Frameworks

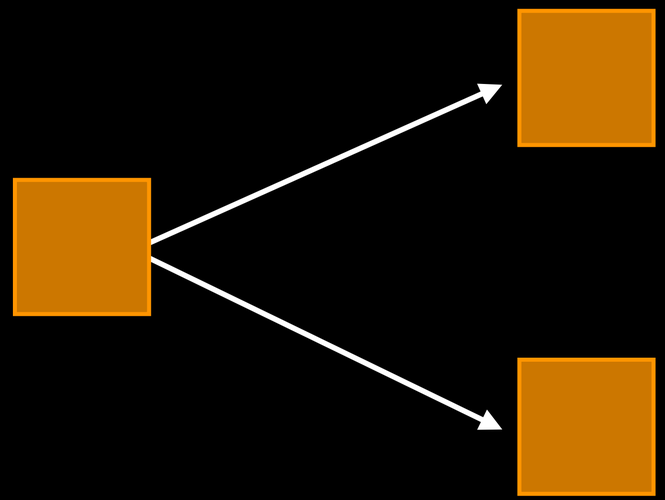
#WWDC14

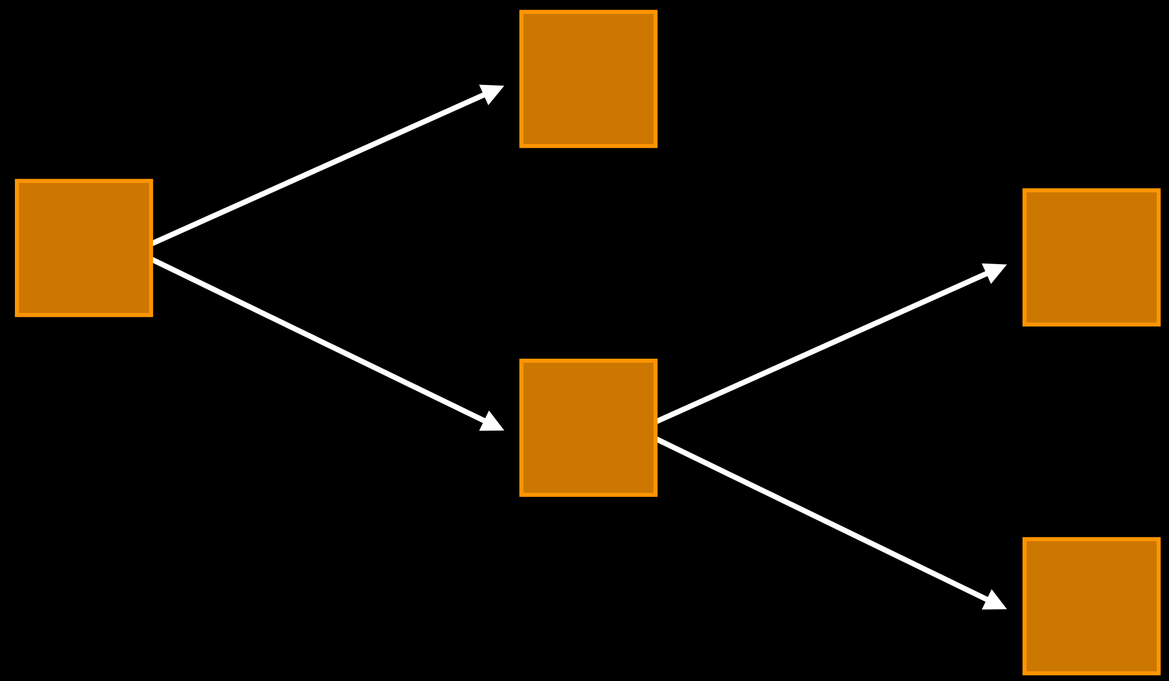
Advanced iOS Application Architecture and Patterns

