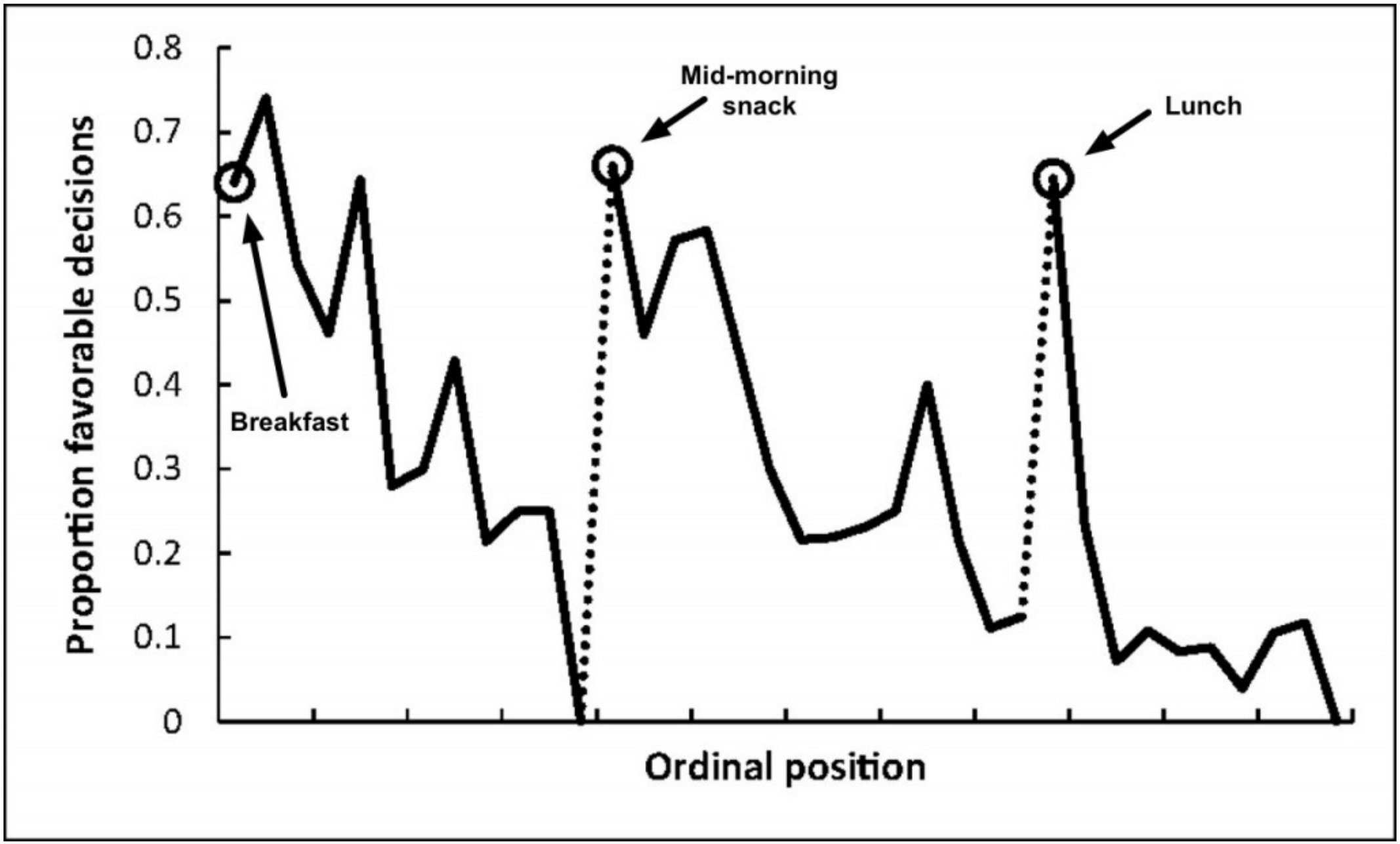
