

Meet Mason



Introduction to Templating & Custom Code Generation



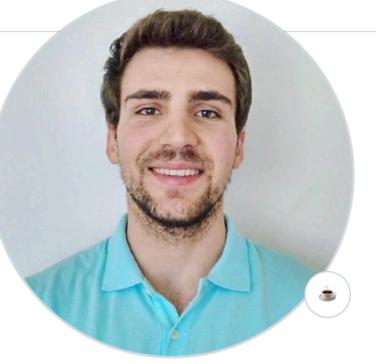
Felix Angelov @ Very Good Ventures



About Me



[Overview](#) [Repositories 105](#) [Projects](#) [Packages 2](#) [Stars 437](#)



Felix Angelov
felangel

Follow [Sponsor](#)

software engineer by day, software engineer by night.

6.3k followers · 60 following

@VGVentures
Chicago
@felangelov

Sponsors



Achievements

felangel / README.md

Hi there👋

I'm a software engineer at [Very Good Ventures](#) in Chicago, IL 🇺🇸

- 🔭 I'm currently working on the [bloc library](#)
- ☕️ I'm learning how to make latte art
- 🐱 I'm looking to collaborate on open source projects
- 💬 Ask me about [Flutter](#) and [Dart](#).
- 😄 Pronouns: he/him/his
- ⚡ Fun fact: I have a pet bunny named Coco 🐰

Open Source Projects

[bloc](#)
A predictable state management library that helps implement the BLoC design pattern

Dart ⭐ 9.4k ⚡ 2.8k

[bloc.js](#)
A predictable state management library that helps implement the BLoC design pattern in JavaScript

TypeScript ⭐ 170 ⚡ 19

[equatable](#)
A Dart package that helps to implement value based equality without needing to explicitly override == and hashCode.

Dart ⭐ 685 ⚡ 84

[fresh](#)
A token refresh library for Dart.

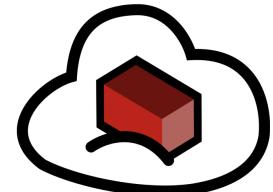
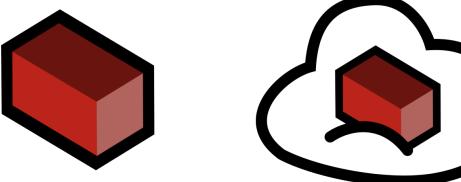
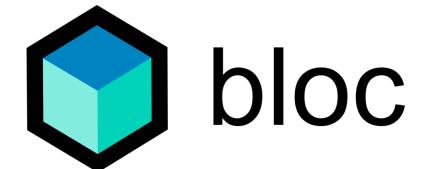
Dart ⭐ 227 ⚡ 35

[flow_builder](#)
Flutter Flows made easy! A Flutter package which simplifies navigation flows with a flexible, declarative API.

Dart ⭐ 315 ⚡ 49

[mocktail](#)
A mock library for Dart inspired by Mockito

Dart ⭐ 379 ⚡ 51



Story Time



I joined a new team and picked up a new feature...



Story Time



So I did what any developer would do...

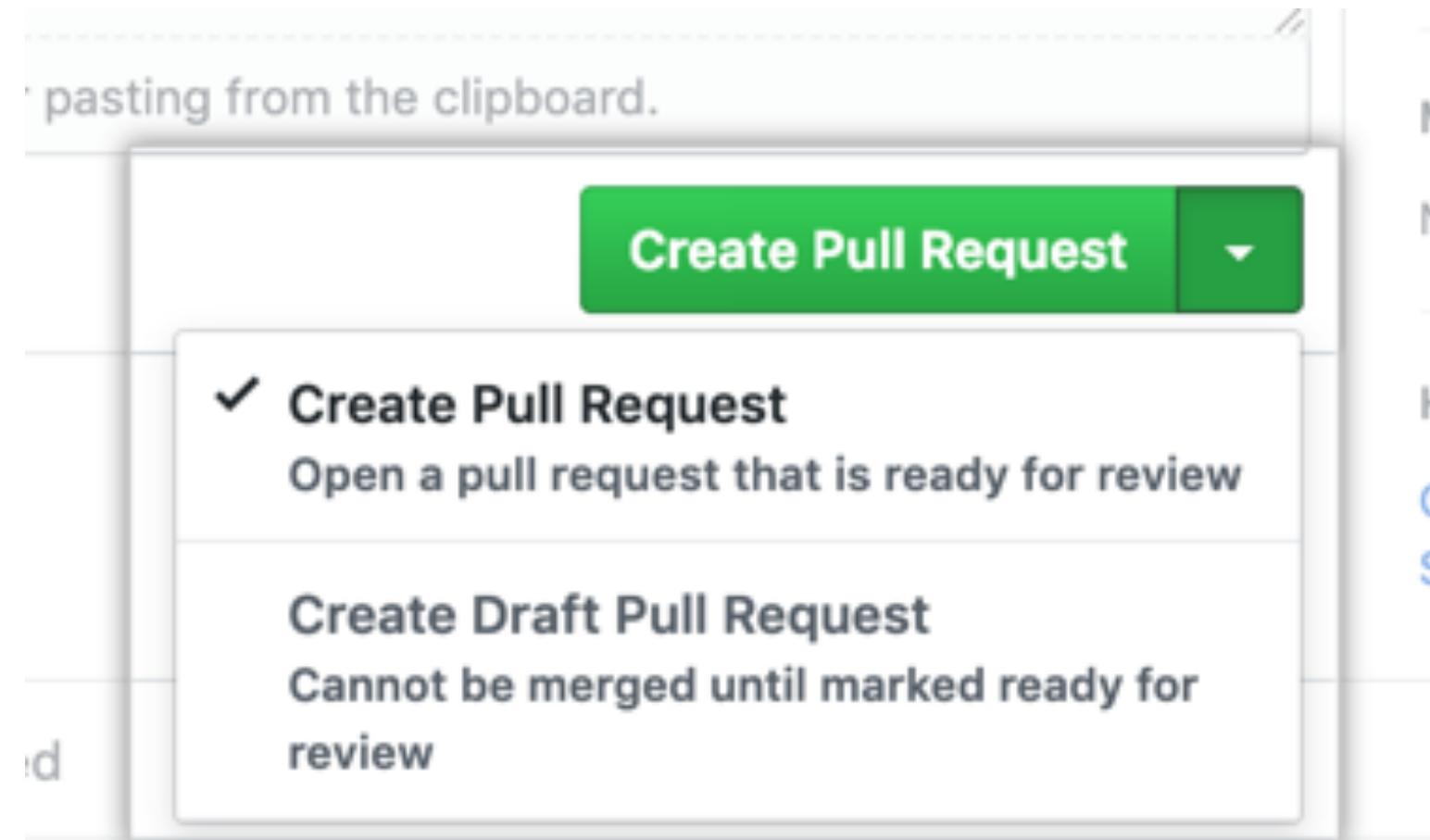


The Key 
Copy, Paste, Prosper

Story Time



And opened a pull request...



Story Time



👨‍💻 I copied a "legacy" feature! 😭

 **Changes requested**
1 review requesting changes [Learn more.](#) [Hide all reviewers](#)

 **1 change requested**

 **megbird** requested changes ...

 **This branch has no conflicts with the base branch**
Merging can be performed automatically.

Merge pull request ▾ You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Story Time



- 📝 Back to square one...
- ⏳ Lots of wasted time...
- 😢 Both sides frustrated...



