgechev


tsc


twitter.com/mgechev


tslint


twitter.com/mgechev

Components as elements

STYLE 05-03

> **Do** give components an *element* selector, as opposed to *attribute* or *class* selectors.

> **Why?** components have templates containing HTML and optional Angular template syntax. They display content. Developers place components on the page as they would native HTML elements and WebComponents.

> **Why?** It is easier to recognize that a symbol is a component by looking at the template's html.


```
@Component({
  selector: '[tooltip]',
  template: '...'
})
class TooltipComponent { ... }
```

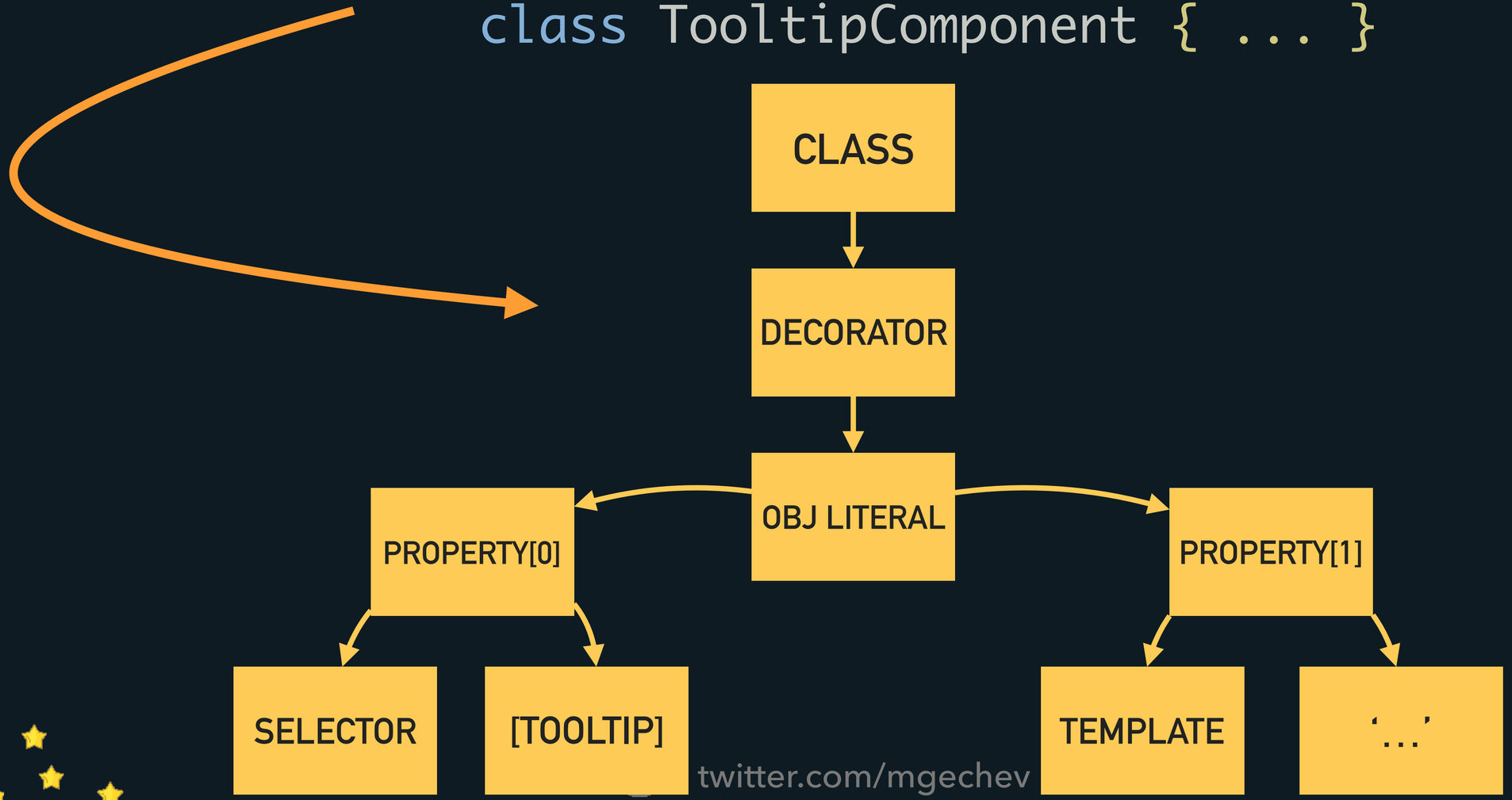


```
tsc.tokenize(`
   @Component({
      selector: '[tooltip]',
      template: '...'
   })
   class TooltipComponent { ... }
`)
```


```
ast = tsc.parse(
  tsc.tokenize(`
    @Component({
      selector: '[tooltip]',
      template: '...'
    })
    class TooltipComponent { ... }
  `)
);
```


```
@Component({
  selector: '[tooltip]',
  template: '...'
})
class TooltipComponent { ... }
```


```
@Component({
  selector: '[tooltip]',
  template: '...'
})
class TooltipComponent { ... }
```



