Ember.js

Jan Kopřiva, Vojtěch Jasný

Představení

- Vojta a Honza
- frontend engineers @ GoodData
- už nějakou dobu píšeme webové aplikace v JavaScriptu
- ...posledních pár let v Ember.js :-)
- program: průlet Emberem pro začátečníky a dojde i na live coding demo

Čím se GoodData zabývají

- Děláme business intelligence v cloudu
- Vize: BI pro masy
- Platforma: od uploadu dat k vizualizaci
- Široce dostupné, proto v JS a v cloudu

Úvod

Co jsou to webové aplikace?

Problém: DOM

Moře knihoven a frameworků

YUI 2 & YUI 3
jQuery
Capuccino
SproutCore
JavaScriptMVC
Dojo Toolkit
Backbone.js & Spine

Knockout
Ember
Angular
Meteor

...

React + Flux + Immutable.js?

GoodData use case – YUI 2

```
(function() {
    var dd, dd2, dd3;
    YAHOO.util.Event.onDOMReady(function() {
        dd = new YAHOO.util.DD("dd-demo-1");
        dd2 = new YAHOO.util.DD("dd-demo-2");
        dd3 = new YAHOO.util.DD("dd-demo-3");
    });
})();
```

GoodData use case – YUI 3

```
YUI().use('node', function(Y){
    var boxes = Y.all('.box-row li');
    var handleBoxClick = function(e) {
        boxes.setStyle('backgroundColor', '#F4E6B8');
        e.currentTarget.setHTML('ouch!');
        e.currentTarget.setStyle('backgroundColor', '#C4DAED');
    };
    Y.one('.box-row').delegate('click', handleBoxClick, 'li');
});
```

GoodData use case

YUI3 + custom MVC

Ember.js

Co je Ember.js

- MVC aplikační framework pro tvorbu SPA
- původ ve SproutCore
- Tom Dale & Yehuda Katz
- inspirace Cocoa a Ruby on Rails
- důraz na konvence – “opinionated”
- hlavní konkurent – angular.js


ember

Hlavní přísady

Model
Aplikace
Router
Controller

Šablony
Komponenty
Container
Run loop

Ember.Object (model)

- předek všeho v Ember.js
- poskytuje:
  - implementaci “klasické” dědičnosti
  - bindings
  - observers
  - computed properties
- nutnost používat get/set!

Ember.Application

- srdce emberové aplikace
- tvoří jmenný prostor aplikace
- koordinuje činnost ostatních objektů
- pomocí routeru a containeru startuje celou aplikaci

Ember.Router

- bookmarkovatelný stav – funkční back a forward v browseru
- říká jak budou vypadat URLs
- zachycuje hierarchickou strukturu aplikace
- v route dochází k nahrání modelu
- route+controller+view

Ember.Router (příklad)

```
App.Router.map(function() {
  this.resource(notes, function() {
    this.route('note', { path: ':title' });
  });
});
```

/notes/moje-poznamka/

Ember.Controller

view controller = zobrazovací logika pro view:

```
export default Ember.ObjectController.extend({
    duration: function() {
        var duration = this.get('model.duration'),
        minutes = Math.floor(duration / 60),
        seconds = duration % 60;

        return [minutes, seconds].join(':');
    }.property('model.duration')
});
```

Templaty (handlebars)