I wish we could...

- Produce code in a way that is:
 -  Consistent
 -  Fast
 -  Customizable
 -  Reusable

Some Possible Solutions...

Copy + Paste

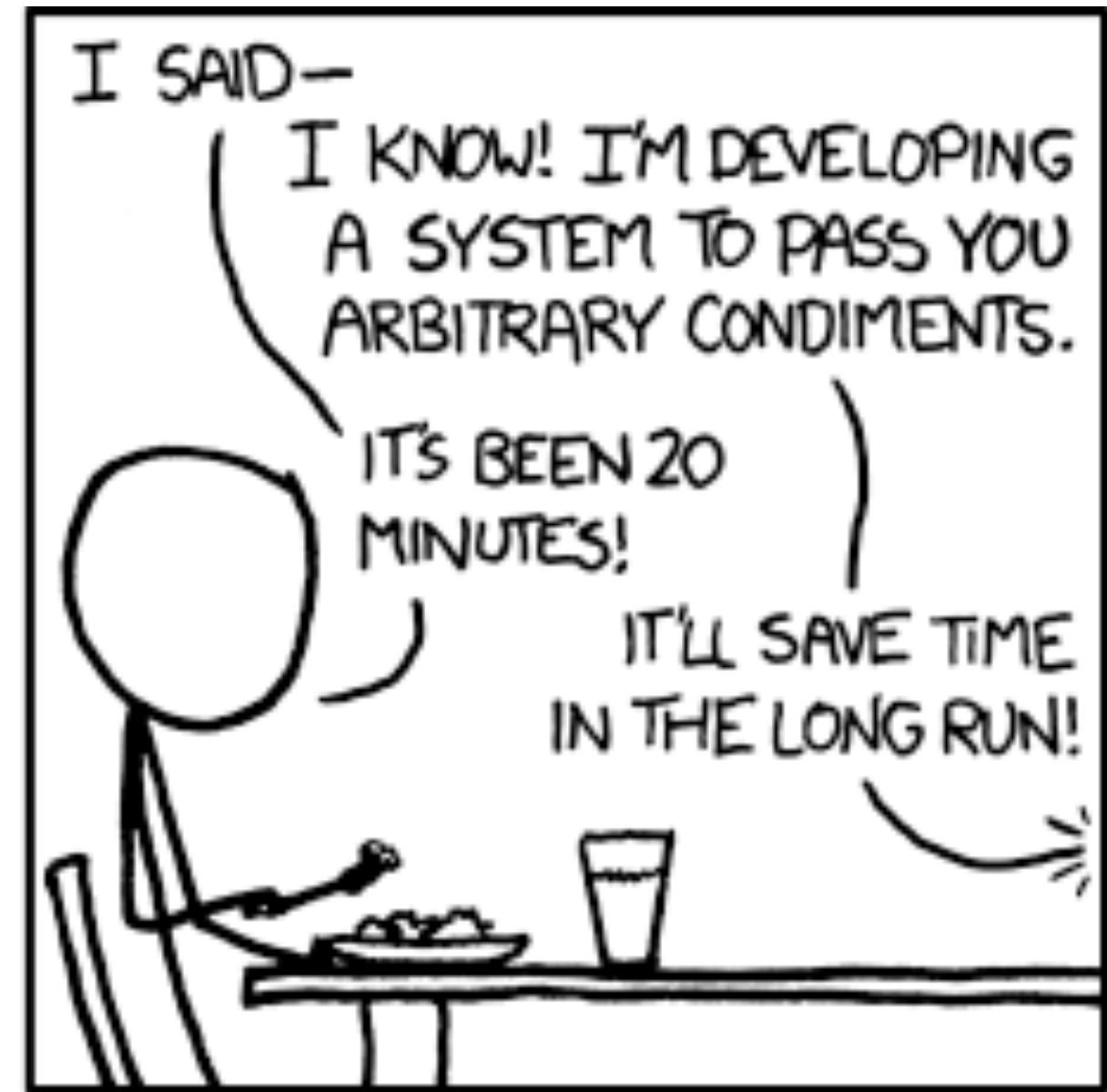
Gists

GitHub Templates

VSCode/IntelliJ Snippets

Custom Tooling (CLIs, IDE Extensions, etc.)

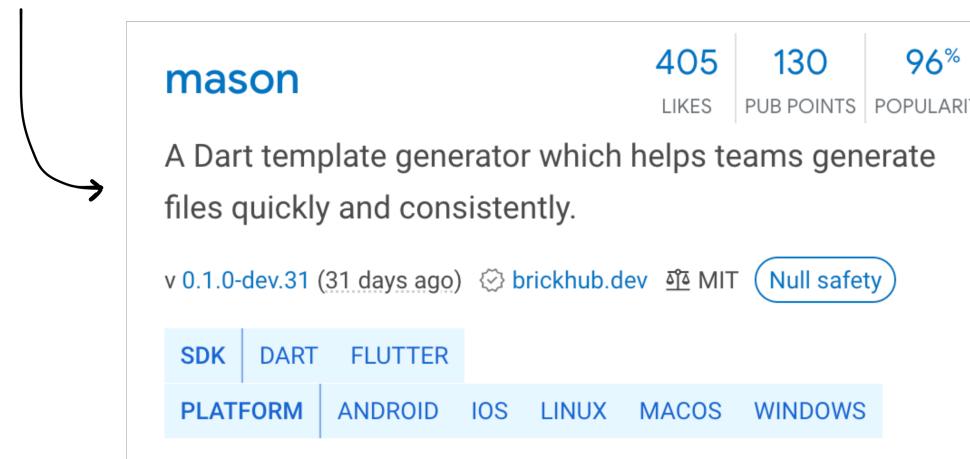
Or...