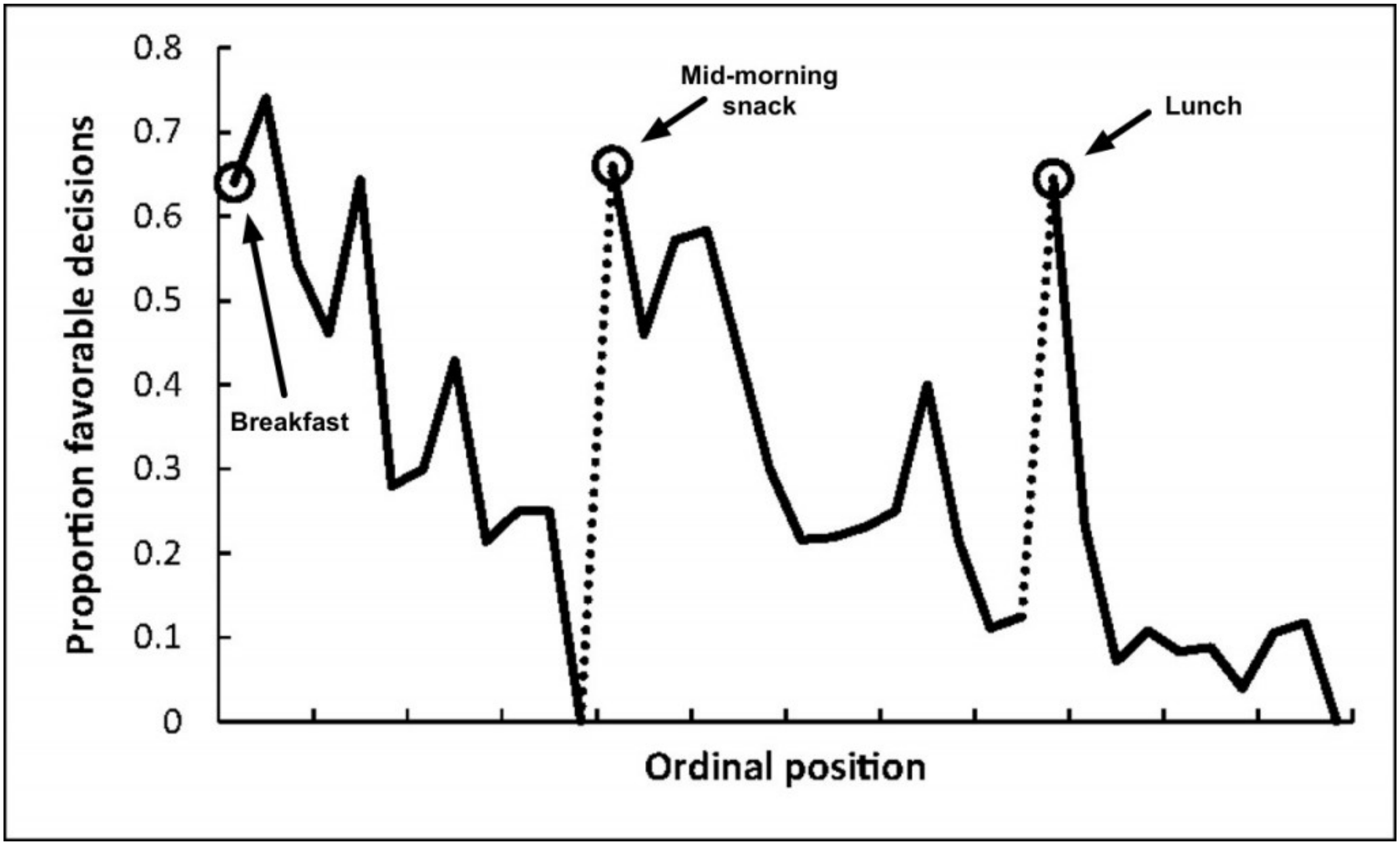
Testing Ember.js Apps

