Best practices running an iOS **open source** project on GitHub

Oliver Drobnik

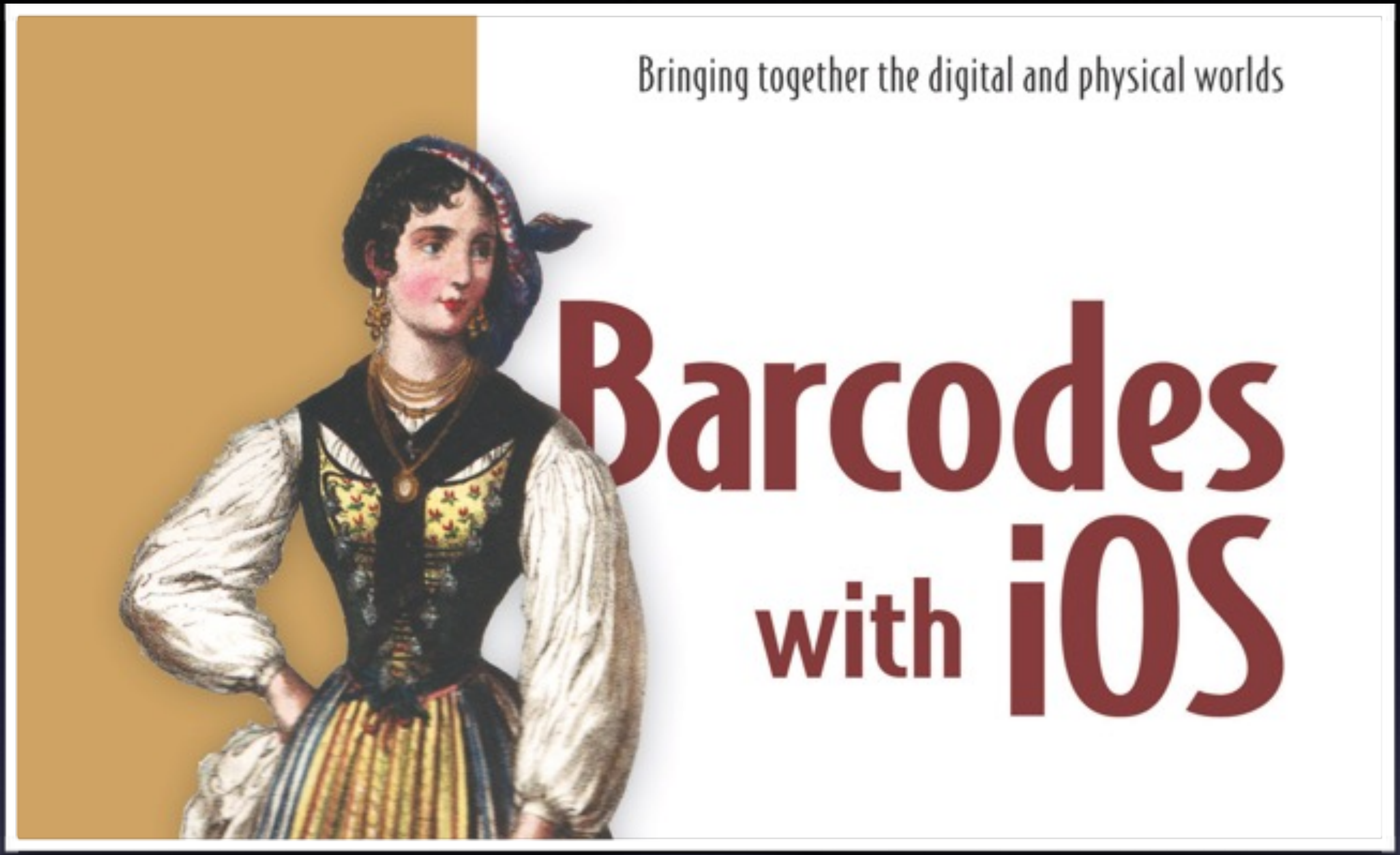
iOS Development & Consulting

@cocoanetics

cocoanetics.com

Full-time iOS developer and blogger since January 2010

	Stars	Forks
DTCoreText	2567	589
DTFoundation	462	120
DTBonjour	215	21
DTMarkdownParser	125	8
DTLocalizableStringScanner	96	19
AutoIngest	89	7
DTWebArchive	80	13
DTITCReportDownloader	78	2
DTDownload	60	8



44% off all books manning.com through Dec 3rd
with promo code **mobicftw**

Tweet about my talk with **#barcodes_iOS**
for chance to win one free copy

Open Source Wisdom

"Whatever is worth doing at all, is worth doing well."

–Philip Stanhope, 4th Earl of Chesterfield, in a letter to his son 1746

Worth doing?

- You develop an **abundance mentality**
- Others can **add functionality** outside of my own needs if they fit with project philosophy
- Others can point out **false assumptions**, you learn
- Others can **fix bugs** only apparent in their apps
- Others can **pay** for using it without attribution
- Others can see the **quality** of your work

Worth doing well!

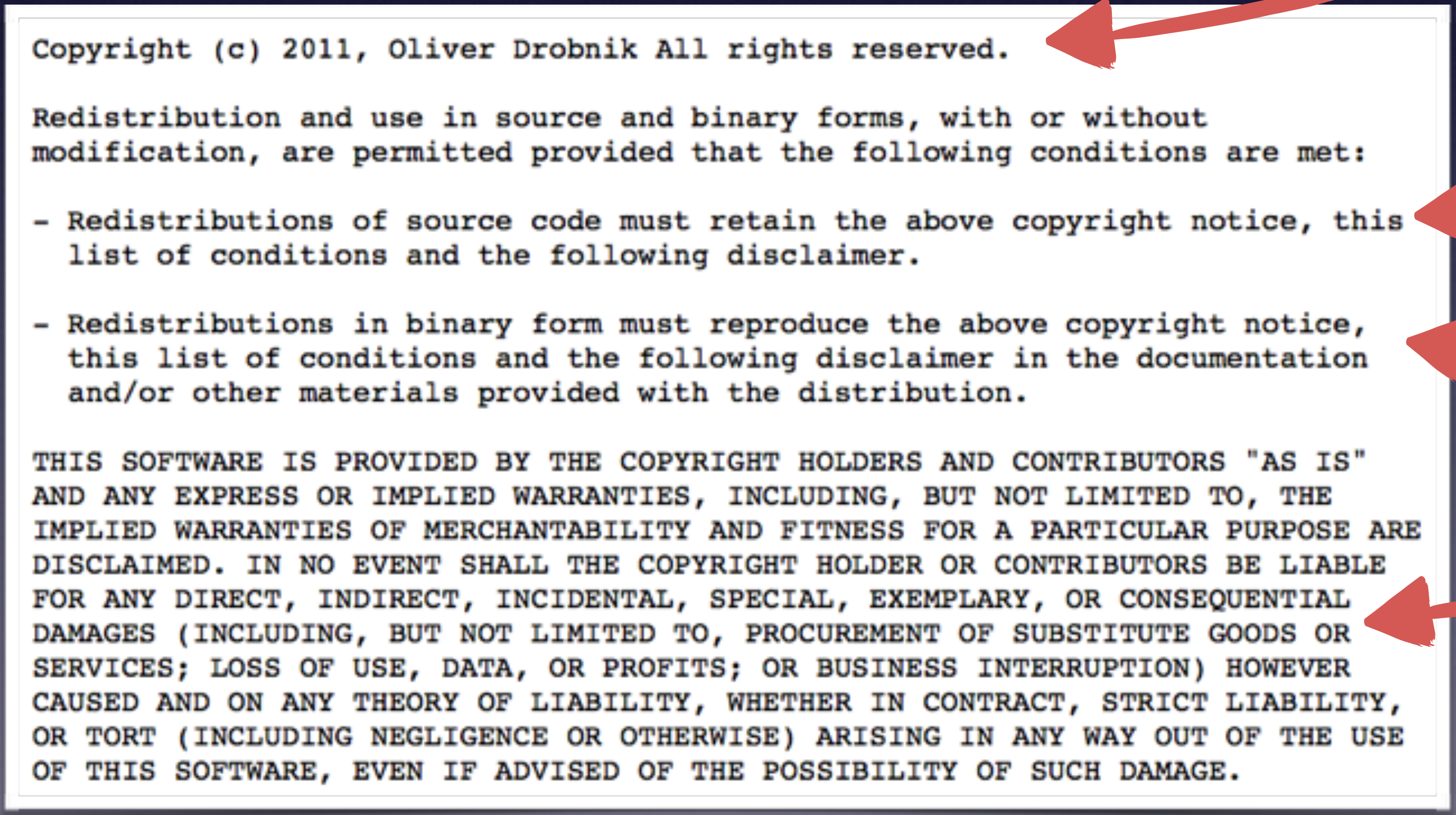
- A well-organized folder structure
- Integration options
- Documentation
- GIT Branching + Pull Requests
- Continuous Integration + Code Coverage
- Issues, Milestones, Beautiful Releases