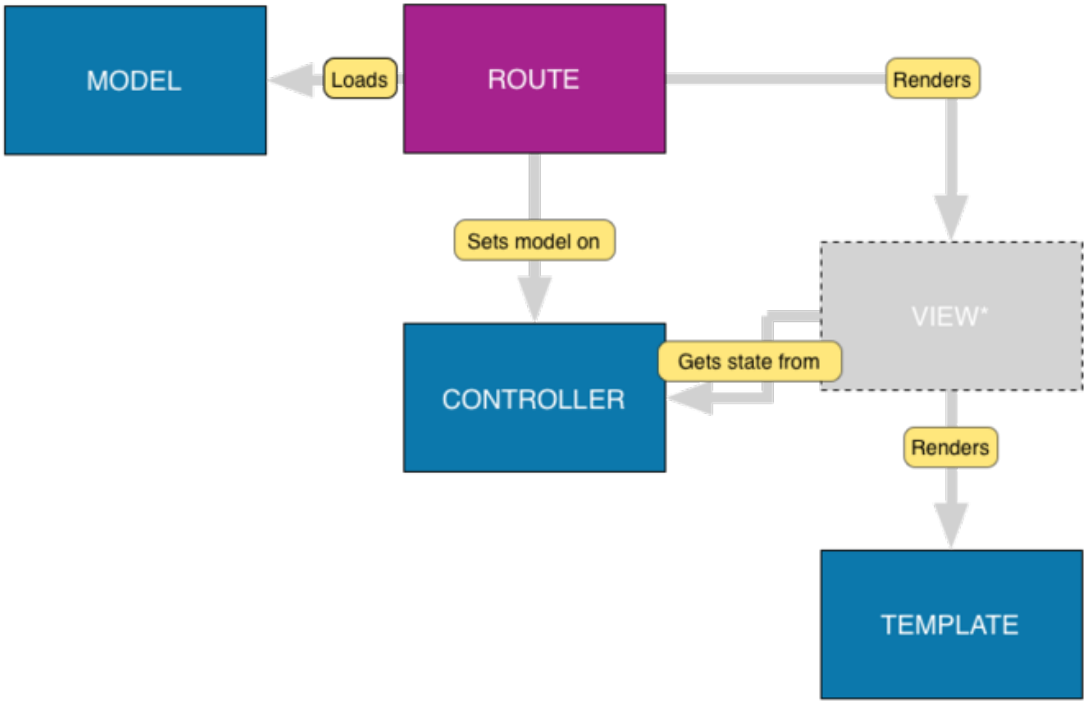
```
<ul class="teams">
    {{#each team in teams}}
    <li>
        {{#linkTo "team" team}}
            {{team.name}}
        {{/linkTo}}
    </li>
    {{/each}}
</ul>
```

Ember.View

v případě, že chci složitější event handling dom events na app events:

```
export default Ember.View.extend({
    click: function() {
        this.get('controller').send('openDialog');
    }
});
```

Ember MVC



* There is actually no longer a need to use a view, but for now this is how Ember works

Ember.Component

znovupoužitelné (self-contained) elementy

chování popsáno v JS:

```
{{#each}}
   {{#blog-post title=title action=”delete”}}
     {{body}}
   {{/blog-post}}
{{/each}}
```

Ember.Container – DI

```
export default Ember.ObjectController.extend({
    actions: {
        findItems: function(){
            var controller = this;
            this.store.find('item').then(function(items){
                controller.set('items', items);
            });
        }
    }
});
```

Ember.Container – API

```
var logger = {
    log: function(message) {
        console.log(message);
    }
}

application.register('logger:main', logger,
        { instantiate: false });

application.inject('controller', 'logger', 'logger:main');
```

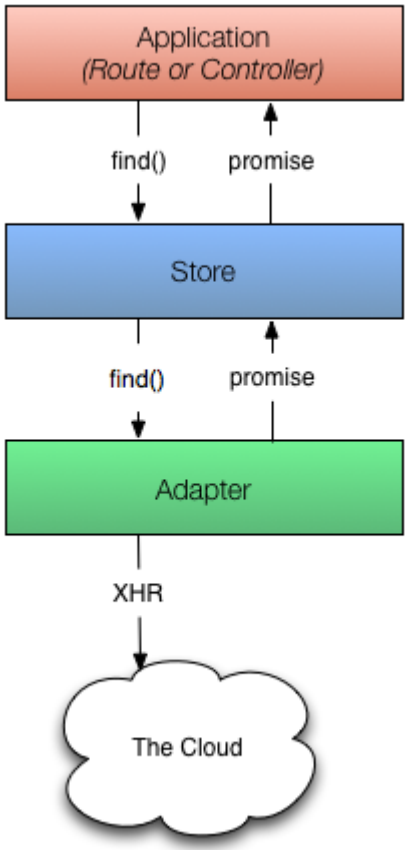
Run loop

- mechanismus pro seskupení některých drahých operací → update DOM
- efektivní rendering
- fronty – akce, bindingy, render, afterRender
- naplánovat do fronty lze i vlastní funkce

