# ANGULARJS

by Google

## als Oberfläche für Enterprise Java Projekte

**Bastian Voigt**
@BastianVoigt
xing.com/profile/Bastian_Voigt

ConceptPeople IT-Talk 8.09.2014

# Gliederung

1) Über mich

2) Was ist Enterprise Java Softwareentwicklung?

3) Spaßbremsen bei der Web-Entwicklung mit Java

4) Was ist AngularJS ?

5) Wie kann der Buildprozess aussehen?

6) Best Practices

# Über mich

Dipl.-Inf. Bastian Voigt

**freier softwareentwickler**

Freiberuflicher Consultant seit 2008

Schwerpunkte:

Java-Entwicklung, Projektmanagement

Agil und pragmatisch

Training und Coaching

# Was ist Enterprise Java?

- Serverbasierte Java-Anwendung mit Web-Oberfläche

  EJB, Spring, JSP, JSF, Wicket, Hibernate, etc.

- Langlebige Software
  - Viele Projekte sind heute schon 10-15 Jahre alt
  - Werden immer noch aktiv weiterentwickelt

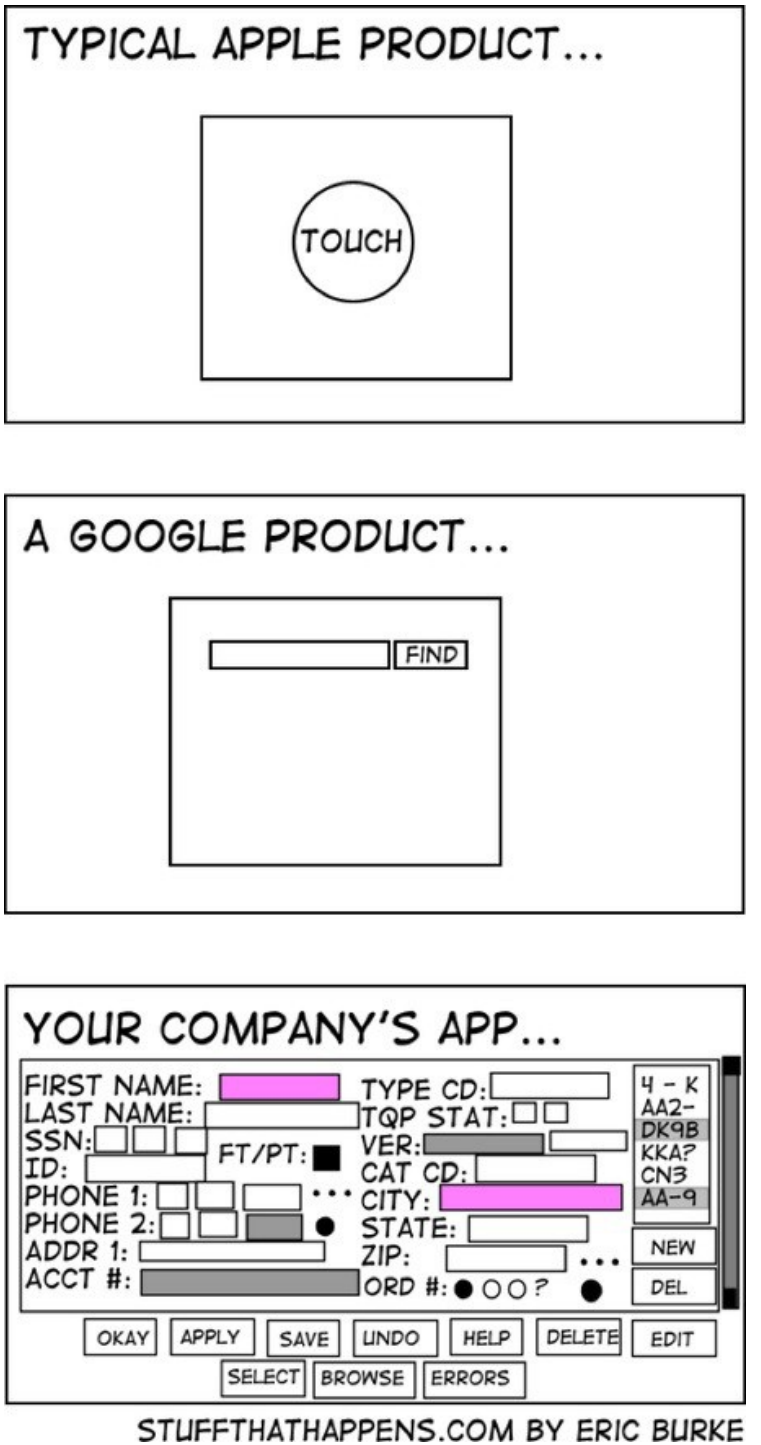
- Meist geschäftliche / kaufmännische Anwendungen
  - Geschäftsprozesse, Kundendaten, etc.