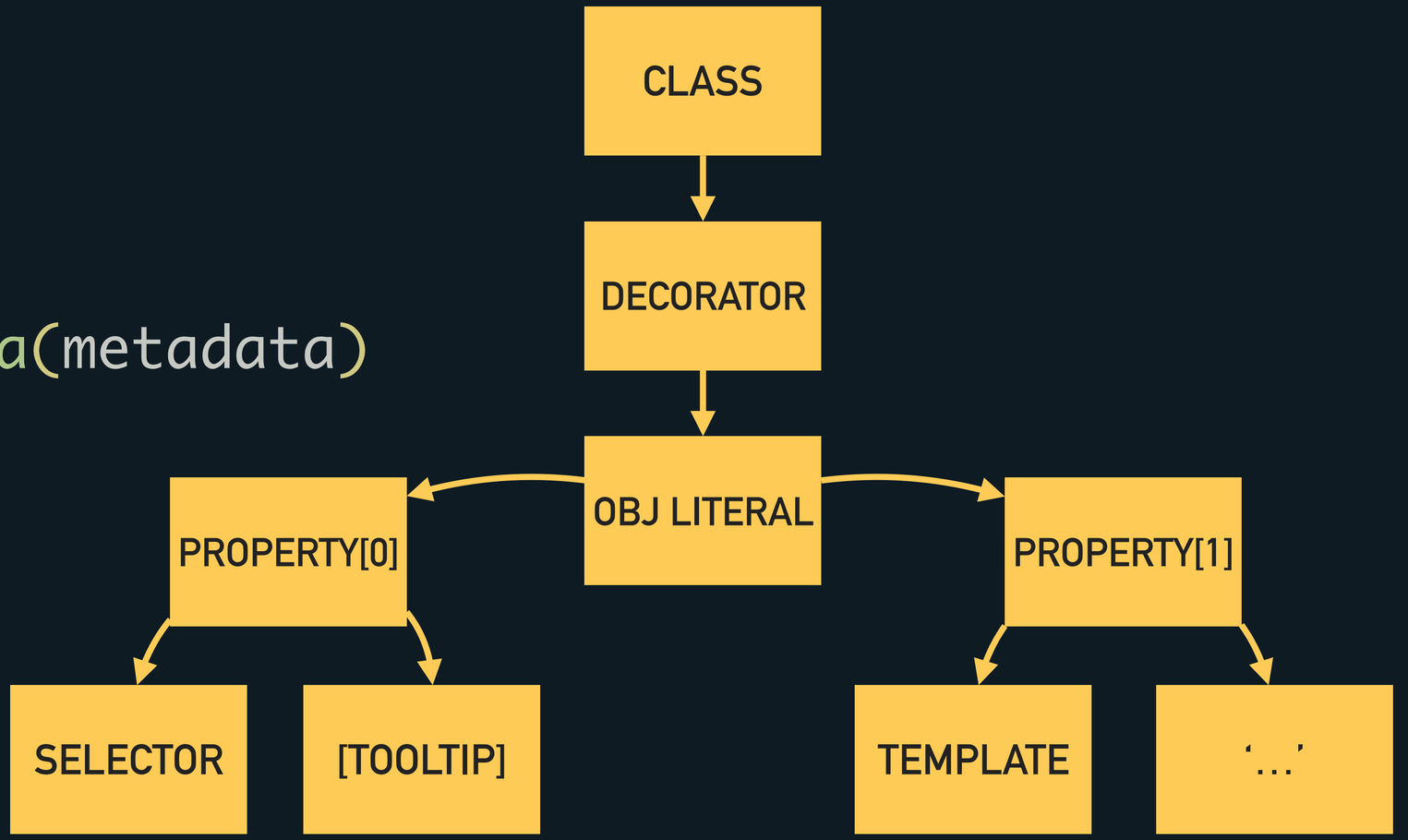
Validate Selector

```
visitClass(node) {
  node.decorators.forEach(visitDecorator);
}

visitDecorator(node) {
  if (node.name === 'Component') {
    visitComponentMetadata(readMetadata(node))
  }
}

visitComponentMetadata(metadata) {
  if (!parseSelector(metadata.selector).element) {
    tslint.report('Components should have element selectors');
  }
}
```



