Basics + Habits

Building your software projects to last

Session 212

Ken Kocienda

That means great applications

That's why you're at WWDC, right?

Success means change

Updates, bug fixes, new features, etc.

Radical changes rarely work

Most mutations kill the organism

Incremental change is better

Easy-to-change code ideas

I talked about that last year…

…but there were some problems.

Notifications

"Notifications are good. They promote loose coupling."

"Notifications are bad. They're glorified goto statements."

Hygiene

"The best writing is rewriting."

"Don't throw away code."

Paradoxes

"Too many cooks spoil the broth."

"Many hands make light work."

Context

Decisions don’t exist in a vacuum

Basics + Habits

The context for your incremental change

Basics

Fundamental choices you make

Habits

Things you do every day

Basics + Habits

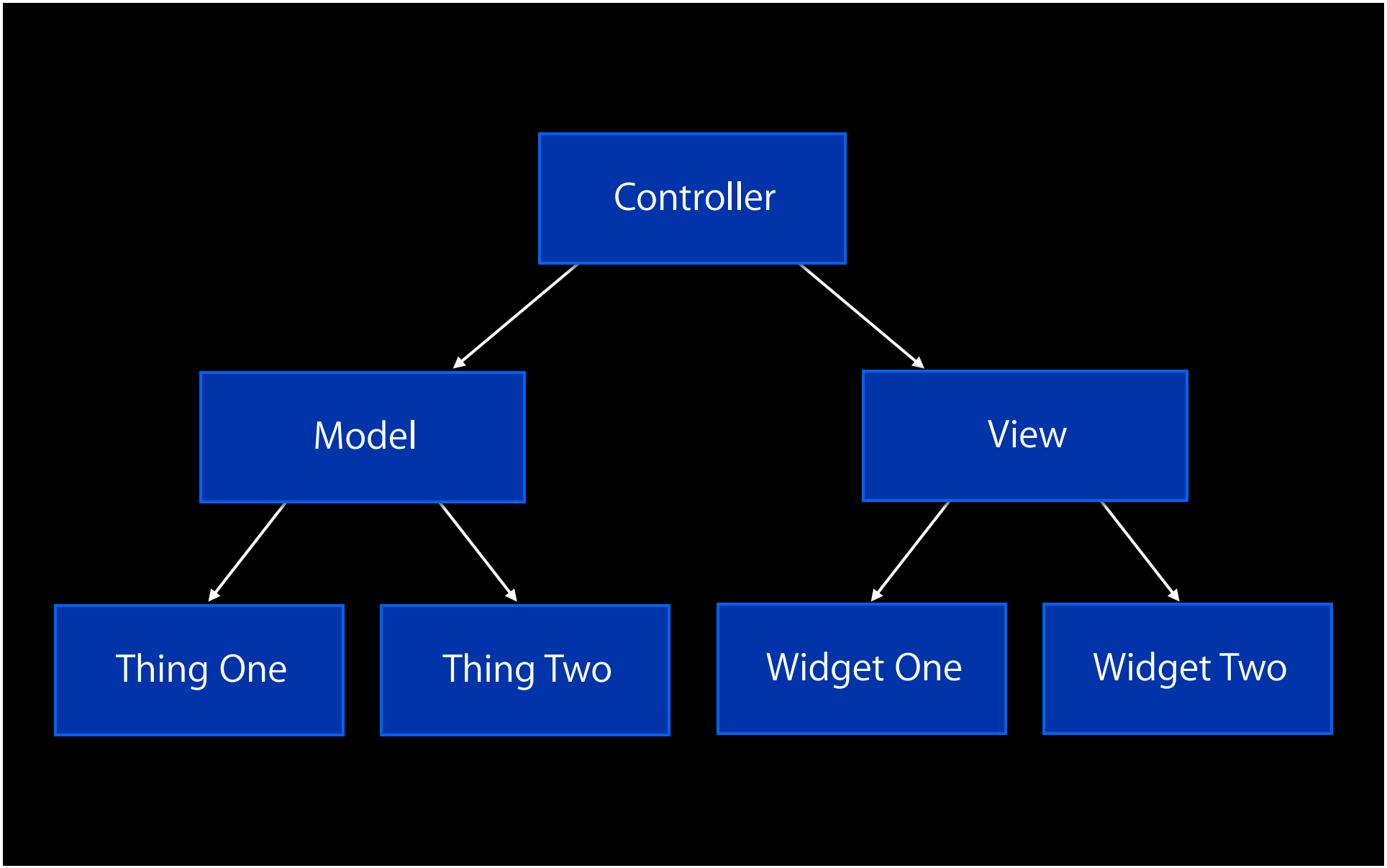
Building your software projects to last

6 Basics

1. Define your physics and chemistry
2.
3.
4.
5.
6.

Physics and chemistry

Fundamental laws and the ways things can mix


Controller
Model
View
Thing One
Thing Two
Widget One
Widget Two

Nouns and verbs

Nouns are abstractions

Verbs combine

Objects and interfaces

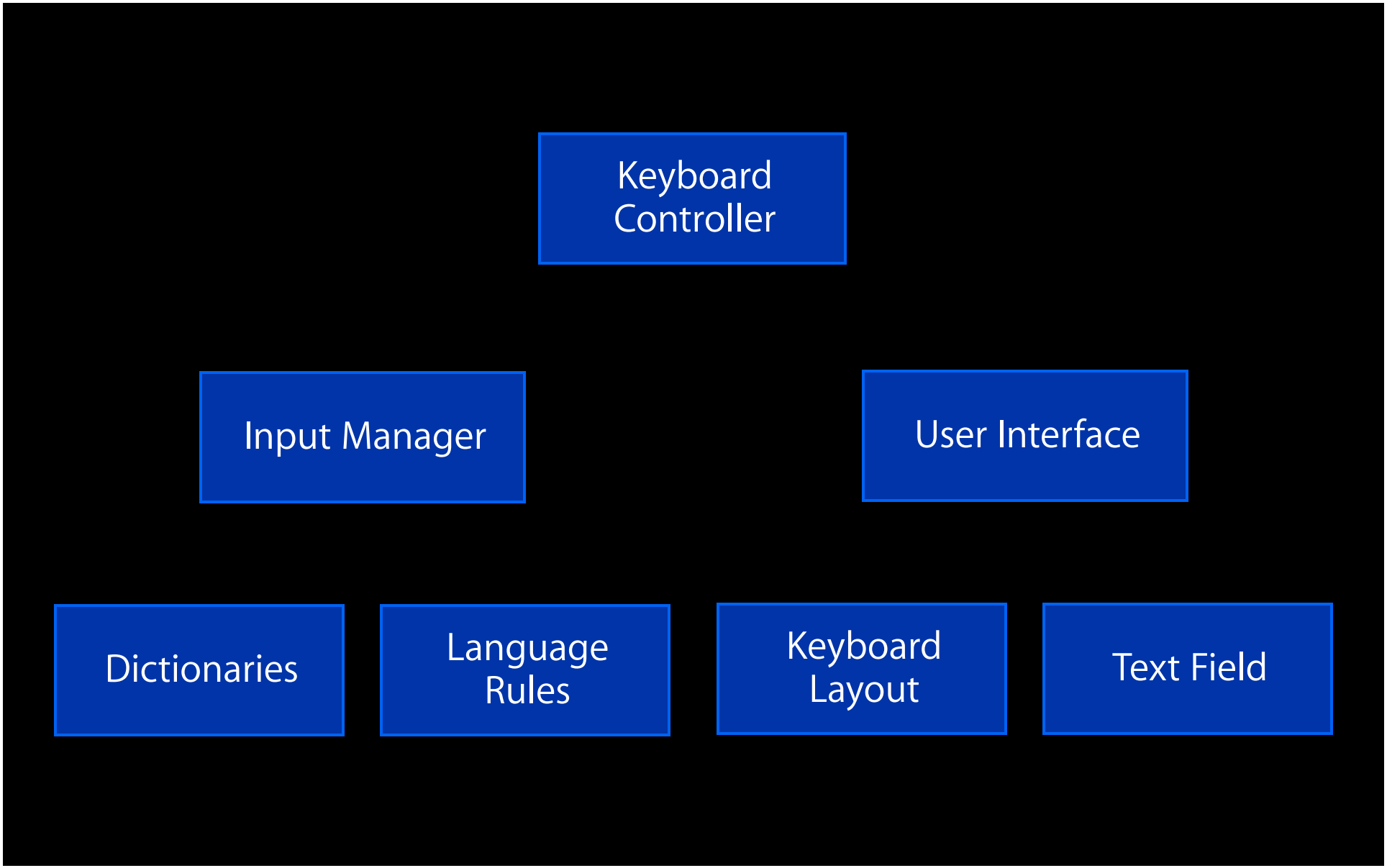
Encapsulated vs. exposed bits

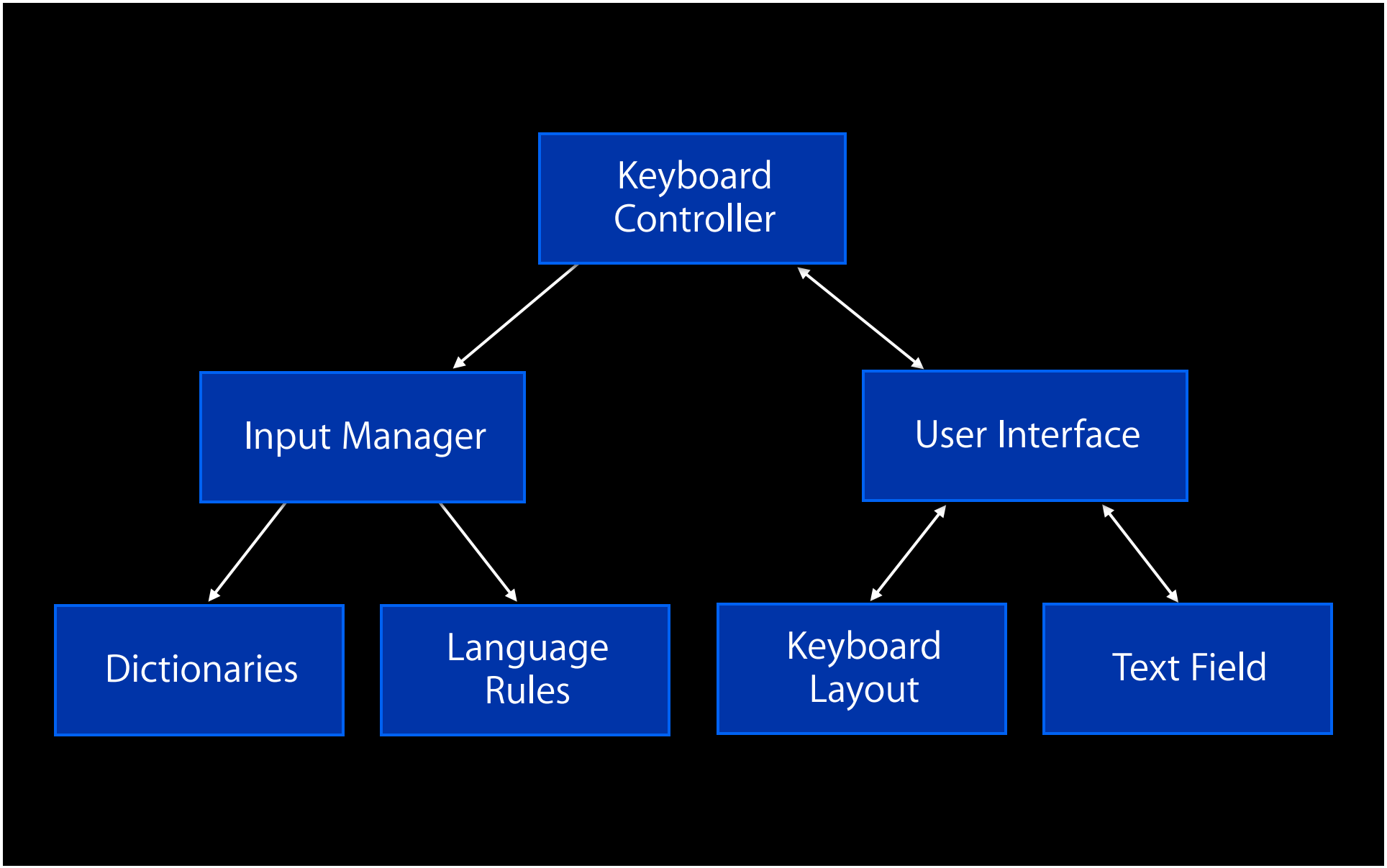
Physics and chemistry

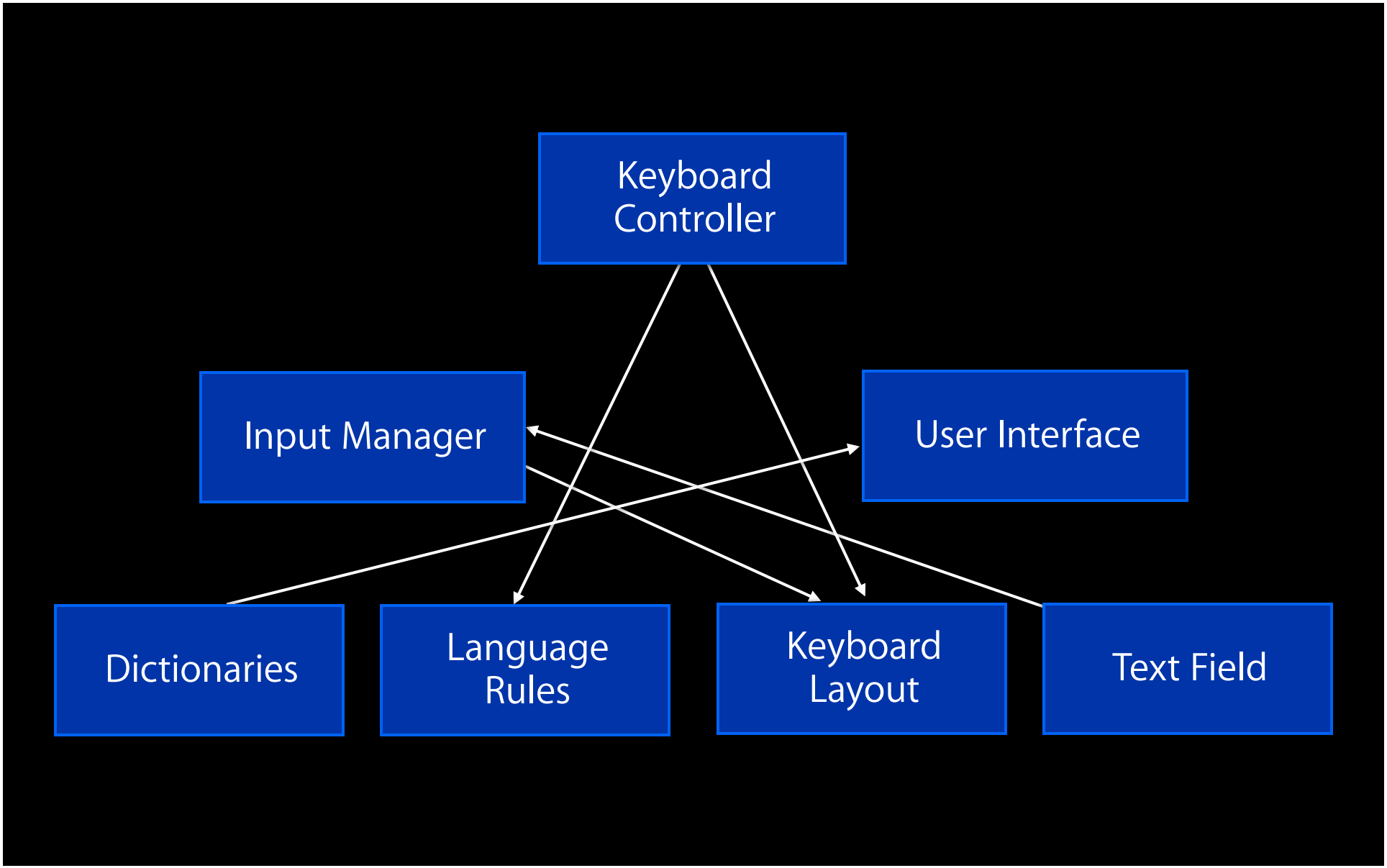
Best analogy

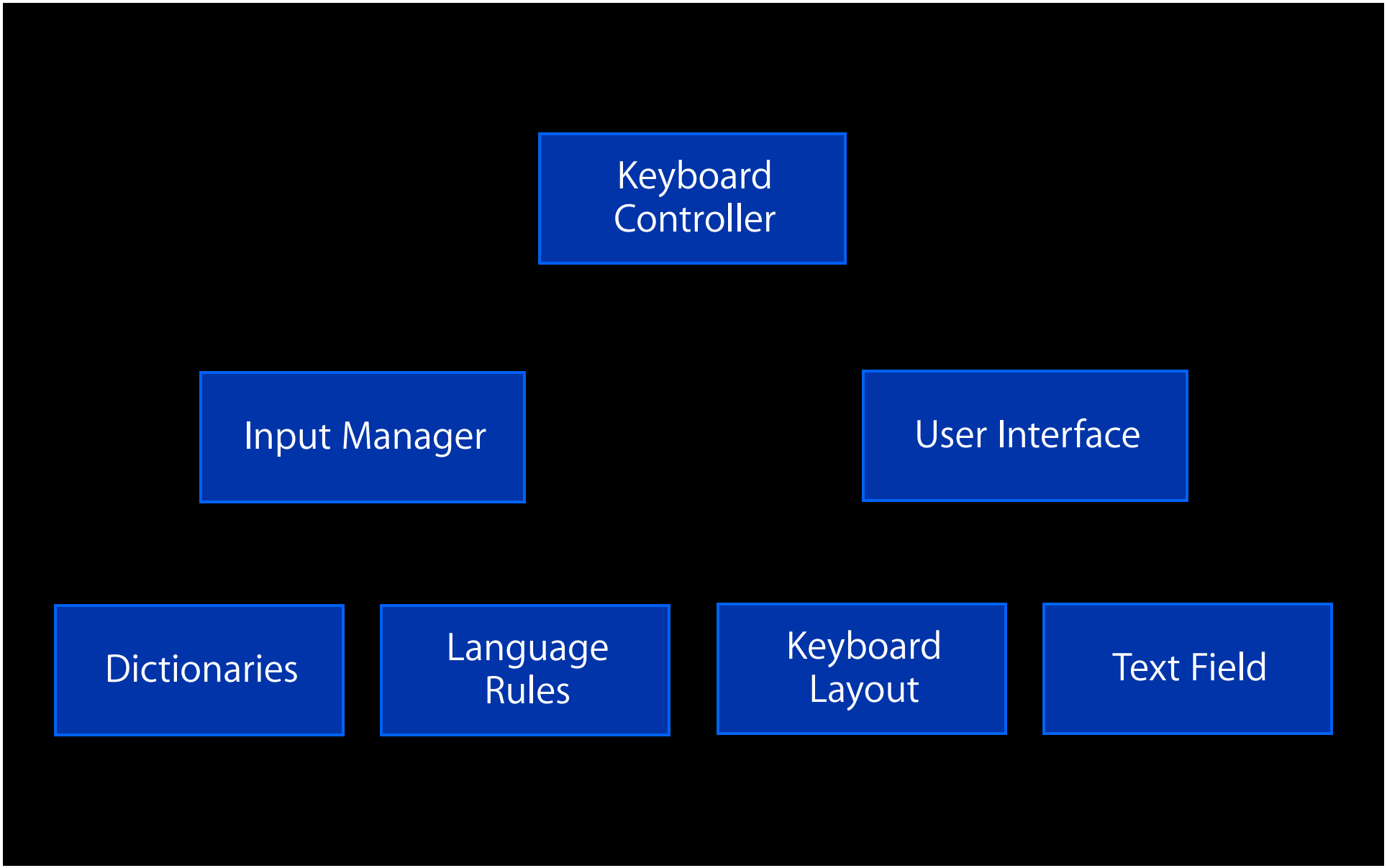
Most coding is chemistry

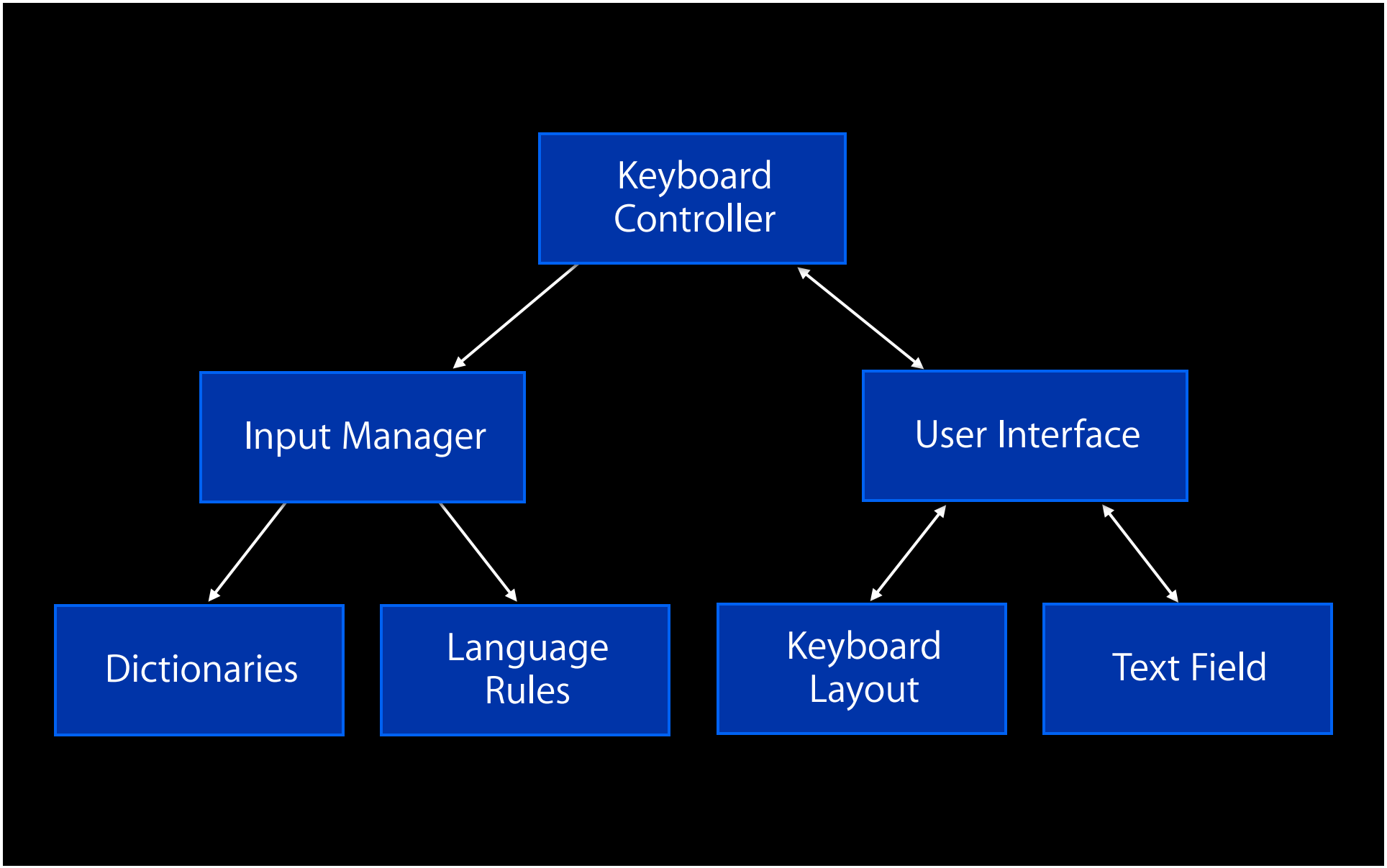
Expressed in terms of physics


Keyboard Controller
Input Manager
User Interface
Dictionaries
Language Rules
Keyboard Layout
Text Field


Keyboard Controller
Input Manager
User Interface
Dictionaries
Language Rules
Keyboard Layout
Text Field


Keyboard Controller
Input Manager
User Interface
Dictionaries
Language Rules
Keyboard Layout
Text Field


Keyboard Controller
Input Manager
User Interface
Dictionaries
Language Rules
Keyboard Layout
Text Field


Keyboard Controller
Input Manager
User Interface
Dictionaries
Language Rules
Keyboard Layout
Text Field

Strive for solid physics

Don’t redefine your universe lightly

Make a sandbox

One you would like to play in

6 Basics

1. Define your physics and chemistry
2.
3.
4.
5.
6.

6 Basics

1. Define your physics and chemistry
2. Choose the right technology
3.
4.
5.
6.

Choosing technology

You’re here at WWDC! You already win!

Choosing technology

You're here at WWDC! You already win!

Hammer or wrench?

Which tool should you reach for?

Core Data

Great for application developers

Core Data

What if you're building a web search engine?

Learn our frameworks!

Know what's available

Matching

Technology to your tasks

Outside the box

Know about what you're giving up

NSString

Great for general-purpose programming

NSString

Great for high-performance string handling?

iPhone keyboard

Needed high-performance string handling

Custom C++ string class

Stack allocated

Short-string optimization

```
namespace KB {

class String
{
public:
    enum { ShortStringCapacity = 16 };

    String();
    ~String();

private:
    char *m_buffer;
    char m_short_string_buffer[ShortStringCapacity];
};
```

Should you do this?

Probably not

Optimization

Physics

Optimization

A technology choice… not an activity

Learn our frameworks!

Know what's available

6 Basics

1. Define your physics and chemistry
2. Choose the right technology
3.
4.
5.
6.

6 Basics

1. Define your physics and chemistry
