Tastydoc

A documentation tool for dotty using TASTy files

Bryan Abate

19th June 2019

Table of Contents

Introduction

Features

Architecture

Dottydoc vs Tastydoc

Problems & Further work

What is Tastydoc ?

- Documentation tool for Dotty

What is Tastydoc ?

- Documentation tool for Dotty
- Uses TASTy files

What is Tastydoc ?

- Documentation tool for Dotty
- Uses TASTy files
- Outputs Markdown

Table of Contents

Introduction

Features

Architecture

Dottydoc vs Tastydoc

Problems & Further work

Accessible information

TASTy

-

Linking

Markdown

-

Table of Contents

Introduction

Features

Architecture

Dottydoc vs Tastydoc

Problems & Further work

Representation

Reference

User documentation

-

Workflow

Table of Contents

Introduction

Features

Architecture

Dottydoc vs Tastydoc

Problems & Further work

General comparison

- Compiler internals

General comparison

- Compiler internals
- Markdown vs HTML/CSS

Extra features

- Scope modifiers

Extra features

- Scope modifiers
- Known subclasses

Bugs fixed

- Buggy output

Example

```
[31m2L[0m
```

Bugs fixed

- Buggy output

Example

```
[31m2L[0m
```

- Wrong parents

Bugs fixed

- Buggy output

Example

```
[31m2L[0m
```

- Wrong parents
- Annotations

Bugs fixed

- Buggy output

Example

```
[31m2L[0m
```

- Wrong parents
- Annotations
- Compiler artifacts

Table of Contents

Introduction

Features

Architecture

Dottydoc vs Tastydoc

Problems & Further work

Problems

- Markdown escaping

Problems

- Markdown escaping
- Linking inside code blocks

Problems

- Markdown escaping
- Linking inside code blocks
- Section

Problems

- Markdown escaping
- Linking inside code blocks
- Section
- IDs for linking

Further work

- Markdown escaping

Further work

- Markdown escaping
- Type lambdas

Further work

- Markdown escaping
- Type lambdas
- Complex types

```
class Graph {
    type Node = Int
}
def linkingGraph(g: Graph): g.Node = ???
```

Further work

- Markdown escaping
- Type lambdas
- Complex types

  ```
  class Graph {
      type Node = Int
  }
  def linkingGraph(g: Graph): g.Node = ???
  ```

- Default values

Further work

- Markdown escaping
- Type lambdas
- Complex types

```
class Graph {
    type Node = Int
}
def linkingGraph(g: Graph): g.Node = ???
```

- Default values
- Extra user-documentation parsing

Further work

- Markdown escaping
- Type lambdas
- Complex types

  ```
  class Graph {
      type Node = Int
  }
  def linkingGraph(g: Graph): g.Node = ???
  ```

- Default values
- Extra user-documentation parsing
- HTML/CSS