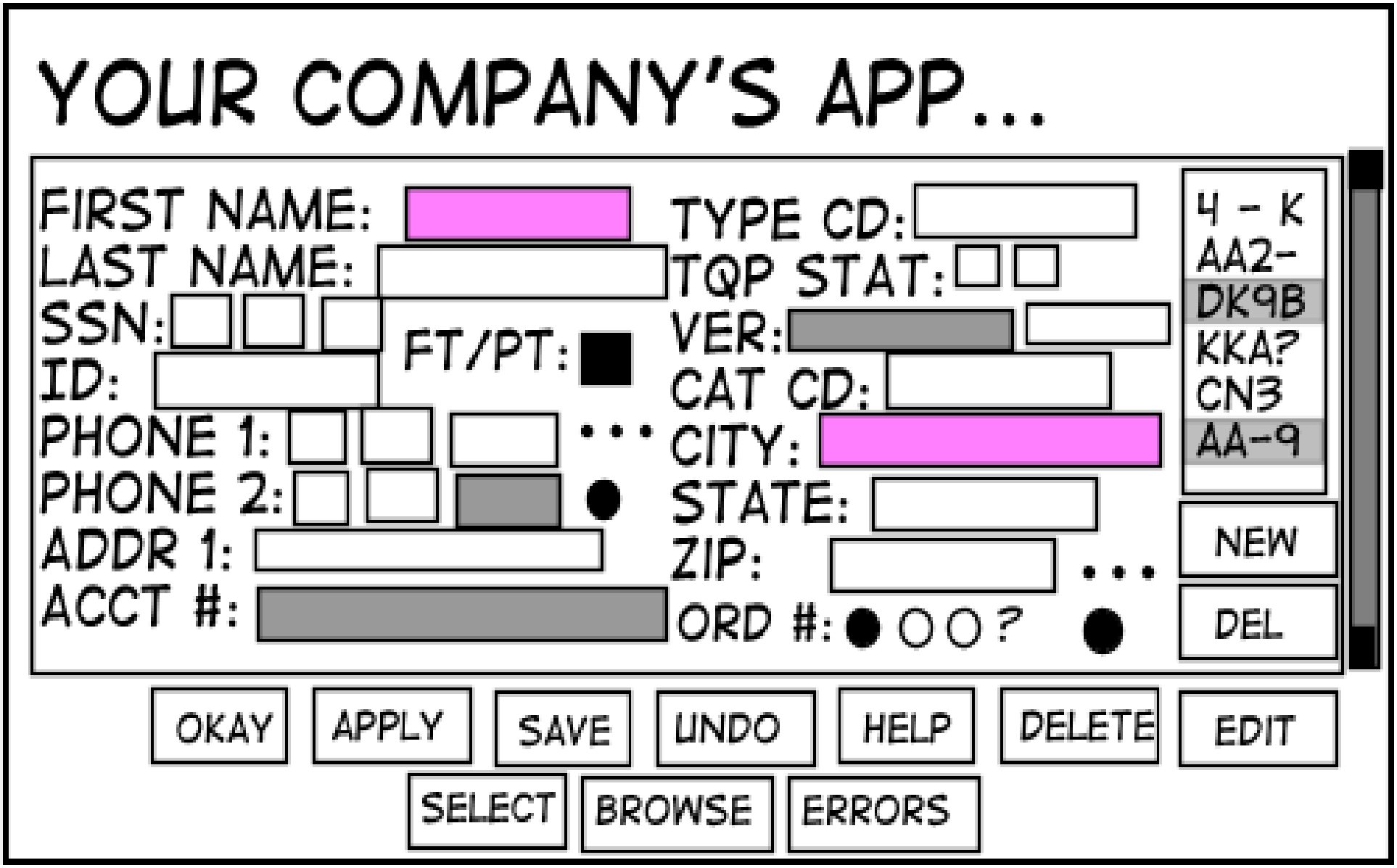
# Back-End

- Transaktionale Dienste
- Geschäftsprozesse
- Persistente Datenhaltung
- Ungefähr so sexy wie ein Serverschrank
- Über die Jahre haben sich große Mengen Daten und Code angesammelt

