facebook

facebook

Functional Programming in Facebook for iOS

Adam Ernst
Facebook New York

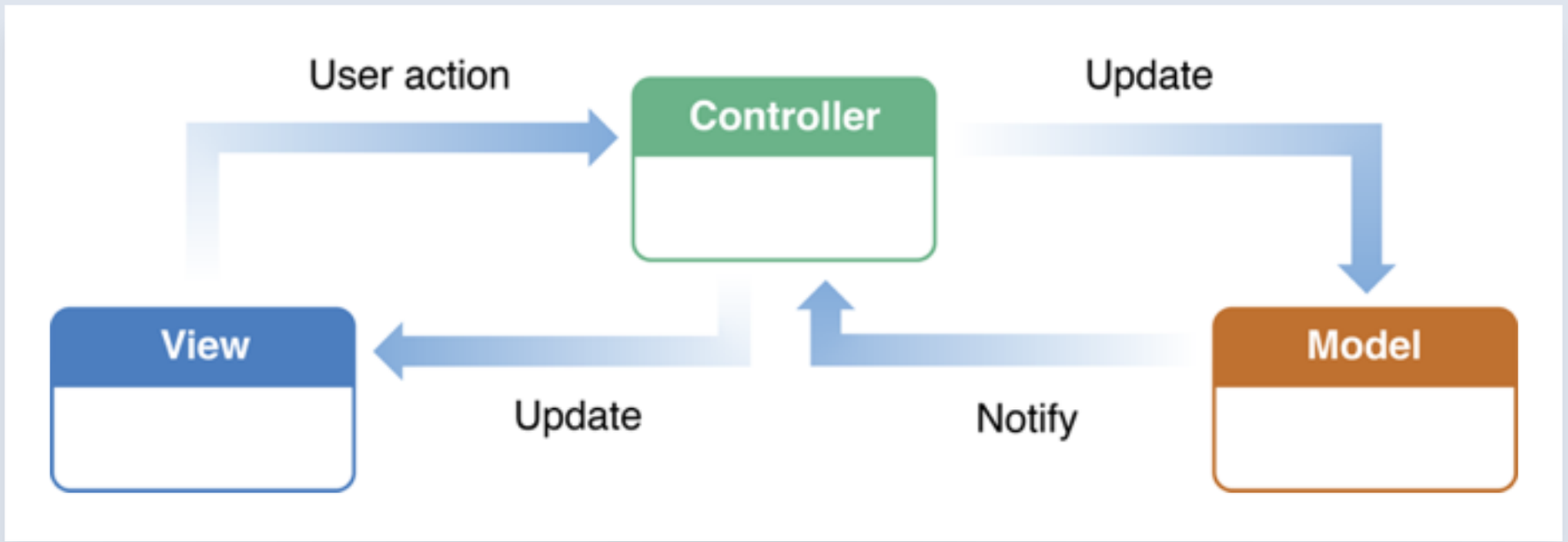


Image Credit: Apple

Feed View

Feed Controller

Story View

Story Controller

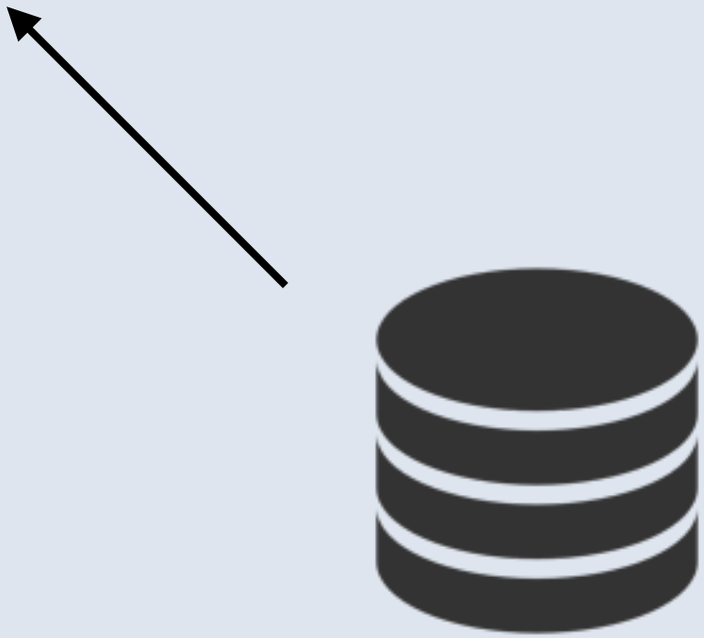
Model

Like Button

Like Controller

Feed View

Feed Controller



Model

Story View

Story Controller

Like Button

Like Controller

Feed View

Feed Controller

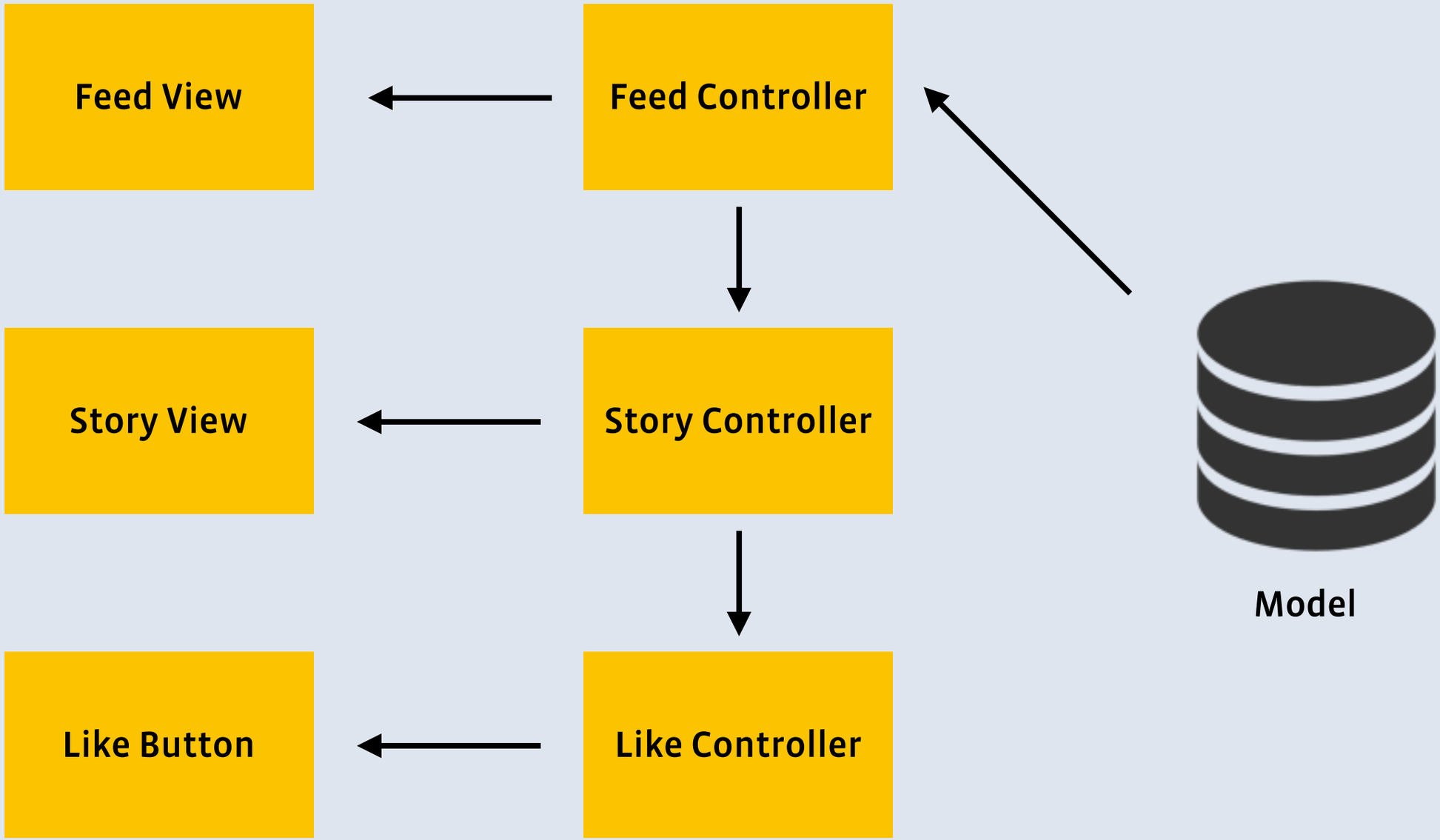