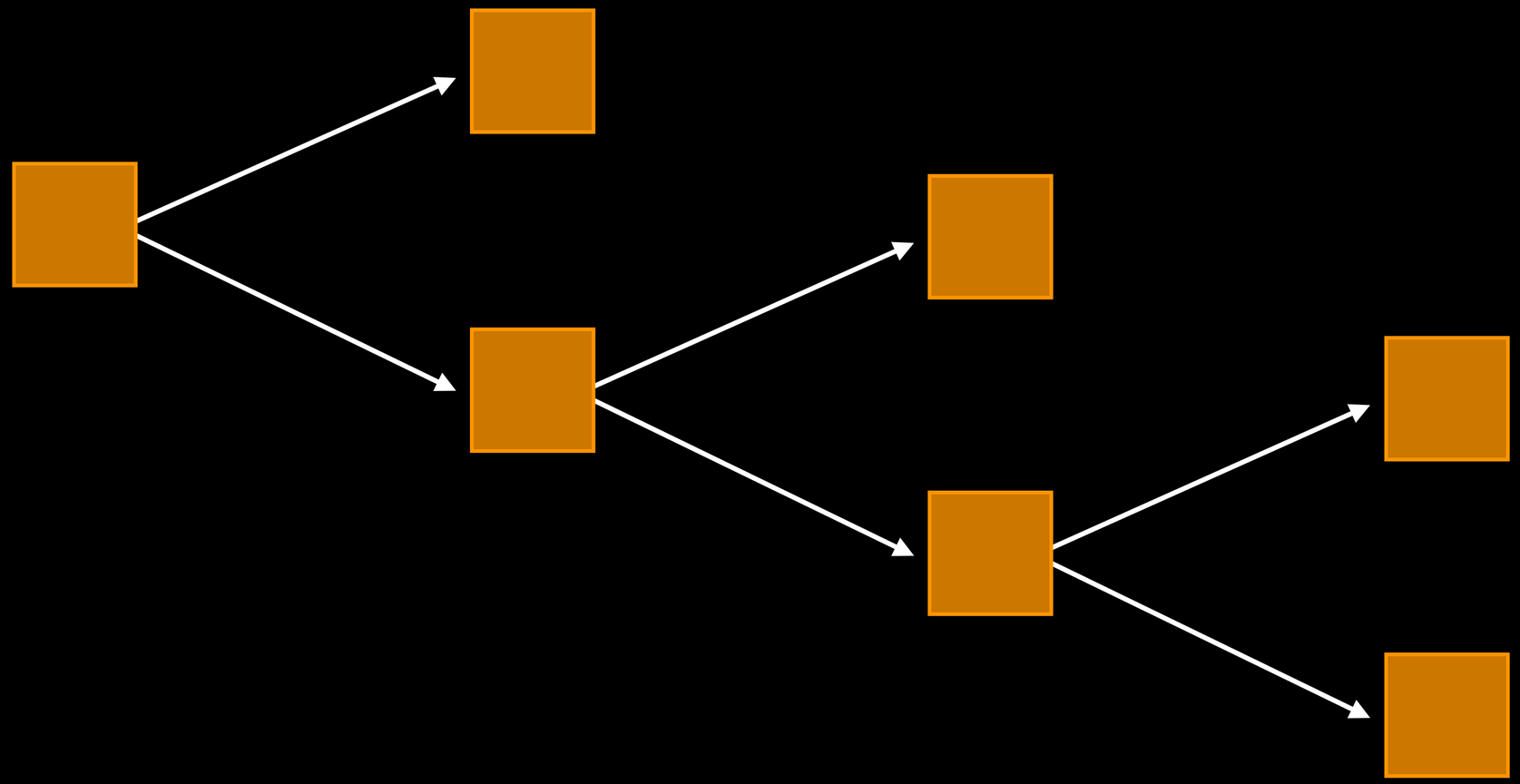
Session 229

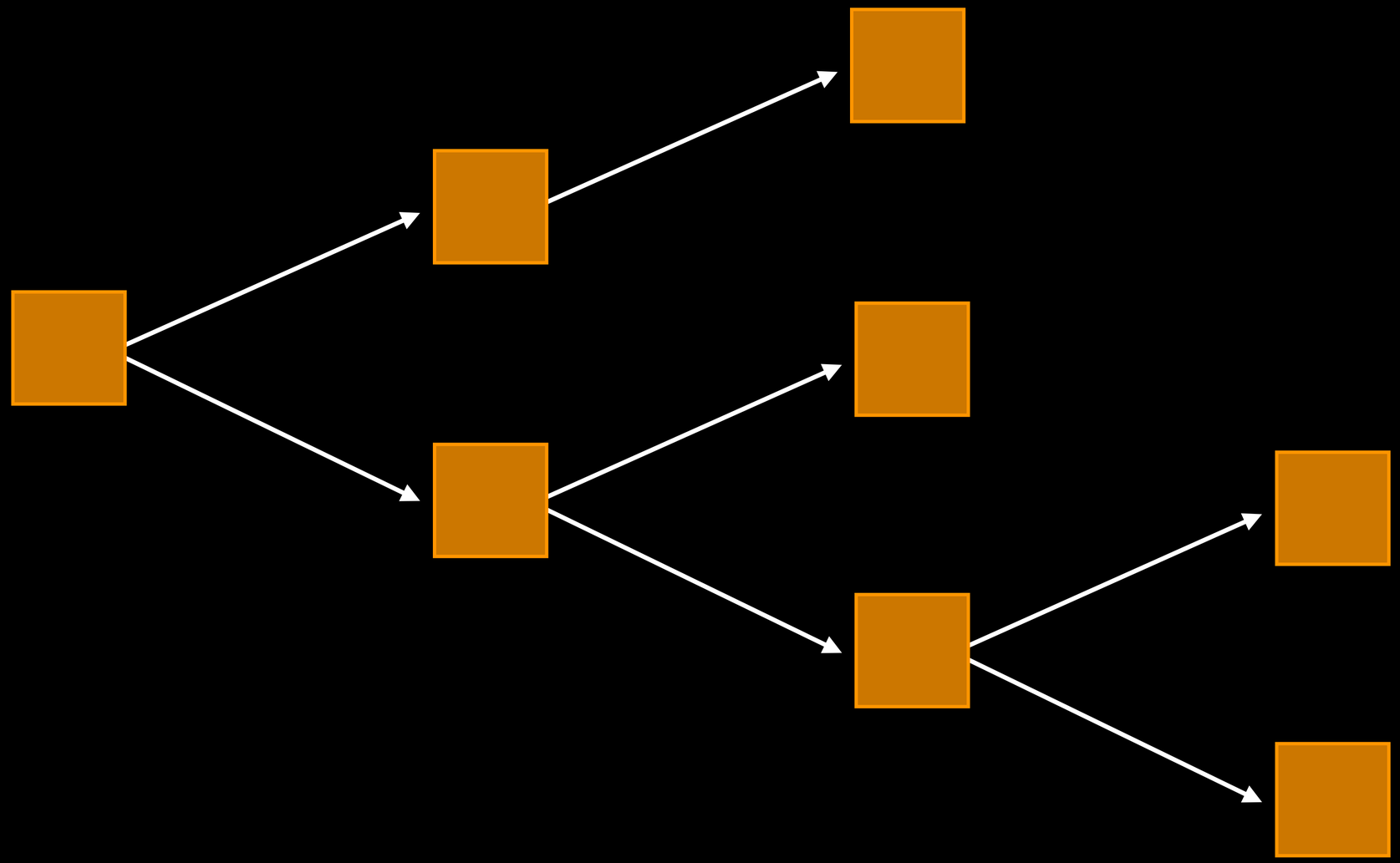
Andy Matuschak
iOS Apps and Frameworks

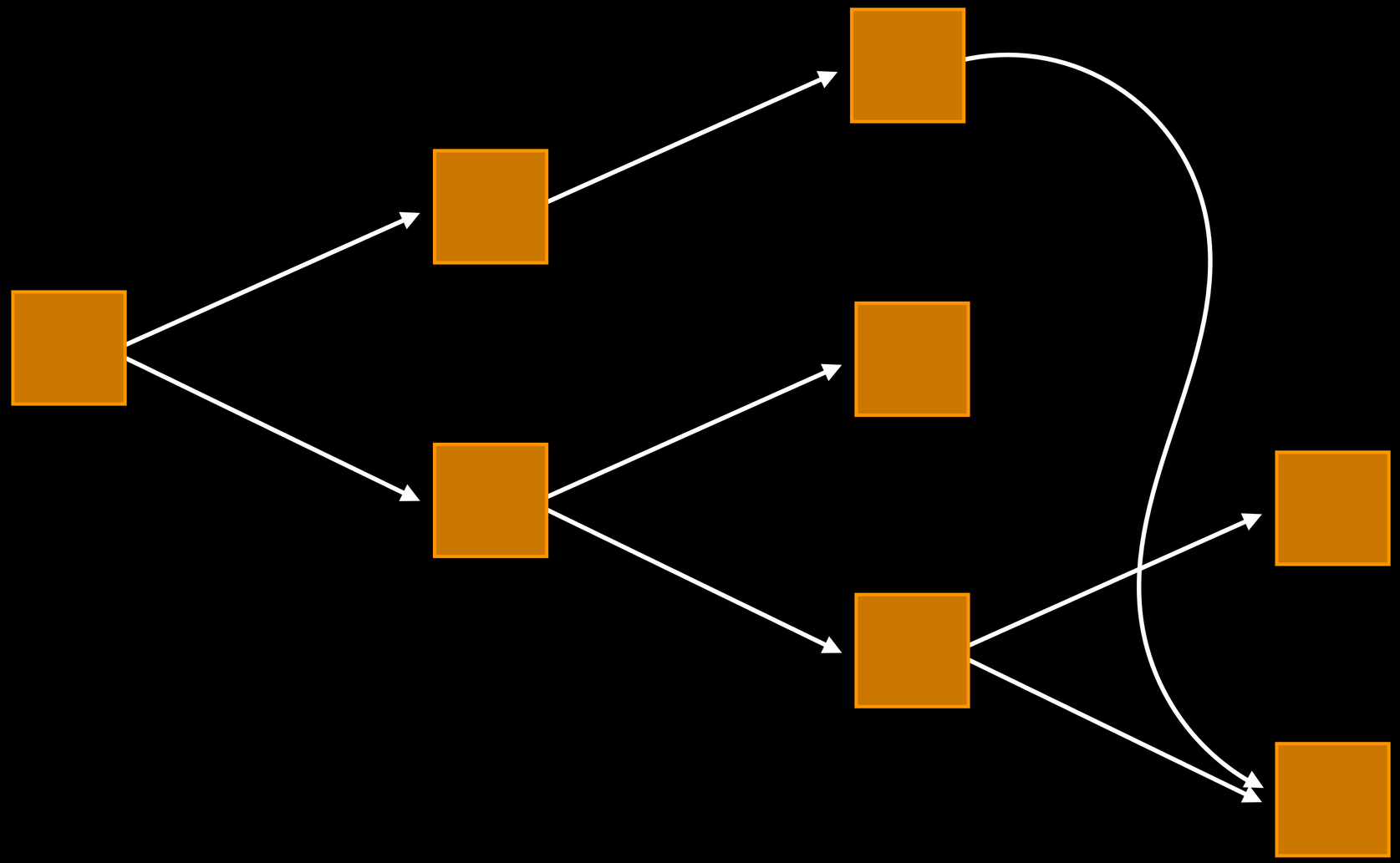
Colin Barrett
iOS Apps and Frameworks

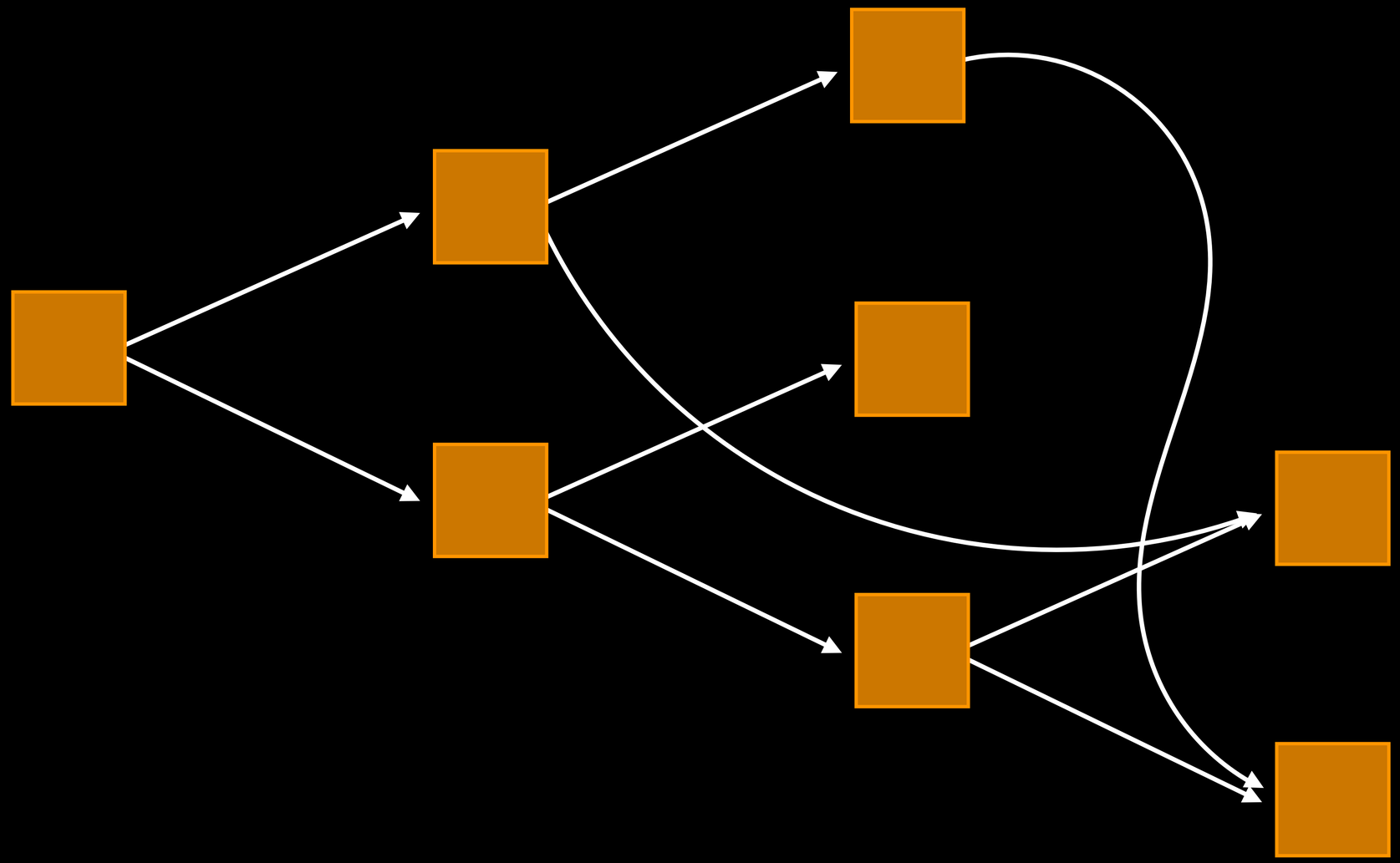


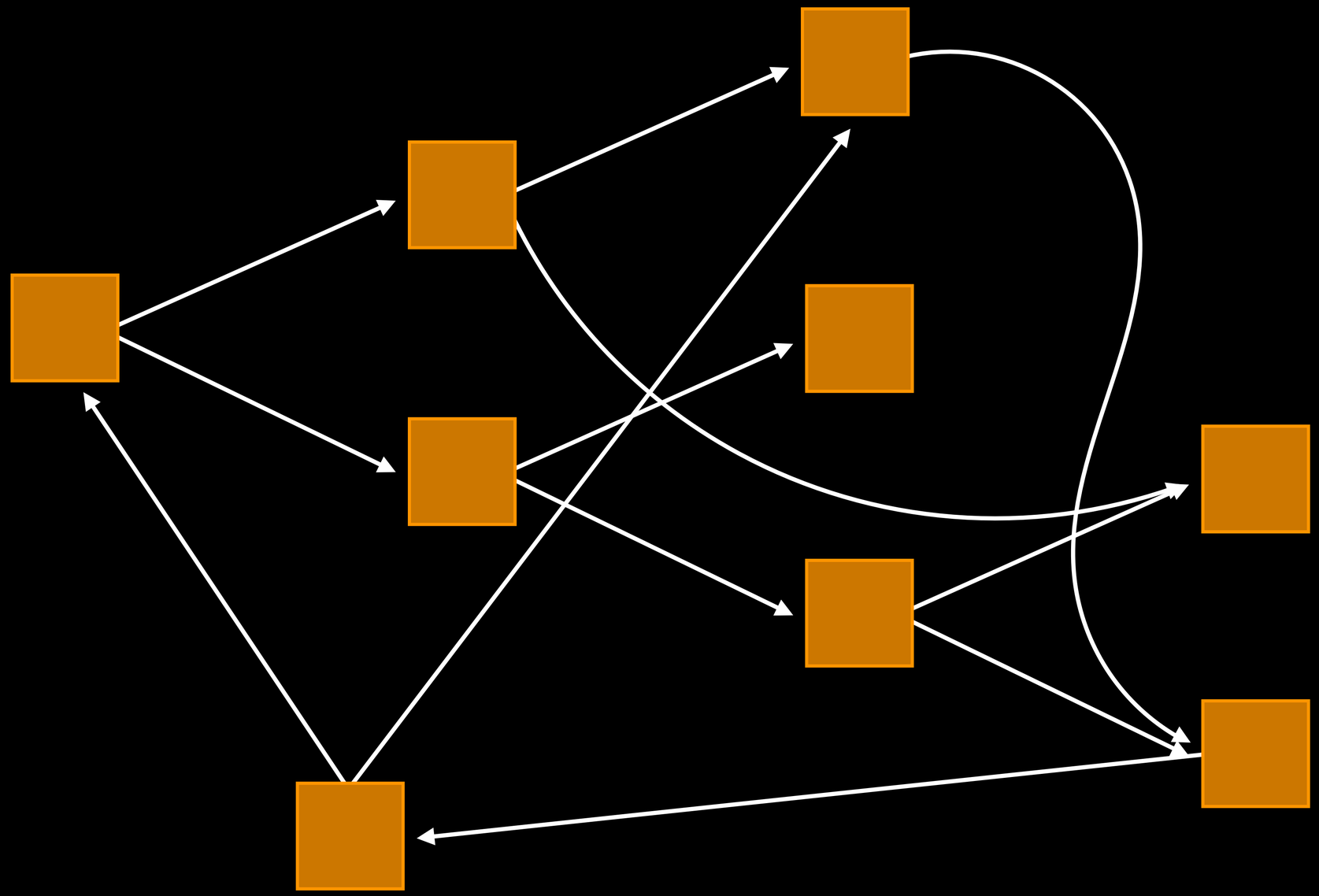


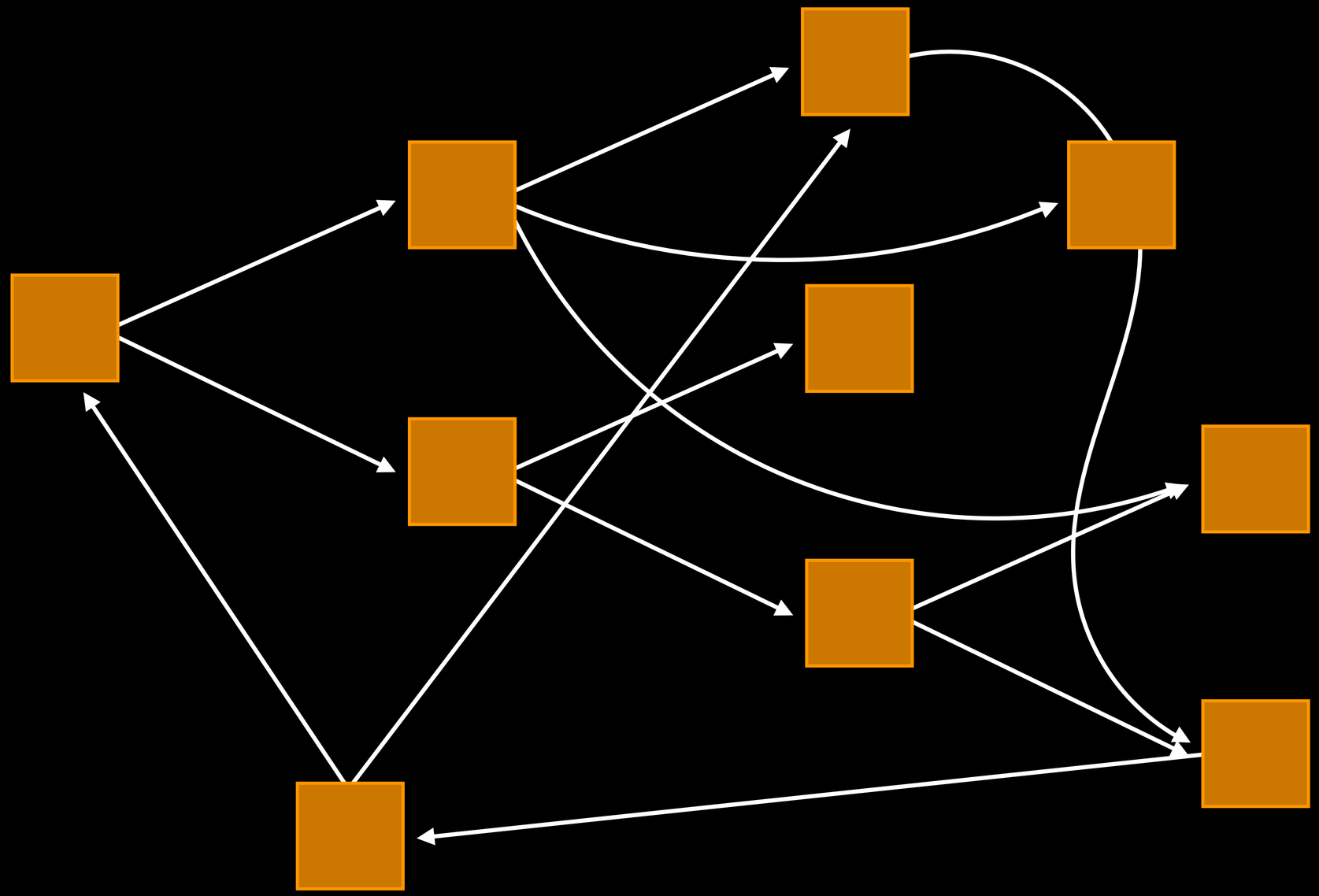


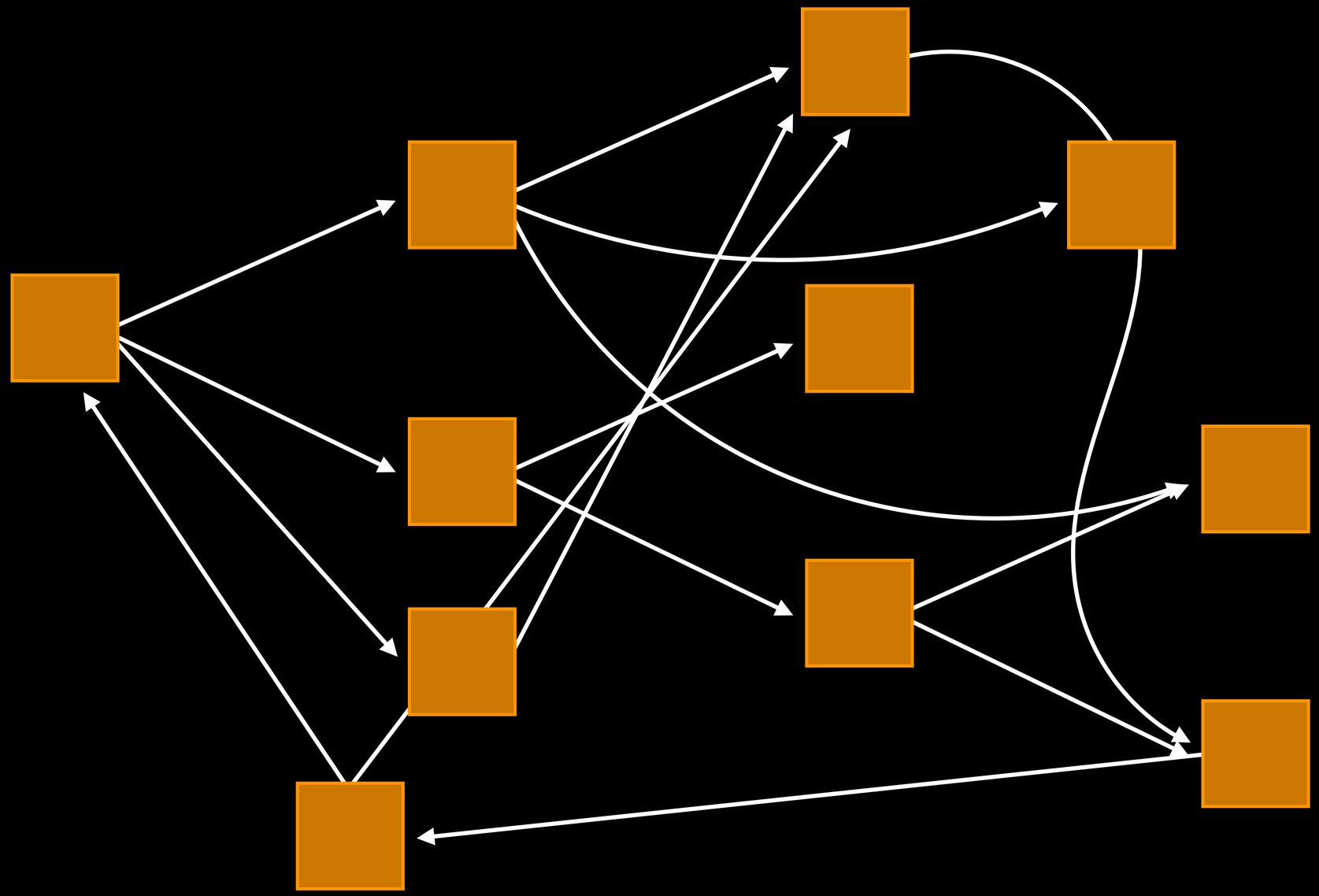


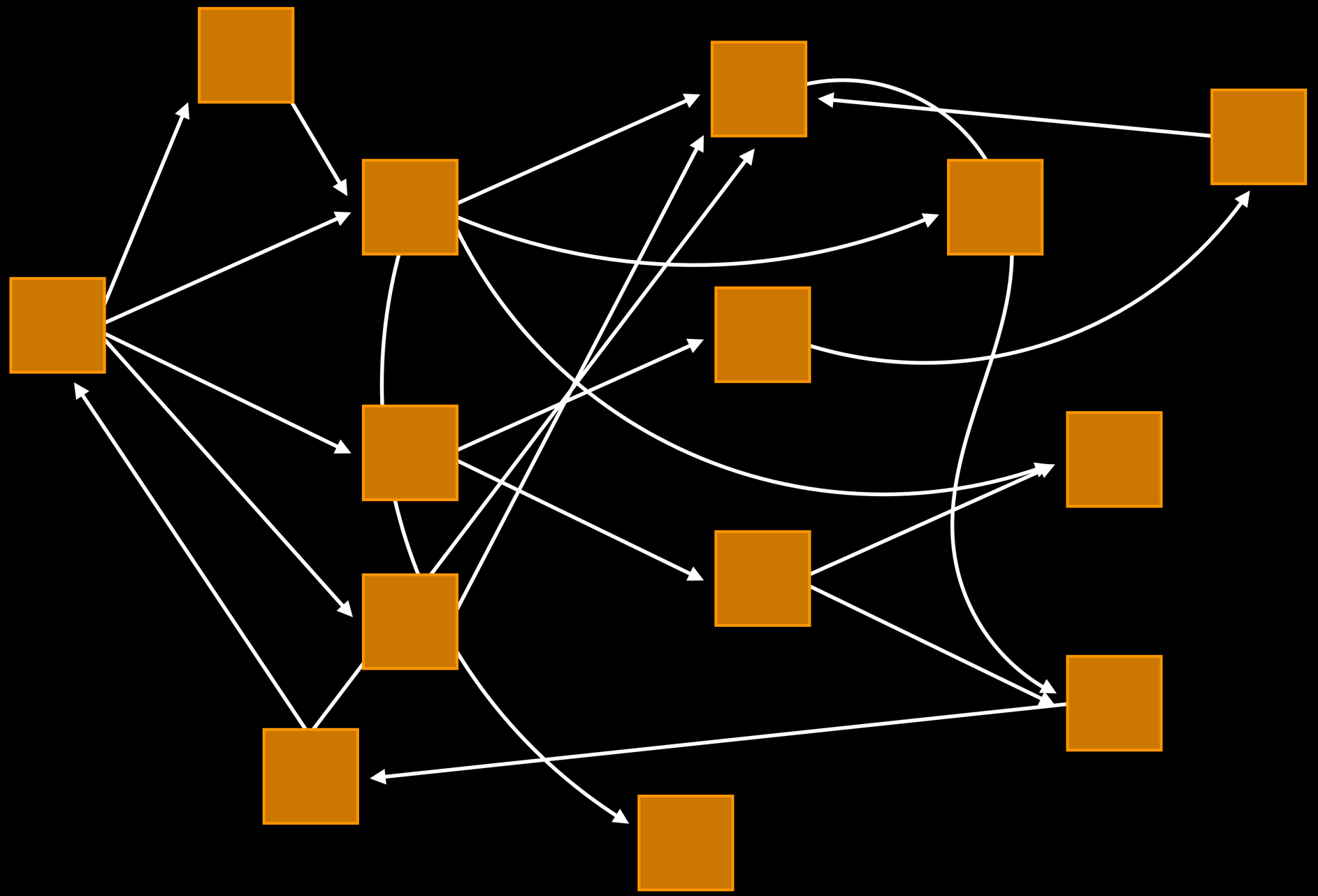


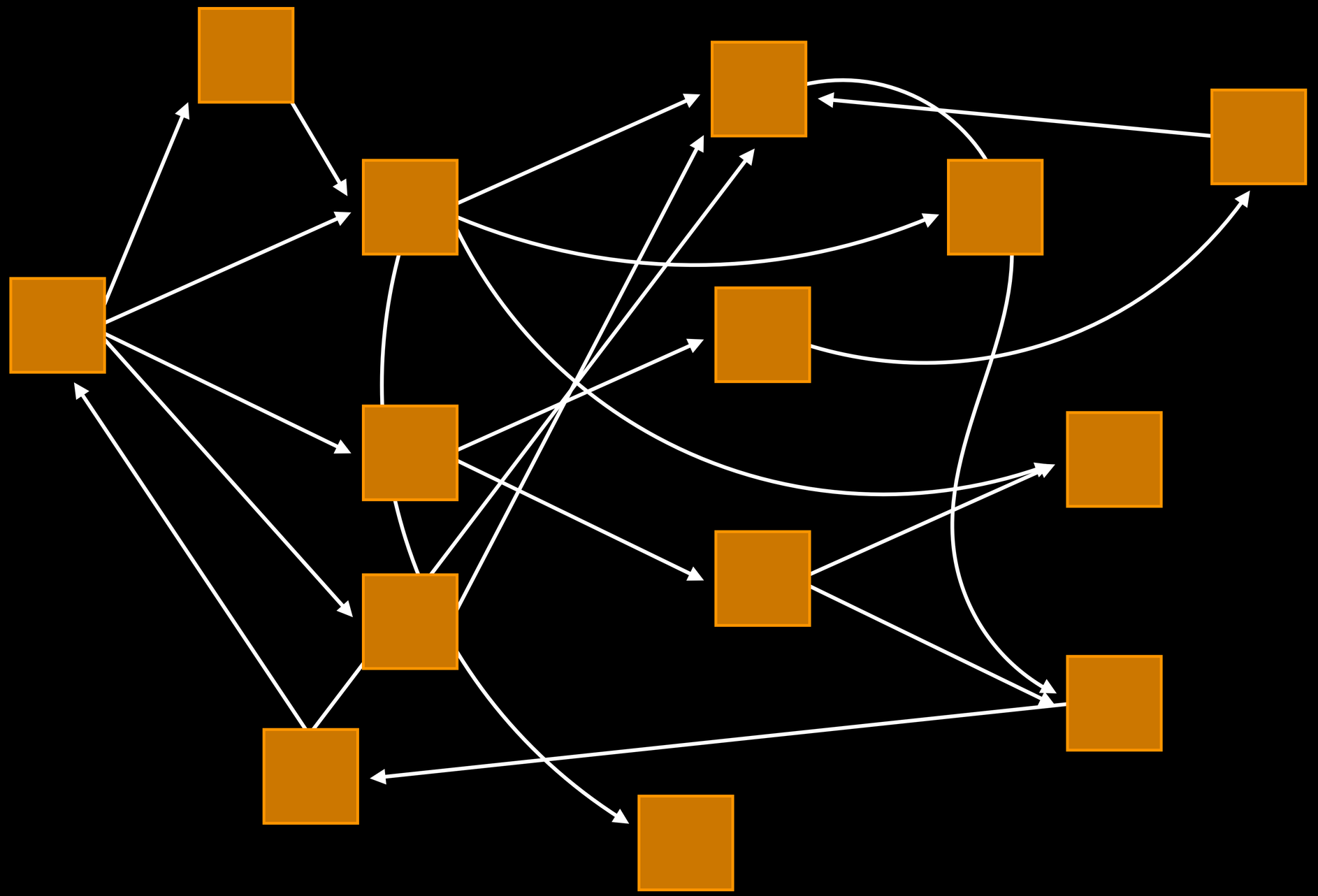


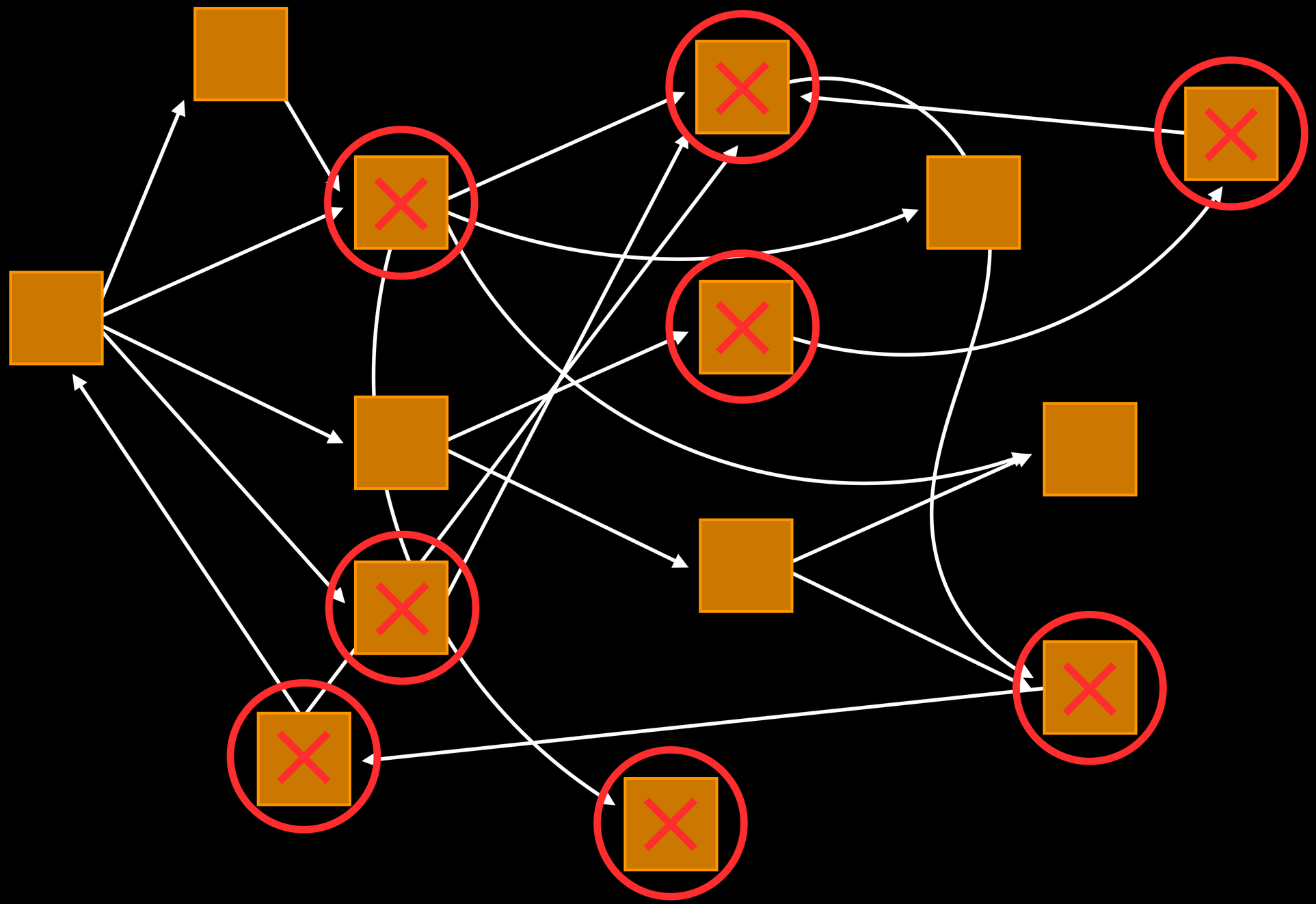


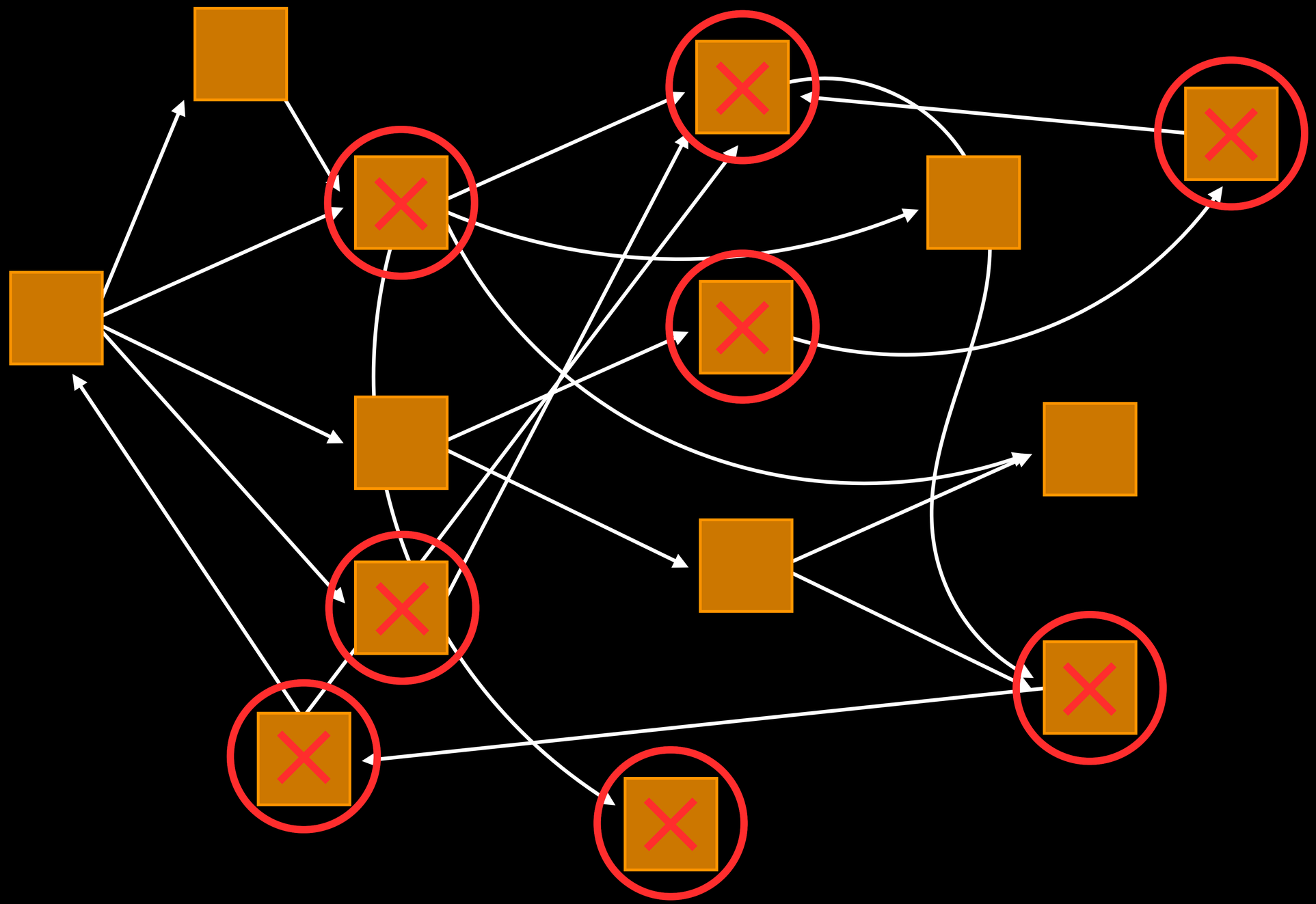


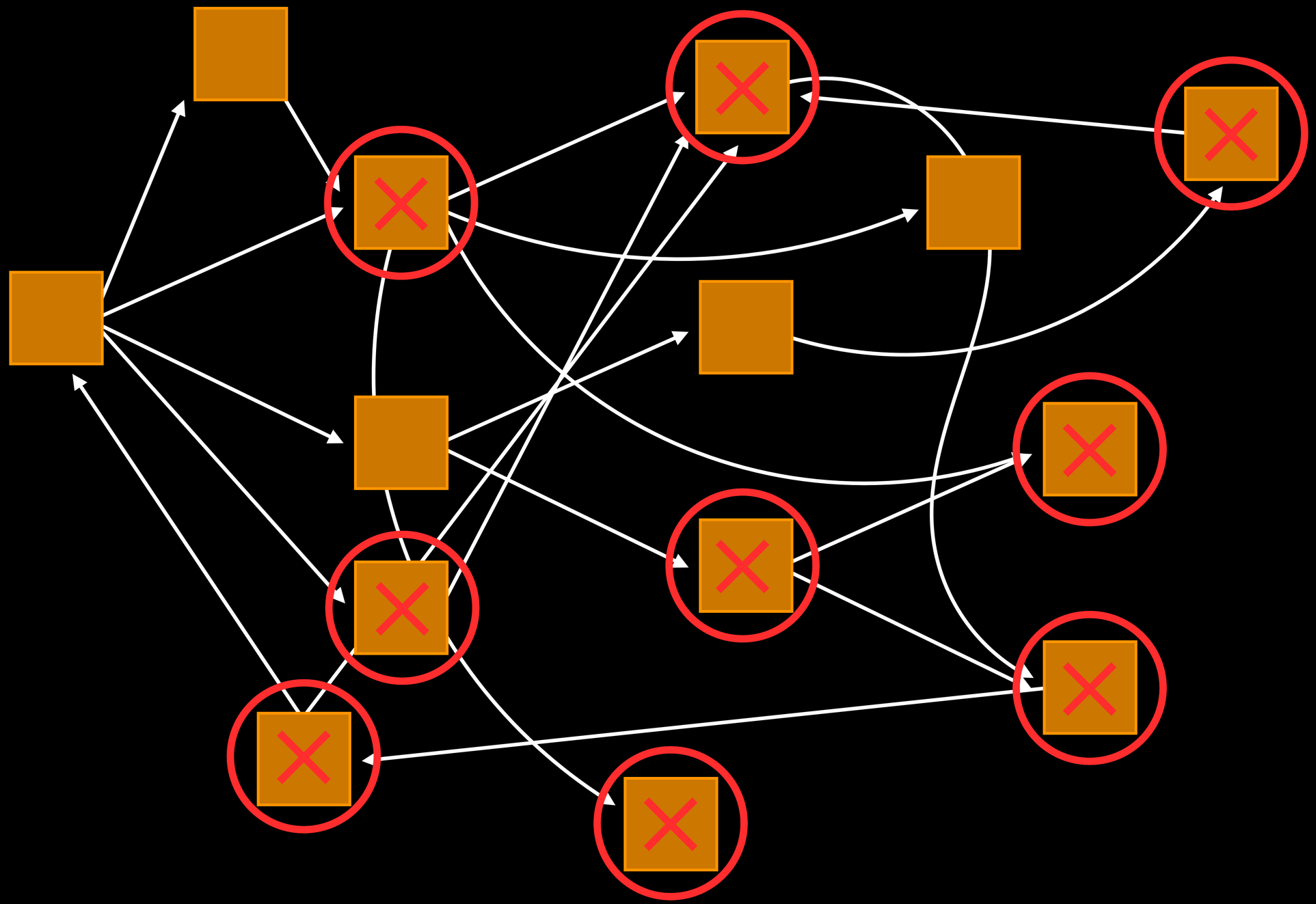


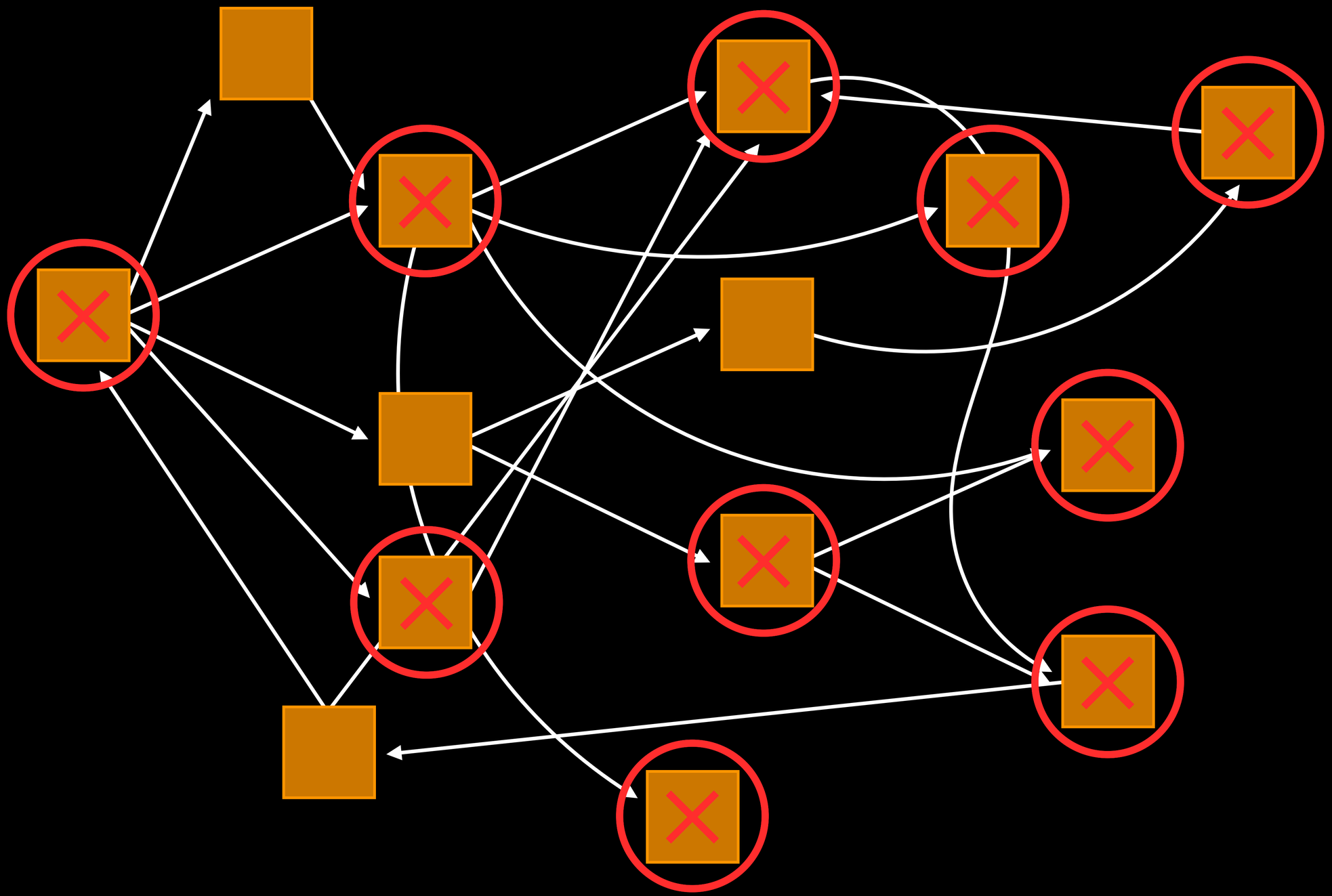


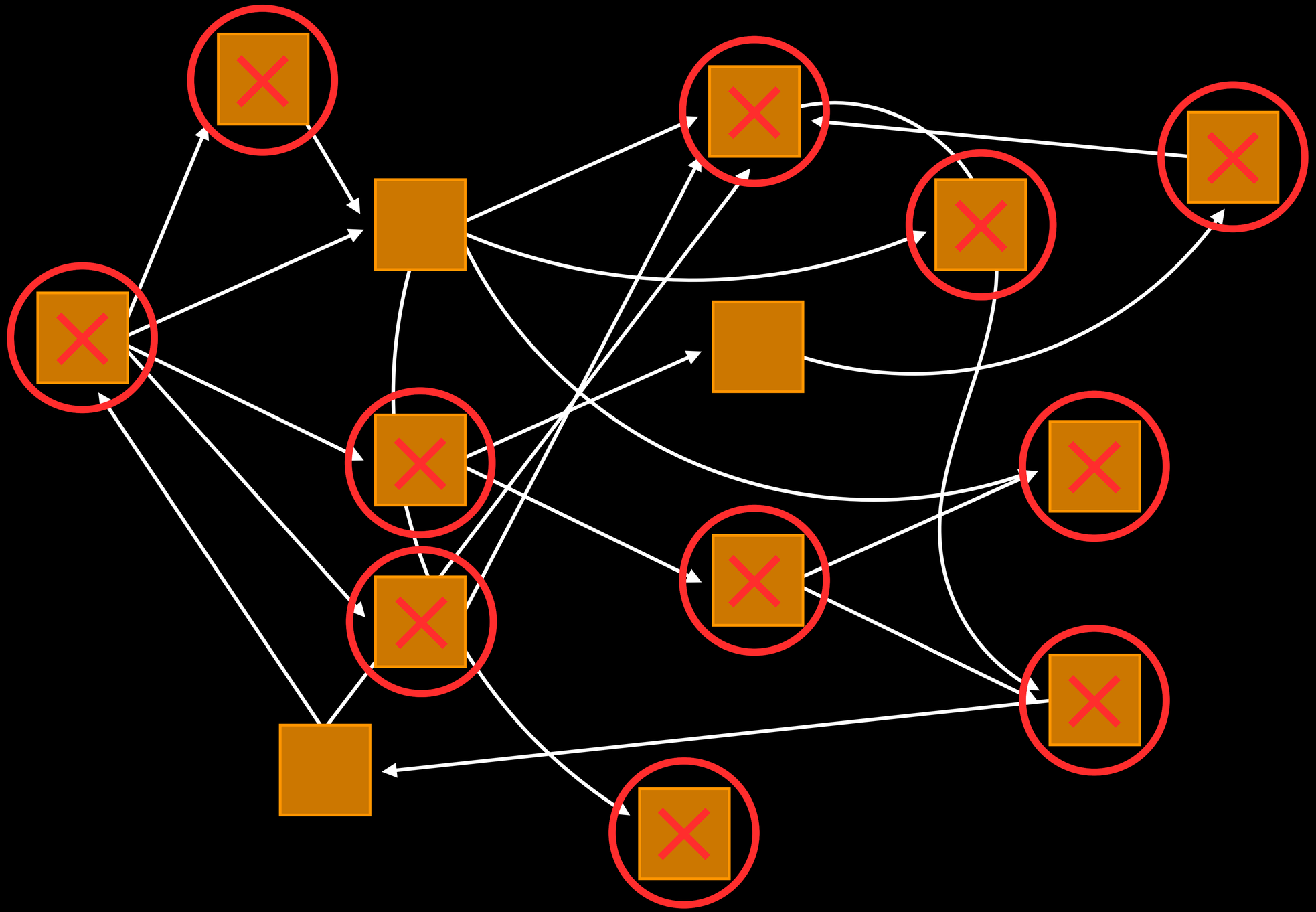


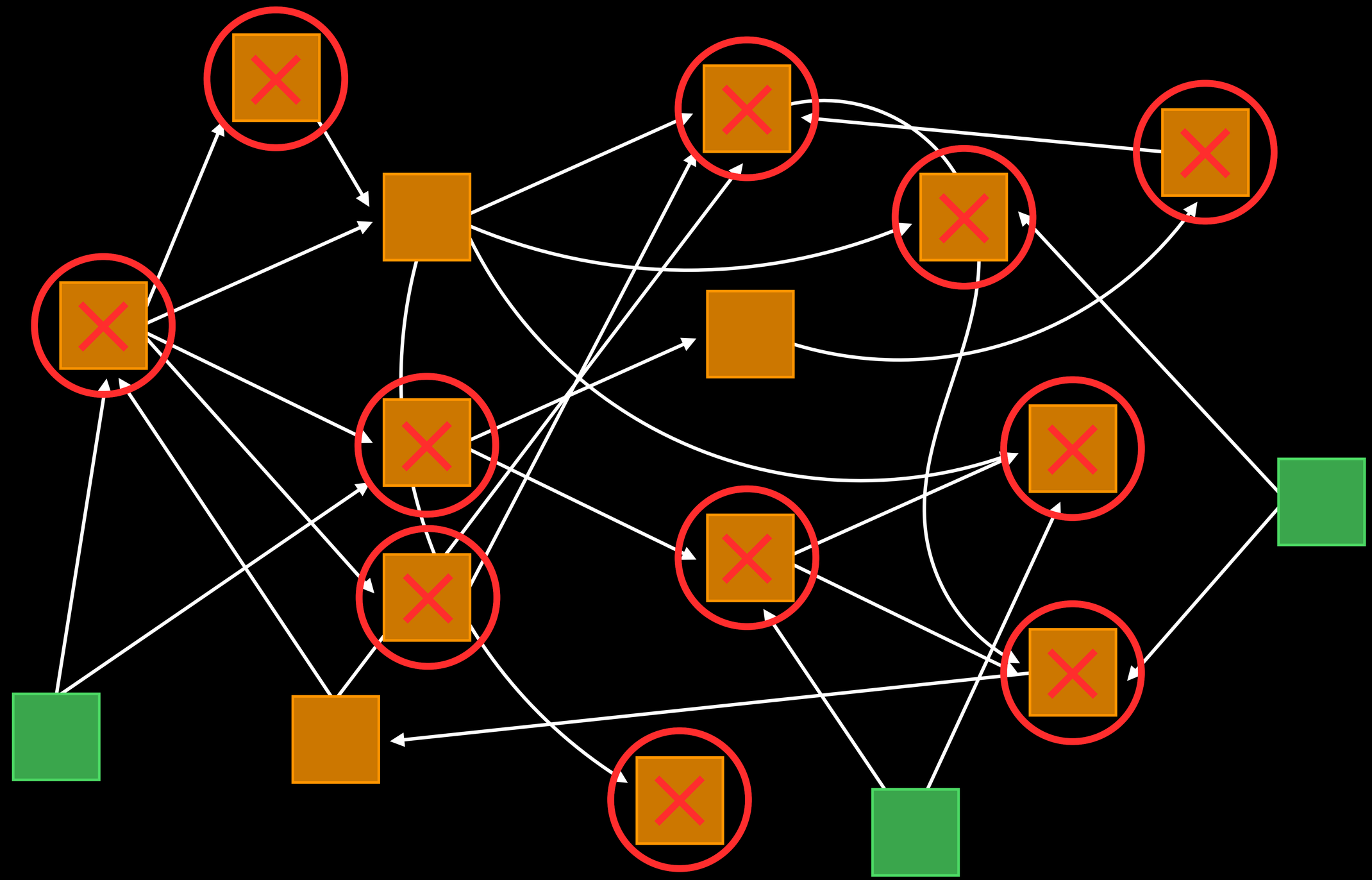


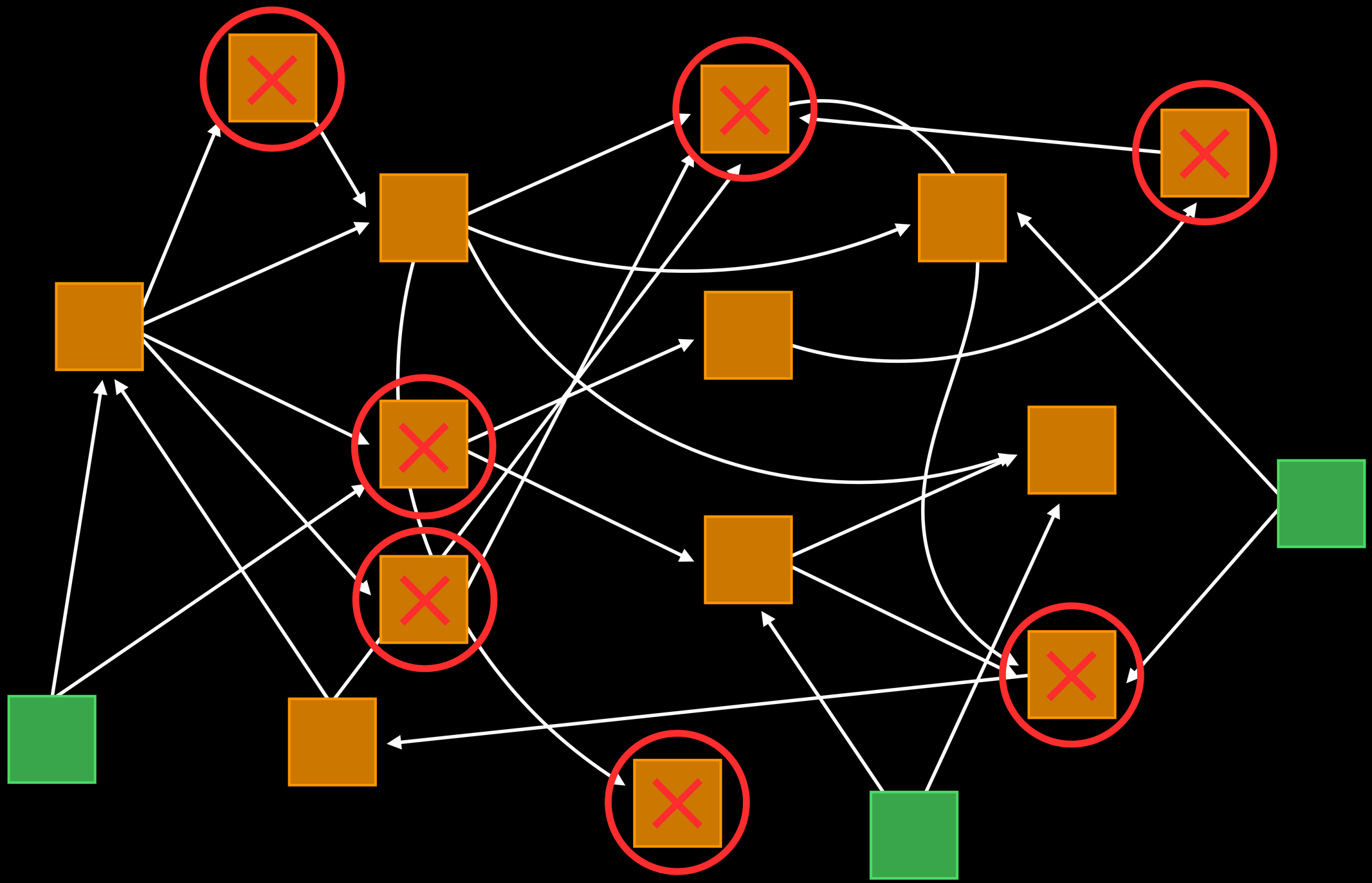


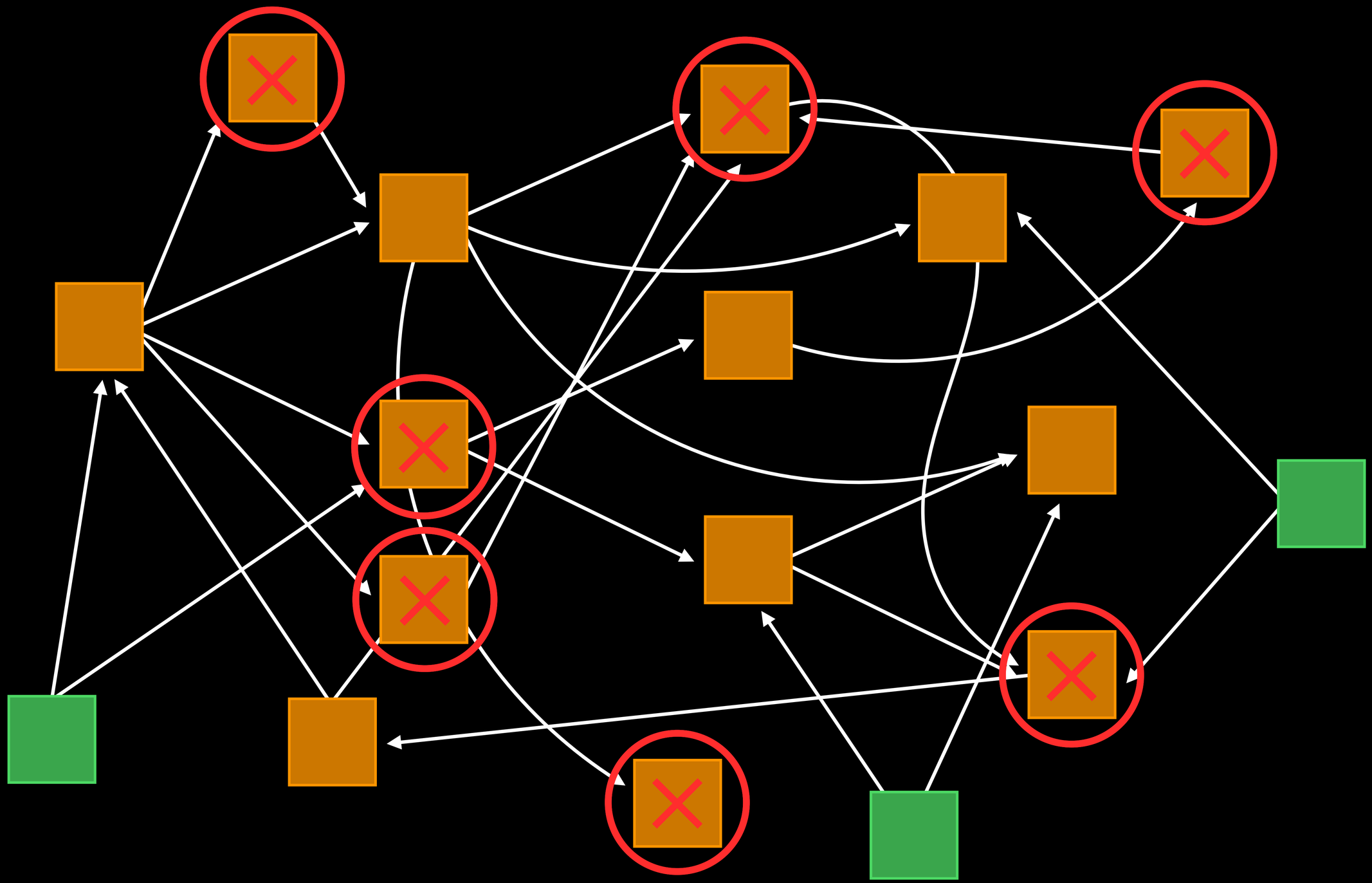


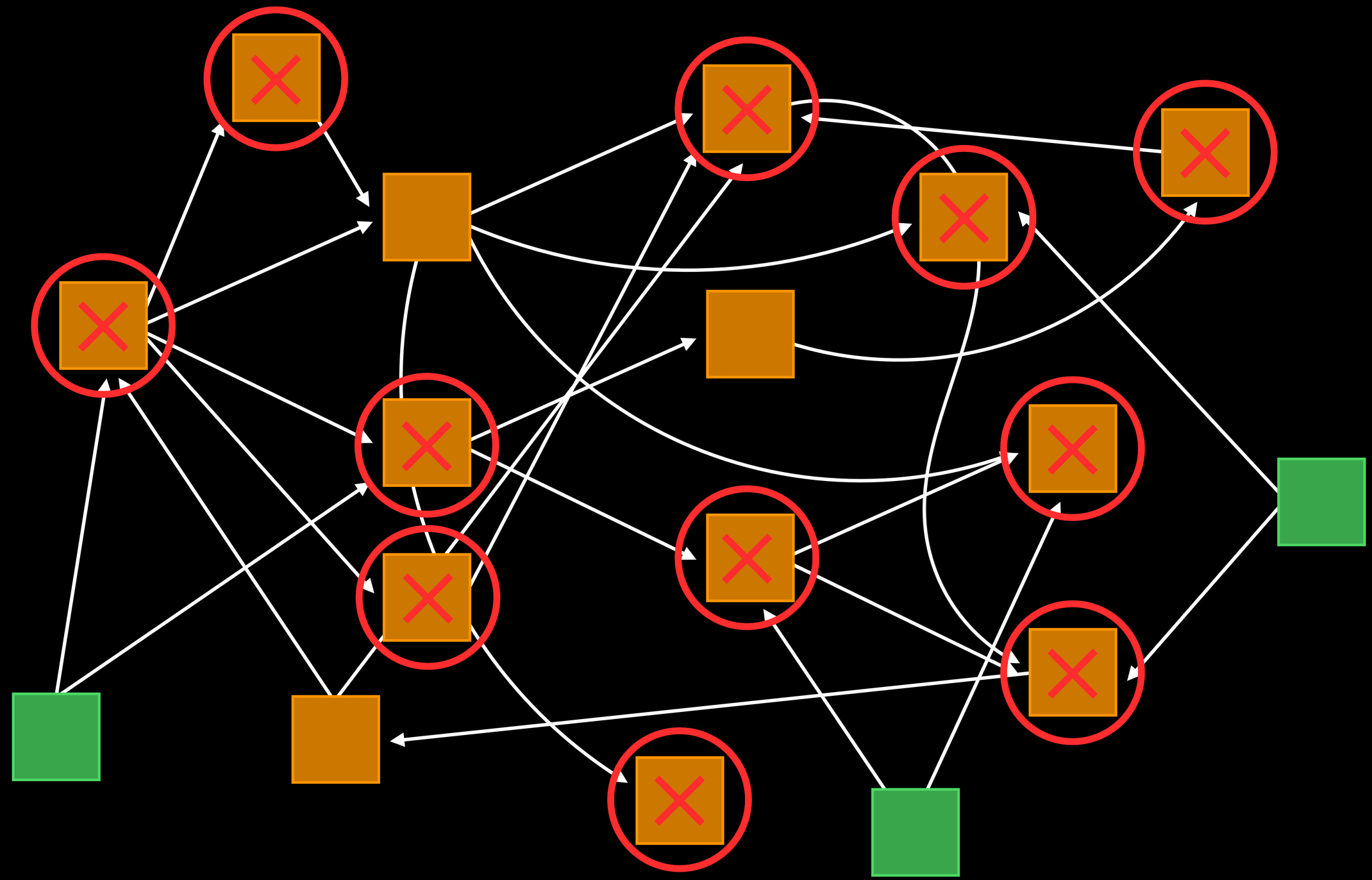


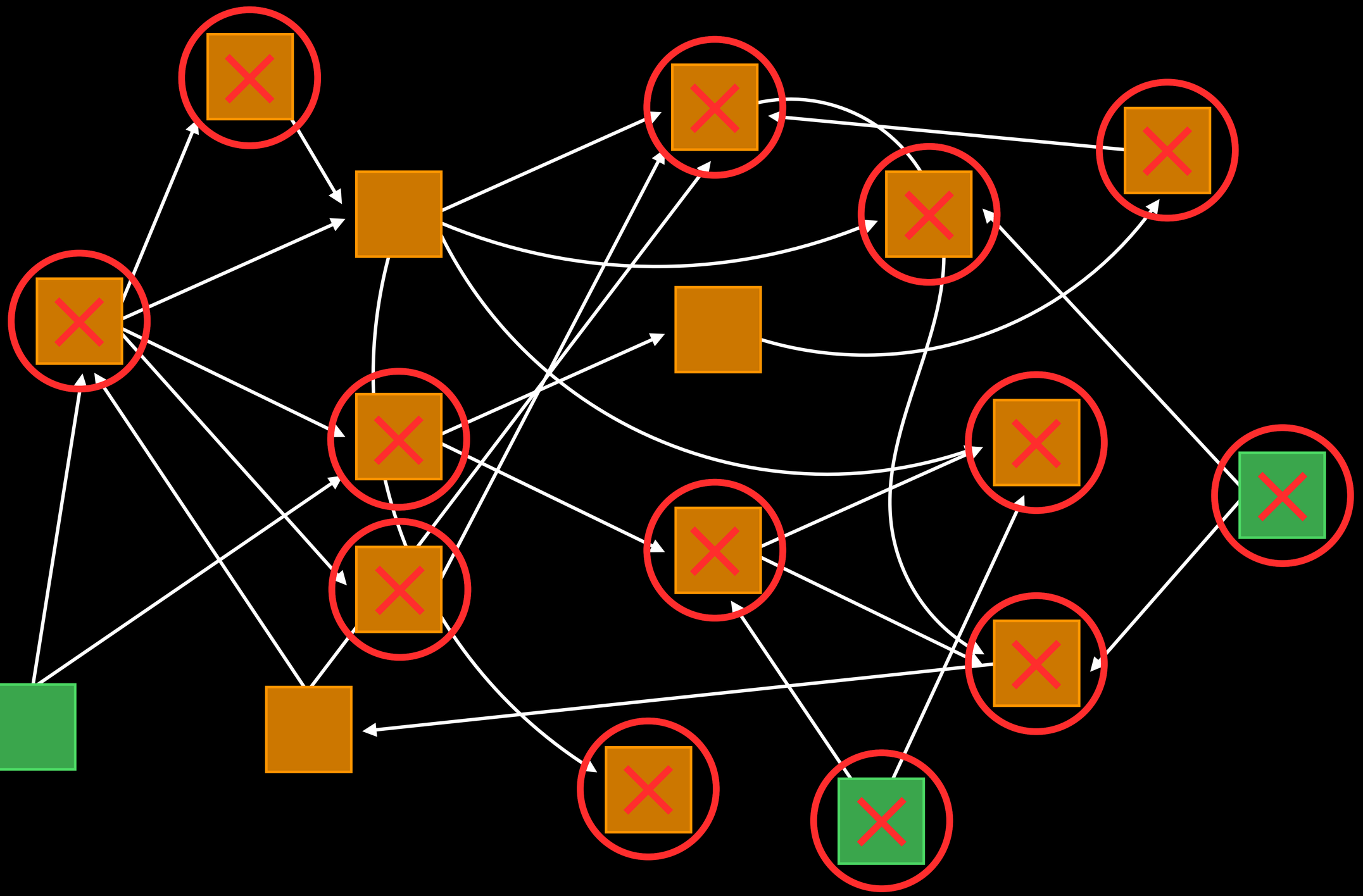












Software Architecture

Software Architecture...?