# Front-End

- Meist komplexe Masken mit vielen Eingabefeldern
- Über die Jahre gewachsene unübersichtliche Menüstrukturen
- „Sowjet-Charme“: in die Jahre gekommenes Aussehen der Eingabemasken
- Darstellung in aktuellen Browsern wird irgendwann zum Problem






STUFFTHATHAPPENS.COM BY ERIC BURKE

**we spend 90% of the TCO of an application post launch**

source: Gartner analyst (via James Lewis / Thoughtworks)

# Neue Oberfläche?




# Nur welche?



APACHE WICKET

# Das beste Webframework?

Unmöglich das seriös zu beurteilen. Deshalb eine einfache Theorie:

**Wer beim Entwickeln Spaß hat, arbeitet konzentrierter, schneller und besser**

→ Ziel: Spaßbremsen eliminieren

# Spaßbremsen 1: Komponentenbasierte Frameworks

Wollen HTTP, HTML und AJAX vor dem Entwickler verstecken, sind dabei nicht besonders erfolgreich.


**Ziel:**
Entwickler, die aus dem Desktop-GUI bzw. Client-Server Zeitalter kommen, müssen sich nicht umstellen

APACHE WICKET

**Tatsächliches Ergebnis:**
Träge laufende Anwendungen, schwer zu ändern

vaadin }>

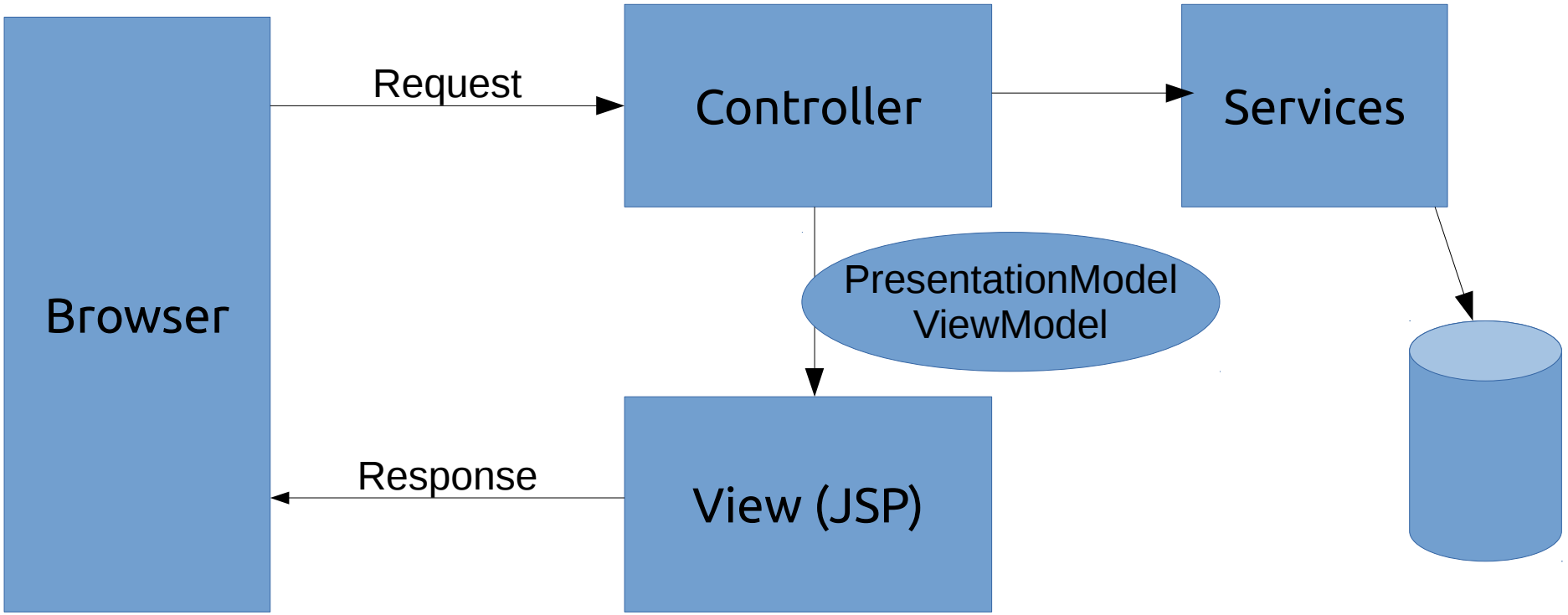
# Spaßbremsen 2: Template-basierte Frameworks