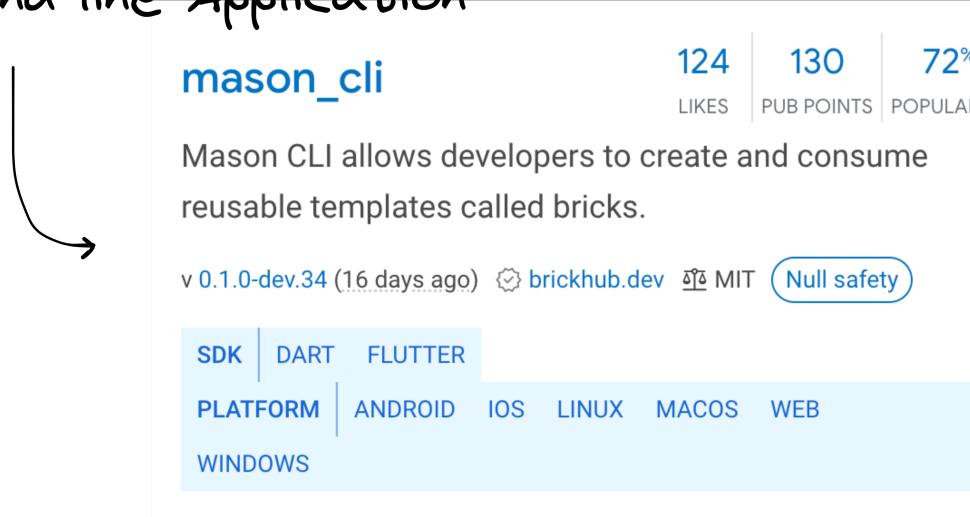
Meet Mason



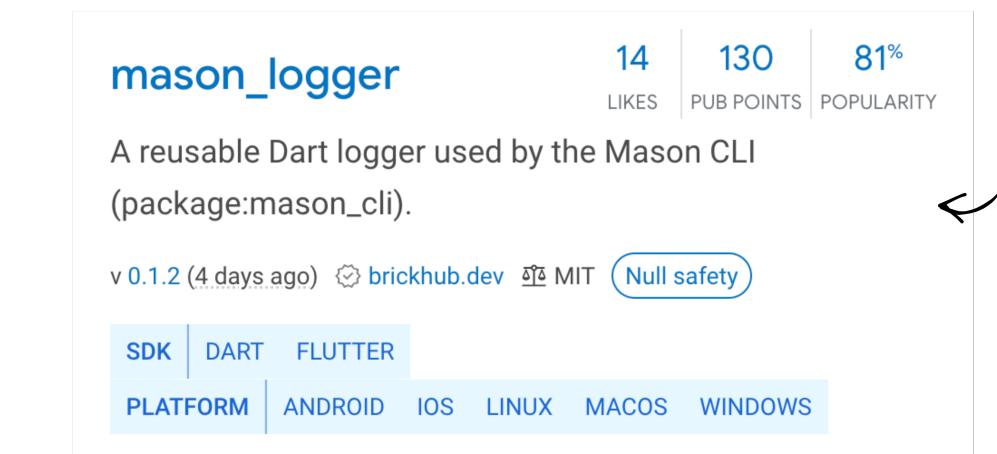
Core Templating Engine



Command-line Application



Reusable Logger



API Client for managing remote bricks



Intro to Mason CLI



- Create and consume reusable templates called bricks 🪚
- Powered by Dart and Mustache 🎯🤓
- Inspired by Stagehand, Yeoman, and Cookiecutter ✨

Chapter 1: Getting Started



- Install Mason CLI
- Initialize Mason in a Workspace
- Install a brick
- Use a brick

Installing Mason



```
# 🎯 Activate from https://pub.dev
dart pub global activate mason_cli
```

```
# 🍺 Install from https://brew.sh
brew tap felangel/mason
brew install mason
```

Mason CLI Overview

```
$ mason
 mason • lay the foundation!
```

Usage: mason <command> [arguments]

Global options:

-h, --help	Print this usage information.
--version	Print the current version.

Available commands: . . .

Chapter 1: Checkpoint



-  Install Mason CLI
-  Initialize Mason in a Workspace
-  Install a brick
-  Use a brick

```
$ mason init
```



Create a locally scoped workspace for working with bricks

- initializes mason in the current directory
 - generates a mason.yaml
- allows you to work with locally scoped bricks
- mason always uses nearest parent mason.yaml

\$ mason init in action

```
$ mason init
✓ Initializing (47ms)
✓ Getting bricks (10ms)
✓ Generated 1 file(s):
  /me/mason_playground/mason.yaml (new)
```

Run "mason make hello" to use your first brick.

Anatomy of the mason .yaml

```
# Register bricks which can be consumed via the Mason CLI.  
# https://github.com/felangel/mason  
bricks:  
  # Sample Brick  
  # Run `mason make hello` to try it out.  
  hello: "0.1.0+1"  
  # Bricks can also be imported via git url.  
  # Uncomment the following lines to import  
  # a brick from a remote git url.  
  # widget:  
  #   git:  
  #     url: https://github.com/felangel/mason  
  #     path: bricks/widget
```

Chapter 1: Checkpoint



-  Install Mason CLI
-  Initialize Mason in a Workspace
-  Install a brick
-  Use a brick

```
$ mason get 
```

Install all bricks registered in the nearest parent mason.yaml

- Think dart pub get
- Generated mason-lock.json
- Bricks are cached locally for offline use
 - add .mason to .gitignore

\$ mason get in action

```
$ mason get
```

```
✓ Getting bricks (22ms)
```



```
$ mason list
```



List all installed bricks

- Use mason ls shorthand
- Outputs installed bricks
- Defaults to locally installed bricks
- Use --global or -g for globally installed bricks

\$ mason list in action

```
$ mason list  
/me/mason_playground  
└ hello 0.1.0+1 -> registry.brickhub.dev
```

\$ mason list in action

```
# Change to a directory outside the workspace  
$ cd /me/dart_playground
```

```
# List available bricks  
$ mason ls  
/me/dart_playground  
└ (empty)
```

Chapter 1: Checkpoint



-  Install Mason CLI
-  Initialize Mason in a Workspace
-  Install a brick
-  Use a brick

```
$ mason make
```



Generate code from a brick

- looks up brick metadata
- prompts for any required variables
- generates code in the desired output directory

\$ mason make in action

```
$ mason make hello  
? What is your name? (Dash) Felix  
✓ Made brick hello (52ms)  
✓ Generated 1 file:  
/me/mason_playground/HELLO.md (new)
```

HELLO.md

Hello Felix! 

\$ mason make: command-line args



```
$ mason make hello --name Felix
✓ Made brick hello (41ms)
✓ Generated 1 file:
  /me/mason_playground/HELLO.md (identical)
```

\$ mason make: config file



```
$ mason make hello -c config.json
✓ Made brick hello (41ms)
✓ Generated 1 file:
  /me/mason_playground/HELLO.md (identical)
```

config.json

```
{
  "name": "Felix"
}
```

\$ mason make: output directory



```
$ mason make hello --name Felix -o ./out
✓ Made brick hello (41ms)
✓ Generated 1 file:
  /me/mason_playground/out/HELLO.md (new)
```

\$ mason make: conflicts !

```
$ mason make hello --name Dash  
conflict /me/mason_playground/HELLO.md  
Overwrite HELLO.md? (Yyna) y  
✓ Made brick hello (32.5s)  
✓ Generated 1 file:  
  /me/mason_playground/HELLO.md (new)
```

BEFORE

Hello Felix! 

AFTER

Hello Dash! 

Conflict Resolution Strategies

By default, mason will prompt on each file conflict

Options:

- y - yes, overwrite (default)
- Y - yes, overwrite this and all others
- n - no, do not overwrite
- a - append to existing file

\$ mason make: conflict resolution

? Always prompt when there is a file conflict (default)

```
$ mason make hello --name Dash --on-conflict prompt
```

🖊 Always overwrite when there is a file conflict

```
$ mason make hello --name Dash --on-conflict overwrite
```

🤝 Always skip when there is a file conflict

```
$ mason make hello --name Dash --on-conflict skip
```

+ Always append when there is a file conflict

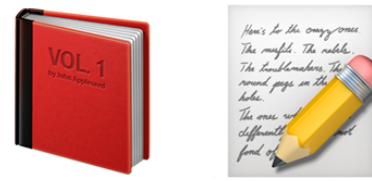
```
$ mason make hello --name Dash --on-conflict append
```

Chapter 1 Complete



- Install Mason CLI
- Initialize Mason in a Workspace
- Install a brick
- Use a brick

Chapter 1 Summary



🎯 Activate from <https://pub.dev>

```
$ dart pub global activate mason_cli
```

📁 Initialize mason in the current workspace

```
$ mason init
```

☁️ Install all bricks defined in `mason.yaml`

```
$ mason get
```

🚧 Generate code from a brick

```
$ mason make hello
```

Chapter 2: Creating a Brick



- Generating a new brick
- Anatomy of a brick
- Brick template syntax
- Hooks

```
$ mason new
```