Story View

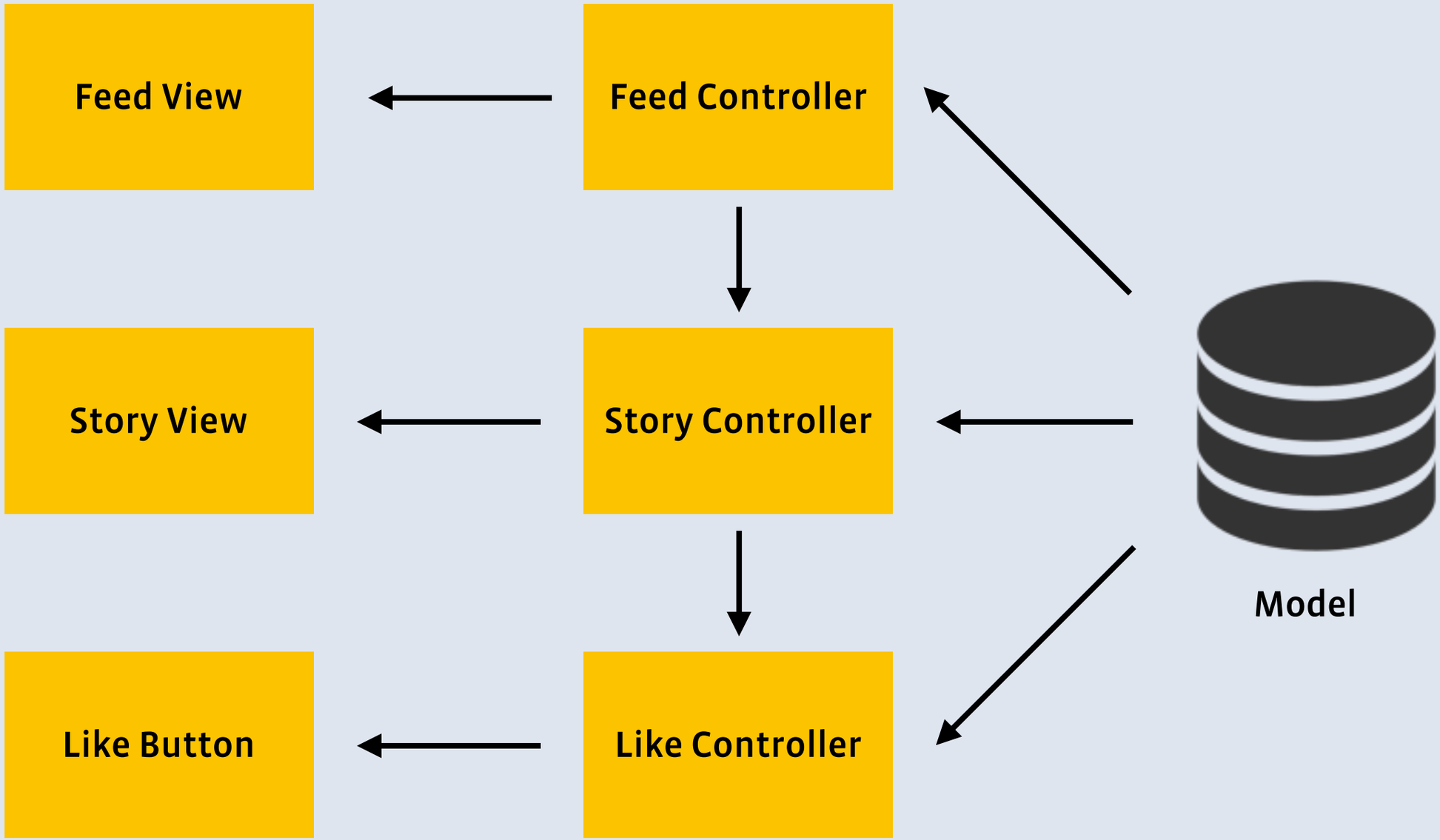
Story Controller

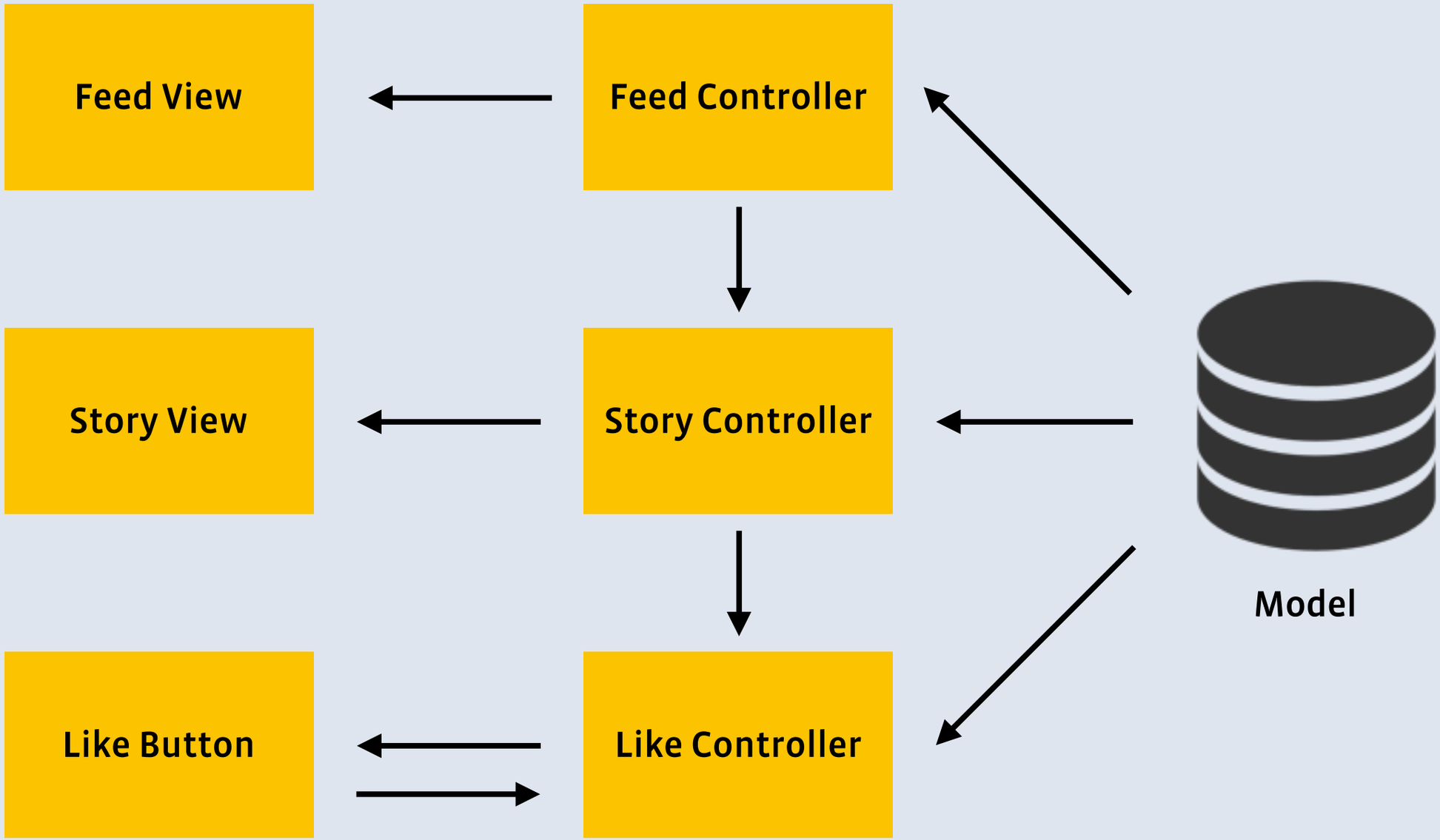
Like Button

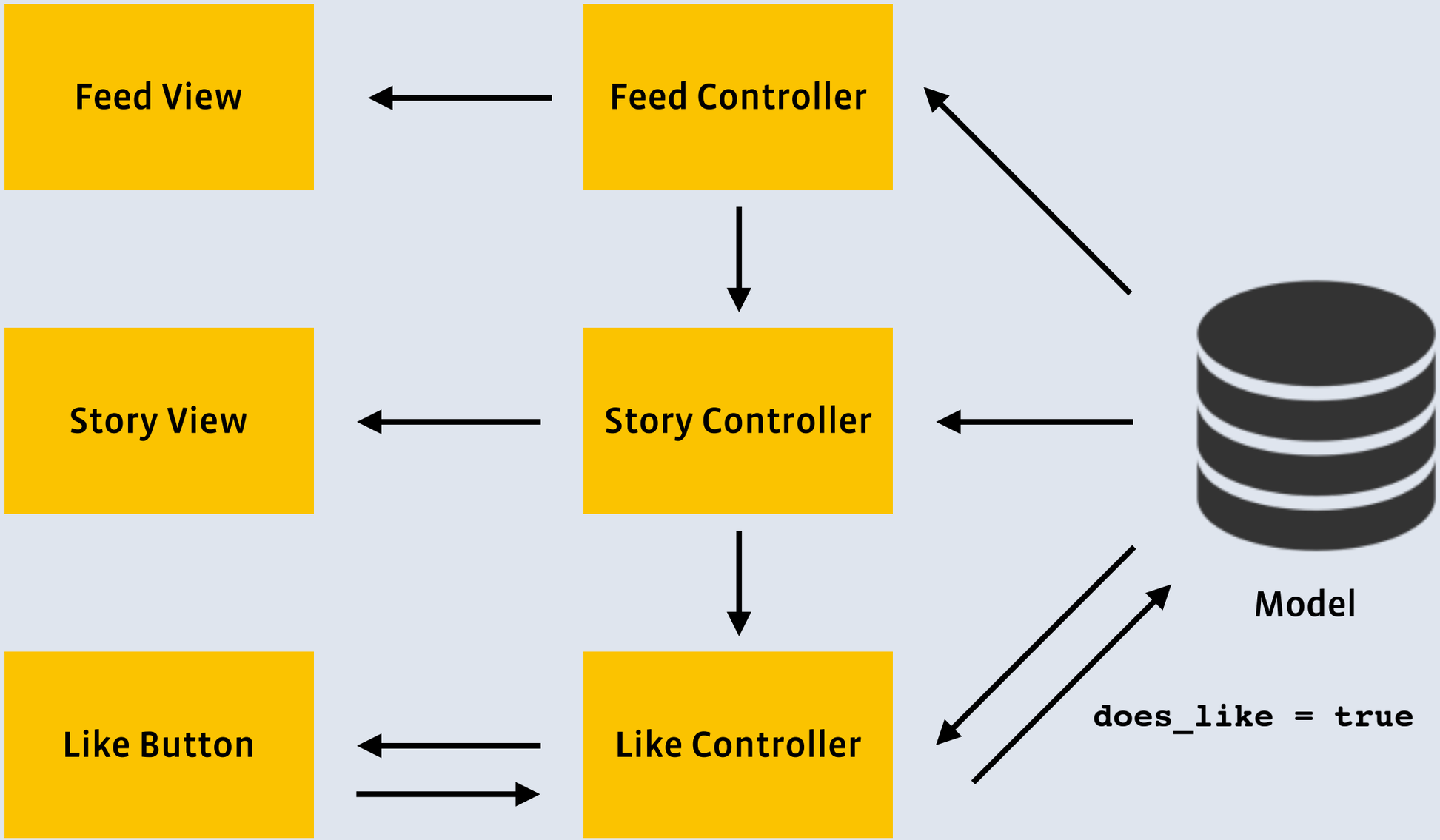
Like Controller

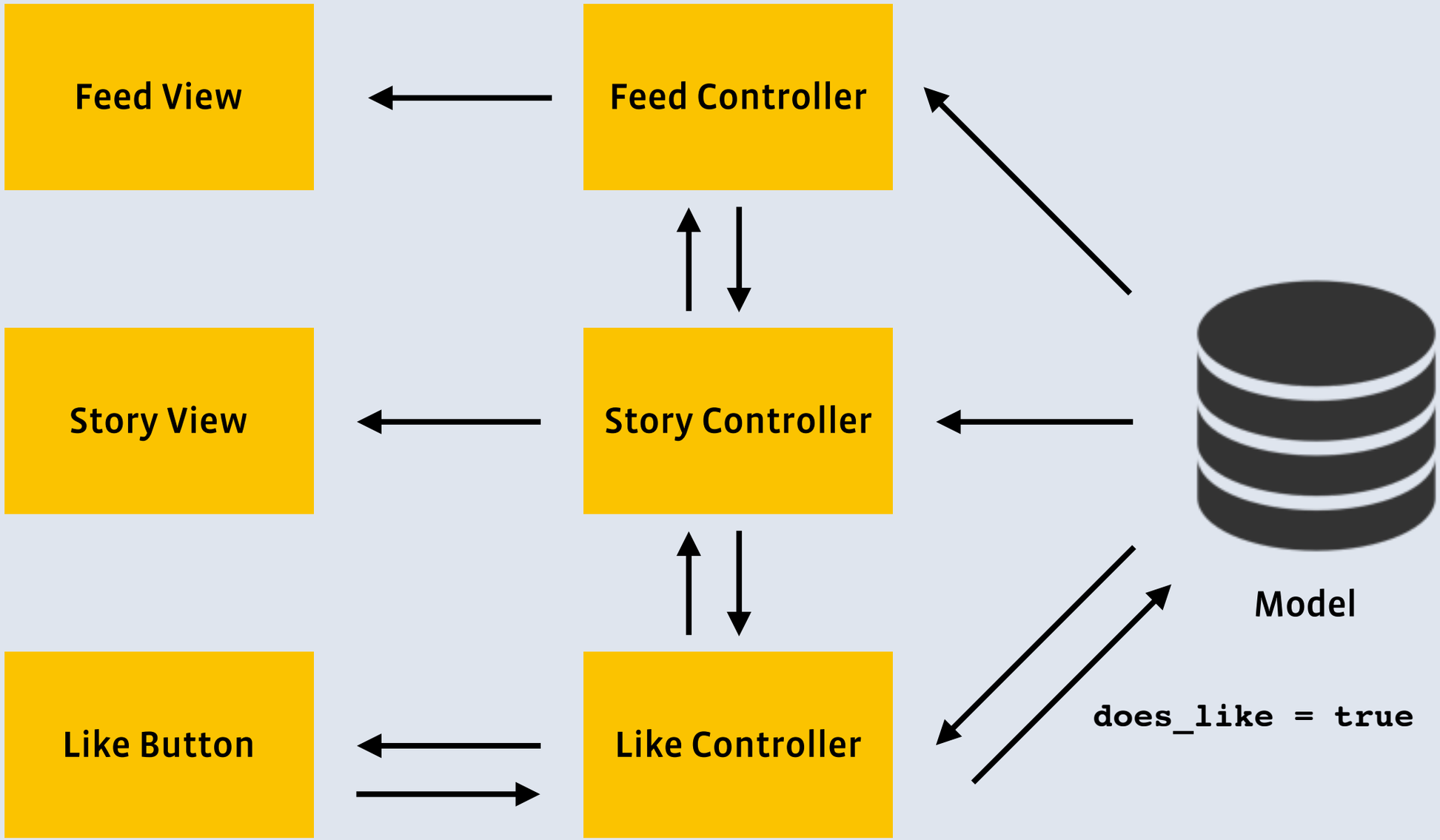
Model

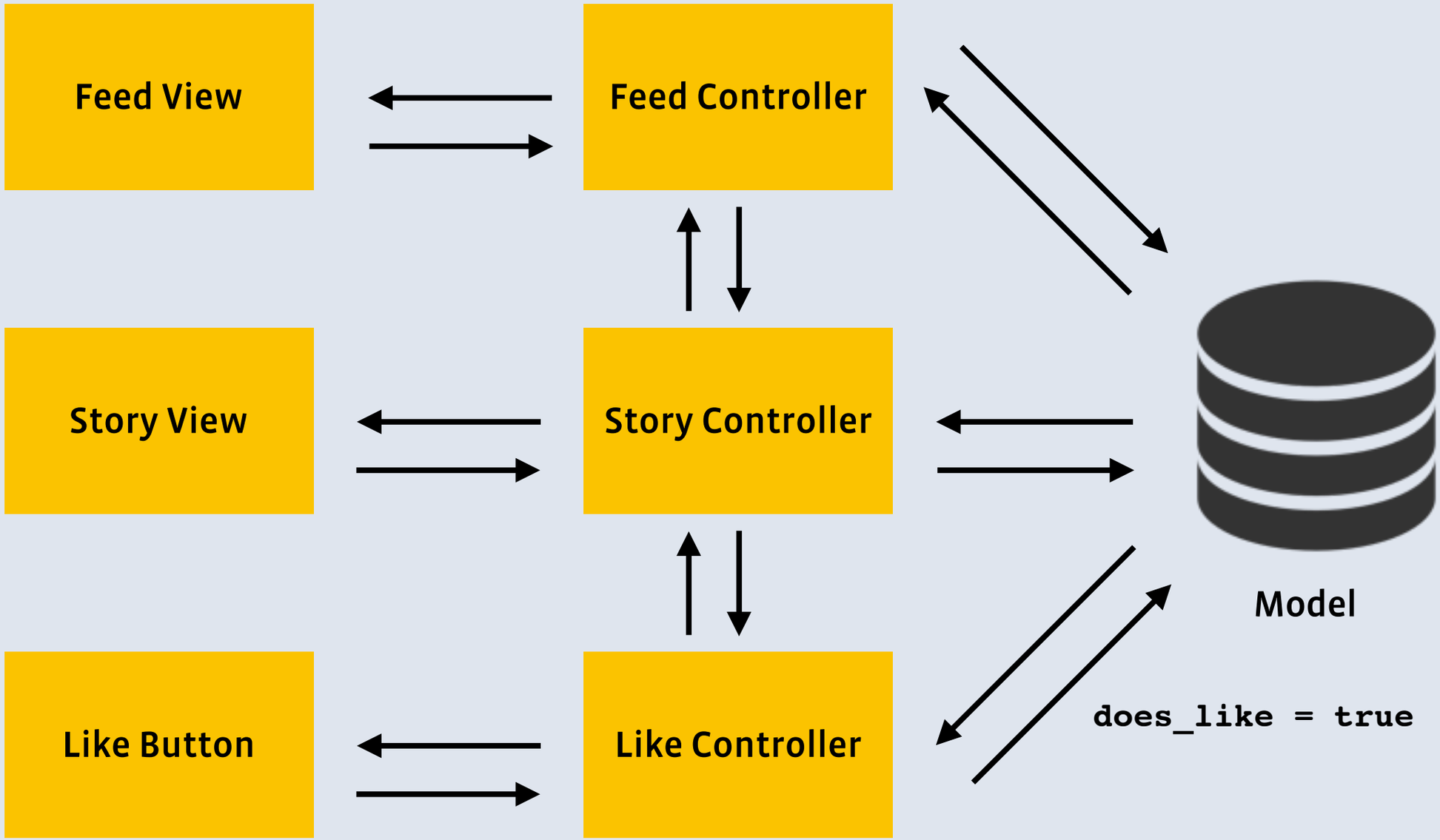

Feed View
Feed Controller
Story View
Story Controller
Like Button
Like Controller
Model


Feed View
Feed Controller
Story View
Story Controller
Like Button