Managing Dependency



A guaranteed Successful Talk



A guaranteed Successful Talk ✓

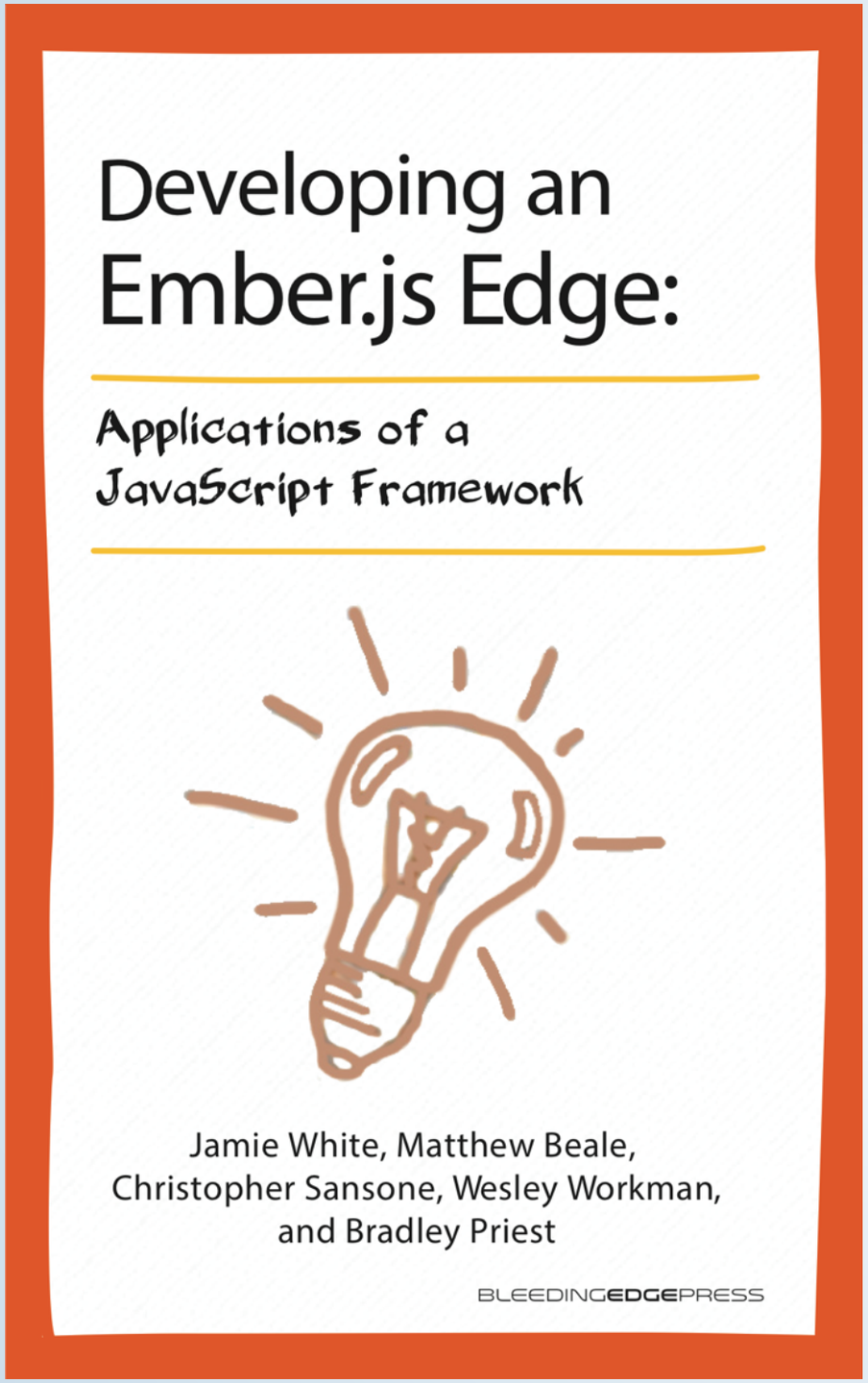
@MIXONIC

HTTP://MADHATTED.COM

MATT.BEALE@MADHATTED.COM

201 Created

We build 🛸-age apps with Ember.js. We take teams from 🚲 to 🚀 in no time flat.



HTTP://BIT.LY/EMBERFEST-EDGE

A conference planning app

- Submitting a talk
- Purchasing a ticket
- Preparing a talk

A conference planning app

The domain

- Submitting a talk
- Purchasing a ticket
- Preparing a talk

A conference planning app

- Submitting a talk
- Purchasing a ticket
- Preparing a talk

A conference planning app

- Adapting to a server
- Managing asynchronous APIs
- Browser APIs

- Submitting a talk
- Purchasing a ticket
- Preparing a talk

A conference planning app

- Adapting to a server
- Managing asynchronous APIs
- Browser APIs

The domain →

- Submitting a talk
- Purchasing a ticket
- Preparing a talk

A conference planning app

What is this?!

- Adapting to a server
- Managing asynchronous APIs
- Browser APIs

- Submitting a talk
- Purchasing a ticket
- Preparing a talk

Dependencies

Dependencies

Domain


Own it
Dependencies
Domain

Own it

Dependencies

Domain

Ember-CLI Review

```
1 // app/utils/some-object.js
2 export default SomeObject;
3 export default Ember.Route.extend({
4 });
5
6 import MyObject from "app/utils/some-object";
7
8 // app/utils/many-things.js
9 export {gadget, anotherGadget};
10
11 import {anotherGadget} from "app/utils/many-things";
12
13 // tests/routes/proposal-test.js
14 moduleFor("route:routeName", "Test Module Name", {
15   // options include: needs, subject, setup, teardown
16 });
```

Constraining Dependencies

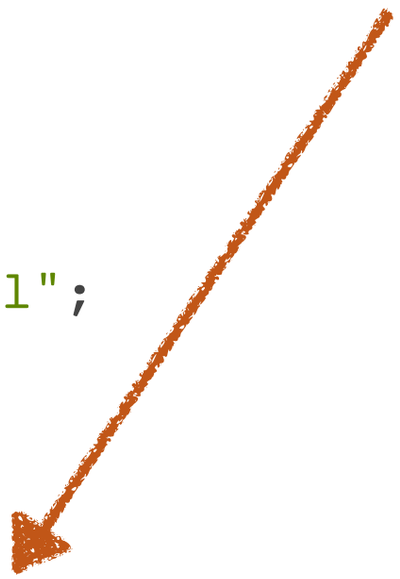
1. Isolate implementation
2. Normalize behavior