Five Things to Never Do in Your App

It’s software architecture! We promise!

1. Don’t ever use class names explicitly
2. Avoid bare C functions with singletons
3. Why are you reading this?
4. It’s a joke; these are bad ideas
5. Or maybe just dogmatic ideas

“You Ain’t Gonna Need It!”

“You Ain’t Gonna Need It!”

vs.

“Big Design Up Front.”

taste

architecture

ability

taste

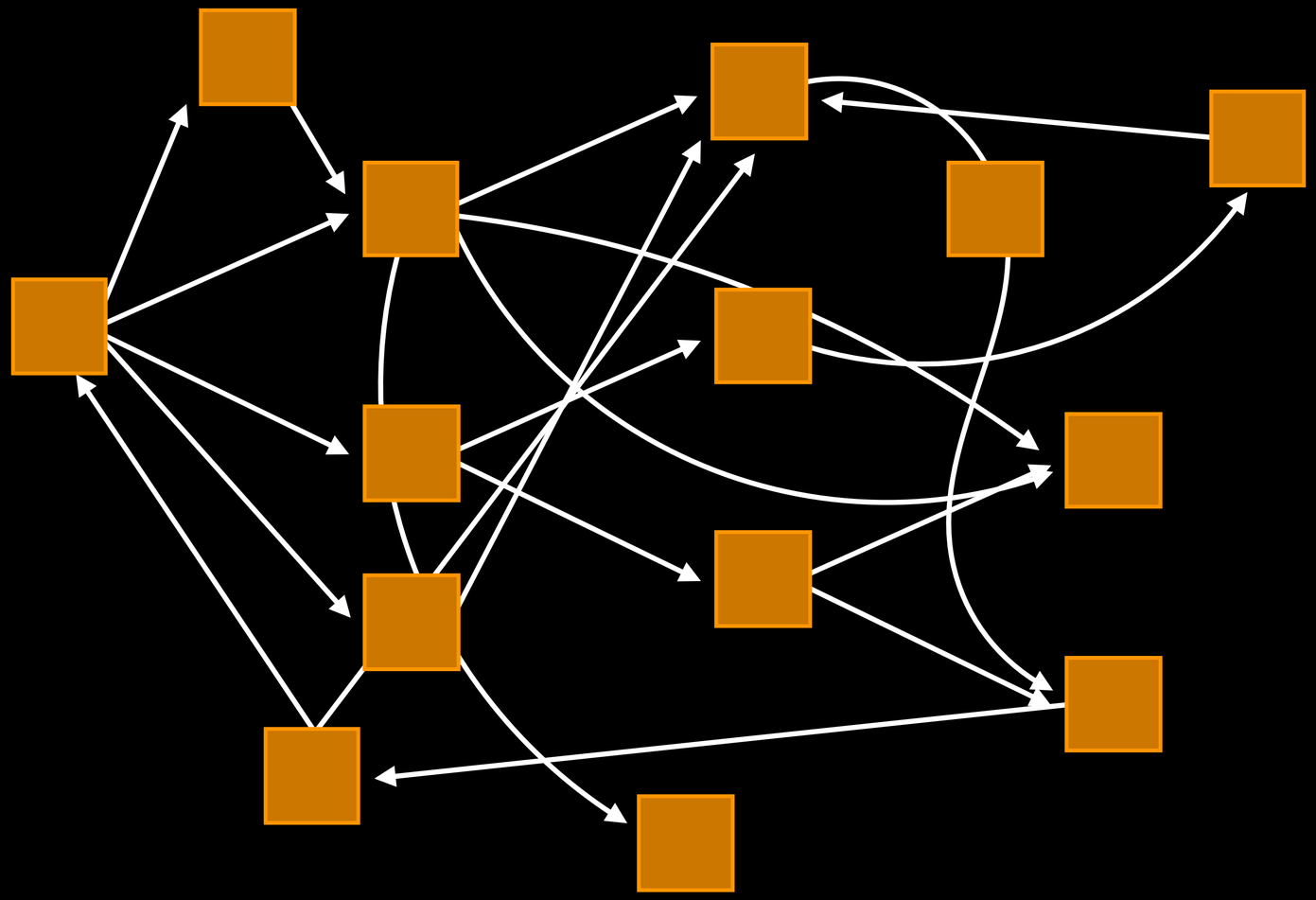
architecture

ability

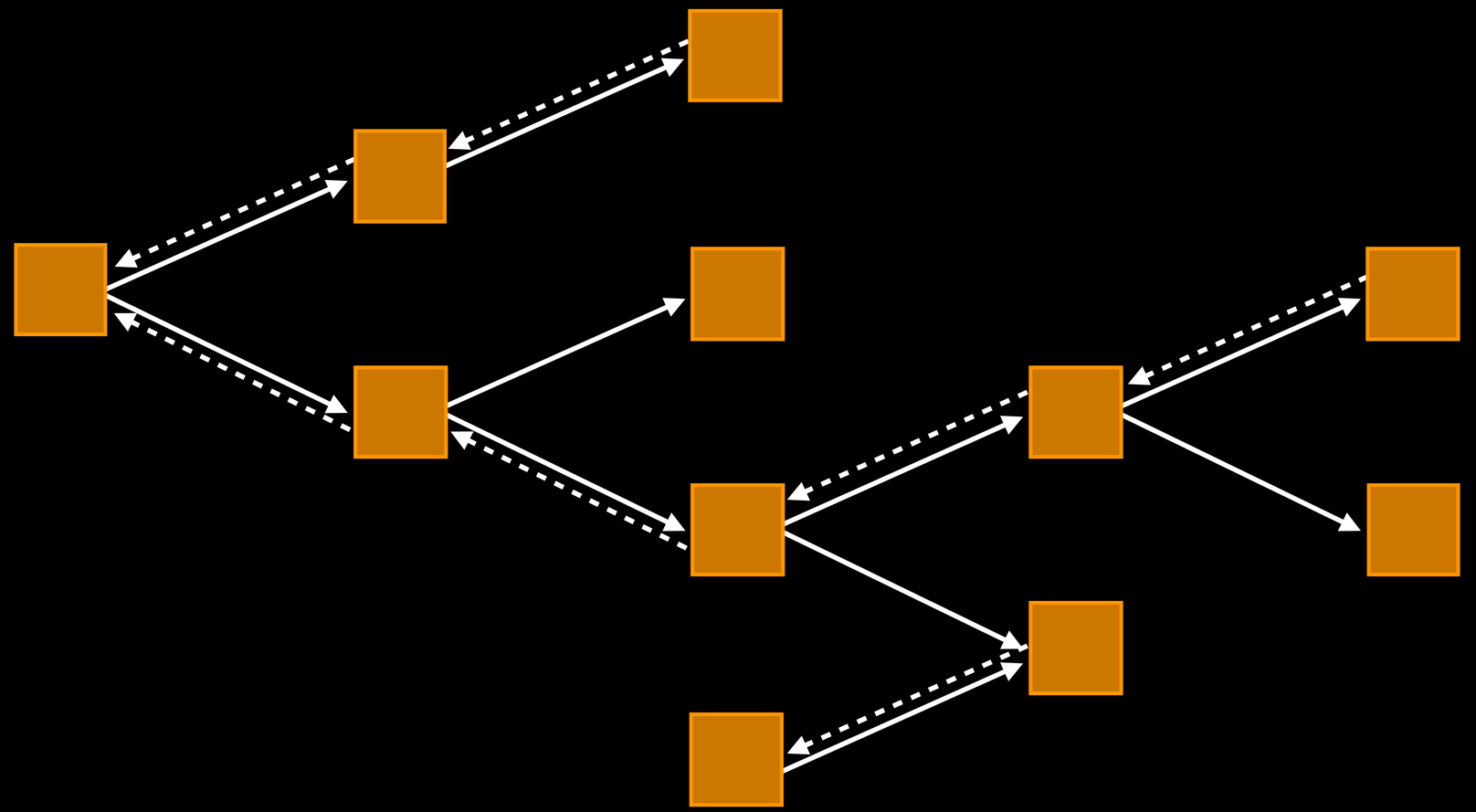
Insight, not dogma

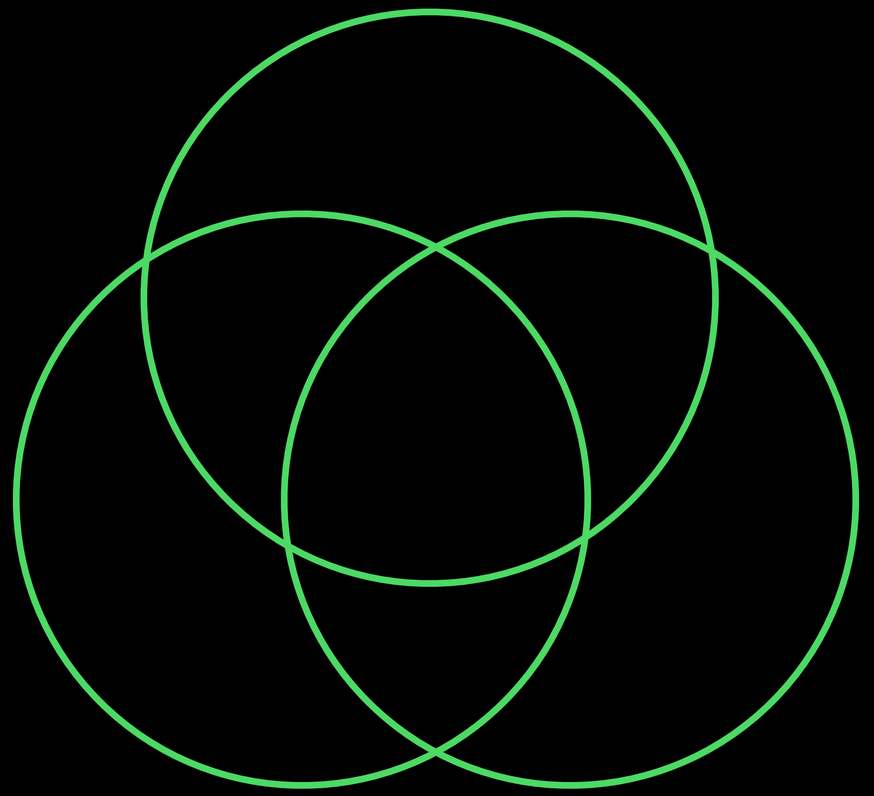
Insight, not dogma

Why, not how

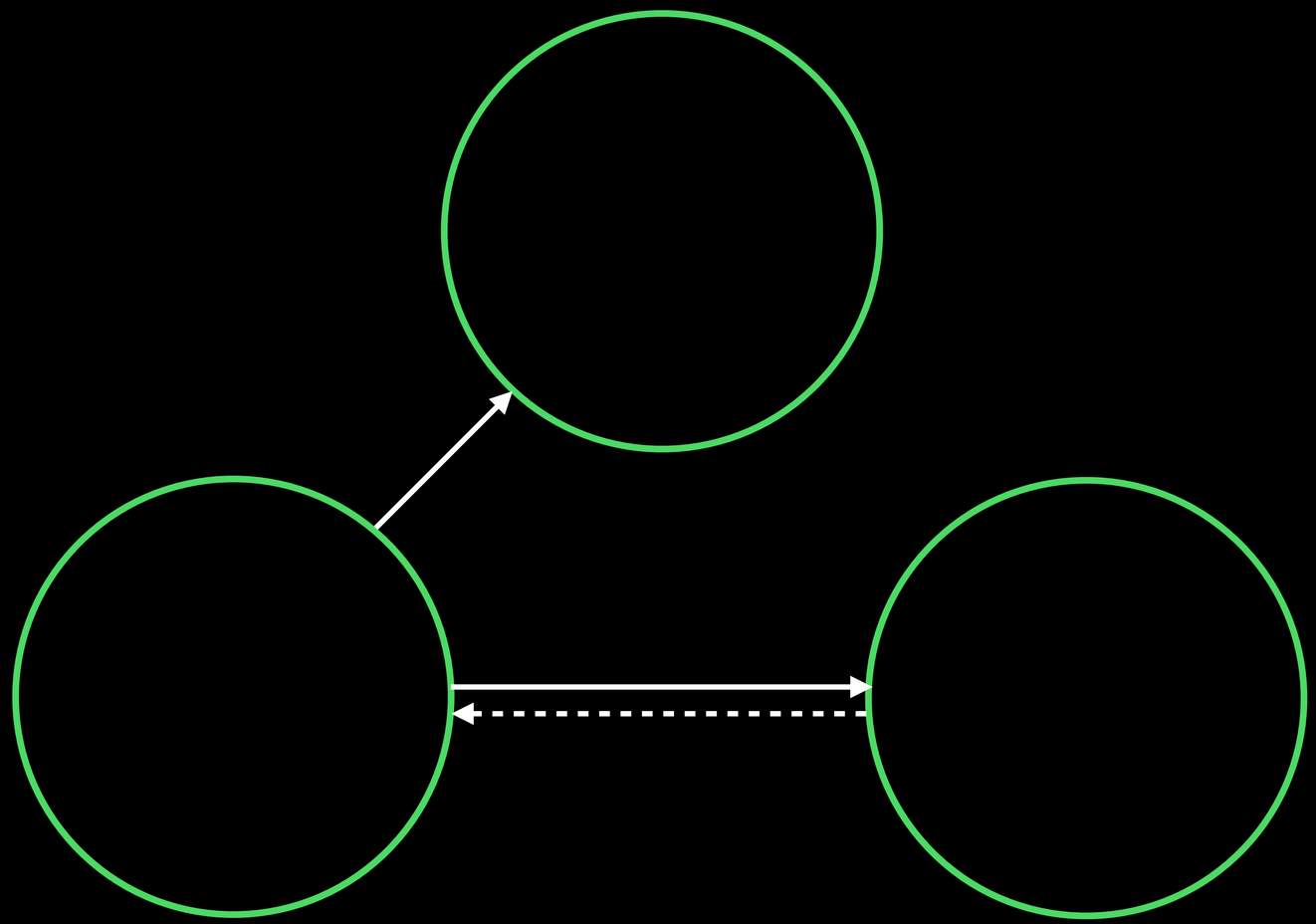


① Design Information Flow



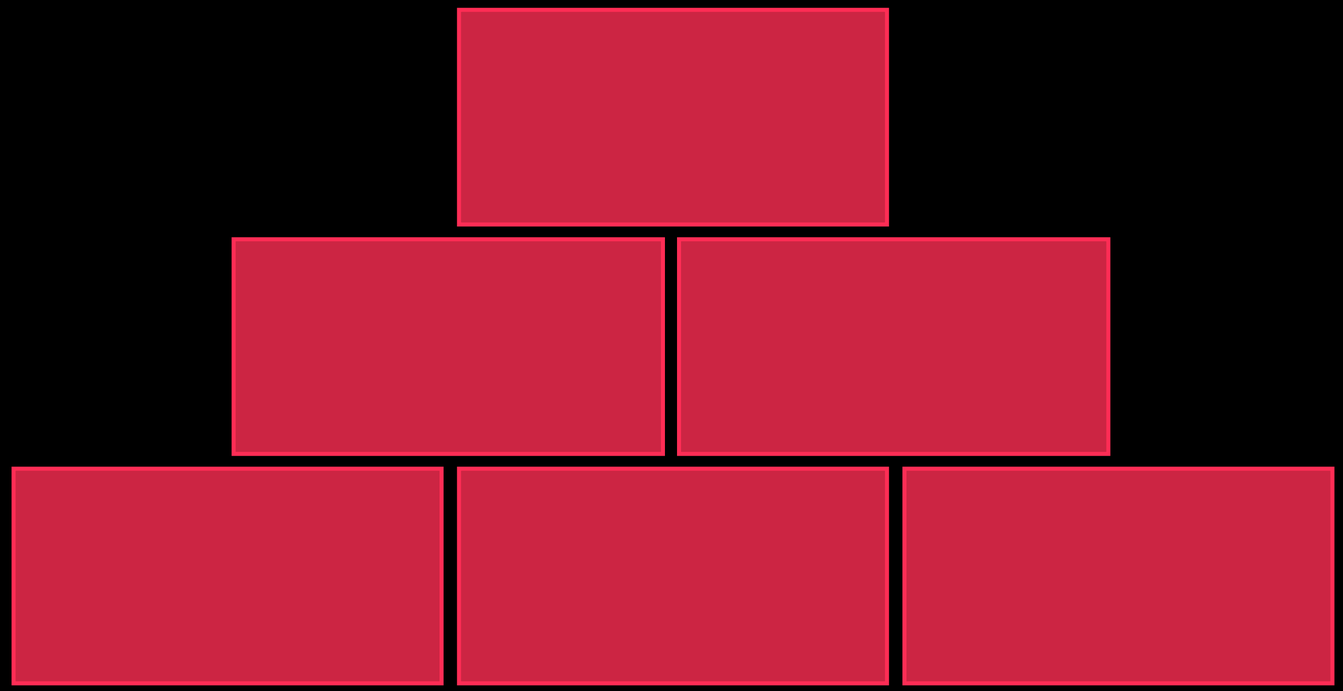


② Define Clear Responsibilities

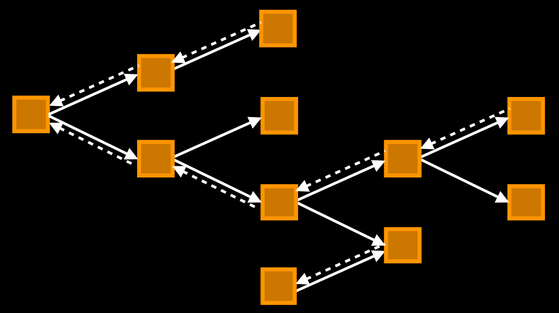


3 Simplify with Immutability

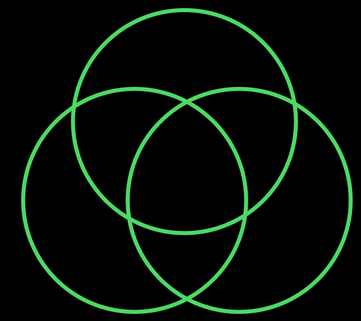
③ Simplify with Immutability



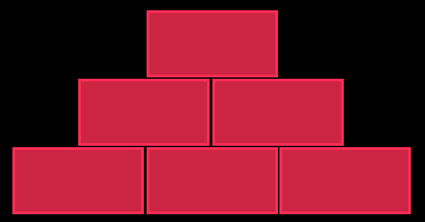
1. Design information flow



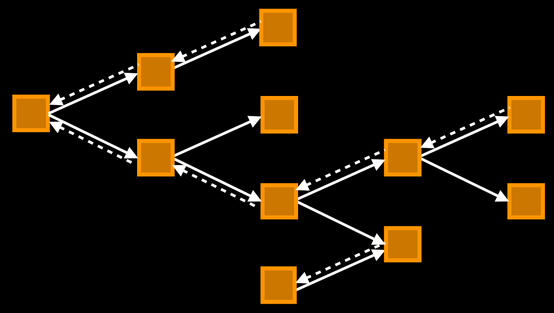
2. Define clear responsibilities



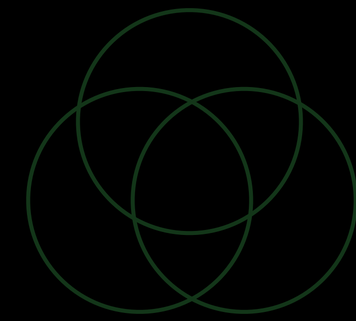
3. Simplify with immutability



1 Design information flow

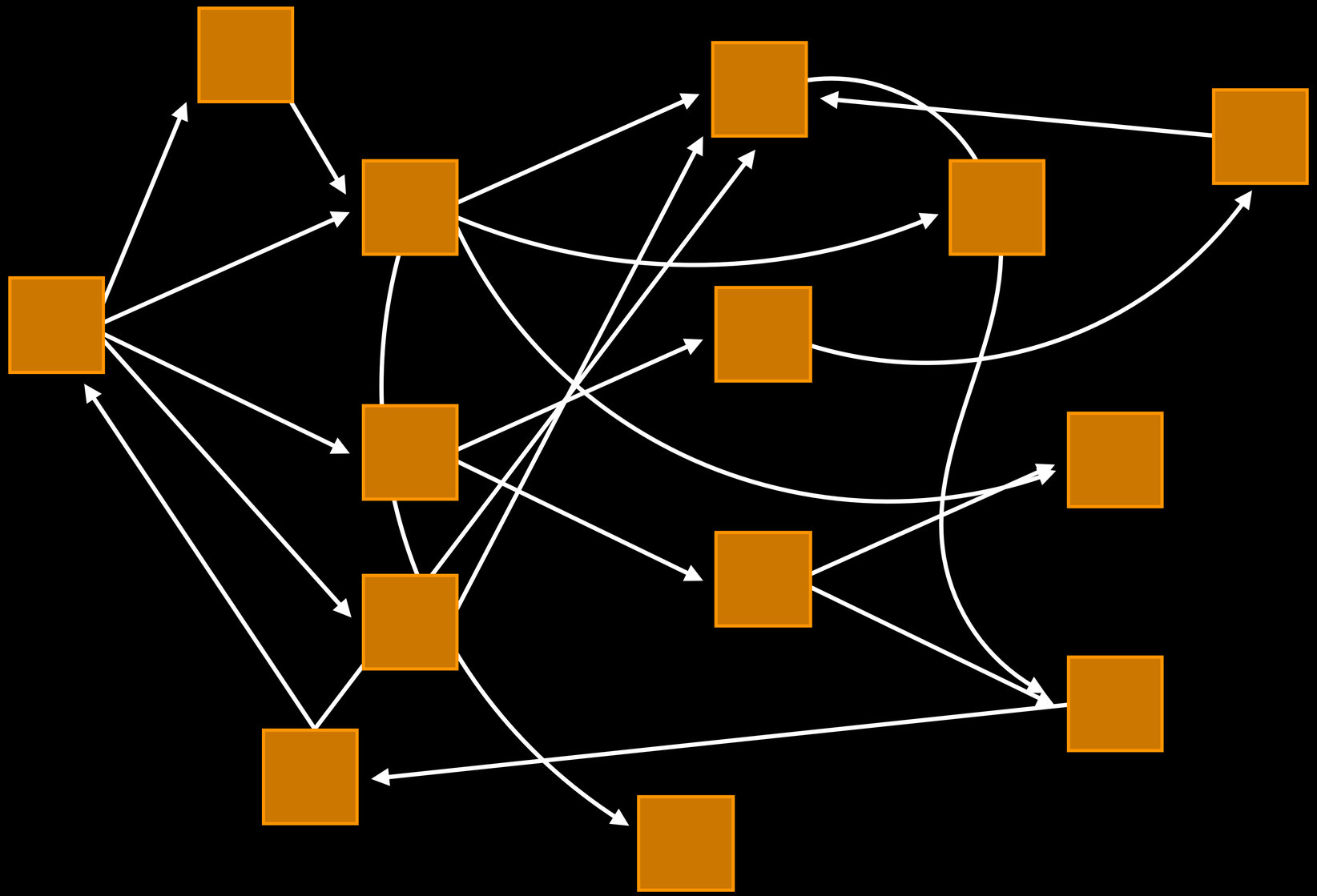


2 Define clear responsibilities



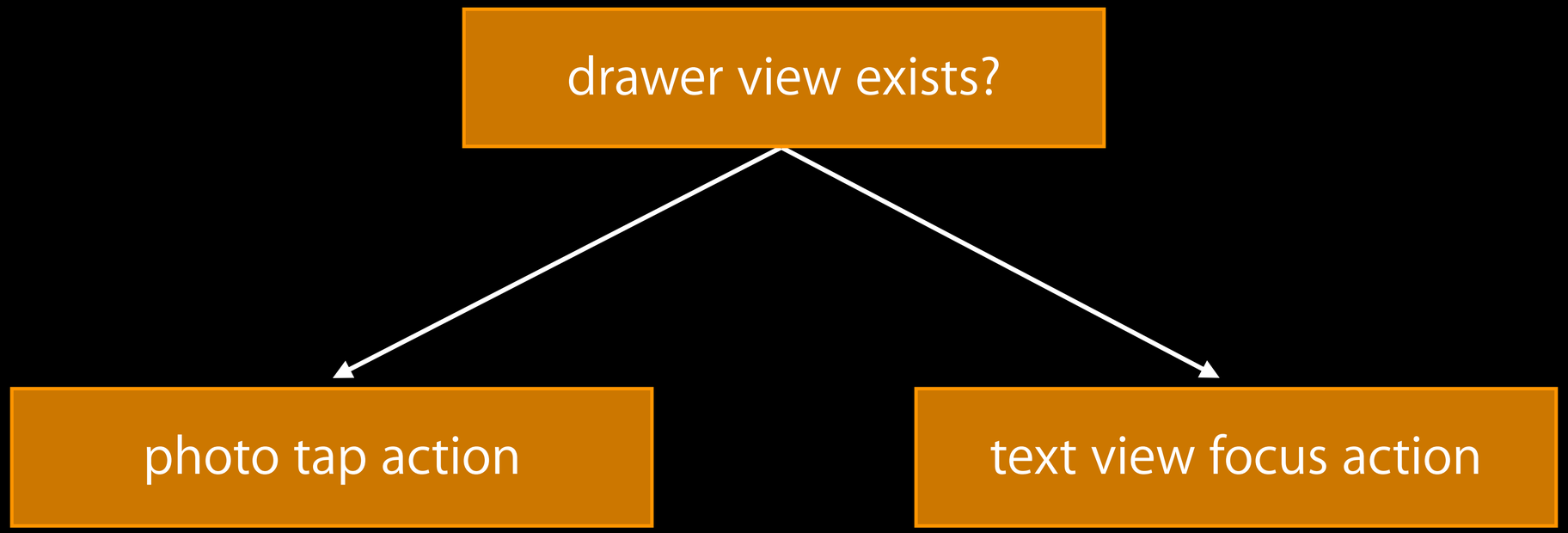
3 Simplify with immutability

Where Is “Truth”?



Demo

Where is truth?

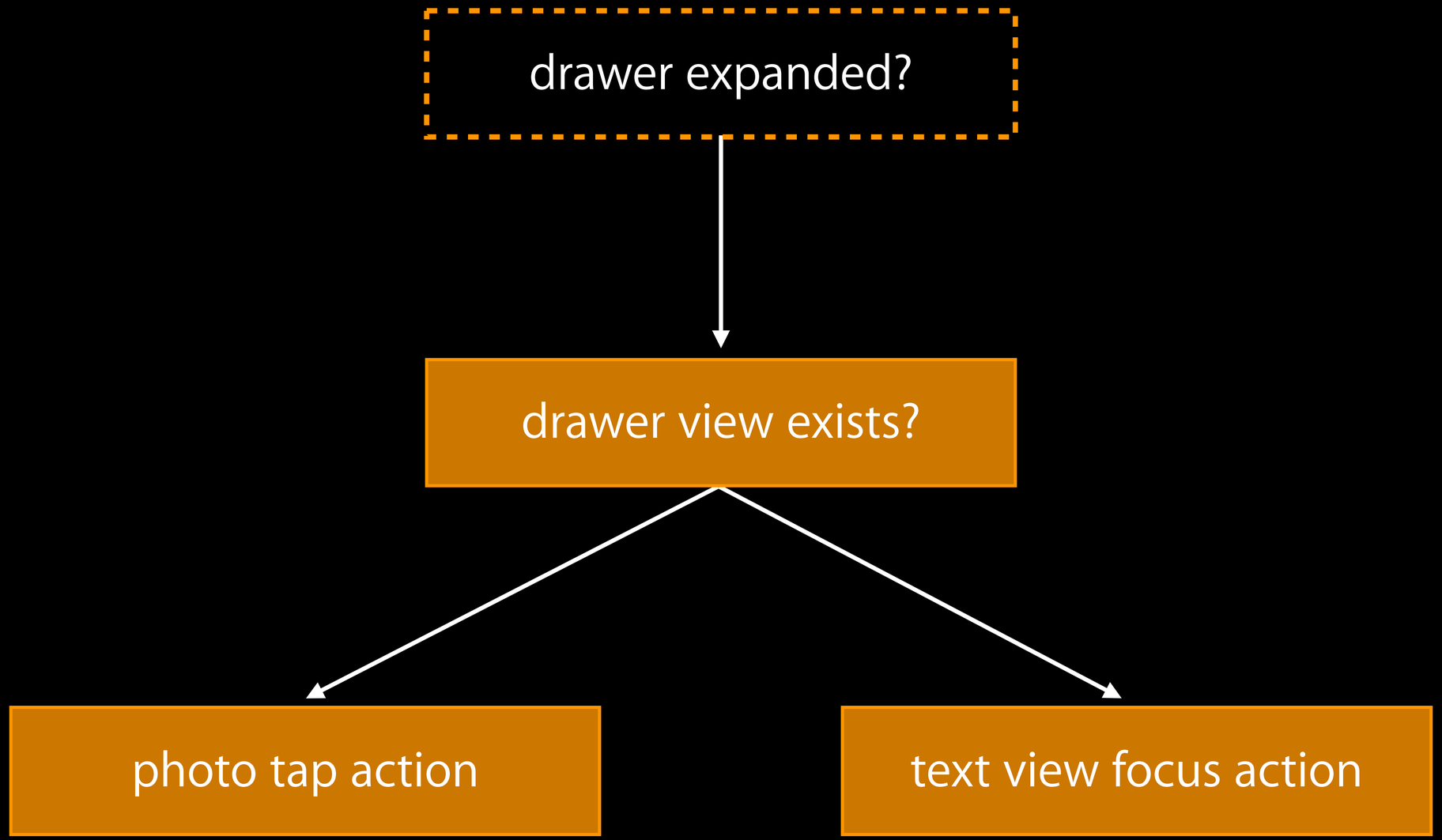

drawer view exists?
photo tap action
text view focus action

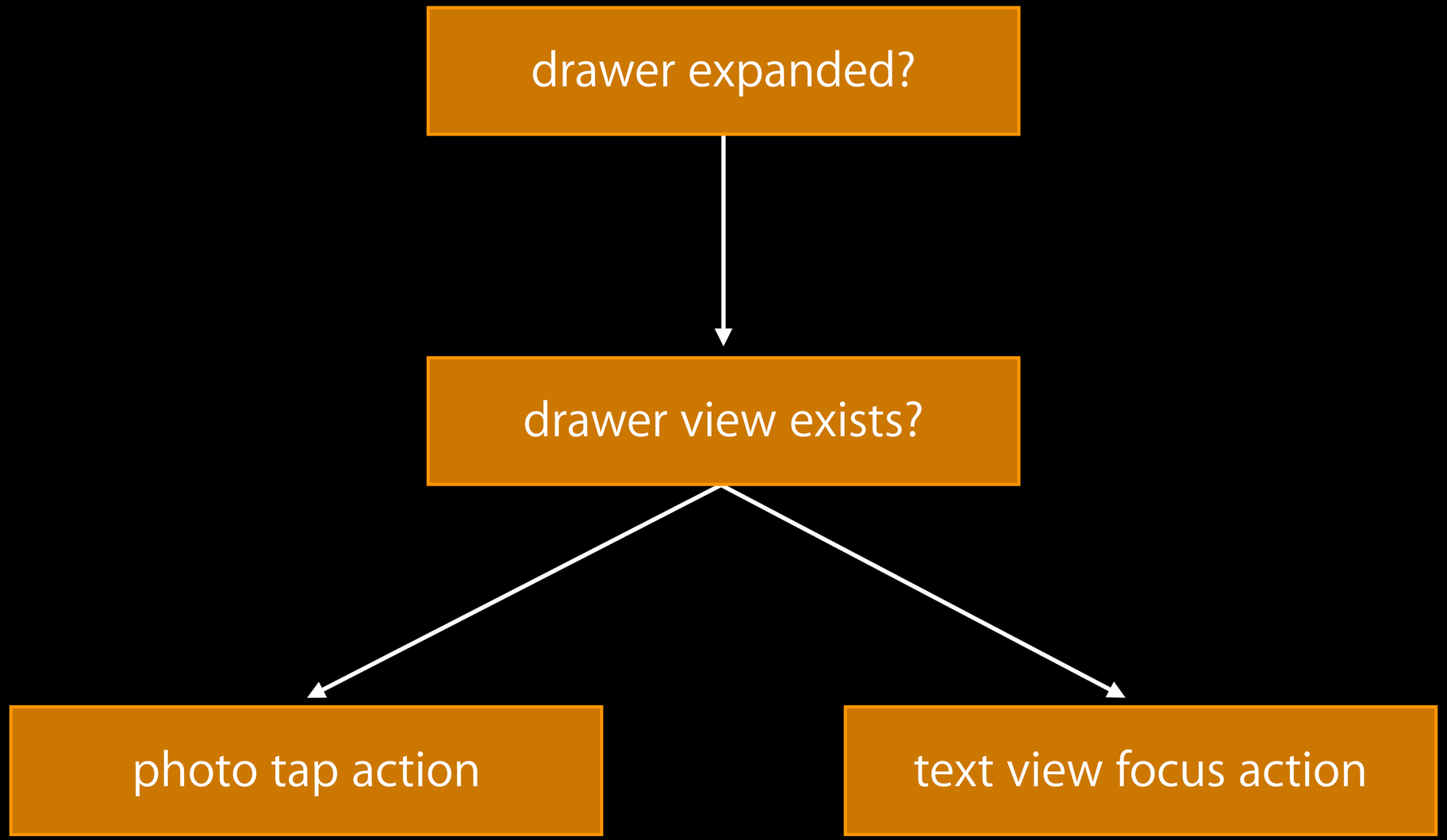
drawer expanded?

drawer view exists?

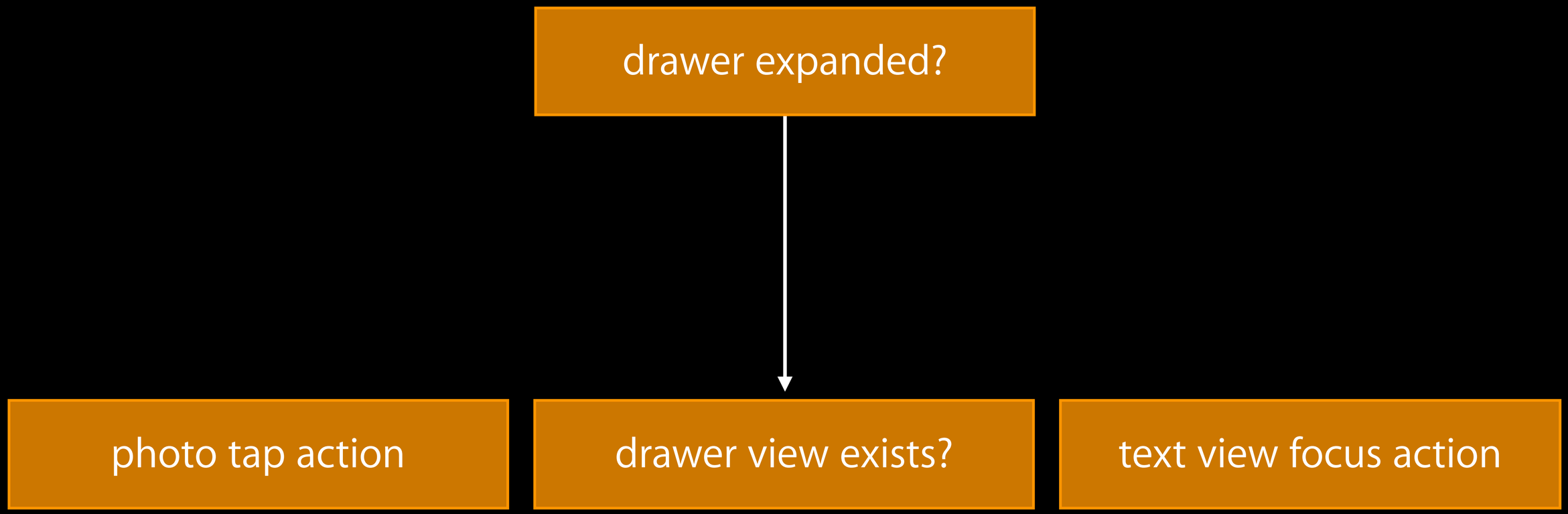
photo tap action

text view focus action

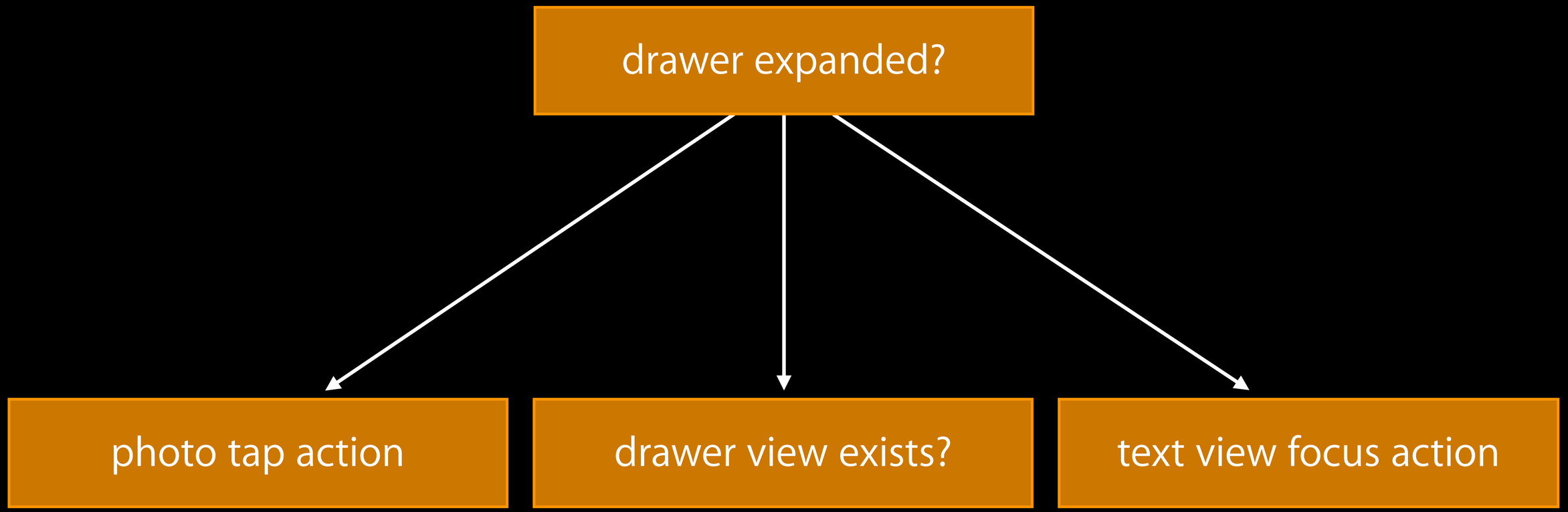

drawer expanded?
drawer view exists?
photo tap action
text view focus action


drawer expanded?
drawer view exists?
photo tap action
text view focus action

Where Is “Truth”?