WARNING:
PERSONAL
OPINIONS

A Well-Organized Folder Structure

Project Root

- READ ME: Project overview, how to use
- LICENSE: Conditions for redistribution



```
Copyright (c) 2011, Oliver Drobnik All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

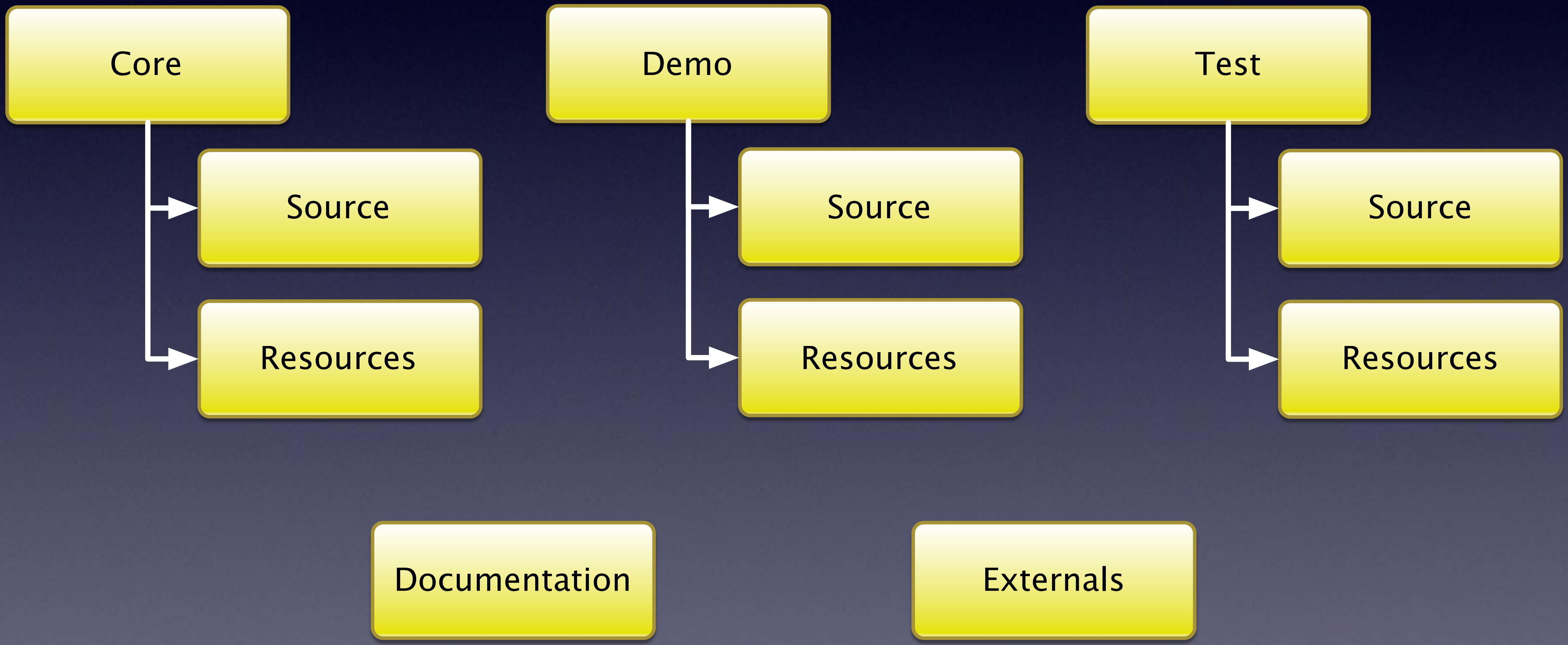
copyright

for source

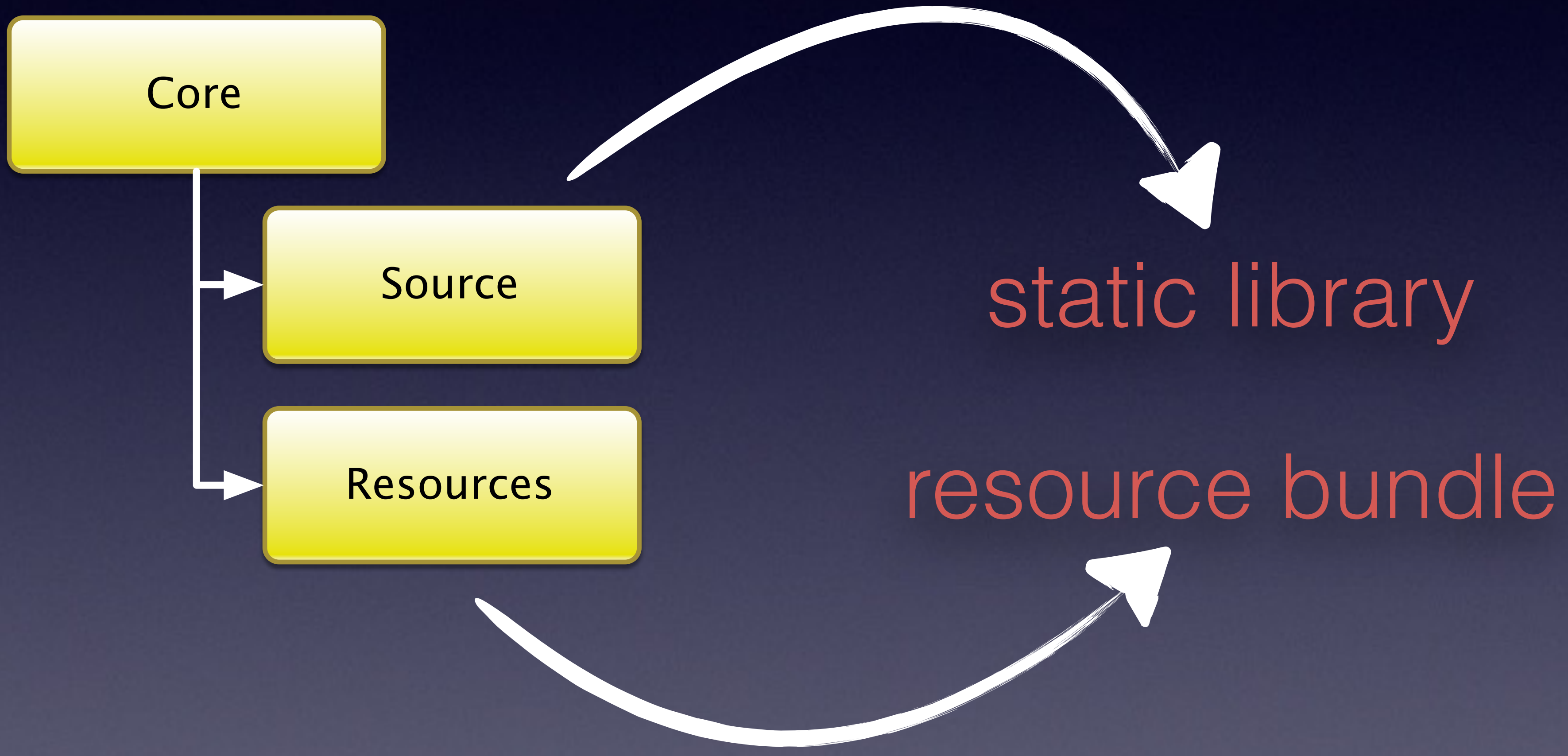