Create a new brick template

- creates a brick in the current directory
 - generates a `brick.yaml`
 - generates a `__brick__` directory
- includes `README`, `LICENSE`, and `CHANGELOG`

\$ mason new in action

```
$ mason new example
✓ Created new brick: example (74ms)
✓ Generated 5 file(s):
  /me/mason_playground/example/brick.yaml (new)
  /me/mason_playground/example/README.md (new)
  /me/mason_playground/example/CHANGELOG.md (new)
  /me/mason_playground/example/LICENSE (new)
  /me/mason_playground/example/__brick__/HELLO.md (new)
```

Chapter 2: Checkpoint



- Generating a new brick
- Anatomy of a brick
- Brick template syntax
- Hooks

Anatomy of a Brick



```
.  
├── CHANGELOG.md  
├── LICENSE  
├── README.md  
└── __brick__  
    └── brick.yaml
```

- **__brick__**
 - brick template
- **brick.yaml**
 - brick metadata
(pubspec.yaml)

Anatomy of the brick .yaml

Defines the metadata for a specific brick

```
name: example
description: An example brick
version: "0.1.0+1"

environment:
  mason: ">=0.1.0-dev <0.1.0"

vars:
  name:
    type: string
    description: Your name.
    default: Dash
    prompt: What is your name?
```

- can contain zero or more variables
- variable types include:
 - string, number, boolean, enum, array

Variable Type: array

```
vars:  
  flavors:  
    type: array  
    description: Supported flavors  
    prompt: What flavors would you like to generate?  
  defaults:  
    - development  
    - production  
  values:  
    - development  
    - integration  
    - staging  
    - production
```

```
$ mason make example  
? What flavors would you like to generate?  
➤ ● development  
○ integration  
○ staging  
● production
```

Variable Type: enum

```
vars:  
  license:  
    type: enum  
    default: MIT license  
    prompt: "Choose a License:"  
  values:  
    - "Apache License 2.0"  
    - 'BSD 3-Clause "New" or "Revised" license'  
    - "GNU General Public License (GPL)"  
    - "MIT license"  
    - "Mozilla Public License 2.0"
```

```
$ mason make example  
? Choose a License:  
  ○ Apache License 2.0  
  ○ BSD 3-Clause "New" or "Revised" license  
  ○ GNU General Public License (GPL)  
  > ● MIT license  
  ○ Mozilla Public License 2.0
```

Anatomy of the `__brick__`

`__brick__`

```
├── README.md  
├── lib  
│   └── {{name}}.dart  
└── pubspec.yaml
```

output

```
├── README.md  
├── lib  
│   └── flutter_vikings_2022.dart  
└── pubspec.yaml
```

vars

```
{  
    "name": "flutter_vikings_2022"  
}
```

Putting it Together

```
├── CHANGELOG.md  
├── LICENSE  
├── README.md  
└── __brick__  
    └── HELLO.md  
└── brick.yaml
```

`__brick__/HELLO.md`

Hello {{name}}! 

`brick.yaml`

```
name: hello  
description: An example brick  
version: "0.1.0+1"
```

`vars:`

`name:`

`type: string`

`description: Your name.`

`prompt: What is your name?`

\$ mason make hello

? What is your name? Vikings

`HELLO.md`

Hello Vikings! 

Chapter 2: Checkpoint



- Generating a new brick
- Anatomy of a brick
- Brick template syntax
- Hooks

Template Syntax

- Powered by Mustache 😎
 - [package:mustache_template](#)
 - <https://mustache.github.io/mustache.5.html>

Template Syntax: Conditionals

`__brick__/pubspec.yaml`

```
{ {^publish} }
publish_to: none
{/publish}

dependencies:
  flutter:
    sdk: flutter

{ #{useGoogleFonts} }
google_fonts: latest
{/useGoogleFonts}
```

`pubspec.yaml`

```
publish_to: none

dependencies:
  flutter:
    sdk: flutter

  google_fonts: latest
```

`vars`

```
{
  "publish": false,
  "useGoogleFonts": true
}
```

Template Syntax: Loops

`__brick__/README.md`

```
{ {#platforms} }  
{ { . } }  
{ {/platforms} }
```

`README.md`

iOS

Android

Web

`vars`

```
{  
  "platforms": ["iOS", "Android", "Web"]  
}
```

Template Syntax: Lambdas

`__brick__/{{name.snakeCase()}}.dart`

```
import 'package:flutter/widgets.dart';

class {{name.pascalCase()}} extends StatelessWidget {
  const {{name.pascalCase()}}({super.key});

  @override
  Widget build(BuildContext context) {
    return const SizedBox();
  }
}
```

`viking_ship.dart`

```
import 'package:flutter/widgets.dart';

class VikingShip extends StatelessWidget {
  const VikingShip({super.key});

  @override
  Widget build(BuildContext context) {
    return const SizedBox();
  }
}
```

`vars`

```
{ "name": "viking ship" }
```

Built-in Lambdas

Name	Example	Shorthand Syntax	Full Syntax
camelCase	helloWorld	<code>{variable.camelCase()}</code>	<code>{#camelCase}{variable}{/camelCase}</code>
constantCase	HELLO_WORLD	<code>{variable.constantCase()}</code>	<code>{#constantCase}{variable}{/constantCase}</code>
dotCase	hello.world	<code>{variable.dotCase()}</code>	<code>{#dotCase}{variable}{/dotCase}</code>
headerCase	Hello-World	<code>{variable.headerCase()}</code>	<code>{#headerCase}{variable}{/headerCase}</code>
lowerCase	hello world	<code>{variable.lowerCase()}</code>	<code>{#lowerCase}{variable}{/lowerCase}</code>
mustacheCase	<code>{ Hello World }</code>	<code>{variable.mustacheCase()}</code>	<code>{#mustacheCase}{variable}{/mustacheCase}</code>
pascalCase	HelloWorld	<code>{variable.pascalCase()}</code>	<code>{#pascalCase}{variable}{/pascalCase}</code>
paramCase	hello-world	<code>{variable.paramCase()}</code>	<code>{#paramCase}{variable}{/paramCase}</code>
pathCase	hello/world	<code>{variable.pathCase()}</code>	<code>{#pathCase}{variable}{/pathCase}</code>
sentenceCase	Hello world	<code>{variable.sentenceCase()}</code>	<code>{#sentenceCase}{variable}{/sentenceCase}</code>
snakeCase	hello_world	<code>{variable.snakeCase()}</code>	<code>{#snakeCase}{variable}{/snakeCase}</code>
titleCase	Hello World	<code>{variable.titleCase()}</code>	<code>{#titleCase}{variable}{/titleCase}</code>
upperCase	HELLO WORLD	<code>{variable.upperCase()}</code>	<code>{#upperCase}{variable}{/upperCase}</code>

Template Syntax: Partials

Nested Templates within other templates

__brick__

```
|── HELLO.md  
|── {{~ footer.md }}  
└── {{~ header.md }}
```

{{~ header.md }}

 {{name}}

HELLO.md

{{> header.md }}

Hello {{name}} !