- Waren mal richtig gut (JSP 2 / JSTL !)
  - Model-2 Architecture → Separation of Concern
  - Schnelle Feedback-Zyklen

Struts

- Fühlt sich mit AJAX nicht mehr richtig an (Paradigmenbruch)

# Model-2 Architektur



# Spaßbremsen 3: Java-Serverapplikationen

**„Zu langsame Feedbackzyklen!“**


- Application Server Start dauert oft mehrere Minuten
- Bei Änderungen meist Redeploy der Anwendung nötig → wieder 30 Sekunden
- Nach ein paar Neustarts: OutOfMemoryError PermGen Space

## Themen

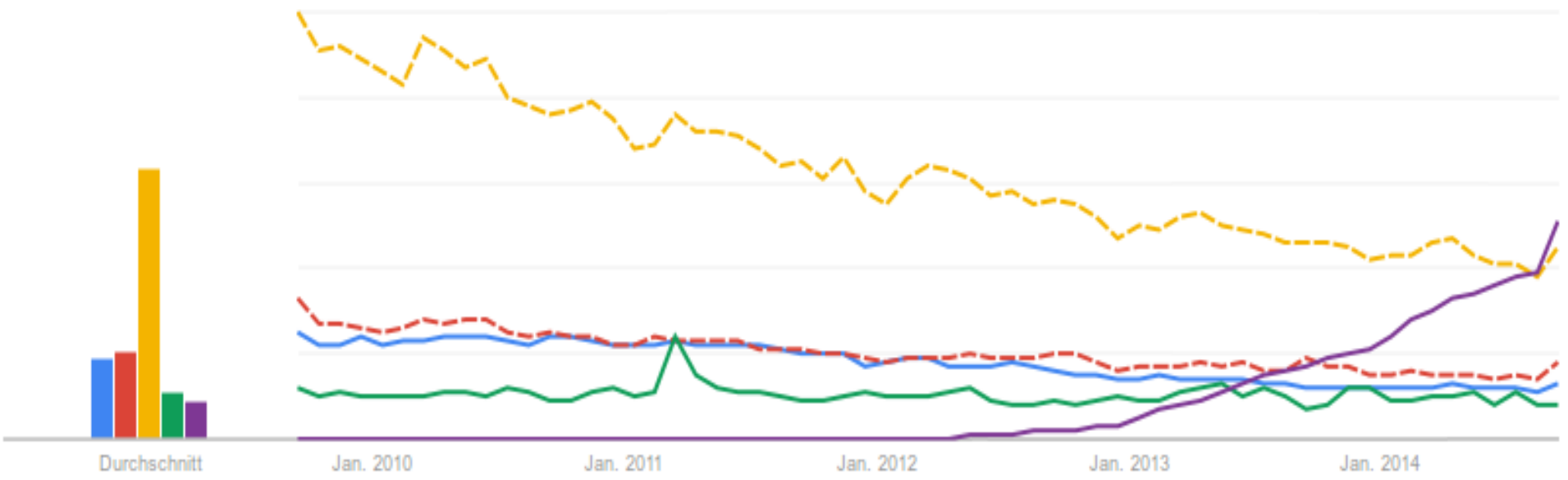
Abonnieren

| gwt | JavaSer... | JavaSer... | wicket | angularjs |
|---|---|---|---|---|
| Suchbegriff | Web Framework | File Format | Suchbegriff | Suchbegriff |

Beta: Mit dieser Betafunktion können Sie das Interesse an einem *Suchthema* abrufen und Sie erhalten schnell genaue Angaben zum gesamten Suchinteresse. Um das Interesse an einer bestimmten *Suchanfrage* abzurufen, wählen Sie die Option "Suchbegriff" aus.

