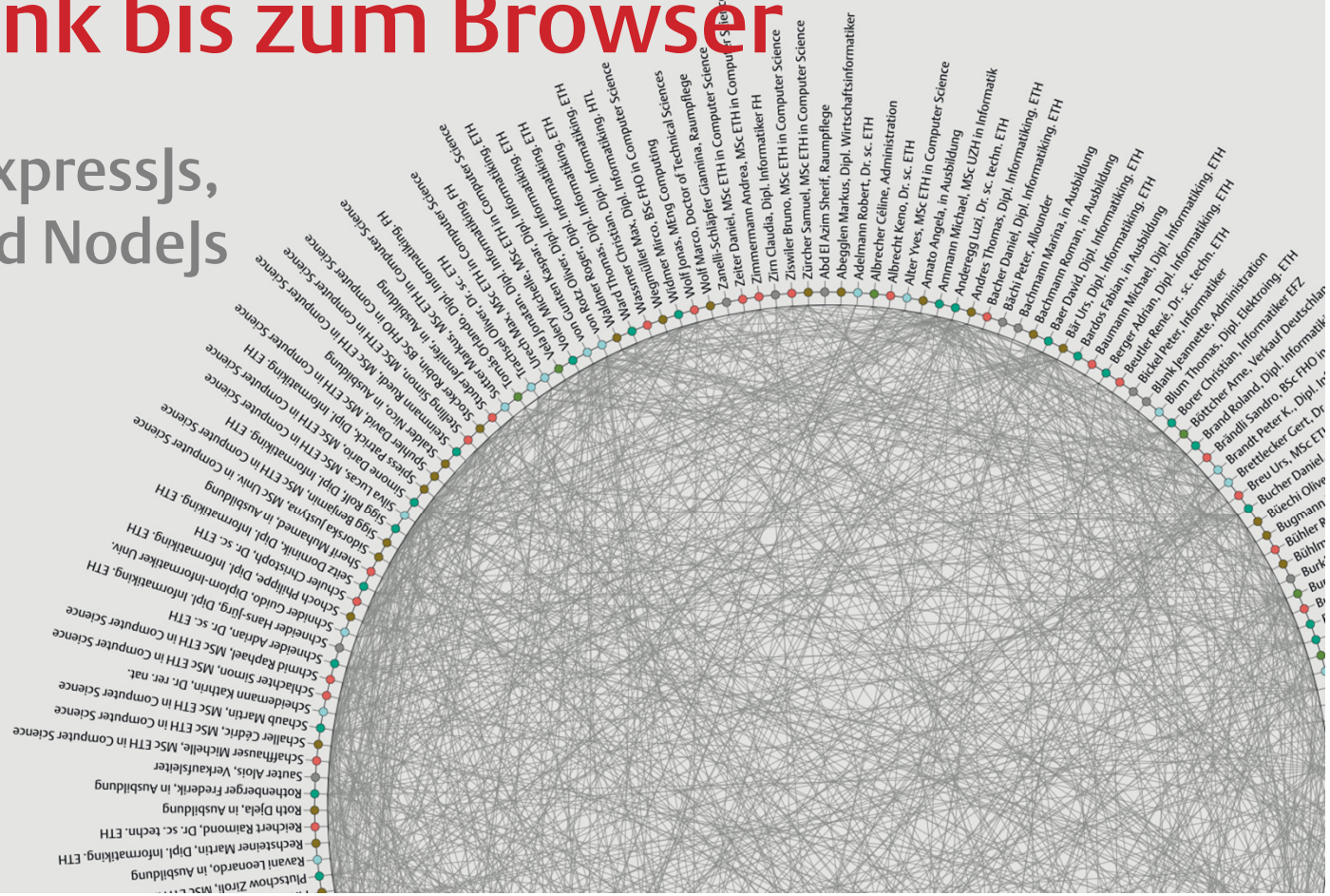
JavaScript von der Datenbank bis zum Browser