2. Choose the right technology
3. Build solid abstractions
4.
5.
6.

```
// The dispatch_async() function schedules blocks for concurrent execution
// within the dispatch(3) framework.

void dispatch_async(dispatch_queue_t queue, void (^block)(void));
```

Great feature

Makes concurrent code much easier

```
@implementation MyAwesomeClass

- (void)doAwesomeWork
{
    for (id object in manyObjects) {
        [self doExpensiveWorkWithObject:object];
    }
}

@end
```

```
@implementation MyAwesomeClass

- (void)doAwesomeWork
{
    for (id object in manyObjects) {
        [self doExpensiveWorkWithObject:object]; // Slow!
    }
}

@end
```

```
@implementation MyAwesomeClass

- (void)doAwesomeWork
{
    for (id object in manyObjects) {
        dispatch_async(some_background_queue(), ^{
            [self doExpensiveWorkWithObject:object];
        });
    }
}

@end
```

Wahoo!


Wa oo!

Chemistry without physics

Sprinkling in `dispatch_async` probably won't help

Concurrent code is complex

You need suitable abstractions

Objects and interfaces

Define work in terms of the job you want done

```
@implementation MyAwesomeClass

- (void)doAwesomeWork
{
    for (id object in manyObjects) {
        [self doExpensiveWorkWithObject:object];
    }
}

@end
```

```
// An object which generates thumbnails asynchronously.

@protocol ThumbnailObserver;

@interface ThumbnailMaker : NSObject
- (void)makeThumbnail:(UIImage *)image observer:(id <ThumbnailObserver>)observer;
@end

@protocol ThumbnailObserver
- (void)thumbnailAvailable:(UIImage *)thumbnail;
@end
```

```
@implementation MyPhotoProgram <ThumbnailMakerObserver>

- (void)buildThumbnailForImage:(UIImage *)image {
        dispatch_async(some_background_queue(), ^{
             [self.thumbnailMaker makeThumbnail:image observer:self];
        });
}

- (void)thumbnailAvailable:(UIImage *)thumbnail { /* use thumbnail */ }

@end

@implementation ThumbnailMaker : NSObject

- (void)makeThumbnail:(UIImage *)image observer:(id <ThumbnailObserver>)observer
{
    // do work to generate thumbnail
    dispatch_async(dispatch_get_main_queue(), ^{
         [observer thumbnailAvailable:thumbnailImage];
    });
}
…
```

```
@implementation MyPhotoProgram <ThumbnailMakerObserver>

- (void)buildThumbnailForImage:(UIImage *)image {
        dispatch_async(some_background_queue(), ^{
             [self.thumbnailMaker makeThumbnail:image observer:self];
        });
}

- (void)thumbnailAvailable:(UIImage *)thumbnail { /* use thumbnail */ }

@end

@implementation ThumbnailMaker : NSObject

- (void)makeThumbnail:(UIImage *)image observer:(id <ThumbnailObserver>)observer
{
    // do work to generate thumbnail
    dispatch_async(dispatch_get_main_queue(), ^{
         [observer thumbnailAvailable:thumbnailImage];
    });
}
…
```

```
@implementation MyPhotoProgram <ThumbnailMakerObserver>

- (void)buildThumbnailForImage:(UIImage *)image {
        dispatch_async(some_background_queue(), ^{
             [self.thumbnailMaker makeThumbnail:image observer:self];
        });
}

- (void)thumbnailAvailable:(UIImage *)thumbnail { /* use thumbnail */ }

@end

@implementation ThumbnailMaker : NSObject

- (void)makeThumbnail:(UIImage *)image observer:(id <ThumbnailObserver>)observer
{
    // do work to generate thumbnail
    dispatch_async(dispatch_get_main_queue(), ^{
         [observer thumbnailAvailable:thumbnailImage];
    });
}
…
```

```
@implementation MyPhotoProgram <ThumbnailMakerObserver>