for apps

no warranty

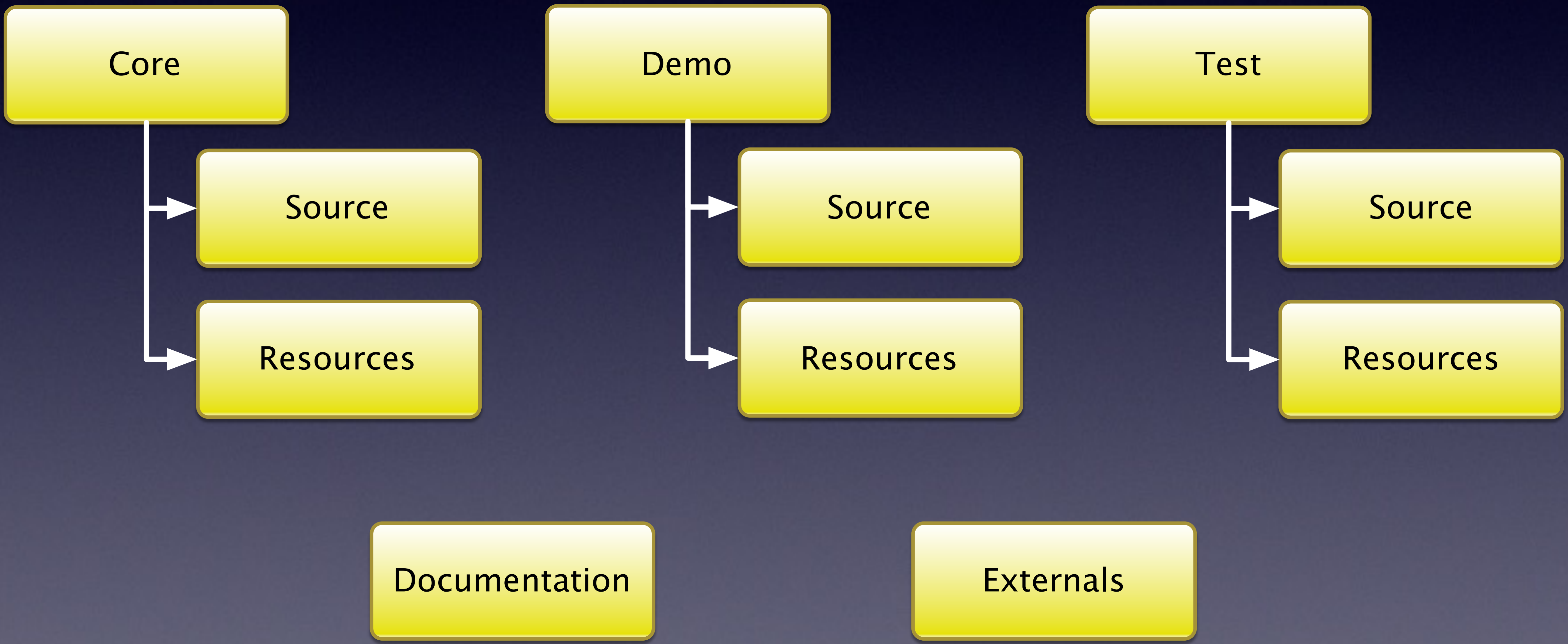
Folder Structure



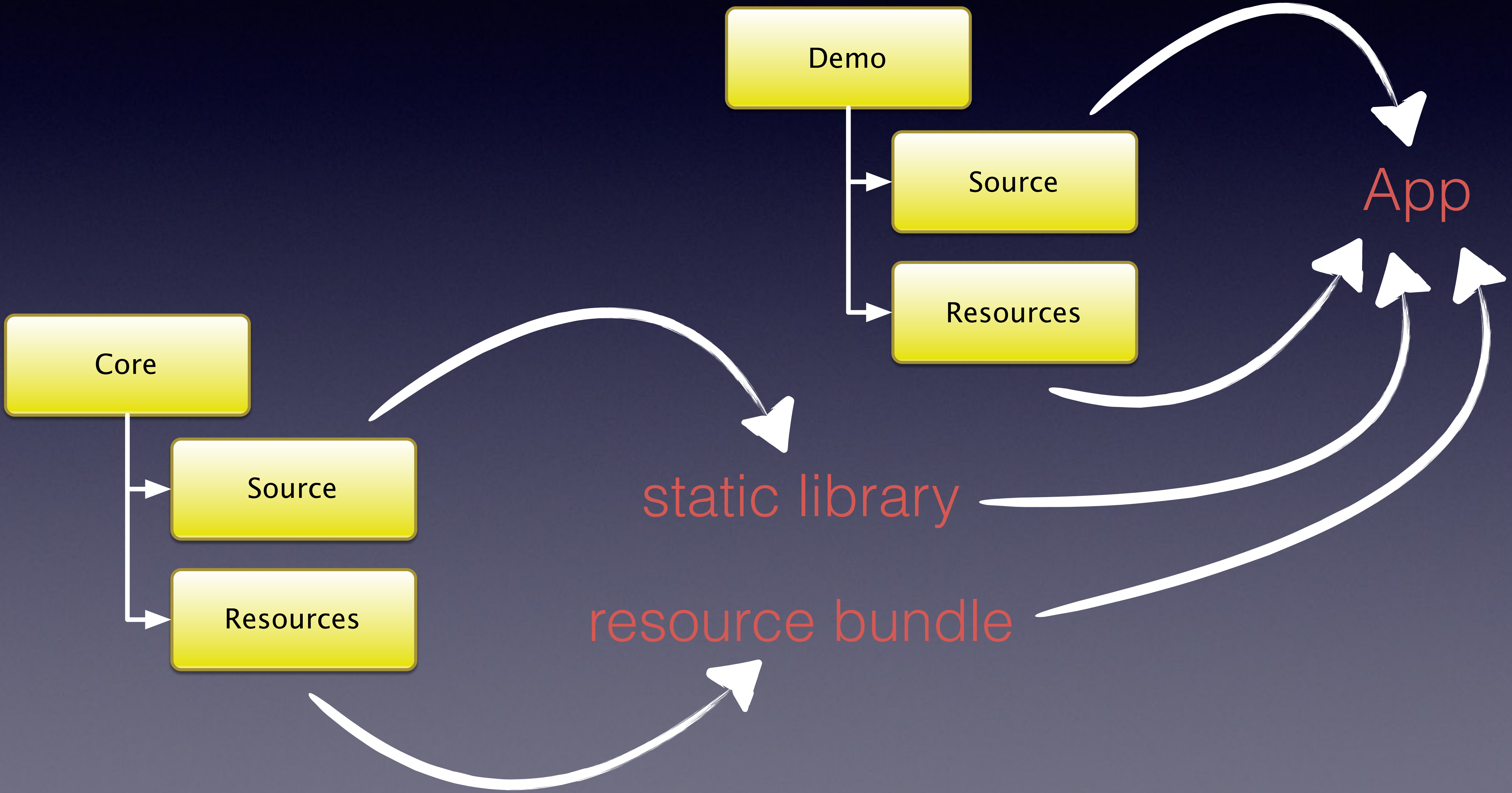
Main Source



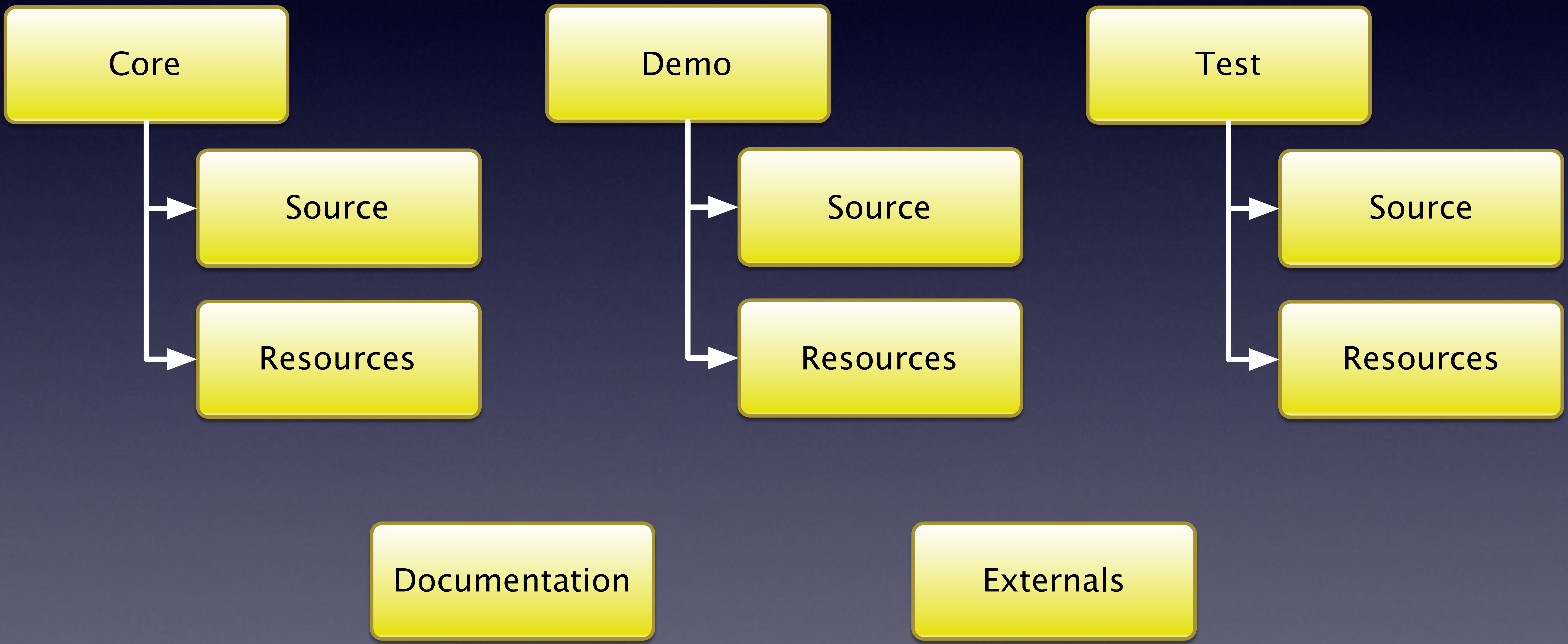
Folder Structure



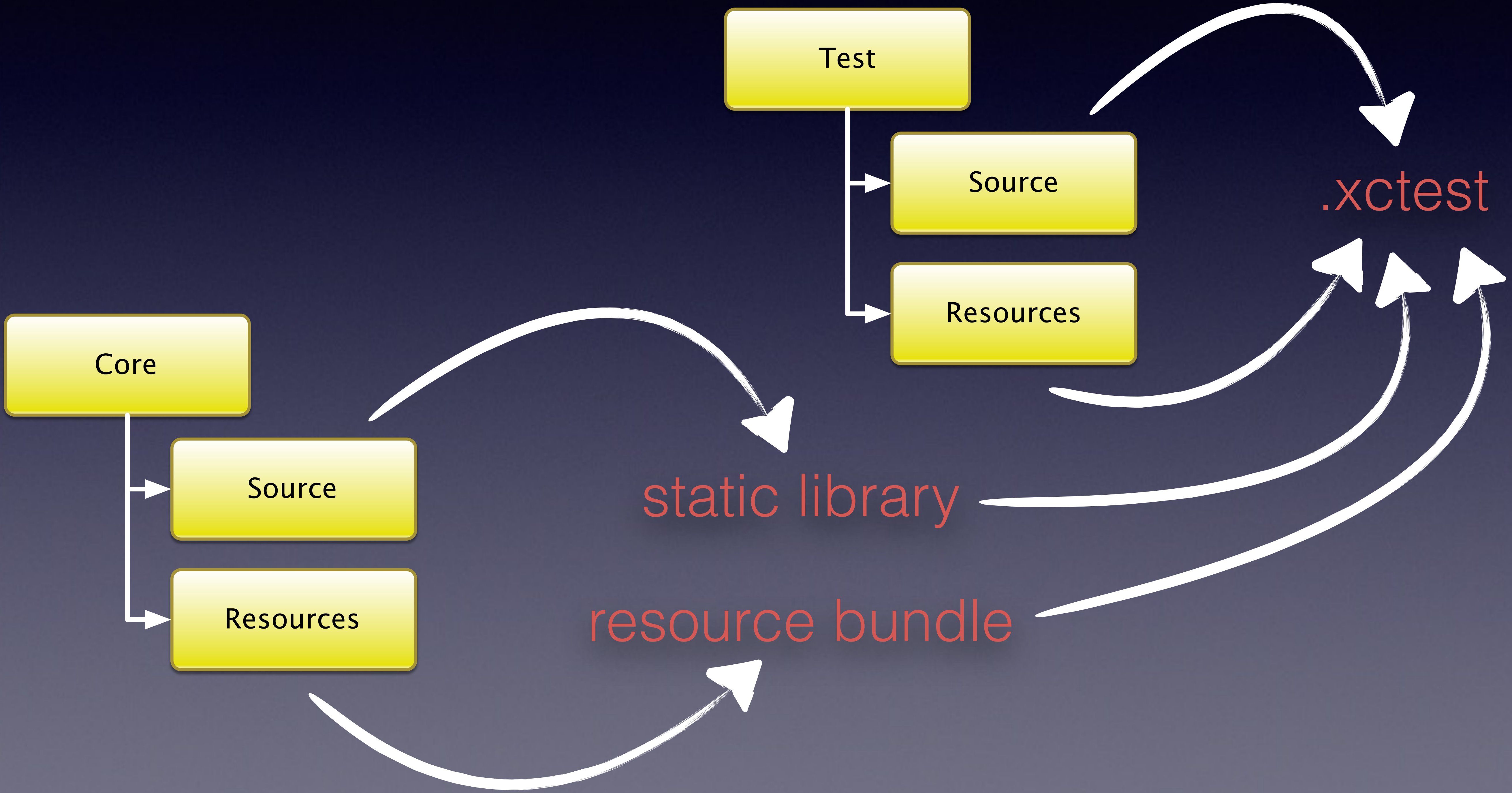
Demo App


Demo
Source
Resources
App
Core
Source
Resources
static library
resource bundle

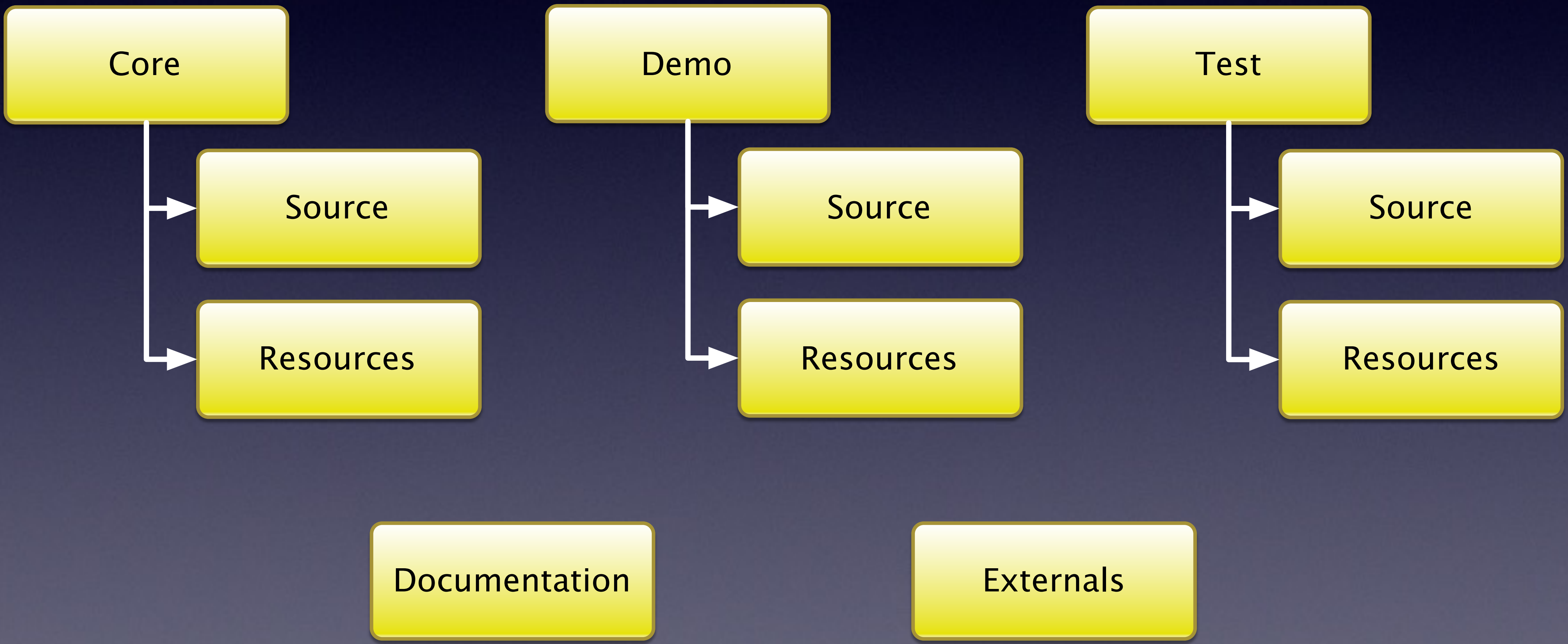
Folder Structure



Unit Tests


Test
Source
Resources
.xctest
Core
Source
Resources
static library
resource bundle

Folder Structure



Integration Options

Integration via GIT Submodule

1. Add GIT submodule in Externals of app
2. Add Xcode reference to module’s xcodeproj
3. Set user header search path
4. Link static library target from sub-project
5. Don’t forget -ObjC linker flag!
6. (If applicable) copy resource bundle

Integration via Cocoapods

1. Add 1 line to Podfile:

   ```
   pod 'DTFoundation/DTSidePanel', '~> 1.6.0’
   ```

2. First `pod install` creates Podfile.lock
3. Subsequent `pod install` uses locked versions
4. Any `pod update` updates Podfile.lock

Creating a Pod Spec

1. Core spec references