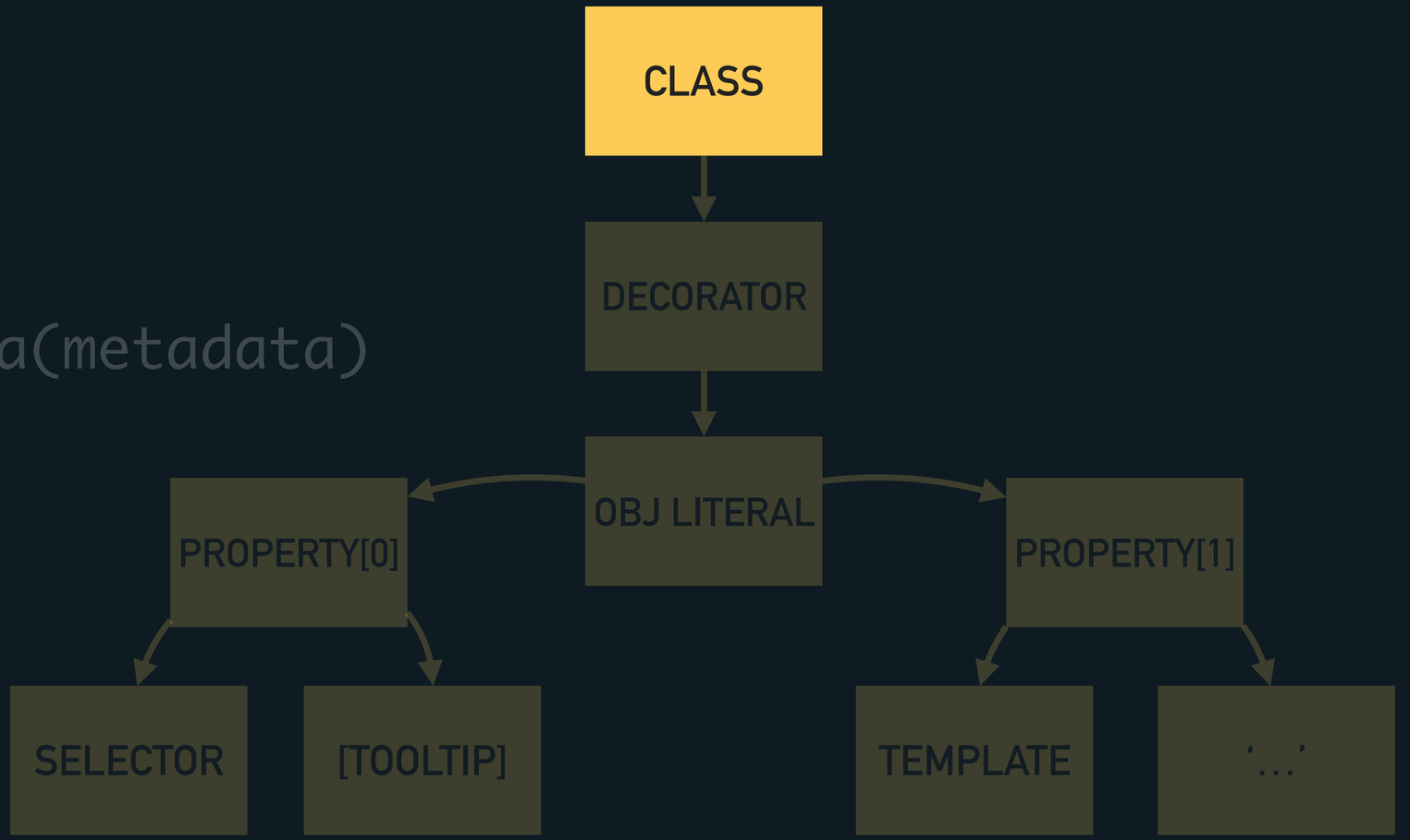

Validate Selector

```
visitClass(node)

visitDecorator(node)

visitComponentMetadata(metadata)
```



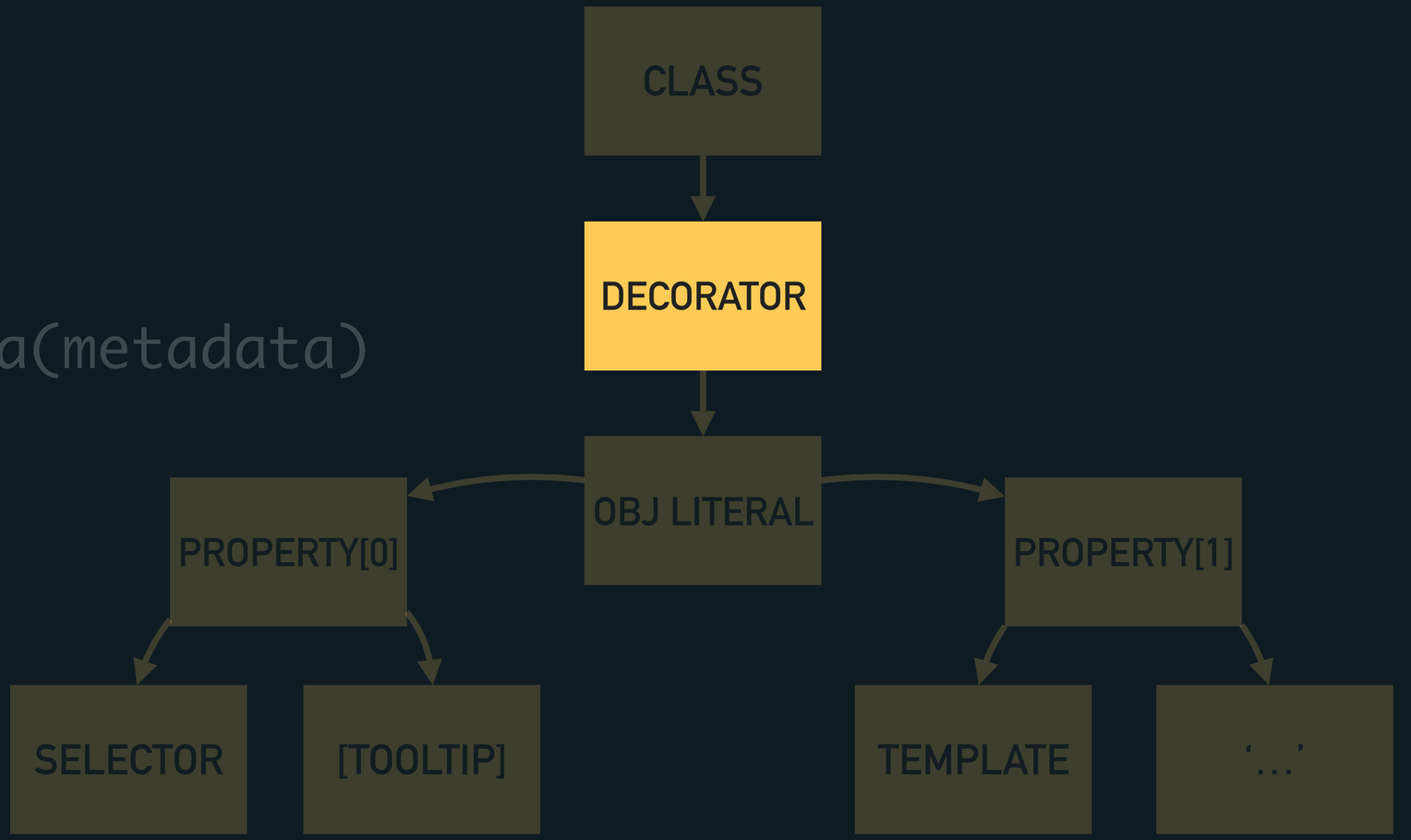
Validate Selector

```
visitClass(node)

visitDecorator(node)

visitComponentMetadata(metadata)
```



