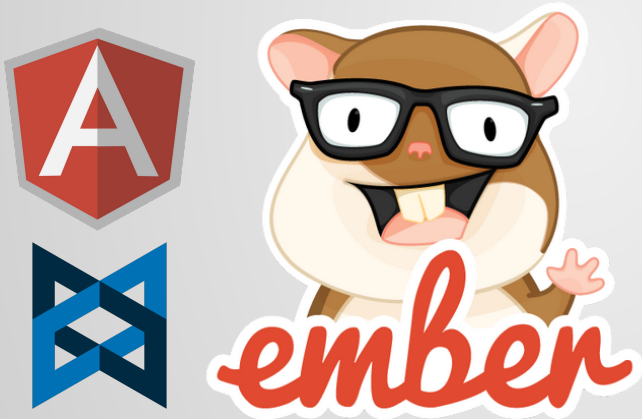
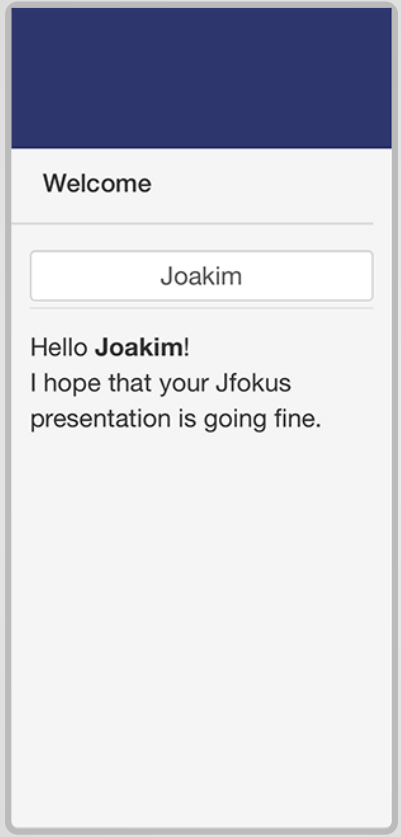
Angular vs Ember vs Backbone

