Linting Angular

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

PROGRAMMING

Minko Gechev

mgechev

Functional time traveler.

California, USA

http://blog.mgechev.com/

Organizations

Overview | Repositories 171 | Stars 98 | Followers 2.8k | Following 3

angular-style-guide

Set of best practices for Angular application development

★ 4.8k 705

angular-seed

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

TypeScript ★ 3.8k 1.4k

angular/mobile-toolkit

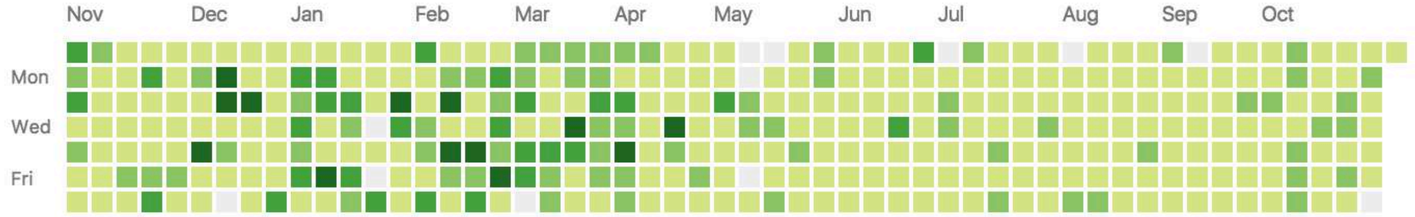
JavaScript implementation of different computer science algorithms.

TypeScript ★ 746 75

codelyzer

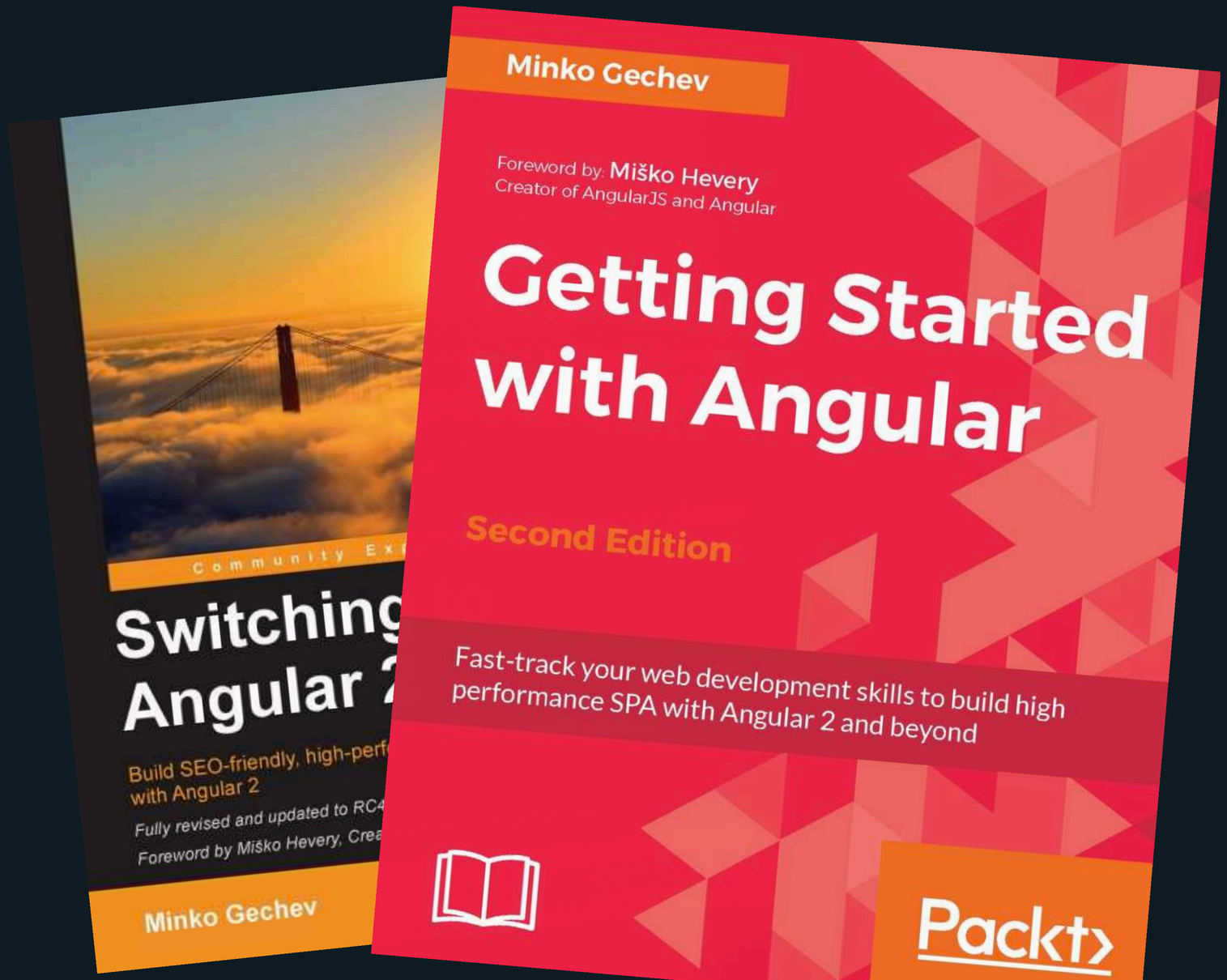
Linting for Angular projects.

TypeScript ★ 929 84



twitter.com/**mgechev**

github.com/**mgechev**


Minko Gechev
Foreword by Miško Hevery
Creator of AngularJS and Angular
Getting Started with Angular
Second Edition
Fast-track your web development skills to build high performance SPA with Angular 2 and beyond
Packt
Minko Gechev
Build SEO-friendly, high-perf
with Angular 2
Fully revised and updated to RC4
Foreword by Miško Hevery, Crea

ANGULAR CRUISE

PRESENTED BY NG-CONF

May 29, 2017! Check our blog.

Workshops by John Papa, Ben Lesh and Joe Eames!

TICKETS

SPONSOR

Jeff Cross

Victor Savkin

Ben Lesh

Alex Castillo

Aysegul Yonet

Minko Gechev

Aaron Frost

Mike Brocchi

Alyssa Nicoll

Sebastian Witalec

Justin Schwartzenberger

Uri Shaked

@UriShaked

Agenda

History

Codelyzer

How codelyzer works?

Compilers 101

Future plans

History

JavaScript Daily
@JavaScriptDaily

Follow

Community-Driven Best Practices and Style Guidelines for Angular 2 Application Development:

mgechev/angular2-style-guide

angular2-style-guide - Community-driven set of best practices and style guidelines for Angular 2 application development

github.com

RETWEETS 57 LIKES 107

7:10 PM - 12 Apr 2016

JavaScript Daily
@JavaScriptDaily

Follow

The Official Angular 2 Style Guide:
angular.io/styleguide

RETWEETS	LIKES
90	120

12:11 PM - 5 May 2016

Applying The Style Guide In Your Project

Enforcing Style

Enforcing Style

- Fork the official style guide
  - Modify the styles according to your needs

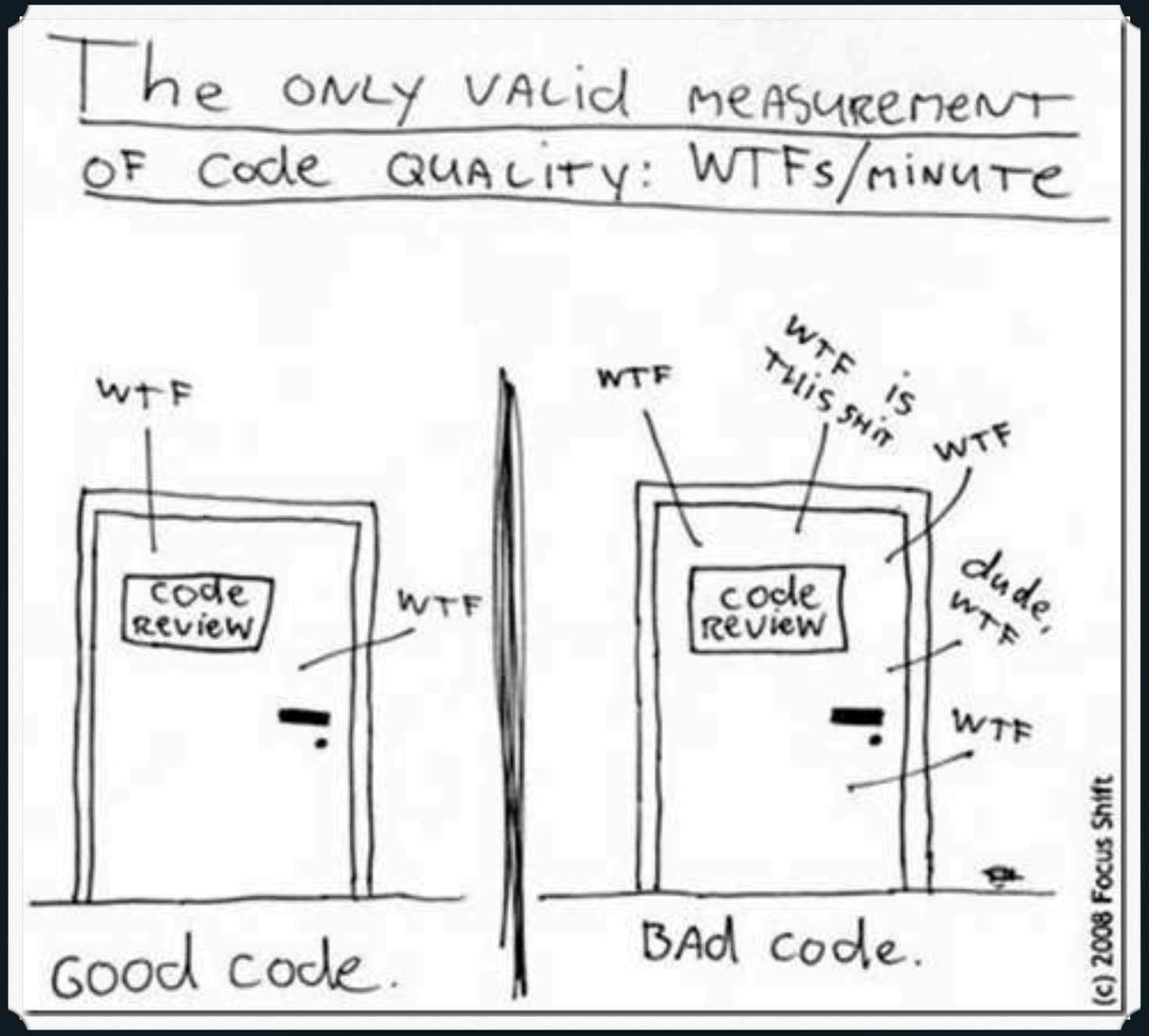
Enforcing Style

- Fork the official style guide
  - Modify the styles according to your needs
- Introduce the style guide to your team