Constraining a dependency

```
1 // app/route/proposal.js
2 import Proposal from "../model/proposal";
3
4 export default Ember.Route.extend({
5
6   model: function(){
7     var recovered = window.localStorage.getItem('proposalSnapshot');
8     if (recovered) {
9       return Proposal.load(recovered);
10    } else {
11      return Proposal.build();
12    }
13  }
14
15 });
```

Constraining a dependency

Implementation



```
1 // app/route/proposal.js
2 import Proposal from "../model/proposal";
3
4 export default Ember.Route.extend({
5
6   model: function(){
7     var recovered = window.localStorage.getItem('proposalSnapshot');
8     if (recovered) {
9       return Proposal.load(recovered);
10    } else {
11      return Proposal.build();
12    }
13  }
14
15 });
```

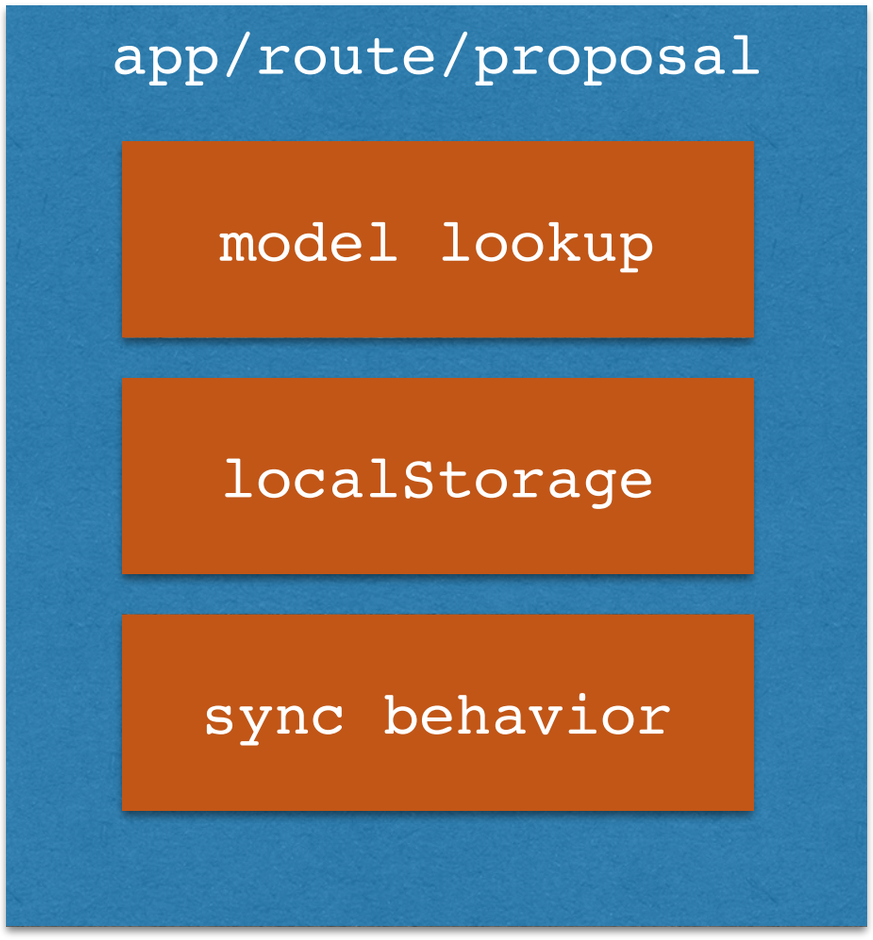
Constraining a dependency

Implementation

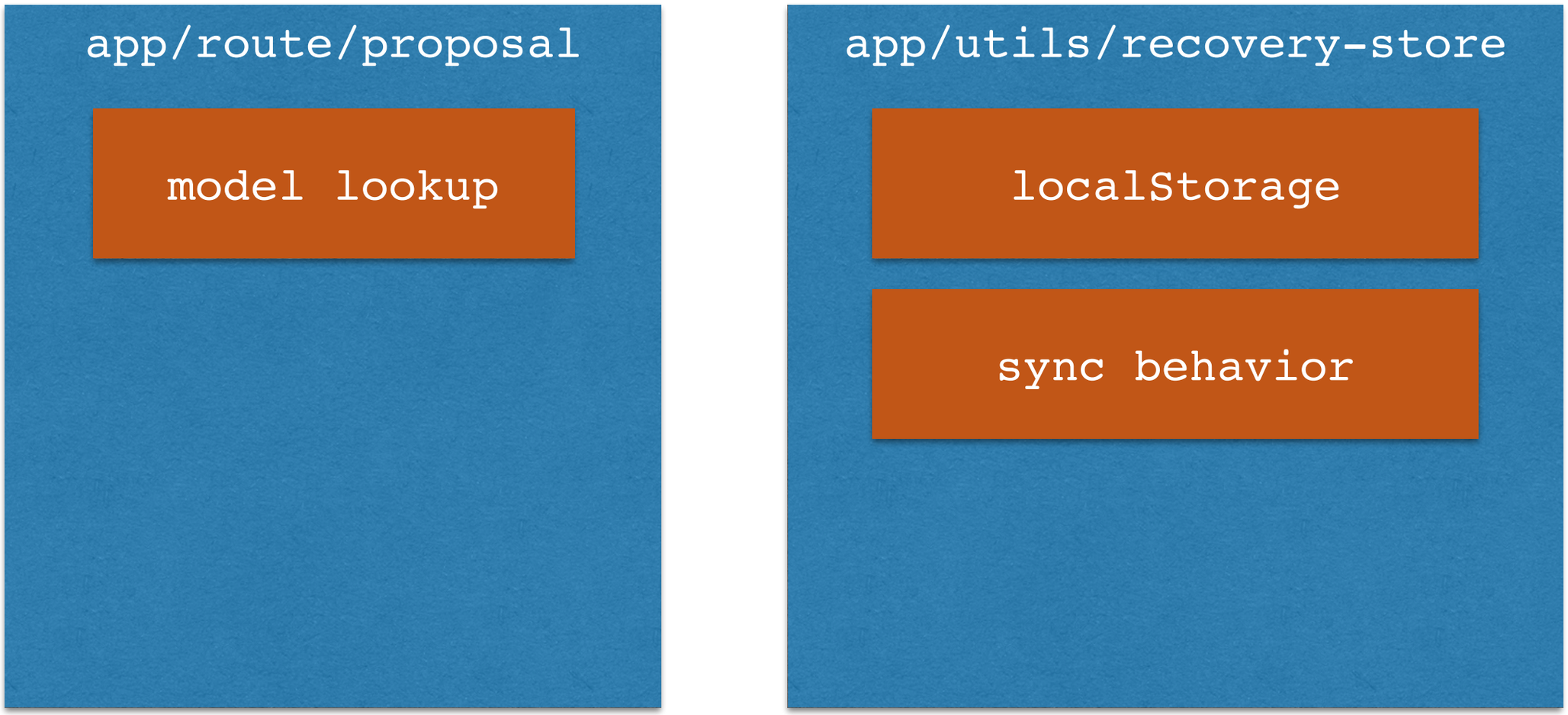
```
1  // app/route/proposal.js
2  import Proposal from "../model/proposal";
3
4  export default Ember.Route.extend({
5
6    model: function(){
7      var recovered = window.localStorage.getItem('proposalSnapshot');
8      if (recovered) {
9        return Proposal.load(recovered);
10     } else {
11       return Proposal.build();
12     }
13   }
14
15 });
```

Synchronous behavior

Constraining a dependency



Constraining a dependency



Constraining a dependency

```
1 // app/utils/recovery-store.js
2
3 export default Ember.Object.extend({
4   recover: function(key){
5     return new Ember.RSVP.Promise(function(resolve, reject){
6       resolve(window.localStorage.getItem(key));
7     });
8   }
9 });
```

Constraining a dependency

Isolated in recoveryStore

```
1 // app/utils/recovery-store.js
2
3 export default Ember.Object.extend({
4   recover: function(key){
5     return new Ember.RSVP.Promise(function(resolve, reject){