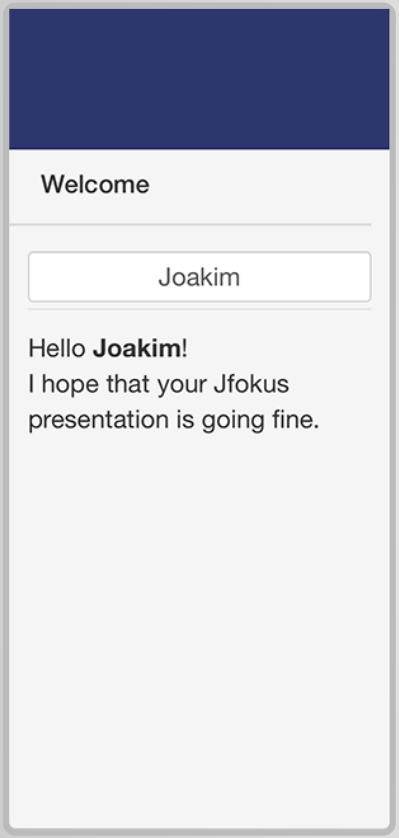
(and Knockout) in real applications



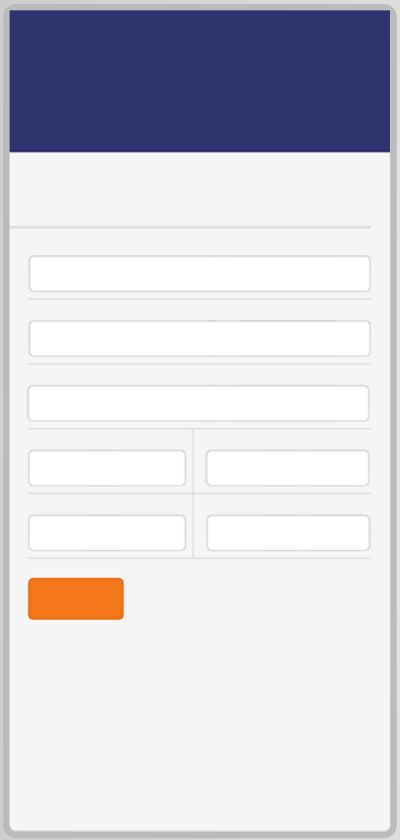
Joakim Kemeny
@joakimkemeny

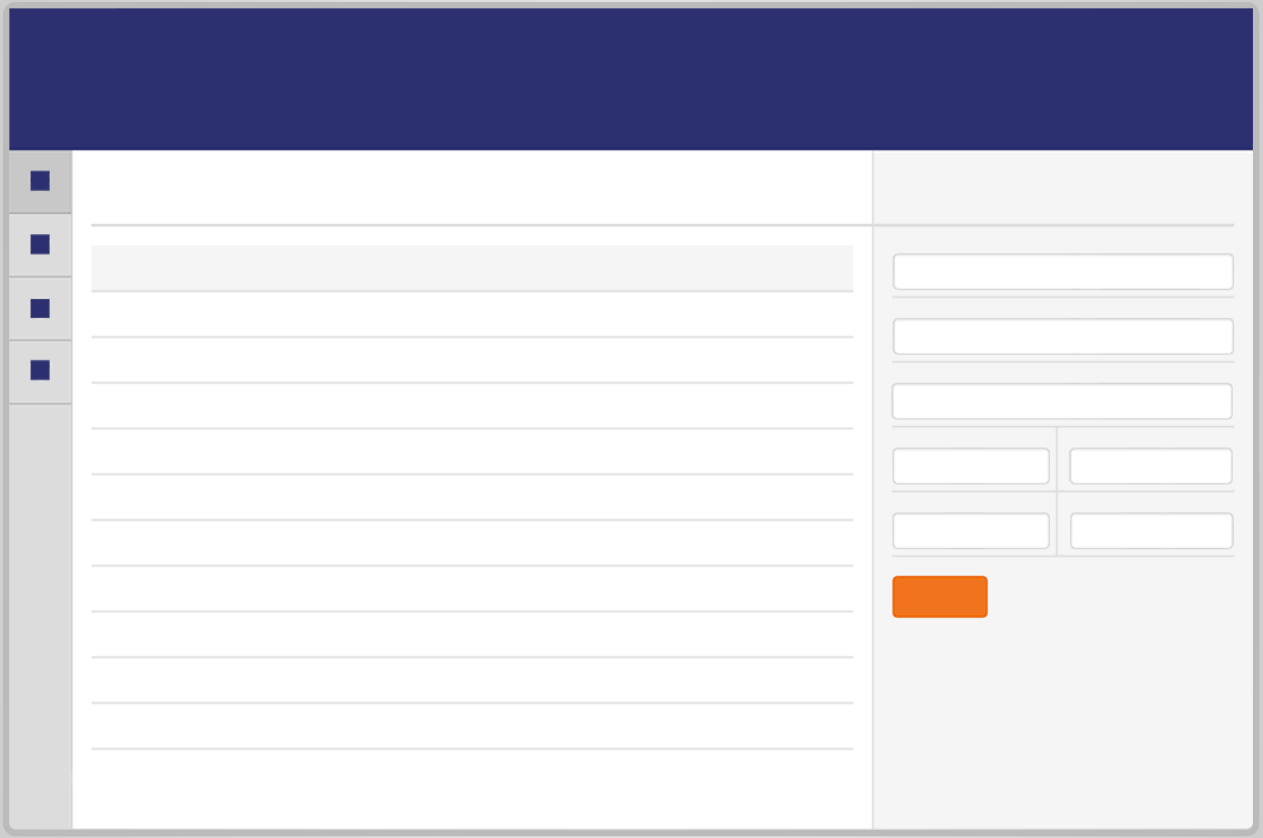
CALLISTA
Enterprise

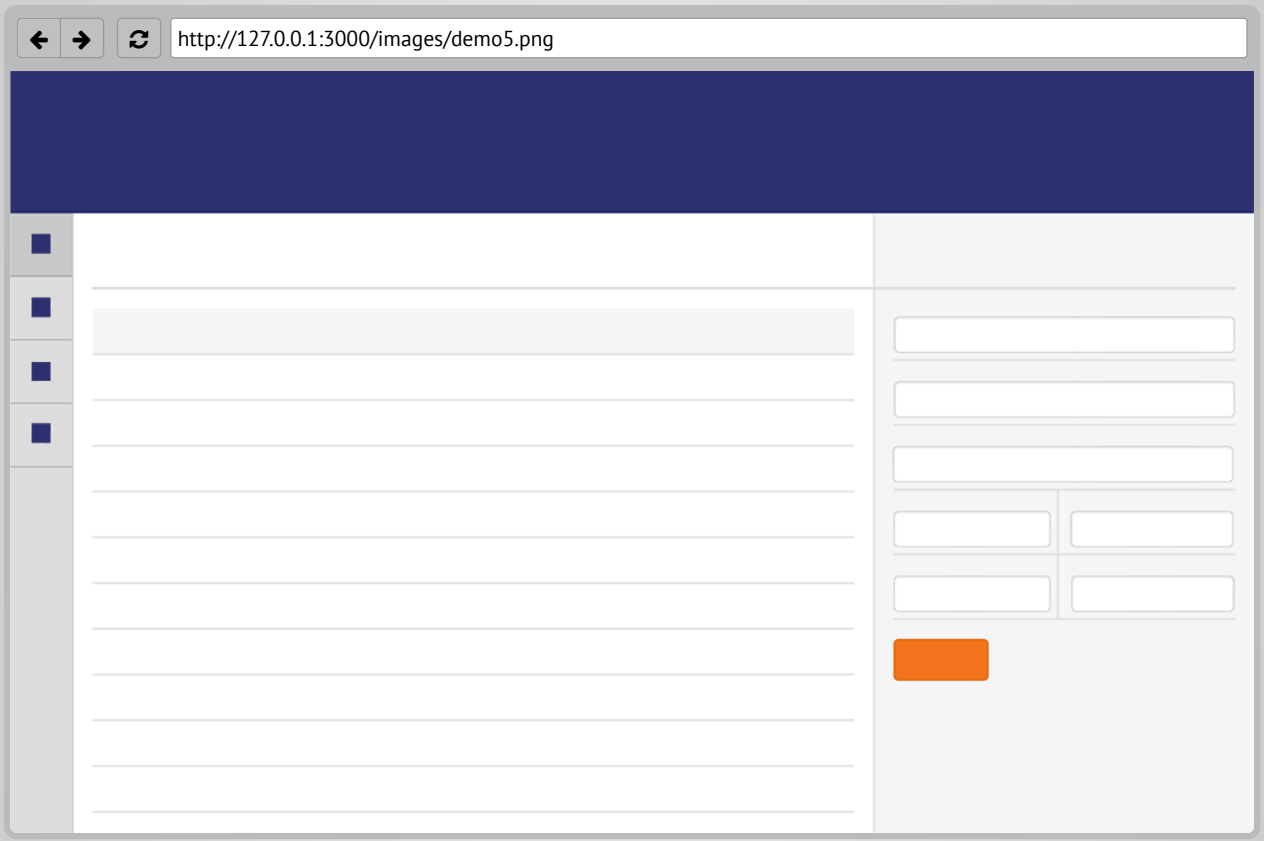

Welcome
Joakim
Hello **Joakim**!
I hope that your Jfokus
presentation is going fine.