Enforcing Style

- Fork the official style guide
  - Modify the styles according to your needs
- Introduce the style guide to your team
- Verify that each individual code change follows it

Core Review Process


The ONLY VALID MEASUREMENT OF Code QUALITY: WTFs/MINUTE
WTF
code review
WTF
Good code.
WTF
WTF
This is SHIT
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift

- Manual
- Boring
- Error Prone

codelyzer

Codelyzer is a project which aims to enforce common style using static code analysis

Codelyzer Rules

Codelyzer Rules

Components / directives name suffix

Codelyzer Rules

Components / directives name suffix

Components / directives selectors

Codelyzer Rules

Components / directives name suffix

Components / directives selectors

Implement life cycle hook interfaces

Codelyzer Rules

Components / directives name suffix

Components / directives selectors

Implement life cycle hook interfaces

Bind to public members

Codelyzer Rules

Components / directives name suffix

Components / directives selectors

Implement life cycle hook interfaces

Bind to public members

No dead CSS styles

Codelyzer Rules

Components / directives name suffix

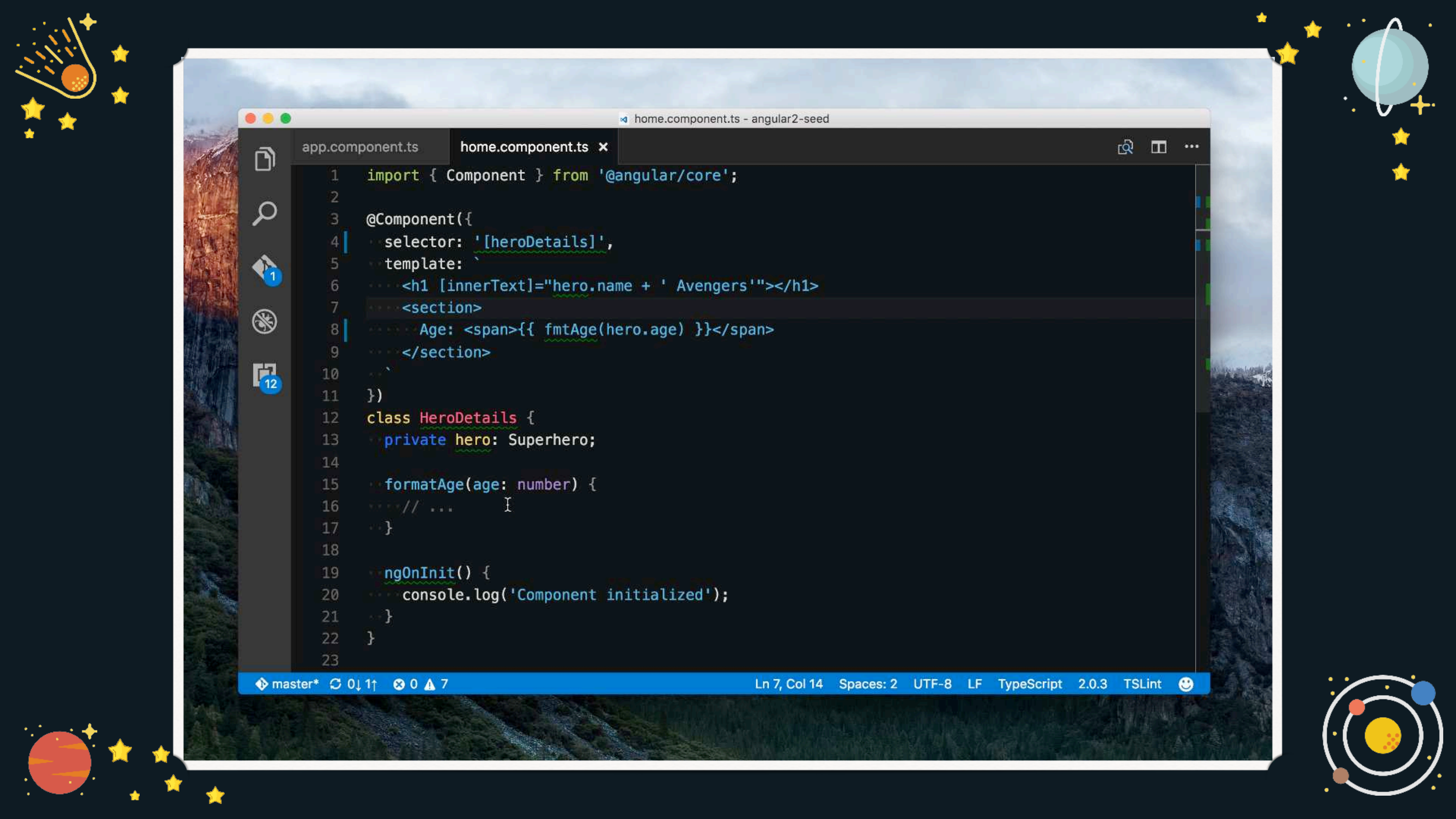
Components / directives selectors

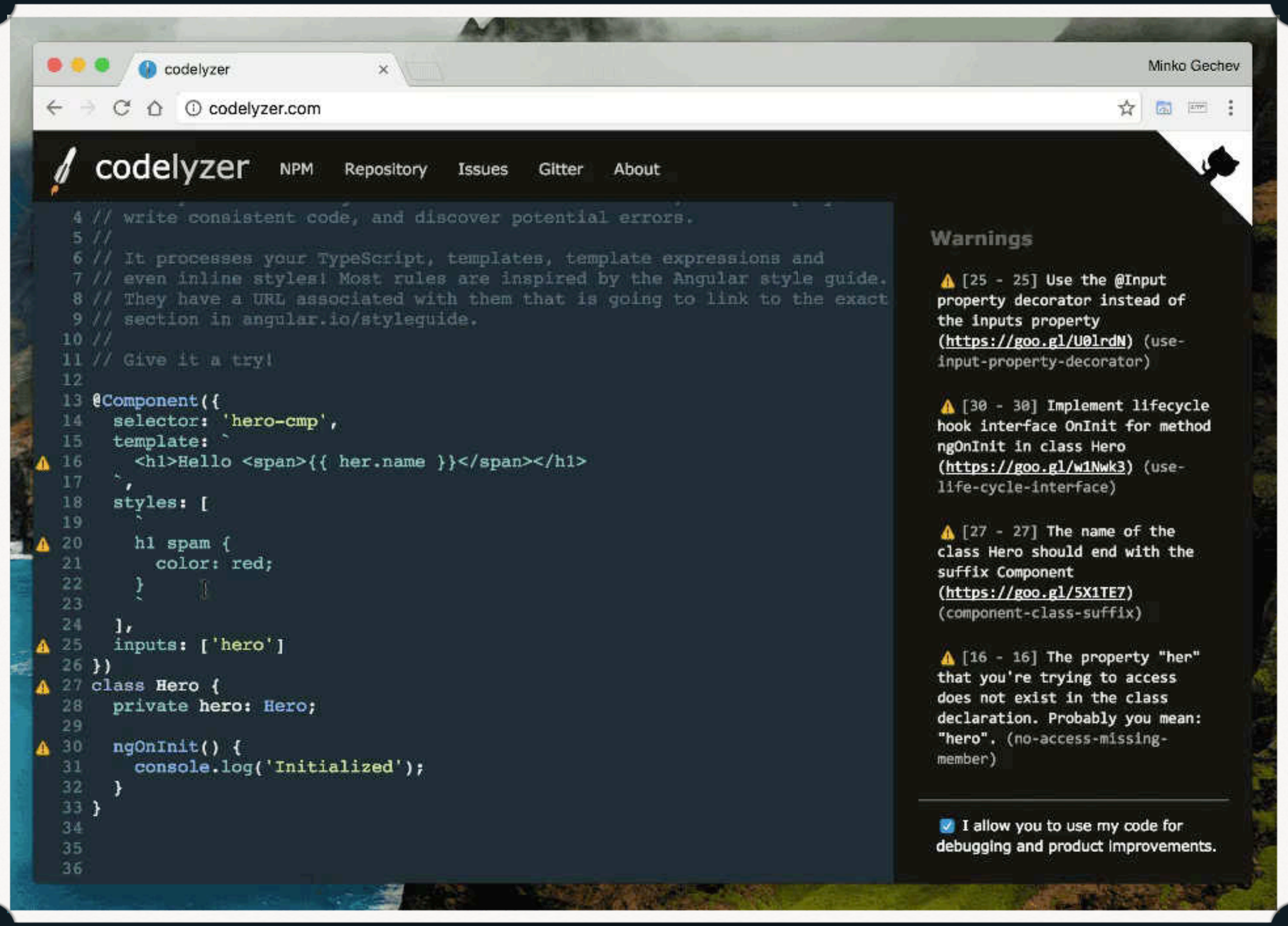
Implement life cycle hook interfaces

Bind to public members

No dead CSS styles

...