Where Is “Truth”?



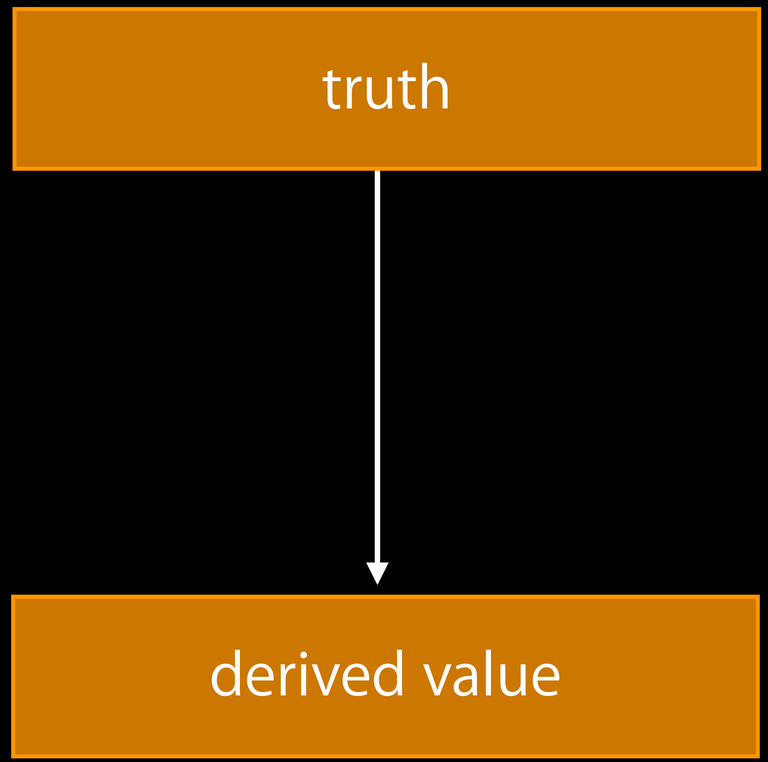
drawer expanded?

drawer view exists?

Truth

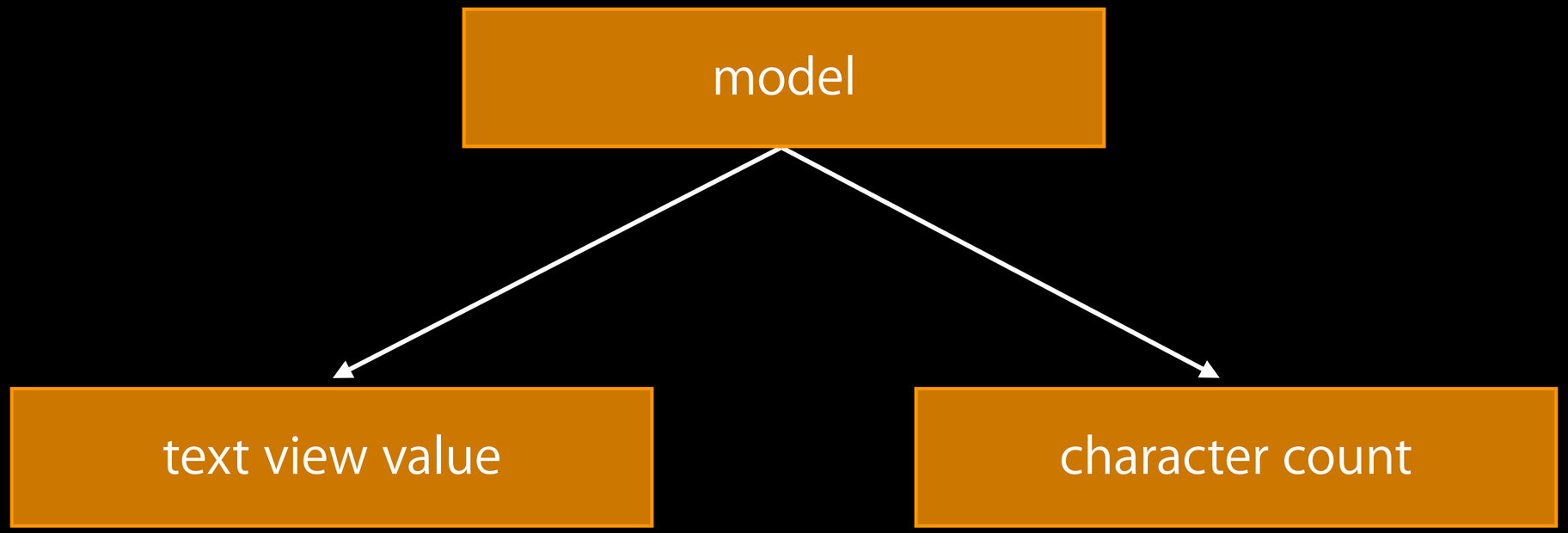
vs.

Derived values

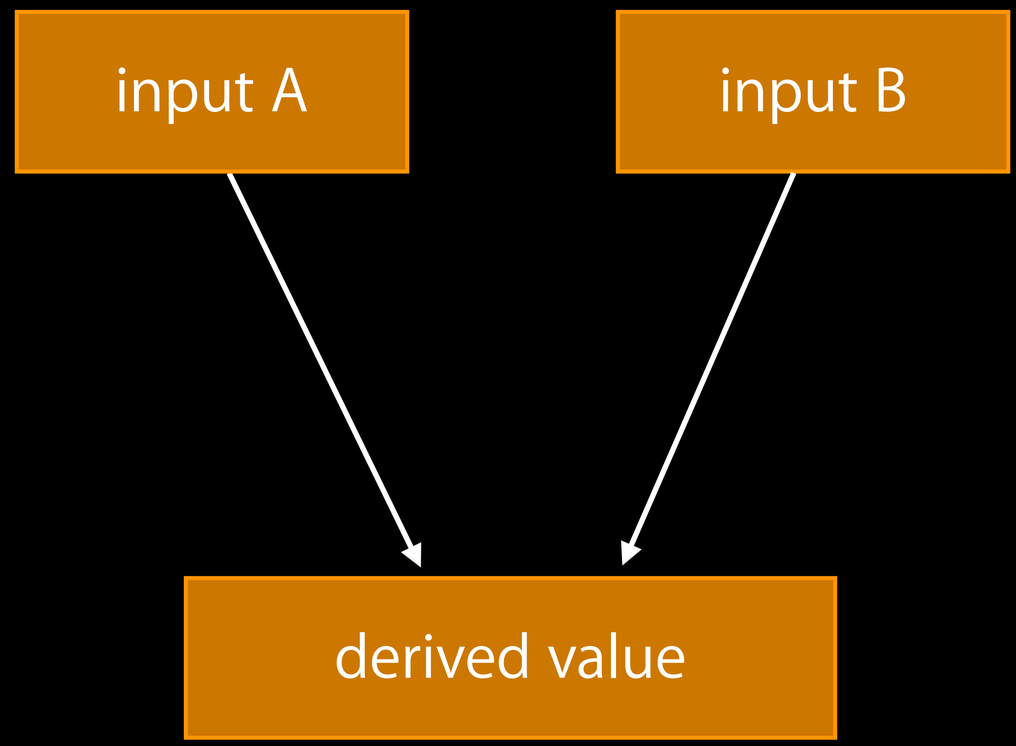


Demo

Truth vs. derived values

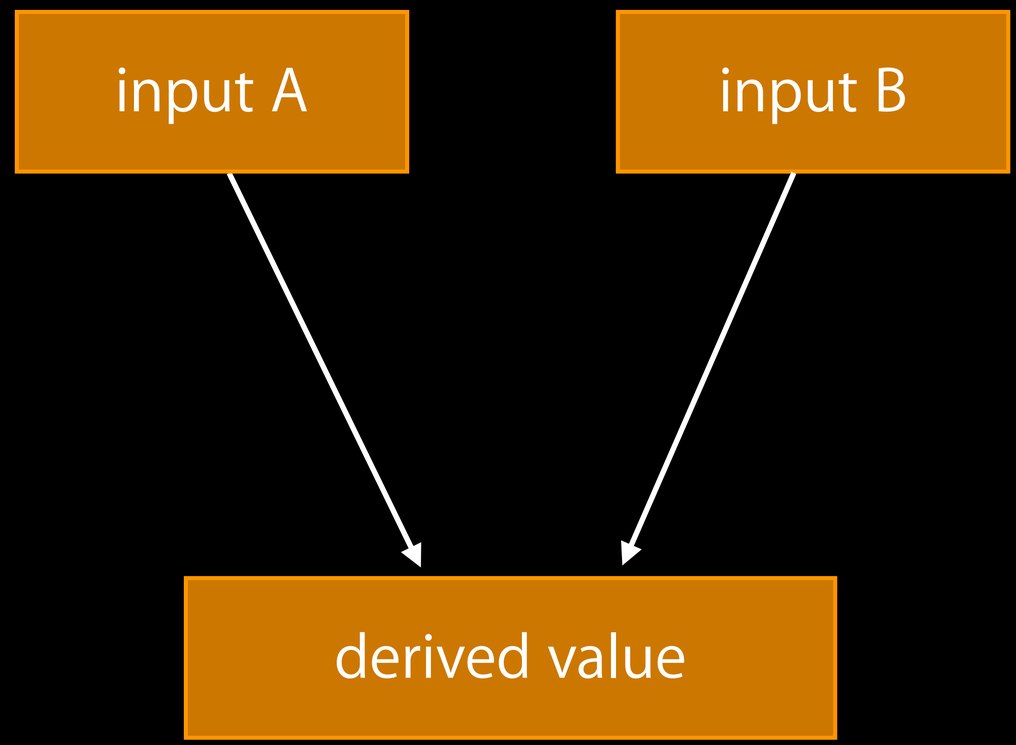

model
text view value
character count

Derived Values



Derived Values

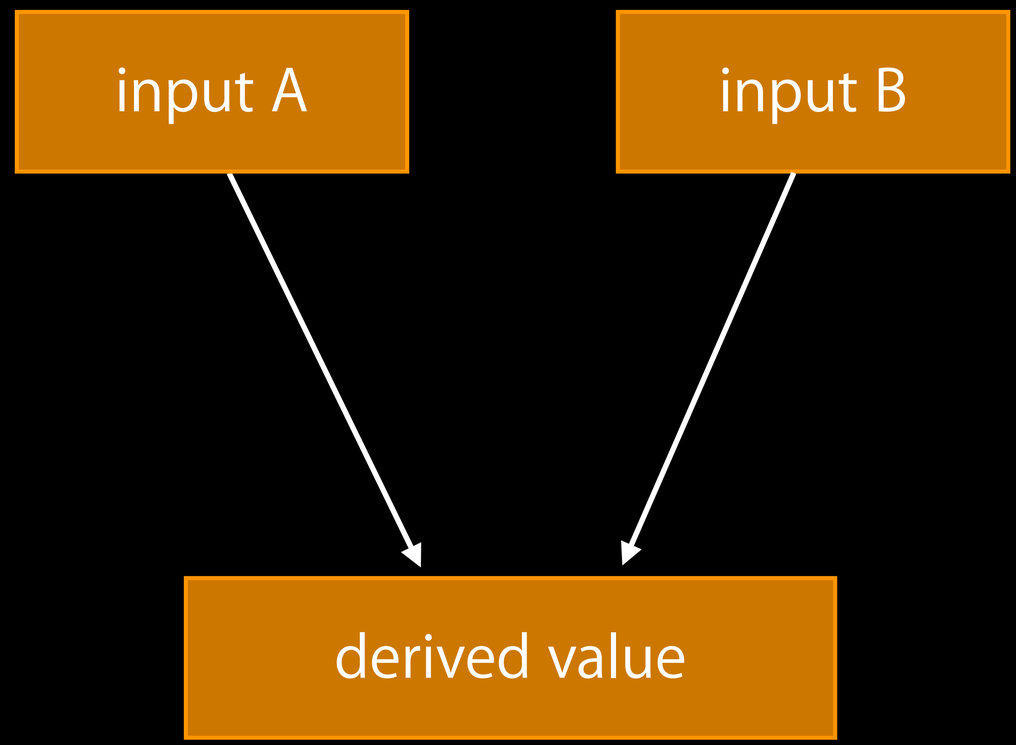
Compute from inputs



Derived Values

Compute from inputs

Recompute when inputs change

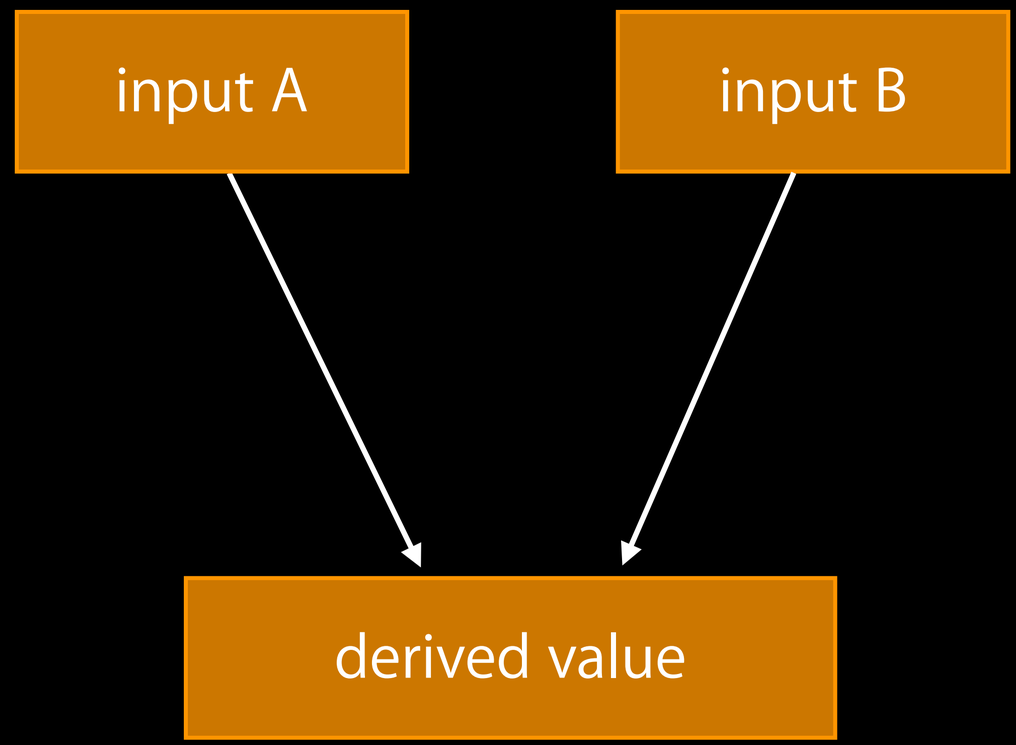


Derived Values

Compute from inputs

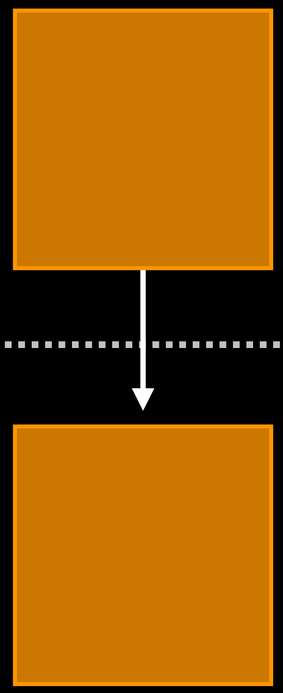
Recompute when inputs change

…Like a cache!