{{~ footer.md }}

made with ❤ by mason

{{> footer.md }}

Chapter 2: Checkpoint



- Generating a new brick
- Anatomy of a brick
- Brick template syntax
- Hooks

Hooks



Custom code that executes before or after generation

```
$ mason new example --hooks
```

```
. └── CHANGELOG.md  
    └── LICENSE  
    └── README.md  
        └── __brick__  
            └── HELLO.md  
        └── brick.yaml  
    └── hooks  
        ├── .gitignore  
        ├── post_gen.dart  
        └── pre_gen.dart  
    └── pubspec.yaml
```

Hooks Example: pre_gen.dart

```
// pre_gen.dart
import 'dart:io';
import 'package:mason/mason.dart';

void run(HookContext context) {
    // Read vars.
    final name = context.vars['name'];

    // Use the `Logger` instance.
    context.logger.info('Hello $name!');

    // Update vars.
    context.vars['current_year'] = DateTime.now().year;
}
```

Hooks Example: post_gen.dart

```
// post_gen.dart
import 'dart:io';
import 'package:mason/mason.dart';

Future<void> run(HookContext context) async {
  final progress = context.logger.progress('Installing packages');

  // Run `flutter packages get` after generation.
  await Process.run('flutter', ['packages', 'get']);

  progress.complete();
}
```

Chapter 2 Complete

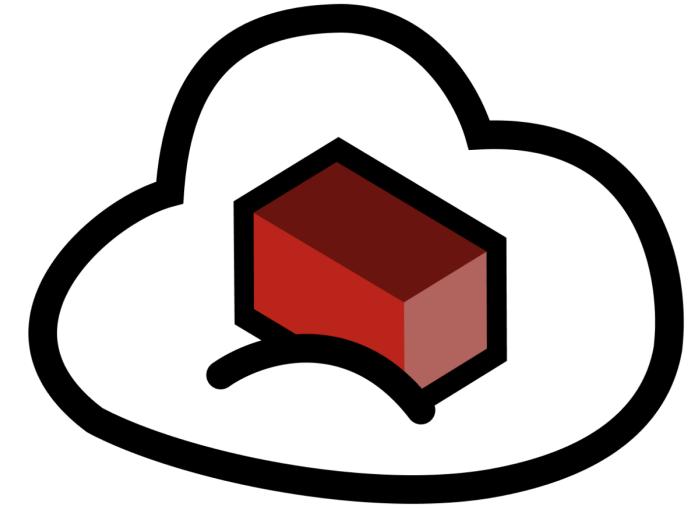


- Generating a new brick
- Anatomy of a brick
- Brick template syntax
- Hooks

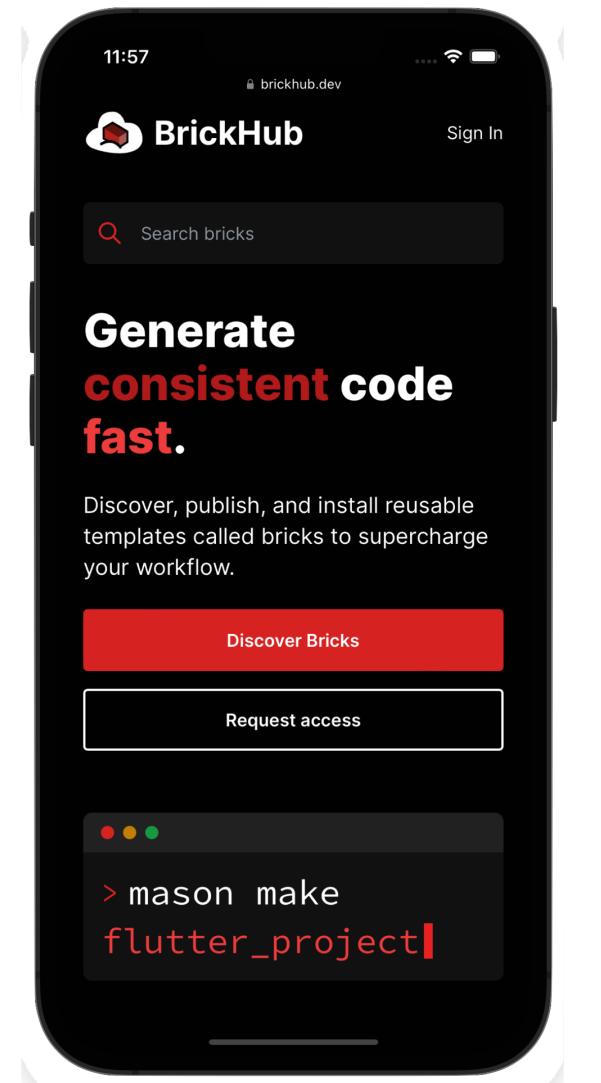
Chapter 3: Brick Management



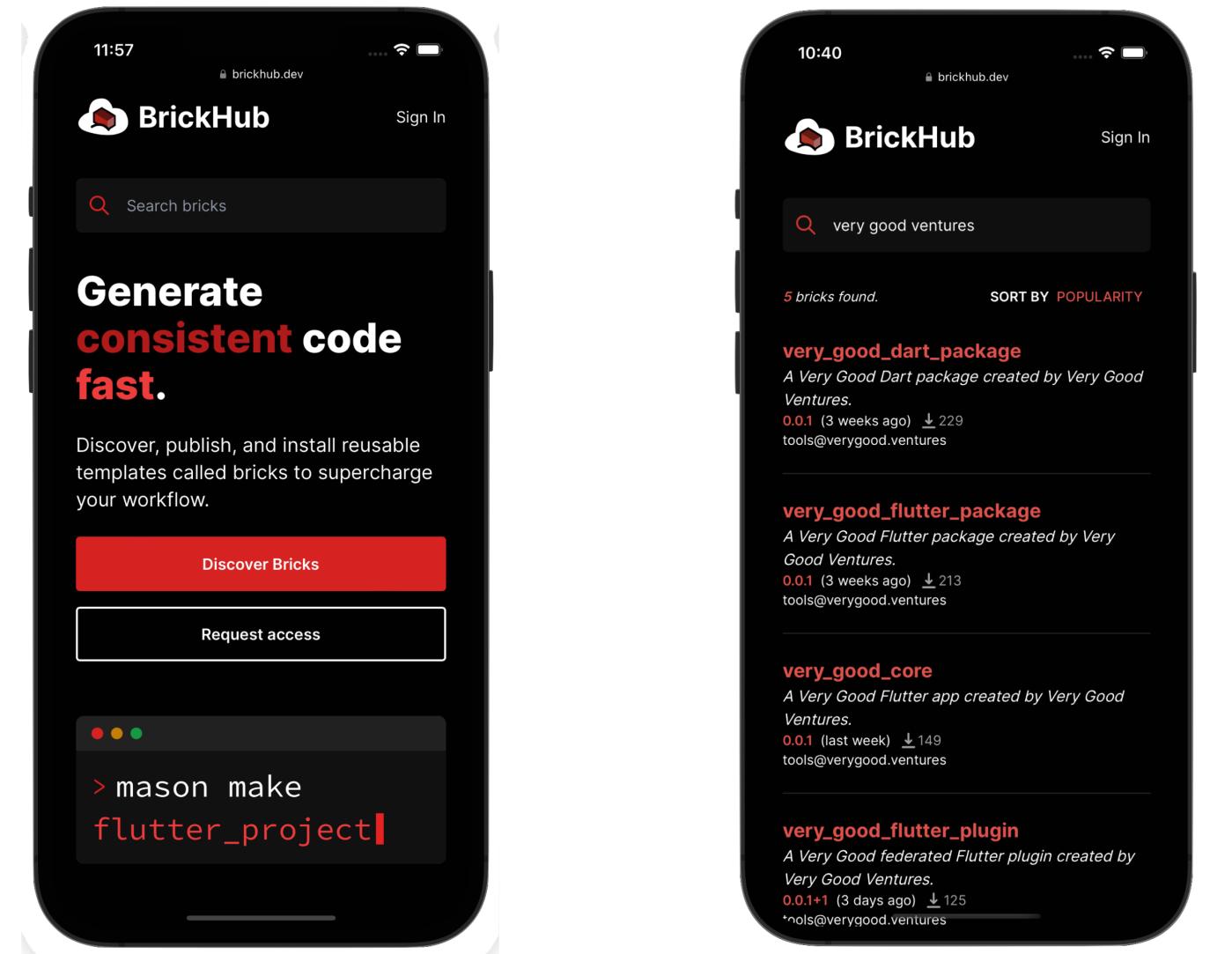
- Intro to BrickHub
- Publishing a brick
- Adding / Removing bricks
- IDE Integration