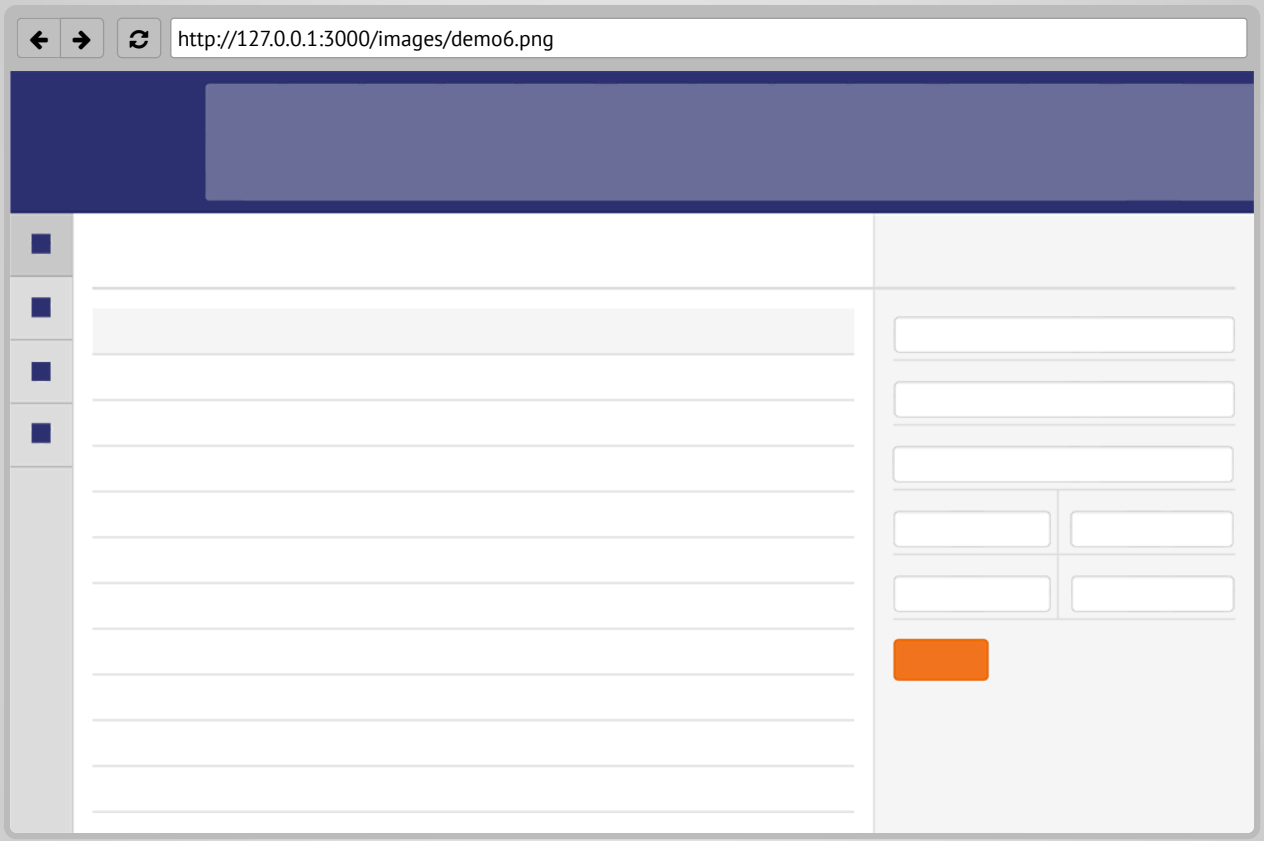


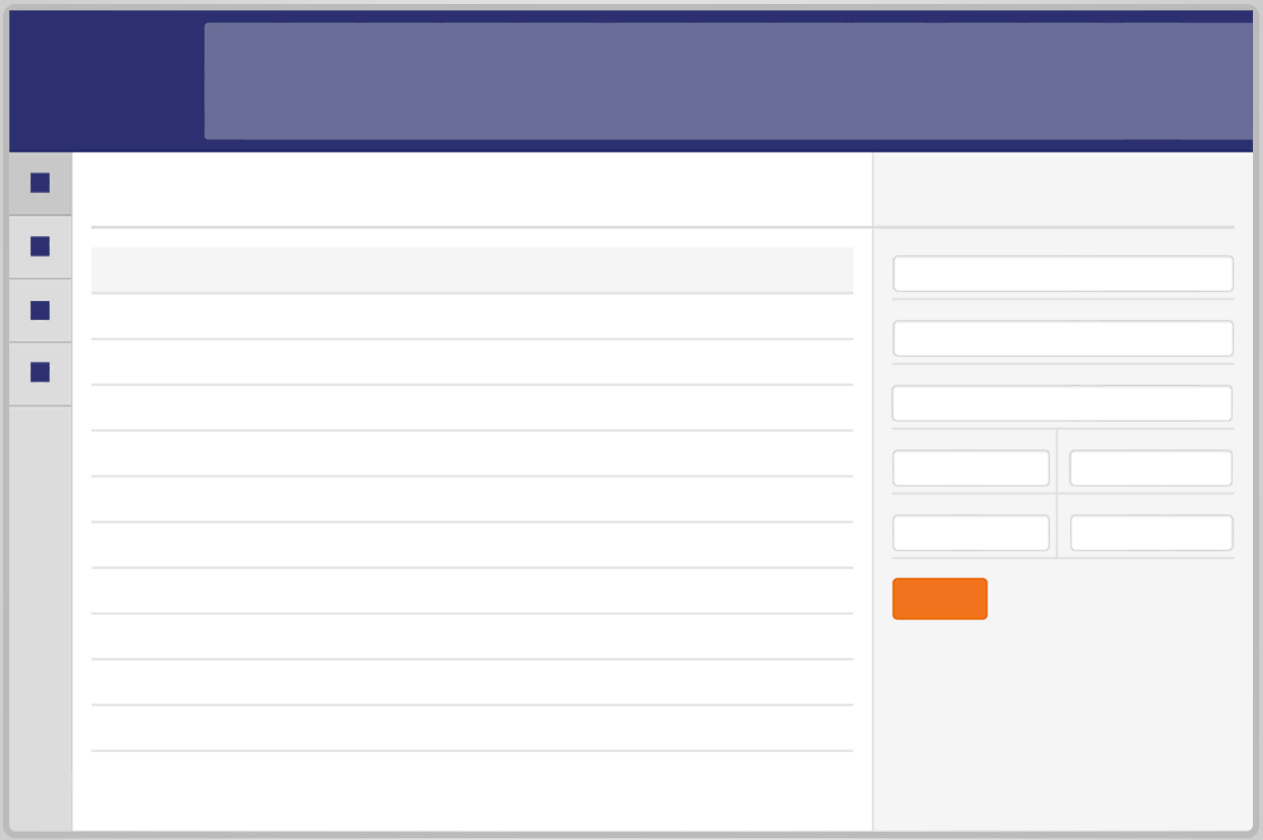

```
<h3>Welcome</h3>

<input type="text" ng-model="yourName">

<p ng-if="yourName">
  Hello <b>{{ yourName }}</b>!<br>
  I hope that your Jfokus presentation
  is going fine.
</p>
```