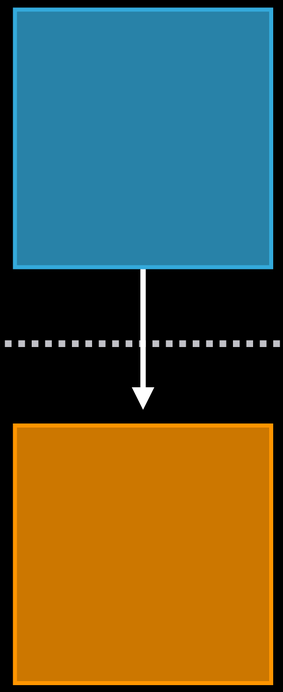
original data

cached data



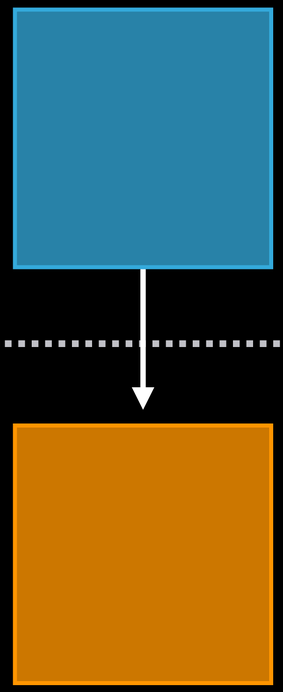
original data

cached data



original data

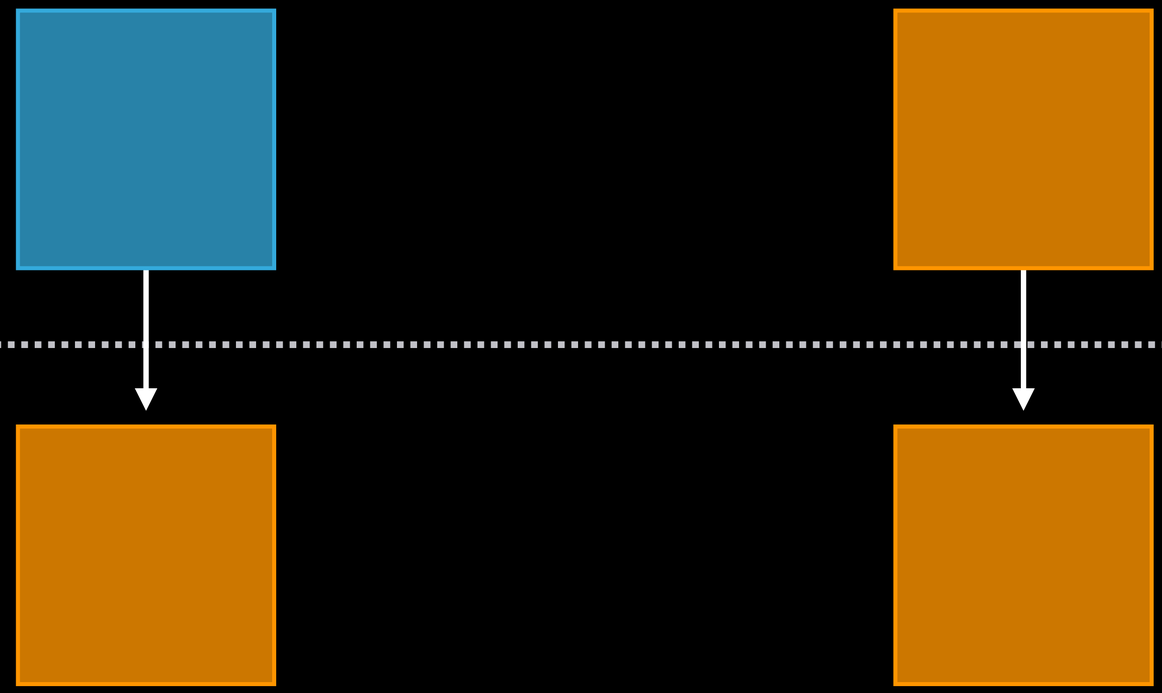
cached data



staleness

original data

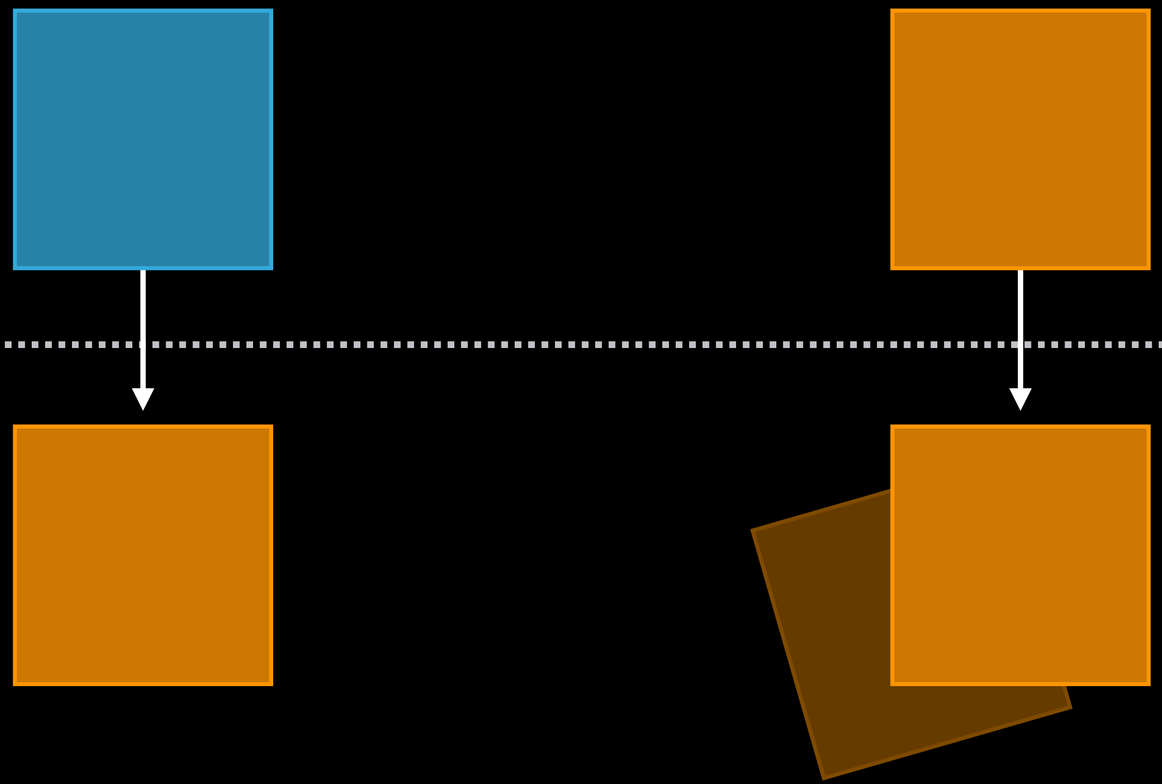
cached data



staleness

original data

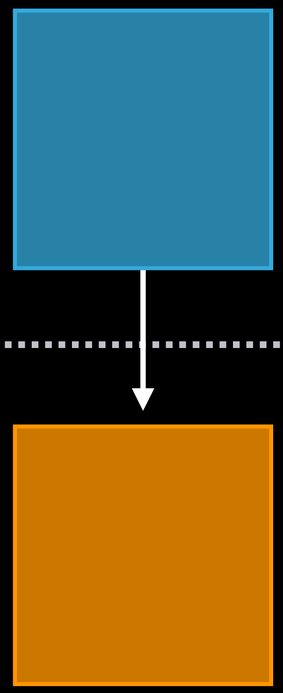
cached data

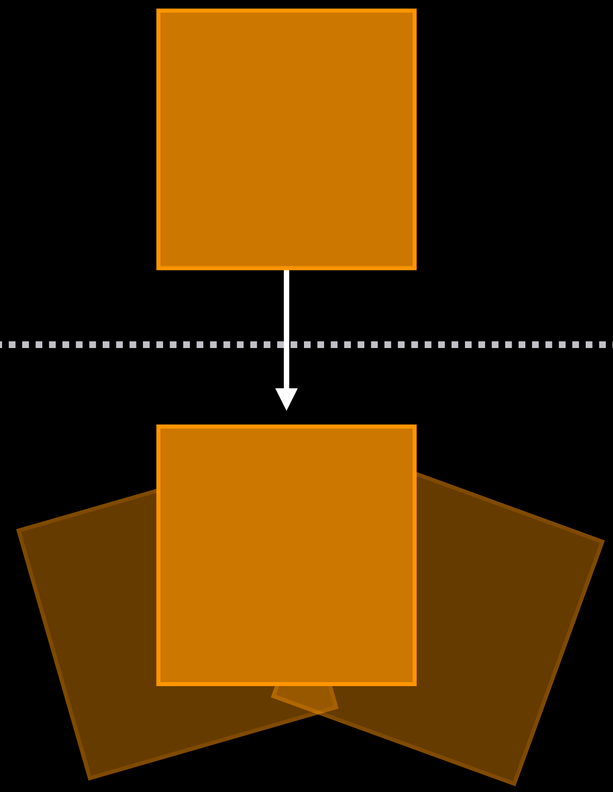


staleness

original data

cached data

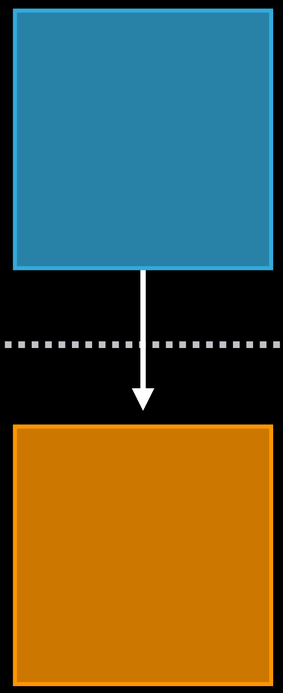




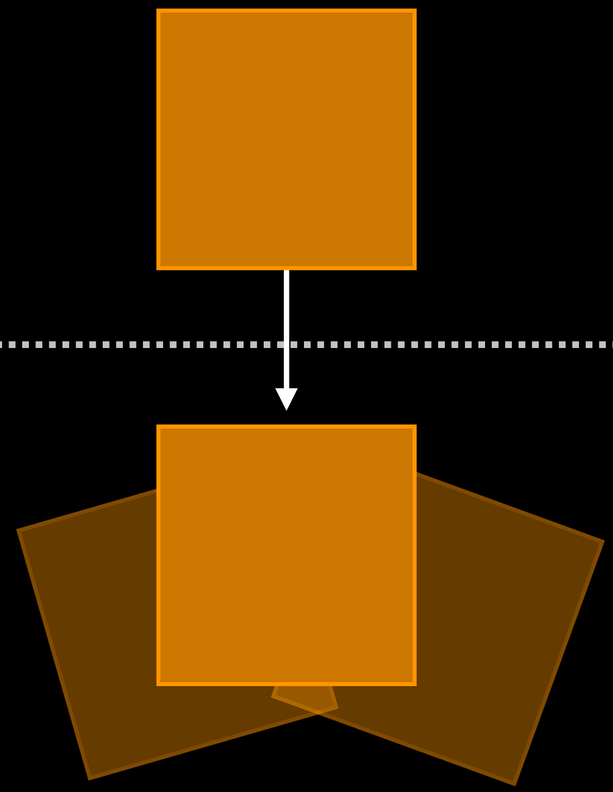
staleness

original data

cached data



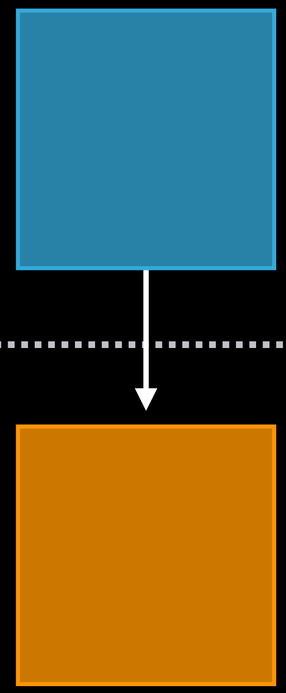
staleness



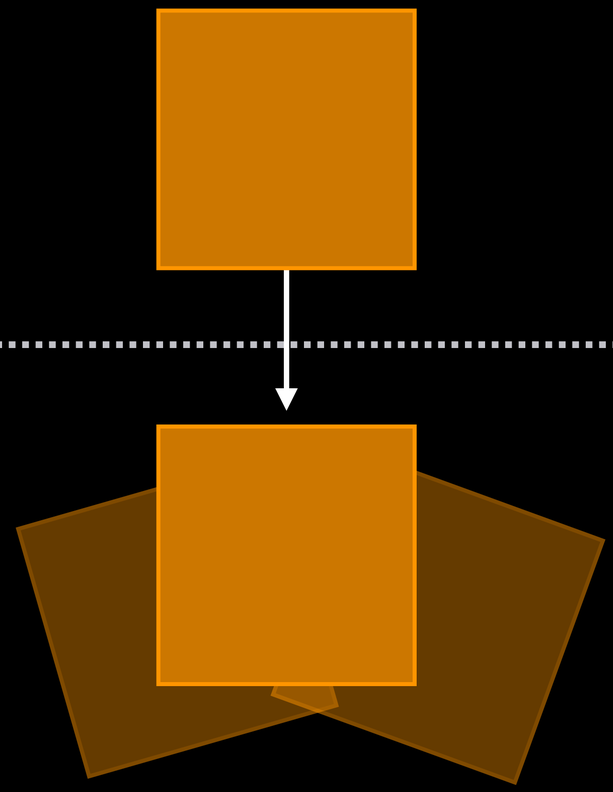
efficiency

truth

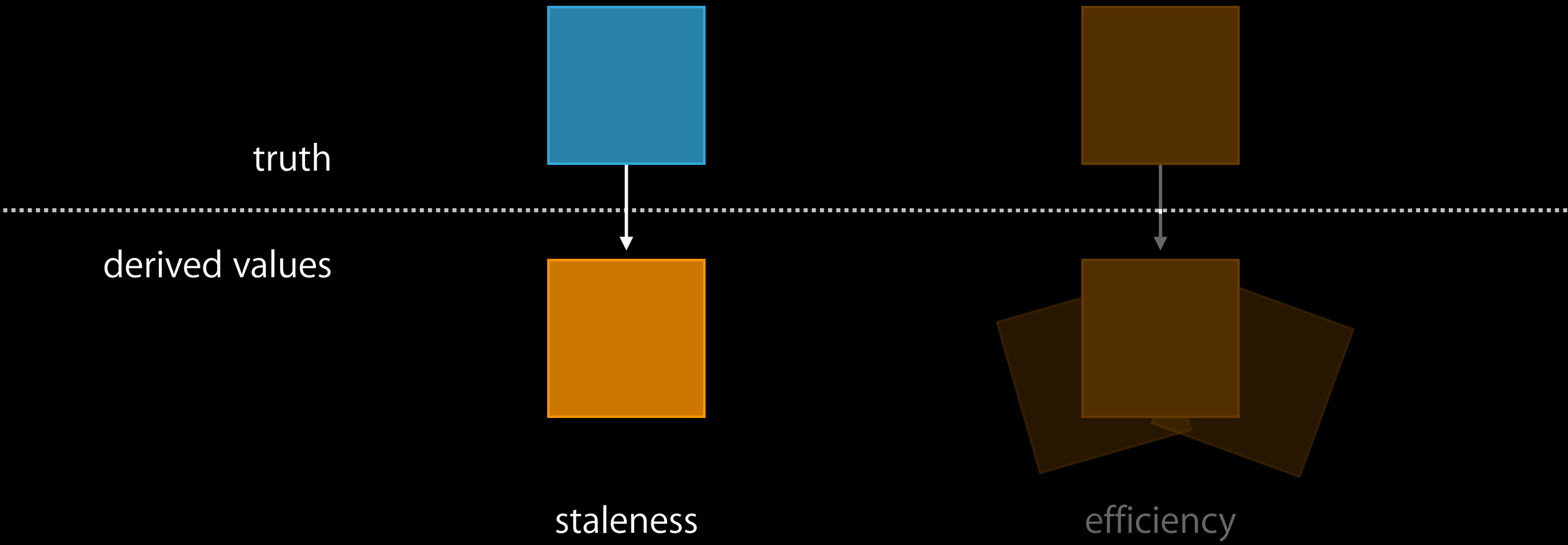
derived values

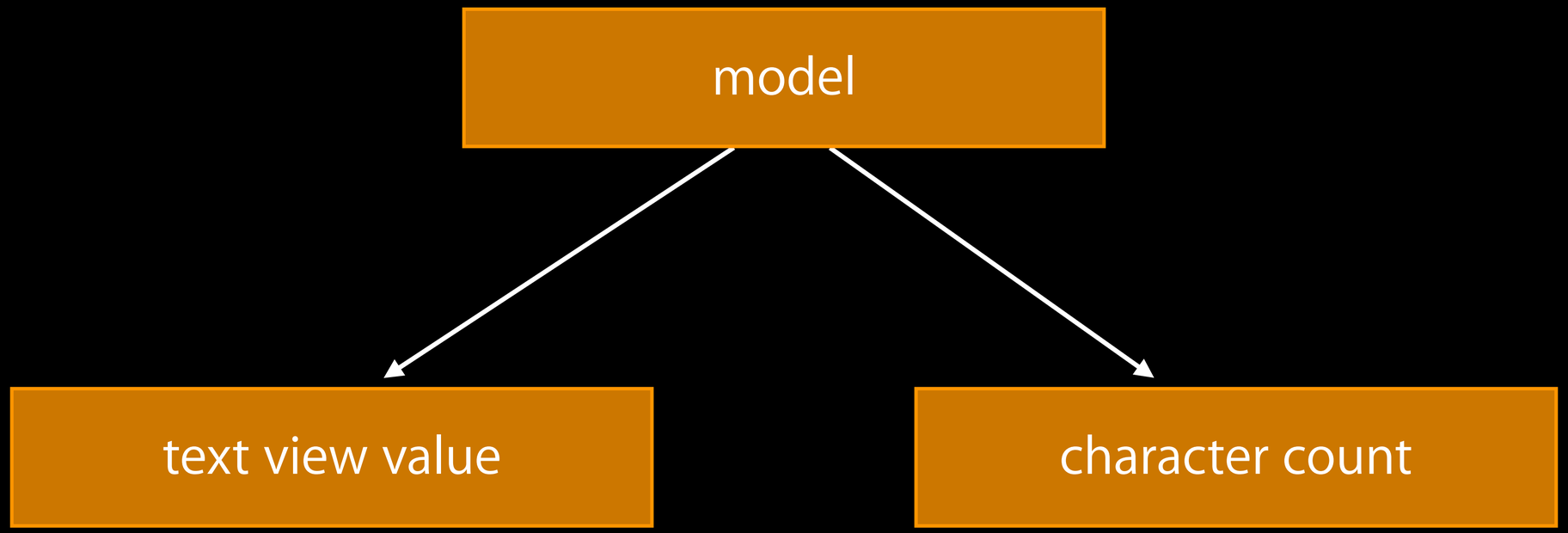


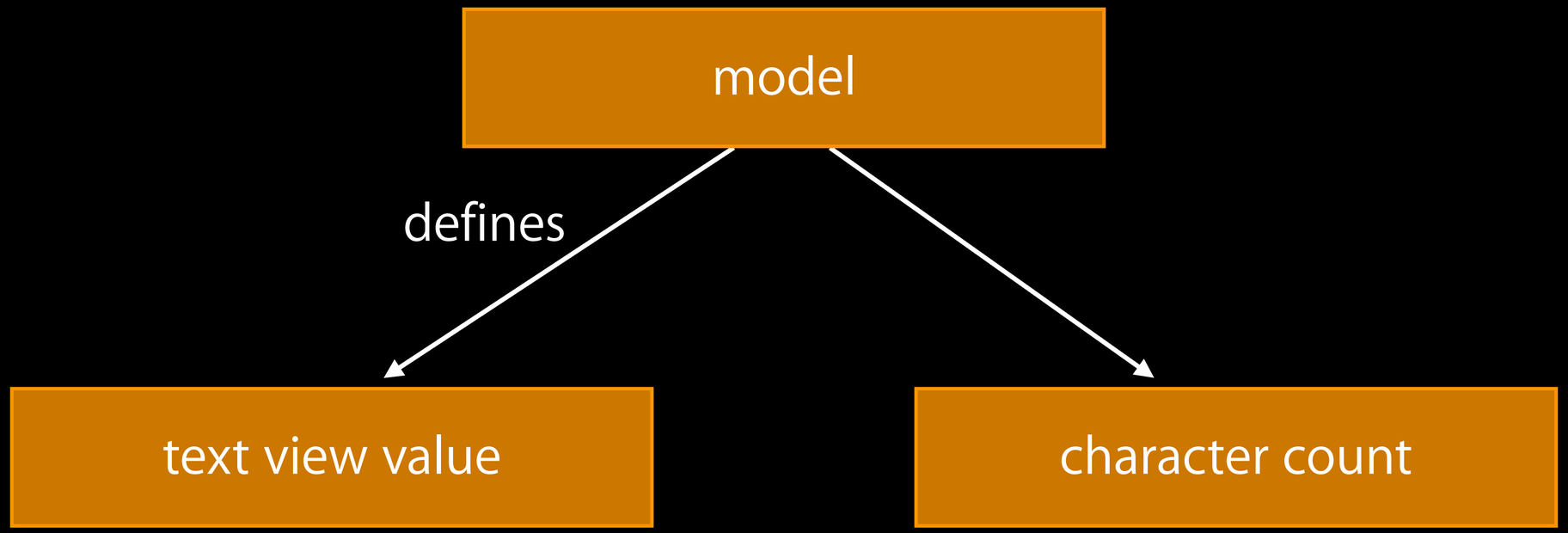
staleness

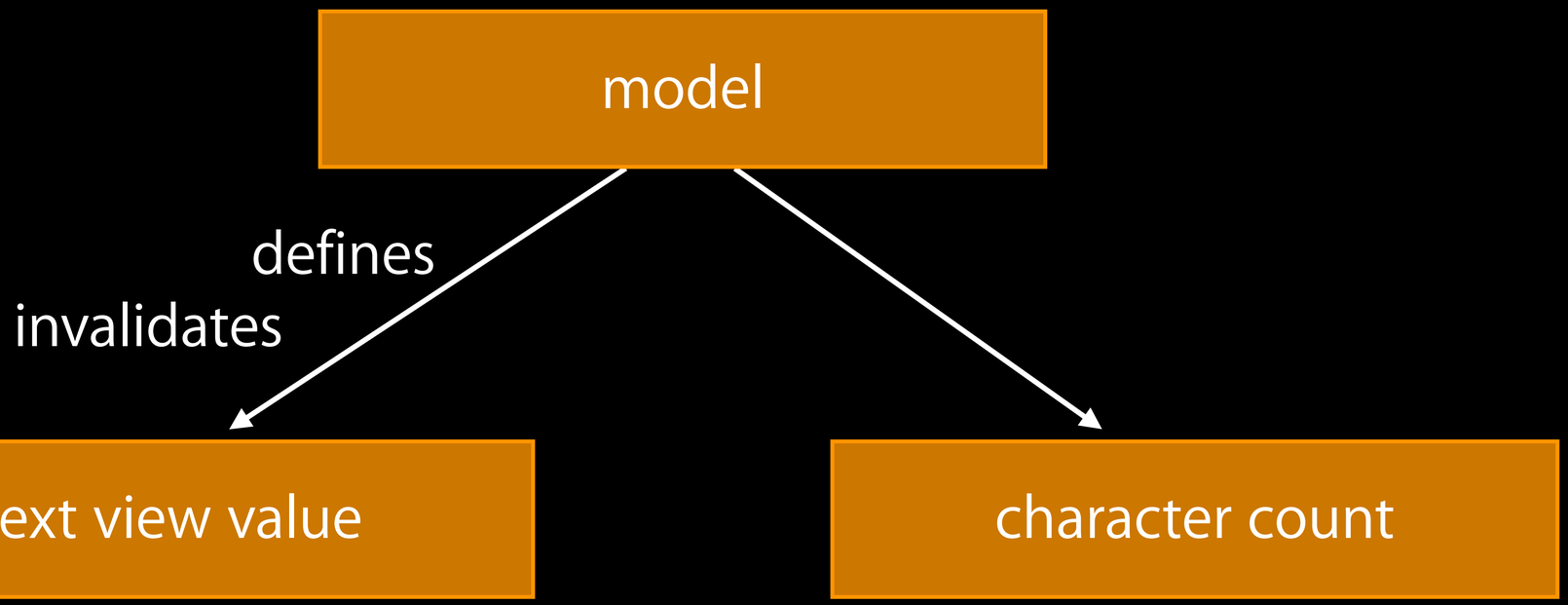


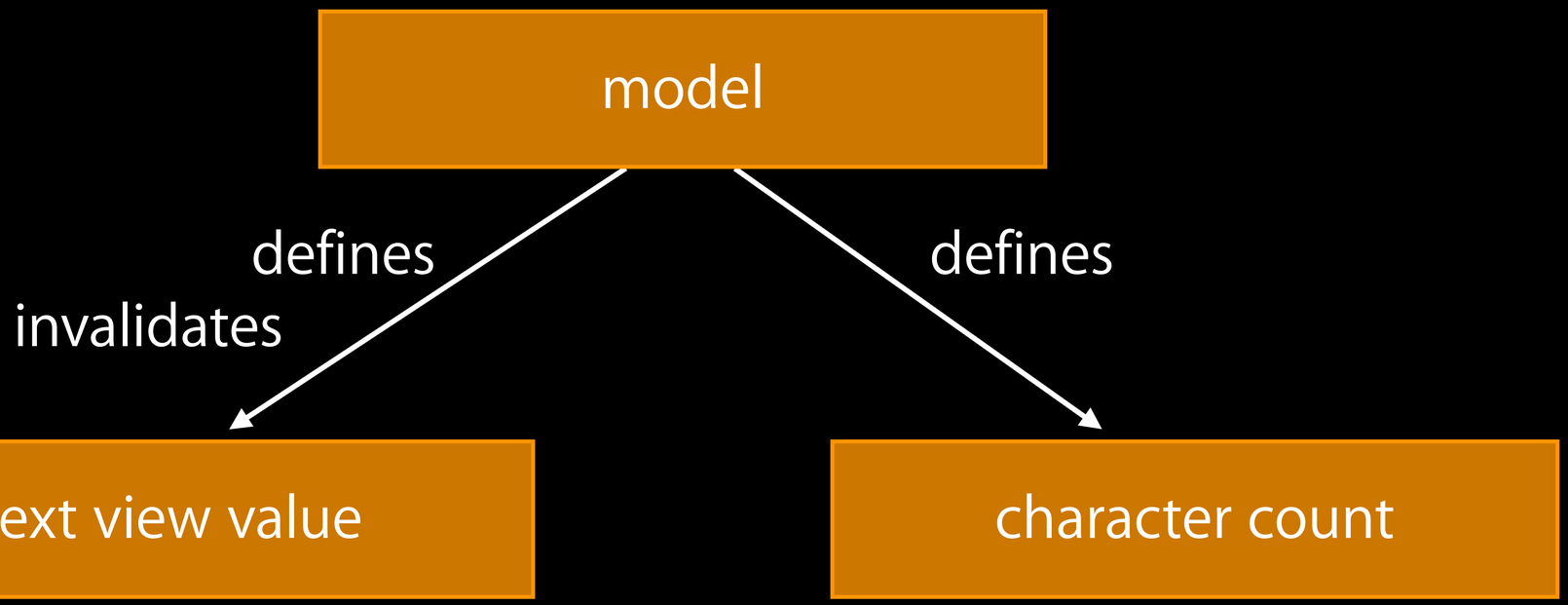
efficiency

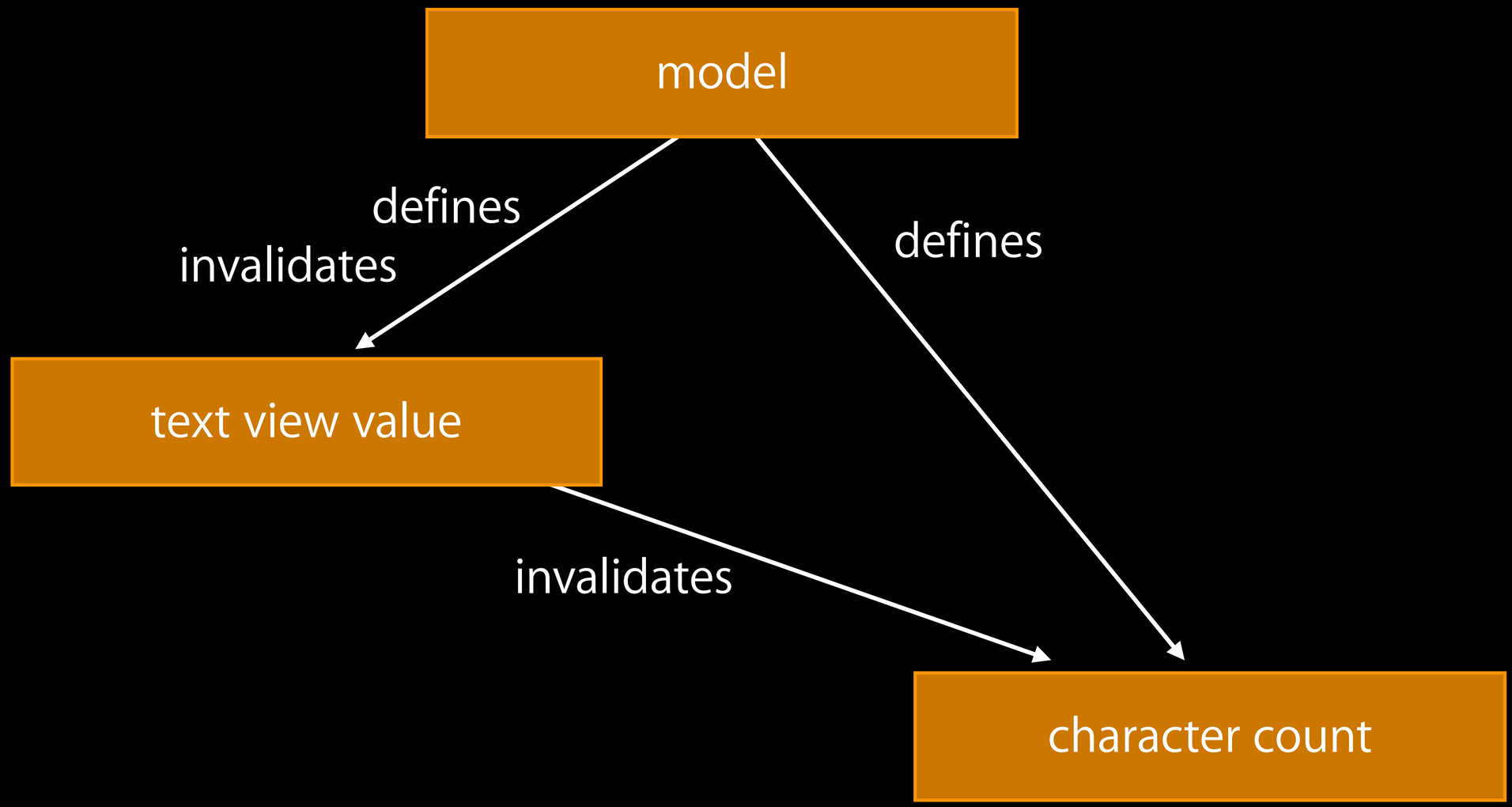

truth
derived values
staleness
efficiency

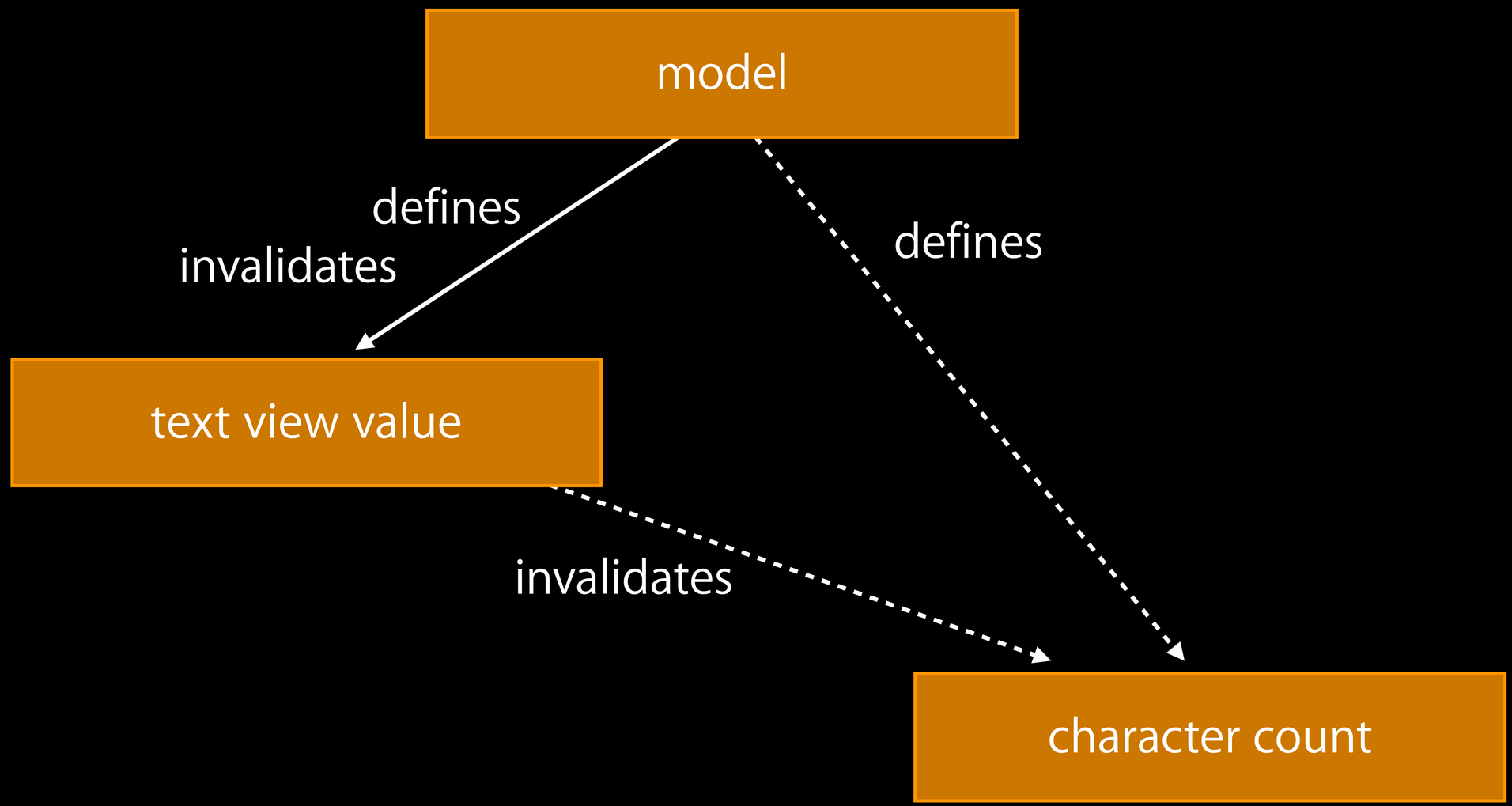

model
text view value
character count

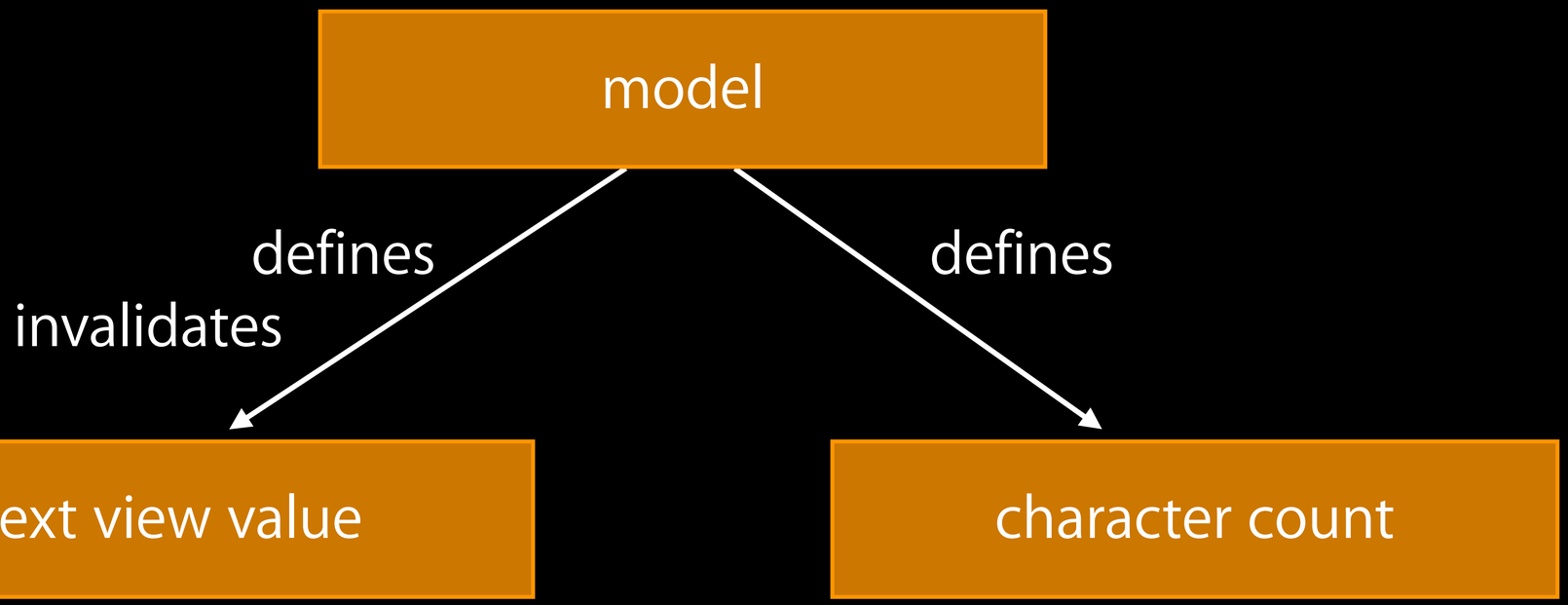

model
defines
text view value
character count

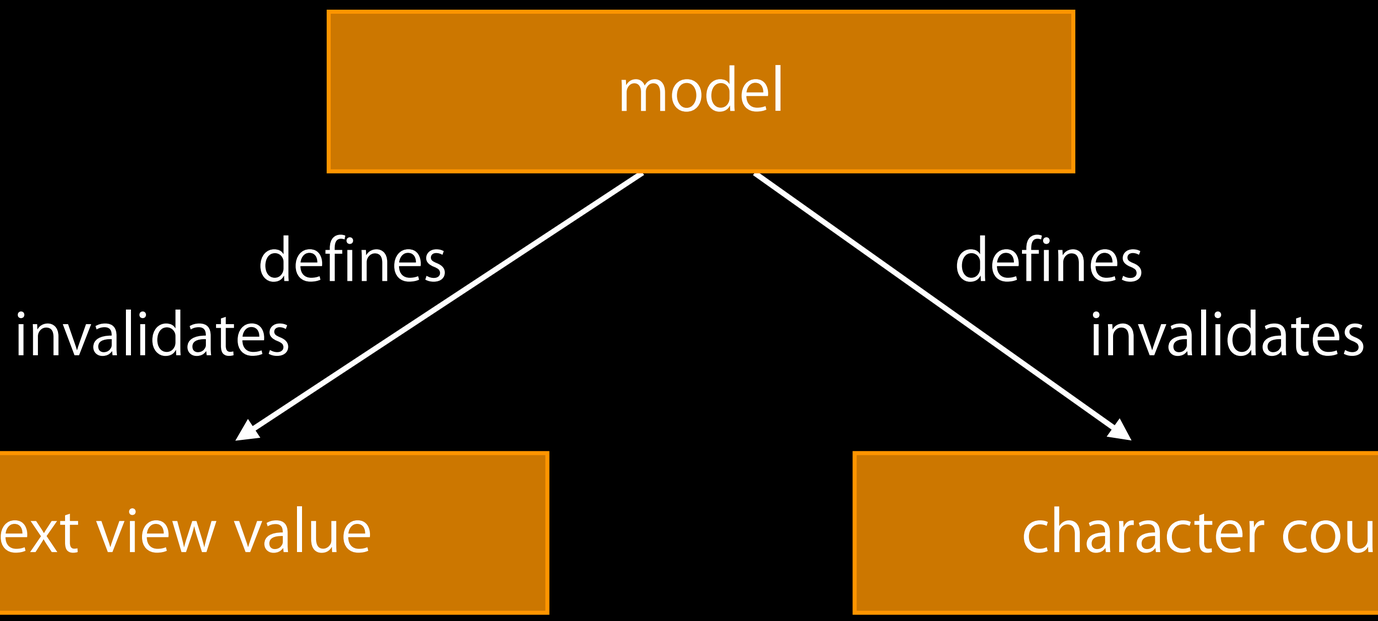

model
defines
invalidates
text view value
character count


model
defines
invalidates
defines
text view value
character count

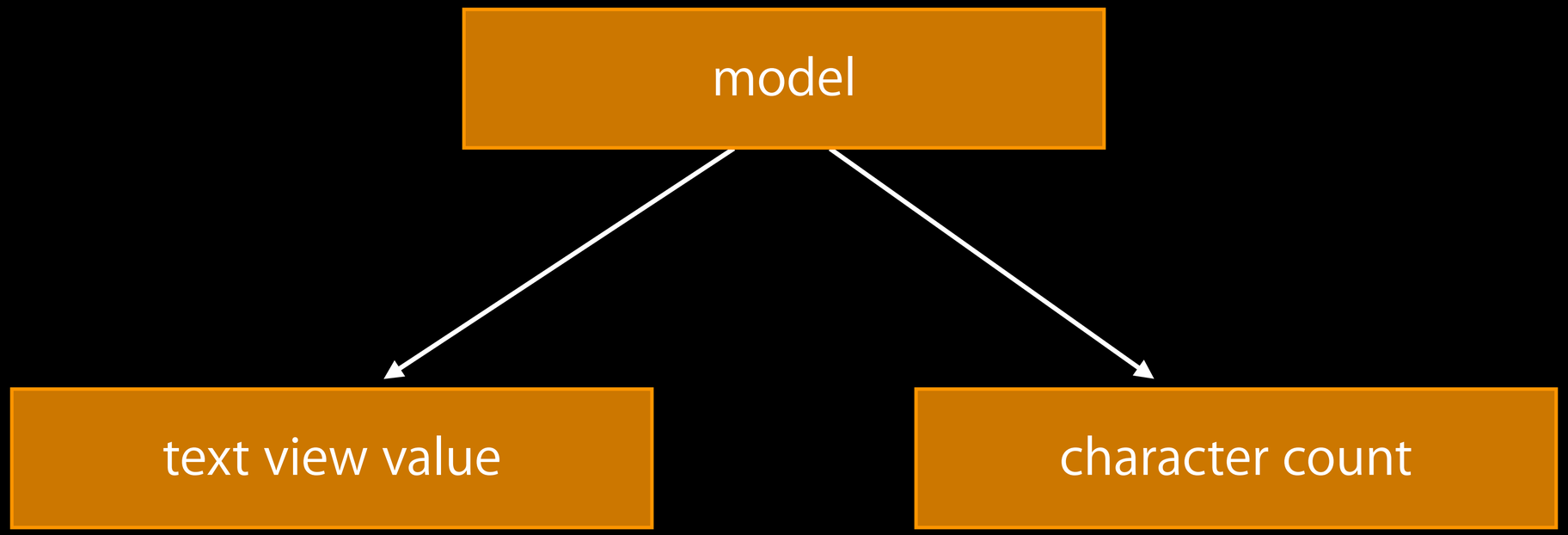

model
defines
invalidates
defines
text view value
invalidates
character count

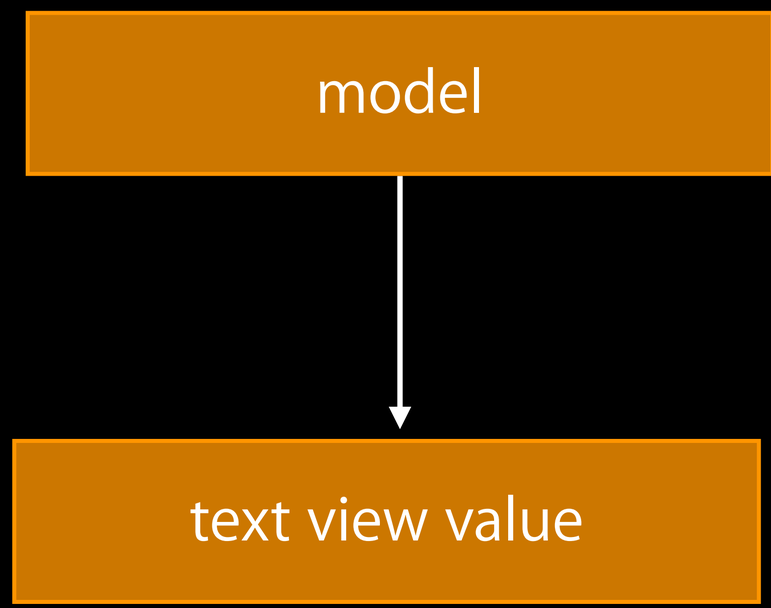

model
defines
invalidates
defines
text view value
invalidates
character count

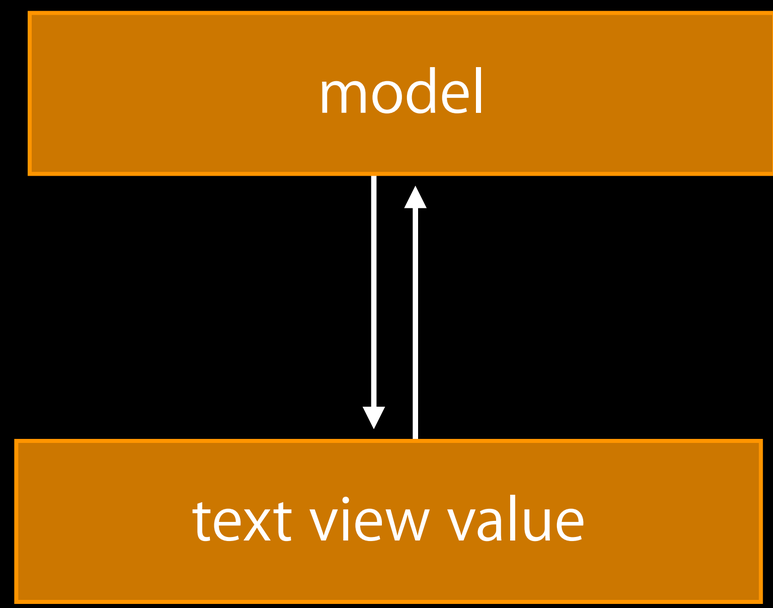

model
defines
invalidates
defines
text view value
character count

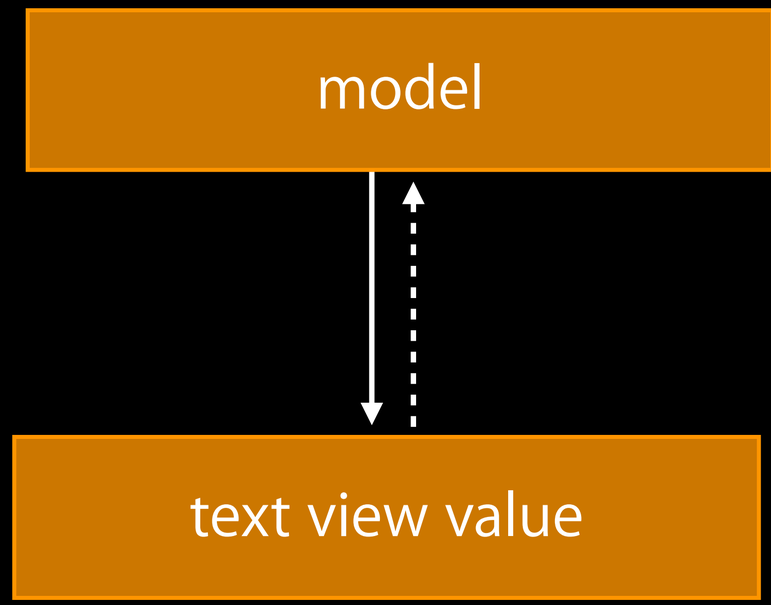

model
defines
invalidates
defines
invalidates
text view value
character count

Truth vs. Derived Values

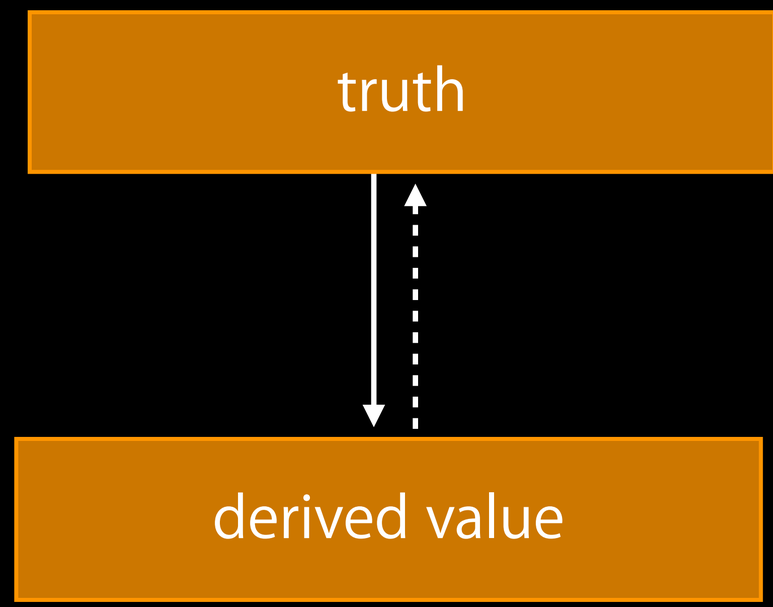



model
text view value


model
text view value

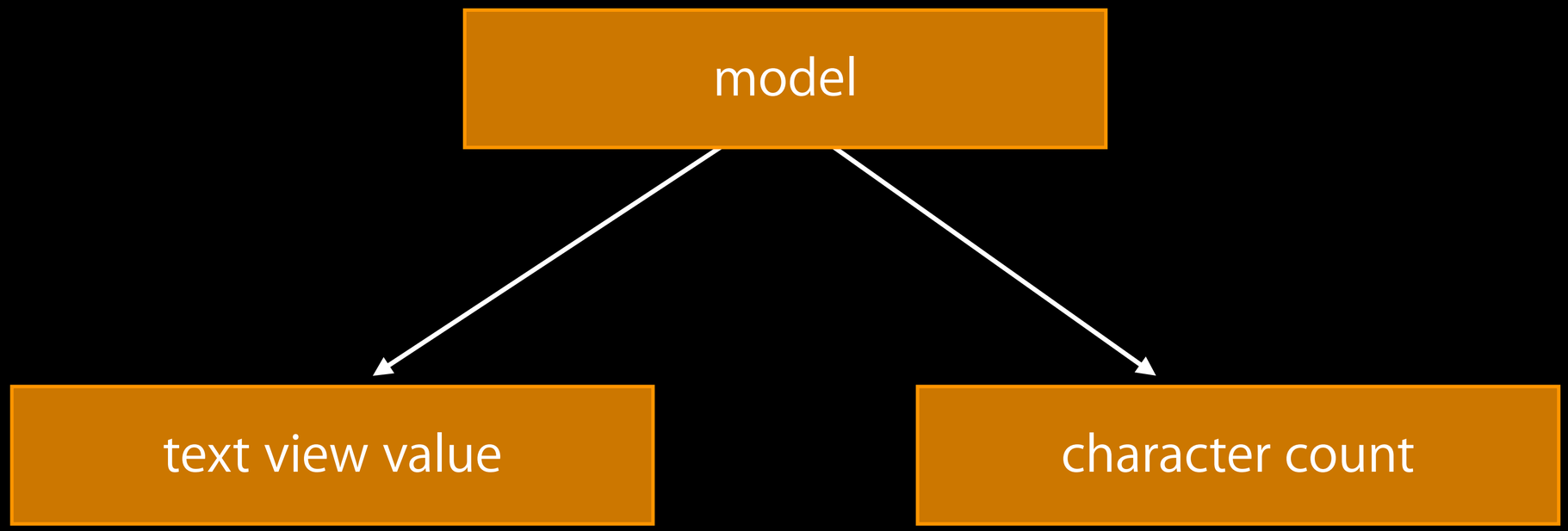

model
text view value

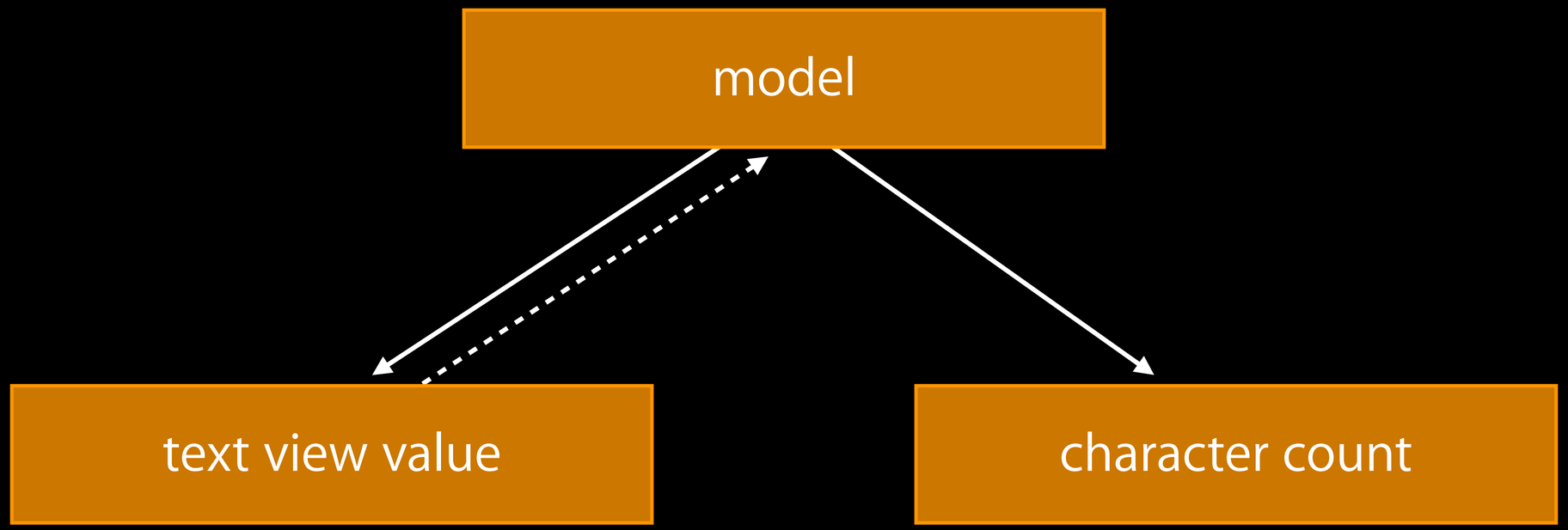
Creating New Truth



Demo

Creating new truth


model
text view value
character count


model
text view value
character count

entry viewing controller

entry editing controller

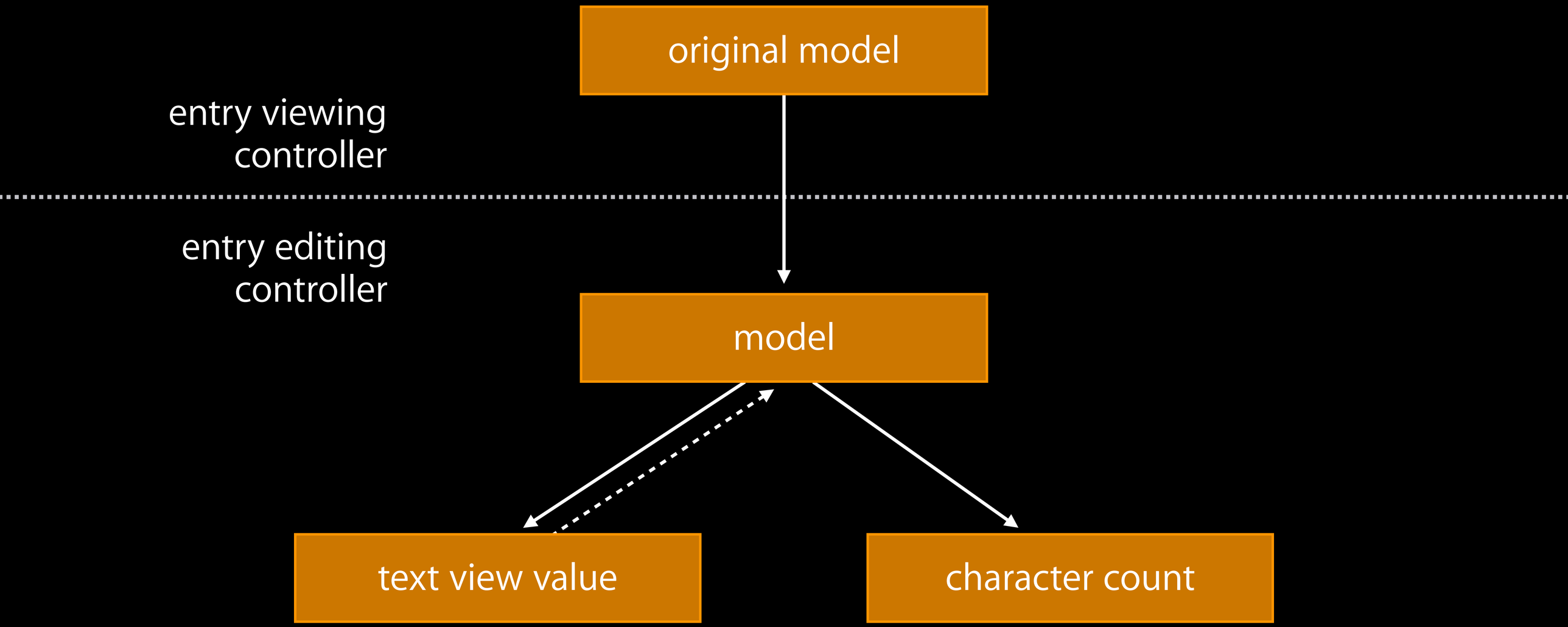
original model

model

text view value

character count

Creating New Truth



① Design Information Flow

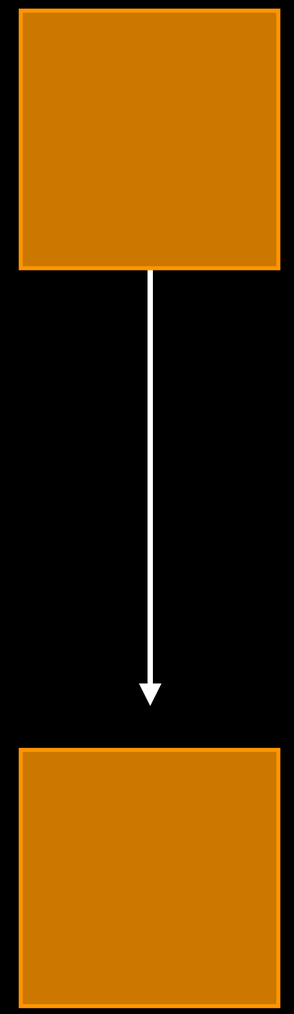
1 Design Information Flow

Where is “truth”?