brickhub.dev

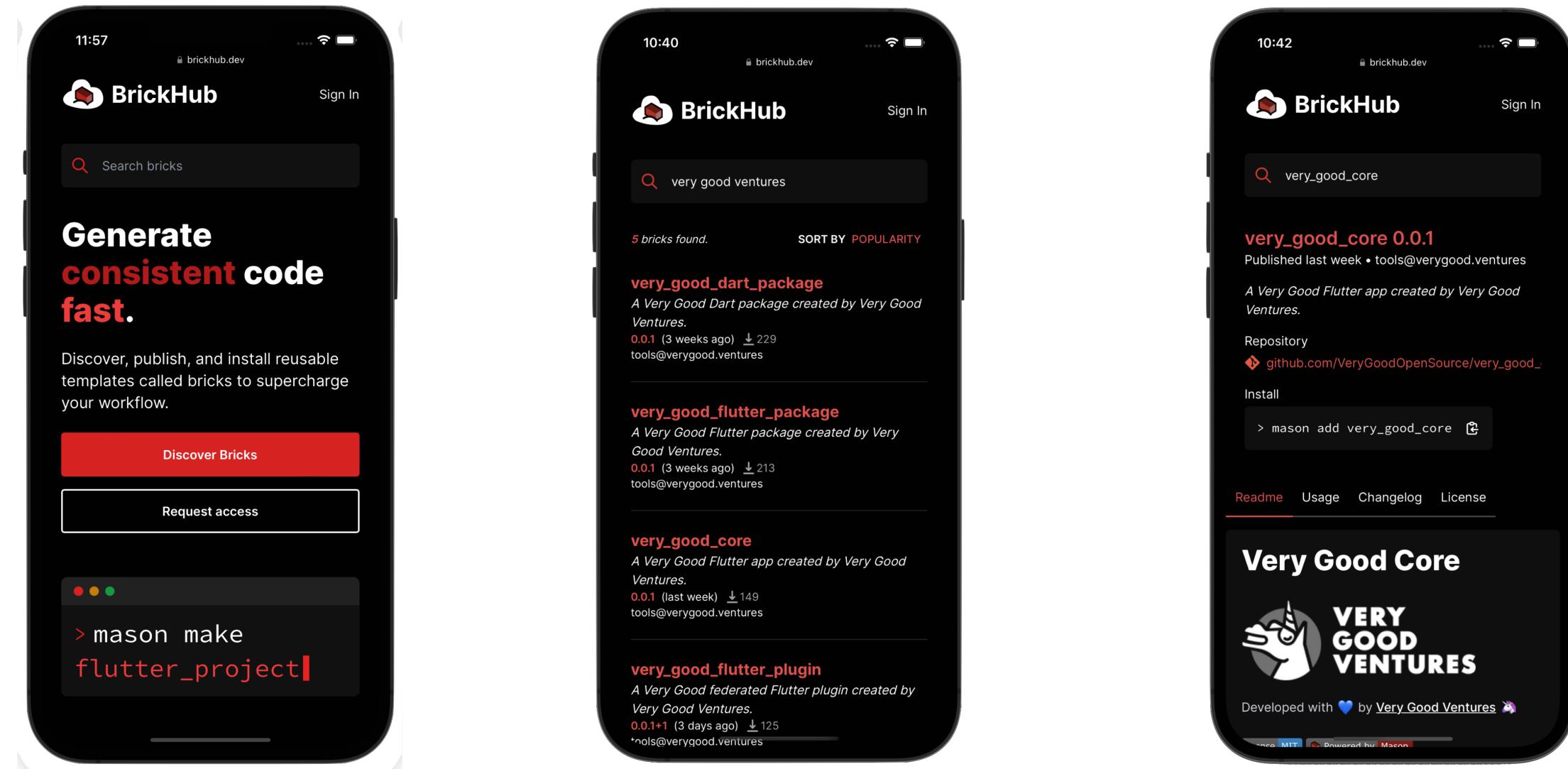


Install



Install

Discover



Install

Discover

Publish

basic_flame_game

Creates a basic flutter application which displays a Flame Game.
0.1.0+1 (2 months ago) [↓ 258](#)

flutterfire_starter

A template to have a quickstart with FlutterFire
0.1.0 (last month) [↓ 235](#)

bloc

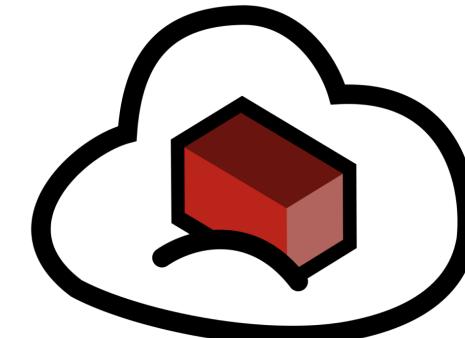
Generate a new Bloc in Dart. Built for the bloc state management library.
0.1.3+2 (last month) [↓ 685](#)

model

A brick to create your model with properties and all the supporting methods, copyWith, to/from json, equatable and more!
0.3.7 (2 months ago) [↓ 6033](#)

120+

bricks created



53,400+

brick downloads

feature_brick

A brick to create a feature using best practices and your state management of choice!
0.5.1 (2 weeks ago) [↓ 4400](#)

amplify_starter

A highly opinionated AWS Amplify starter project with Authenticator, Amplify Datastore and bloc state management.
0.2.2 (3 weeks ago) [↓ 704](#)

macosui_starter

A starter Flutter application for macOS that uses macos_ui
1.2.1+2 (last month) [↓ 828](#)

github_actions_dart

A brick that simplifies generating Github actions for Dart/Flutter monorepos
0.0.11 (last month) [↓ 1304](#)

Chapter 3: Checkpoint



- Intro to BrickHub
- Publishing a brick
- Adding / Removing bricks
- IDE Integration

Signing Up



- 🤝 Request access at <https://brickhub.dev>
- 📬 Receive email invite
- ✎ Sign up at <https://brickhub.dev/signup>
- 📧 Verify email

Logging In



Log into brickhub.dev

```
$ mason login  
email: me@email.com  
password: *****  
✓ Logged into brickhub.dev (0.5s)  
You are now logged in as <me@email.com>
```

Publishing a Brick



Publish a brick to `brickhub.dev`

```
$ mason publish --directory ./my_brick  
Do you want to publish my_brick 0.1.0+1? (y/N) y  
✓ Bundled my_brick (0.1s)  
✓ Published my_brick to brickhub.dev (0.1s)
```

Logging Out



Log out of brickhub.dev

```
$ mason logout  
✓ Logged out of brickhub.dev (3ms)
```

