A introduction to custom events

AsyncJS • 25th August 2011 • Aron Carroll

- Simply a way of informing objects of changes
- One object broadcasts events (or topics)
- Others register interest (or subscribe) by providing a callback for a event to the broadcaster
- Callbacks are called each time a event is broadcast

Navigation

```
var jeff = {
  read: function (issue) {
    alert("Wow, what interesting content");
  }
};

newspaper.subscribe("issue", jeff.read);

newspaper.publish("issue", new Issue());

// ALERT: "Wow, what interesting content"
```

- Common idiom on the web
- We’re all used to **click**, **submit**, **load** events
- Great for decoupling code into smaller modules

- Publish / Subscribe
- Observer
- Custom Events

Examples in the wild

DOM Events

```
redButton.onclick = function () {
  launchTheRocketShip();
};
```

Examples / DOM Events

```
redButton.addEventListener(
  "click",
  function () {
    launchTheRocketShip();
  },
  false
);
```

jQuery.ajax()

```
function cookRecipe(recipe) {
  // pots, pans, timings etc...
}

function getMagicRecipe(cb) {
  return jQuery.get("magic-recipe.txt", cb);
}

getMagicRecipe(cookRecipe);
```

```
// As of jQuery 1.5
var request = getMagicRecipe(cookRecipe);
request.error(function () {
  orderFishWithChips();
});
```

Observing Models

```
var thermos = new Thermos({
  beverage:    "tea",
  fullness:    0.8,
  temperature: 50
});

// View watches for changes to properties
var view = new ThermosStatusView({
  model: thermos
});

// View automatically updates
thermos.drink(0.8);
```

```
var Thermos = Model.extend({
  drink: function (drunk) {
    var oldAmount = this.get("fullness"),
        newAmount = oldAmount - drunk;

    // .set() publishes a “change” event
    return this.set({fullness: newAmount});
  }
});
```