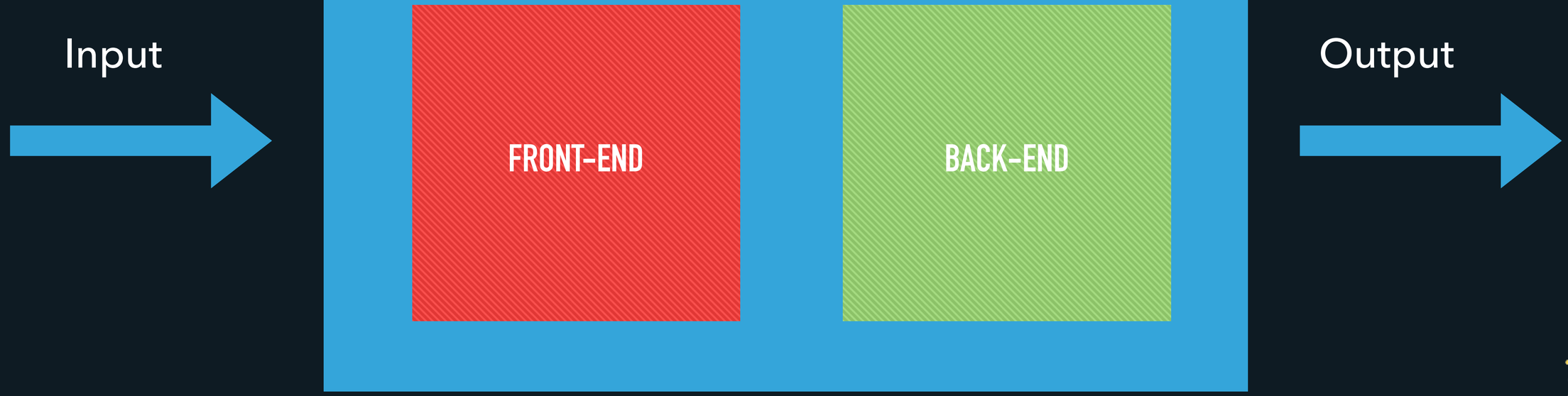

home.component.ts - angular2-seed
app.component.ts
home.component.ts
import { Component } from '@angular/core';
@Component({
selector: '[heroDetails]',
template: `
<h1 [innerText]="hero.name + ' Avengers'"></h1>
<section>
Age: {{ fmtAge(hero.age) }}
</section>
`
})
class HeroDetails {
private hero: Superhero;
formatAge(age: number) {
// ...
}
ngOnInit() {
console.log('Component initialized');
}
}
master* 0↓ 1↑ 0 7
Ln 7, Col 14 Spaces: 2 UTF-8 LF TypeScript 2.0.3 TSLint


codelyzer
Minko Gechev
codelyzer.com
codelyzer
NPM
Repository
Issues
Gitter
About
// write consistent code, and discover potential errors.
//
// It processes your TypeScript, templates, template expressions and
// even inline styles! Most rules are inspired by the Angular style guide.
// They have a URL associated with them that is going to link to the exact
// section in angular.io/styleguide.
//
// Give it a try!
@Component({
selector: 'hero-cmp',
template: `
<h1>Hello {{ her.name }}</h1>
`,
styles: [
`
h1 spam {
color: red;
}
`
],
inputs: ['hero']
})
class Hero {
private hero: Hero;
ngOnInit() {
console.log('Initialized');
}
}
Warnings
[25 - 25] Use the @Input property decorator instead of the inputs property (https://goo.gl/U0lrdN) (use-input-property-decorator)
[30 - 30] Implement lifecycle hook interface OnInit for method ngOnInit in class Hero (https://goo.gl/w1Nwk3) (use-life-cycle-interface)
[27 - 27] The name of the class Hero should end with the suffix Component (https://goo.gl/5X1TE7) (component-class-suffix)
[16 - 16] The property "her" that you're trying to access does not exist in the class declaration. Probably you mean: "hero". (no-access-missing-member)
I allow you to use my code for debugging and product improvements.

How Codelyzer Works?