- (void)buildThumbnailForImage:(UIImage *)image {
        dispatch_async(some_background_queue(), ^{
             [self.thumbnailMaker makeThumbnail:image observer:self];
        });
}

- (void)thumbnailAvailable:(UIImage *)thumbnail { /* use thumbnail */ }

@end

@implementation ThumbnailMaker : NSObject

- (void)makeThumbnail:(UIImage *)image observer:(id <ThumbnailObserver>)observer
{
    // do work to generate thumbnail
    dispatch_async(dispatch_get_main_queue(), ^{
         [observer thumbnailAvailable:thumbnailImage];
    });
}
…
```

```
@implementation MyPhotoProgram <ThumbnailMakerObserver>

- (void)buildThumbnailForImage:(UIImage *)image {
        dispatch_async(some_background_queue(), ^{
             [self.thumbnailMaker makeThumbnail:image observer:self];
        });
}

- (void)thumbnailAvailable:(UIImage *)thumbnail { /* use thumbnail */ }

@end

@implementation ThumbnailMaker : NSObject

- (void)makeThumbnail:(UIImage *)image observer:(id <ThumbnailObserver>)observer
{
    // do work to generate thumbnail
    dispatch_async(dispatch_get_main_queue(), ^{
         [observer thumbnailAvailable:thumbnailImage];
    });
}
…
```

```
@implementation MyPhotoProgram <ThumbnailMakerObserver>

- (void)buildThumbnailForImage:(UIImage *)image {
        dispatch_async(some_background_queue(), ^{
             [self.thumbnailMaker makeThumbnail:image observer:self];
        });
}

- (void)thumbnailAvailable:(UIImage *)thumbnail { /* use thumbnail */ }

@end

@implementation ThumbnailMaker : NSObject

- (void)makeThumbnail:(UIImage *)image observer:(id <ThumbnailObserver>)observer
{
    // do work to generate thumbnail
    dispatch_async(dispatch_get_main_queue(), ^{
         [observer thumbnailAvailable:thumbnailImage];
    });
}
…
```

```
@implementation MyPhotoProgram <ThumbnailMakerObserver>

- (void)buildThumbnailForImage:(UIImage *)image {
        dispatch_async(some_background_queue(), ^{
             [self.thumbnailMaker makeThumbnail:image observer:self];
        });
}

- (void)thumbnailAvailable:(UIImage *)thumbnail { /* use thumbnail */ }

@end

@implementation ThumbnailMaker : NSObject

- (void)makeThumbnail:(UIImage *)image observer:(id <ThumbnailObserver>)observer
{
    // do work to generate thumbnail
    dispatch_async(dispatch_get_main_queue(), ^{
         [observer thumbnailAvailable:thumbnailImage];
    });
}
…
```

```
@implementation MyPhotoProgram <ThumbnailMakerObserver>

- (void)buildThumbnailForImage:(UIImage *)image {
        dispatch_async(some_background_queue(), ^{
             [self.thumbnailMaker makeThumbnail:image observer:self];
        });
}

- (void)thumbnailAvailable:(UIImage *)thumbnail { /* use thumbnail */ }

@end

@implementation ThumbnailMaker : NSObject

- (void)makeThumbnail:(UIImage *)image observer:(id <ThumbnailObserver>)observer
{
    // do work to generate thumbnail
    dispatch_async(dispatch_get_main_queue(), ^{
         [observer thumbnailAvailable:thumbnailImage];
    });
}
…
```

Chemistry with physics

`dispatch_async`

Thinking

Thumbnails not dispatch queues

Interface

Stand the test of time

Implementation

Change to server-generated thumbnails

Solid abstraction

Define work in terms of the job you want done

6 Basics

1. Define your physics and chemistry
2. Choose the right technology
3. Build solid abstractions
4.
5.
6.

6 Basics

1. Define your physics and chemistry
2. Choose the right technology
3. Build solid abstractions
4. Optimize for humans
5.
6.