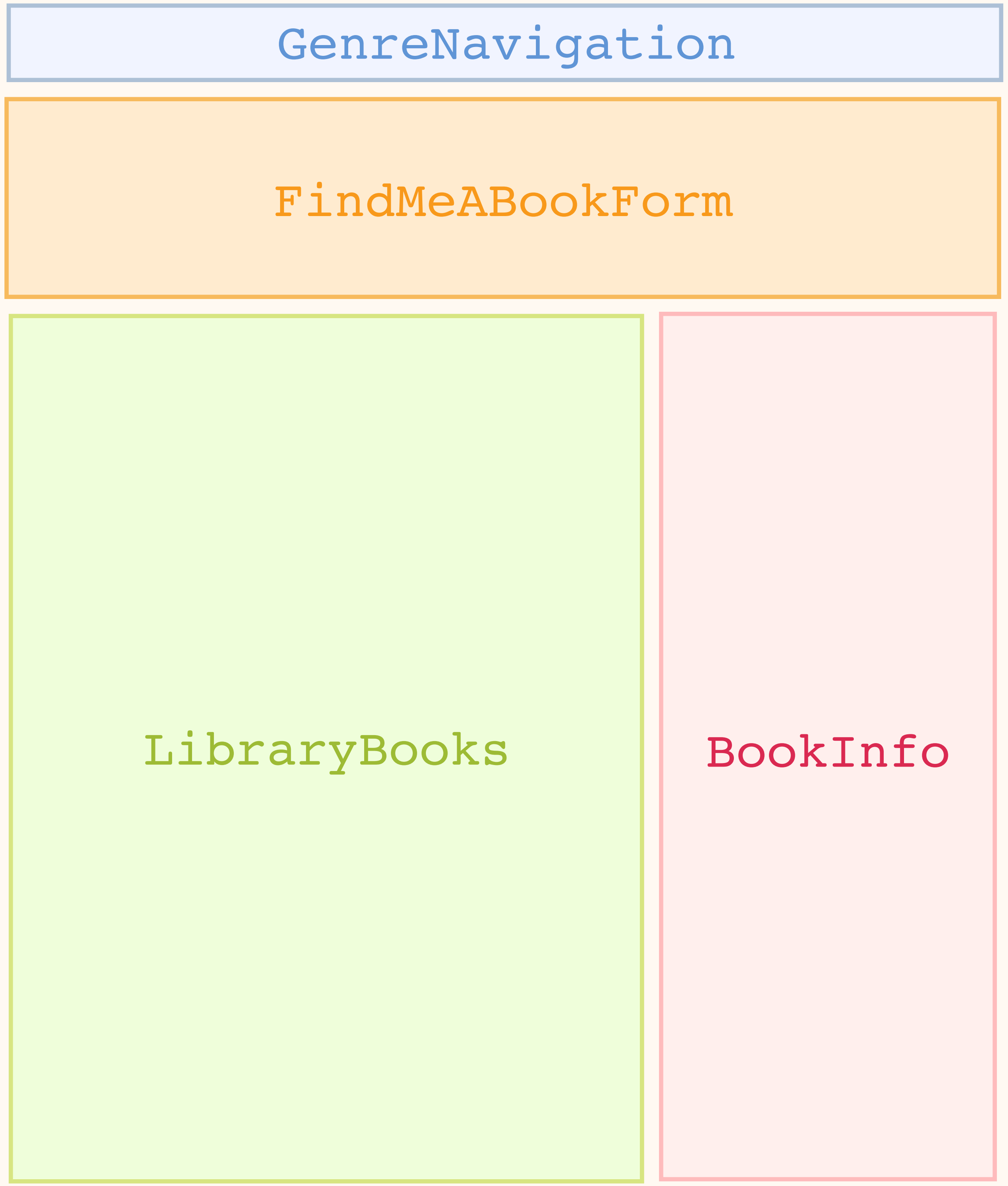
Examples / Backbone

```
var ThermosStatusView = View.extend({
  intialize: function () {
    // View watches for “change”
    this.model.bind(
      "change:fullness", this.onFullnessChange
    );
  },
  onFullnessChange: function (amount) {
    if (amount > 0) {
      this.updateThermosGauge(amount);
    } else {
      this.displaySeriousWarningAlert();
    }
  }
});
```