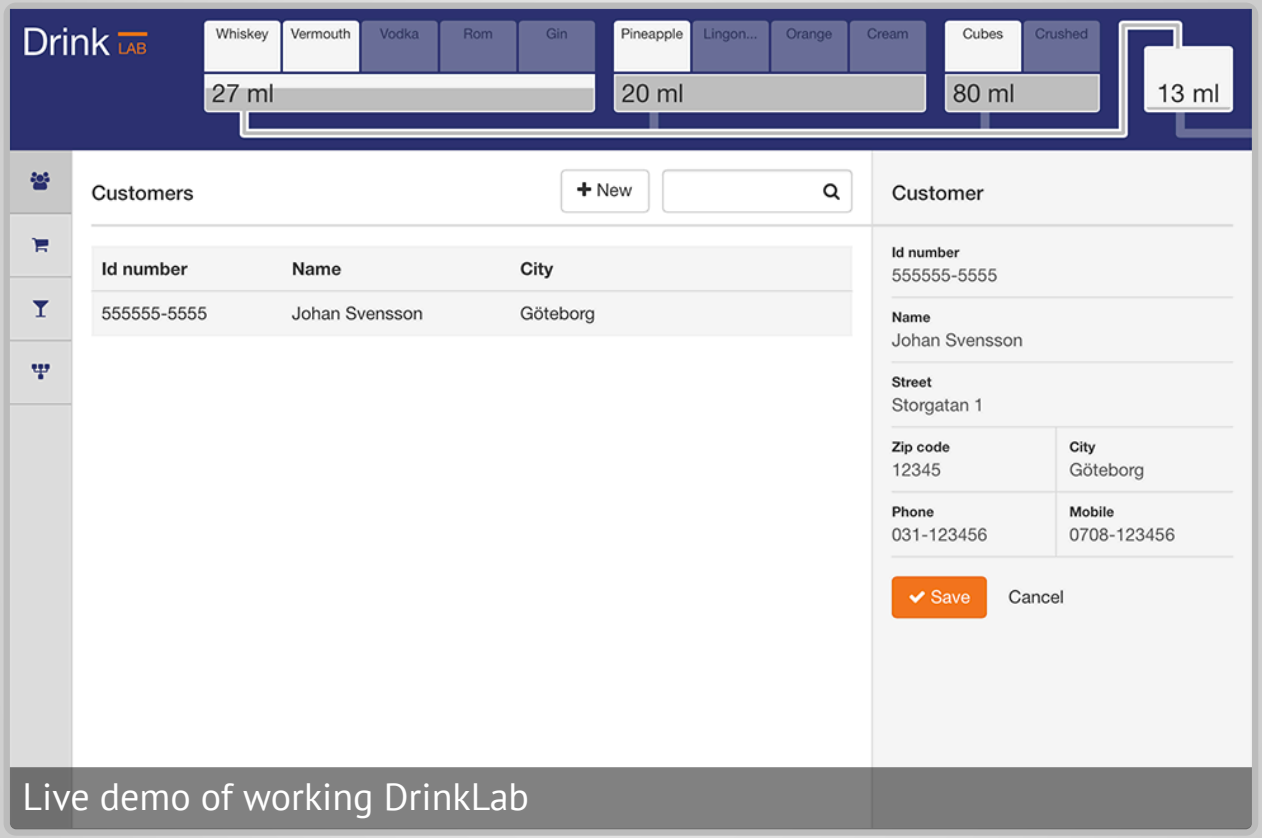





http://127.0.0.1:3000/images/demo5.png


http://127.0.0.1:3000/images/demo6.png





Live demo of working DrinkLab

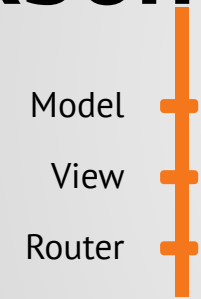
Backbone.js

Backbone.js

Quick facts

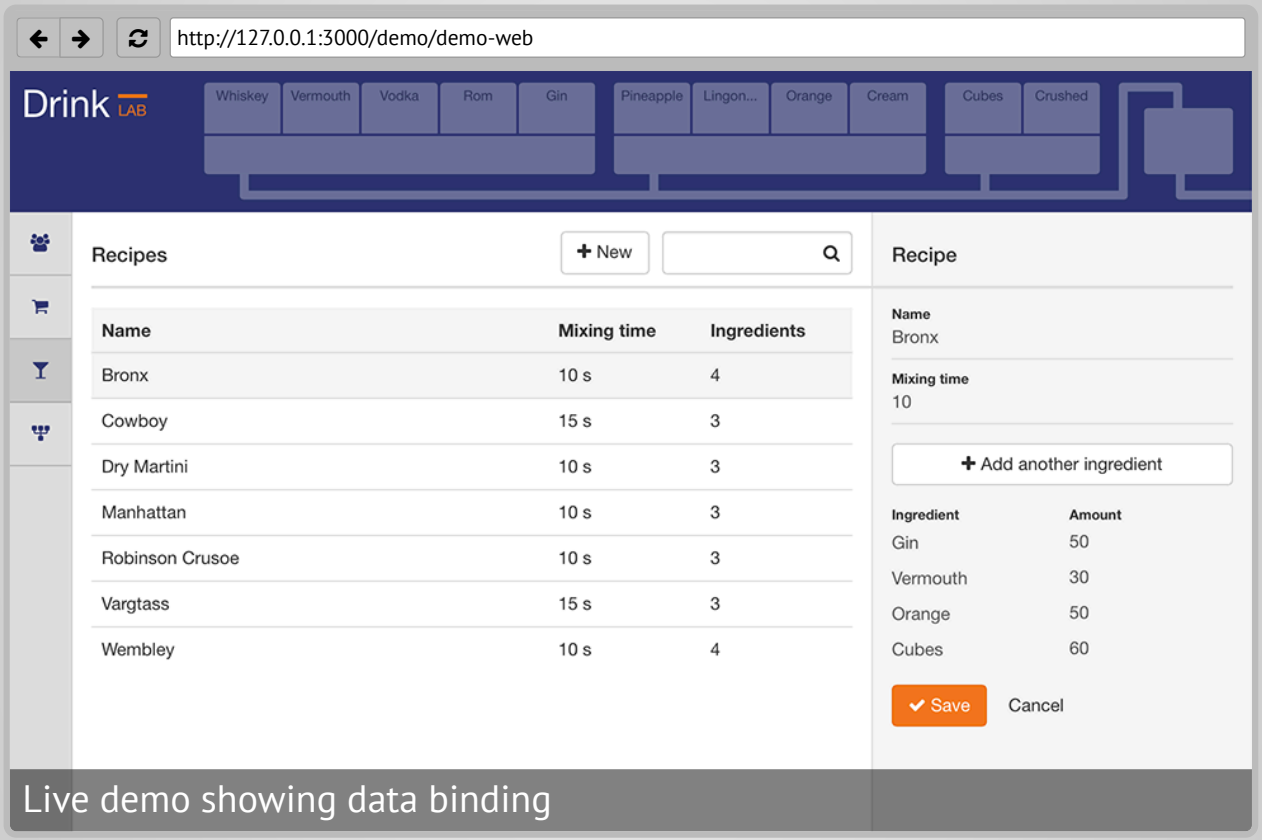
First release	2010, october
Developers	Jeremy Ashkenas
Version	1.1.0
Size	20 + 14 KB
★ Stars	16 940
Forks	3 662