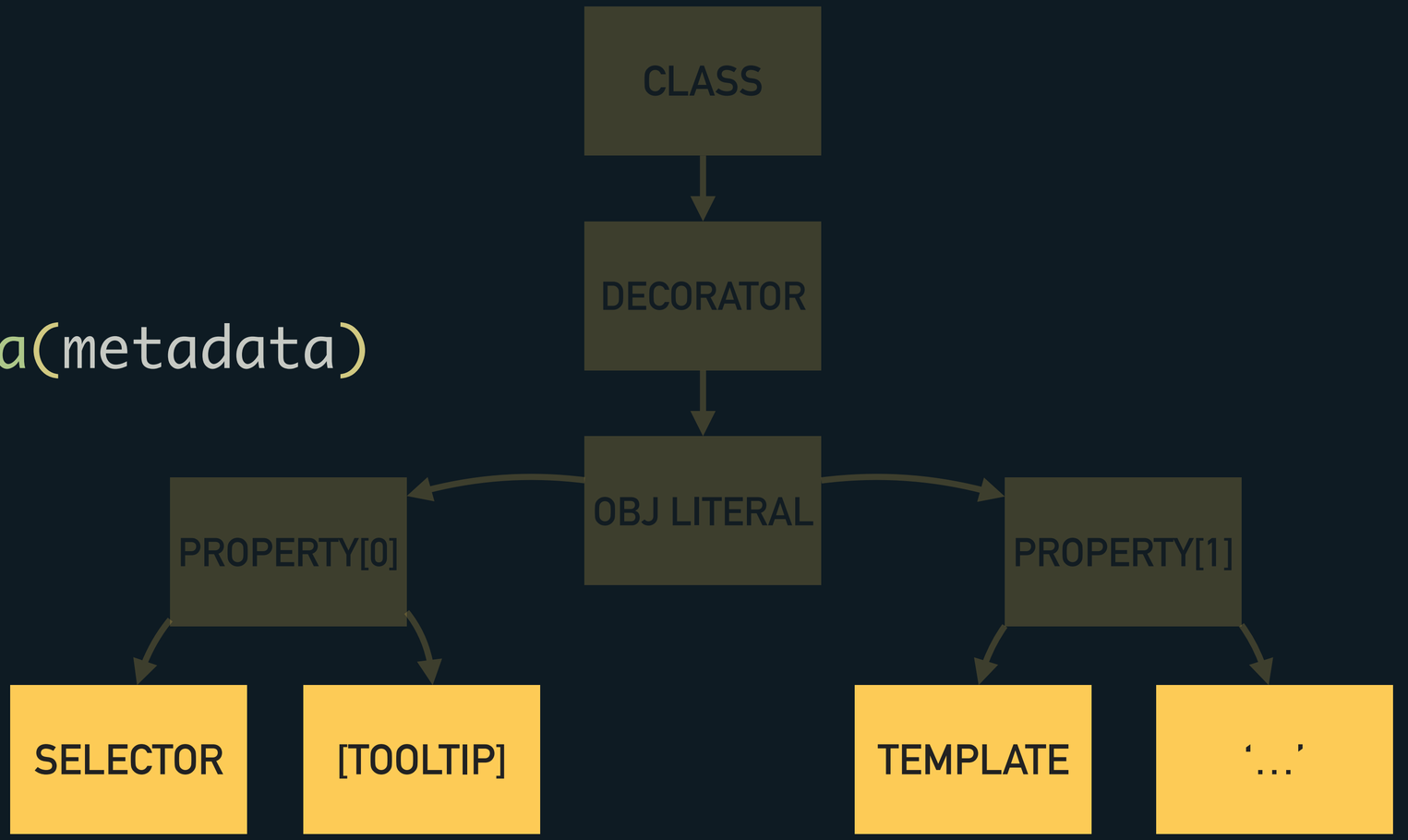
Validate Selector

```
visitClass(node)

visitDecorator(node)

visitComponentMetadata(metadata)
```



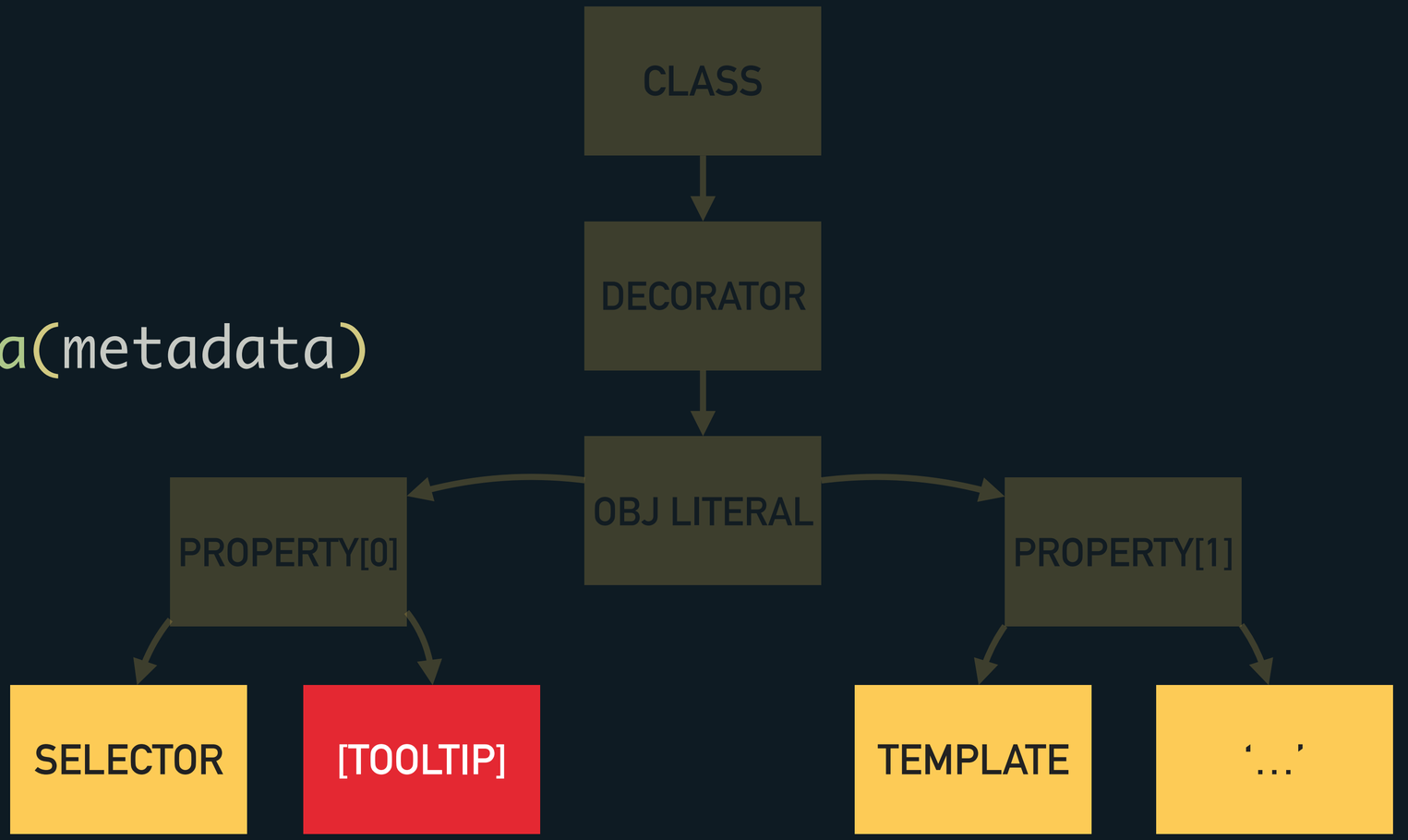
Validate Selector

```
visitClass(node)

visitDecorator(node)

visitComponentMetadata(metadata)
```