6       resolve(window.localStorage.getItem(key));
7     });
8   }
9 });
```

Constraining a dependency

Isolated in recoveryStore

```
1 // app/utils/recovery-store.js
2
3 export default Ember.Object.extend({
4   recover: function(key){
5     return new Ember.RSVP.Promise(function(resolve, reject){
6       resolve(window.localStorage.getItem(key));
7     });
8   }
9 });
```

Normalize sync and async

Using a synchronous Promise

```
1 import RecoveryStore from "app/utils/recovery-store";
2
3 var recoveryStore = new RecoveryStore();
4 recoveryStore.recover('proposalSnapshot').then(function(data){
5   console.log('recovering ', data);
6 });
```

Using a synchronous Promise

```
1 import RecoveryStore from "app/utils/recovery-store";
2
3 var recoveryStore = new RecoveryStore();
4 recoveryStore.recover('proposalSnapshot').then(function(data){
5   console.log('recovering ', data);
6 });
```

Using an asynchronous Promise

```
1 import RecoveryStore from "app/utils/recovery-store";
2
3 var recoveryStore = new RecoveryStore();
4 recoveryStore.recover('proposalSnapshot').then(function(data){
5   console.log('recovering ', data);
6 });
```

Constraining a dependency

```
1 // app/utils/recovery-store.js
2
3 export default Ember.Object.extend({
4   recover: function(key){
5     return new Ember.RSVP.Promise(function(resolve, reject){
6       resolve(window.localStorage.getItem(key));
7     });
8   }
9 });
```

localStorage

Constraining a dependency

```
1 // app/utils/recovery-store.js
2
3 export default Ember.Object.extend({
4   recover: function(key){
5     return new Ember.RSVP.Promise(function(resolve, reject){
6       Ember.$.ajax("GET", "/recovery/"+key, {dataType: 'json'}, {
7         success: Ember.run.bind(this, function(data){
8           resolve(data);
9         })
10      });
11    });
12  }
13 });
```

AJAX!