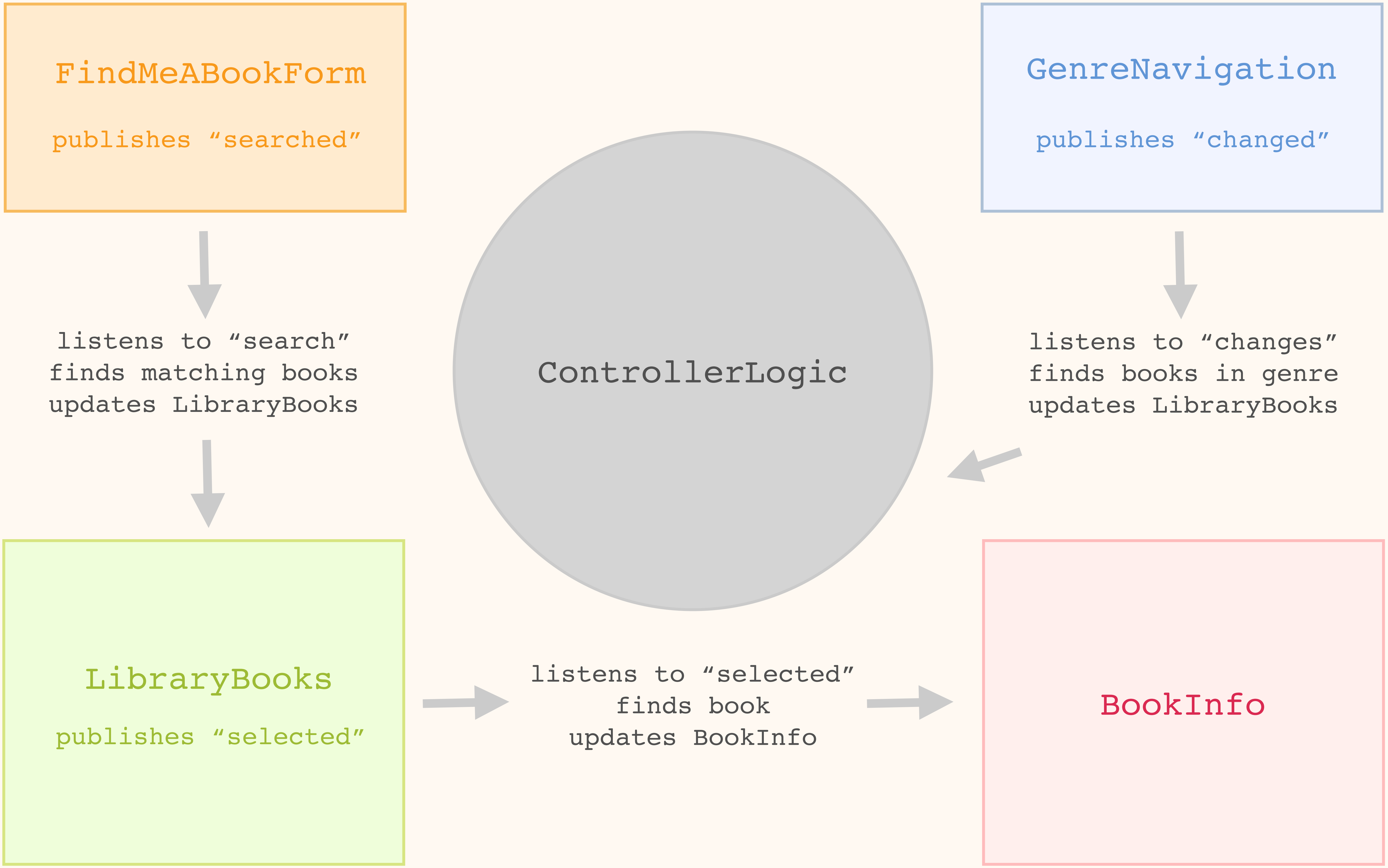
So why is this useful?