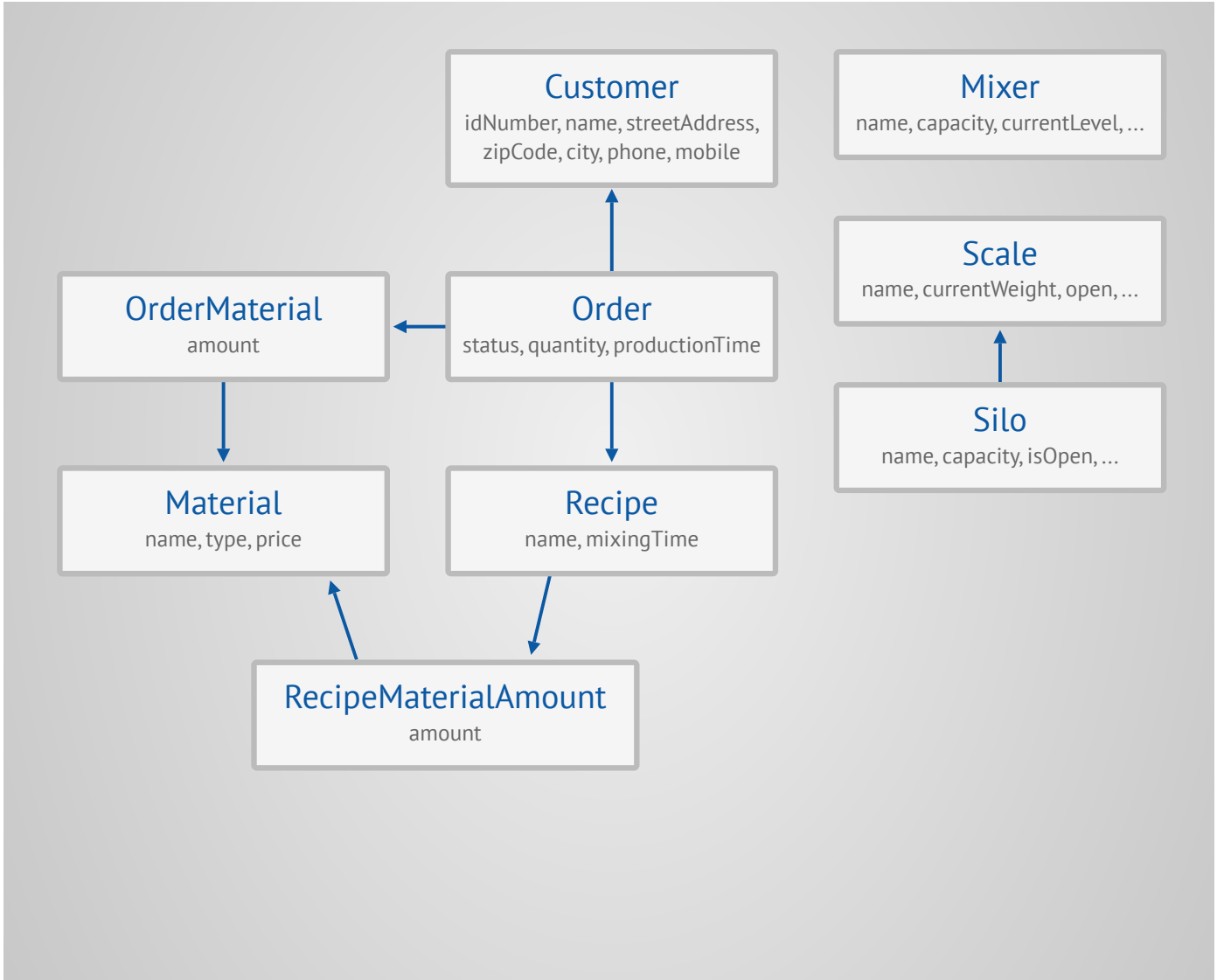
Backbone.js

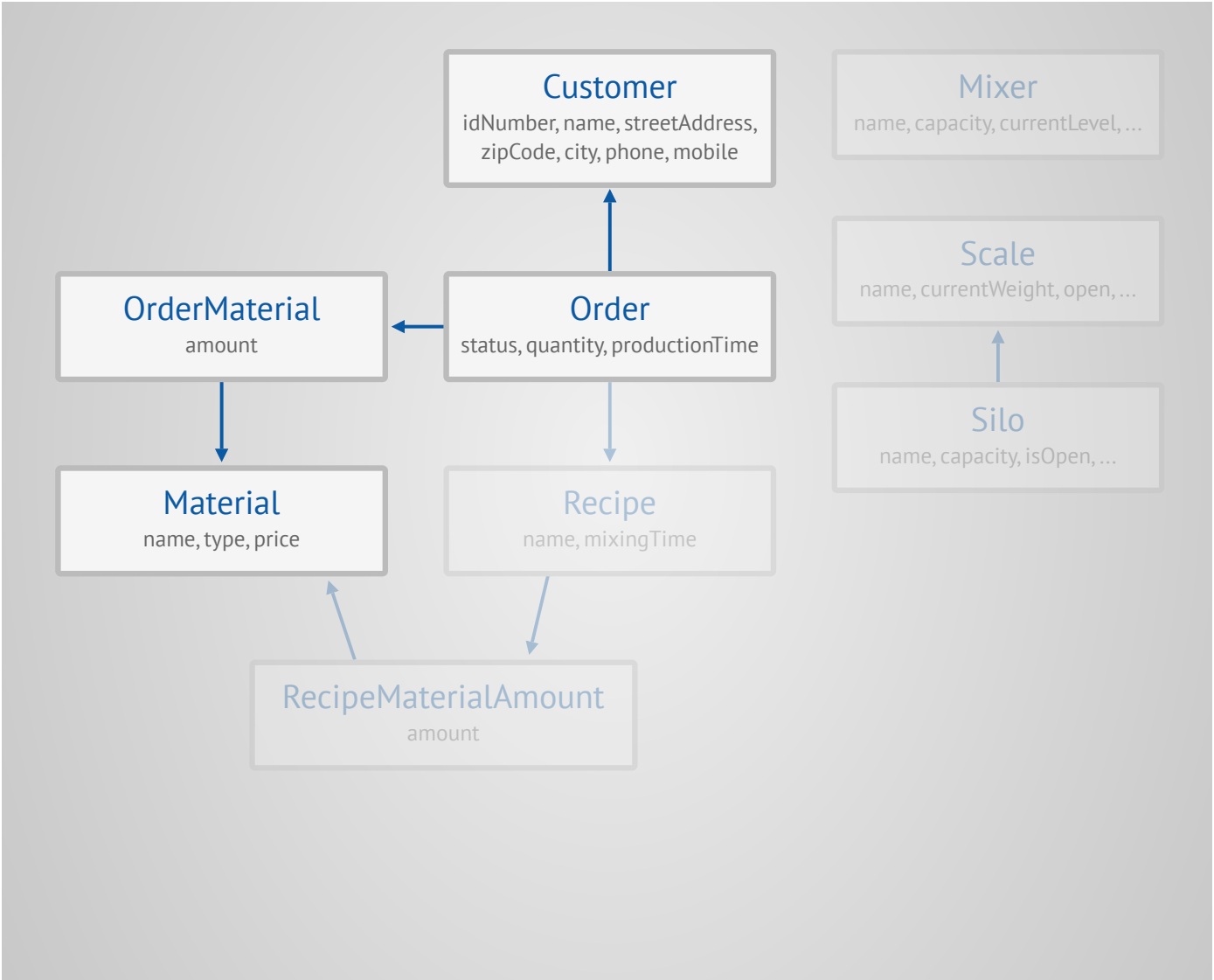


Quick facts

First release	2010, october
Developers	Jeremy Ashkenas
Version	1.1.0
Size	20 + 14 KB
★ Stars	16 940
Forks	3 662


http://127.0.0.1:3000/demo/demo-web
Drink LAB
Whiskey
Vermouth
Vodka
Rom
Gin
Pineapple
Lingon...
Orange
Cream
Cubes
Crushed
Recipes
+ New
Name
Mixing time
Ingredients
Bronx
10 s
4
Cowboy
15 s
3
Dry Martini
10 s
3
Manhattan
10 s
3
Robinson Crusoe
10 s
3
Vargtass
15 s
3
Wembley
10 s
4
Recipe
Name
Bronx
Mixing time
10
+ Add another ingredient
Ingredient
Amount
Gin
50
Vermouth
30
Orange
50
Cubes
60
Save
Cancel
Live demo showing data binding