1 Design Information Flow

Where is “truth”?

Truth vs. derived values

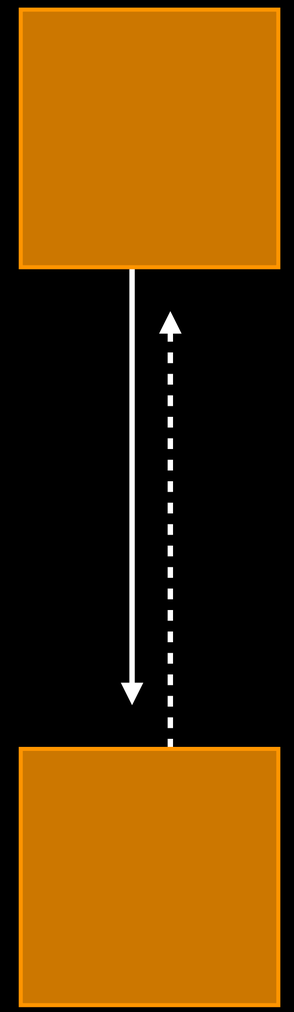


1 Design Information Flow

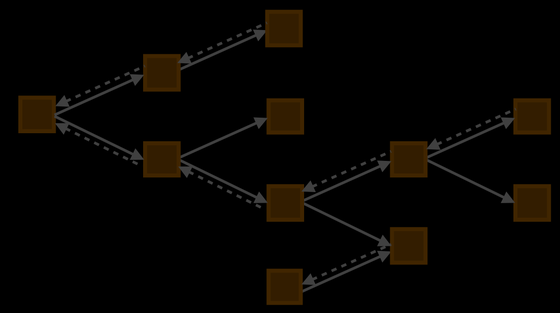
Where is “truth”?

Truth vs. derived values

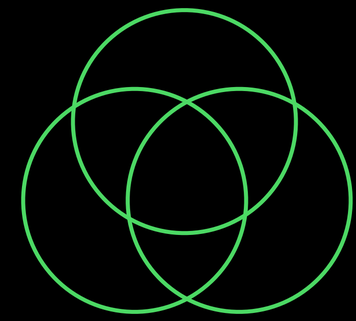
Creating new truth



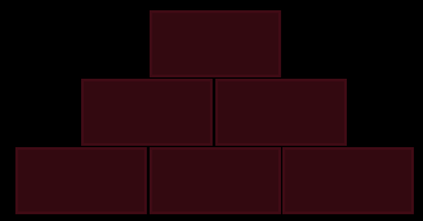
1 Design information flow



2 Define clear responsibilities



3 Simplify with immutability



Sign In

Invalid username or password

Username Johny Appleseed ?

Password ••••••••••••••

Confirm Password ••••••••••••••

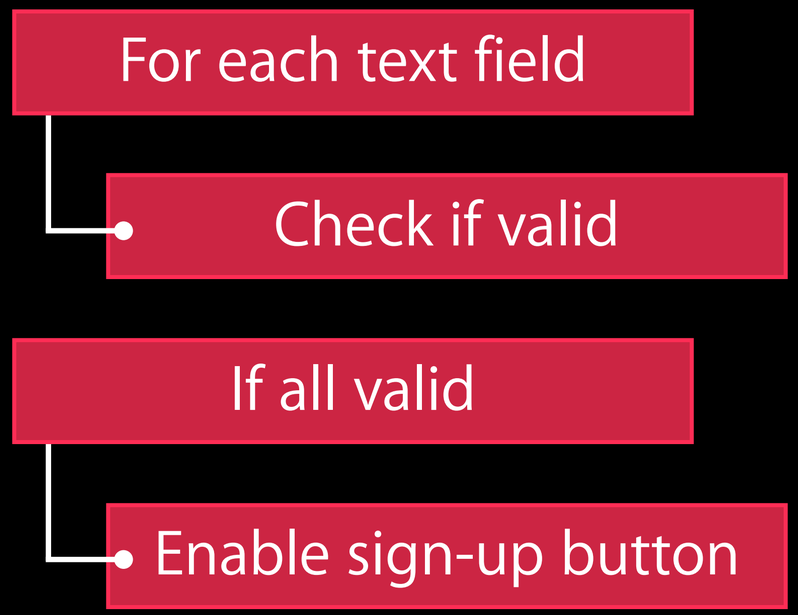
Email johnnyappleseed@icloud.com

Sign In

Forgot password?

Validation

Validation


For each text field
Check if valid
If all valid
Enable sign-up button

Validation

For each text field

Check if valid

If all valid

Enable sign-up button

Validation

For each text field

Check if valid

If all valid

Enable sign-up button

Validation

For each text field

Check if valid

If all valid

Enable sign-up button

Check username field

Check first password field

Check second password field equals first

Check email address field

Validation

For each text field

Check if valid

If all valid

Enable sign-up button

Check username field

Check first password field

Check second password field equals first

Check email address field

Validation

Check username field

Check first password field

Check second password field equals first

Check email address field

Validation

Check username field

Check first password field

Check second password field equals first

Check email address field

Validation

Check username field

Check first password field

Check second password field equals first

Check email address field

Create a regular expression

Find matches in username

If have any matches

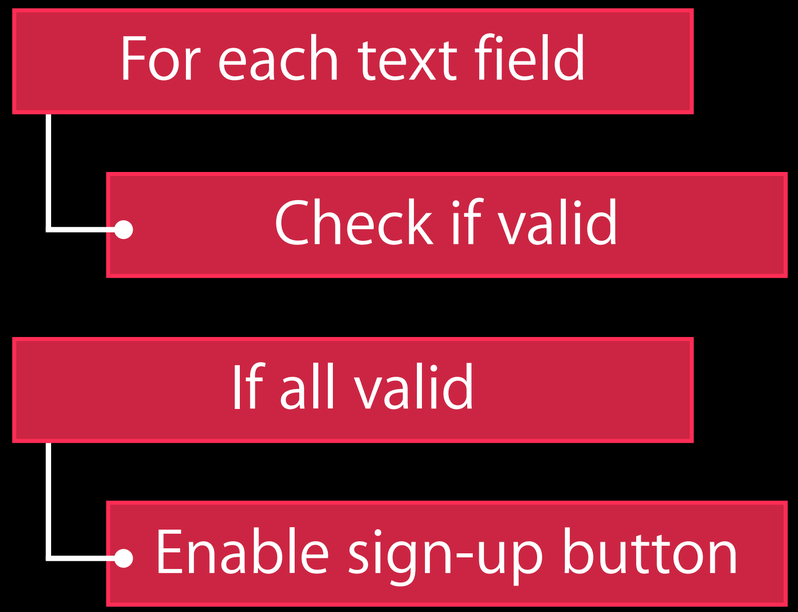
Username is valid

Otherwise

Decorate text field

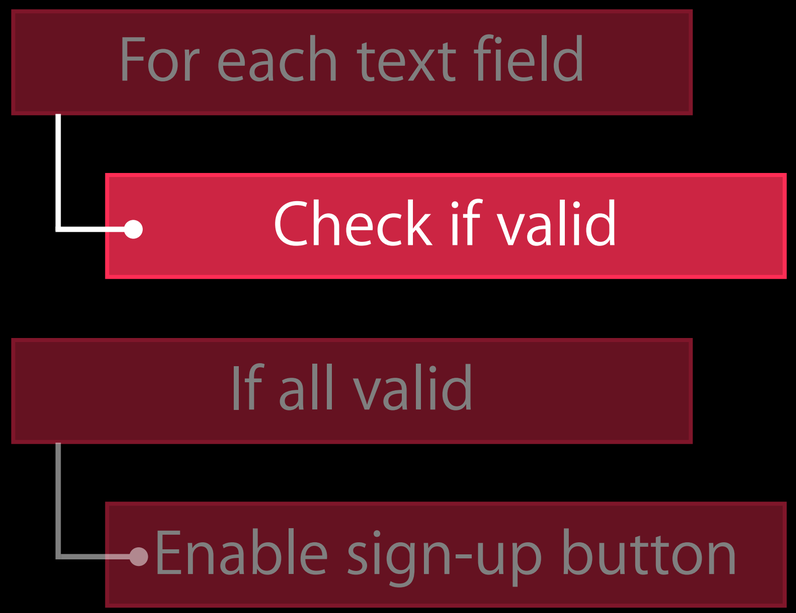
Validation

Handling nil



Validation

Handling nil



Validation

Handling nil

For each text field

Check if valid

If all valid

Enable sign-up button

Validation

Handling nil

For each text field

Check if valid

If all valid

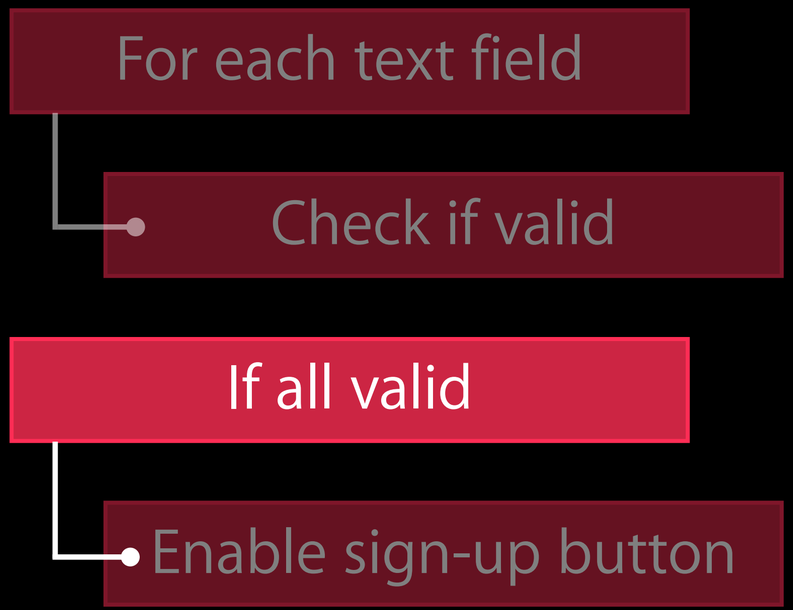
Enable sign-up button

If nil

Leave text field alone

Validation

Handling nil



Validation

Handling nil

For each text field

Check if valid

If all valid

Enable sign-up button

Validation

Handling nil

For each text field

Check if valid

If all valid

Enable sign-up button

If any nil

Leave sign-up button disabled

Validation

Check username field

Check first password field

Check second password field equals first

Check email address field

Create a regular expression

Find matches in username

If have any matches

Username is valid

Otherwise

Decorate text field

Validation

```
    NSString *username = [self.usernameField text];
    NSRegularExpression *regex = [NSRegularExpression
regularExpressionWithPattern:@“[a-zA-Z0-9_]{6,}” options:0 error:nil];
    NSRange result = [regex rangeOfFirstMatchInString:username
options:NSMatchingAnchored range:NSMakeRange(0, [username length])];
    if (username && result.location == NSNotFound) {
        allValid = NO;
        [self.usernameField setBackgroundColor:[UIColor redColor]];
    } else {
        if (!username) {
            anyNil = YES;
        }
        [self.usernameField setBackgroundColor:[UIColor whiteColor]]];
    }
```

Validation

```
    NSString *username = [self.usernameField text];
    NSRegularExpression *regex = [NSRegularExpression
regularExpressionWithPattern:@“[a-zA-Z0-9_]{6,}” options:0 error:nil];
    NSRange result = [regex rangeOfFirstMatchInString:username
options:NSMatchingAnchored range:NSMakeRange(0, [username length])];
    if (username && result.location == NSNotFound) {
        allValid = NO;
        [self.usernameField setBackgroundColor:[UIColor redColor]];
    } else {
        if (!username) {
            anyNil = YES;
        }
        [self.usernameField setBackgroundColor:[UIColor whiteColor]]];
    }
```

View Controller A

Text Field

Text Field

Text Field

Button

View Controller A

Validation

Text Field

Text Field

Text Field

Button

View Controller A

Validation

Text Field

Text Field

Text Field

Button

View Controller B

Validation

Text Field

Text Field

Button

View Controller A

Text Field

Text Field

Text Field

Button

Validation

View Controller B

Text Field

Text Field

Button

Validation

Validation

What is validation?

Validation

What is validation?

Inputs	Outputs

Validation

What is validation?

- Given an input value

Inputs	Outputs
inputValue: Any	

Validation

What is validation?

- Given an input value
- Is input value valid?

Inputs	Outputs
inputValue: Any	isValid: Bool

Validation

What is validation?

- Given an input value
- Is input value valid?
- If not, why not?

Inputs	Outputs
inputValue: Any	isValid: Bool
	error: NSError?

Validation

Foundations

Validation

Foundations

```
protocol Validator {
    validateWithError(error: NSErrorPointer) -> Bool
}
```

Validation

Foundations

```
protocol Validator {
    validateWithError(error: NSErrorPointer) -> Bool
}
```

Definition of “input” left open to interpretation

Validation

Foundations

```
protocol Validator {
    validateWithError(error: NSErrorPointer) -> Bool
}
```

Definition of “input” left open to interpretation

Build larger validators out of smaller ones

Validation

Foundations

```
protocol Validator {
    validateWithError(error: NSErrorPointer) -> Bool
}
```

Definition of “input” left open to interpretation

Build larger validators out of smaller ones

Composition

Validation

Foundations

```
protocol Validator {
    validateWithError(error: NSErrorPointer) -> Bool
}
```

Definition of “input” left open to interpretation

Build larger validators out of smaller ones

Composition

Works fine in Objective-C, too

Validation

Usernames

Validation

Usernames

```
class UsernameValidator: Validator {
    var input: NSString?

    func validateWithError(error: NSErrorPointer) -> Bool {
        let regex = NSRegularExpression(pattern: ...)
        ...
    }
}
```

Validation

Passwords

Validation

Passwords

```
class PasswordValidator: Validator {
    var input: NSString?
    ...
}
```

Validation

Passwords

```
class PasswordValidator: Validator {
    var input: NSString?
    ...
}
```

Represents the validity of a single password field

Validation

Passwords

```
class PasswordValidator: Validator {
    var input: NSString?
    ...
}
```

Represents the validity of a single password field

```
class SetPasswordValidator: Validator {
    let firstPasswordValidator = PasswordValidator()
    let secondPasswordValidator = PasswordValidator()
    ...
}
```

Validation

Passwords

```
class PasswordValidator: Validator {
    var input: NSString?
    ...
}
```

Represents the validity of a single password field

```
class SetPasswordValidator: Validator {
    let firstPasswordValidator = PasswordValidator()
    let secondPasswordValidator = PasswordValidator()
    ...
}
```

Represents two password fields that must match in value

Validation

Overall form

Validation

Overall form

```
class SignUpValidator: Validator {
    let usernameValidator = UsernameValidator()
    let setPasswordValidator = SetPasswordValidator()
    let emailAddressValidator = EmailAddressValidator()
    ...
}
```

Validation

Overall form

```
class SignUpValidator: Validator {
    let usernameValidator = UsernameValidator()
    let setPasswordValidator = SetPasswordValidator()
    let emailAddressValidator = EmailAddressValidator()
    ...
}
```

Represents the validity of the entire form

Validation

Overall form

```
class SignUpValidator: Validator {
    let usernameValidator = UsernameValidator()
    let setPasswordValidator = SetPasswordValidator()
    let emailAddressValidator = EmailAddressValidator()
    ...
}
```

Represents the validity of the entire form

Behavior of nil

Validation

Overall form

```
class SignUpValidator: Validator {
    let usernameValidator = UsernameValidator()
    let setPasswordValidator = SetPasswordValidator()
    let emailAddressValidator = EmailAddressValidator()
    ...
}
```

Represents the validity of the entire form

Behavior of nil

- Most validators allow nil

Validation

Overall form

```
class SignUpValidator: Validator {
    let usernameValidator = UsernameValidator()
    let setPasswordValidator = SetPasswordValidator()
    let emailAddressValidator = EmailAddressValidator()
    ...
}
```

Represents the validity of the entire form

Behavior of nil

- Most validators allow nil
- SignUpValidator verifies contained inputs are non-nil

Validation

Validation

For each text field

- Set input on corresponding validator

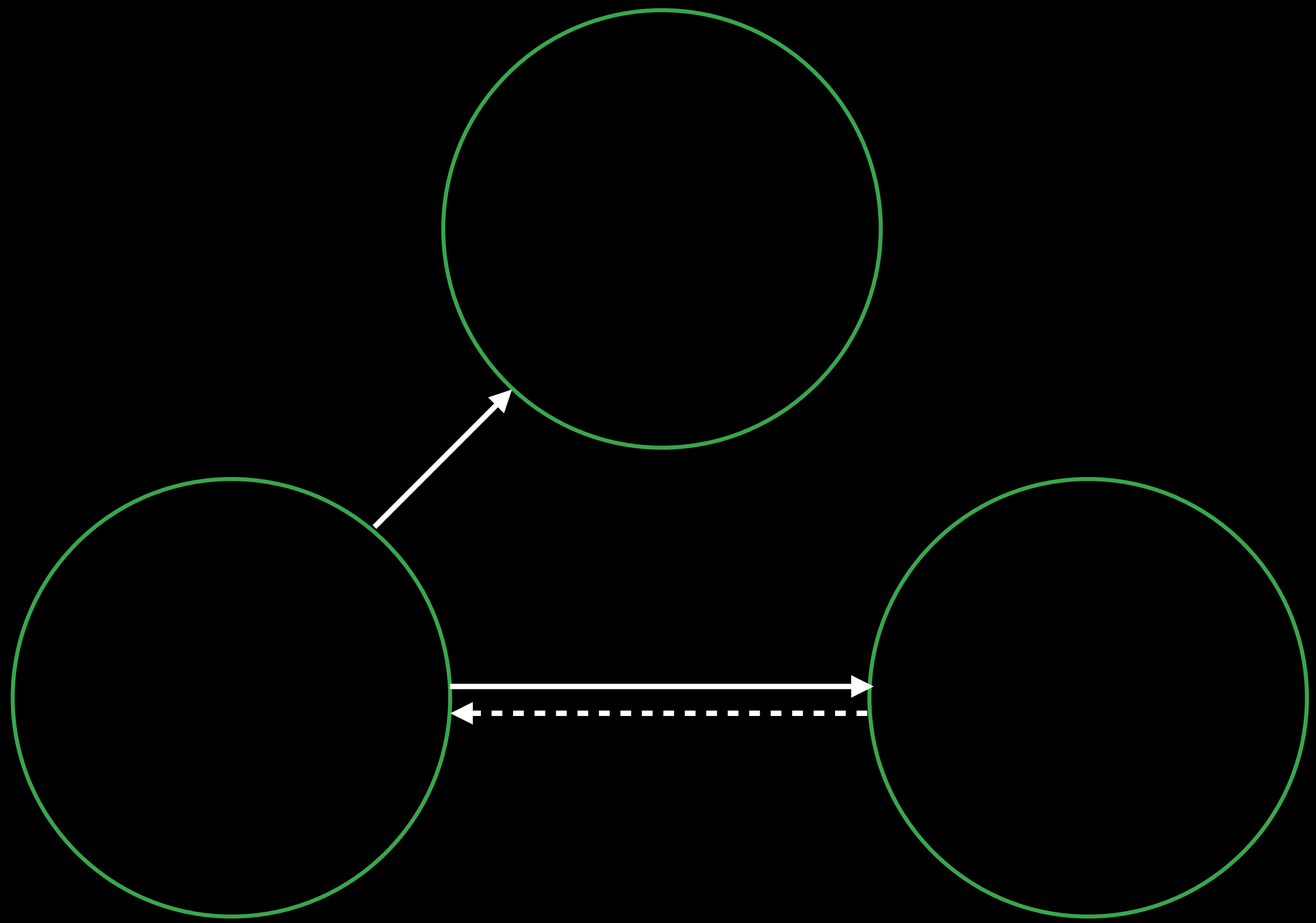
If sign-up validator OK

- Enable sign-up button

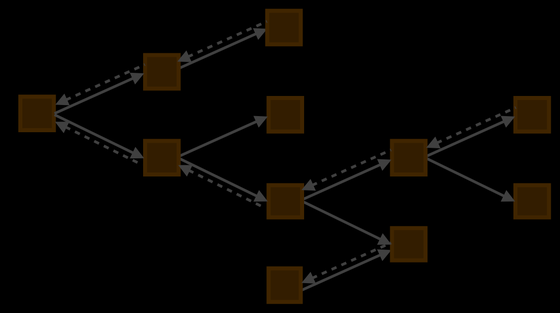
Otherwise

- Decorate invalid text fields

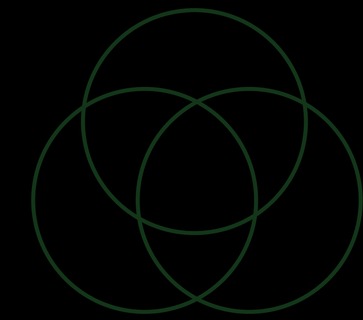
2 Define Clear Responsibilities



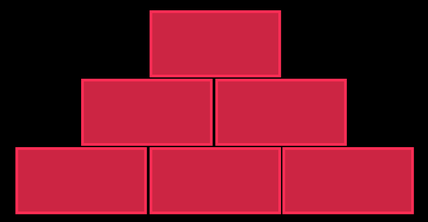
1 Design information flow



2 Define clear responsibilities



3 Simplify with immutability

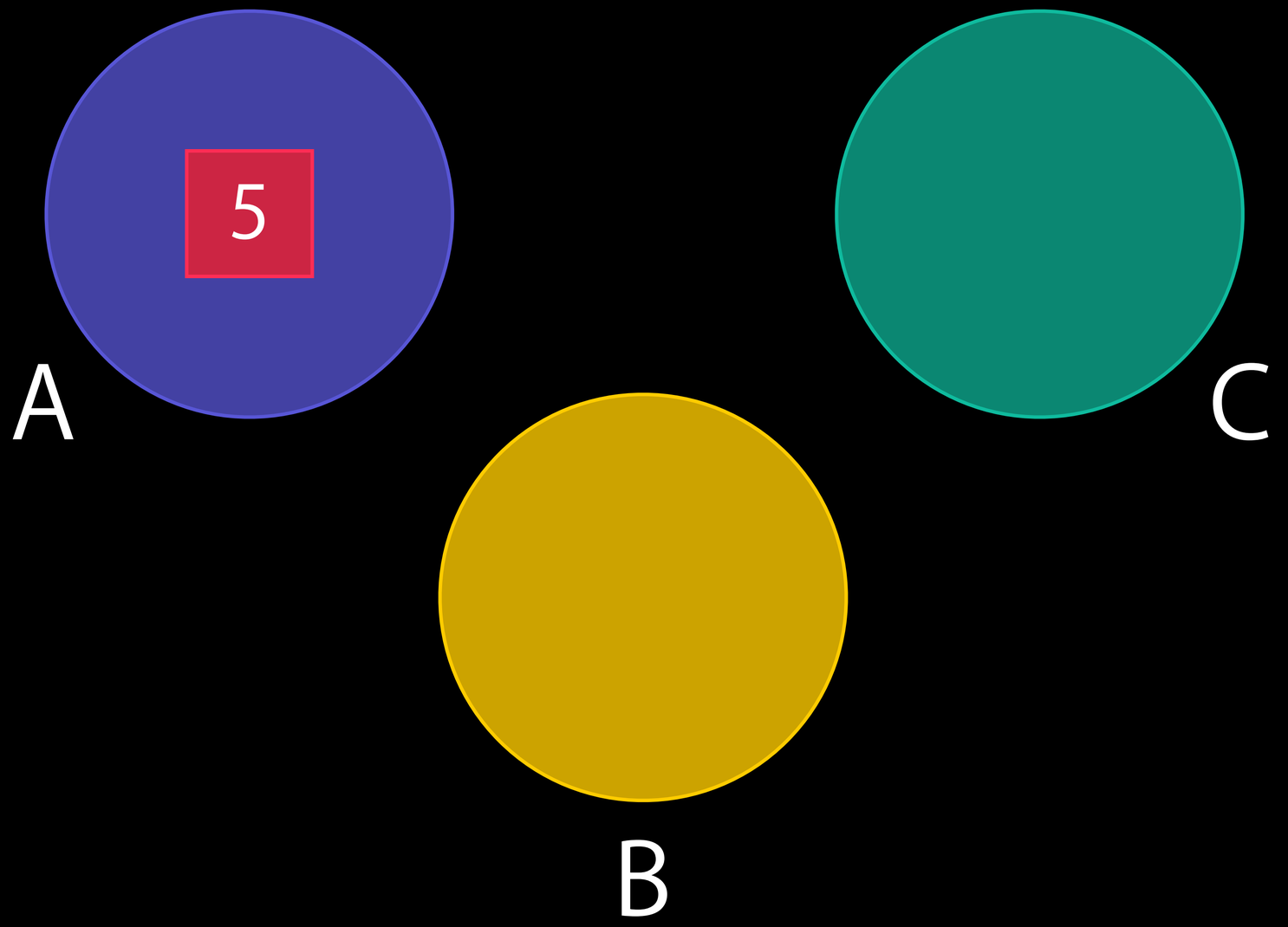


“Mutability is bad and you should feel bad for using it”