Using a synchronous Promise

```
1 import RecoveryStore from "app/utils/recovery-store";
2
3 var recoveryStore = new RecoveryStore();
4 recoveryStore.recover('proposalSnapshot').then(function(data){
5   console.log('recovering ', data);
6 });
```

Using an asynchronous Promise

```
1 import RecoveryStore from "app/utils/recovery-store";
2
3 var recoveryStore = new RecoveryStore();
4 recoveryStore.recover('proposalSnapshot').then(function(data){
5   console.log('recovering ', data);
6 });
```

Unit Testing Dependencies

Mocking a dependency in units

```
1  // app/route/proposal.js
2  import RecoveryStore from "../utils/recovery-store";
3  import Proposal from "../model/proposal";
4
5  export default Ember.Route.extend({
6
7    recoveryStore: function(){
8      return new RecoveryStore();
9    }.property(),
10
11   model: function(){
12     return this.get('recoveryStore').recover('proposalSnapshot').
13       then(function(data){
14         if (data) {
15           return Proposal.load(data);
16         } else {
17           return Proposal.build();
18         }
19       });
20   }
21
22 });
```

Mocking a dependency in units

```
1  import { test, moduleFor } from 'ember-qunit';
2
3  moduleFor('route:proposal', 'ProposalRoute');
4
5  test('recovers a proposal', function() {
6    expect(1);
7    window.localStorage.setItem('proposalSnapshot', {body: 'pitch Angular'});
8    var route = this.subject();
9    return route.model().then(function(proposal){
10     ok(proposal.get('body'), 'pitch Angular');
11   });
12 });
```

Mocking a dependency in units

Implementation Leak

```
1  import { test, moduleFor } from 'ember-qunit';
2
3  moduleFor('route:proposal', 'ProposalRoute');
4
5  test('recovers a proposal', function() {
6    expect(1);
7    window.localStorage.setItem('proposalSnapshot', {body: 'pitch Angular'});
8    var route = this.subject();
9    return route.model().then(function(proposal){
10     ok(proposal.get('body'), 'pitch Angular');
11   });
12 });
```

(ʃ ŏ Дŏ)ʃ⌨

Mocking a dependency in units

```
1 import { test, moduleFor } from 'ember-qunit';
2
3 var mockRecoveryStore = { recover: function(){
4   return new Ember.RSVP.Promise(function(resolve){
5     resolve(Ember.Object.create({body: 'pitch Angular'}));
6   });
7 } };
8
9 moduleFor('route:proposal', 'ProposalRoute');
10
11 test('recovers a proposal', function() {
12   expect(1);
13   var route = this.subject({recoveryStore: mockRecoveryStore});
14   return route.model().then(function(proposal){
15     ok(proposal.get('body'), 'pitch Angular');
16   });
17 });
```

Mocking a dependency in units

```
1 import { test, moduleFor } from 'ember-qunit';
2
3 var mockRecoveryStore = { recover: function(){
4   return new Ember.RSVP.Promise(function(resolve){
5     resolve(Ember.Object.create({body: 'pitch Angular'}));
6   });
7 } };
8
9 moduleFor('route:proposal', 'ProposalRoute', {
10   subject: function(options, klass, container){
11     return klass.create(Ember.merge({
12       recoveryStore: mockRecoveryStore
13     }, options));
14   }
15 });
16
17 test('recovers a proposal', function() {
18   expect(1);
19   var route = this.subject();
20   return route.model().then(function(proposal){
21     ok(proposal.get('body'), 'pitch Angular');
22   });
23 });
```

Mocking a dependency in units

```
1 import { test, moduleFor } from 'ember-qunit';
2
3 var mockRecoveryStore = { recover: function(){
4   return new Ember.RSVP.Promise(function(resolve){
5     resolve(Ember.Object.create({body: 'pitch Angular'}));
6   });
7 } };
8
9 moduleFor('route:proposal', 'ProposalRoute', {
10   subject: function(options, klass, container){
11     return klass.create(Ember.merge({
12       recoveryStore: mockRecoveryStore
13     }, options));
14   }
15 });
16
17 test('recovers a proposal', function() {
18   expect(1);
19   var route = this.subject();
20   return route.model().then(function(proposal){
21     ok(proposal.get('body'), 'pitch Angular');
22   });
23 });
```

Why is this code good?