## Interesse im zeitlichen Verlauf

Nachrichtenschlagzeilen  Prognose

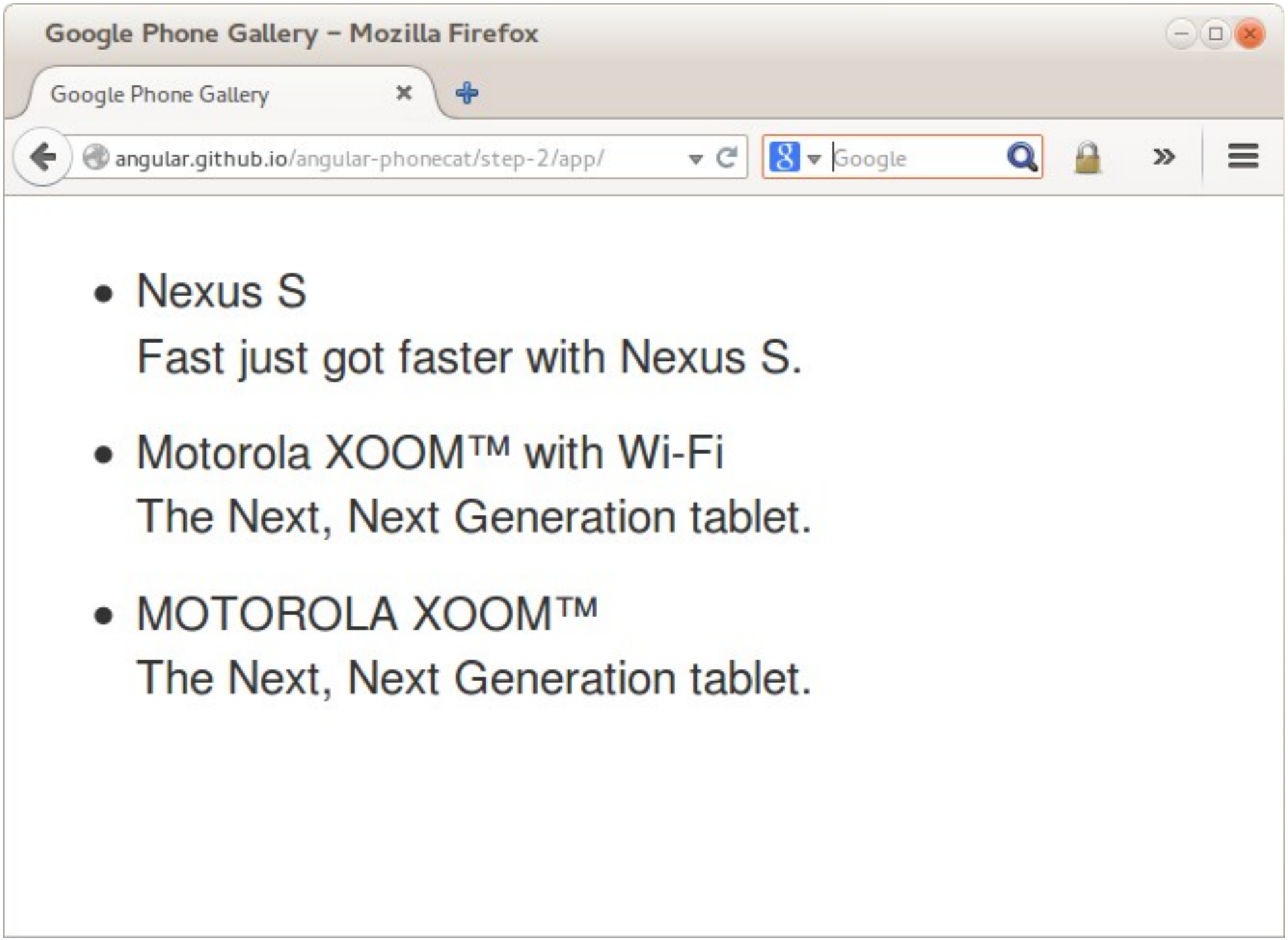


# Anforderungen an ein modernes Web-Framework

- Separation of Concern
  - Förderung sauberer Codestruktur
  - Testbarkeit
  - Änderbarkeit
- Direkte Arbeit mit HTTP, HTML, CSS, etc.
- Schnelles Feedback ohne langwierige Neustarts
- AJAX ohne Paradigmenbruch integrieren
- Ich möchte nicht der einzige User sein

# Features

- Single Page Application
  - Applikation, nicht „Seite“
- Separation of Concern
  - Dependency Injection
  - MVW (Model, View, Whatever)
  - Unit-Test-Framework inklusive
- AJAX ist der einzige Weg Daten vom Server zu holen
  - Kein Paradigmenbruch mehr
- Flüssiges Entwickeln
  - Sofort visuelles Feedback
  - Kein langwieriges Neustarten oder Deployen