Validate Selector

```
visitClass(node)

visitDecorator(node)

visitComponentMetadata(metadata)
```



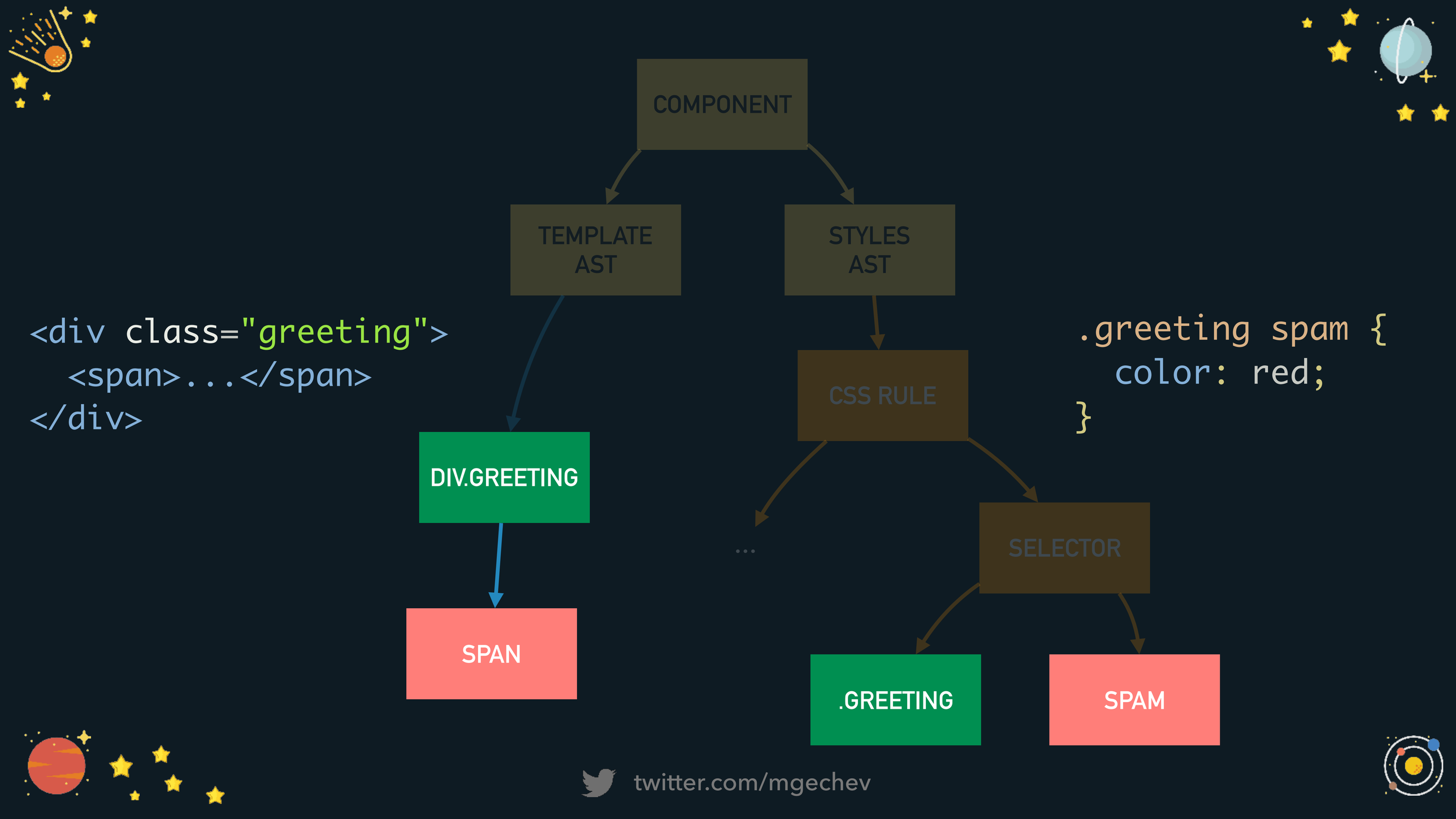

Analysis of Styles and Templates


twitter.com/mgechev

```
@Component({
  selector: 'header-cmp',
  template: `

    <h1 class="greeting">Hello <span>{{ name }}</span>!</div>
  `,
  styles: [`

    .greeting spam {
      color: red;
    }
  `]
})
class HeaderComponent { ... }
```




```
@Component({
  selector: 'header-cmp',
  template: `
    <h1 class="greeting">Hello <span>{{ name }}</span>!</div>
  `,
  styles: [`
    .greeting spam {
      color: red;
    }
  `]
})
class HeaderComponent { ... }
```


COMPONENT
TEMPLATE AST
STYLES AST

...