Breaking UI into components

Uses / Loose Coupling



Uses / Loose Coupling



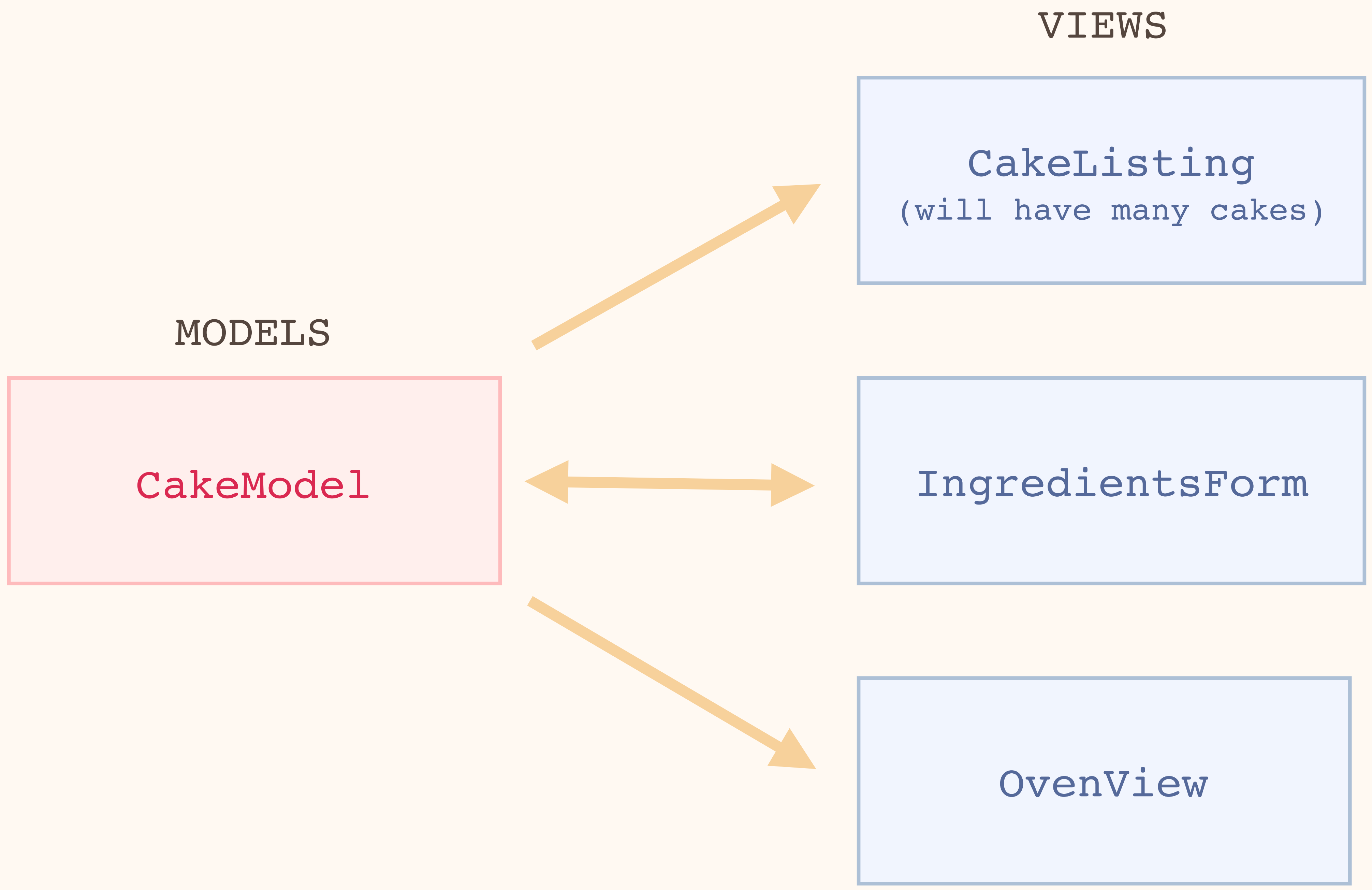
```
$stall = $("#circusStall").coconutShy();

$stall.bind("hit", function (coconut) {
  if (!coconut.isGluedToStand()) {
    alert("Prizes!");
  }
});
```

Notification of changing data & state

Uses / Notification of changes

- UI Elements: Sliders, Pop-ups & Select boxes
- Counters & Flags: Message counts, Notifications
- Model data: Change of username

Uses / Notification of changes



Extensibility

Extend an objects functionality by using events to perform a different action or enhance an existing one.

This is great for plugins & frameworks.

```
// Update the position of a custom marker
// when the user pans the map.
google.maps.addListener(
  map, "drag", updateCustomUIPosition
);
```

Disadvantages

Disadvantages

- Harder to debug
- Loose coupling

Quick Example

- Ben Allman’s TinyPubSub- Great if you use jQuery. But slow as tied to DOM.
- PubSub js - Simple and fast.
- MicroEvents - Alternative to PubSub
- EventEmitter in Node.js
- Most JavaScript libraries have own implementation

Summary

- Break your JavaScript into components
- Fire events when interesting things happen
- (I) prefer observable objects over global pub/sub

Useful Links

- http://msdn.microsoft.com/en-us/scriptjunkie/hh201955
- http://www.addyosmani.com/resources/essentialjsdesignpatterns/book/#observerpatternjavascript
- http://blog.rebeccamurphey.com/pubsub-screencast
- http://yehudakatz.com/2009/04/20/evented-programming-with-jquery/

Useful Links

- TinyPubSub – https://gist.github.com/661855
- PubSubJS – https://github.com/phiggins42/bloody-jquery-plugins/blob/master/pubsub.js
- MicroEvents – http://notes.jetienne.com/2011/03/22/microeventjs.html
- EventEmitter – http://nodejs.org/docs/v0.4.9/api/events.html