The Internet, probably…

Insight, not dogma

Why, not how

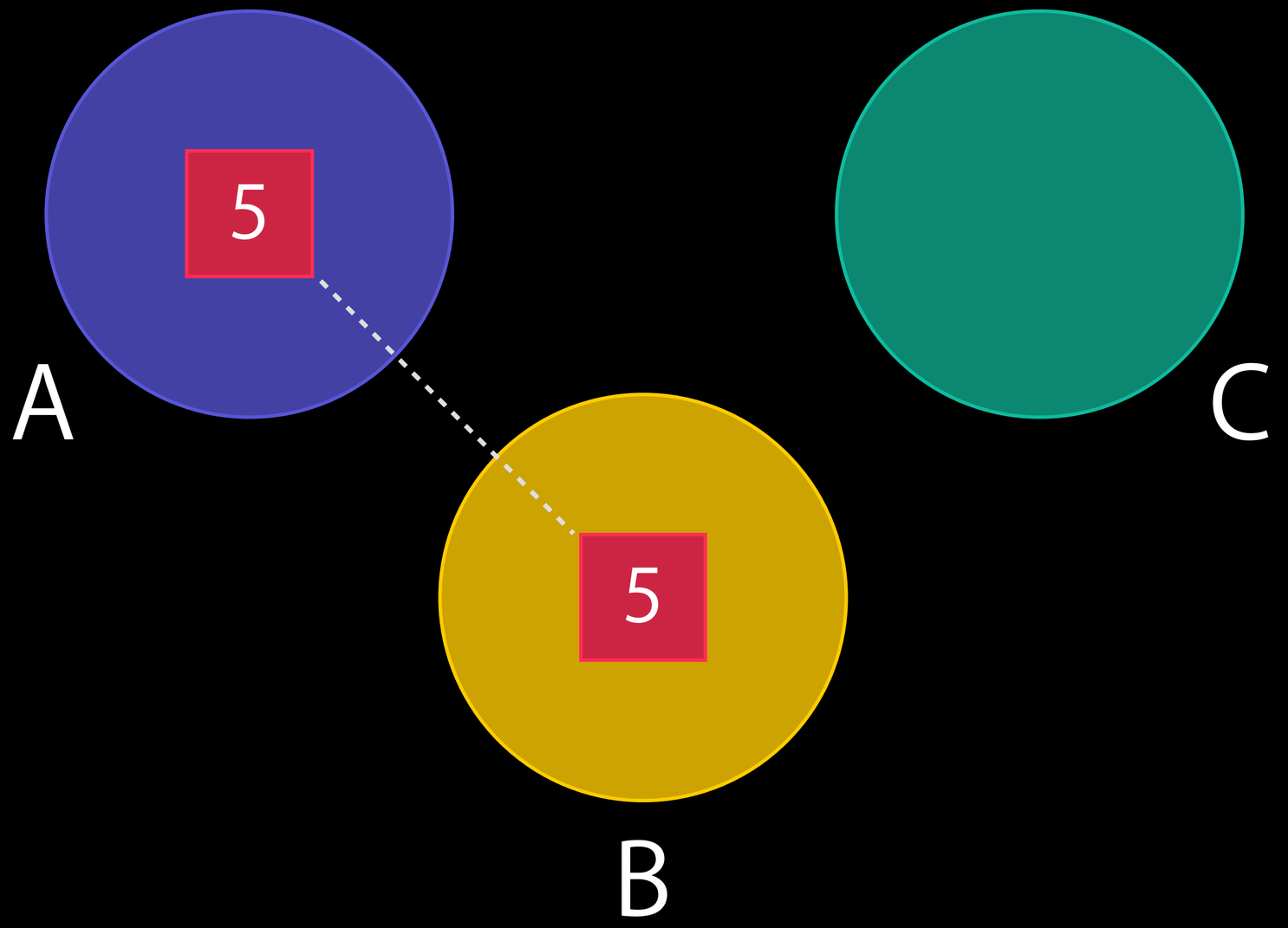

5
A
C
B

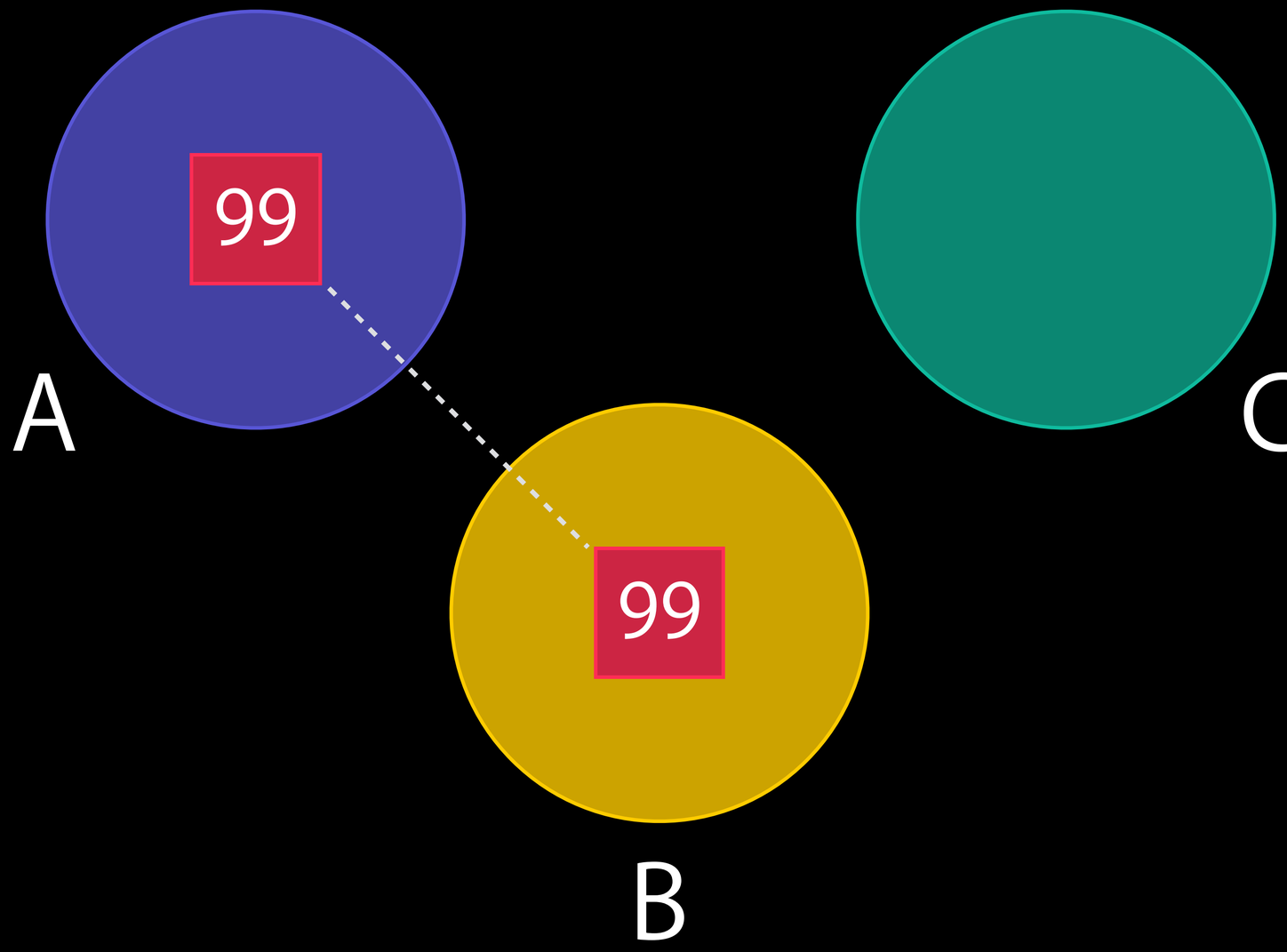
A

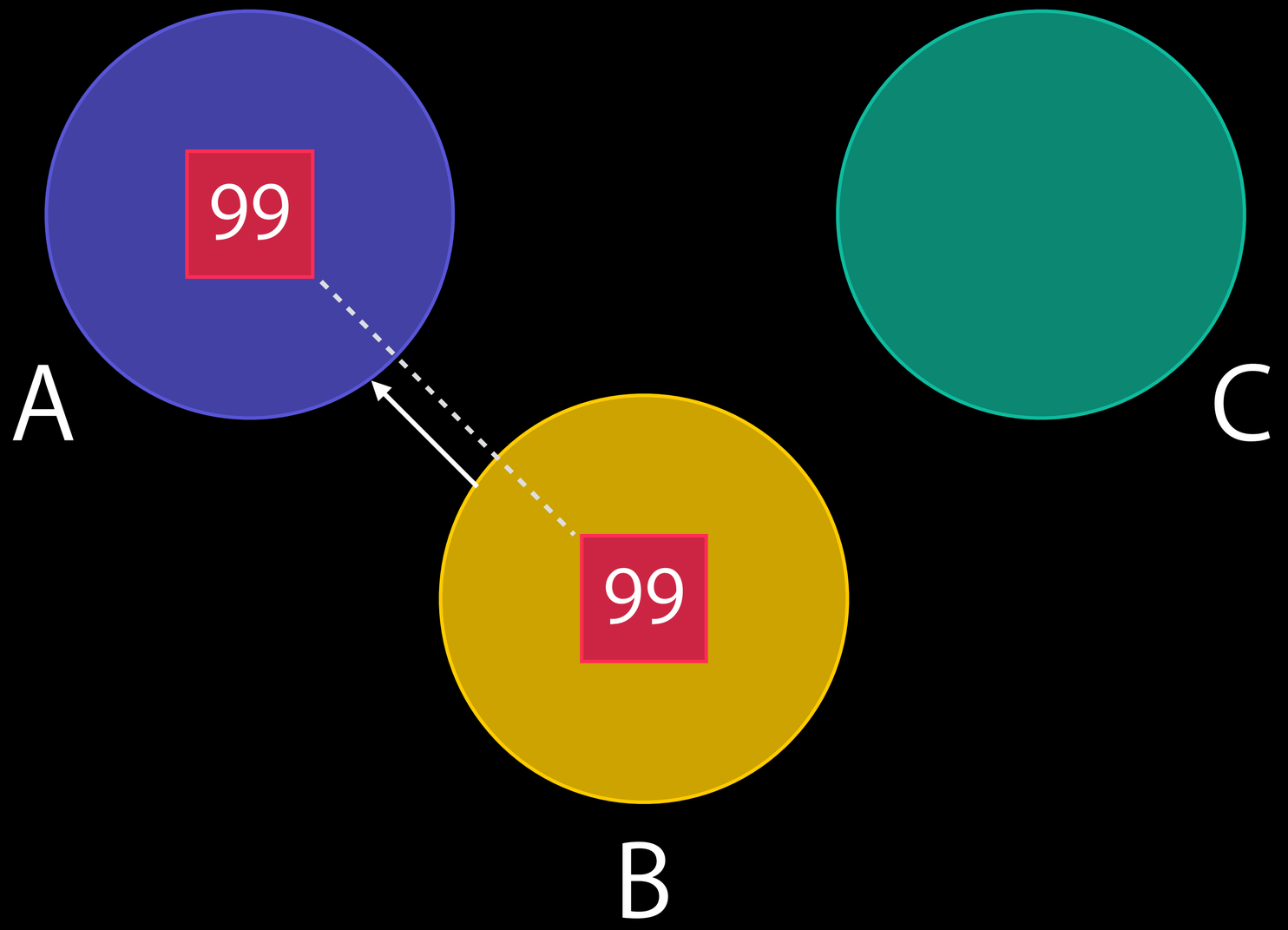
C

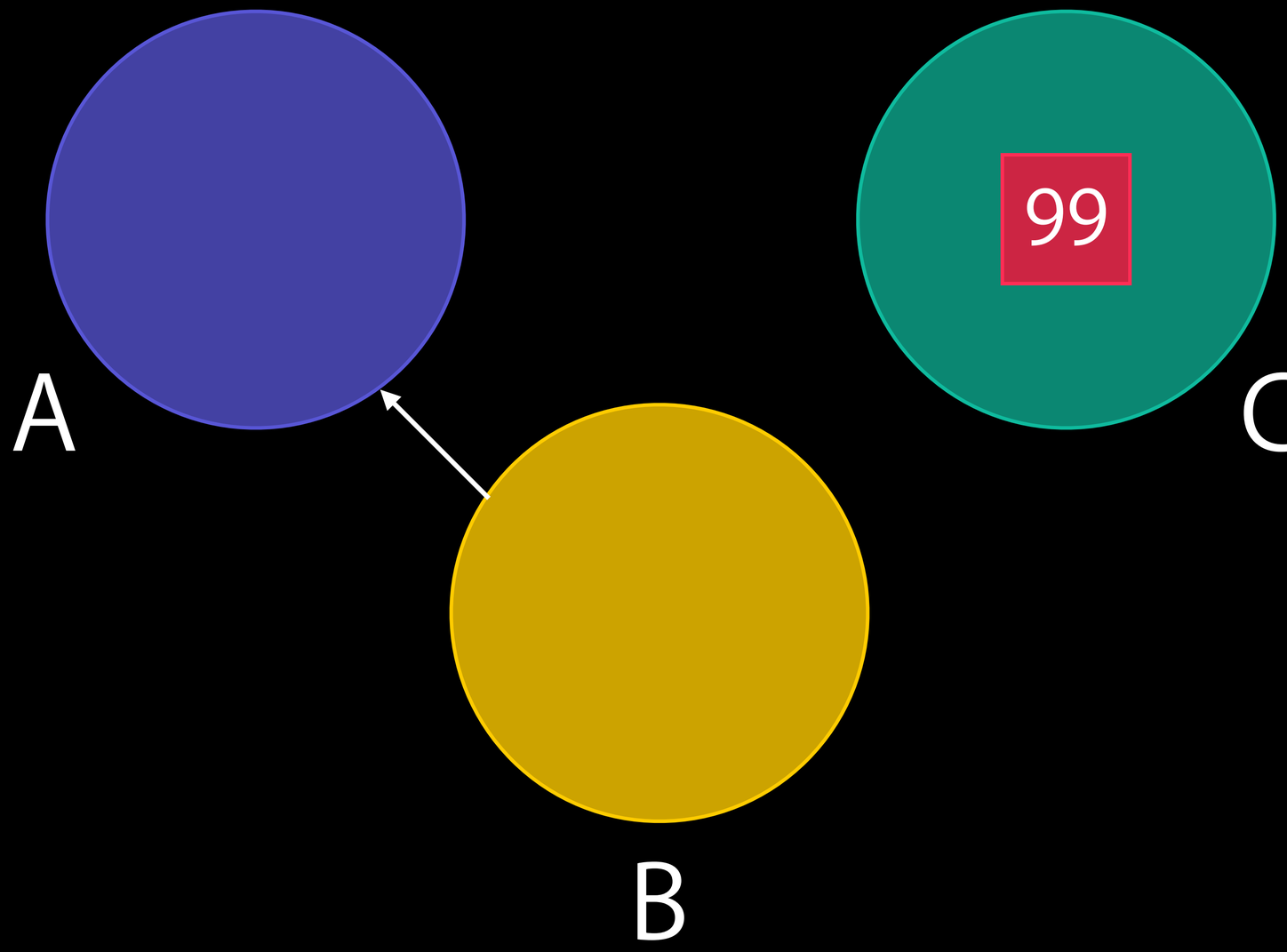
5

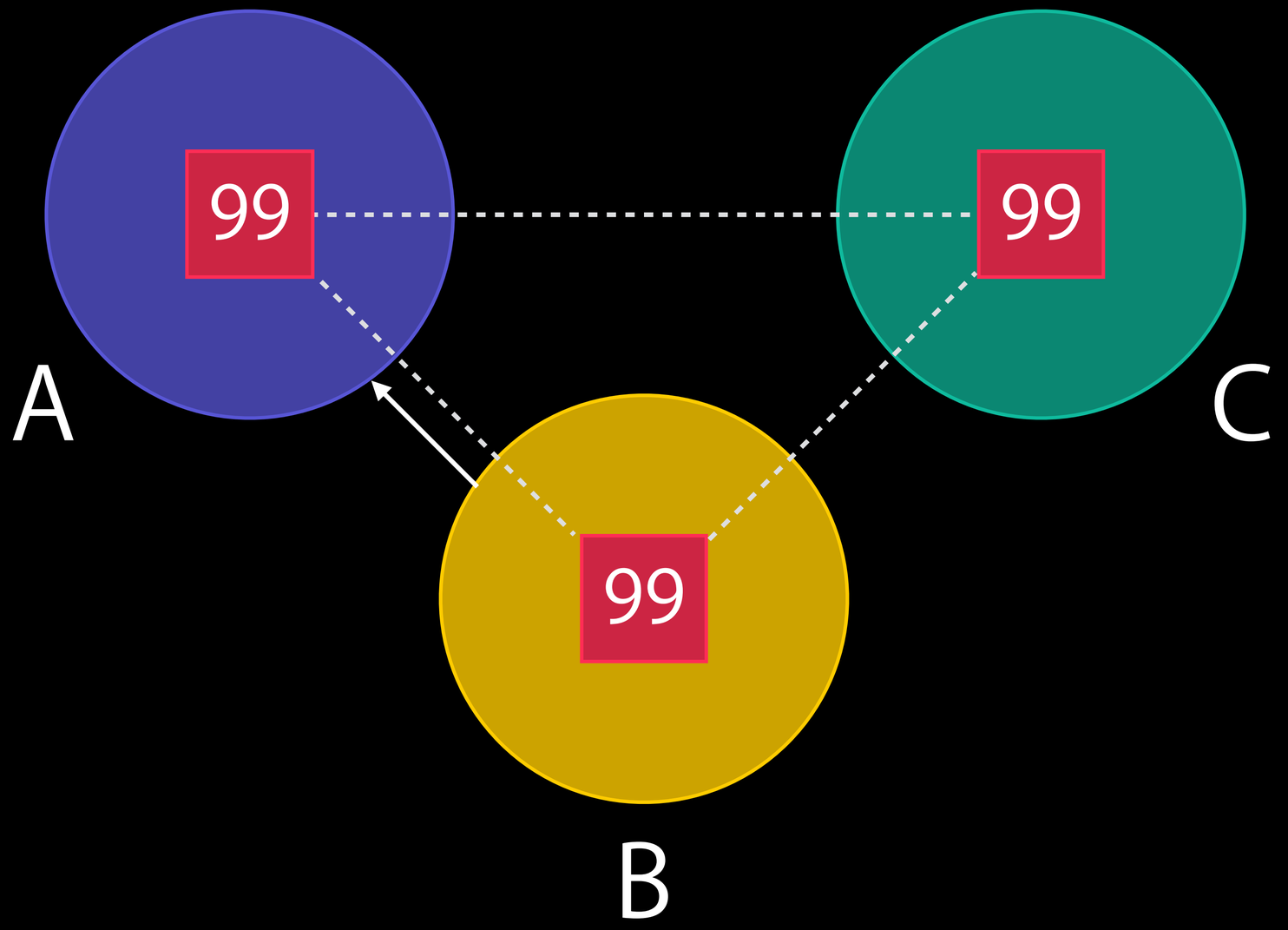
B

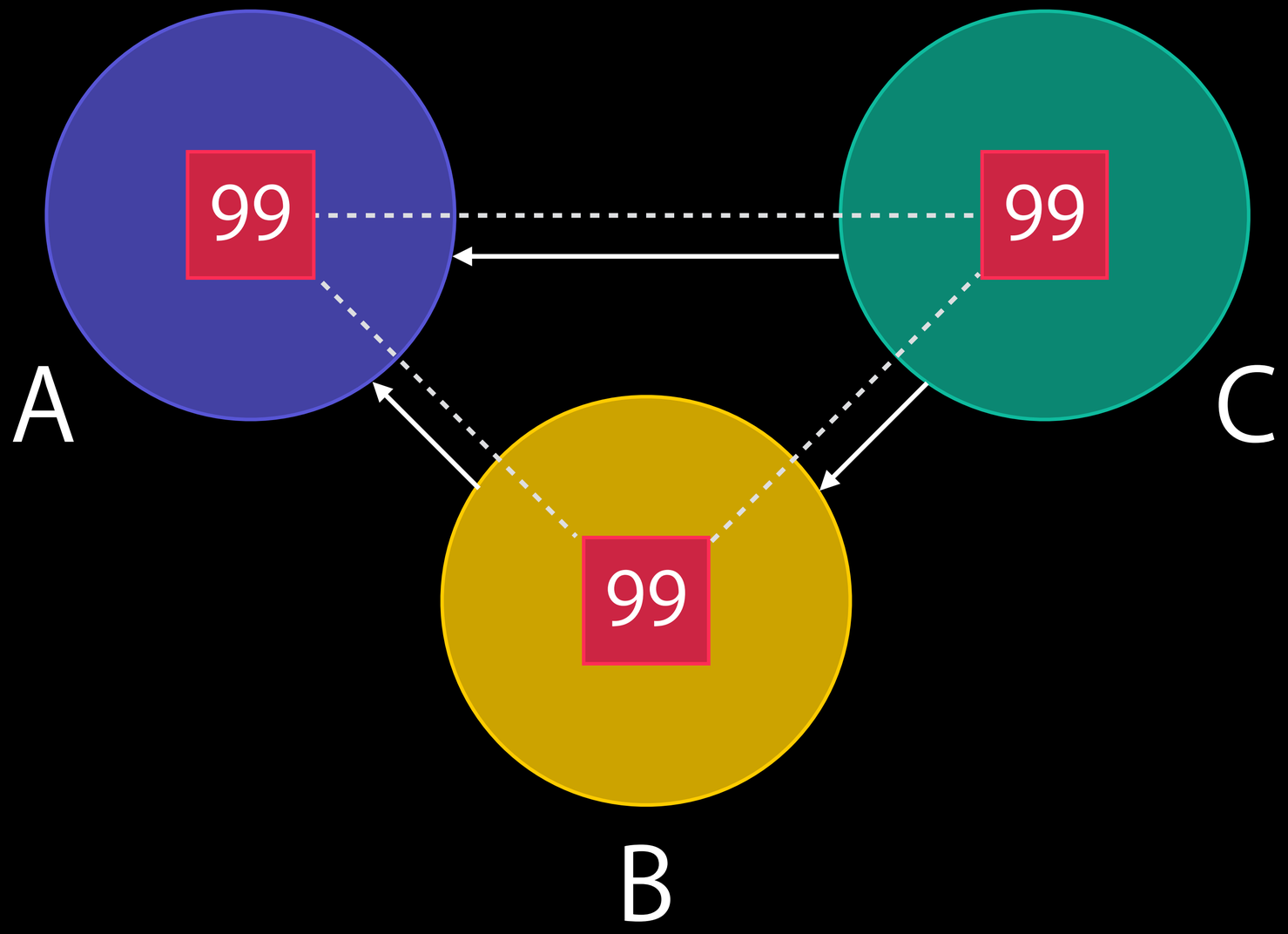

5
A
C
5
B

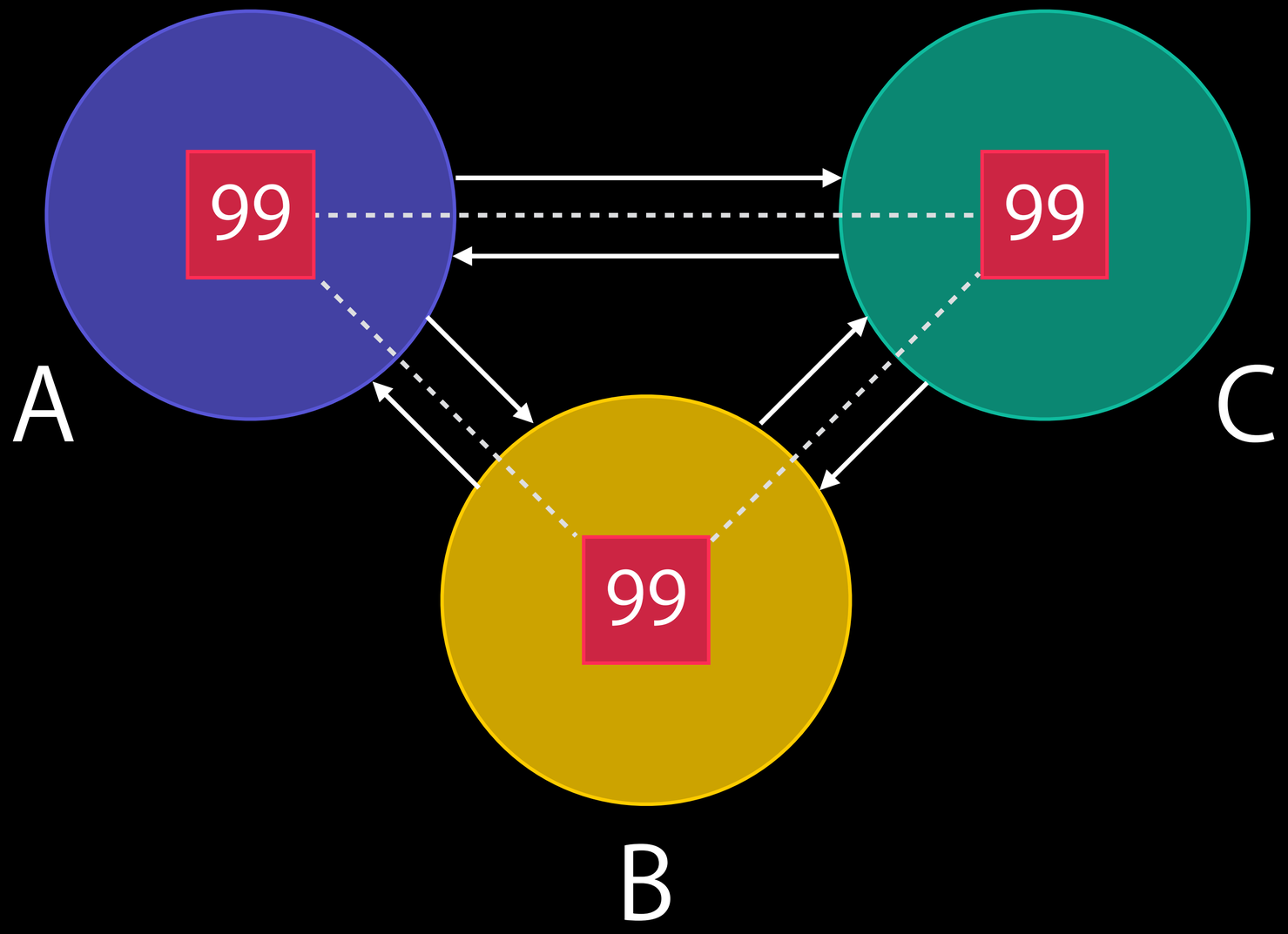

99
A
C
99
B

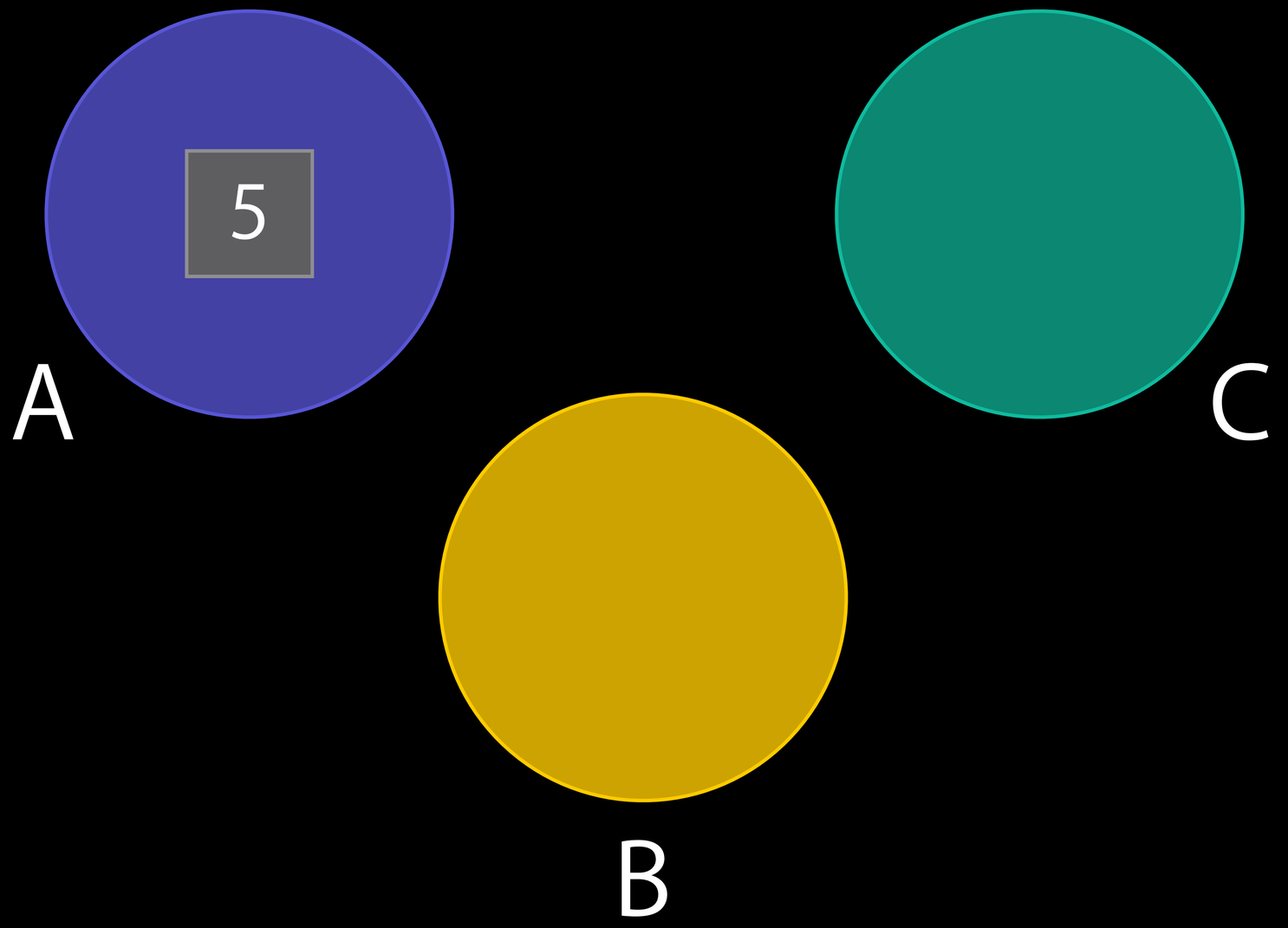

99
A
C
99
B

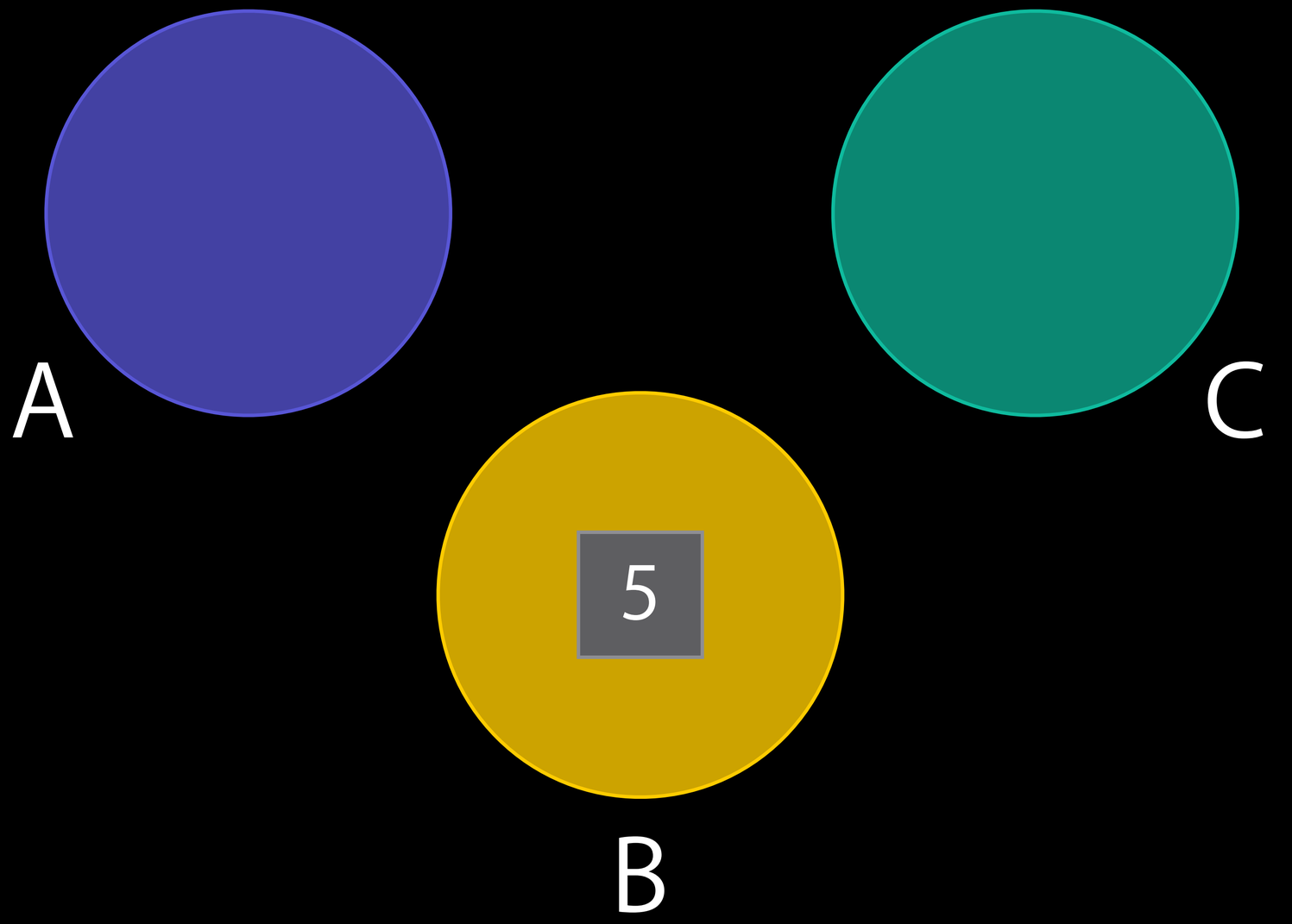

99
A
B
C

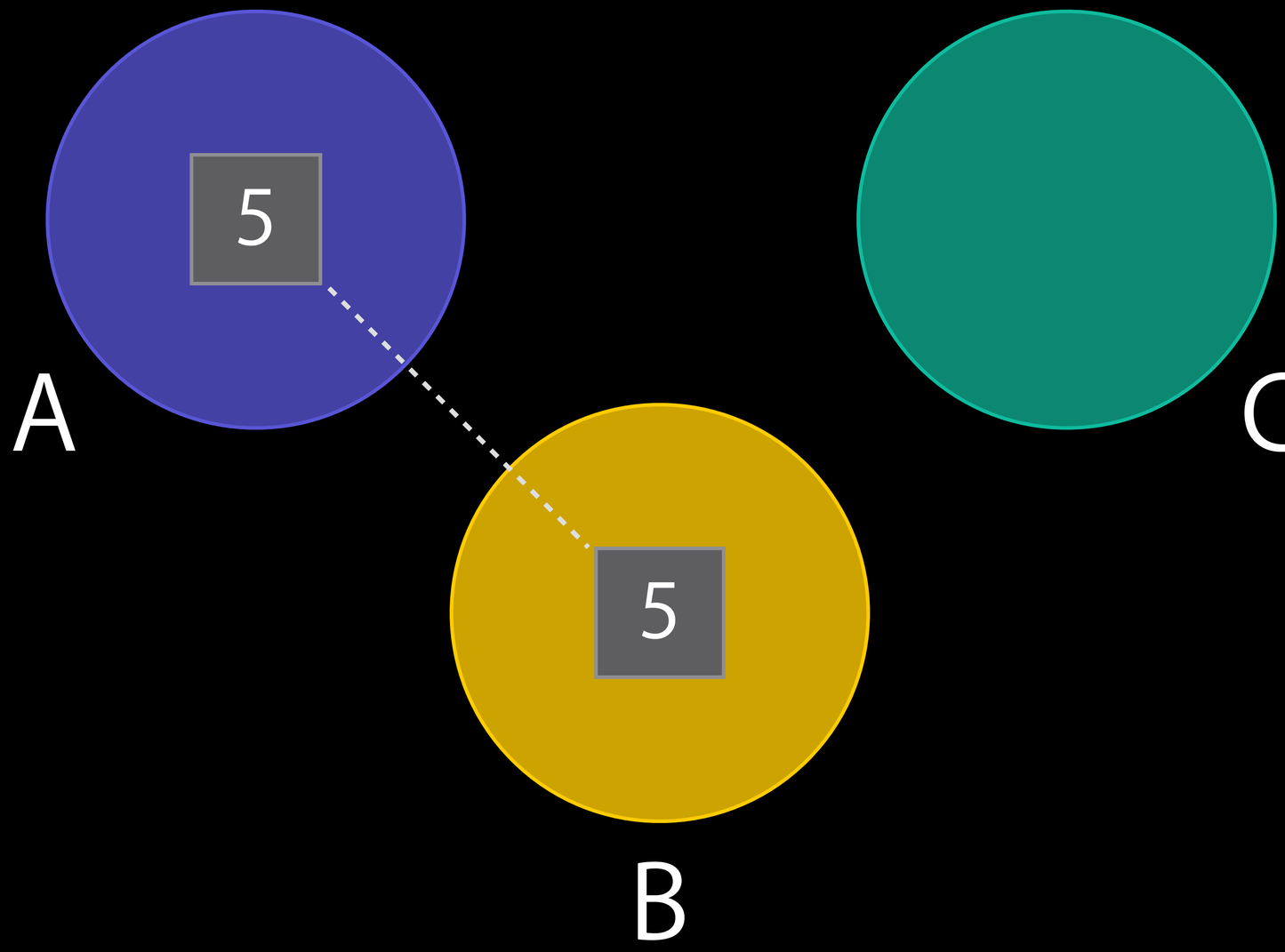

99
99
99
A
B
C

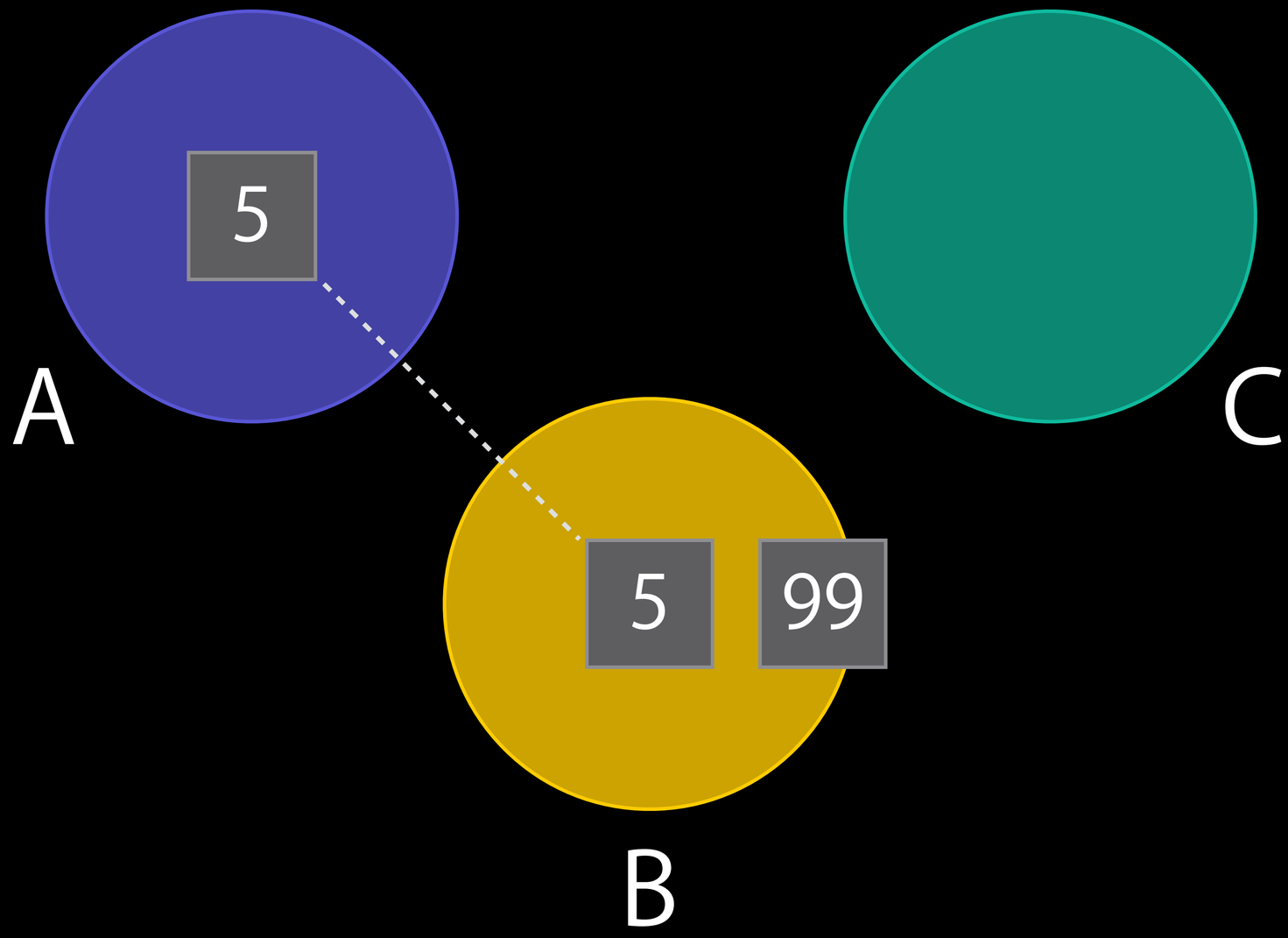

99
99
99
A
C
B

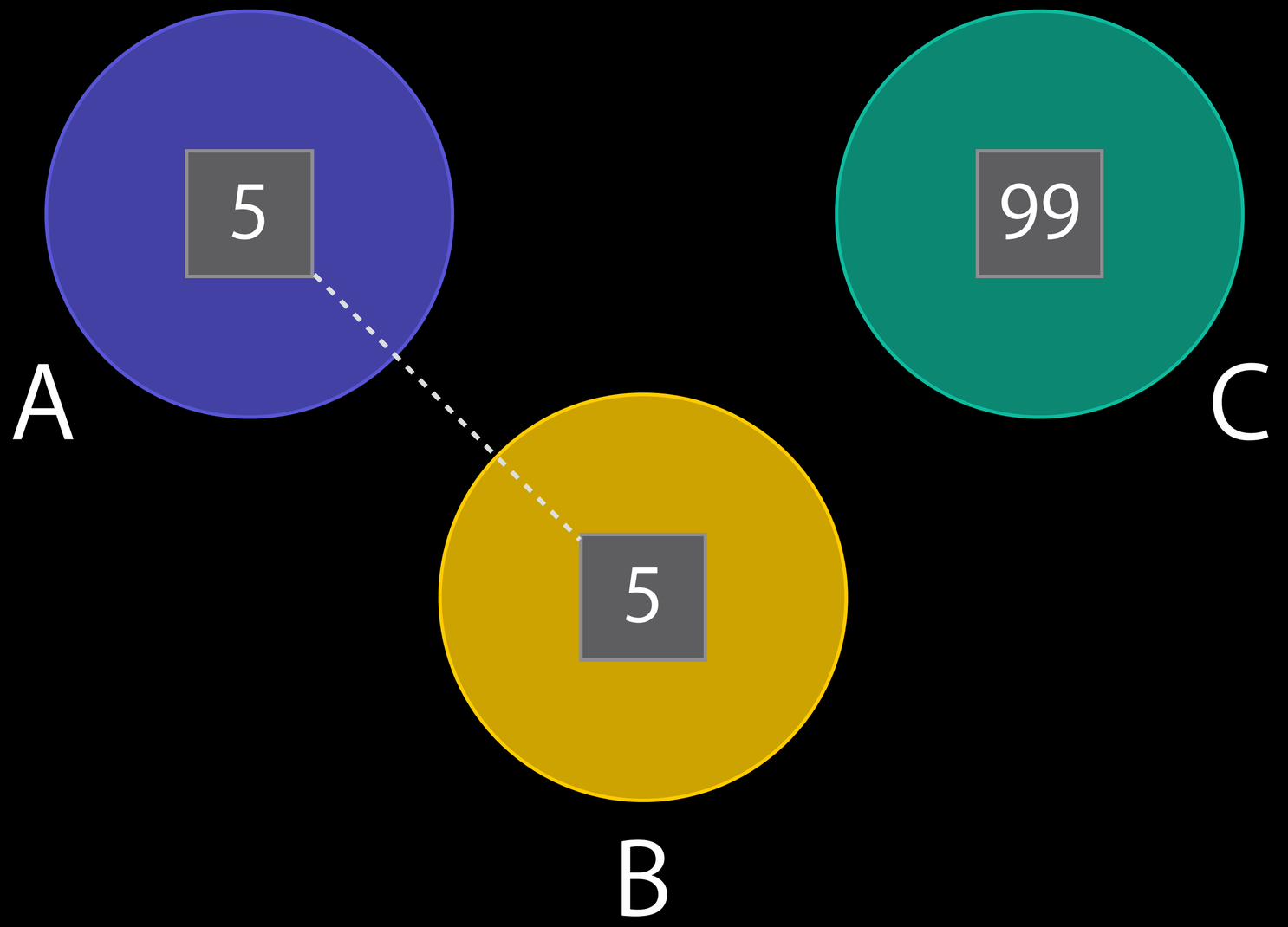

99
99
99
A
B
C


5
A
B
C


A
C
5
B


5
A
C
5
B


5
A
C
5
99
B


5
99
A
C
5
B

Swift Structs

Best of both worlds

Swift Structs

Best of both worlds

Opt-in mutability

- `mutating` keyword

Swift Structs

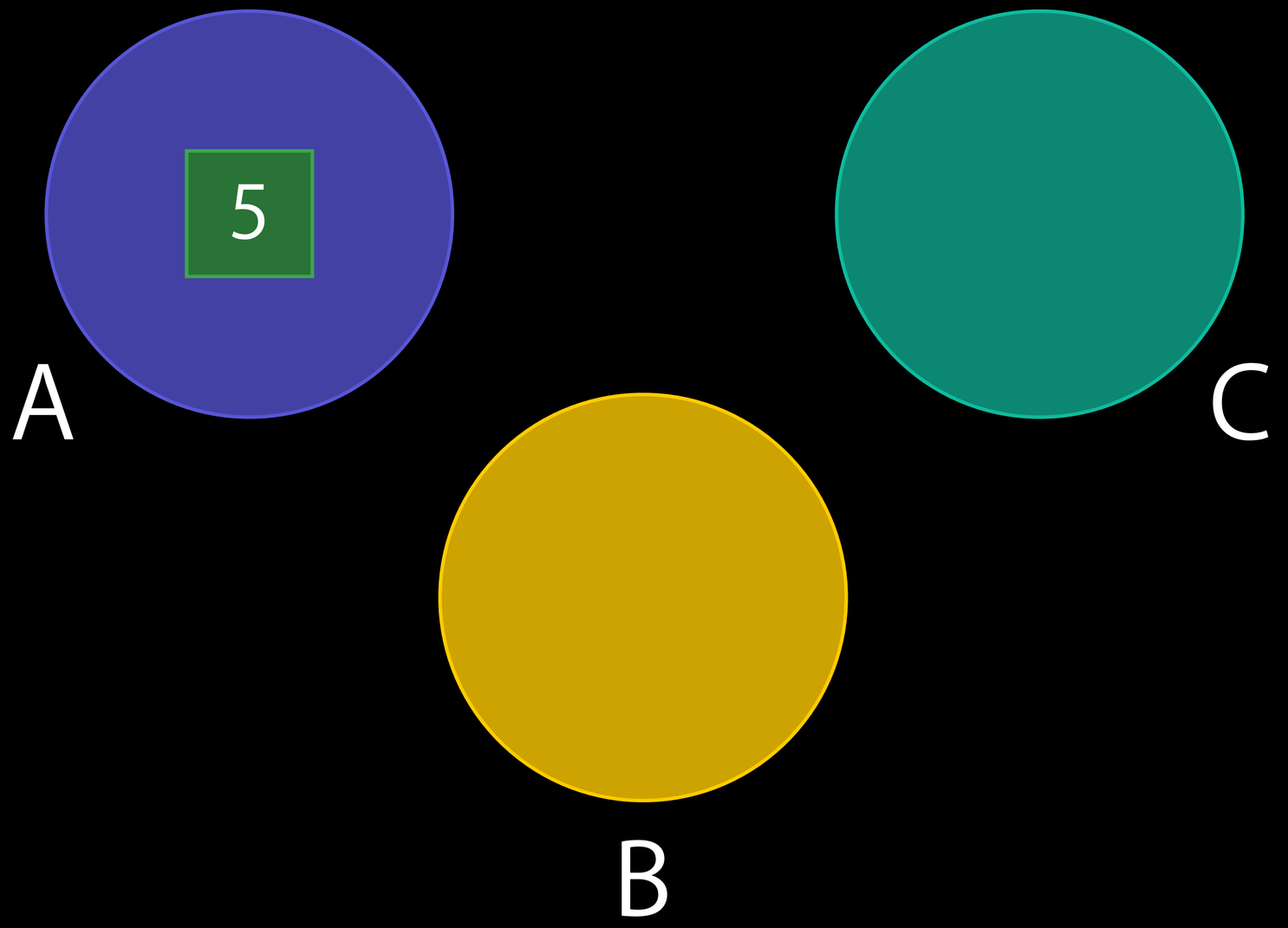
Best of both worlds

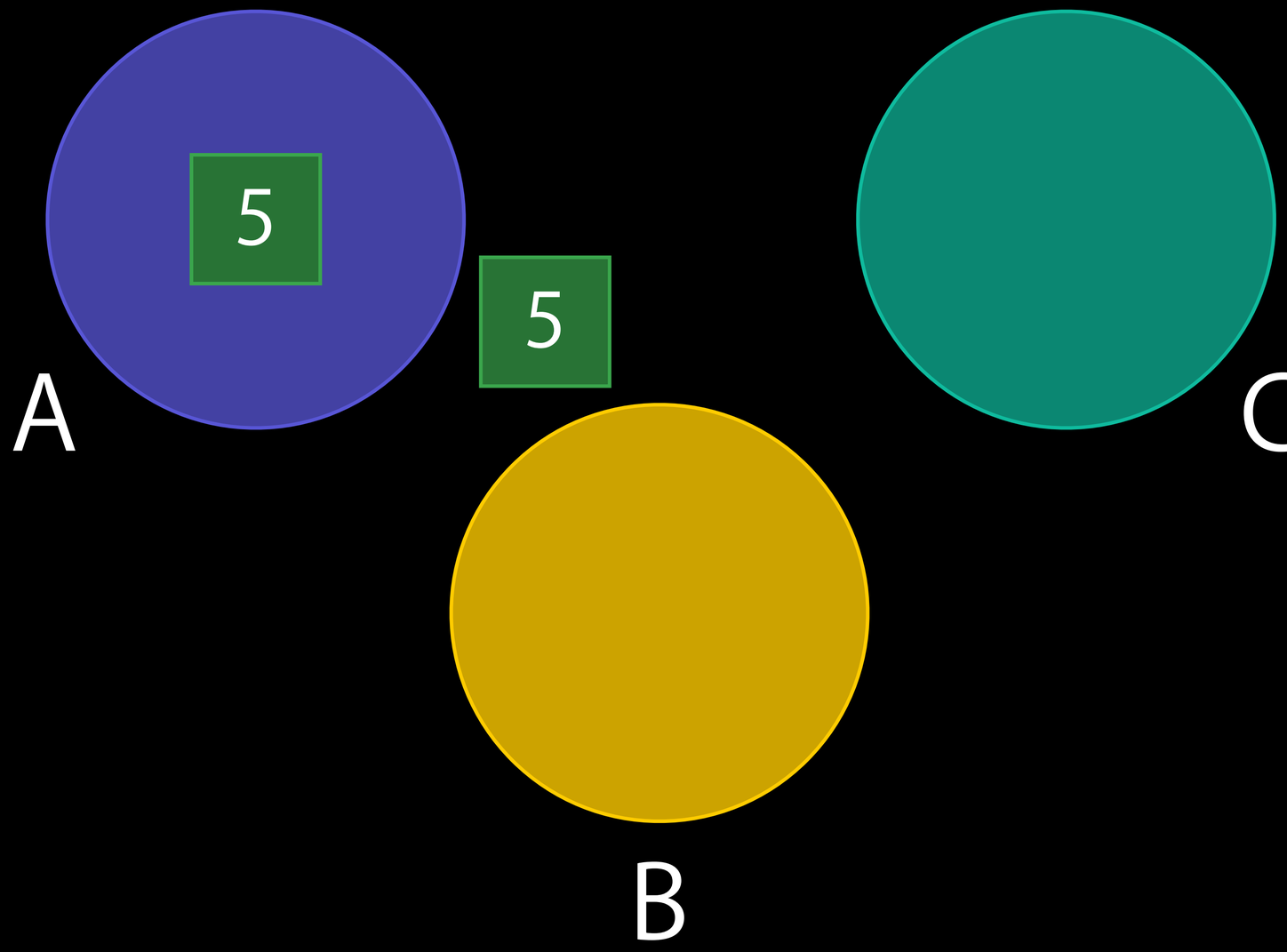
Opt-in mutability

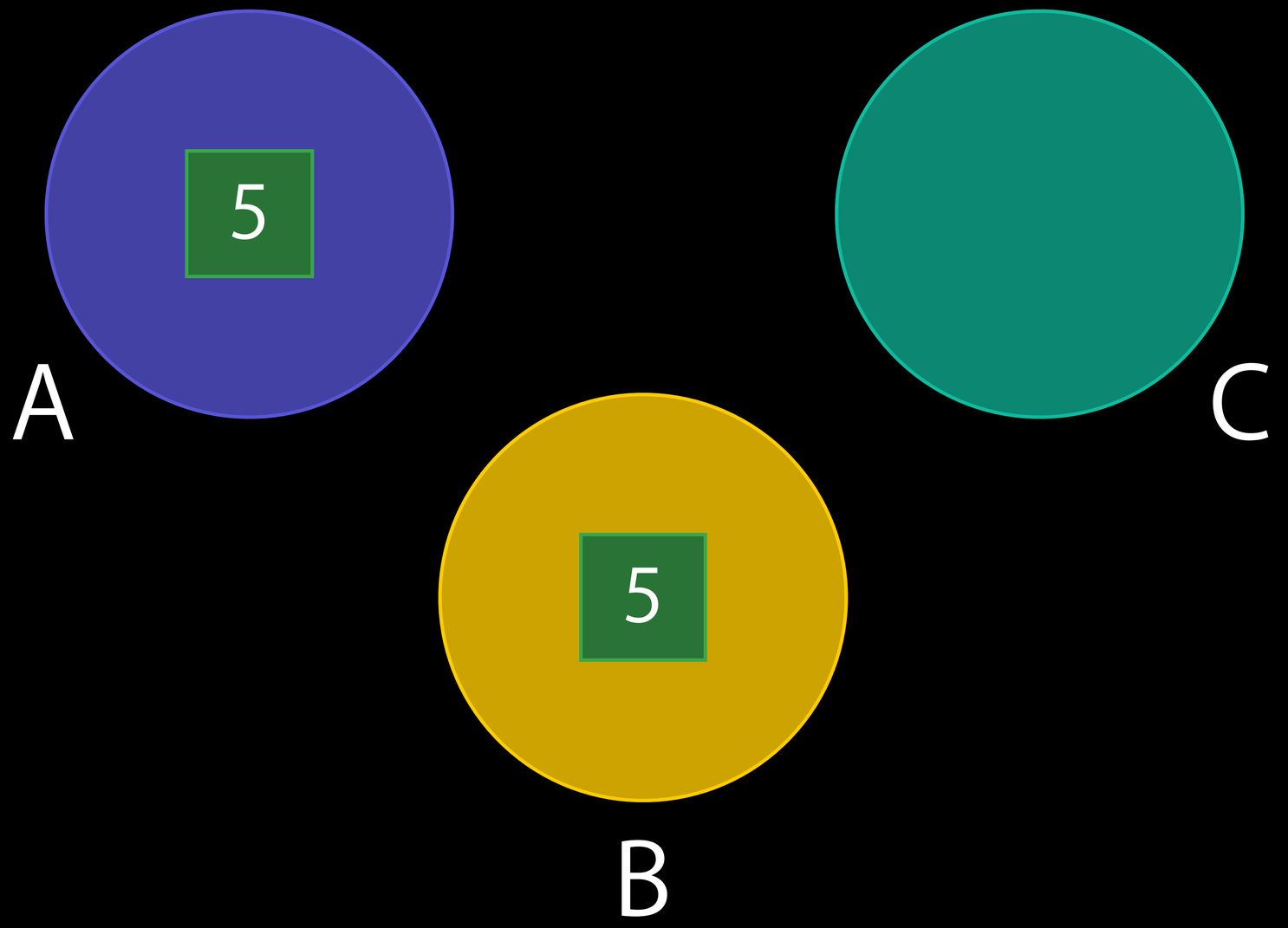
- `mutating` keyword

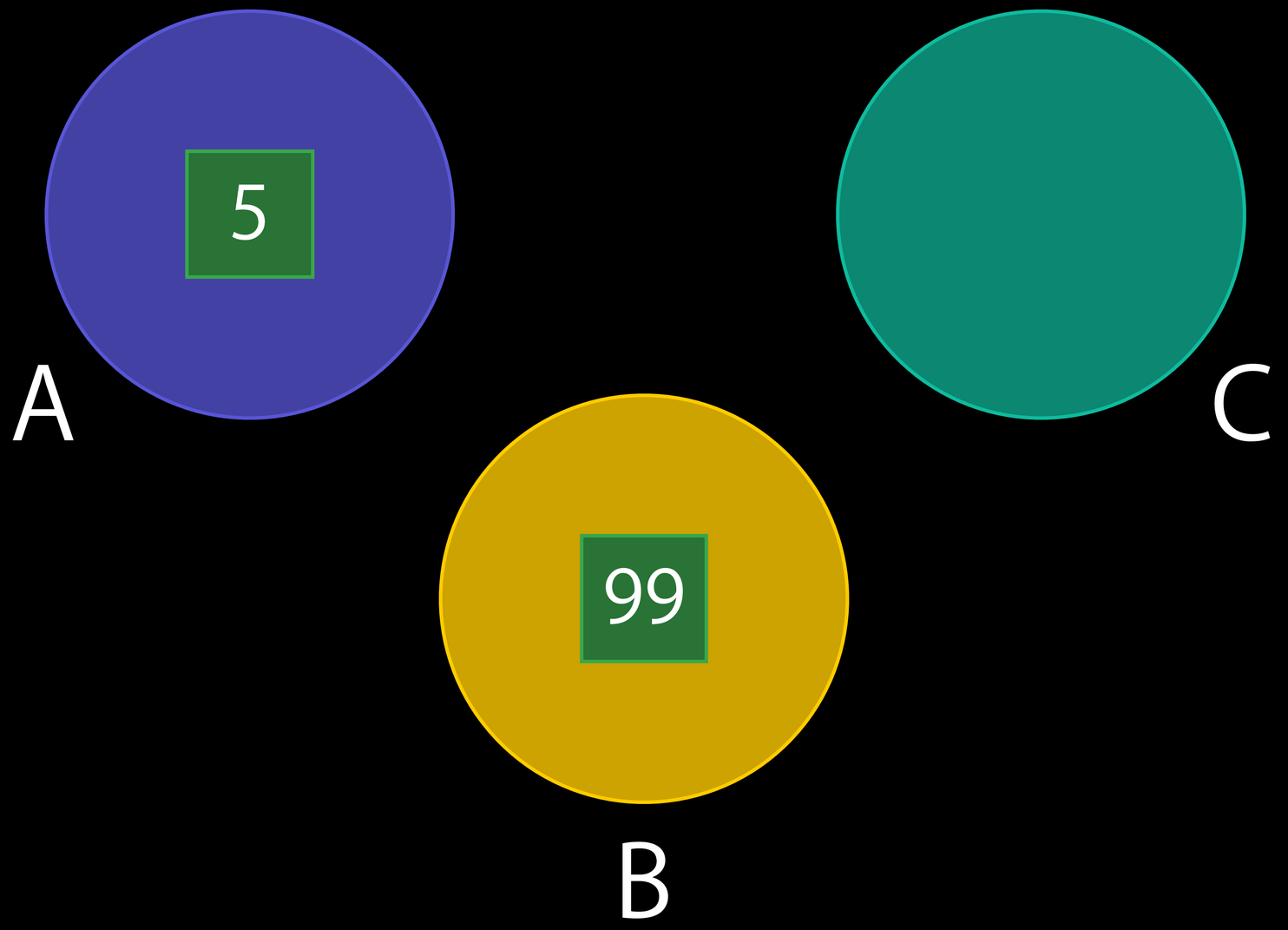
Call-by-value

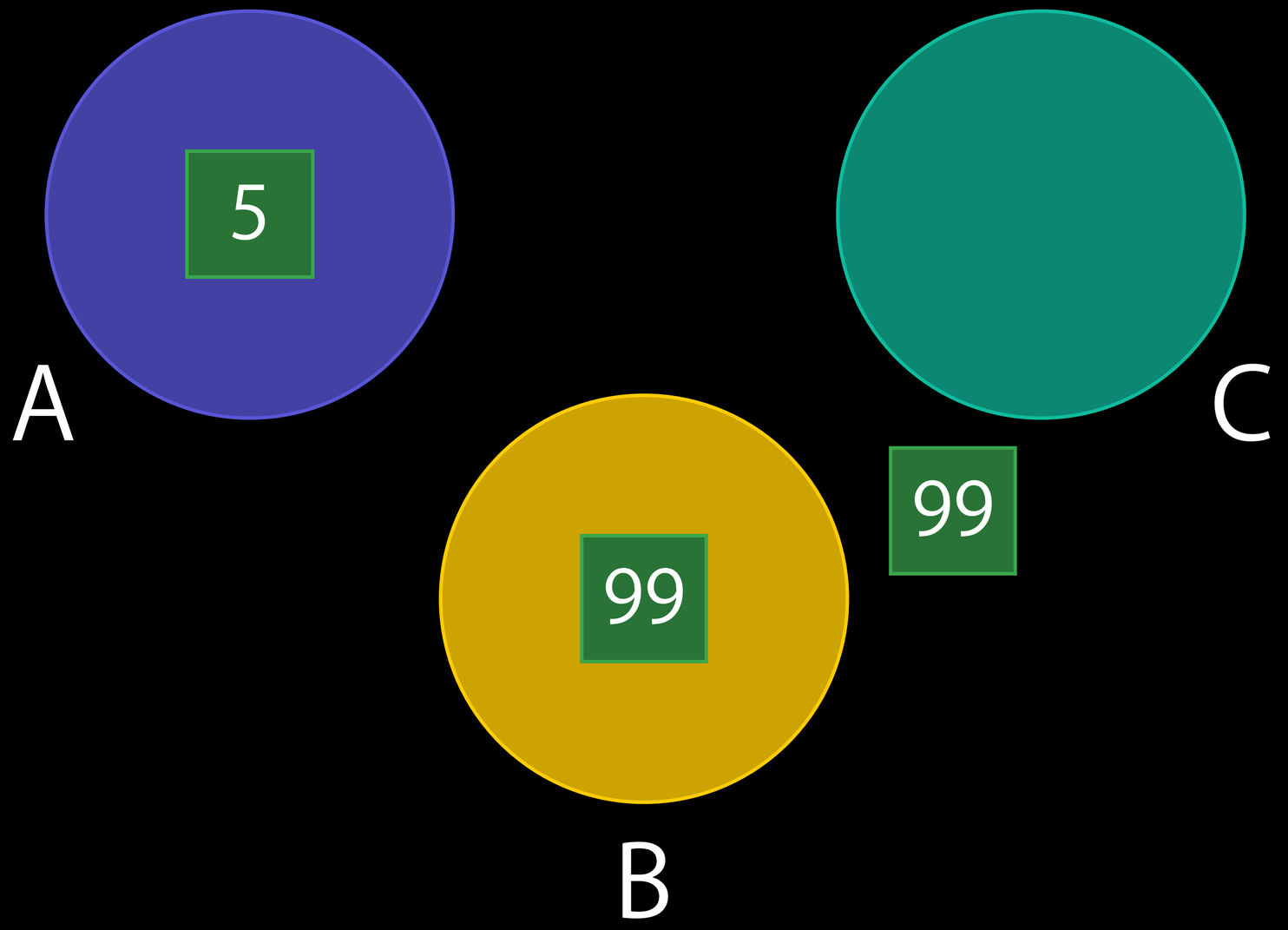
- New copy automatically created when passing struct to another function

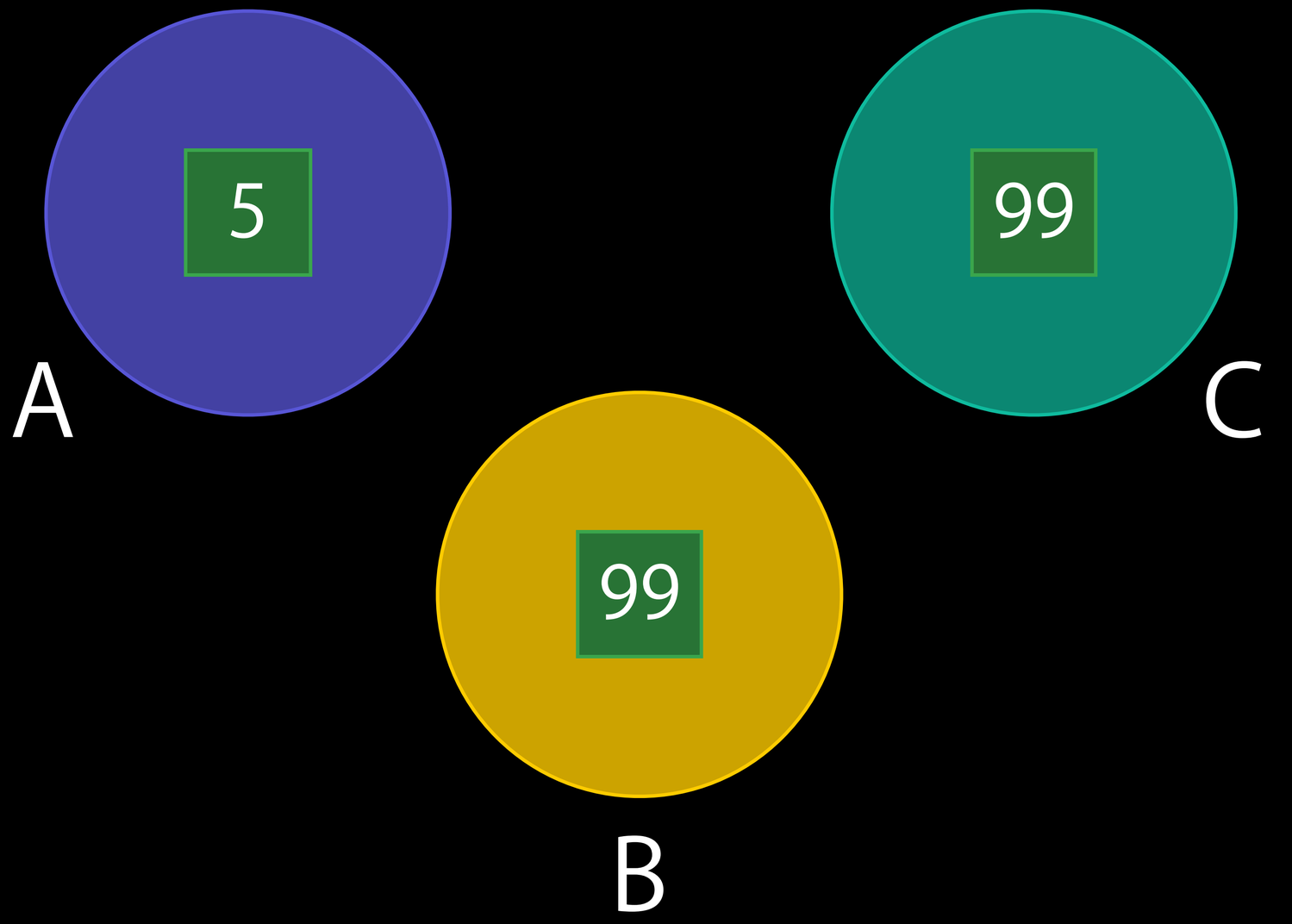

5
A
C
B


5
5
A
C
B


5
A
C
5
B


5
A
C
99
B


5
A
C
99
99
B


5
A
99
C
99
B

UIMotionEffect

UIMotionEffect

Adjust views based on gyroscope data

UIMotionEffect

Adjust views based on gyroscope data

Used to achieve parallax effects

UIMotionEffect

Adjust views based on gyroscope data