Compiler

Compiler



Compiler

Compiler

Input
FRONT-END
...
LEXICAL ANALYSIS
SYNTAX ANALYSIS
SEMANTIC ANALYSIS
Output

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

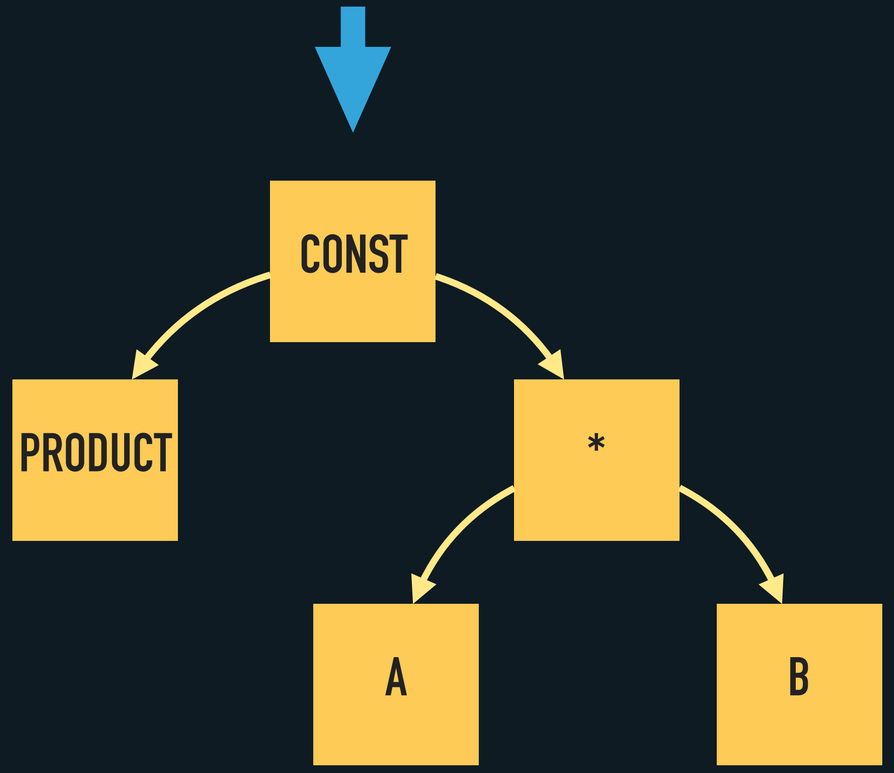
```
parse(tokenize('...'))
```

Syntax Analysis

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

Syntax Analysis

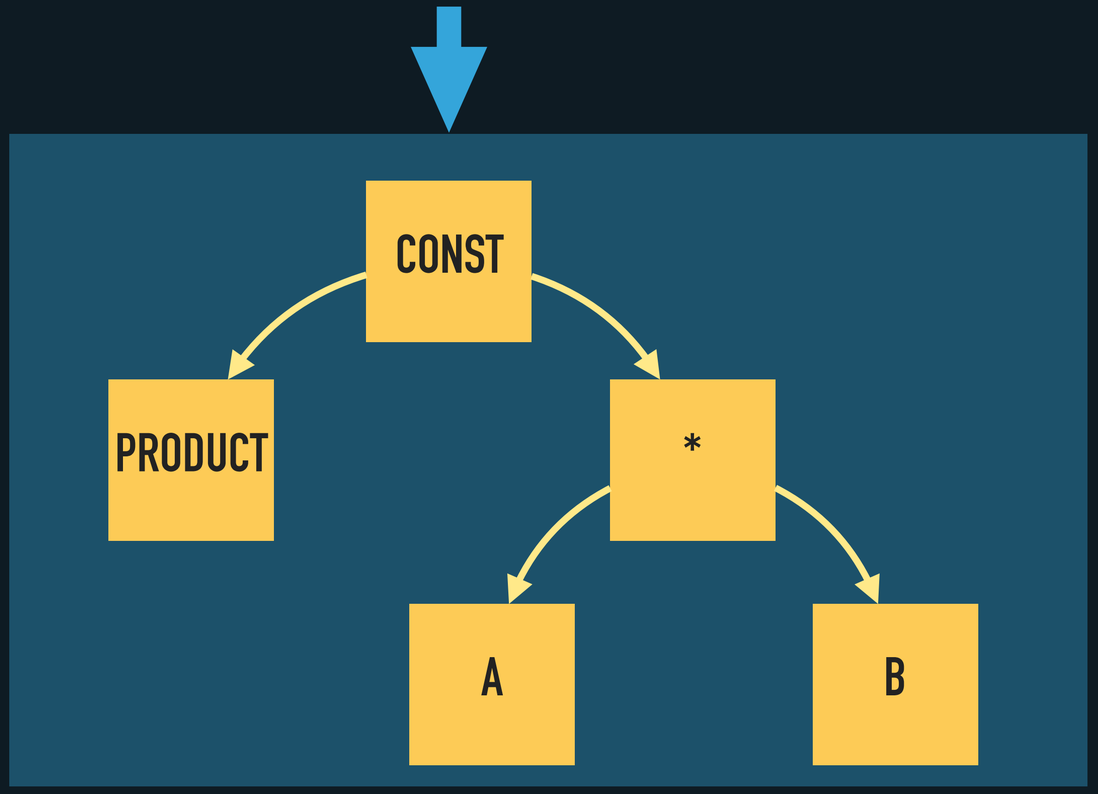
```
parse(tokenize('...'))
```



Syntax Analysis

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

AST

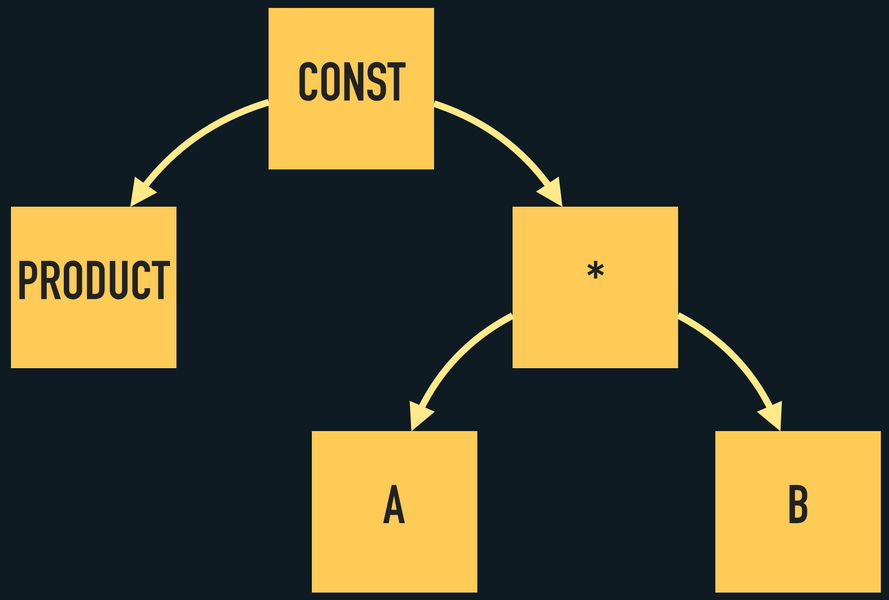


Semantic Analysis

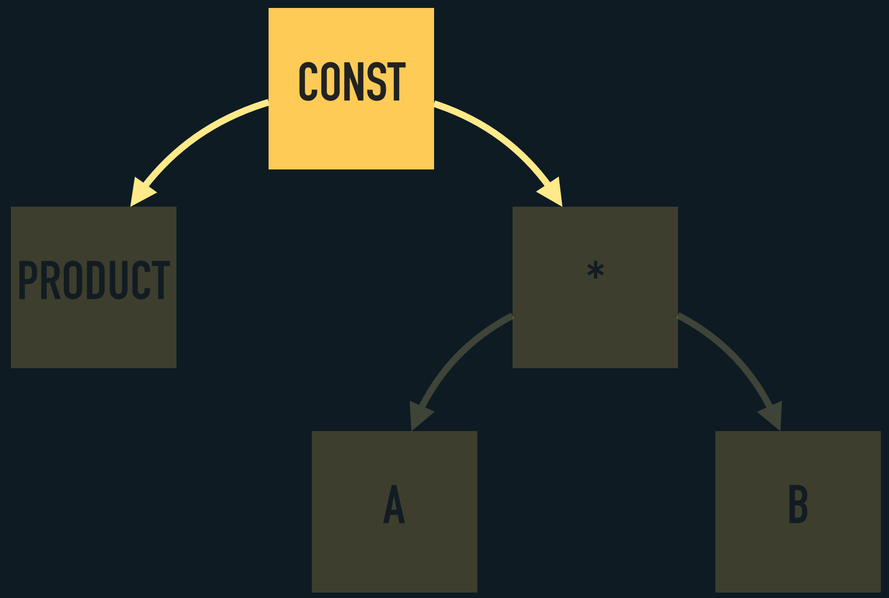
```
typeCheck(parse(tokenize('...')))
```

```
checkConst(node) {
  return checkExpression(node.initializer)
}

checkExpression(node) {
  if (node instanceof BinOp) {
    return checkBinOperation(node);
  }
  ...
}

checkBinOperation(node) {
  if (isMultiplication(node)) {
    if (!isNumber(node.left) ||
        !isNumber(node.right)) {
      throw new Error("Node must be 'number'");
    }
    ...
  }
}
```

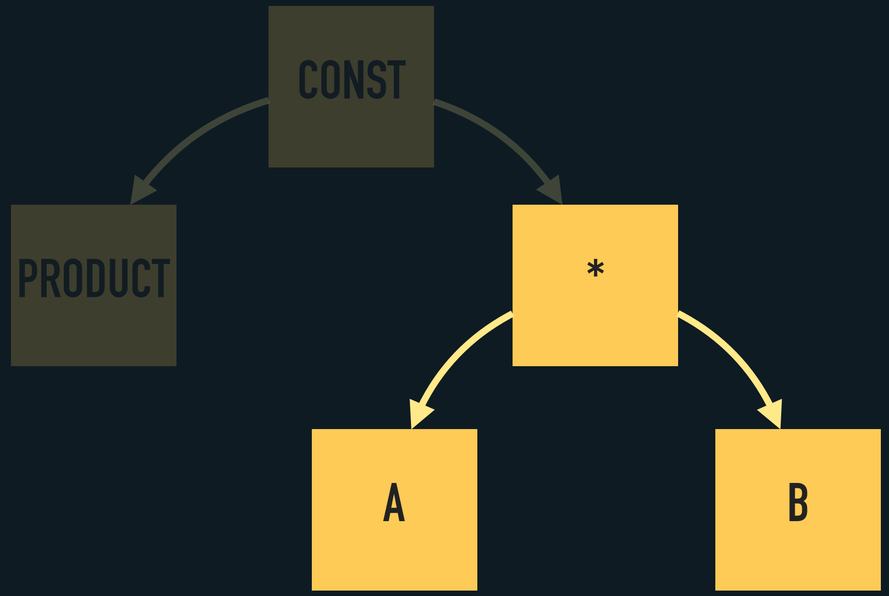
```
checkConst(node)

checkExpression(node)

checkBinOperation(node)

isNumber(node)
```



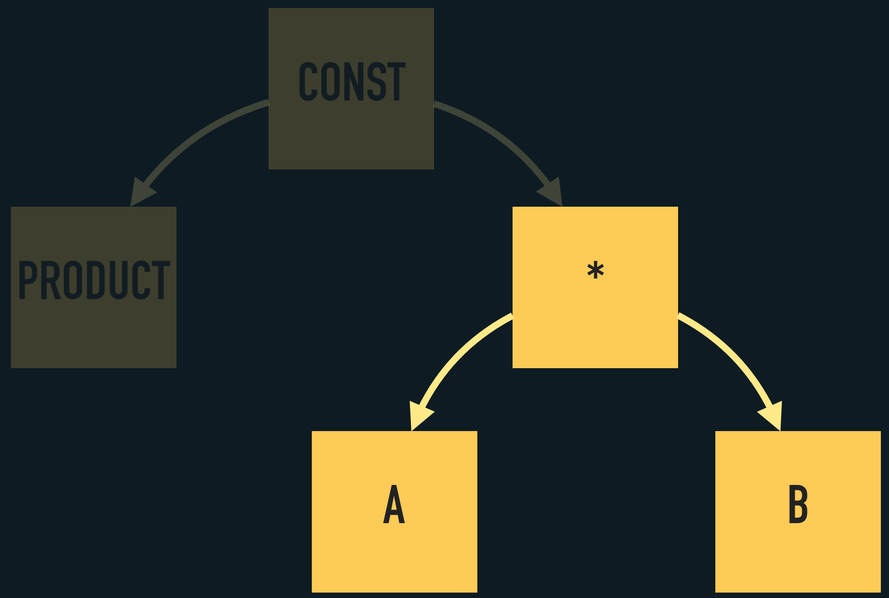
```
checkConst(node)

checkExpression(node)

checkBinOperation(node)

isNumber(node)
```



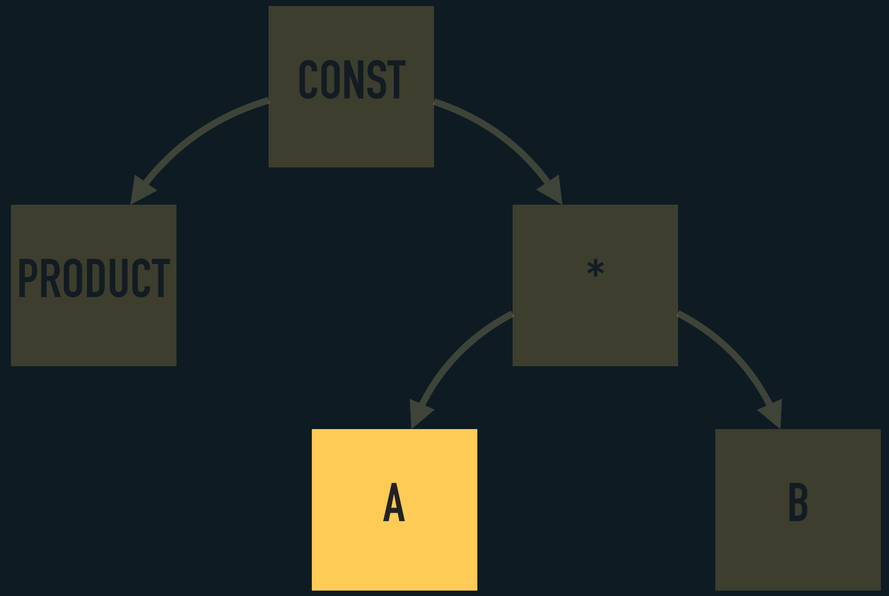
```
checkConst(node)

checkExpression(node)

checkBinOperation(node)

isNumber(node)
```



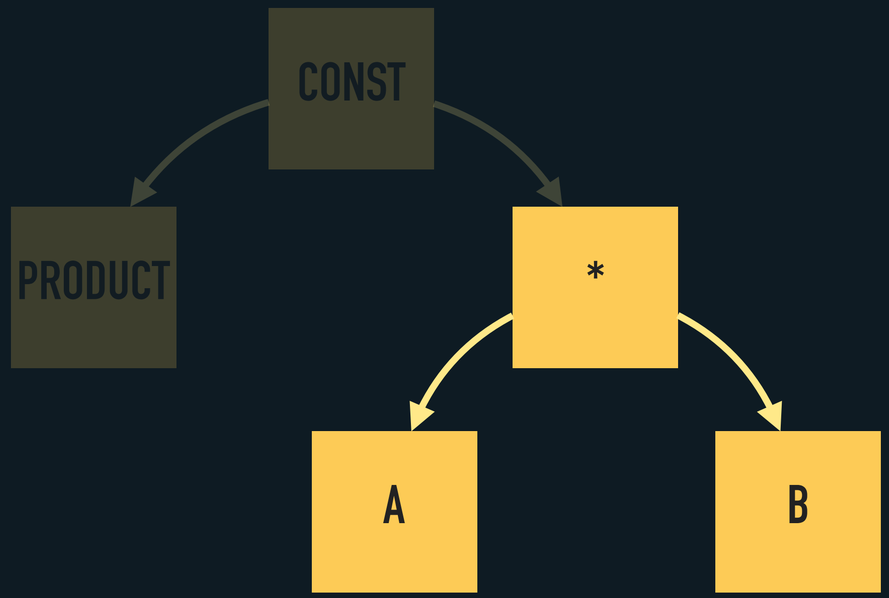
```
checkConst(node)

checkExpression(node)

checkBinOperation(node)

isNumber(node)
```