Data and wire formats

XML, JSON, plists, Google Protocol Buffers

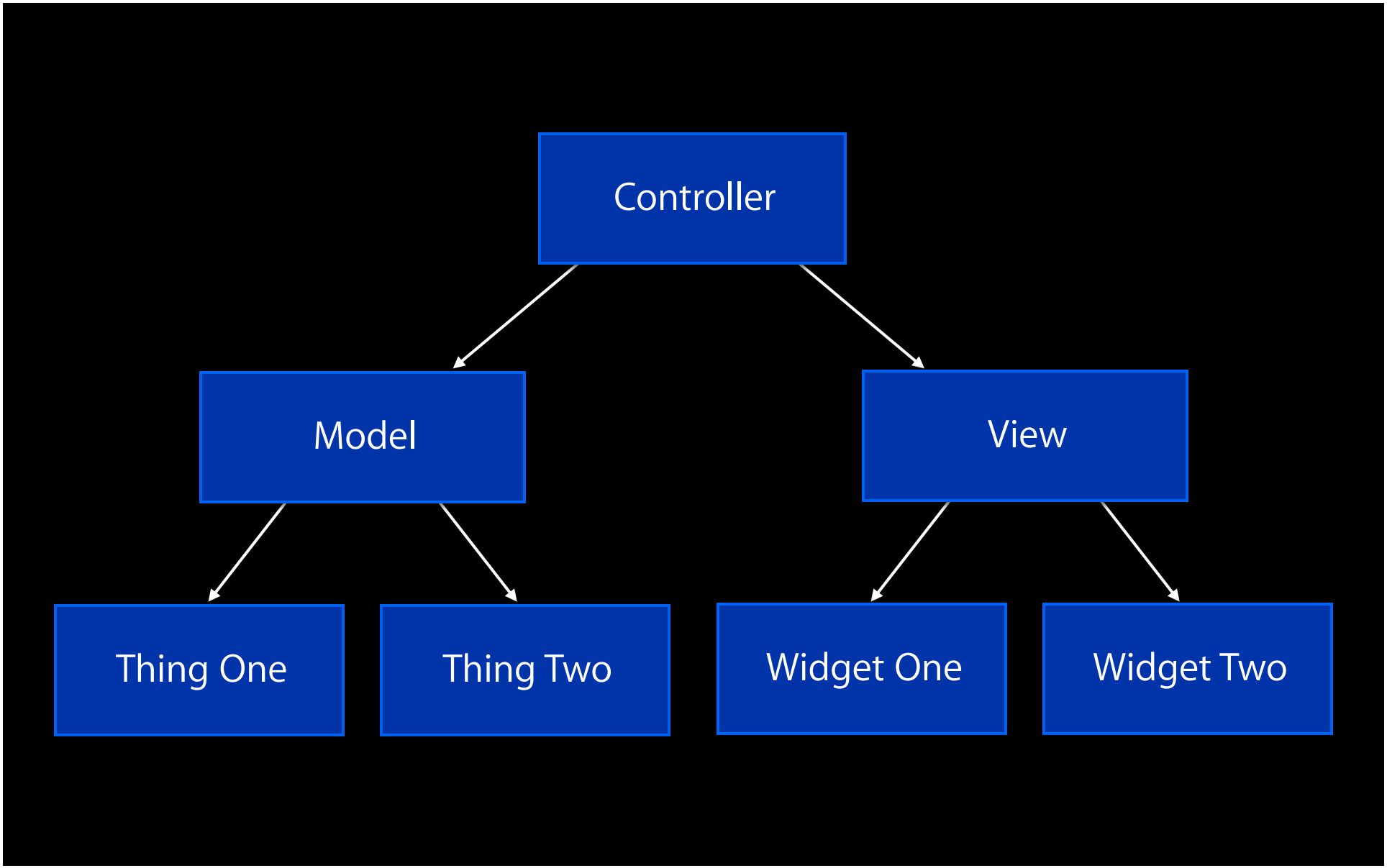
```
// A JSON structure.
{
   "class" : "Coupon",
   "properties" : {
      "amount" : 25,
      "expiration" : "Wed Jun  20 00:00:00 PDT 2012",
      "name" : "Geek Goodies",
      "pitch" : "Get $25 off on your next purchase!",
   },
}
```

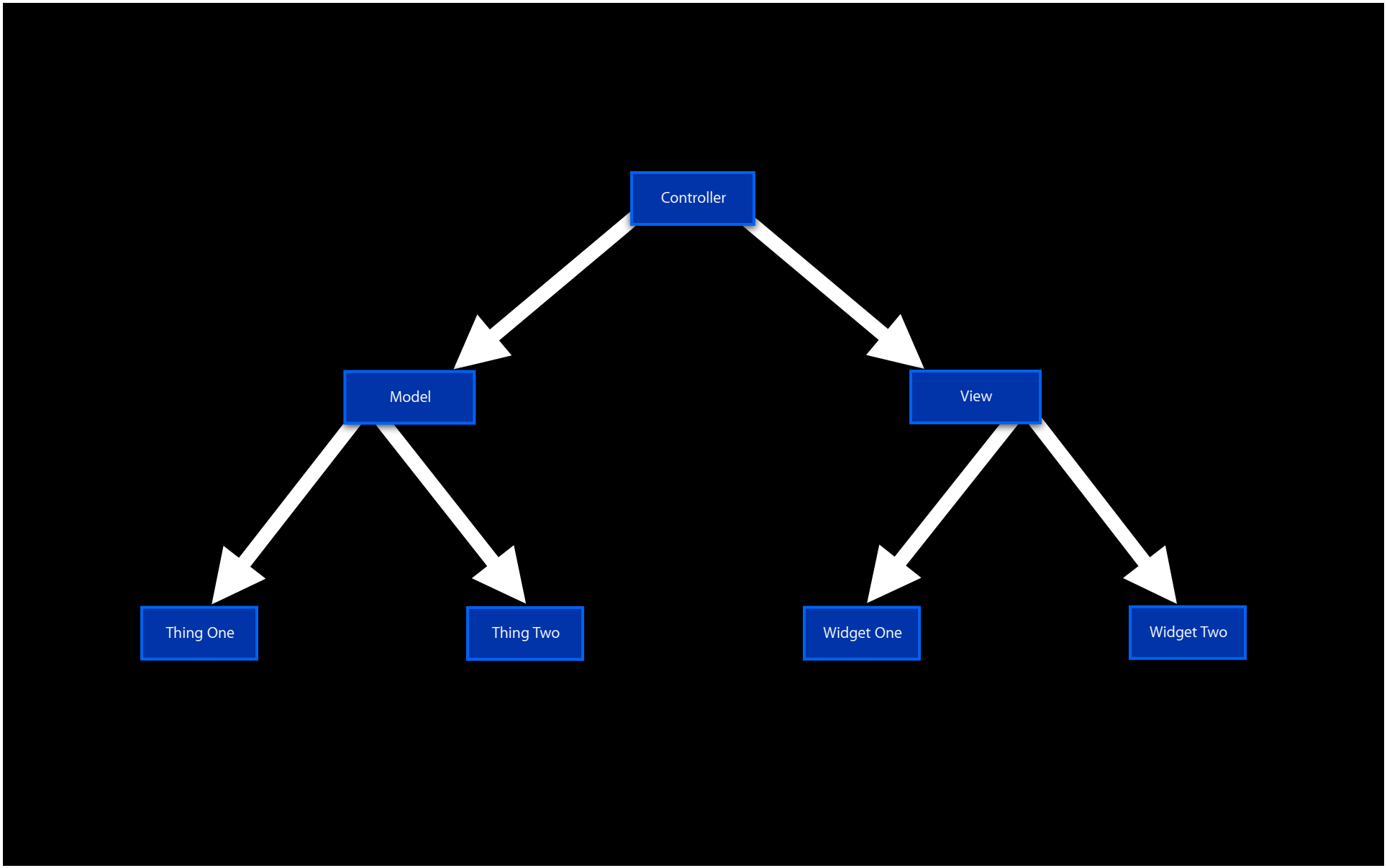
```
// A JSON structure.
{
   "class" : "Coupon",
   "properties" : {
      "amount" : 25,
      "expiration" : "Wed Jun  20 00:00:00 PDT 2012",
      "name" : "Geek Goodies",
      "pitch" : "Get $25 off on your next purchase!",
   },
}



// An object backed by this JSON structure.
@interface Coupon : NSObject

- (id)getProperty:(NSString *)key;
- (void)setProperty:(id)object forKey:(NSString *)key;

@end
```

```
// A JSON structure.
{
   "class" : "Coupon",
   "properties" : {
      "amount" : 25,
      "expiration" : "Wed Jun  20 00:00:00 PDT 2012",
      "name" : "Geek Goodies",
      "pitch" : "Get $25 off on your next purchase!",
   },
}


// An object backed by this JSON structure.
@interface Coupon : NSObject

- (id)getProperty:(NSString *)key;
- (void)setProperty:(id)object forKey:(NSString *)key;

@end
```

Chemistry trumps physics

Weird universe


Controller
Model
View
Thing One
Thing Two
Widget One
Widget Two


Controller
Model
View
Thing One
Thing Two
Widget One
Widget Two

Brain power is the scarcest resource

Not CPU power or network bandwidth or…

```
// A JSON structure.
{
   "class" : "Coupon",
   "properties" : {
      "amount" : 25,
      "expiration" : "Wed Jun  20 00:00:00 PDT 2012",
      "name" : "Geek Goodies",
      "pitch" : "Get $25 off on your next purchase!",
   },
}



// An object backed by this JSON structure.
@interface Coupon : NSObject

- (id)getProperty:(NSString *)key;
- (void)setProperty:(id)object forKey:(NSString *)key;

@end
```

```
// An object backed by this JSON structure.
@interface Coupon : NSObject

- (id)getProperty:(NSString *)key;
- (void)setProperty:(id)object forKey:(NSString *)key;

@end
```

Weak Object Interface

```
// An object backed by this JSON structure.
@interface Coupon : NSObject

- (id)getProperty:(NSString *)key;
- (void)setProperty:(id)object forKey:(NSString *)key;

@end
```

Better Object Interface

```
// An coupon object backed by a JSON structure.

@interface Coupon : NSObject

- (id)initWithJSONData:(NSData *)data;
- (NSData *)asJSONData;

@property (strong) NSString *name;
@property (strong) NSString *pitch;
@property (strong) NSNumber *amount;
@property (strong) NSDate *expirationDate;

@end
```

Better Object Interface

```
// An coupon object backed by a JSON structure.

@interface Coupon : NSObject

- (id)initWithJSONData:(NSData *)data;
- (NSData *)asJSONData;

@property (strong) NSString *name;
@property (strong) NSString *pitch;
@property (strong) NSNumber *amount;
@property (strong) NSDate *expirationDate;

@end
```

Better Object Interface

```
// An coupon object backed by a JSON structure.

@interface Coupon : NSObject

- (id)initWithJSONData:(NSData *)data;
- (NSData *)asJSONData;

@property (strong) NSString *name;
@property (strong) NSString *pitch;
@property (strong) NSNumber *amount;
@property (strong) NSDate *expirationDate;

@end
```

Optimize for humans

Not object serialization or networking protocols

6 Basics

1. Define your physics and chemistry
2. Choose the right technology
3. Build solid abstractions
4. Optimize for humans
5.
6.