Used to achieve parallax effects

Reusable across views

UIMotionEffect

Adjust views based on gyroscope data

Used to achieve parallax effects

Reusable across views

Low-latency requirements

UIMotionEffect

UIMotionEffect

Inputs	Outputs

UIMotionEffect

Inputs	Outputs
Device pose	

UIMotionEffect

Inputs	Outputs
Device pose	Relative offset for each key path

UIMotionEffect

Input deltas

Inputs

Outputs

UIMotionEffect

Input deltas

Inputs	Outputs
Device pose deltas	

UIMotionEffect

Input deltas

Inputs	Outputs
Device pose deltas	Relative offset for each key path

UIMotionEffect

Input deltas

Inputs	Outputs
Device pose deltas	Relative offset for each key path
Previous poses	

UIMotionEffect

Absolute offsets

Inputs	Outputs

UIMotionEffect

Absolute offsets

Inputs	Outputs
Device pose	

UIMotionEffect

Absolute offsets

Inputs	Outputs
Device pose	Value for each key path

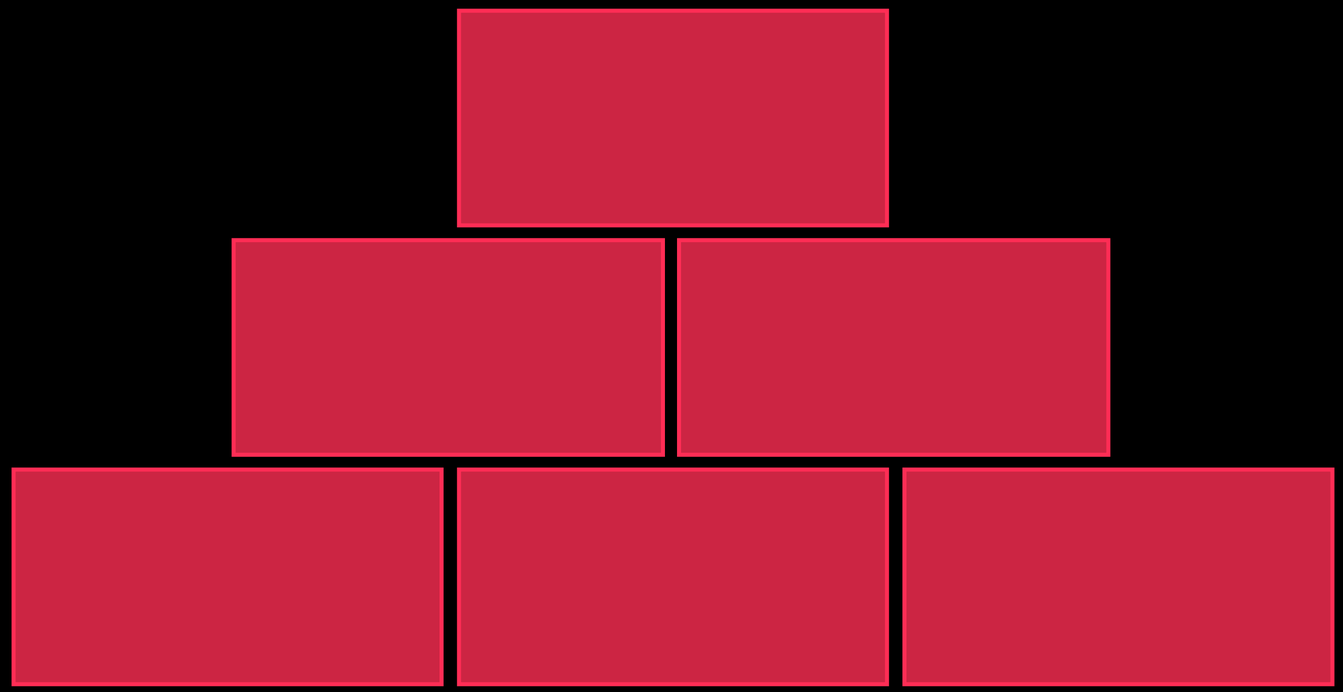
UIMotionEffect

Absolute offsets

Inputs	Outputs
Device pose	Value for each key path

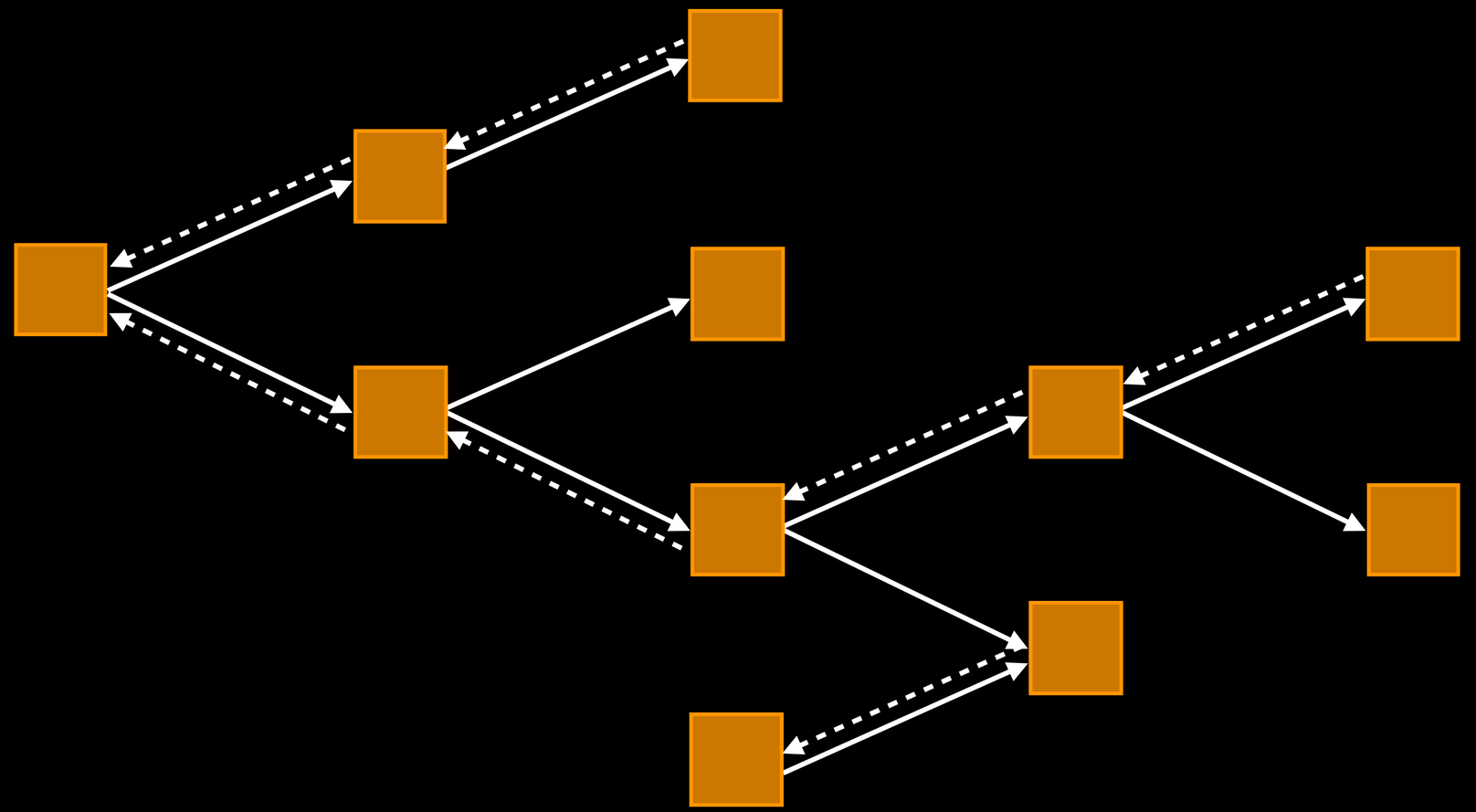
A particular view

③ Simplify with Immutability

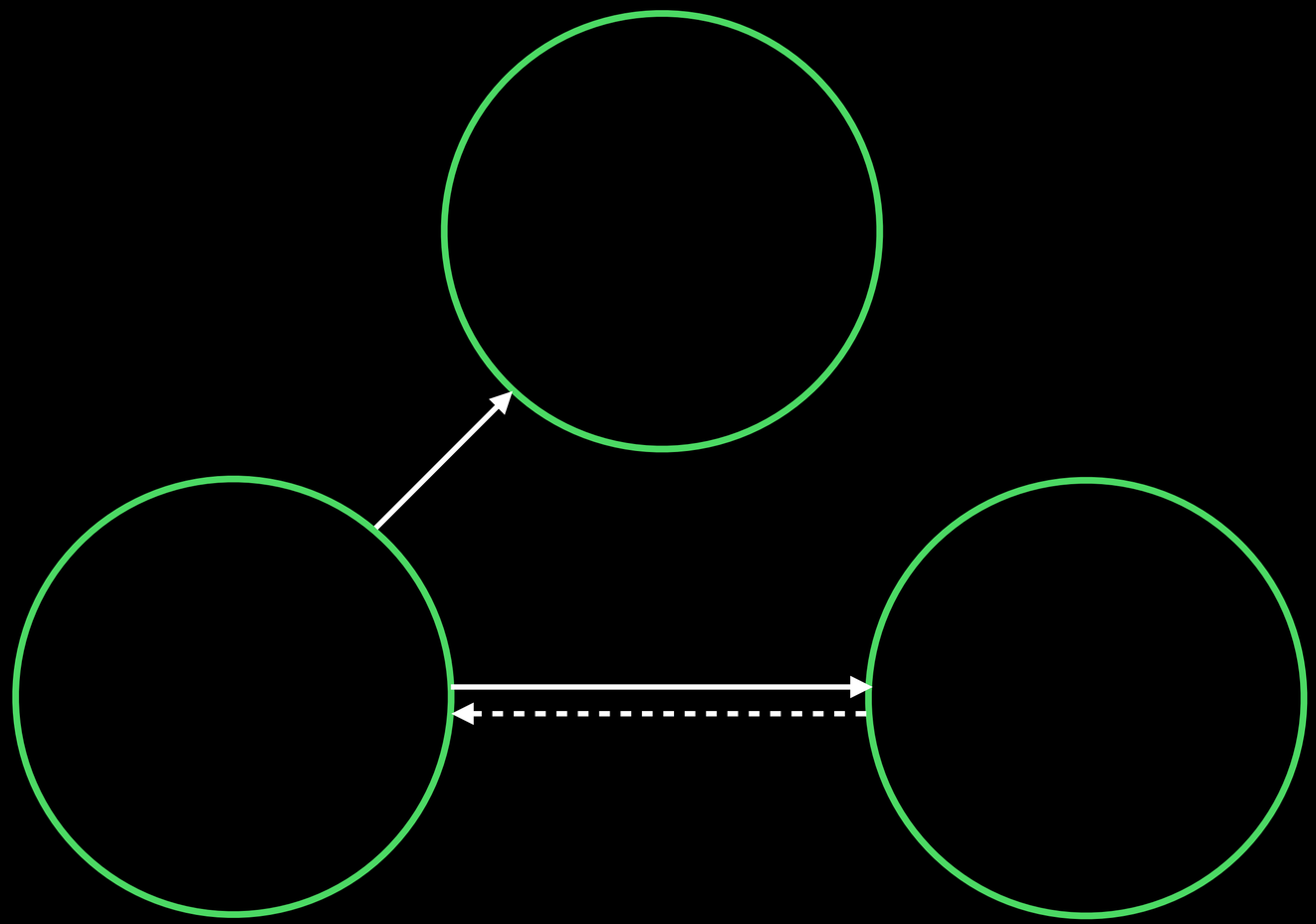


Review

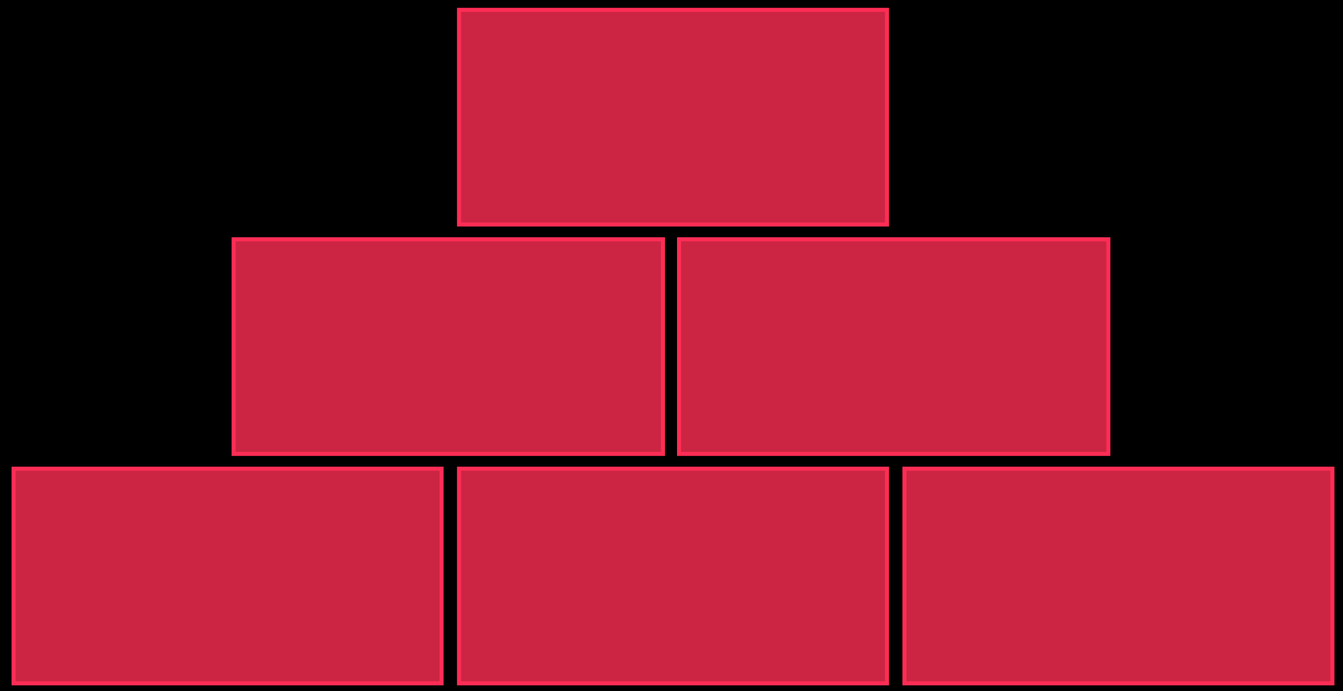
① Design Information Flow



② Define Clear Responsibilities



③ Simplify with Immutability



What now?

More Information

Jake Behrens
App Frameworks Evangelist
behrens@apple.com

Apple Developer Forums
http://devforums.apple.com

Related Sessions

• Core iOS Application Architectural Patterns	Mission	Thursday 9:00AM

WWDC14