```
spec.source_files = 'Core/Source/*.{h,m}'
```

2. Sub-spec references

```
ss.ios.source_files = 'Core/Source/iOS/*.{h,m}'
```

3. Resource bundle reference

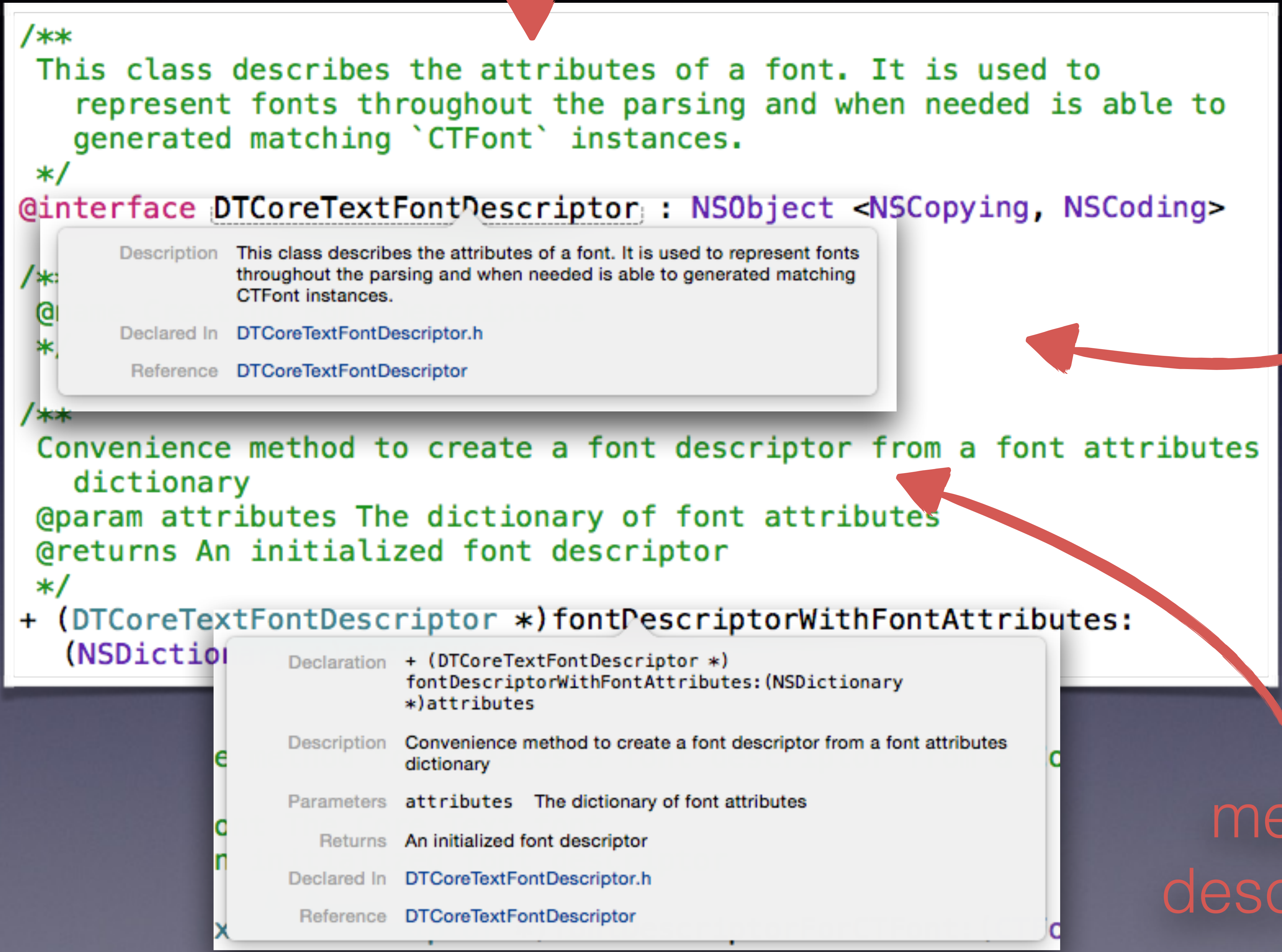
```
spec.resource_bundles =
{ 'DTLoupe' => ['Core/Resources/*.png'] }
```

4. Submit spec to CocoaPods trunk

Documentation

Why, oh Why?

- Document what you thought when creating these classes
- Clarify why purpose of public methods
- Documentation popup: Option+click on method name
- Autocompletion shows parameters and description
- Generate documentation to be installed in Xcode
- Generate HTML documentation to be put on your website
- Force yourself to have lean public interfaces



```
/**
 This class describes the attributes of a font. It is used to
   represent fonts throughout the parsing and when needed is able to
   generated matching `CTFont` instances.
 */
@interface DTCoreTextFontDescriptor : NSObject <NSCopying, NSCoding>
```

Description	This class describes the attributes of a font. It is used to represent fonts throughout the parsing and when needed is able to generated matching CTFont instances.
Declared In	DTCoreTextFontDescriptor.h
Reference	DTCoreTextFontDescriptor