Like Controller
Model


Feed View
Feed Controller
Story View
Story Controller
Model
Like Button
Like Controller


Feed View
Feed Controller
Story View
Story Controller
Model
Like Button
Like Controller
does_like = true


Feed View
Feed Controller
Story View
Story Controller
Model
Like Button
Like Controller
does_like = true


Feed View
Feed Controller
Story View
Story Controller
Model
Like Button
Like Controller
does_like = true

Mutable State and Multithreading

Mutable State and Multithreading

- All main thread ➠ sluggish performance

Mutable State and Multithreading

- All main thread ➠ sluggish performance
- Make everything `atomic` ➠ logic races everywhere

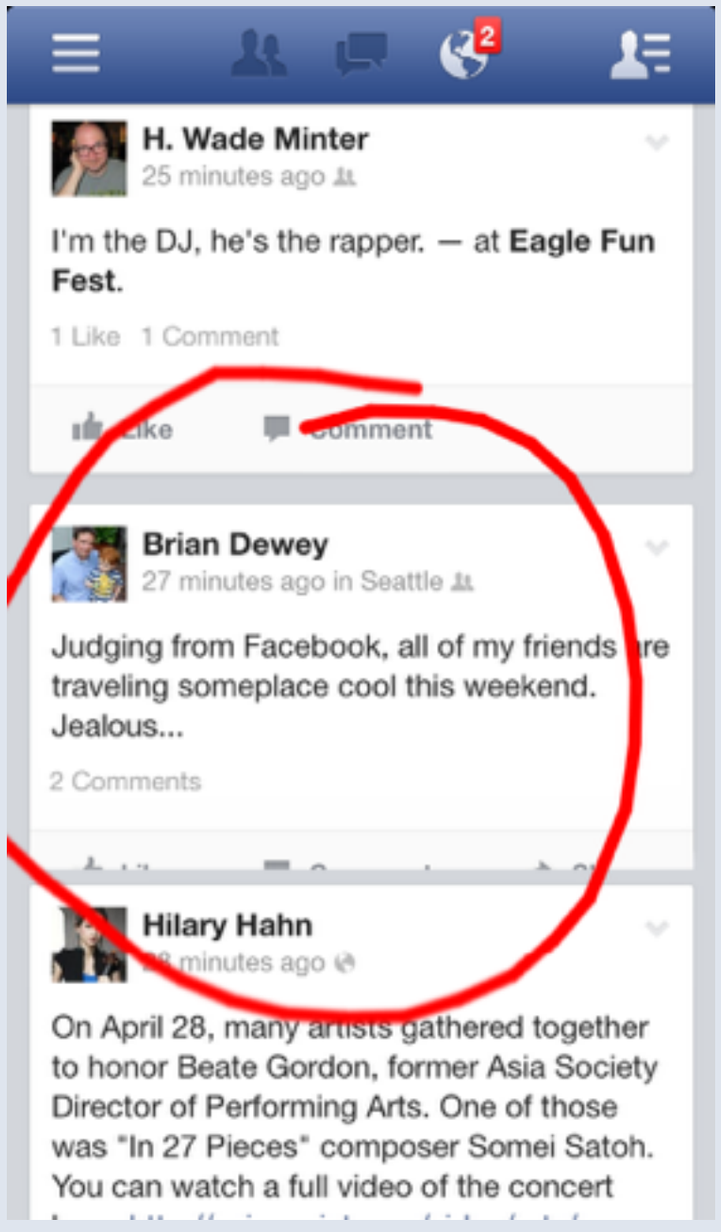
Mutable State and Multithreading

- All main thread ➠ sluggish performance
- Make everything `atomic` ➠ logic races everywhere
- Share locks for related properties ➠ complex, deadlock risk