Google Phone Gallery – Mozilla Firefox
Google Phone Gallery
angular.github.io/angular-phonecat/step-2/app/
Google
Nexus S
Fast just got faster with Nexus S.
Motorola XOOM™ with Wi-Fi
The Next, Next Generation tablet.
MOTOROLA XOOM™
The Next, Next Generation tablet.

```
<html ng-app="phonecatApp">
<head>
  ...
  <script src="bower_components/angular/angular.js"></script>
  <script src="js/controllers.js"></script>
</head>
<body ng-controller="PhoneListCtrl">

  <ul>
    <li ng-repeat="phone in phones">
      {{phone.name}}
      <p>{{phone.snippet}}</p>
    </li>
  </ul>

</body>
</html>
```

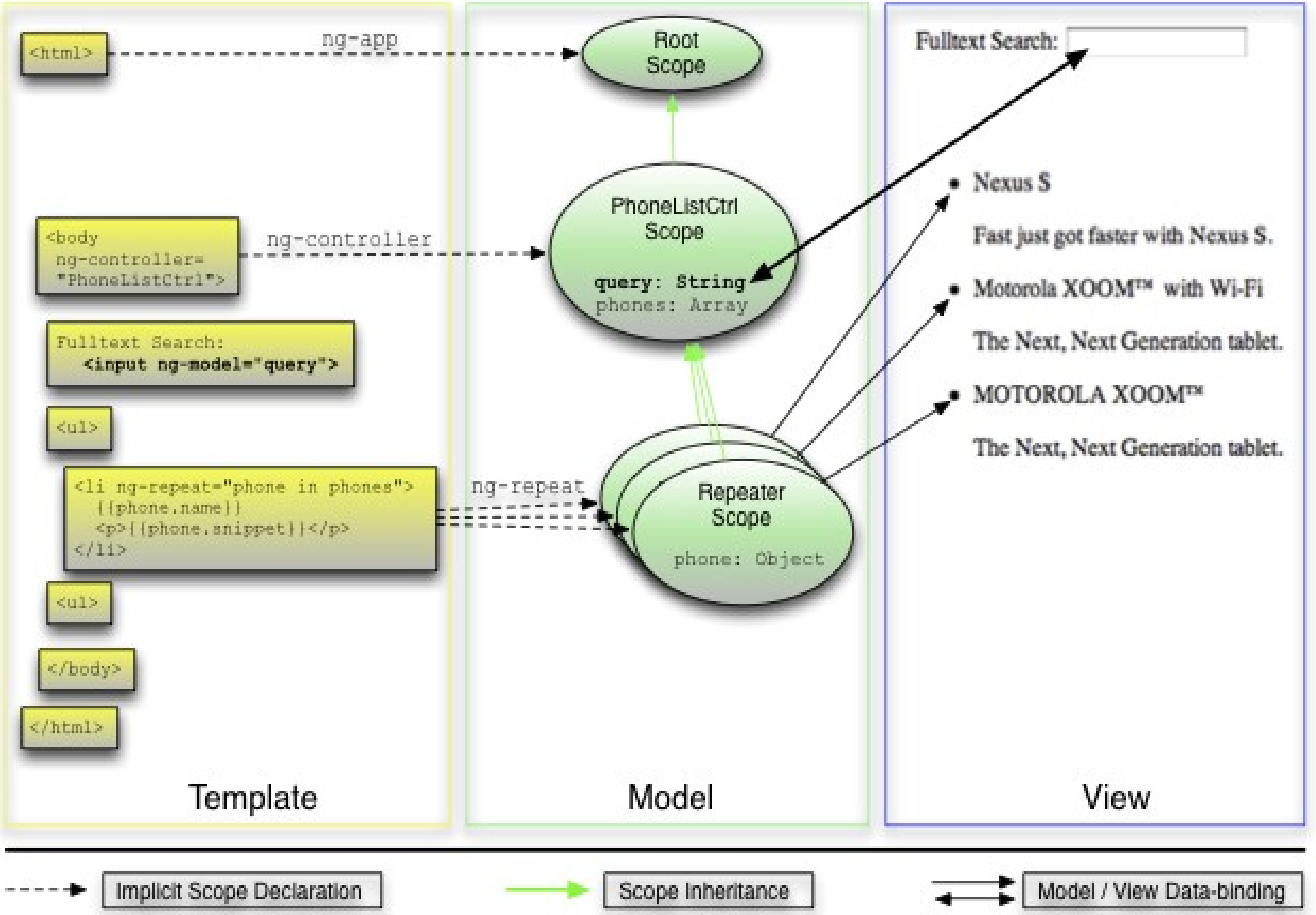
# Direktiven

- HTML-Elemente oder Attribute mit spezieller Funktionalität anreichern
- Möglichkeit, Low Level DOM-Manipulation wiederverwendbar auszulagern
- Es gibt viele vordefinierte Direktiven
- Third-Party-Bibliotheken definieren neue Direktiven (bspw. ui-bootstrap)
- Man kann auch eigene schreiben
- Direktiven haben eigenen Scope und eigenen Controller

```
angular.module('phonecatApp')
.controller('PhoneListCtrl', function ($scope) {

  $scope.phones = [
    {'name': 'Nexus S',
     'snippet': 'Fast just got faster with Nexus S.'},
    {'name': 'Motorola XOOM™ with Wi-Fi',
     'snippet': 'The Next, Next Generation tablet.'},
    {'name': 'MOTOROLA XOOM™',
     'snippet': 'The Next, Next Generation tablet.'}
  ];

});
```


<html>
ng-app
Root Scope
Fulltext Search:
<body ng-controller="PhoneListCtrl">
ng-controller
PhoneListCtrl Scope
query: String
phones: Array
Nexus S
Fast just got faster with Nexus S.
Motorola XOOM™ with Wi-Fi
The Next, Next Generation tablet.
MOTOROLA XOOM™
The Next, Next Generation tablet.
Fulltext Search:
<input ng-model="query">
<ul>
<li ng-repeat="phone in phones">
{{phone.name}}
{{phone.snippet}}
</li>
ng-repeat
Repeater Scope
phone: Object
<ul>
</body>
</html>
Template
Model
View
Implicit Scope Declaration
Scope Inheritance
Model / View Data-binding

```
describe('PhoneListCtrl', function(){

  beforeEach(module('phonecatApp'));

  it('should create "phones" model with 3 phones',