Chapter 3: Checkpoint



- Intro to BrickHub
- Publishing a brick
- Adding / Removing bricks
- IDE Integration

Adding Bricks to Workspace



```
# Install example from path  
$ mason add example --path ./example
```

`mason.yaml`

```
bricks:  
  example:  
    path: ./example
```

Adding Bricks to Workspace



```
# Install widget from git  
$ mason add widget  
  --git-url https://github.com/felangel/mason  
  --git-path bricks/widget
```

mason.yaml

```
bricks:  
  widget:  
    git:  
      url: https://github.com/felangel/mason  
      path: bricks/widget
```

Adding Bricks to Workspace



```
# Install amplify_starter from https://brickhub.dev  
$ mason add amplify_starter
```

mason.yaml

```
bricks:  
  amplify_starter: ^0.2.2
```

Let's List 'em



```
$ mason ls  
/me/mason_playground  
└─ amplify_starter 0.2.2 -> registry.brickhub.dev  
└─ example 0.1.0+1 -> /me/mason_playground/example  
└─ widget 0.1.0+1 -> https://github.com/felangel/mason  
  
$ cd /me/dart_playground  
$ mason ls  
/me/dart_playground  
└─ (empty)
```

Adding Bricks Globally



Globally installed bricks can be used from anywhere on your machine

```
# Install example from path  
$ mason add -g example --path ./example  
  
# Install widget from git  
$ mason add -g widget  
  --git-url https://github.com/felangel/mason  
  --git-path bricks/widget  
  
# Install amplify_starter from https://brickhub.dev  
$ mason add -g amplify_starter
```

Listing Global Bricks



```
$ cd /me/dart_playground
```

```
$ mason ls  
/me/dart_playground  
└─ (empty)
```

```
$ mason ls -g  
/me/.mason-cache/global  
├─ amplify_starter 0.2.2 -> registry.brickhub.dev  
├─ example 0.1.0+1 -> /me/mason_playground/example  
└─ widget 0.1.0+1 -> https://github.com/felangel/mason
```

Removing Bricks



```
# Remove local brick  
$ mason remove example  
✓ Removed example (22ms)
```

```
# Remove global brick  
$ mason remove -g widget  
✓ Removed widget (21ms)
```