Customer
idNumber, name, streetAddress, zipCode, city, phone, mobile
Mixer
name, capacity, currentLevel, ...
Scale
name, currentWeight, open, ...
OrderMaterial
amount
Order
status, quantity, productionTime
Silo
name, capacity, isOpen, ...
Material
name, type, price
Recipe
name, mixingTime
RecipeMaterialAmount
amount


Customer
idNumber, name, streetAddress, zipCode, city, phone, mobile
Mixer
name, capacity, currentLevel, ...
Scale
name, currentWeight, open, ...
OrderMaterial
amount
Order
status, quantity, productionTime
Silo
name, capacity, isOpen, ...
Material
name, type, price
Recipe
name, mixingTime
RecipeMaterialAmount
amount

Customer

```
// Declare the model and collection
DL.Model.Customer = Backbone.Model.extend({
  defaults: { city: 'Stockholm' }
});
DL.Collection.Customer = Backbone.Collection.extend({
  model: DL.Model.Customer,
  url: 'http://localhost:8080/api/customers'
});
```

```
// Get a collection of customers from the server
var customers = new DL.Collection.Customer();
customers.fetch({
  success: function (customers) {
    console.log('Successfully fetched customers');
  }
});
```

```
// Update a model and save it
```

```
  url: 'http://localhost:8080/api/customers'
});
```

```
// Get a collection of customers from the server
var customers = new DL.Collection.Customer();
customers.fetch({
  success: function (customers) {
    console.log('Successfully fetched customers');
  }
});
```

```
// Update a model and save it
var customer = customers.at(0);
customer.set({ name: 'Kalle Svensson' });
customer.save({}, {
  success: function (customer) {
    console.log('Successfully saved a customer');
  }
});
```

```
  }
});
```

```
// Update a model and save it ————————————————————————————
var customer = customers.at(0);
customer.set({ name: 'Kalle Svensson' });
customer.save({}, {
  success: function (customer) {
    console.log('Successfully saved a customer');
  }
});
```

Order data

```
{
  "status": "Created",
  "quantity": 1,
  "productionTime": "2014-01-10T09:00:00",
  "customer": 1,
  "recipe": 1,
  "materialAmounts": [
    {
      "amount": 40,
      "material": "1"
    },
    ...
  ]
}
```

Order

```
// Declare the model and collection of the relation ————————————
DL.Model.MaterialAmount = Backbone.Model.extend();
DL.Collection.MaterialAmount = Backbone.Collection.extend({
    model: DL.Model.MaterialAmount
});


// Declare the order model ——————————————————————————————————
DL.Model.Order = Backbone.Model.extend({
  defaults: { quantity: 1 },

  // Declare how our relations should behave ————————————————
  relations: {
    materialAmounts: DL.Collection.MaterialAmount
  },

  // Override parse to handle relations ———————————————————
  parse: function (response) {
```

```
DL.Model.Order = Backbone.Model.extend({
  defaults: { quantity: 1 },

  // Declare how our relations should behave ————————————————
  relations: {
    materialAmounts: DL.Collection.MaterialAmount
  },

  // Override parse to handle relations ————————————————————
  parse: function (response) {
    for (var key in this.relations) {
      if (this.relations.hasOwnProperty(key)) {
        response[key] =
          new this.relations[key](response[key],
              { parse: true });
      }
    }
    return response;
  }
});
```

```
    materialAmounts: DL.Collection.MaterialAmount
  },

  // Override parse to handle relations ————————————————————
  parse: function (response) {
    for (var key in this.relations) {
      if (this.relations.hasOwnProperty(key)) {
        response[key] =
          new this.relations[key](response[key],
              { parse: true });
      }
    }
    return response;
  }
});
```

Customers.html

```
...
<tbody>
  <% _.each(customers, function(customer) { %>
  <tr>
    <td><%= customer.idNumber %></td>
    <td><%= customer.name %></td>
    <td><%= customer.city %></td>
  </tr>
  <% }); %>
</tbody>
...
```

Order

```
var dlServices = angular.module('dlServices', ['ngResource']);
dlServices.factory('Order', [ '$resource',
  function ($resource) {
    return $resource('api/orders/:orderId',
        { orderId: '@id' });
  }
]);
```

```
var dlApp = angular.module('dlApp', ['dlServices']);
dlApp.controller('OrderListCtrl', [
  '$scope', 'Order', function ($scope, Order) {

    $scope.orders = Order.query();

    $scope.save = function () {
      var order = $scope.orders[0];
      order.quantity = 2;
      order.$save();
    };
```

```
  }
]);
```

```
var dlApp = angular.module('dlApp', ['dlServices']);
dlApp.controller('OrderListCtrl', [
  '$scope', 'Order', function ($scope, Order) {

    $scope.orders = Order.query();

    $scope.save = function () {
      var order = $scope.orders[0];
      order.quantity = 2;
      order.$save();
    };
  }
]);
```

Order data

```
{
  "status": "Created",
  "quantity": 1,
  "productionTime": "2014-01-10T09:00:00",
  "customer": {
    "id": 1,
    "name": "Kalle Svensson"
  },
  "recipe": ...,
  "materialAmounts": [
    {
      "amount": 40,
      "material": ...
    },
    ...
```

Orders.html

```
<tbody>
  <tr ng-repeat="order in orders">
    <td>{{ order.customer.name }}</td>
    <td>{{ order.quantity }}</td>
    <td>{{ order.recipe.name }}</td>
    <td>
      <ul>
        <li ng-repeat="amount in order.materialAmounts">
          {{ amount.material.name }}
        </li>
      </ul>
    </td>
  </tr>
</tbody>
```

See **Marius Gundersen at JSConf EU** for performance considerations

Orders.html

```
<tbody>
  <tr ng-repeat="order in orders">
    <td>{{ order.customer.name }}</td>
    <td>{{ order.quantity }}</td>
    <td>{{ order.recipe.name }}</td>
    <td>
      <ul>
        <li ng-repeat="amount in order.materialAmounts">
          {{ amount.material.name }}
        </li>
      </ul>
    </td>
  </tr>
</tbody>
```