DIV.GREETING
SPAN
CSS RULE
...
SELECTOR
.GREETING
SPAN
.greeting spam {
color: red;
}

```
3  @Component({
4    selector: 'sd-app',
5    template: `
6      <h1 class="header">Weather Forecast</h1>
7      <div [class.hot]="temperature > 85">{{ temperature }}</div>
8    `,
9    styles: [`
10     h1.head {
11       font-size: 22px;
12     }
13     .hot {
14       color: red;
15     }
16     div.medium {
17       color: orange;
18     }
19     .cold {
20       color: blue;
21     }
22   `]
23 })
24 export class AppComponent {
25   temperature = 80;
26 }
```


Breaking Changes

Smooth Upgrade to v4

```
$ npm install
   @angular/
     {common,compiler,compiler-cli,
      core,forms,http,platform-browser,
      platform-browser-dynamic,
      platform-server,router,animations}
   @^4.0.0 typescript@^2.2.2 --save --save-exact
```




Smooth Upgrade to v4

```
$ npm install
   @angular/
     {common,compiler,compiler-cli,
      core,forms,http,platform-browser,
      platform-browser-dynamic,
      platform-server,router,animations}
   @^4.0.0 typescript@^2.2.2 --save --save-exact
```

...with a few deprecations


<template>

Renderer

RendererType

RendererFactory



twitter.com/mgechev



`<template>`	`<ng-template>`
`Renderer`	`Renderer2`
`RendererType`	`RendererType2`
`RendererFactory`	`RendererFactory2`

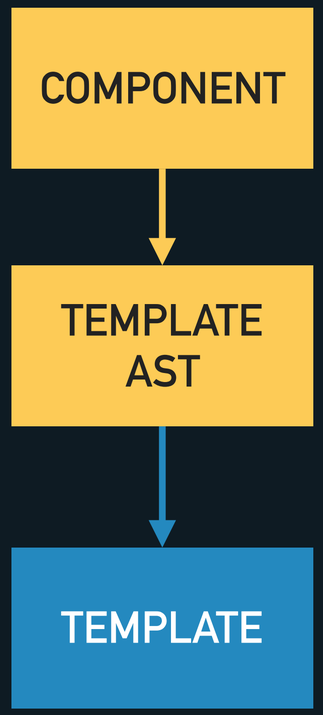

Is it that easy?

```
s/<template/<ng-template
s/<\/template/<\/ng-template
```


```
@Component({
  selector: 'content-cmp',
  template: '<template...></template>'
})
class ContentComponent {
  content = '<template> is an HTML5 element';
}
```


```
@Component({
  selector: 'content-cmp',
  template: '<template...></template>'
})
class ContentComponent {
  content = '<template> is an HTML5 element';
}
```

Context Aware Replacement



Demonstration

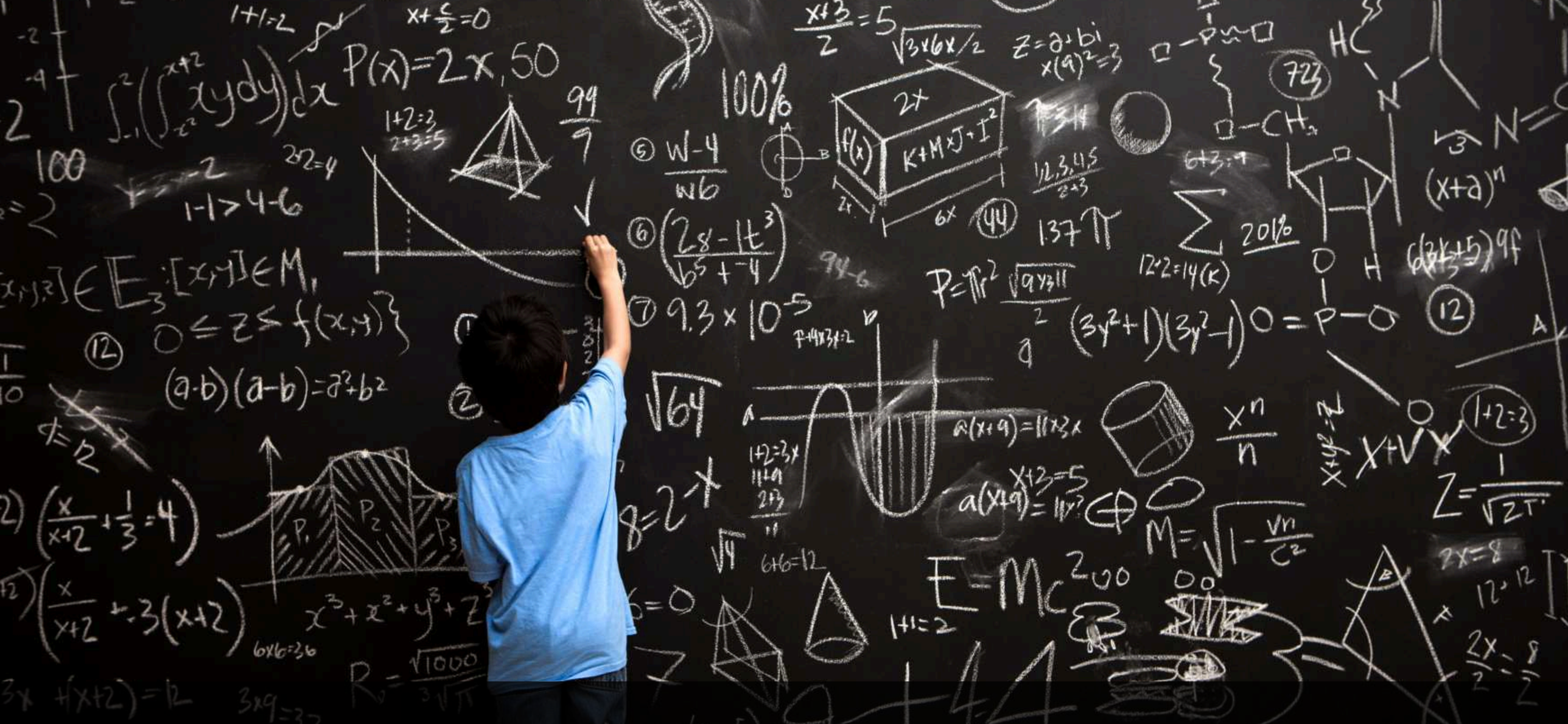
github.com/mgechev/ngmigrate

```
~/Projects/angular2-app master ❯ ngmigrate src/**/*.ts
```

```
Let's migrate your templates!

I found these deprecations:

src/about/about.component.html[4, 1]: You should use <ng-template> instead of <template>
src/home/home.component.html[7, 1]: You should use <ng-template> instead of <template>
src/home/home.component.html[15, 1]: You should use <ng-template> instead of <template>
src/todo/todo.component.ts[7, 5]: You should use <ng-template> instead of <template>

? Do you want me to fix them for you? Yes

Fixed 1 error(s) in src/about/about.component.html
Fixed 2 error(s) in src/home/home.component.html
Fixed 1 error(s) in src/todo/todo.component.ts
```