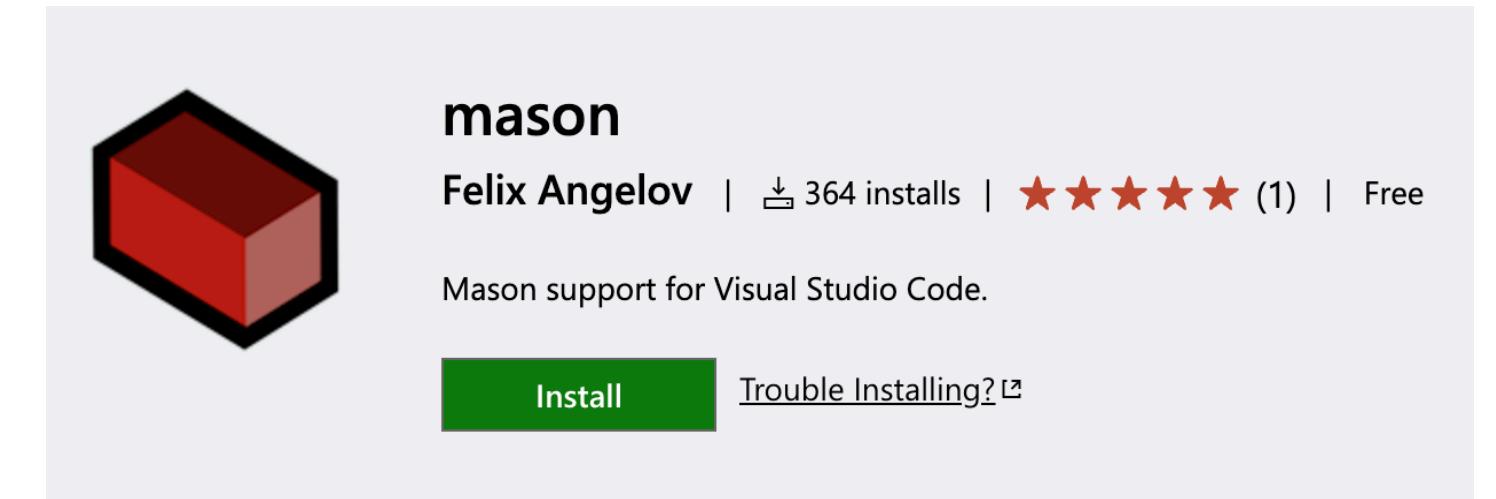
Chapter 3: Checkpoint

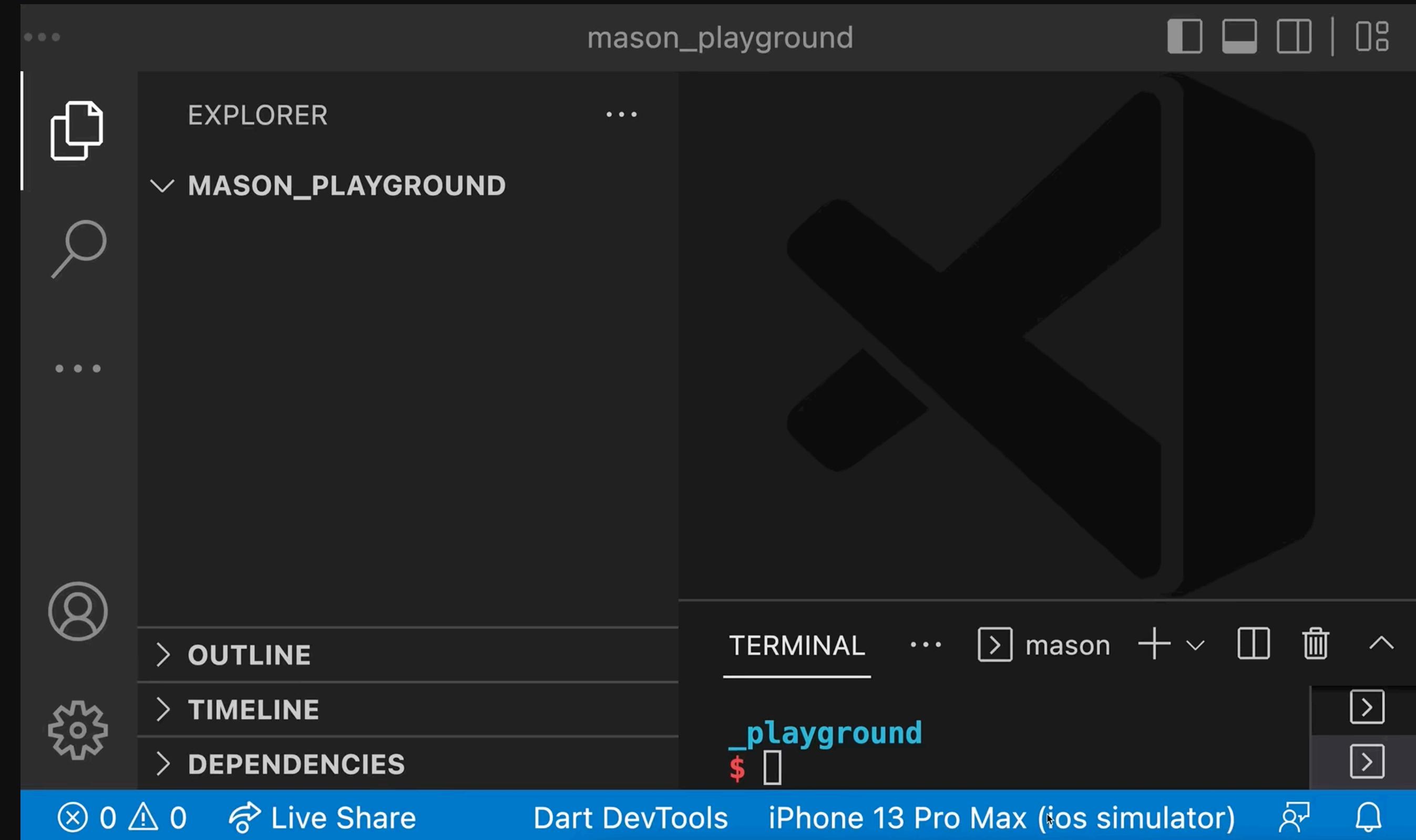


- Intro to BrickHub
- Publishing a brick
- Adding / Removing bricks
- IDE Integration

IDE Integration ⚡

- mason init command
- mason add and mason remove commands
- mason make via right-click
- mason get on save





Chapter 3: Complete



- Intro to BrickHub
- Publishing a brick
- Adding / Removing bricks
- IDE Integration

Now



When a new teammate joins and picks up a new feature...



Now



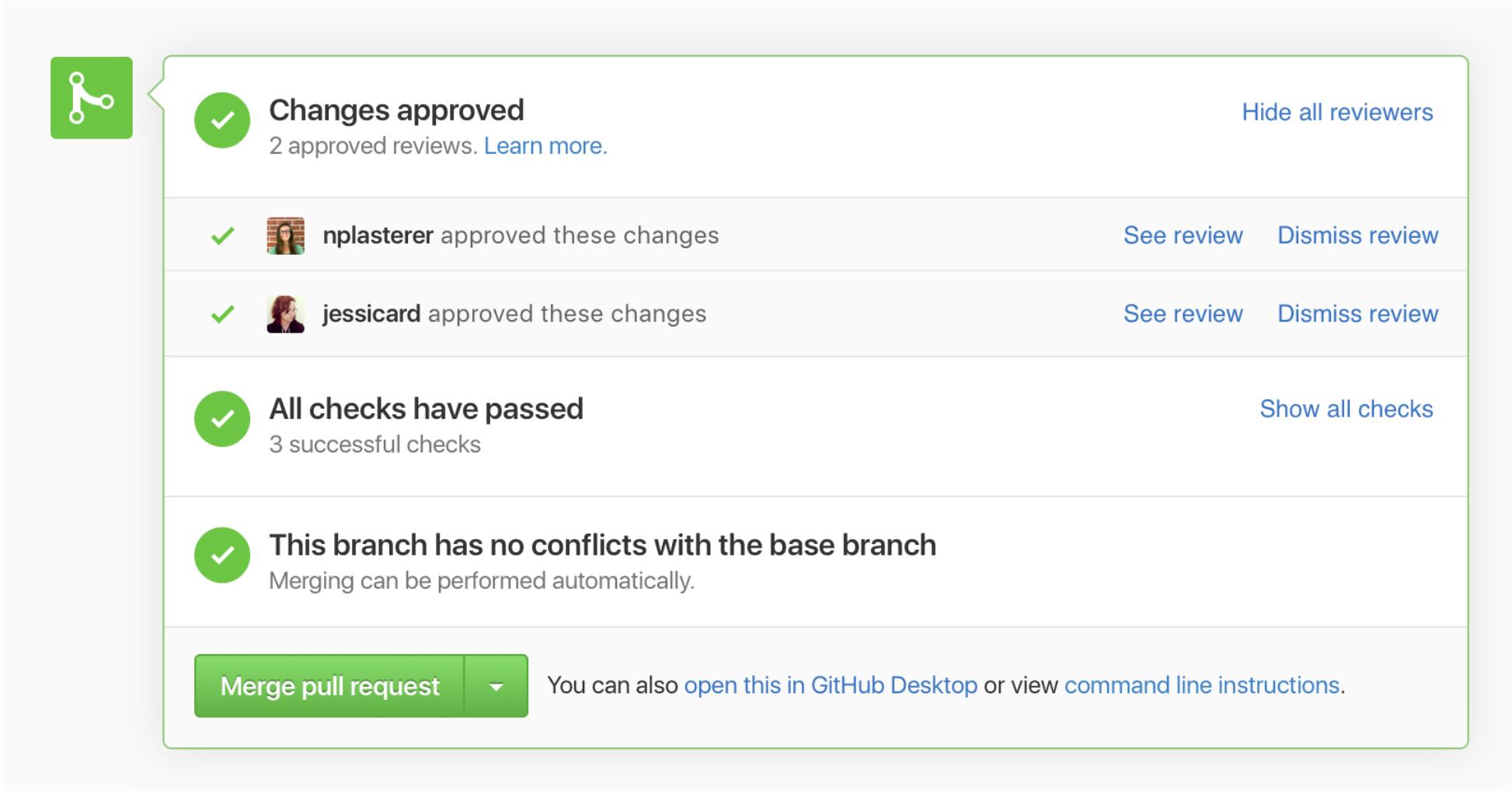
Use mason to generate a new feature...

```
$ mason make feature_brick
? What is the feature name? login
✓ Made brick feature_brick (0.3s)
✓ Generated 8 file(s):
/me/mason_playground/my_app/lib/login/widgets/login_body.dart (new)
/me/mason_playground/my_app/lib/login/view/login_page.dart (new)
/me/mason_playground/my_app/lib/login/bloc/login_event.dart (new)
/me/mason_playground/my_app/lib/login/bloc/login_state.dart (new)
/me/mason_playground/my_app/lib/login/widgets/widgets.dart (new)
/me/mason_playground/my_app/lib/login/login.dart (new)
/me/mason_playground/my_app/lib/login/bloc/bloc.dart (new)
/me/mason_playground/my_app/lib/login/bloc/login_bloc.dart (new)
```

Now



And get that pull request merged efficiently 🎉



The image shows a screenshot of a GitHub pull request merge interface. At the top left is a green icon with a person and a gear. To its right, a green circle with a checkmark contains the text "Changes approved". Below this, it says "2 approved reviews." and a link "Learn more.". On the far right is a blue link "Hide all reviewers".

Below the review section, there are two entries, each with a green checkmark and a user profile picture:

- nplasterer** approved these changes. To the right are "See review" and "Dismiss review" links.
- jessicard** approved these changes. To the right are "See review" and "Dismiss review" links.

Below these is a section with a green checkmark and the text "All checks have passed". It says "3 successful checks" and a blue link "Show all checks".

Further down is another section with a green checkmark and the text "This branch has no conflicts with the base branch". It says "Merging can be performed automatically."

At the bottom left is a green button with white text that says "Merge pull request" followed by a dropdown arrow. To its right, a note says "You can also open this in GitHub Desktop or view command line instructions."

Let's Recap



- Mason allows us to generate code:
 - Consistently
 - Fast
 - Customizable
 - Reusable

Roadmap



- Multiple Template Engine Support
- Template Extensions
- Multiple Publishers
- IntelliJ Plugin

Thank You!



Twitter @felangelov
GitHub @felangel