See **Marius Gundersen at JSConf EU** for performance considerations

Order

```
// Declare the order model ----------
DL.Order = DS.Model.extend({
  status: DS.attr(),
  quantity: DS.attr('number'),
  productionTime: DS.attr('date'),

  customer: DS.belongsTo('customer'),
  recipe: DS.belongsTo('recipe'),

  materialAmounts: DS.hasMany('orderMaterialAmount')
});
```

```
// Declare the order material amount model ----------
DL.OrderMaterialAmount = DS.Model.extend({
  amount: DS.attr('number'),

  material: DS.belongsTo('material')
});
```

```
  materialAmounts: DS.hasMany('orderMaterialAmount')
});

// Declare the order material amount model ----------
DL.OrderMaterialAmount = DS.Model.extend({
  amount: DS.attr('number'),

  material: DS.belongsTo('material')
});

// Get a collection of orders from the server -------
var orders = this.store.find('order');
orders.then(function (orders) {
  console.log('Successfully fetched orders');
});

// Update a model and save it -----------------------
var order = orders.firstObject();
order.set('name', 'Kalle Svensson');
order.save().then(function (order) {
  console.log('Successfully saved a order');
});
```

```
  material: DS.belongsTo('material')
});
```

```
// Get a collection of orders from the server
var orders = this.store.find('order');
orders.then(function (orders) {
  console.log('Successfully fetched orders');
});
```

```
// Update a model and save it
var order = orders.firstObject();
order.set('name', 'Kalle Svensson');
order.save().then(function (order) {
  console.log('Successfully saved a order');
});
```

```
  console.log('Successfully fetched orders');
});
```

```
// Update a model and save it ————————————————————————————
var order = orders.firstObject();
order.set('name', 'Kalle Svensson');
order.save().then(function (order) {
  console.log('Successfully saved a order');
});
```

Order

```
<tbody>
  {{#each order in orders}}
  <tr>
    <td>{{ order.customer.name }}</td>
    <td>{{ order.quantity }}</td>
    <td>{{ order.recipe.name }}</td>
    <td>
      <ul>
        {{#each amount in order.materialAmounts}}
        <li>{{ amount.material.name }}</li>
        {{/each}}
      </ul>
    </td>
  </tr>
  {{/each}}
</tbody>
```

Knockout

Knockout

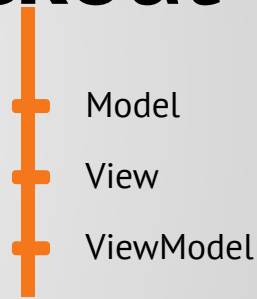
Quick facts

First release	2010, july
Developers	Steve Sandersson
Version	3.0.0
Size	46 KB
★ Stars	4 585
Forks	755

Quick facts

First release	2010, july
Developers	Steve Sandersson
Version	3.0.0
Size	46 KB
★ Stars	4 585
Forks	755

Knockout



Orders.html

```
...
<tbody data-bind="foreach: customers">
  <tr>
    <td data-bind="text: idNumber"></td>
    <td data-bind="text: name"></td>
    <td data-bind="text: city"></td>
  </tr>
</tbody>
...
```

Knockback

```
var model = new DL.Model.Order({ ... });

// Create a view model from our model
var view_model = Knockback.viewModel(model);
```

Knockback

```
var model = new DL.Model.Order({ ... });

// Create a view model from our model ———————————————————
var view_model = Knockback.viewModel(model);

// Apply the bindings to the view ———————————————————————
Knockout.applyBindings(view_model, $('#our-view')[0]);
```

Technical Dept

Custom solutions

Technical Dept

Custom solutions

Technical Dept

Dependencies

Technical Dept

Impossible to maintain

Access server data

1. Ember
2. Backbone
3. Angular

Access server data

1 Ember 2 Backbone 3 Angular

Handle relational data

1 Ember 2 Angular 3 Backbone

Access server data

1 Ember 2 Backbone 3 Angular

Handle relational data

1 Ember 2 Angular 3 Backbone

Bind data to the DOM

1 Angular 2 Ember 3 Backbone

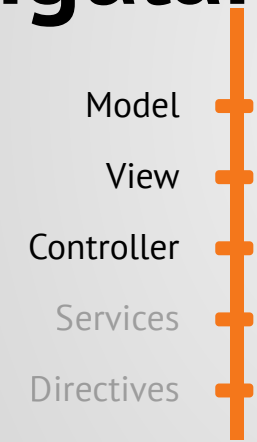
AngularJS

AngularJS

Quick facts

First release	2009
Developers	Google Inc. & community
Version	1.2.10
Size	100 + 33 KB
★ Stars	19 758
Forks	5 839

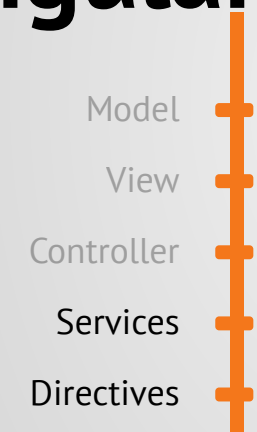
AngularJS



Quick facts

First release	2009
Developers	Google Inc. & community
Version	1.2.10
Size	100 + 33 KB
★ Stars	19 758
Forks	5 839

AngularJS



Quick facts

First release	2009
Developers	Google Inc. & community
Version	1.2.10
Size	100 + 33 KB
★ Stars	19 758
Forks	5 839

Ember.js

Ember.js

Quick facts

First release	2011
Developers	Yehuda Katz, Tom Dale & contributors
Version	1.3.1
Size	263 + 77 KB
★ Stars	9 256
Forks	1 962

Quick facts

First release	2011
Developers	Yehuda Katz, Tom Dale & contributors
Version	1.3.1
Size	263 + 77 KB
★ Stars	9 256
Forks	1 962

Ember.js

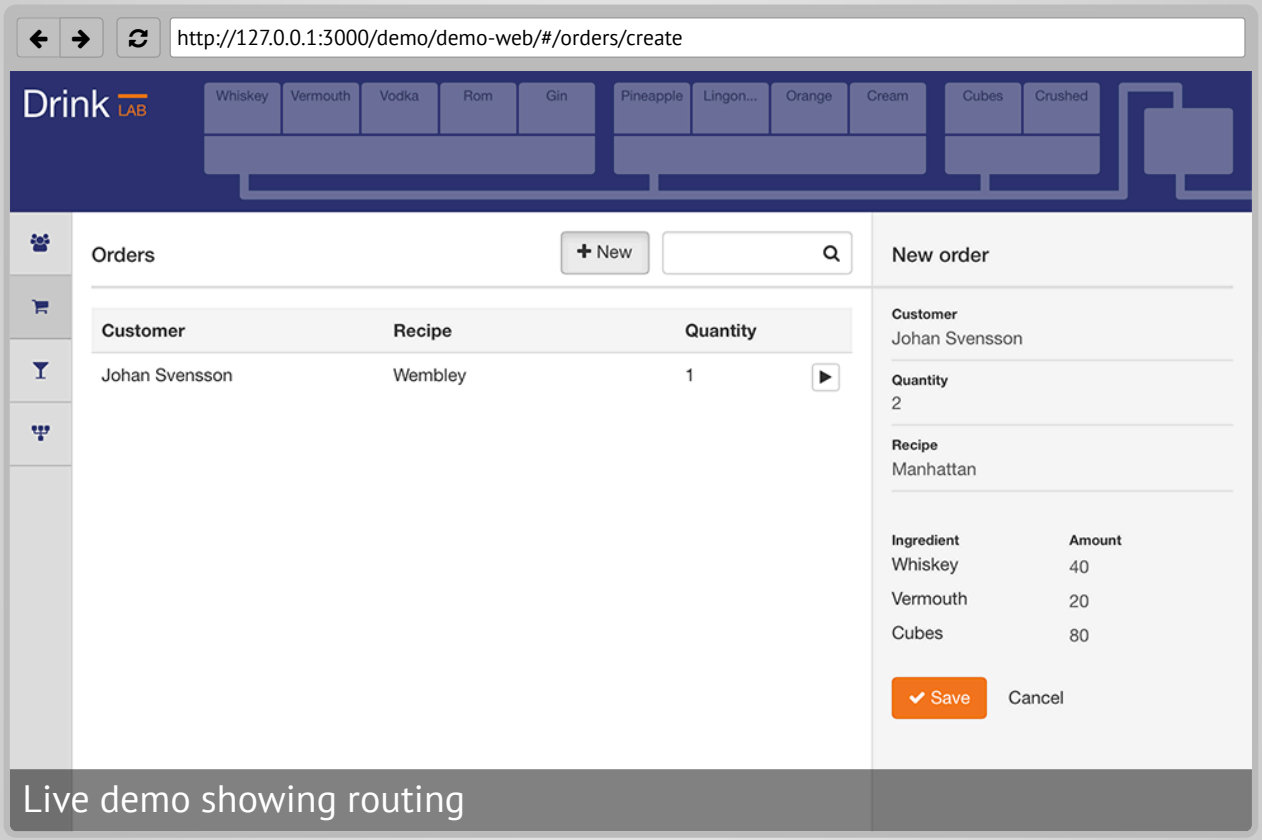
- Model
- View
- Controller
- Router
- Components