Mutable State and Multithreading

- All main thread ➠ sluggish performance
- Make everything `atomic` ➠ logic races everywhere
- Share locks for related properties ➠ complex, deadlock risk

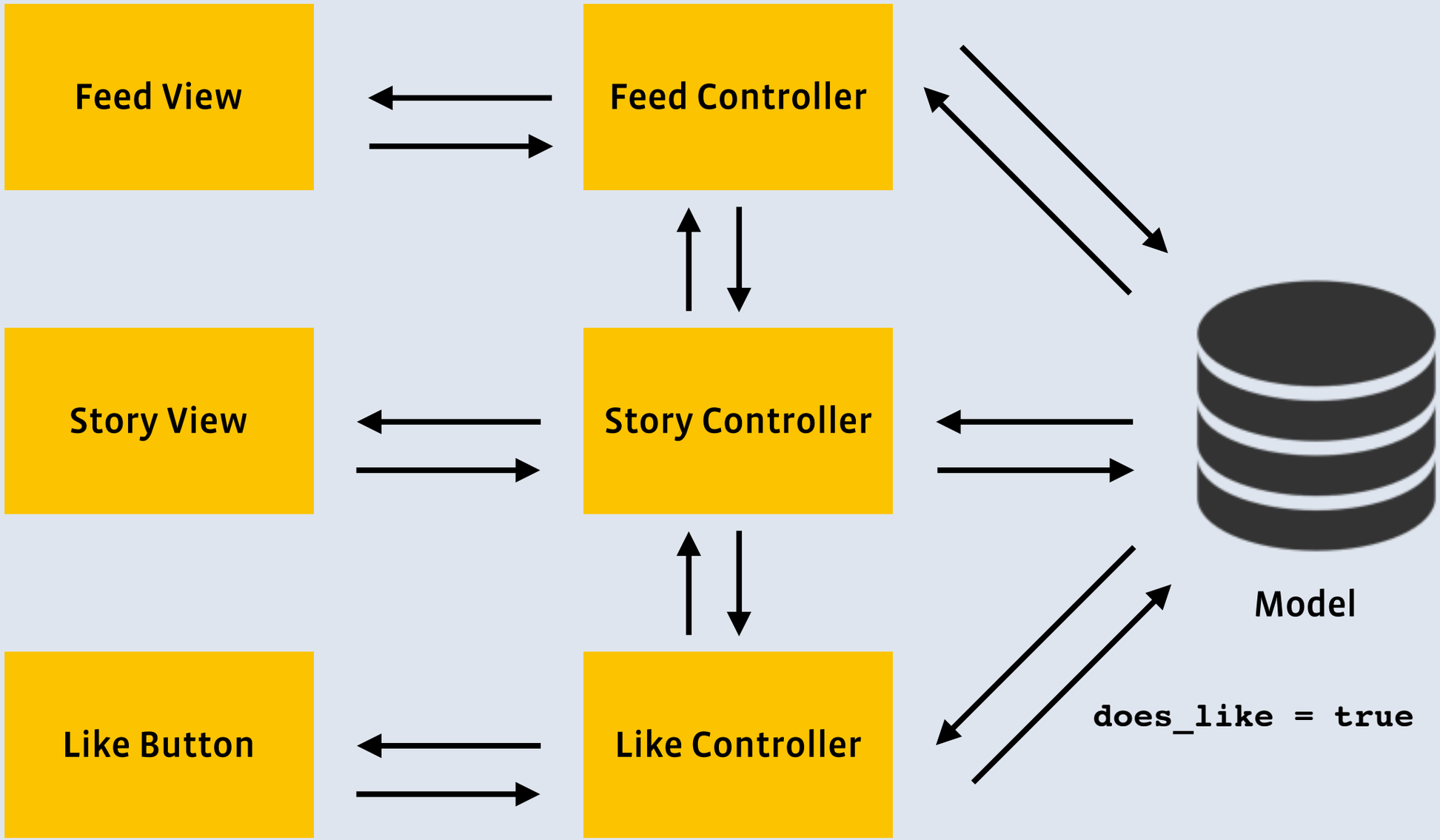
It is extremely difficult to reason about race conditions in multithreaded code with shared, mutable state.

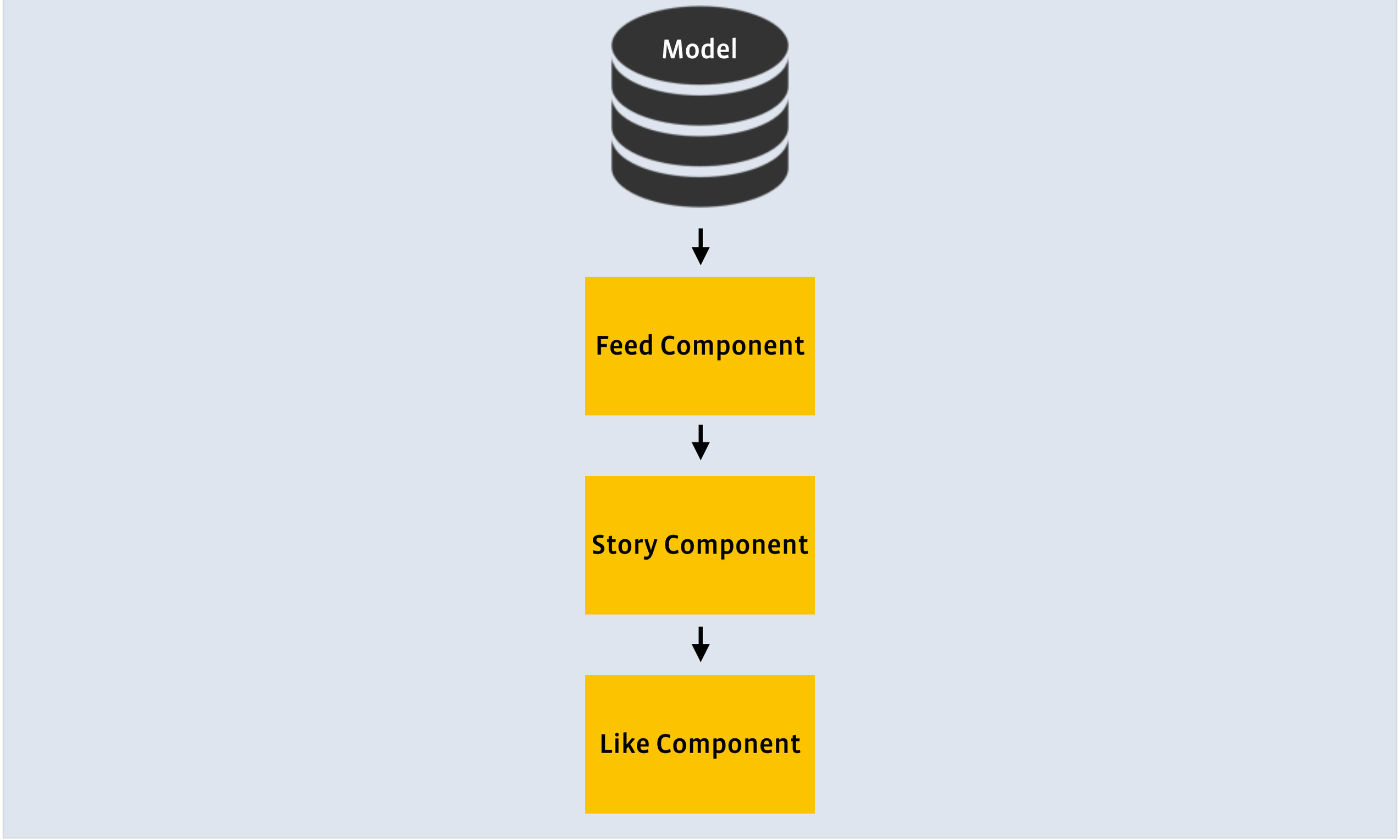

2
H. Wade Minter
25 minutes ago
I'm the DJ, he's the rapper. — at Eagle Fun Fest.
1 Like 1 Comment
Like
Comment
Brian Dewey
27 minutes ago in Seattle
Judging from Facebook, all of my friends are traveling someplace cool this weekend. Jealous...
2 Comments
Hilary Hahn
minutes ago
On April 28, many artists gathered together to honor Beate Gordon, former Asia Society Director of Performing Arts. One of those was "In 27 Pieces" composer Somei Satoh. You can watch a full video of the concert