6 Basics

1. Define your physics and chemistry
2. Choose the right technology
3. Build solid abstractions
4. Optimize for humans
5. Focus your development effort
6.

Focus

What are you trying to be great at?

Build that

Stand on the shoulders of iOS and OS X

Learn our frameworks!

Know what's available

Recursive physics and chemistry

Our chemistry becomes your physics

iOS device autorotation?

Learn about UIViewController

Need database?

Learn about Core Data

Need animations?

Learn about Core Animation

Need X?

Ask in labs, developer forums, etc.

Make or break your application

Work on that!

6 Basics

1. Define your physics and chemistry
2. Choose the right technology
3. Build solid abstractions
4. Optimize for humans
5. Focus your development effort
6.

6 Basics

1. Define your physics and chemistry
2. Choose the right technology
3. Build solid abstractions
4. Optimize for humans
5. Focus your development effort
6. Look to the horizon

What if you're successful?

Can you handle it?

Performance

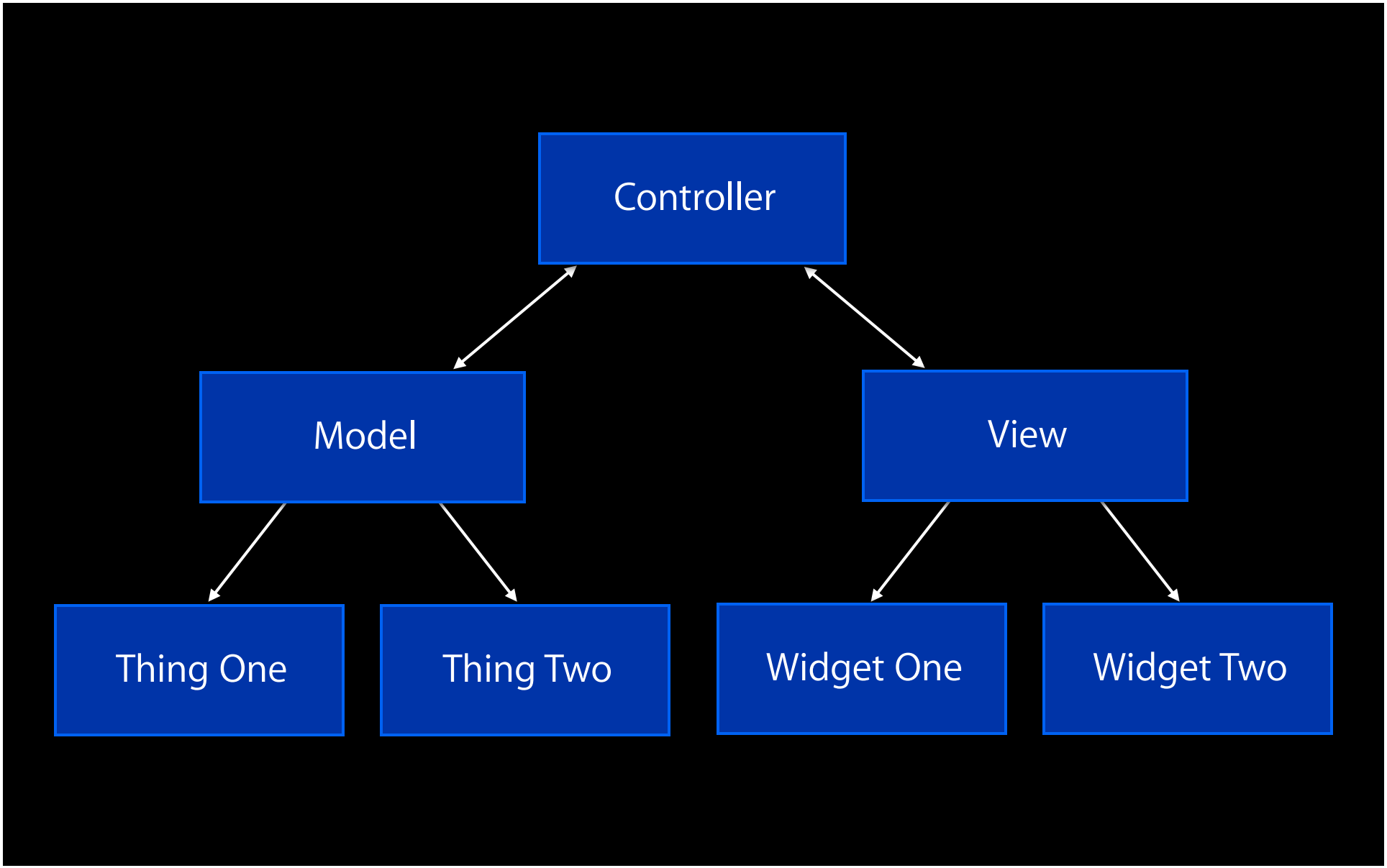
How much headroom do you have?

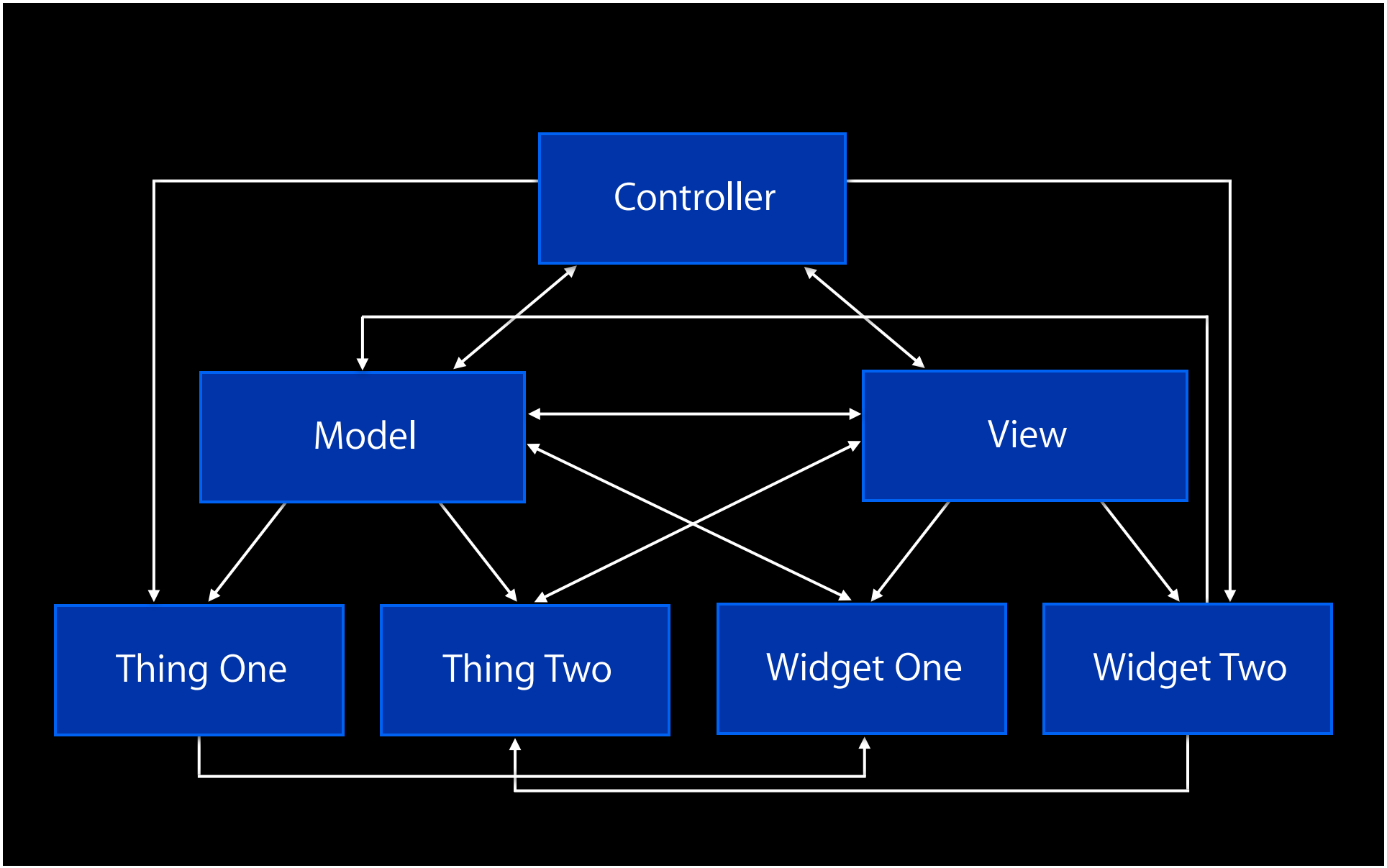
Internationalization

Strings, dates, times, addresses, names

Next feature idea?

Will it blend?