Why is this code good? *Well Gary Bernhardt sez…*

Three goals of testing

Why is this code good? *Well Gary Bernhardt sez…*

#1: Prevent regressions

#2: Prevent fear

#3: Prevent bad design

Why is this code good? *Well Gary Bernhardt sez…*

#1: Prevent regressions ✓

#2: Prevent fear

#3: Prevent bad design

Why is this code good? *Well Gary Bernhardt sez…*

#1: Prevent regressions ✓

#2: Prevent fear ✓

#3: Prevent bad design

Why is this code good? *Well Gary Bernhardt sez…*

#1: Prevent regressions ✓

#2: Prevent fear ✓✓ Is fast!

#3: Prevent bad design

Why is this code good? *Well Gary Bernhardt sez…*

#1: Prevent regressions ✓

#2: Prevent fear ✓✓ Is fast!

#3: Prevent bad design ✓

ᕕ(ᐛ)ᕗ

Acceptance Testing Dependencies

Mocking a dependency in acceptance

```
1  // app/route/proposal.js
2  import RecoveryStore from "../utils/recovery-store";
3  import Proposal from "../model/proposal";
4
5  export default Ember.Route.extend({
6
7    recoveryStore: function(){
8      return new RecoveryStore();
9    }.property(),
10
11   model: function(){
12     return this.get('recoveryStore').recovery('proposalSnapshot').
13       then(function(data){
14         if (data) {
15           return Proposal.load(data);
16         } else {
17           return Proposal.build();
18         }
19       });
20   }
21
22 });
```

Mocking a dependency in acceptance

```
1  // tests/acceptance/proposal-test.js
2
3  import Ember from 'ember';
4  import startApp from '../helpers/start-app';
5
6  var App;
7
8  module('Acceptance: Proposal', {
9    setup: function() {
10     App = startApp();
11   },
12   teardown: function() {
13     Ember.run(App, 'destroy');
14   }
15 });
16
17 test('visiting /proposal', function() {
18   visit('/proposal');
19
20   andThen(function() {
21     equal(currentPath(), 'proposal');
22   });
23 });
```

Mocking a dependency in acceptance

```
1  // tests/acceptance/proposal-test.js
2
3  import Ember from 'ember';
4  import startApp from '../helpers/start-app';
5
6  var App;
7
8  module('Acceptance: Proposal', {
9    setup: function() {
10     App = startApp();
11   },
12   teardown: function() {
13     Ember.run(App, 'destroy');
14   }
15 });
16
17 test('visiting /proposal', function() {
18   visit('/proposal');
19
20   andThen(function() {
21     equal(currentPath(), 'proposal');
22   });
23 });
```

Where can we mock recoveryStore?

Mocking a dependency in acceptance

```
1  // app/route/proposal.js
2  import RecoveryStore from "../utils/recovery-store";
3  import Proposal from "../model/proposal";
4
5  export default Ember.Route.extend({
6
7    recoveryStore: function(){
8      return new RecoveryStore();
9    }.property(),
10
11   model: function(){
12     return this.get('recoveryStore').recovery('proposalSnapshot').
13       then(function(data){
14         return Proposal.load(data);
15       }, function(){
16         return Proposal.build();
17       });
18   }
19
20 });
```

"Dependency Lookup"

Dependency Lookup

```
1 var Gadget = Ember.Object.create({
2   shelf: function(){
3     return Shelf.create();
4   }
5 });
6
7 Gadget.create();
```

Dependency Lookup

```
1 var Gadget = Ember.Object.create({
2   shelf: function(){
3     return Shelf.create();
4   }
5 });
6
7 Gadget.create();
```

Gadget is in charge

Dependency Lookup

```
1 var Gadget = Ember.Object.create({
2   shelf: function(){
3     return Shelf.create();
4   }
5 });
6
7 Gadget.create();
```

Dependency Injection

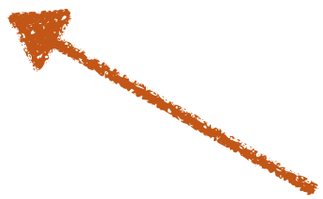
```
1 var Gadget = Ember.Object.create({
2 });
3
4 function buildGadget(){
5   return Gadget.create({
6     shelf: Shelf.create()
7   });
8 }
```

Dependency Lookup

```
1 var Gadget = Ember.Object.create({
2   shelf: function(){
3     return Shelf.create();
4   }
5 });
6
7 Gadget.create();
```

Dependency Injection

```
1 var Gadget = Ember.Object.create({
2 });
3
4 function buildGadget(){
5   return Gadget.create({
6     shelf: Shelf.create()
7   });
8 }
```