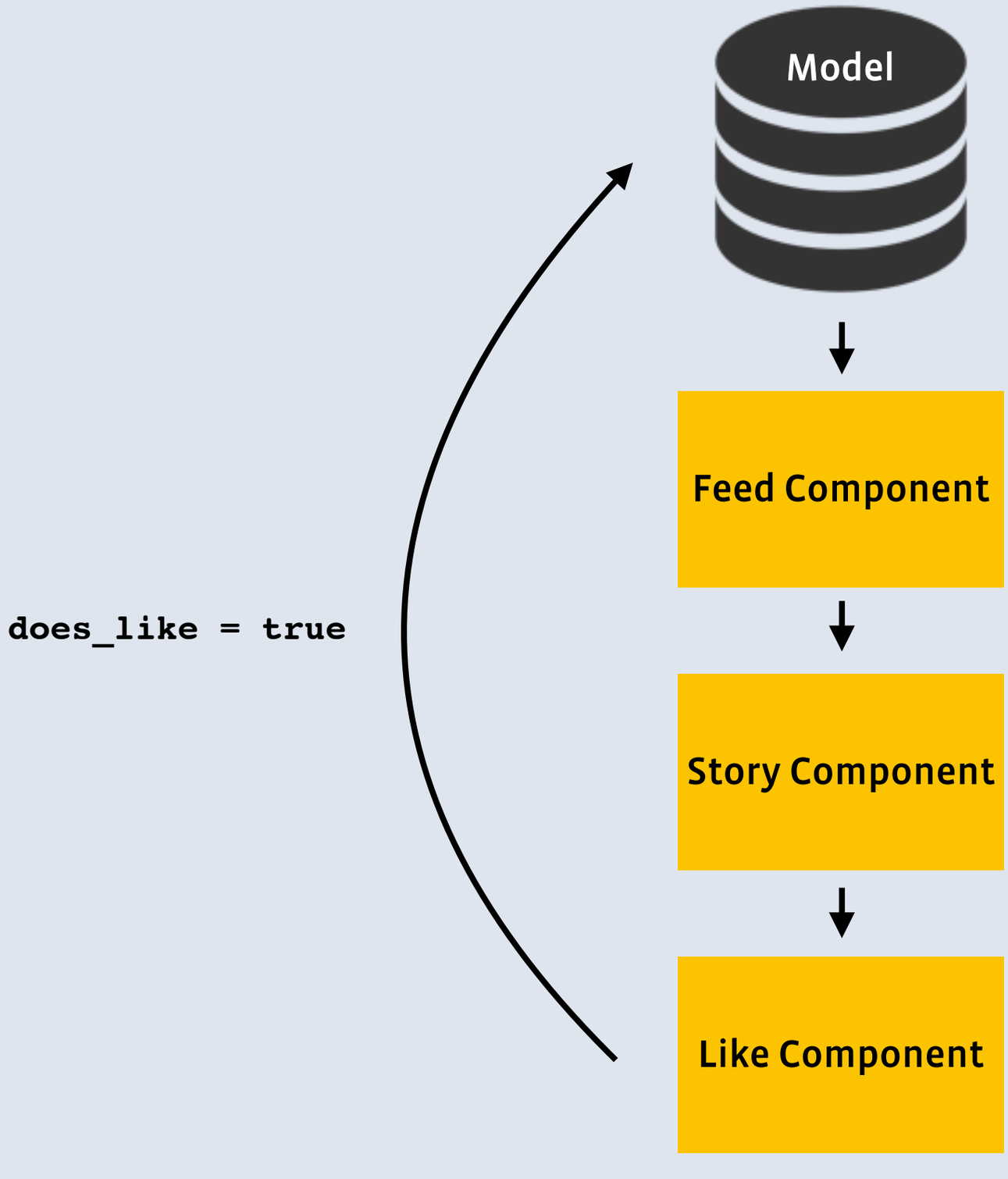
Testing

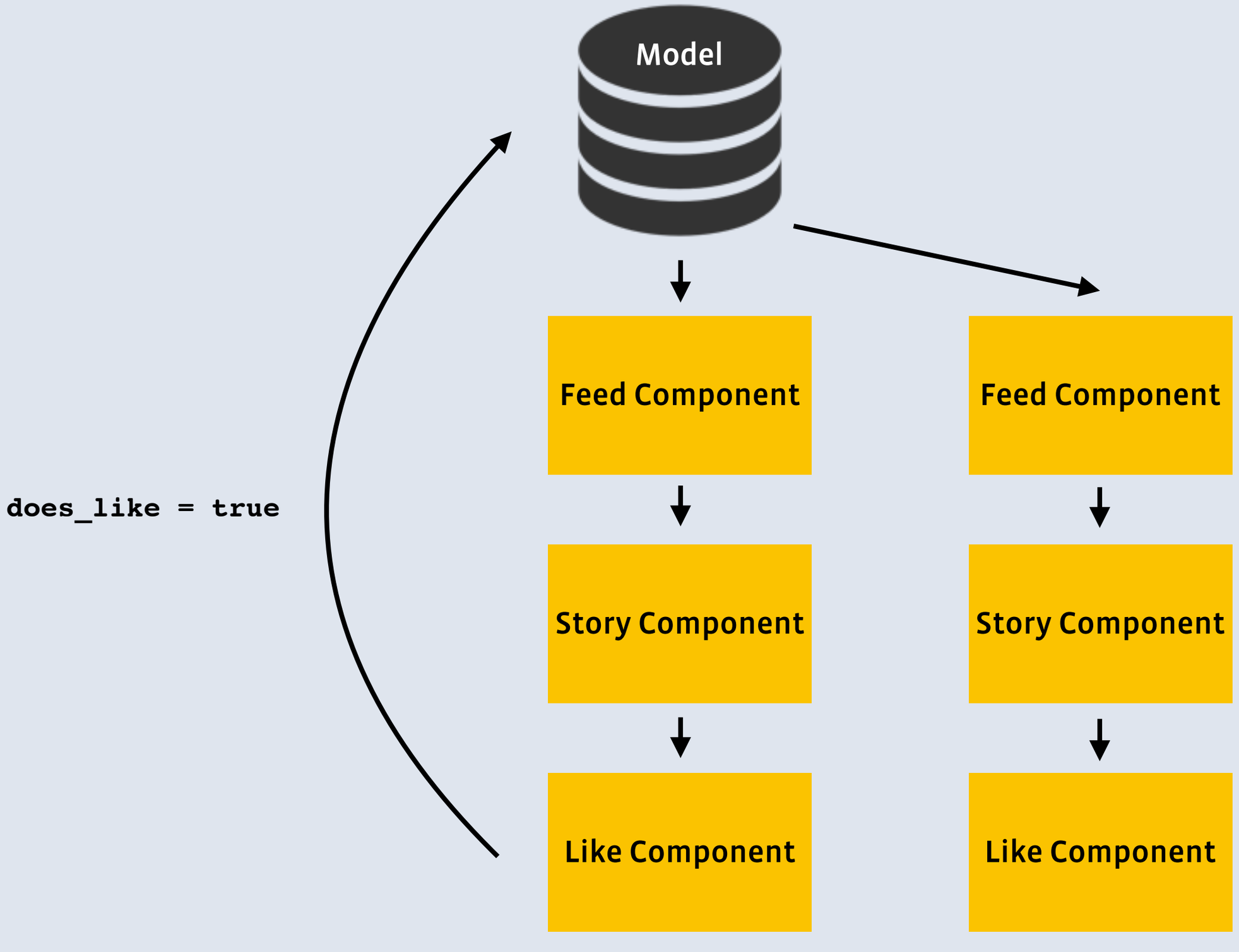
- Test that when the like status is changed, the like controller posts a notification that its height has changed
- Test that when the like controller posts a notification that its height has changed, the story controller posts a notification that *its* height has changed
- Test that when the like status is changed, the like controller updates its view to show the new status
- (Nearly infinite permutations of changes)