     inject(function($controller) {
    var scope = {},
    ctrl = $controller('PhoneListCtrl', {$scope:scope});

    expect(scope.phones.length).toBe(3);
  }));

});
```

```
angular.module('phonecatApp')
.controller('PhoneListCtrl', function ($scope, $http) {

  $http.get('phones/phones.json')
  .success(function(data) {
    $scope.phones = data;
  });

  $scope.orderProp = 'age';

});
```

```
[
 {
  "age": 13,
  "id": "motorola-defy-with-motoblur",
  "name": "Motorola DEFY\u2122 with MOTOBLUR\u2122",
  "snippet": "Are you ready for everything life
              throws your way?"
  ...
 },
...
]
```

```
describe('PhoneCat controllers', function() {
  describe('PhoneListCtrl', function(){
    var scope, ctrl, $httpBackend;

    beforeEach(module('phonecatApp'));

    beforeEach(inject(function(_$httpBackend_, $rootScope, $controller) {
      $httpBackend = _$httpBackend_;
      $httpBackend.expectGET('phones/phones.json').
          respond([{name: 'Nexus S'}, {name: 'Motorola DROID'}]);
      scope = $rootScope.$new();
      ctrl = $controller('PhoneListCtrl', {$scope: scope});
    }));


    it('should create "phones" model with 2 phones fetched from xhr',
      function() {
        expect(scope.phones).toBeUndefined();
        $httpBackend.flush();

        expect(scope.phones).toEqual([{name: 'Nexus S'},
                                      {name: 'Motorola DROID'}]);
      });
```

# Historie

- 2009 gestartet von Miško Hevery and Adam Abrons, die damit eine Geschäftsidee umsetzen wollten
- Seit 2010 Open Source
- Version 1.0 released 2012
- 2012 Launch der neuen Doubleclick-Oberfläche auf AngularJS-Basis
- 2014 Release des Protractor E2E Framework
- Angular wird heute von einem 15-köpfigen Team bei Google weiterentwickelt
- Jährliche Konferenz ng-conf

# Komplexe Toolwelt

# NodeJS / NPM

- Kommandozeilen-Ausführungsumgebung für Javascript
- NPM lädt dependencies für die Entwicklung (Grunt, Karma, Yeoman, etc.)
- Bower lädt Dependencies der Webanwendung (Bootstrap UI, Compass, Schriftarten, ...)