```
/**
 Convenience method to create a font descriptor from a font attributes
   dictionary
 @param attributes The dictionary of font attributes
 @returns An initialized font descriptor
 */
+ (DTCoreTextFontDescriptor *)fontDescriptorWithFontAttributes:
   (NSDictio
```

Declaration	+ (DTCoreTextFontDescriptor *) fontDescriptorWithFontAttributes:(NSDictionary *)attributes
Description	Convenience method to create a font descriptor from a font attributes dictionary
Parameters	attributes The dictionary of font attributes
Returns	An initialized font descriptor
Declared In	DTCoreTextFontDescriptor.h
Reference	DTCoreTextFontDescriptor

appledoc by Tomaz Kragelj

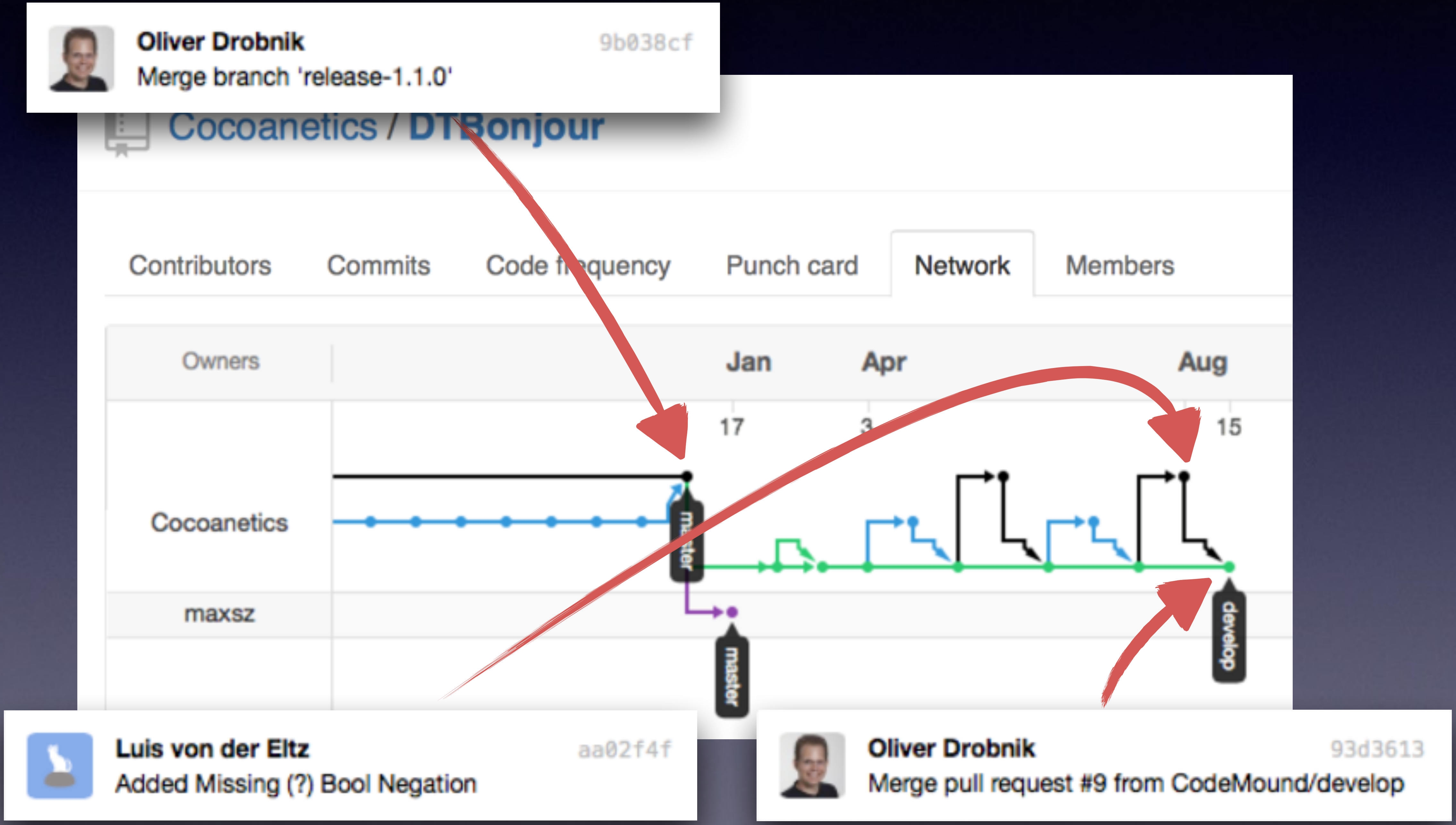
- Doxygen and appledoc mostly compatible
- Scans specified files for doc comments
- AppledocSettings.plist in project root
- `appledoc -o /tmp .`
- Enable “Documentation Comments” warning!

GIT Branching
+ Pull Requests

A Successful Git Branching Model

- Google “successful git branching” for blog post
- **develop** branch for development
- feature branches for multiple-commit work
- releases via **release-1.0** branch into **master**
- Tip: Set **develop** as default branch on GitHub

Regular Development



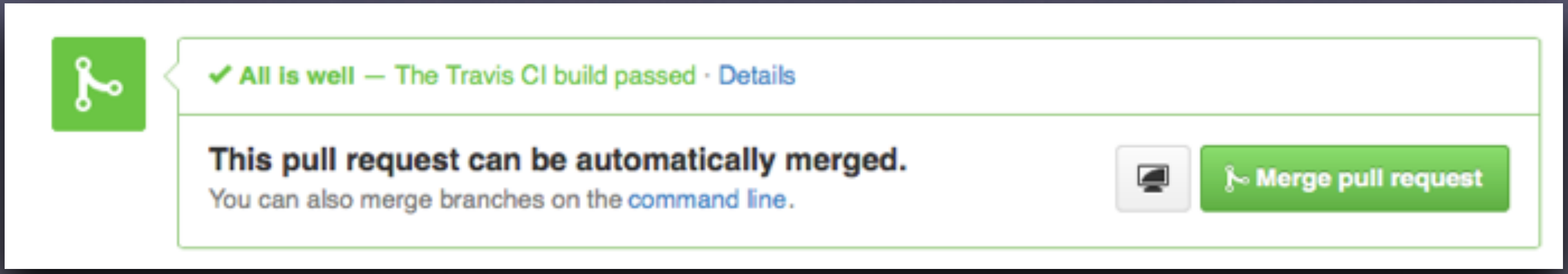
Oliver Drobnik 9b038cf
Merge branch 'release-1.1.0'

Luis von der Eltz aa02f4f
Added Missing (?) Bool Negation

Oliver Drobnik 93d3613
Merge pull request #9 from CodeMound/develop

Pull Request Quality

- Travis-CI is free for Open Source
- Continuous Integration
- A fresh virtual build machine every time
- Builds on every push to any branch
- Builds pull requests and comments

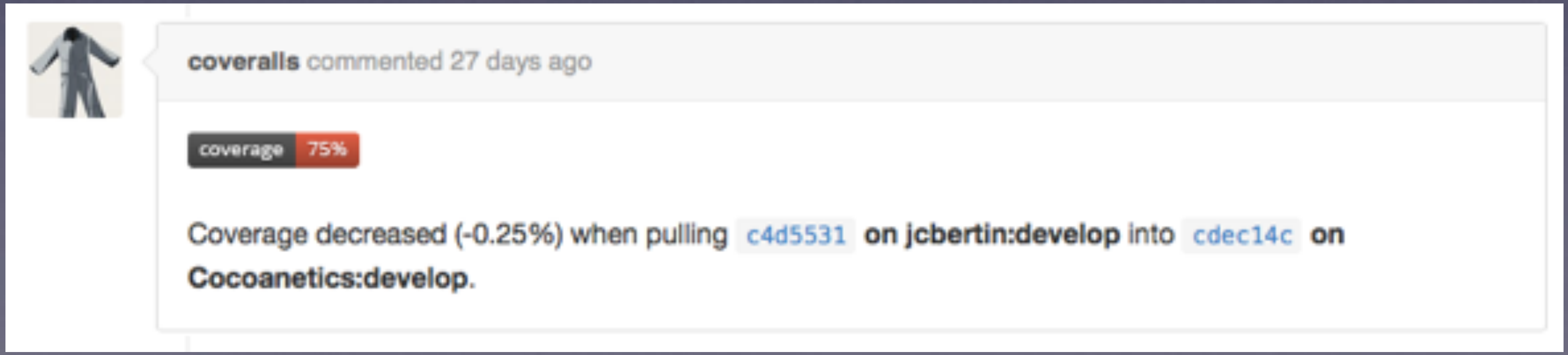


✔ All is well — The Travis CI build passed · Details

This pull request can be automatically merged.
You can also merge branches on the command line.

Pull Request Quality

- Coveralls.io is free for Open Source
- Track changes in coverage
- Only useful if you have unit tests
- Coverage info is pushed by Travis-CI
- Adds comment to GitHub pull request


coveralls commented 27 days ago

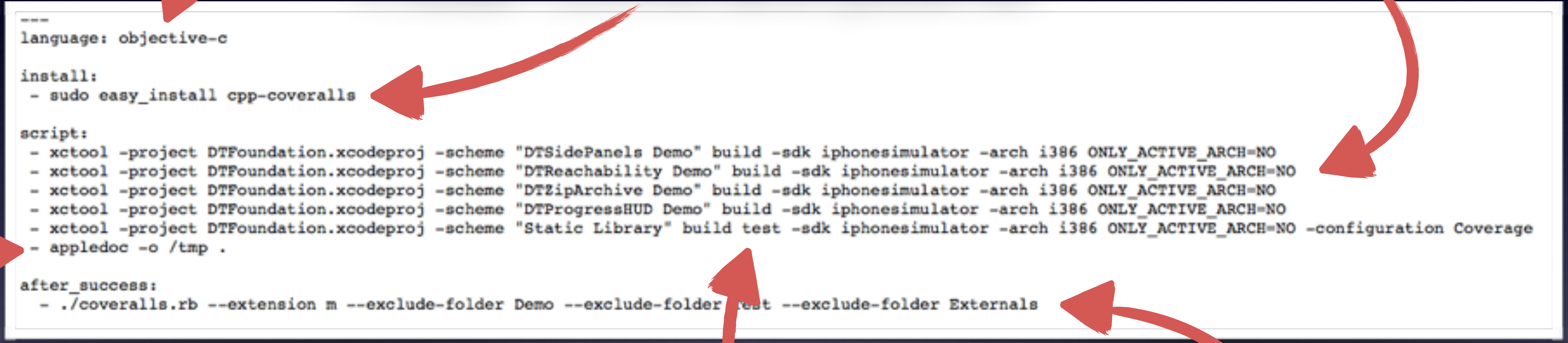
Coverage decreased (-0.25%) when pulling c4d5531 on jcbertin:develop into cdec14c on Cocoanetics:develop.

Travis-CI Build File

Obj-C Project

Build demo apps

Install Coveralls pusher