ONE WAY


Feed View
Feed Controller
Story View
Story Controller
Model
Like Button
Like Controller
does_like = true


Model
Feed Component
Story Component
Like Component


Model
Feed Component
Story Component
Like Component
does_like = true


Model
Feed Component
Feed Component
Story Component
Story Component
Like Component
Like Component
does_like = true

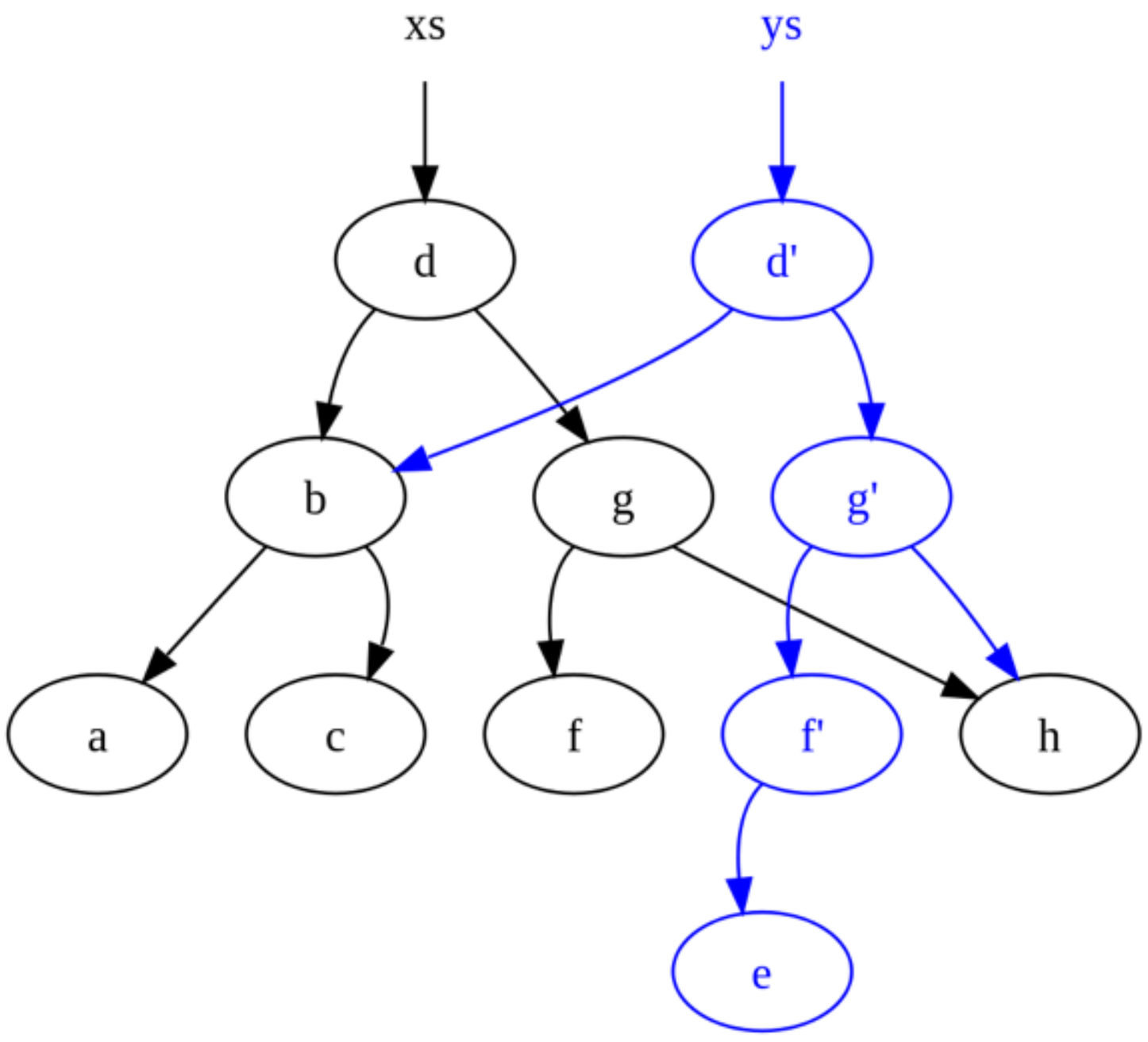
Immutable Models

Immutable Model Object

- Traditional model frameworks:
  - Mutable, thread-affined objects
  - Property change notifications
  - “Save” to propagate changes to elsewhere in app

Immutable Model Object

- Traditional model frameworks:
  - Mutable, thread-affined objects
  - Property change notifications
  - “Save” to propagate changes to elsewhere in app
- Immutable models:
  - Deep-immutable objects
  - When anything changes, brand new top-level tree
  - Top-level “stores” are given “updates” to apply (async)



http://en.wikipedia.org/wiki/Persistent_data_structure

Technical Details

- Code-generated based on server schema
- Plain ol' NSObjects (no faulting, uniquing, or lazy loading)
- Generated by custom templates used with mogenerator
- Support NSCoding for serialization/deserialization (no app-wide "global store")
- Lightweight "consistent cache" keeps trees in sync across products — *eventual* consistency

Immutable Views

“Components”

- Instructions for how to create views
- Immutable
- Composable
- One way dataflow
- Can be constructed off-main-thread

Instructions for Creating Views

- Components are not views; they are instructions for how to create them
- The infrastructure creates, configures, and recycles UIView instances
- Provides layer of indirection between product code and UIKit

Sample Configuration

```
{
  [UIImageView class],
  {
    {@selector(setImage:), image}
    {@selector(setContentMode:), @(UIViewContentModeScale)},
    {@selector(setClipsToBounds:), @YES},
  }
}
```

- Efficient recycling: don’t re-apply “attributes” unless value actually changes
- Encourages declarative style in view code

Immutable

```
- (void)userTappedLike
{
    [FBAPI sendLikeRequestForStory:self.story];
    self.story.doesLike = YES;
    self.story.likeCount += 1;
    self.likeButton.selected = YES;
    [self.likeButton addAnimation:FBLikeAnimation()];
    self.likeCountLabel.text = [self likeCountText];
    if (self.likeCountLabel.hidden) {
        self.likeCountLabel.hidden = NO;
        [self.view setNeedsLayout];
    }
}
```

Immutable

```
- (void)userTappedLike
{
    [FBAPI sendLikeRequestForStory:self.story];
    self.story.doesLike = YES;
    self.story.likeCount += 1;
    self.likeButton.selected = YES;
    [self.likeButton addAnimation:FBLikeAnimation()];
    self.likeCountLabel.text = [self likeCountText];
    if (self.likeCountLabel.hidden) {
        self.likeCountLabel.hidden = NO;
        [self.view setNeedsLayout];
    }
}

- (void)userTappedLike
{
    [FBMutator applyLikeMutation:self.story];
}
```

Immutable

`UIView`
`-layoutSubviews`

Use current (mutable) size to mutate the state of subviews

`CPComponent`
`-layoutThatFits:`

Given a size (parameter), return positions and sizes for children components

Composable

- Favor composition over inheritance
- Far easier to reason about
- Fits naturally with functional style

Threadsafe Creation

- Components are essentially a pure function from data model to an immutable object
- Since their creation has no side effects, they can be created off the main thread
- Important for performance—feed is complex!

