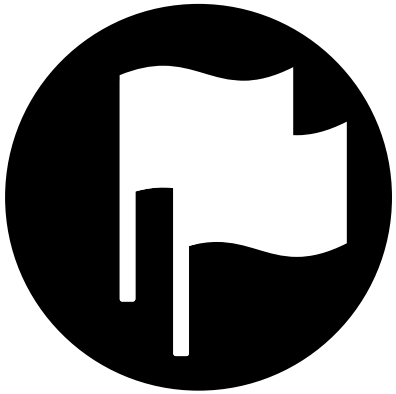


FeatureFlags

feature flags, a/b & mvt testing framework for ios

Feature Flags

- Configured via JSON
- Supports feature flags, A/B testing and MVT testing
- Automatically rollout a feature to 100%
- Visualise the state of flags and tests

Configuration

- JSON file may be locally bundled or remote
- Features can be included as part of existing configuration file or defined in an entirely new file

```
{
    "features": []
}
```

Flags & Tests

- Feature flag
- A/B test
- Feature A/B test
- MVT test

Flags & Tests

```
{
    "features": [{
        "name": "Example Feature Flag",
        "enabled": false
    }, {
        "name": "Example A/B Test",
        "enabled": true, // whether or not the test is enabled
        "test-variations": ["Group A", "Group B"],
        "labels": ["label1-for-analytics", "label2-for-analytics"]
    }, {
        "name": "Example Feature On/Off A/B Test",
        "enabled": true,
        "test-biases": [80, 20],
        "test-variations": ["enabled", "disabled"],
        "labels": ["label1-for-analytics", "label2-for-analytics"]
    }, {
        "name": "Example MVT Test",
        "enabled": true, // whether or not the test is enabled
        "test-biases": [70, 20, 10],
        "test-variations": ["Group A", "Group B", "Group C"],
        "labels": ["label1-for-analytics", "label2-for-analytics", "label3-for-
analytics"]
    }]
}
```

Phased Roll Out

To phase roll out of a feature to users:

```
{
    "features": [{
        "name": "Example A/B Test",
        "enabled": true,
        "test-biases": [10, 90],
        "test-variations": ["Group A", "Group B"],
        "labels": ["label1-for-analytics", "label2-for-analytics"]
    }]
}
```

Update the test biases in your remotely-hosted configuration.

User will be assigned new test groups when the app next checks the configuration file.

Rolling Out

To roll out an A/B test to all users:

```
{
    "features": [{
        "name": "Example A/B Test",
        "enabled": true,
        "test-biases": [50, 50],
        "test-variations": ["Group A", "Group B"],
        "labels": ["label1-for-analytics", "label2-for-analytics"]
    }]
}
```

becomes:

```
{
    "features": [{
        "name": "Example A/B Test",
        "enabled": true
    }]
}
```

API

- Feature flag
  - `Feature.isEnabled(.exampleFeatureFlag))` used to check whether a feature flag is enabled.
- A/B test

```
if let test = ABTest(rawValue: .exampleABTest) {
    print("Is in group A? -> \(test.isGroupA())")
    print("Is in group B? -> \(test.isGroupB())")
}
```

- A/B feature test (all of the following work)

```
if let feature = Feature.named(.exampleFeatureOnOffTest) {
    print("Feature name -> \(feature.name)")
    print("Is enabled? -> \(feature.isTestVariation(.enabled))")
    print("Is disabled -> \(feature.isTestVariation(.disabled))")
    print("Test variation -> \(feature.testVariation())")
}
```

Extensibility

For MVT tests (or A/B tests which require further granularity):