```
---
language: objective-c

install:
 - sudo easy_install cpp-coveralls

script:
 - xctool -project DTFoundation.xcodeproj -scheme "DTSidePanels Demo" build -sdk iphonesimulator -arch i386 ONLY_ACTIVE_ARCH=NO
 - xctool -project DTFoundation.xcodeproj -scheme "DTReachability Demo" build -sdk iphonesimulator -arch i386 ONLY_ACTIVE_ARCH=NO
 - xctool -project DTFoundation.xcodeproj -scheme "DTZipArchive Demo" build -sdk iphonesimulator -arch i386 ONLY_ACTIVE_ARCH=NO
 - xctool -project DTFoundation.xcodeproj -scheme "DTProgressHUD Demo" build -sdk iphonesimulator -arch i386 ONLY_ACTIVE_ARCH=NO
 - xctool -project DTFoundation.xcodeproj -scheme "Static Library" build test -sdk iphonesimulator -arch i386 ONLY_ACTIVE_ARCH=NO -configuration Coverage
 - appledoc -o /tmp .

after_success:
  - ./coveralls.rb --extension m --exclude-folder Demo --exclude-folder Test --exclude-folder Externals
```

1. Build Static Library
2. Perform unit tests
3. Collect code coverage data

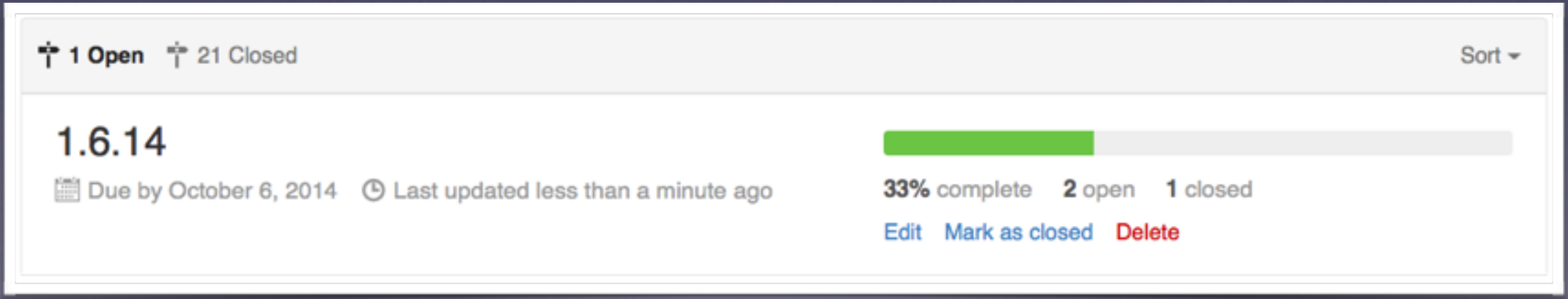
“Docs Unit Test”

Push coverage files

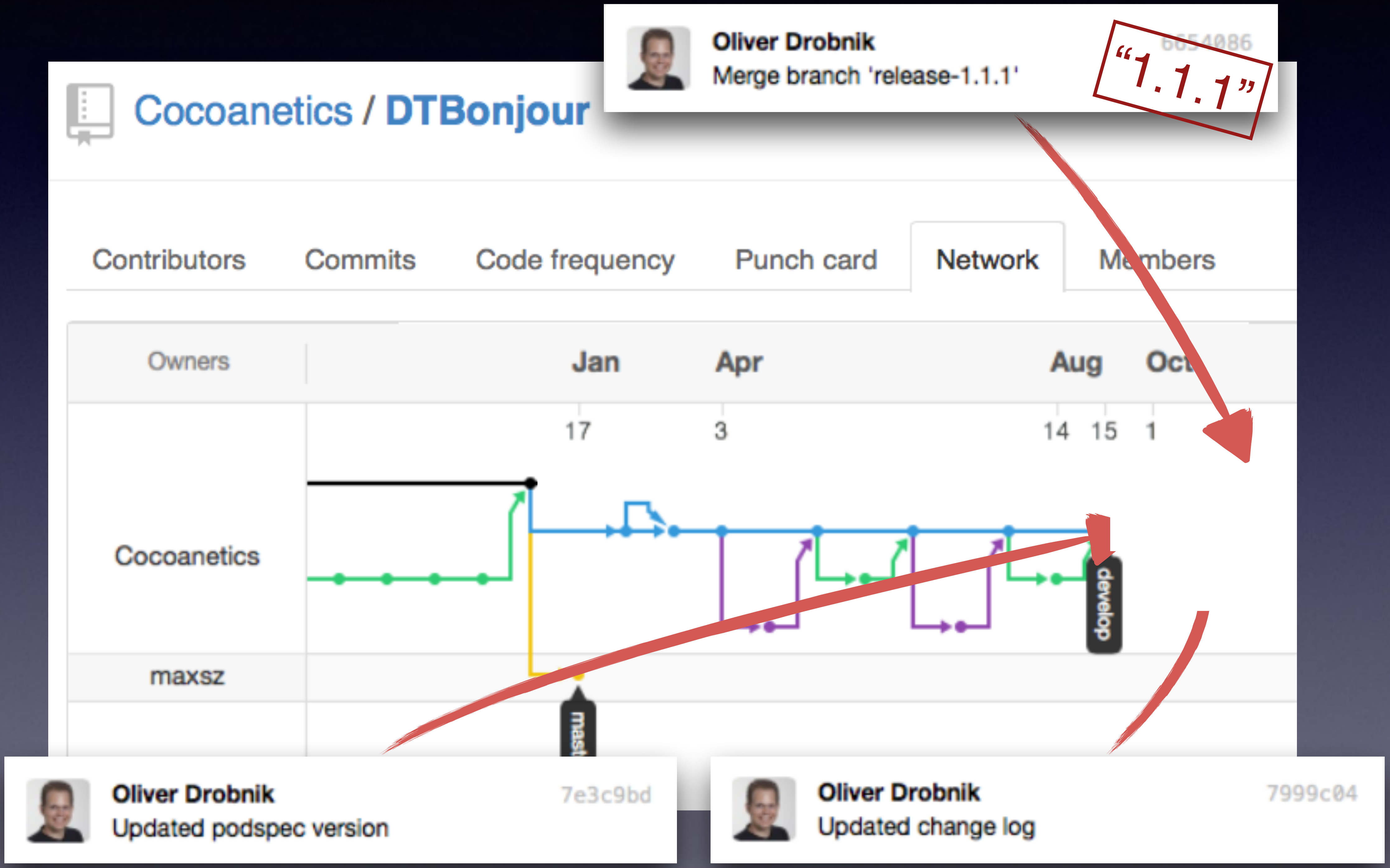
Issues, Milestones, Beautiful Releases

Tracking Issues

- Any problem: create an issue for it
- Use tags to categorise issues, assign to developer
- Pull Requests are issues, too
- Reference issue by # number in commits
- Group issues by release with milestones



Merging the Release Branch



Oliver Drobnik
Merge branch 'release-1.1.1'
6654086

Oliver Drobnik
Updated podspec version
7e3c9bd

Oliver Drobnik
Updated change log
7999c04

Chores for Releases

- Do release branching stuff
  - Update version number in pod spec
  - Update change log
- Copy change log to GitHub release page for tag
- Announce the release on blog
- Merge **master** into **develop** and continue work

Summary

- Enforce organized folder structure
- Provide multiple integration options
- Document public headers
- Use successful GIT branching
- Let CI and Code Coverage track quality
- Track issues and make beautiful releases

Thanks for watching!

@cocoanetics

@productlayer