Why Not ReactiveCocoa?

- ReactiveCocoa is great—we use it elsewhere
- Encourages a declarative style for writing UI
- Encourages composition
- No one-way dataflow: hard to trace rippling changes
- Less immutability: updates mutable state via “signal” bindings
- No built-in separation from UIView, and thus no builtin thread safety for off-main-thread work

7 -[RACLiveSubscriber sendNext:]
8 __54+[RACLiveSubscriber subscriberForwardingToSubscriber:]_block_invoke
9 -[RACLiveSubscriber sendNext:]
10 __54+[RACLiveSubscriber subscriberForwardingToSubscriber:]_block_invoke
11 -[RACLiveSubscriber sendNext:]
12 __36-[RACSignal(Operations) flattenMap:]_block_invoke_275
13 -[RACLiveSubscriber sendNext:]
14 -[RACReturnSignal attachSubscriber:]
15 -[RACSignal(Subscription) subscribeSavingDisposable:next:error:completed:]
16 __36-[RACSignal(Operations) flattenMap:]_block_invoke_2
17 __36-[RACSignal(Operations) flattenMap:]_block_invoke59
18 -[RACLiveSubscriber sendNext:]
19 __30-[RACSignal(Operations) take:]_block_invoke120
20 -[RACLiveSubscriber sendNext:]
21 __36-[RACSignal(Operations) flattenMap:]_block_invoke_275
22 -[RACLiveSubscriber sendNext:]
23 -[RACReturnSignal attachSubscriber:]
24 -[RACSignal(Subscription) subscribeSavingDisposable:next:error:completed:]
25 __36-[RACSignal(Operations) flattenMap:]_block_invoke_2
26 __36-[RACSignal(Operations) flattenMap:]_block_invoke59
27 -[RACLiveSubscriber sendNext:]
28 __36-[RACSignal(Operations) flattenMap:]_block_invoke_275
29 -[RACLiveSubscriber sendNext:]
30 -[RACReturnSignal attachSubscriber:]
31 -[RACSignal(Subscription) subscribeSavingDisposable:next:error:completed:]
32 __36-[RACSignal(Operations) flattenMap:]_block_invoke_2
33 __36-[RACSignal(Operations) flattenMap:]_block_invoke59
34 -[RACLiveSubscriber sendNext:]
35 __36-[RACSignal(Operations) flattenMap:]_block_invoke_275
36 -[RACLiveSubscriber sendNext:]
37 -[RACReturnSignal attachSubscriber:]
38 -[RACSignal(Subscription) subscribeSavingDisposable:next:error:completed:]
39 __36-[RACSignal(Operations) flattenMap:]_block_invoke_2
40 __36-[RACSignal(Operations) flattenMap:]_block_invoke59
41 -[RACLiveSubscriber sendNext:]

67 -[RACDynamicSignal attachSubscriber:]
68 -[RACSignal(Subscription) subscribeSavingDisposable:next:error:completed:]
69 __30-[RACSignal(Operations) take:]_block_invoke
70 -[RACDynamicSignal attachSubscriber:]
71 -[RACSignal(Subscription) subscribeSavingDisposable:next:error:completed:]
72 __36-[RACSignal(Operations) flattenMap:]_block_invoke_2
73 __36-[RACSignal(Operations) flattenMap:]_block_invoke
74 -[RACDynamicSignal attachSubscriber:]
75 -[RACSignal(Subscription) subscribe:]
76 __31+[RACSignal(Operations) defer:]_block_invoke
77 -[RACDynamicSignal attachSubscriber:]
78 -[RACSignal(Subscription) subscribe:]
79 __32-[RACSignal(Operations) concat:]_block_invoke18
80 -[RACLiveSubscriber sendCompleted]
81 __36-[RACSignal(Operations) flattenMap:]_block_invoke47
82 -[RACLiveSubscriber sendCompleted]
83 __31-[RACSignal(Operations) catch:]_block_invoke442
84 -[RACLiveSubscriber sendCompleted]
85 __35-[RACSignal(Operations) deliverOn:]_block_invoke_2922
86 -[RACQueueScheduler performAsCurrentScheduler:]
87 __30-[RACQueueScheduler schedule:]_block_invoke
88 __RACBacktraceBlock_block_invoke
89 _dispatch_call_block_and_release
90 _dispatch_client_callout
91 _dispatch_queue_drain
92 _dispatch_queue_invoke
93 _dispatch_main_queue_callback_4CF
94 __CFRUNLOOP_IS_SERVICING_THE_MAIN_DISPATCH_QUEUE__
95 __CFRunLoopRun
96 CFRunLoopRunSpecific
97 CFRunLoopRunInMode
98 GSEventRunModal
99 GSEventRun
100 UIApplicationMain
101 main

Objective-C++

> “there is real value in pursuing functional programming, but it would be irresponsible to exhort everyone to abandon their C++ compilers and start coding in Lisp, Haskell, or, to be blunt, any other fringe language…
> C++ doesn’t encourage functional programming, but it doesn’t prevent you from doing it, and you retain the power to drop down [to whatever] nitty-gritty goodness you find the need for.”
>
> —John Carmack

http://www.altdevblogaday.com/2012/04/26/functional-programming-in-c/

Type and Const Safety

```
NSArray *stuff; // of what?

std::vector<CPComponent *> components;
std::vector<CPComponentLayout> layouts;

void CPMountLayouts(
  UIView *container,
  const std::vector<CPComponentLayout> &layouts
);
```

Nil Safe Collections

- Objective-C crashes if you attempt to insert nil in an array
- C++ containers (vector, deque, map) of Objective-C object type allow nil objects
- Functional-style map operations can insert nil objects, and filter can remove them

Efficiency

- Stack-allocated objects
- No overhead to field lookup (vs Obj-C properties)
- C++ move semantics

Aggregate Initialization

- Terse syntax for initializing structs
- Allows verbosity as needed

```
{
   .flex = YES,
   .spacingBefore = 10,
   .spacingAfter = 5
}
```

Sample Code

Adam Kopec
Yesterday at 4:13 PM

Thanks for the write up PSFK! http://www.psfk.com/2014/04/make-it-rain-venmo-app.html#!Fpff0

Venmo Lets You Fling Virtual Dollar Bills Into Your Friends Faces While...
psfk.com

4 Likes

Like Comment Share

Root Story Component

```
[CPStackLayoutComponent
 newWithView:{}
 style:{
   .direction = CPStackLayoutDirectionVertical,
   .alignItems = CPStackLayoutAlignItemsStretch,
   .spacing = 10,
 }
 children:{
   {[CPStoryExplanationComponent newWithStory:story]},
   {[CPStoryHeaderComponent newWithStory:story]},
   {[CPStoryMessageComponent newWithStory:story]},
   {[CPStoryAttachedContentComponent newWithStory:story]},
   {[CPStoryAttachedStoryComponent newWithStory:story]},
   {[CPStorySubstoriesComponent newWithStory:story]},
   {[CPStoryLikeCommentComponent newWithStory:story]}
 }]
```

Button Component

```
[CPButtonComponent
 newWithTitles:{}
 titleColors:{}
 images:
 {
   {UIControlStateNormal, [UIImage imageNamed:@"save"]},
   {UIControlStateHighlighted, [UIImage imageNamed:@"saveHighlighted"]},
   {UIControlStateSelected, [UIImage imageNamed:@"saveSelected"]},
 }
 backgroundImages:{}
 titleFont:nil
 selected:[[entity hasViewerSaved] boolValue]
 enabled:YES
 action:@selector(didTapSaveButton:)
 attributes:{}]
```

Substories

Functional-style map operation

```
CP::map(substories, ^(FBMemFeedStory *substory){
  return CPStackLayoutComponentChild({
    [CPFeedEmbeddedStoryComponent
     newWithStory:substory]
  });
});
```

Future of Components

facebook