Controller
Model
View
Thing One
Thing Two
Widget One
Widget Two


Controller
Model
View
Thing One
Thing Two
Widget One
Widget Two

Stop and clean up

Refactor your code before version 2.0

6 Basics

1. Define your physics and chemistry
2. Choose the right technology
3. Build solid abstractions
4. Optimize for humans
5. Focus your development effort
6. Look to the horizon

Basics

Fundamental choices you make

Habits

Things you do every day

6 Habits

1. Communicate
2.
3.
4.
5.
6.

Communication

Implicit in all the other habits

Shared vision

Everyone should have a big picture

The Whiteboard Test

Draw software architecture on a whiteboard

6 Habits

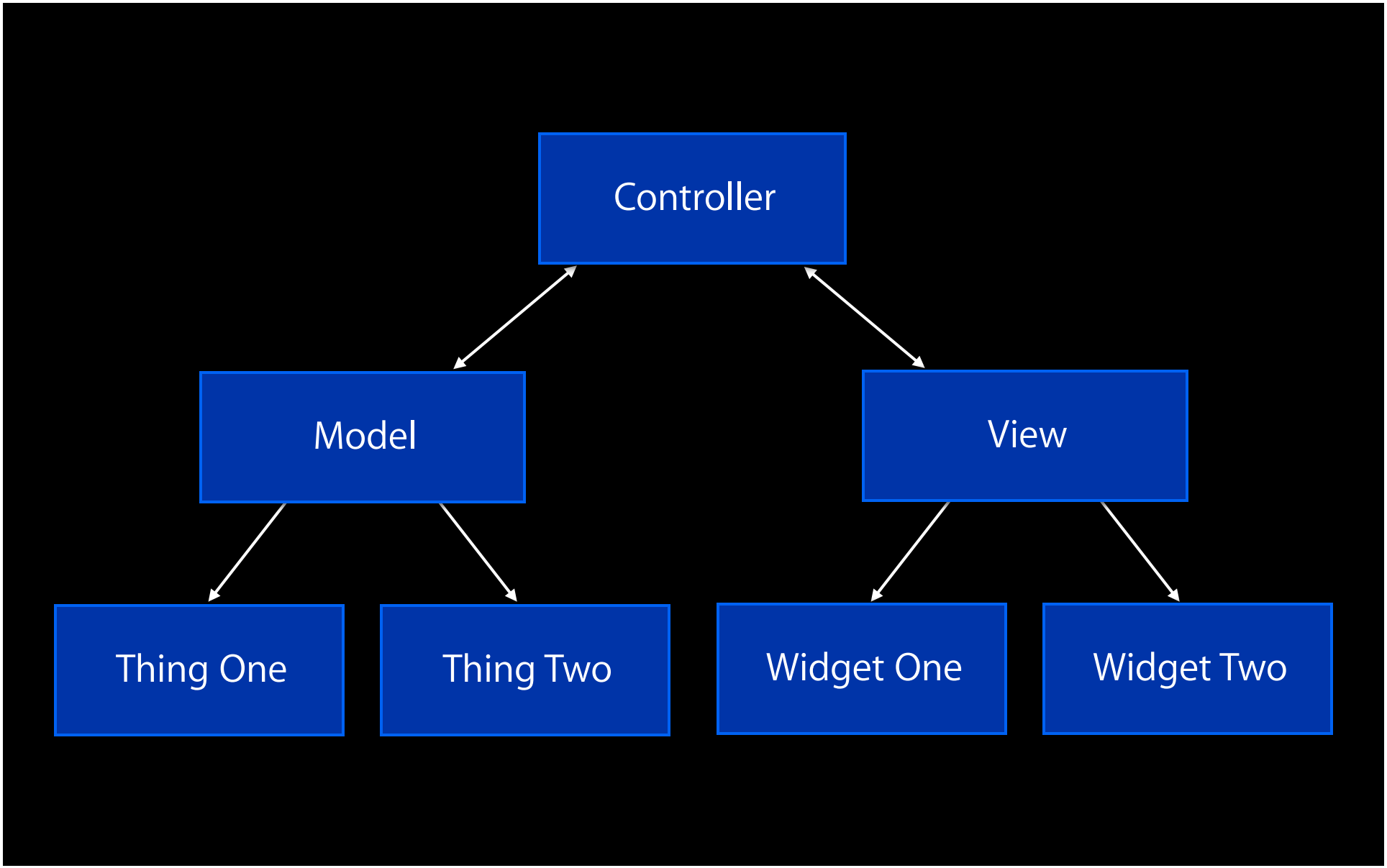
1. Communicate
2.
3.
4.
5.
6.

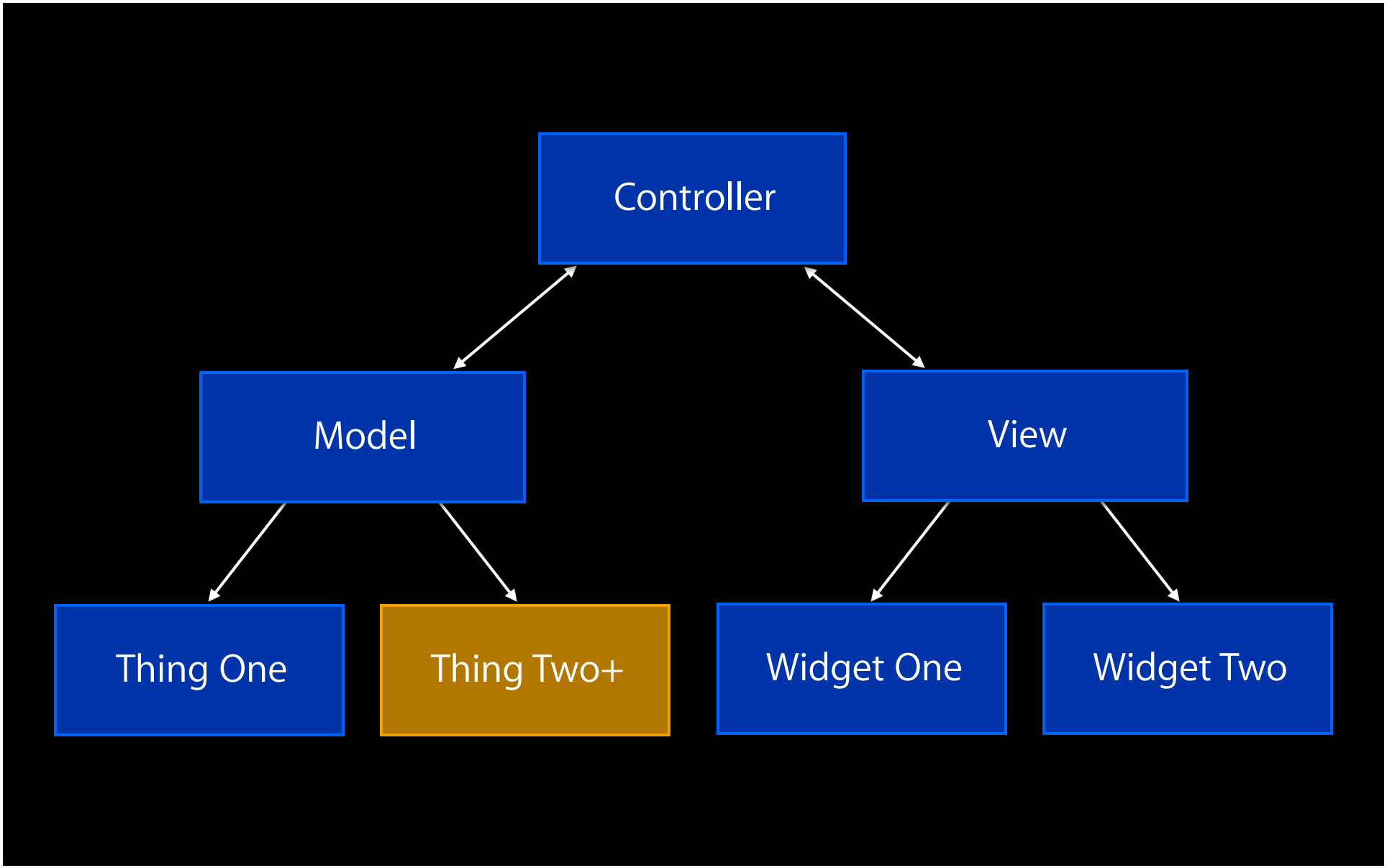
6 Habits

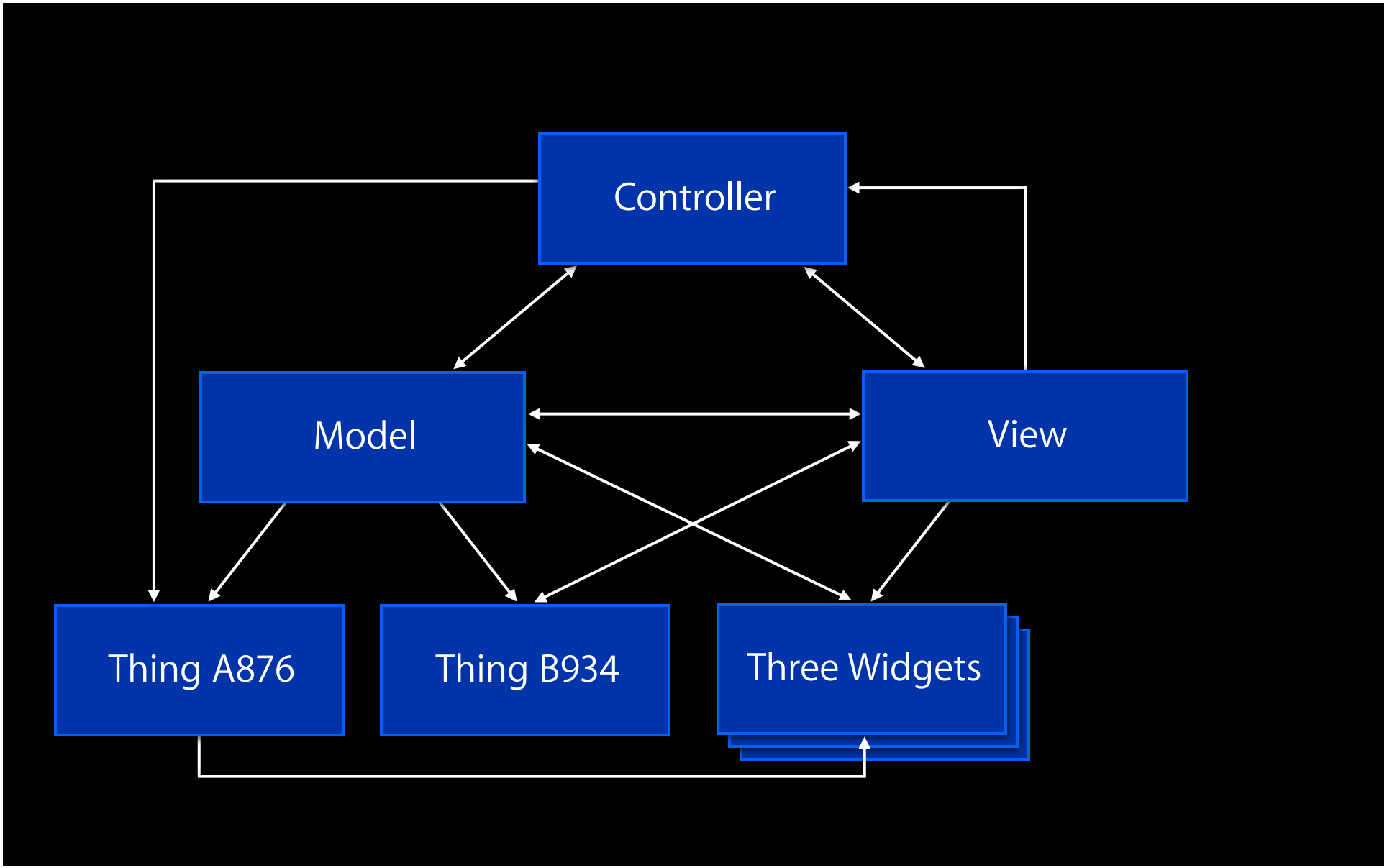
1. Communicate
2. Make changes
3.
4.
5.
6.