- Zur Entwicklungszeit
  - **Livereload**: Geänderte Dateien erkennen und „Teil-Build“ anstoßen
  - **JSHint**: Statische Codeanalyse
  - **Connect**: Ein Webserver, der geänderte Dateien über Websocket zum Browser pusht
  - **Connect Proxy**: Backend-Anfragen an Java App-Server weitergeben


- Beim „richtigen“ Bauen:
  - **Karma**: Tests im Browser ausführen
  - **Filerev**: Dateinamen um einen Hash ergänzen
  - **Usemin/Uglify**: Konkatenieren, minifizieren (JS, HTML, Bilder)

# Build Best Practices

Gruntfile und Verzeichnisstruktur
durch Yeoman generieren lassen!

# Build Best Practices

## Maven-Build mit dem Grunt-Build integrieren?

1. Fehlversuch: grunt-maven-plugin

- Irrsinnig kompliziert zu konfigurieren
- Yeoman kann nicht mehr genutzt werden

2. Fehlversuch: maven-antrun-plugin

- Der Grunt Build läuft immer doppelt innerhalb des Maven Builds
- npm install muss jedes Mal neu ausgeführt werden → lange Downloadzeiten

Akzeptable Lösung: Front-End getrennt vom Back-End bauen und Unit Testen, erst danach zusammenbringen

- z.B. Apache Mod-Proxy oder NodeJS-Proxy
- Laufzeitumgebung möglichst automatisch provisionieren!

# Best Practices

- Zusätzliche UI-Bibliothek einbinden
  - Sehr gute Erfahrungen mit **ui-bootstrap**
  - Bietet Bootstrap-Komponenten als Angular-Direktiven
  - Verwendet bootstrap CSS, aber der Javascript-Teil wurde speziell für Angular neu geschrieben

# Best Practices

- Compass / SCSS für Stylesheets verwenden
  - Vererbung und Modularisierung von Stylesheets
  - Große Bibliothek mit vielen gute Vorlagen, die man nur noch erweitern muss
  - Buildprozess lässt sich sehr einfach in Grunt integrieren

# End to End Testing

Problem: Tests fallen bei jeder noch so kleinen Änderung um, Entwickler sind genervt

- Lösungsansatz: Protractor bringt mehr Stabilität durch zusätzliche Angular-spezifische Selektoren, die man hoffentlich seltener anpassen muss

# Integration Alte GUI – neue GUI

- Doppelten Login vermeiden
  - JSessionID-Cookie als Bindeglied zwischen JSF und REST Services
  - Oder: Login-Seite der Altanwendung generiert das Access-Token für REST Services mit
- Angular Views per Frame / IFrame in die Menüstruktur der Altanwendung einbringen
- Oder umgekehrt

# Wie motiviere Entwickler, Javascript und HTML5 zu lernen?

- Entwickler wollen lernen!
- 3-tägige Schulung und dann die Leute damit allein lassen → riskant
- Besser: Zeitbegrenzte Unterstützung durch erfahrenen Externen, um „Anfängerfehler“ zu vermeiden
- Teams richtig zusammensetzen: Es darf kein „Loser-Team“ entstehen, das sich abgehängt fühlt
- Keine „AngularJS-Experten“ heranzüchten
- Kultur der gegenseitigen Hilfe im Team etablieren

# Nachteile

- NPM lädt die ganze Welt herunter
  - Es gibt zwar lokales Caching der heruntergeladenen Releases, aber ...
  - Transitive Dependencies werden mehrfach heruntergeladen, erst danach durch „dedupe“ zusammengefasst
  - NPM installiert teilweise native libraries, die durch den C-Compiler müssen → dabei kann viel schiefgehen
  - Bricht oft ab mit wildesten Fehlermeldungen
- Das Gruntfile ist ein Monster
  - Hohe Komplexität im Buildscript
  - Zum Glück wird es generiert, und wir brauchen es nicht so oft zu ändern

# Vielen Dank!


Bastian Voigt

Softwareentwicklung und Projektmanagement

http://bastian-voigt.de

@BastianVoigt

post@bastian-voigt.de

Tel. 0179-4826359