Quick facts

First release	2011
Developers	Yehuda Katz, Tom Dale & contributors
Version	1.3.1
Size	263 + 77 KB
★ Stars	9 256
Forks	1 962

Ember.js

- Model
- View
- Controller
- Router
- Components


http://127.0.0.1:3000/demo/demo-web/#/orders/create
Drink LAB
Whiskey
Vermouth
Vodka
Rom
Gin
Pineapple
Lingon...
Orange
Cream
Cubes
Crushed
Orders
+ New
Customer
Recipe
Quantity
Johan Svensson
Wembley
1
New order
Customer
Johan Svensson
Quantity
2
Recipe
Manhattan
Ingredient
Amount
Whiskey
40
Vermouth
20
Cubes
80
Save
Cancel
Live demo showing routing

Routing

```
// Create a router for the application
DL.Routers.ApplicationRouter = Backbone.Router.extend({

  routes : {
    'customers': 'listCustomers',
    'customers/:customerId': 'editCustomer',
    'customers/create': 'createCustomer'
  },

  // Show a list of customer
  listCustomers: function () {
    var customers = new DL.Collection.Customer();
    customers.fetch({
      success : function(collection) {
        new DL.Views.CustomerListView({
          collection : collection
        });
      }
    });
```

```
  'customers/create': 'createCustomer'
},

// Show a list of customer ------------------------------------------
listCustomers: function () {
  var customers = new DL.Collection.Customer();
  customers.fetch({
    success : function(collection) {
      new DL.Views.CustomerListView({
        collection : collection
      });
    }
  });
},

// Show the edit form -----------------------------------------------
editCustomer: function (customerId) {
  var customer = new DL.Model.Customer({id : customerId});
  customer.fetch({
    success : function(model) {
      new DL.Views.CustomerEditView({
        model : model
      });
    }
```

```
    });
  },

  // Show the edit form ------------------------------------------
  editCustomer: function (customerId) {
    var customer = new DL.Model.Customer({id : customerId});
    customer.fetch({
      success : function(model) {
        new DL.Views.CustomerEditView({
          model : model
        });
      }
    });

    // Make sure that the list is also visible ---------------------
    this.listCustomers();
  }
});
```

```
        }
    });

    // Make sure that the list is also visible ————————————————
    this.listCustomers();
  }
});
```

Routing

```
// Create a router for the application
DL.Router.map(function () {

  this.resource('customers', function () {
    this.route('edit', { path: ':customer_id' });
    this.route('create');
  });
});
```

```
// Configure this edit customer route
DL.CustomersEditRoute = Ember.Route.extend({

  model: function (params) {
    return this.store.find('customer', params.customer_id);
  }
});
```

```
// Configure the list customers route
```

```
  });
});
```

```
// Configure this edit customer route ————————————————————
DL.CustomersEditRoute = Ember.Route.extend({

  model: function (params) {
    return this.store.find('customer', params.customer_id);
  }
});
```

```
// Configure the list customers route ————————————————————
DL.CustomersRoute = Ember.Route.extend({

  model: function () {
    return this.store.find('customer');
  }
});
```

```
  }
});
```

```
// Configure the list customers route ————————————————————————
DL.CustomersRoute = Ember.Route.extend({

  model: function () {
    return this.store.find('customer');
  }
});
```

Routing

```
angular.module('dlApp').config([ '$routeProvider',
  function($routeProvider) {

    $routeProvider.when('/customers', {
      templateUrl : '/views/customers/index.html',
      controller : 'CustomersCtrl'

    }).when('/customers/:customerId', {
      templateUrl : '/views/customers/edit.html',
      controller : 'CustomerEditCtrl'

    }).otherwise({
      redirectTo : '/customers'
    });
  }
]);
```

For nested routes in Angular, try **AngularUI Router**

Routing

```
angular.module('dlApp').config([ '$routeProvider',
  function($routeProvider) {

    $routeProvider.when('/customers', {
      templateUrl : '/views/customers/index.html',
      controller : 'CustomersCtrl'

    }).when('/customers/:customerId', {
      templateUrl : '/views/customers/edit.html',
      controller : 'CustomerEditCtrl'

    }).otherwise({
      redirectTo : '/customers'
    });
  }
]);
```

For nested routes in Angular, try **AngularUI Router**

Handle advanced routing

1. Ember
2. Angular
3. Backbone

Handle advanced routing

1 Ember 2 Angular 3 Backbone

Animate transitions

1 Angular 2 Ember, Backbone

Perform simple validation

1. Angular
2. Backbone
3. Ember

Perform simple validation

1 Angular 2 Backbone 3 Ember

Perform advanced validation

1 Angular 2 Ember 3 Backbone

6

Perform simple validation

1 Angular 2 Backbone 3 Ember

7

Perform advanced validation

1 Angular 2 Ember 3 Backbone

8

Handle “static” areas

1 Angular, Ember 2 Backbone

6 Perform simple validation

1 Angular 2 Backbone 3 Ember

7 Perform advanced validation

1 Angular 2 Ember 3 Backbone

8 Handle “static” areas

1 Angular, Ember 2 Backbone

9 React to WebSocket updates

1 Backbone 2 Ember 3 Angular

Winner?

Angular

Angular

...for now.

Angular vs Ember vs Backbone
(and Knockout) in real applications

Joakim Kemeny
@joakimkemeny

CALLISTA
Enterprise