Obvious and unsurprising change

Smaller changes are better


Controller
Model
View
Thing One
Thing Two
Widget One
Widget Two


Controller
Model
View
Thing One
Thing Two+
Widget One
Widget Two


Controller
Model
View
Thing A876
Thing B934
Three Widgets

Consider WebKit

Big changes are possible

Communication

Step by step

6 Habits

1. Communicate
2. Make changes
3.
4.
5.
6.

6 Habits

1. Communicate
2. Make changes
3. Write code
4.
5.
6.

Write code for each other

Self documenting

Still don’t know what this does

```
[_rightView setAlpha:![[_temporary text] length] ? 1.0 : 0.0];
```

```
--- trunk/Safari/mac/LocationTextField.mm 2012-02-17 02:19:50 UTC (rev 40929)
+++ trunk/Safari/mac/LocationTextField.mm 2012-02-17 13:23:42 UTC (rev 40930)
@@ -59,6 +59,7 @@
 // Location text field internal components constants
 static const CGFloat MarginBeforeStandardLeftmostButton          = 7;
 static const CGFloat MarginBeforeRolloverLeftmostButton          = 4;
+static const CGFloat MarginBeforeURL                             = 1;
 static const CGFloat MarginBetweenButtons                        = 3;
 static const CGFloat MarginAfterRightmostButton                  = 5;
 static const CGFloat EVCertificateButtonYOffset                  = 4;
@@ -2764,7 +2765,9 @@
         [component updateFrame:componentFrame];
     }

-    // Only the URL component remains; use the remaining bounds as its frame.
+    // Only the URL component remains; start with the
+    // remaining bounds, then adjust for initial margin.
+    remainingBounds.origin.x += MarginBeforeURL;
+    remainingBounds.size.width -= MarginBeforeURL;
     [_urlComponent updateFrame:remainingBounds];
 }
```

```
--- trunk/Safari/mac/LocationTextField.mm 2012-02-17 02:19:50 UTC (rev 40929)
+++ trunk/Safari/mac/LocationTextField.mm 2012-02-17 13:23:42 UTC (rev 40930)
@@ -59,6 +59,7 @@
 // Location text field internal components constants
 static const CGFloat MarginBeforeStandardLeftmostButton          = 7;
 static const CGFloat MarginBeforeRolloverLeftmostButton          = 4;
+static const CGFloat MarginBeforeURL                             = 1;
 static const CGFloat MarginBetweenButtons                        = 3;
 static const CGFloat MarginAfterRightmostButton                  = 5;
 static const CGFloat EVCertificateButtonYOffset                  = 4;
@@ -2764,7 +2765,9 @@
         [component updateFrame:componentFrame];
     }

-    // Only the URL component remains; use the remaining bounds as its frame.
+    // Only the URL component remains; start with the
+    // remaining bounds, then adjust for initial margin.
+    remainingBounds.origin.x += MarginBeforeURL;
+    remainingBounds.size.width -= MarginBeforeURL;
     [_urlComponent updateFrame:remainingBounds];
 }
```

Not scary

I could jump right in

6 Habits

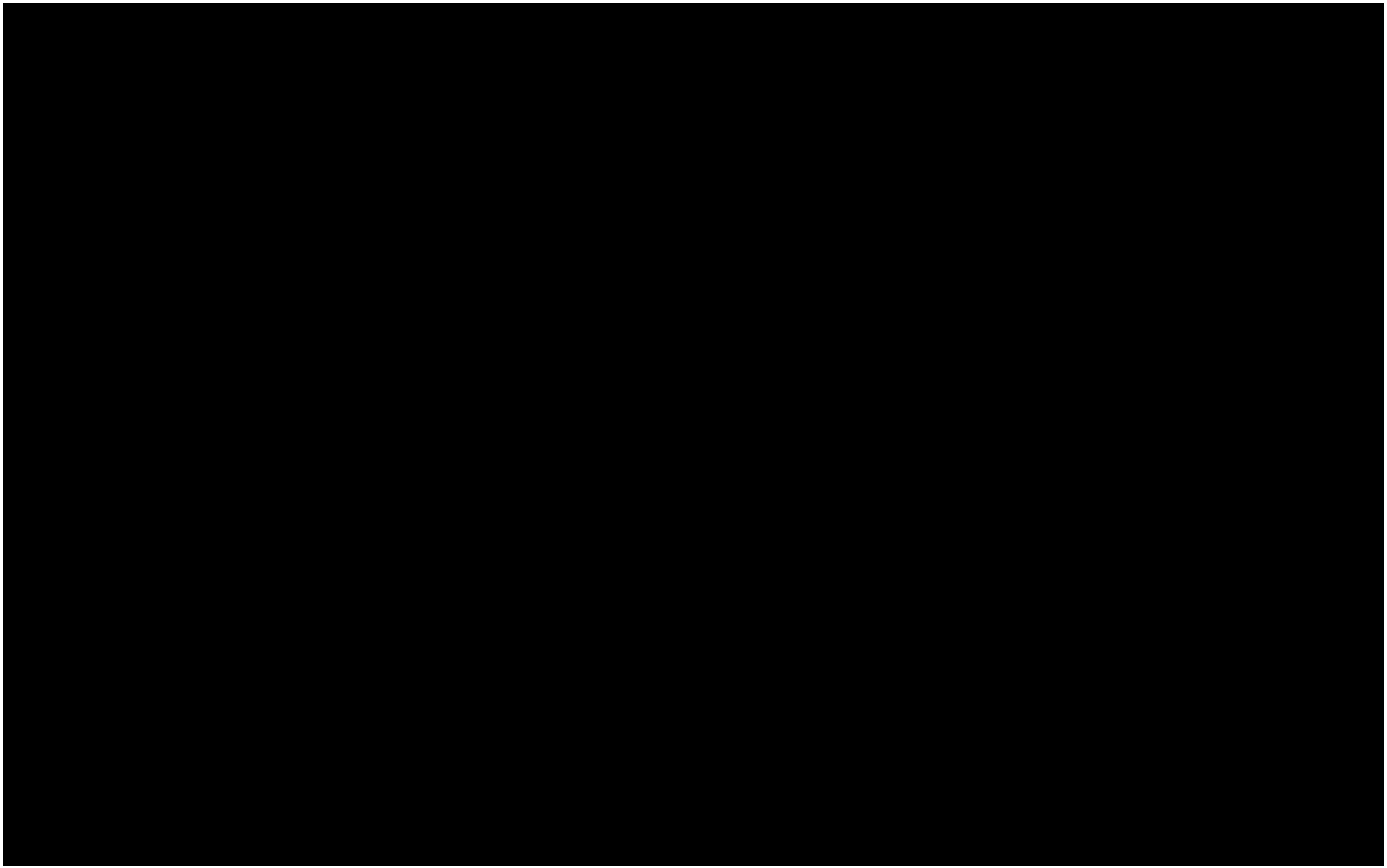
1. Communicate
2. Make changes
3. Write code
4.
5.
6.

6 Habits

1. Communicate
2. Make changes
3. Write code
4. Review code
5.
6.

Richard P. Feynman

On looking for new physical laws



Guess →

Guess → Compute →

Guess → Compute → Observation

Connection to code reviews?

Code reviews are stories

Guess

Why did you think your new code would work?

Compute

What did you do to test?

Observation

Are you sure?

Failure

Sometimes our first guesses are wrong

Code reviews are stories

Build a shared understanding

6 Habits

1. Communicate
2. Make changes
3. Write code
4. Review code
5.
6.

6 Habits

1. Communicate
2. Make changes
3. Write code
4. Review code
5. Test code
6.

Safari

Every code change can break the internet

Page Load Test (PLT)

Built in. Easy to run. One number.

Layout tests

Easy to add. Easy to run. Test correctness.

Tests the whole program

Unit tests++

Testing gives confidence

Without tests, you're guessing

6 Habits

1. Communicate
2. Make changes
3. Write code
4. Review code
5. Test code
6.

6 Habits

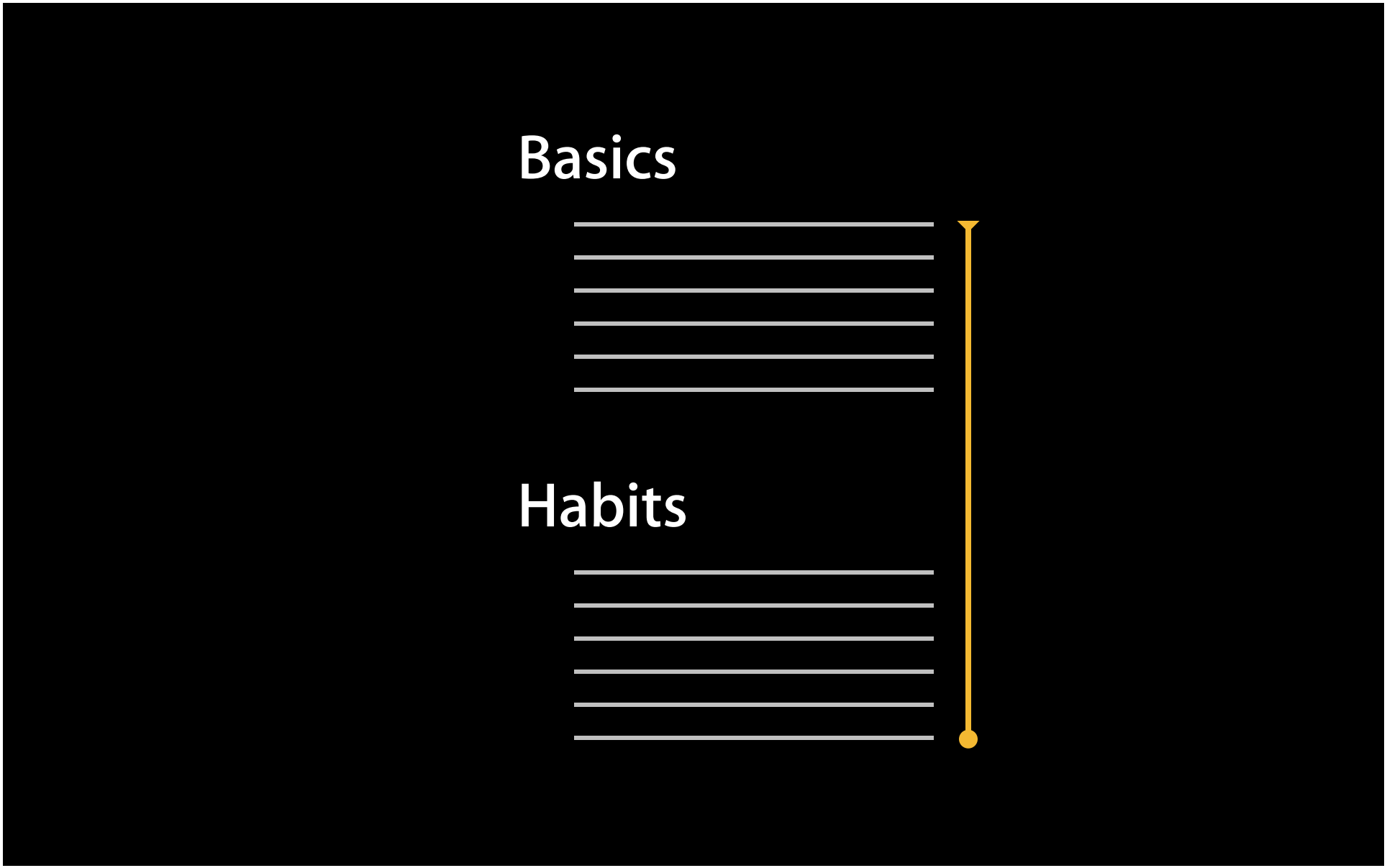
1. Communicate
2. Make changes
3. Write code
4. Review code
5. Test code
6. Reevaluate basics