Code Visualization

twitter.com/mgechev



Large Software Systems are Complex


Understanding Large System

- A lot of code which is hard to digest
- Mixing different levels of abstractions
- Humans are good in “visual thinking”



compiler provides primitives
for parsing an Angular application



github.com/mgechev/

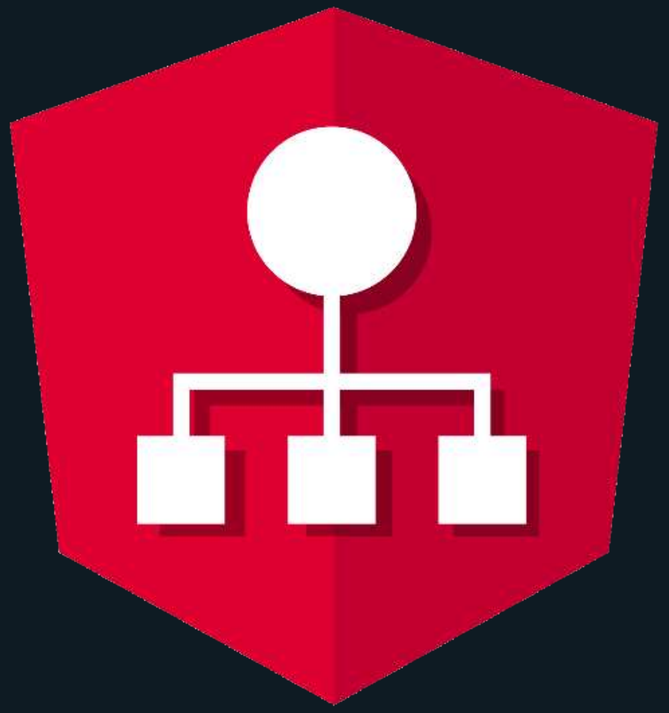
ngast


twitter.com/mgechev


ngast provides

A way to get all:

- Directives / components and their metadata
- Providers and their relations
- etc...