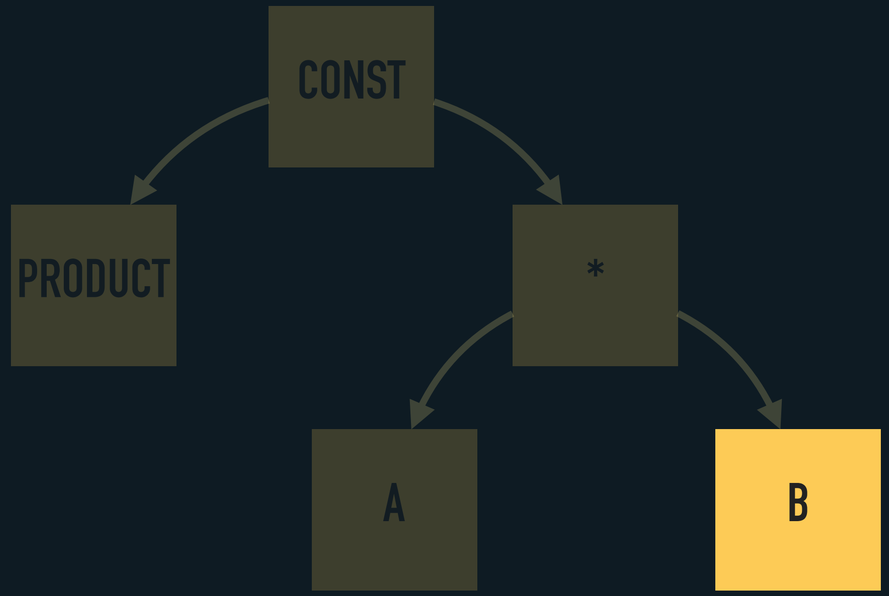
```
checkConst(node)

checkExpression(node)

checkBinOperation(node)

isNumber(node)
```



```
checkConst(node)

checkExpression(node)

checkBinOperation(node)

isNumber(node)
```



```
checkConst(node)

checkExpression(node)

checkBinOperation(node)

isNumber(node)
```



@ 爆笑gif图

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

Wikipedia

codelyzer

Codelyzer Goals

- Analyze the source code statically
- Show warning when a style is violated
- Extensible
- Reuse as much as possible

tsc

tslint

Alex Eagle

Scott Wu

Martin Probst

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.

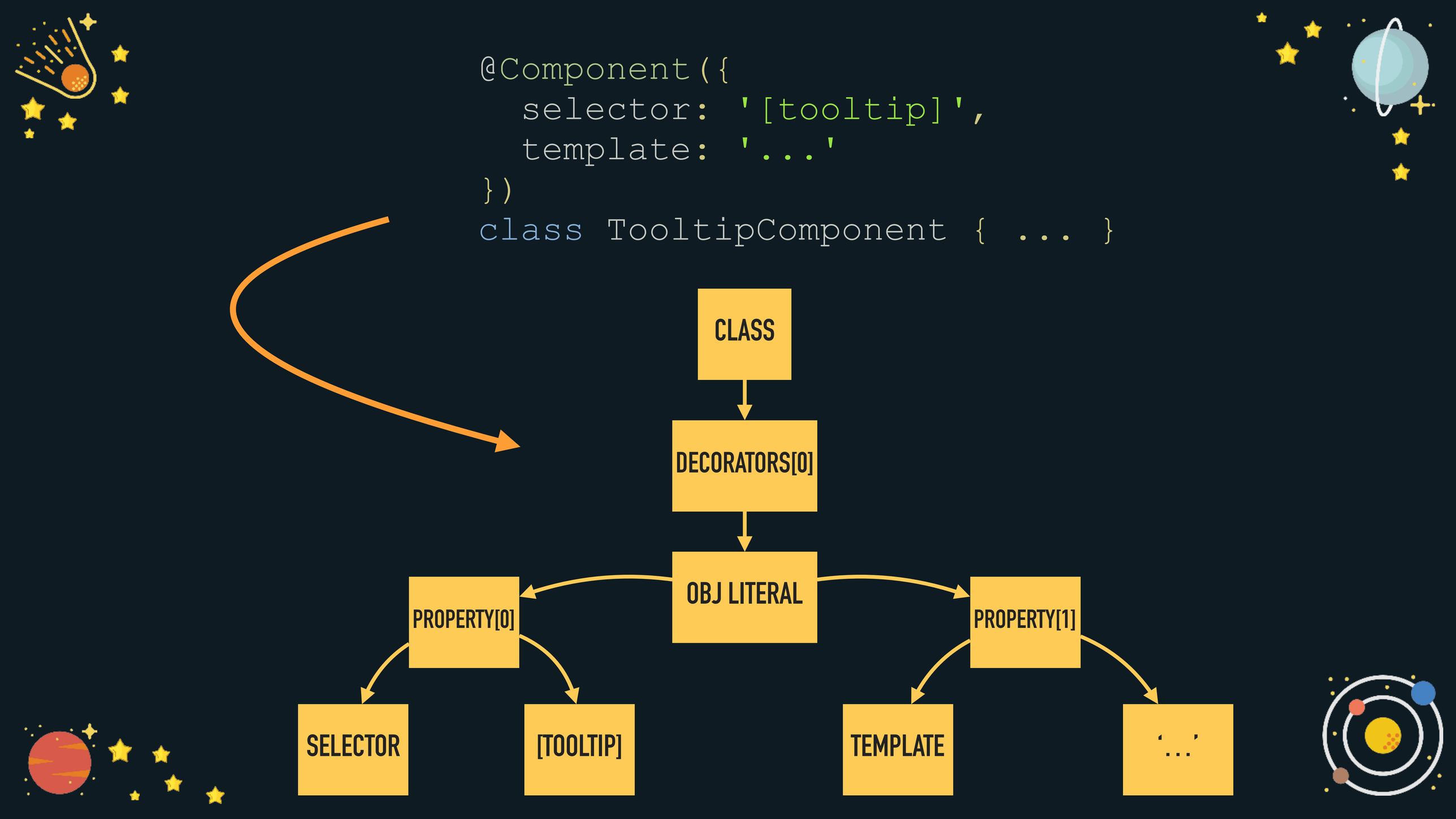
> **Why?** It is easier to recognize that a symbol is a component by looking at the template's html.

```
@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 { ... }
```

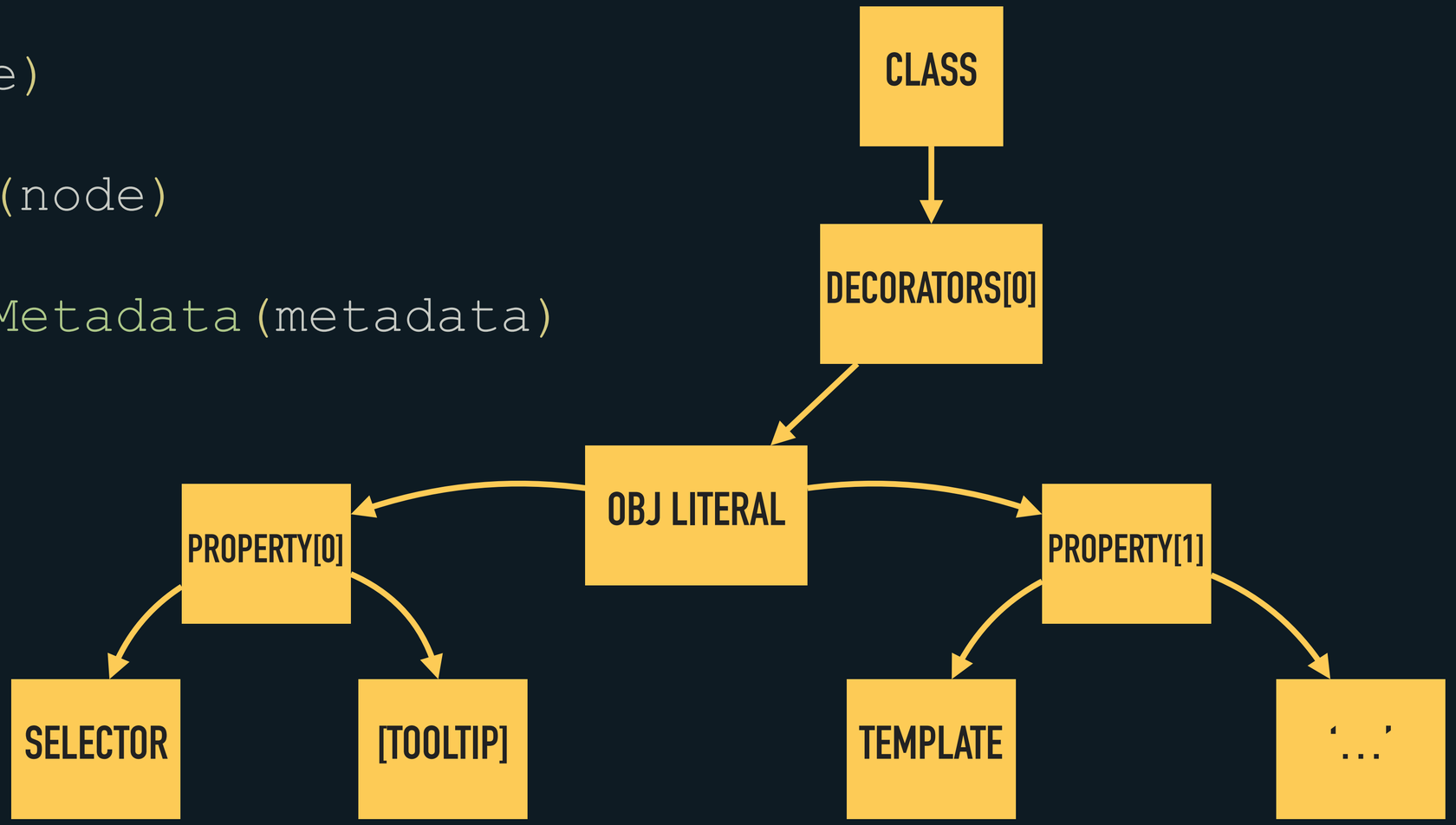

@Component({
selector: '[tooltip]',
template: '...'
})
class TooltipComponent { ... }
CLASS
DECORATORS[0]
OBJ LITERAL
PROPERTY[0]
PROPERTY[1]
SELECTOR
[TOOLTIP]
TEMPLATE
'...'