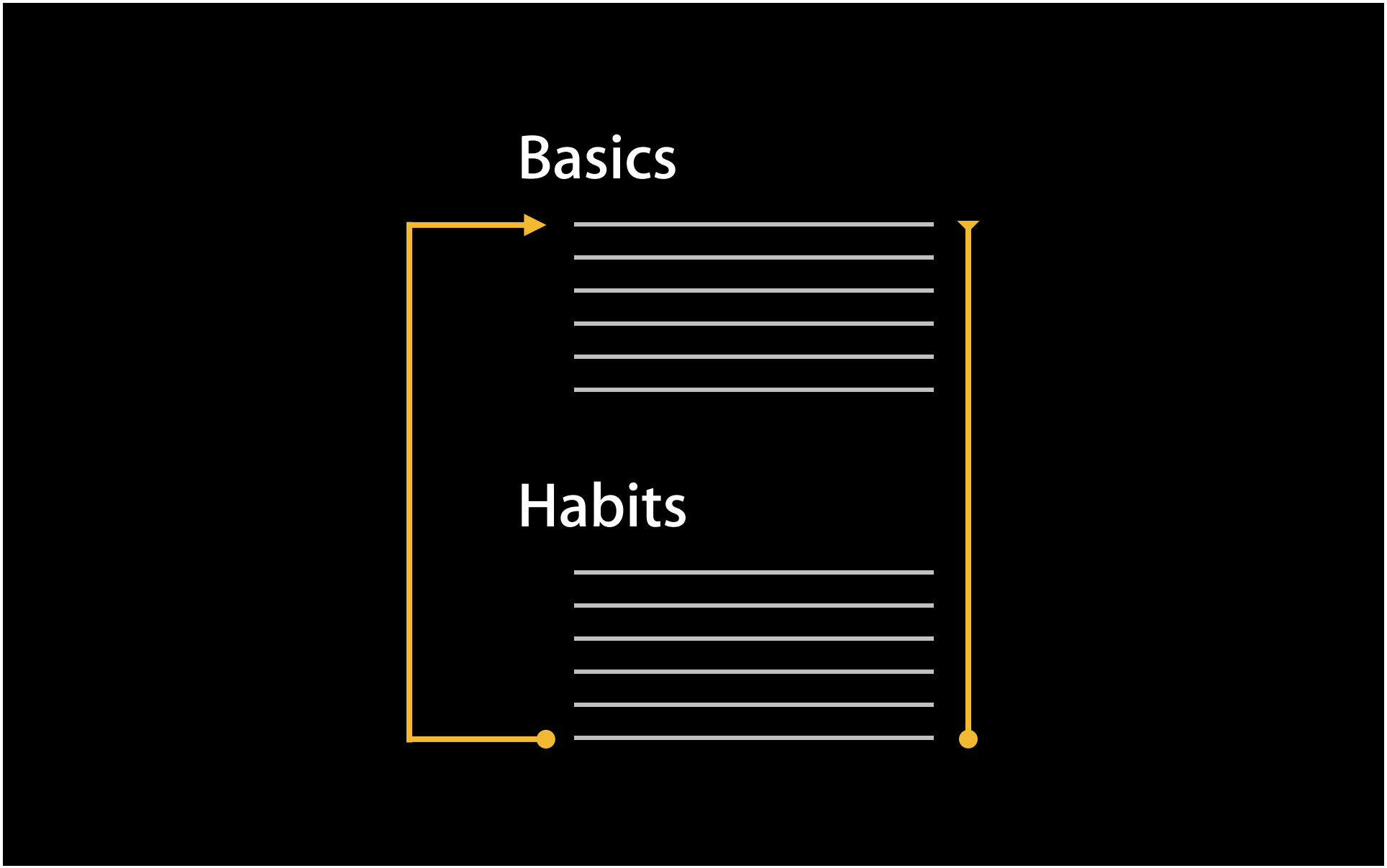
Reevaluate

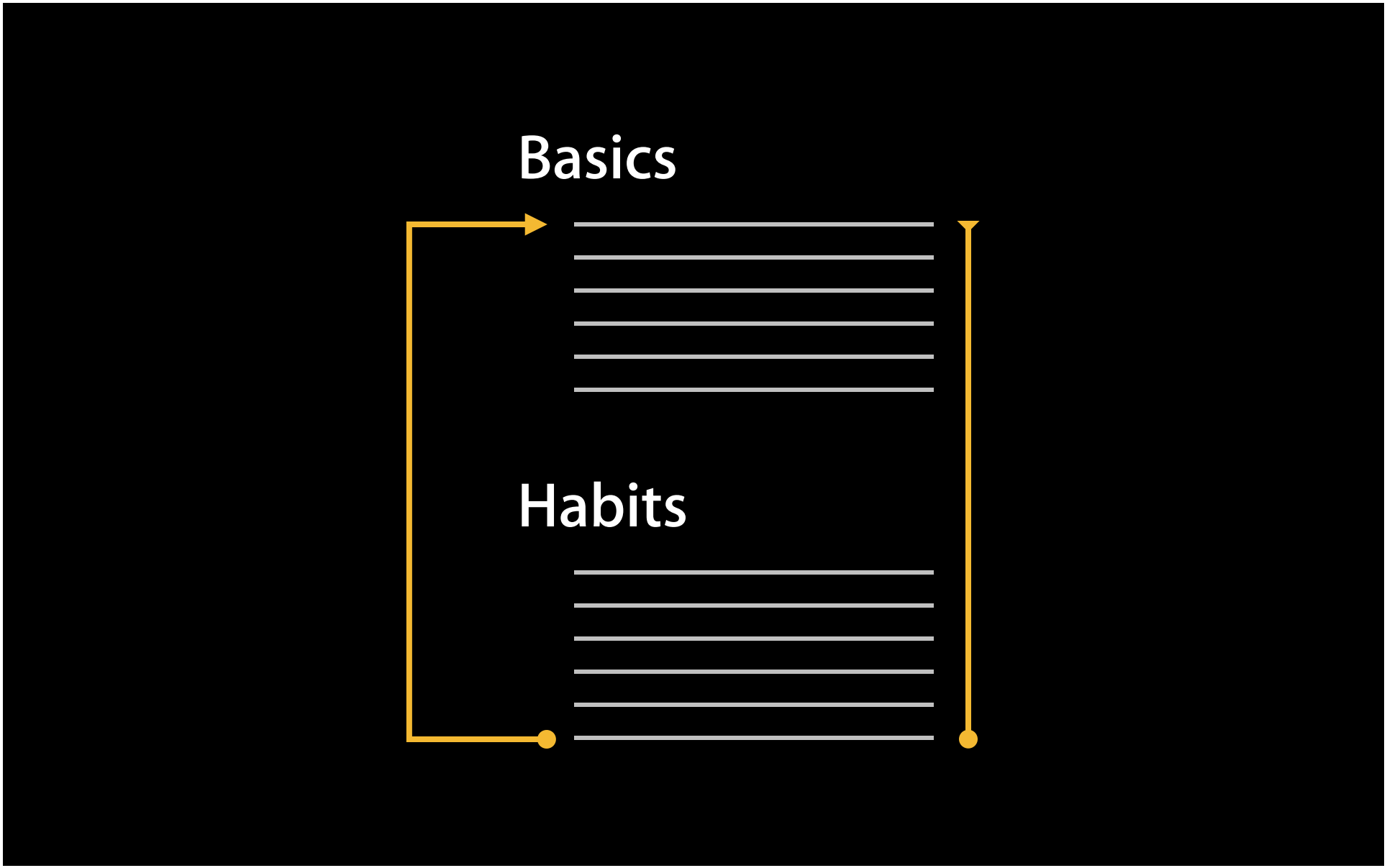
Kick off the feedback loop

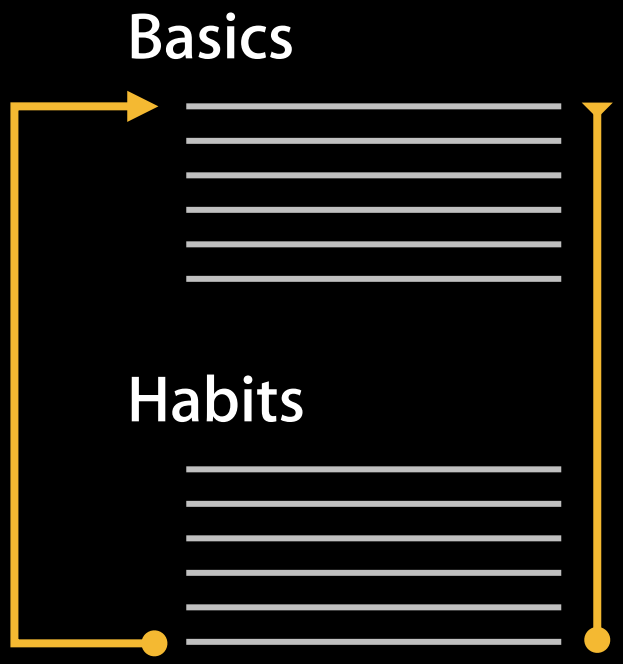
Basics

Habits


Basics
Habits


Basics
Habits


Basics
Habits



A process for long-term change

6 Habits

1. Communicate
2. Make changes
3. Write code
4. Review code
5. Test code
6. Reevaluate basics

Basics

Fundamental choices you make

Habits

Things you do every day

Basics + Habits

Building your software projects to last


WWDC2012