Gadget's dependency is injected

Dependency Lookup

```
1 var Gadget = Ember.Object.create({
2   shelf: function(){
3     return Shelf.create();
4   }
5 });
6
7 Gadget.create();
```

Dependency Injection / Inversion of Control

```
1 var Gadget = Ember.Object.create({
2 });
3
4 function buildGadget(){
5   return Gadget.create({
6     shelf: Shelf.create()
7   });
8 }
```

Converting “Dependency Lookup” to “Dependency Injection”

1. Create an initializer
2. Register the dependency
3. Inject the dependency
4. Drop the lookup

Converting “Dependency Lookup” to “Dependency Injection”

Create an initializer

```
1  // app/initializers/recovery-store.js
2
3  import RecoveryStore from "../utils/recovery-store";
4
5  export default {
6    name: 'recovery-store',
7    initialize: function(container, application) {
8      application.register('services:recoveryStore', RecoveryStore);
9      application.inject('route', 'recoveryStore', 'services:recoveryStore');
10   }
11 };
```

Converting “Dependency Lookup” to “Dependency Injection”

```
1 // app/initializers/recovery-store.js
2
3 import RecoveryStore from "../utils/recovery-store";
4
5 export default {
6   name: 'recovery-store',
7   initialize: function(container, application) {
8     application.register('services:recoveryStore', RecoveryStore);
9     application.inject('route', 'recoveryStore', 'services:recoveryStore');
10  }
11 };
```

Register the dependency

Converting “Dependency Lookup” to “Dependency Injection”

```
1 // app/initializers/recovery-store.js
2
3 import RecoveryStore from "../utils/recovery-store";
4
5 export default {
6   name: 'recovery-store',
7   initialize: function(container, application) {
8     application.register('services:recoveryStore', RecoveryStore);
9     application.inject('route', 'recoveryStore', 'services:recoveryStore');
10  }
11 };
```

Inject the dependency

Converting “Dependency Lookup” to “Dependency Injection”

```
1  // app/route/proposal.js
2  import RecoveryStore from "../utils/recovery-store";
3  import Proposal from "../model/proposal";
4
5  export default Ember.Route.extend({
6
7    model: function(){
8      return this.recoveryStore.recover('proposalSnapshot').
9        then(function(data){
10         return Proposal.load(data);
11       }, function(){
12         return Proposal.build();
13       });
14   }
15
16 });
```

Drop the lookup

Mocking a dependency in acceptance

```
1  // tests/acceptance/proposal-test.js
2
3  import Ember from 'ember';
4  import startApp from '../helpers/start-app';
5  import mockRecoveryStore from '../helpers/mock-recovery-store';
6
7  var App;
8
9  module('Acceptance: Proposal', {
10   setup: function() {
11     App = startApp();
12     // unregister is not yet a first-class API :-/
13     App.__container__.unregister('services:recoveryStore');
14     App.register('services:recoveryStore', mockRecoveryStore, {
15       instantiate: false });
16   },
17   teardown: function() {
18     Ember.run(App, 'destroy');
19   }
20 });
21
22 test('visiting /proposal', function() {
23   visit('/proposal');
24
25   andThen(function() {
26     equal(currentPath(), 'proposal');
27   });
28 });
```

Unregister the normal recoveryStore

Mocking a dependency in acceptance

```
1  // tests/acceptance/proposal-test.js
2
3  import Ember from 'ember';
4  import startApp from '../helpers/start-app';
5  import mockRecoveryStore from '../helpers/mock-recovery-store';
6
7  var App;
8
9  module('Acceptance: Proposal', {
10   setup: function() {
11     App = startApp();
12     // unregister is not yet a first-class API :-/
13     App.__container__.unregister('services:recoveryStore');
14     App.register('services:recoveryStore', mockRecoveryStore, {
15       instantiate: false });
16   },
17   teardown: function() {
18     Ember.run(App, 'destroy');
19   }
20 });
21
22 test('visiting /proposal', function() {
23   visit('/proposal');
24
25   andThen(function() {
26     equal(currentPath(), 'proposal');
27   });
28 });
```

Register the mockRecoveryStore

Why is this code good?

#1: Prevent regressions ✓

#2: Prevent fear ✓✓ Is fast!

#3: Prevent bad design ✓

STOP IT WITH

(ノಠ益ಠ)ノ┻━┻

ALL THE CODE

Starting points:

ember-cli.com

emberjs.com/guides/testing

github.com/rwjblue/ember-qunit

emberjs.com/api/classes/Ember.Test.html

Starting points:

wikipedia.org/wiki/Inversion_of_control

www.martinfowler.com/articles/injection.html

Own your dependencies

Contain their APIs

Own your dependencies

Design better code with better tests


Gracias