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');
  }
}
```

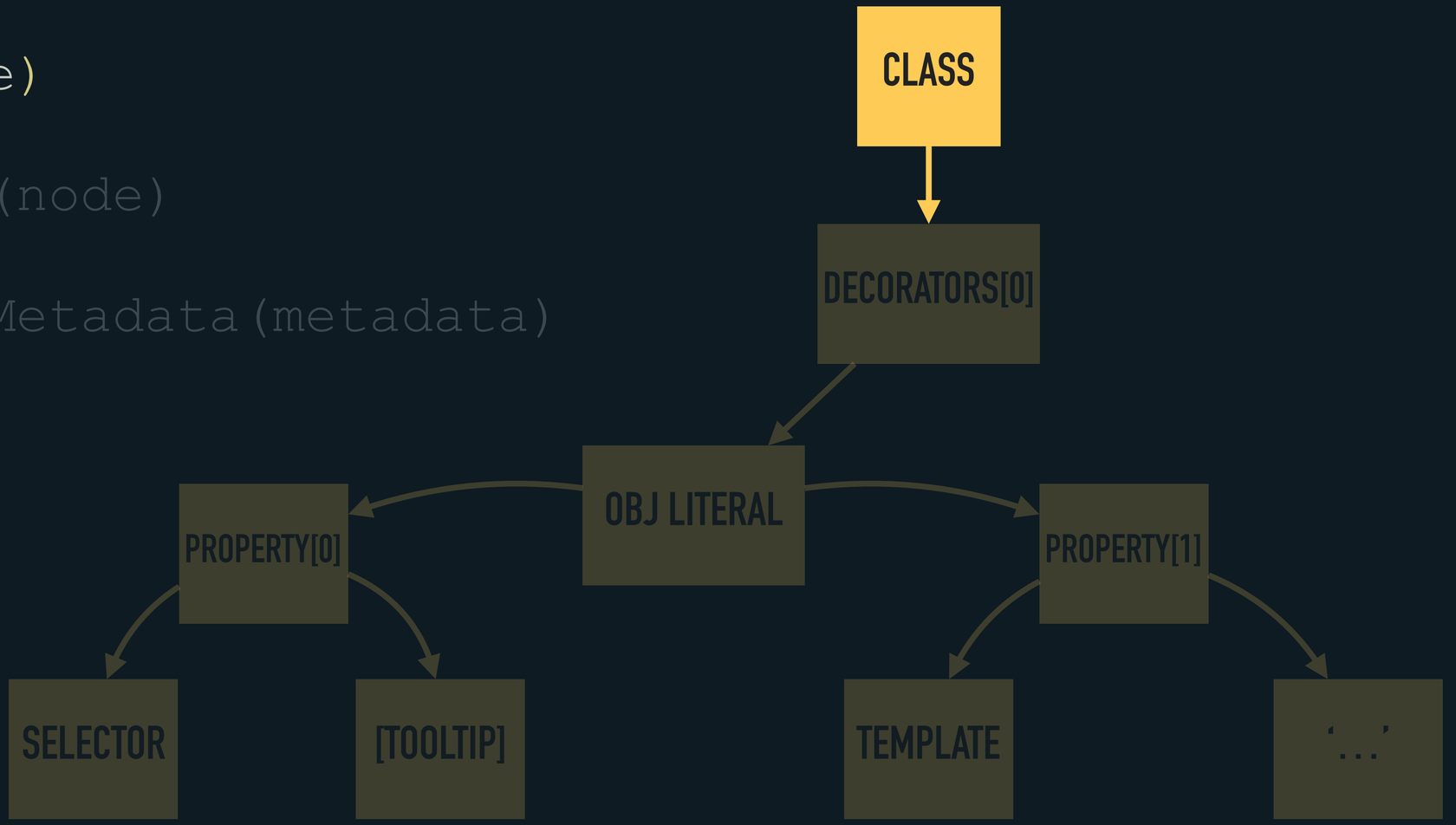
Validate Selector

```
visitClass(node)

visitDecorator(node)

visitComponentMetadata(metadata)
```



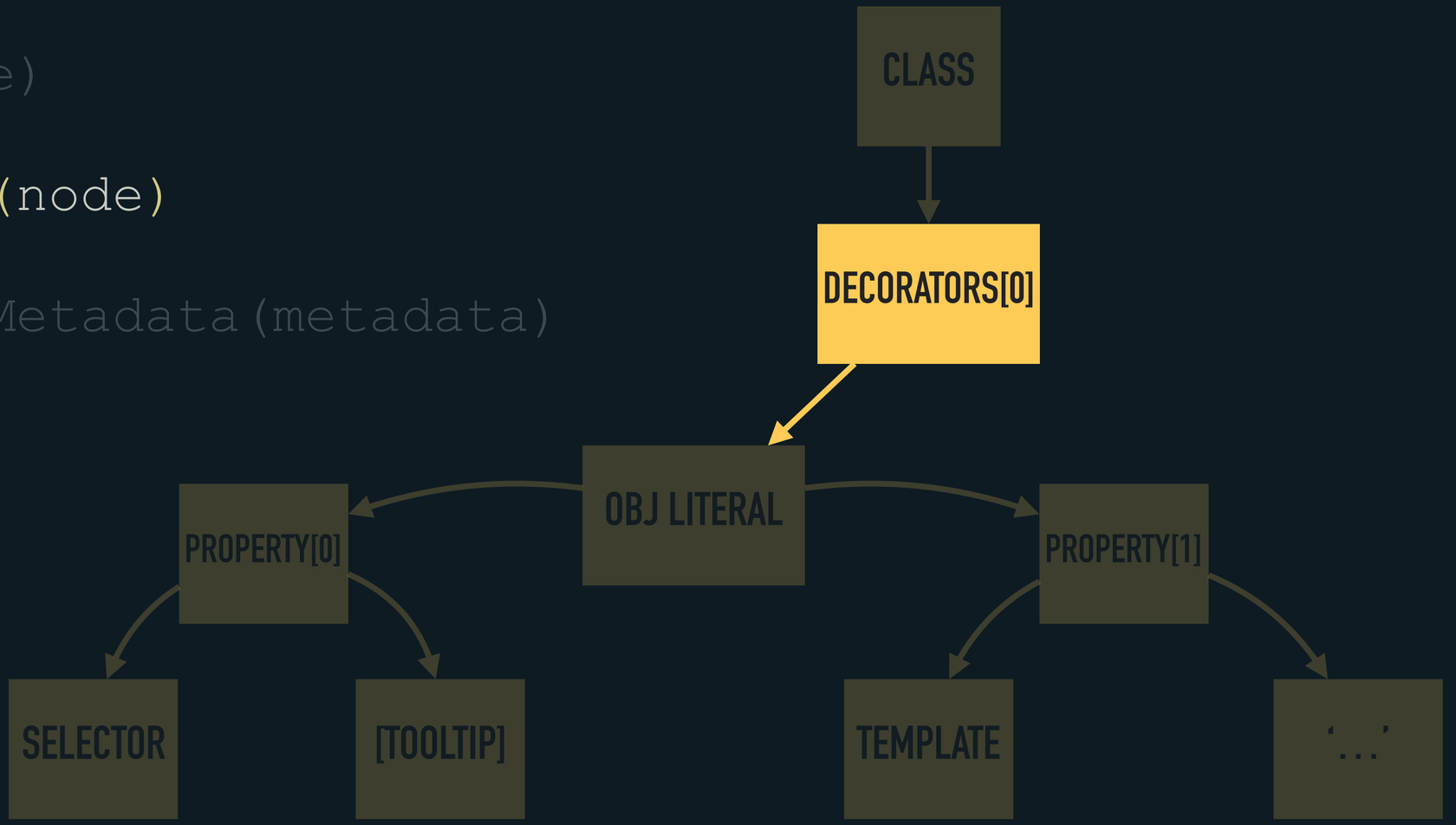
Validate Selector

```
visitClass(node)

visitDecorator(node)

visitComponentMetadata(metadata)
```



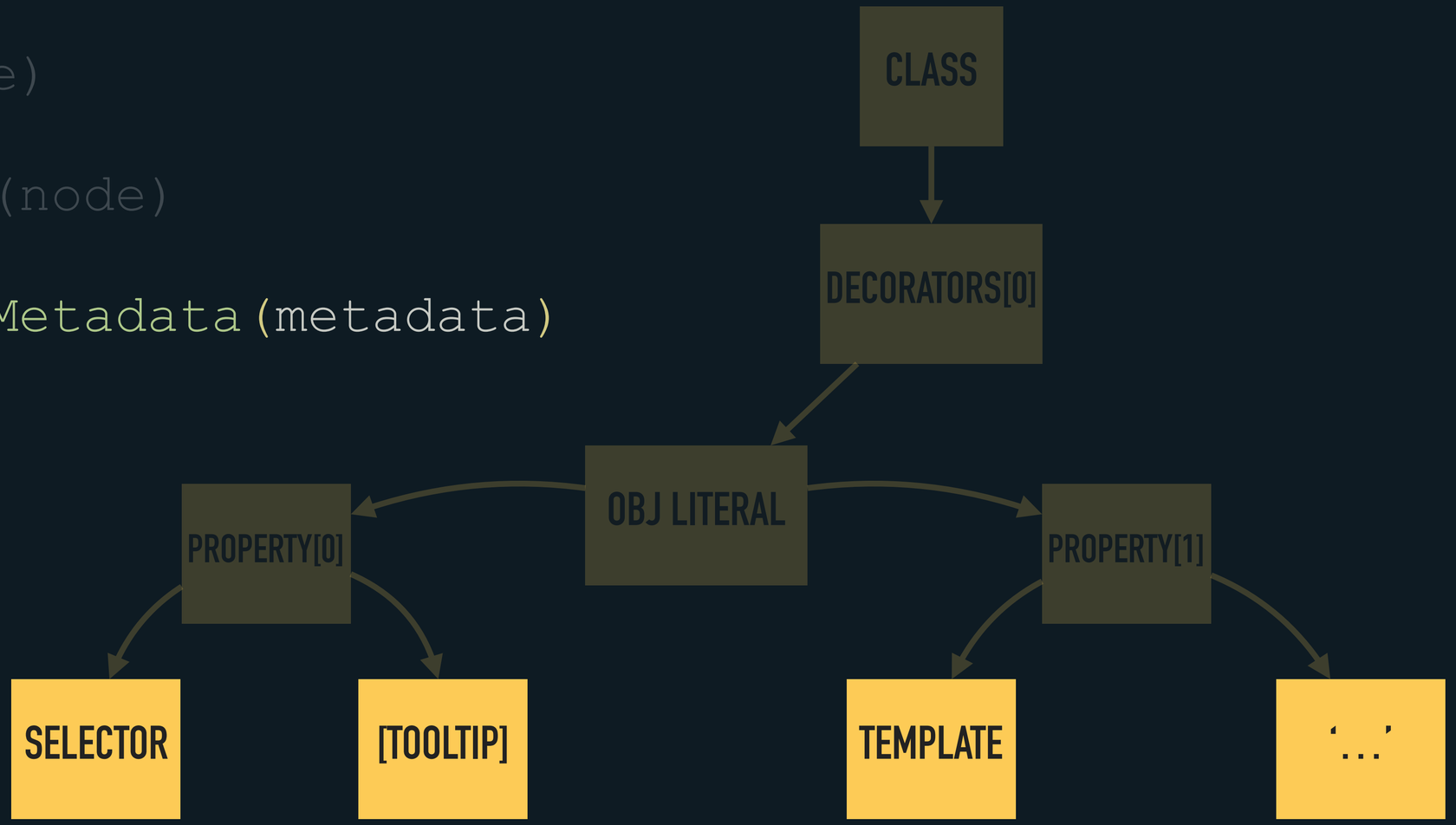
Validate Selector