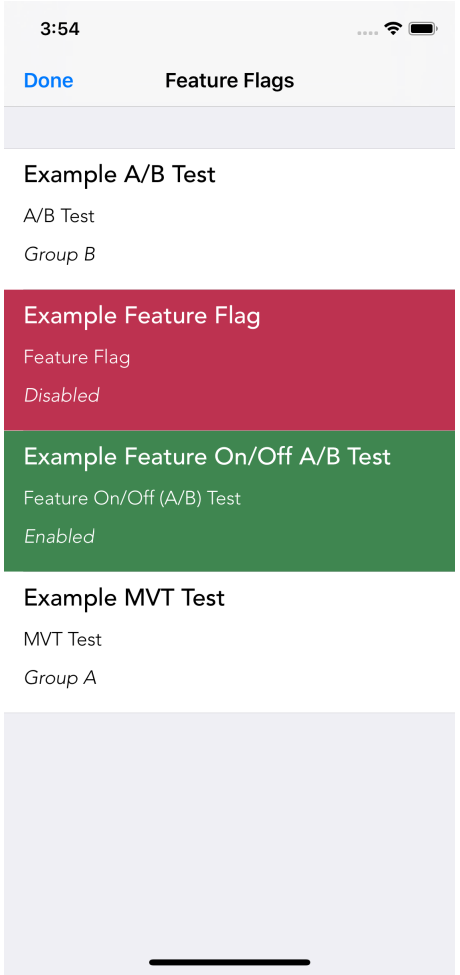
```
extension Feature.Name {
    static let exampleFeatureFlag = Feature.Name(rawValue: "Example Feature Flag")
    static let exampleABTest = Feature.Name(rawValue: "Example A/B Test")
    static let exampleFeatureTest = Feature.Name(rawValue: "Example Feature On/
Off")
    static let exampleMVTTest = Feature.Name(rawValue: "Example MVT Test")
 }

extension Test.Variation {
    static let groupA = Test.Variation(rawValue: "Group A")
    static let groupB = Test.Variation(rawValue: "Group B”)
    static let groupC = Test.Variation(rawValue: "Group C")
}

if let feature = Feature.named(.exampleMVTTest) {
    print("Feature name -> \(feature.name)")
    print("Is group A? -> \(feature.isTestVariation(.groupA))")
    print("Is group B? -> \(feature.isTestVariation(.groupB))")
    print("Is group C? -> \(feature.isTestVariation(.groupC))")
    print("Test variation -> \(feature.testVariation())”)
}
```

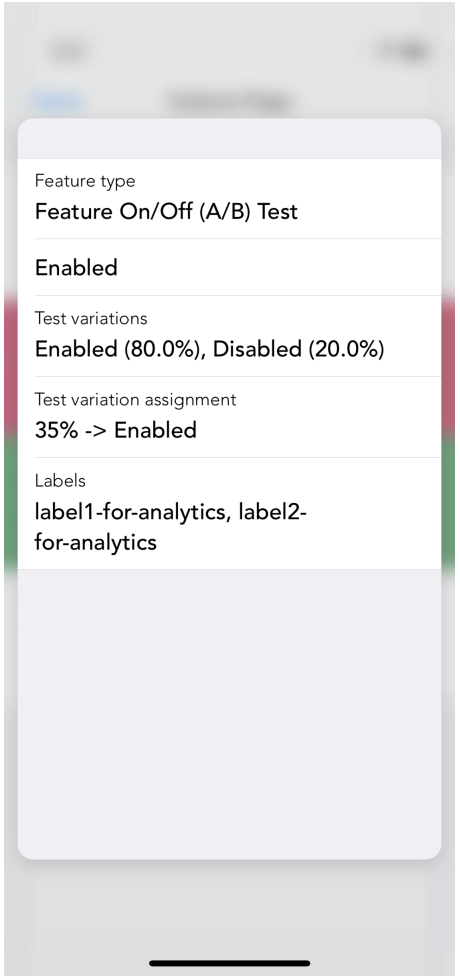
Visualisation

- Tap to switch enable / disable flags
- Tap to toggle A/B or MVT groups
- Swipe to delete cached flags / tests



Visualisation

- 3D Touch to peek more information
- Including analytics labels



API

- `FeatureFlags.configurationURL` used to set URL to remote or bundled configuration JSON .
- `FeatureFlags.localFallbackConfigurationURL` used to set a locally-bundled fallback URL where using a remote configuration.
- `FeatureFlags.refresh()` which accepts an optional completion handler triggers a configuration fetch.
- `FeatureFlags.refreshWithData(_:completion:)` useful where configuration file has already been fetched.

FeatureFlags

https://github.com/rwbutler/FeatureFlags

- Cocoapods

```
pod “FeatureFlags”
```

- Carthage

```
github “rwbutler/FeatureFlags”
```