Ember Data

- ORM v JS v browseru

- model
- store
- adapter
- serializer



Ember Data - Model

- reprezentuje vlastnosti a chování dat uložených na serveru

```
export default DS.Model.extend({
  firstName: DS.attr('string'),
  birthday:  DS.attr('date')
  articles: DS.hasMany('articles')
});
```

Ember Data – Store

- centrální repozitář (cache) modelů v aplikaci
- přístupný v routách a kontrolerech (DI)

Ember Data – Adaptér

- komunikuje s backendem
- zajišťuje, aby requesty vyhovovaly API
  - websockets, http, ...
  - správný formát url

Ember Data – Serializér

- překládá raw data na recordy
- normalizuje data do formátu pro Ember Data
- mění formát klíčů
  - user_name ⟶ userName

Testování

- důležité při výběru frameworku
- unit testy
- end to end testy
- více unit testů, poměr cca. 5:1
- test integrace s backendem - Selenium

Ember CLI

- nástroj v příkazové stránce
- standard jak začít s novou Ember app
- vše je pro vás připraveno:
  - generátory
  - konvence
  - testy a build nástroje

Demo

- api/friends
- api/friends/1234
- api/v2/friends/1234/articles

Výhody Ember.js

- silná komunita
- předvídatelný release cyklus
- slušná dokumentace
- jasná struktura umožňuje dalším vývojářům rychle naskočit do projektu
- efektivní spolupráce s Rails
- inspirace konkurencí

Nevýhody Ember.js

- složitá abstrakce → komplexita → chyby
- run loop je dobrý sluha, ale zlý pán
- nestandardní API → žádná Ember Data
- není za ním velká firma
- menší uživatelská báze
- učící křivka

Budoucnost – Fastboot

- kontext – obsahové weby
- problém:
  - pomalý start aplikací
  - špatné pro indexování vyhledávači
- řešení: generovat HTML už na serveru
- následuje oživení v prohlížeči
- zdarma pro (skoro) všechny Ember apps

Budoucnost – Glimmer

- inspirace z React.js
- naivní implementace: vykreslit vždy celou aplikaci znova
- to ale přeci nejde!
- ve skutečnosti jde – virtual DOM & diff
- výrazné zlepšení výkonnosti view vrstvy
- zjednoduší uvažování o celé aplikaci

New Paradigms

React.js

- JS knihovna pro tvorbu UI
  - view vrstva
- od Facebooku
- jiný přístup:
  - přerendruj všechno od začátku
- rychlý, zatraceně rychlý
  - virtual DOM, diffing, rendering nejen do DOMu
- ostatní, i Ember (Glimmer), se inspirují

React.js

- žádné templaty
  - žádná omezení z toho plynoucí
  - bez dalších abstrakcí
- jen JS, dostatečne mocný na tvorbu UI
- jednoduché API
- => netřeba se učit nic moc nového

Reactive Extensions

- základem observovatelný proud událostí
- události jsou kolekce rozložená v čase
- aplikujeme stejné fce map/reduce atd.
- proudy lze navzájem různě kombinovat
- jiný způsob jak uvažovat o stavbě aplikace

Webpack

- pokročilá knihovna pro build aplikace
- JS aplikace = 1 minifikovaný soubor
- rozšiřitelnost pomocí loaderů
- Babel.js – transpiler do EcmaScript 6
- další možnosti: hotloading

Immutable.js

- JavaScript – datové struktury můžeme měnit Array.push, obj[key] = value, atd.
- Immutable.js – podpora pro immutable struktury v JavaScriptu
- výhody:
  - bezpečnější kód bez defenzivního kopírování
  - snadné porovnání (referencí)

Děkujeme za pozornost!