```
visitClass(node)

visitDecorator(node)

visitComponentMetadata(metadata)
```



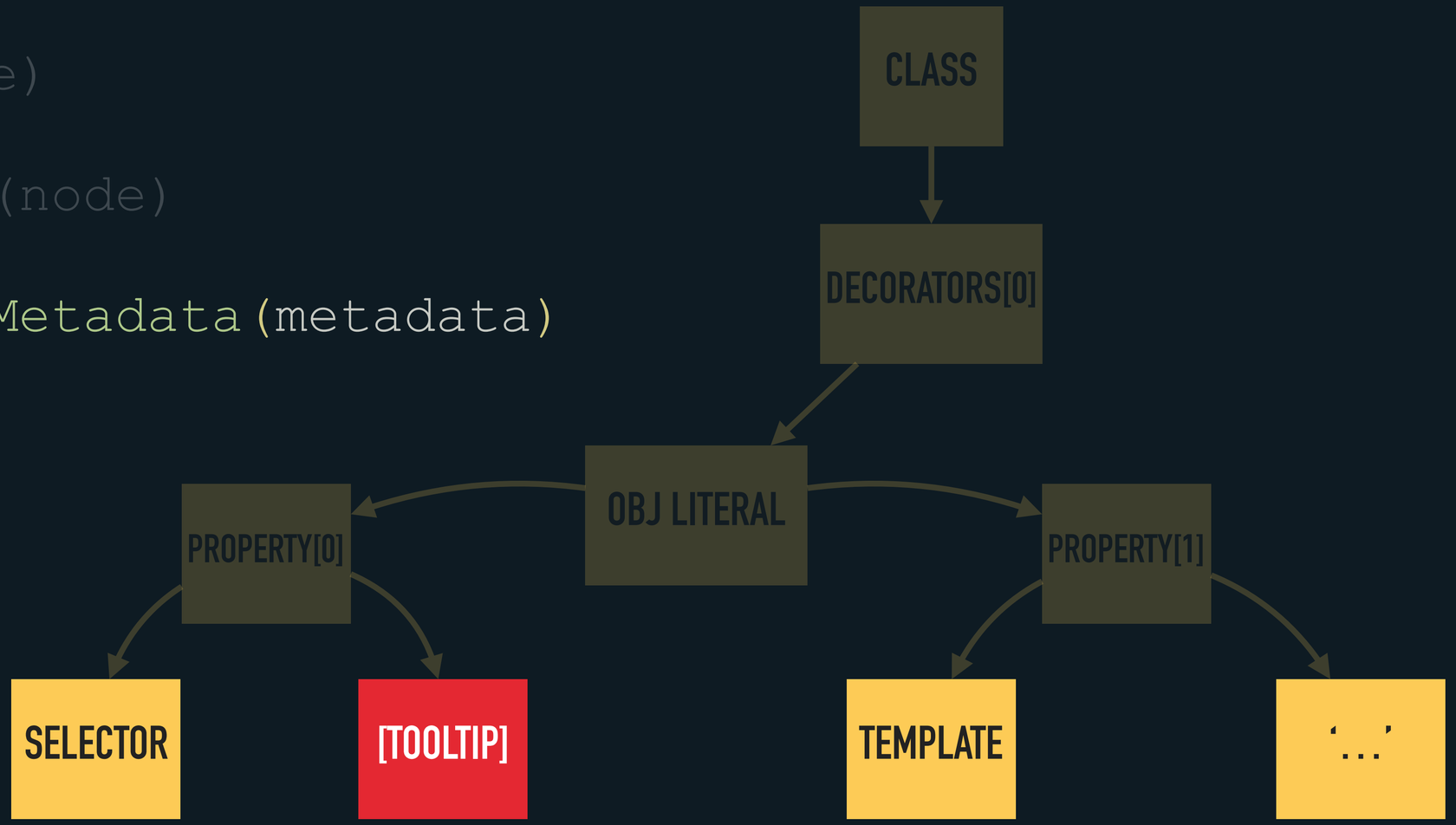
Validate Selector

```
visitClass(node)

visitDecorator(node)

visitComponentMetadata(metadata)
```



Validate Selector

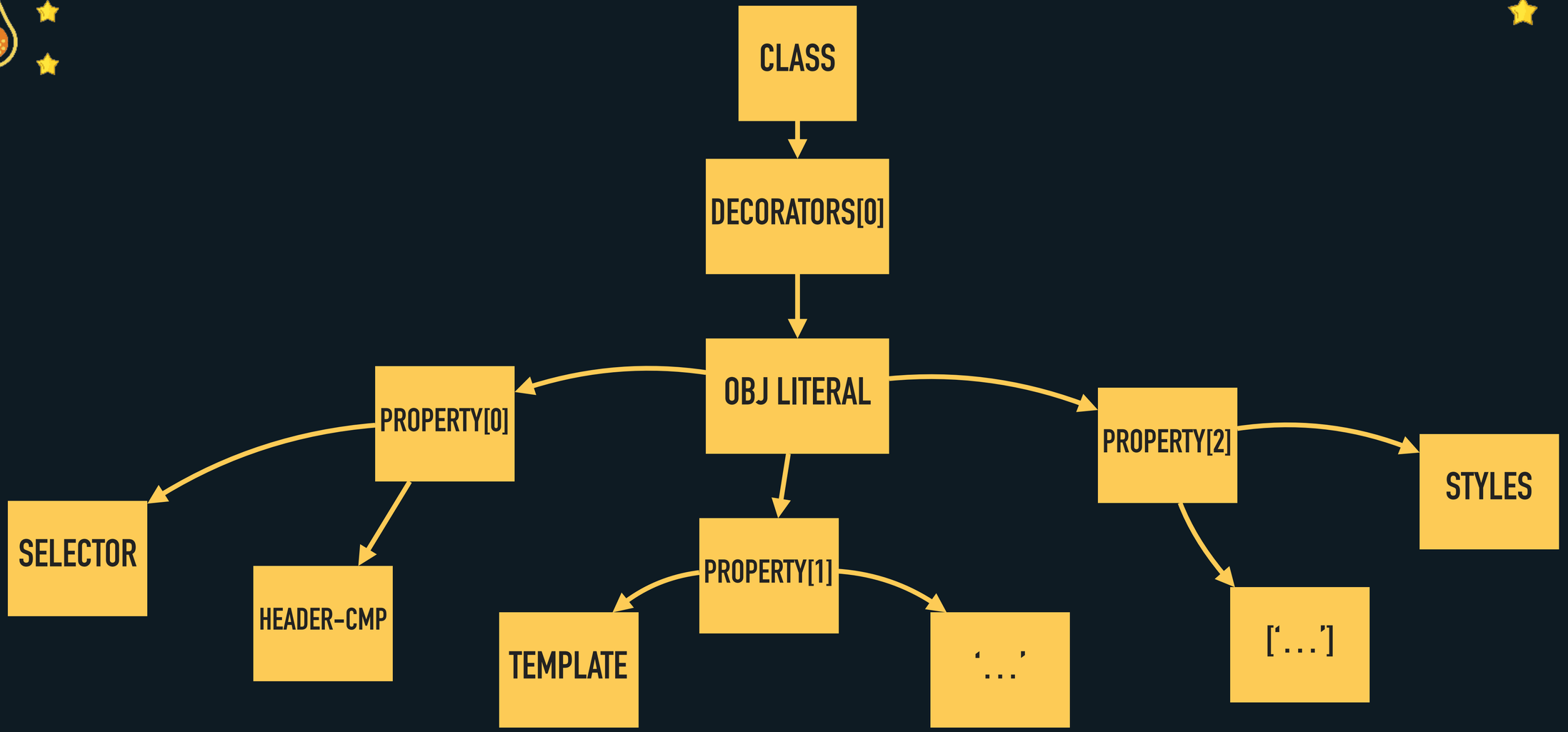
```
visitClass(node)

visitDecorator(node)