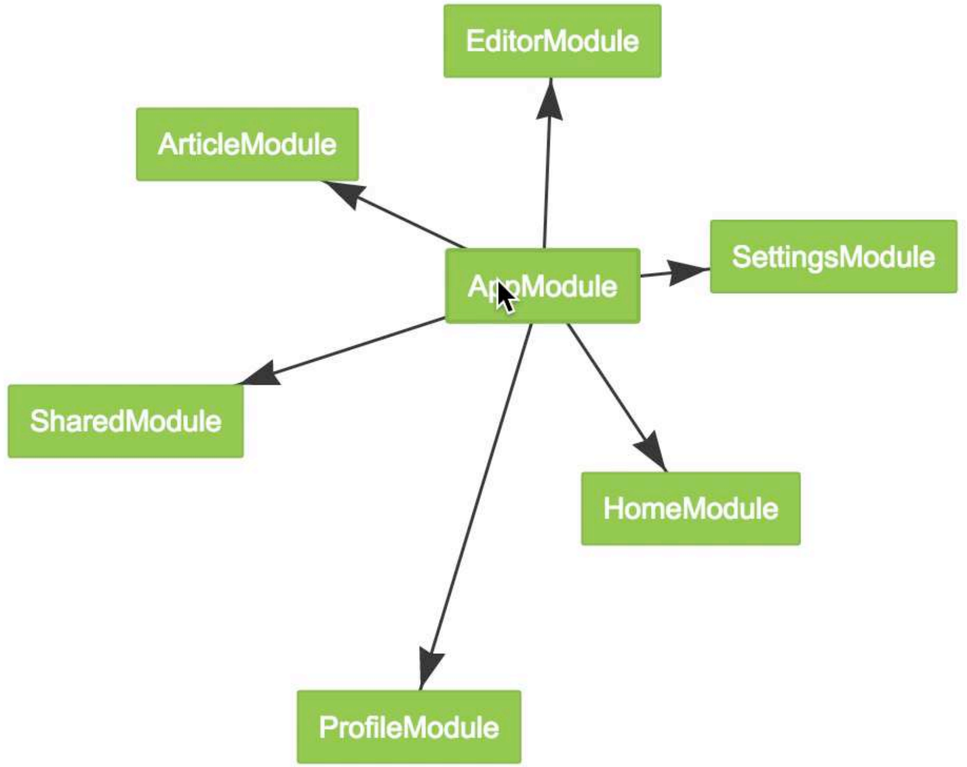


github.com/mgechev/

ngrev

twitter.com/mgechev

ngrev

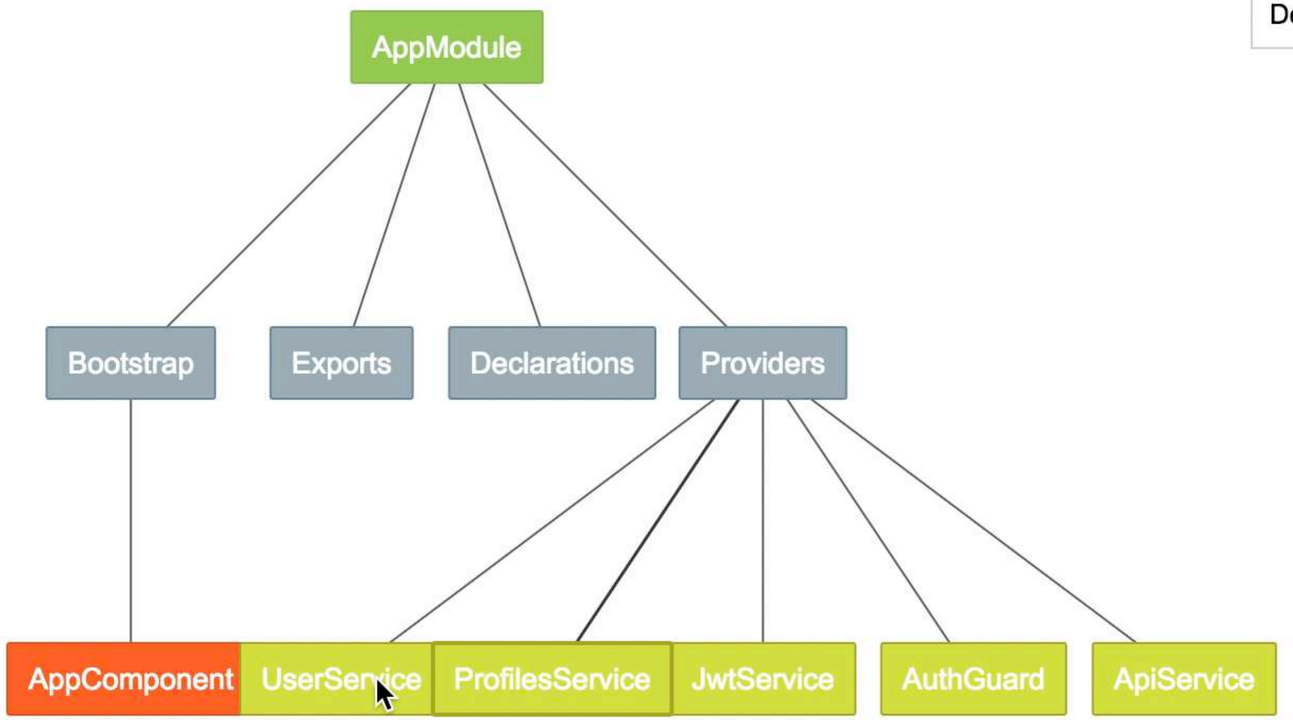


Legend
Module

ngrev

Back

Name	Value
Name	ProfilesService
Multiprovider	false
Dependencies	ApiService



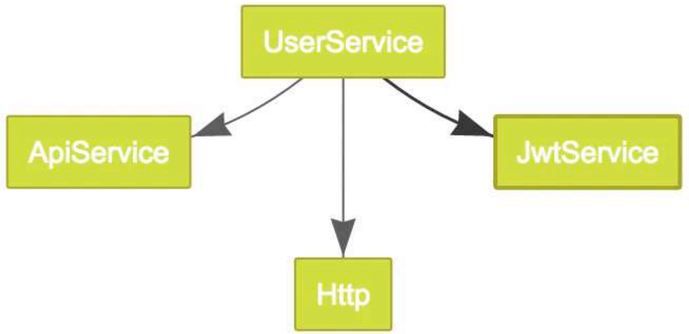
Legend

- Meta
- Module
- Component or Directive
- Provider

ngrev

Back

Name	Value
Name	JwtService
Multiprovider	false

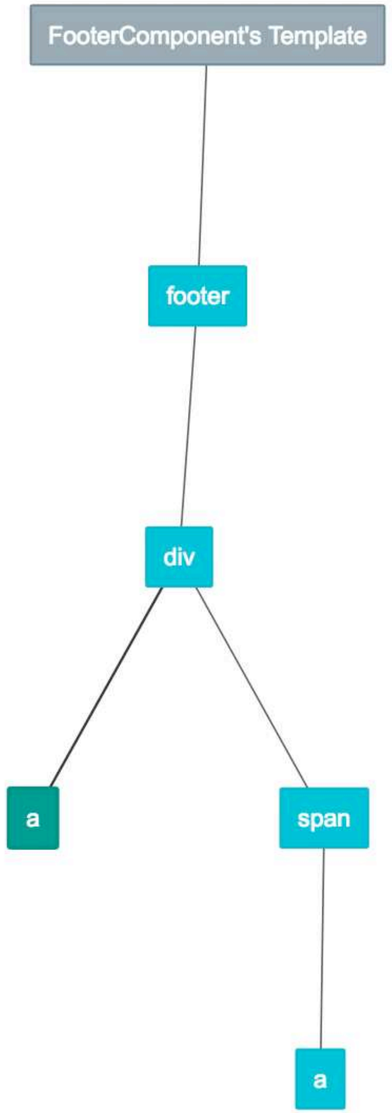


Legend

Provider

ngrev

Back



Name	Value
Name	a
Directives	RouterLinkWithHref
Attributes	[class=logo-font], [routerLink=/]

Legend

- Meta
- HTML element
- HTML element with Directive


lets go
one dimension above



We're putting our source code in our own reality, instead, lets move to its own reality!


Visualization

Demonstration

github.com/mgechev/ngworld


AboutModule
RatingComponent

Thank you!

twitter.com/mgechev
github.com/mgechev
blog.mgechev.com