visitComponentMetadata(metadata)
```

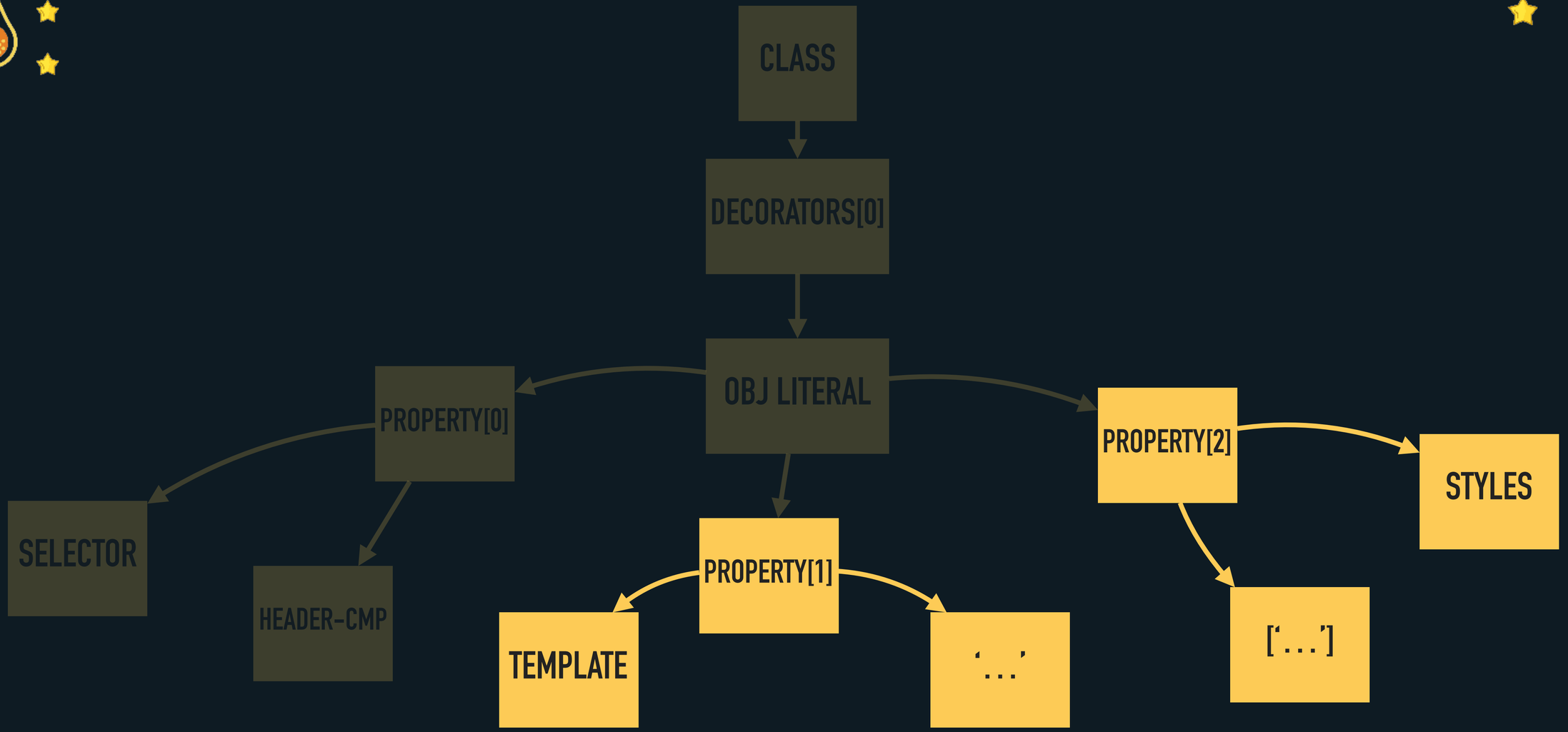


No Dead CSS

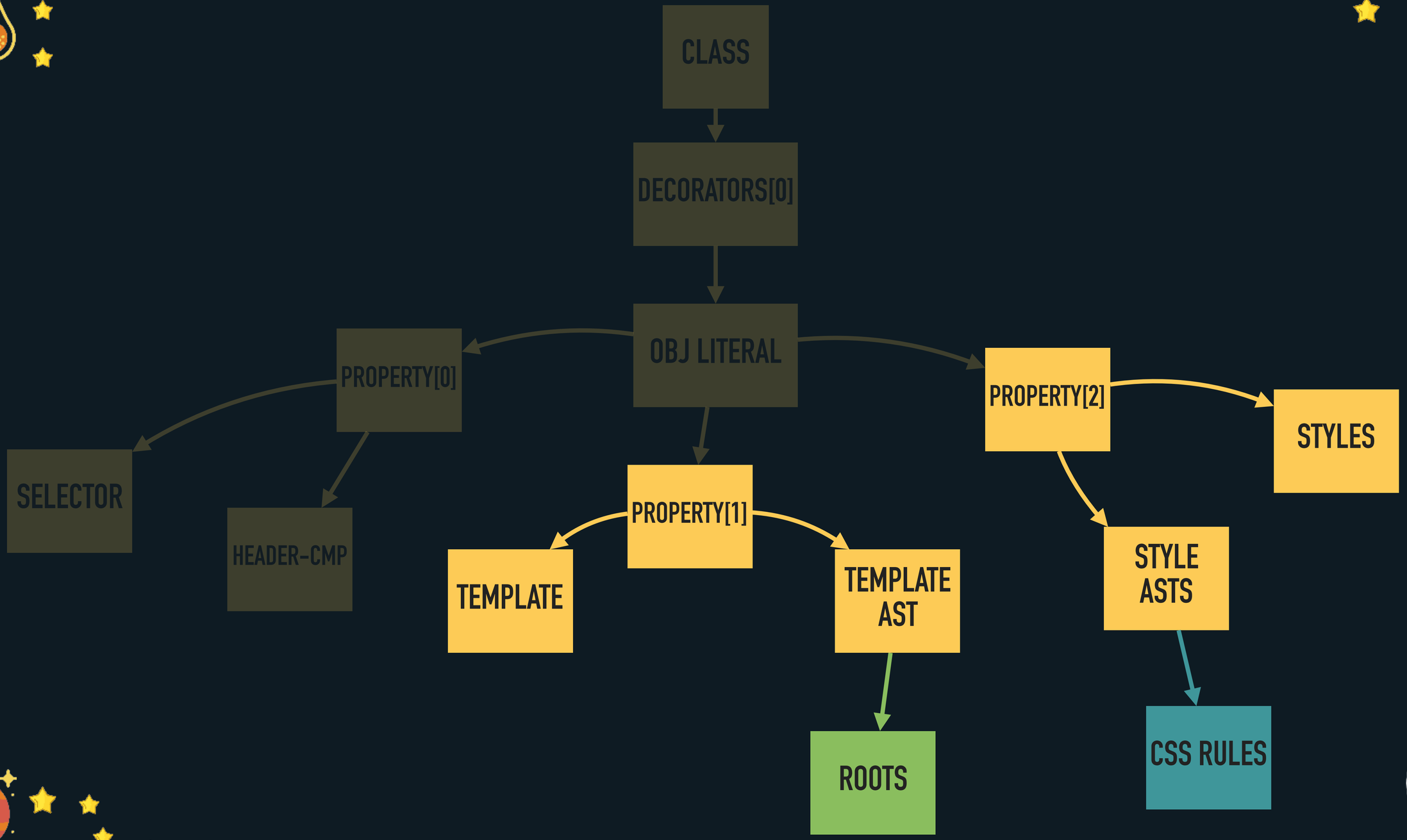
```
@Component({
  selector: 'header-cmp',
  template: `
    <header>
      <h1>{{ title }}</h1>
      <div class="greeting">Hello <span>{{ name }}</span>!</div>
    </header>
  `,
  styles: [`
    h1 {
      font-size: 25px;
    }

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

```
@Component({
  selector: 'header-cmp',
  template: `
    <header>
      <h1>{{ title }}</h1>
      <div class="greeting">Hello <span>{{ name }}</span>!</div>
    </header>
  `,
  styles: [`
    h1 {
      font-size: 25px;
    }

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


CLASS
DECORATORS[0]
OBJ LITERAL
PROPERTY[0]
SELECTOR
HEADER-CMP
PROPERTY[1]
TEMPLATE
'...'
PROPERTY[2]
STYLES
['...']


CLASS
DECORATORS[0]
OBJ LITERAL
PROPERTY[0]
SELECTOR
HEADER-CMP
PROPERTY[1]
TEMPLATE
'...'
PROPERTY[2]
STYLES
['...']

Strings Are Hard To Analyze


CLASS
DECORATORS[0]
OBJ LITERAL
PROPERTY[0]
SELECTOR
HEADER-CMP
PROPERTY[1]
TEMPLATE
TEMPLATE AST
ROOTS
PROPERTY[2]
STYLES
STYLE ASTS
CSS RULES

```
@Component({
  selector: 'sd-app',
  template: `
    <h1 class="header">Weather Forecast</h1>
    <div [class.hot]="temperature > 85">{{ temperature }}</div>
  `,
  styles: [
    `
    h1.head {
      font-size: 22px;
    }
    .hot {
      color: red;
    }
    div.medium {
      color: orange;
    }
    .cold {
      color: blue;
    }
    `
  ]
})
export class AppComponent {
  temperature = 80;
}
```

4GIFS.com

Current Limitations

- Linting per file - not entire program
  - Doesn’t perform type checking
- Will fail when selector is variable

```
import { Component as NgComponent }
  from '@angular/core';

const selector = '[tool' + 'tip]';

@NgComponent({
  selector,
  template: '...'
})
class TooltipComponent { ... }
```

```
import { Component as NgComponent }
  from '@angular/core';

const selector = '[tool' + 'tip]';

@NgComponent({
  selector,
  template: '...'
})
class TooltipComponent { ... }
```

```
import { Component as NgComponent }
  from '@angular/core';

const selector = '[tool' + 'tip]';

@NgComponent({
  selector,
  template: '...'
})
class TooltipComponent { ... }
```

ngast

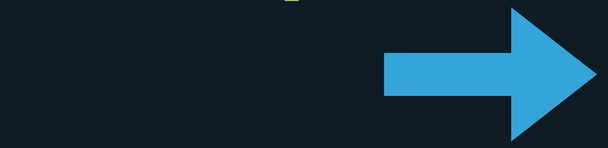
github.com/mgechev/ngast

ngast

- Provides high-level API to the compiler
- Allows:
  - Deep metadata collection
  - Static evaluation of foldable expressions

```
import { Component as NgComponent }
  from '@angular/core';

const selector = '[tool' + 'tip]';

@NgComponent({
  selector,
  template: '...'
})
class TooltipComponent { ... }
```

```
import { Component as NgComponent }
  from '@angular/core';

const selector = '[tool' + 'tip]';

@NgComponent({
  selector,
  template: '...'
})
class TooltipComponent { ... }
```



```
{
  "isHost": false,
  "isComponent": true,
  "selector": "[tooltip]",
  "changeDetection": 1,
  "inputs": {},
  "outputs": {},
  "template": {
    "template": "...",
    ...
  },
  ...
}
```

Codelyzer 3

- Deep metadata collection
- Autofixes
- Much more!

Thank you!

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