MongoDb, ExpressJs, AngularJs und NodeJs im Überblick

ergon



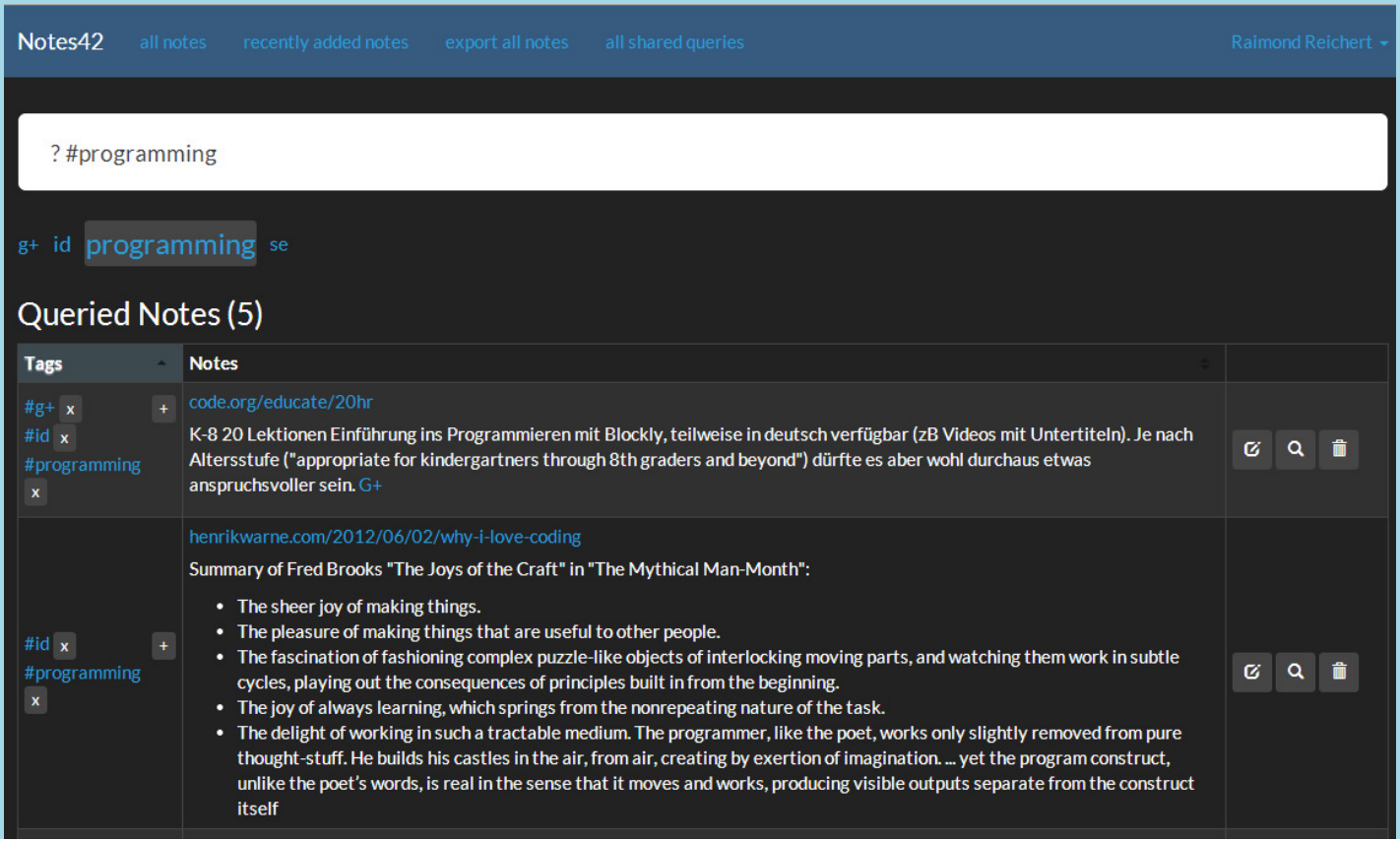
Zusammenfassung

Die Dynamik und Innovationsrate der JavaScript-Welt ist enorm: Grund genug für uns, einen Einstieg zu wagen. Wir zeigen in diesem Vortrag, wie wir als erfahrene Java-Entwickler erste Schritte mit den MEAN-Technologien machten: MongoDb, ExpressJs, AngularJs und NodeJs. Der Vortrag bietet einen Überblick über die Architektur einer MEAN-Applikation sowie über die eingesetzten Frameworks und Tools. Zudem zeigen wir auf, wie wir unsere Einarbeitung in die JavaScript-Welt als Mini-Projekt vorbereitet und durchgeführt haben. Das Resultat findet sich unter notes42.herokuapp.com, ein Prototyp einer "geeky" Web-App für Links und Notizen.

Das Ziel
Mini-Projekt basierend auf MEAN Stack

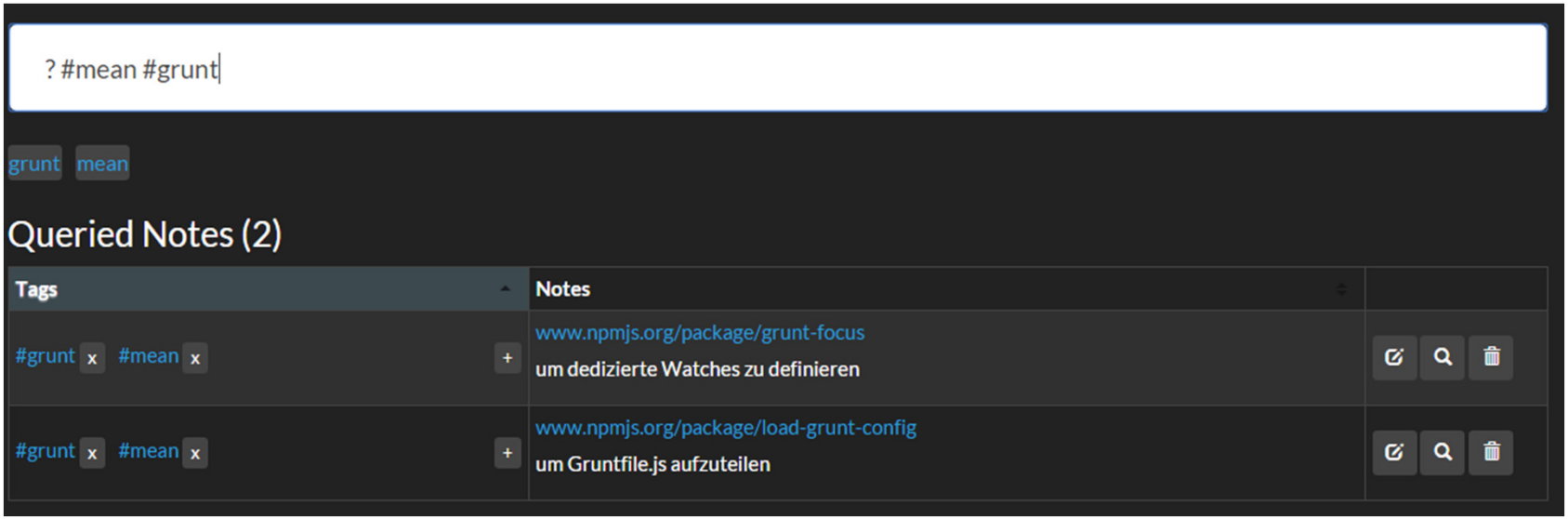
ergon

Die Idee
«geeky» Web-App für Links, Notizen

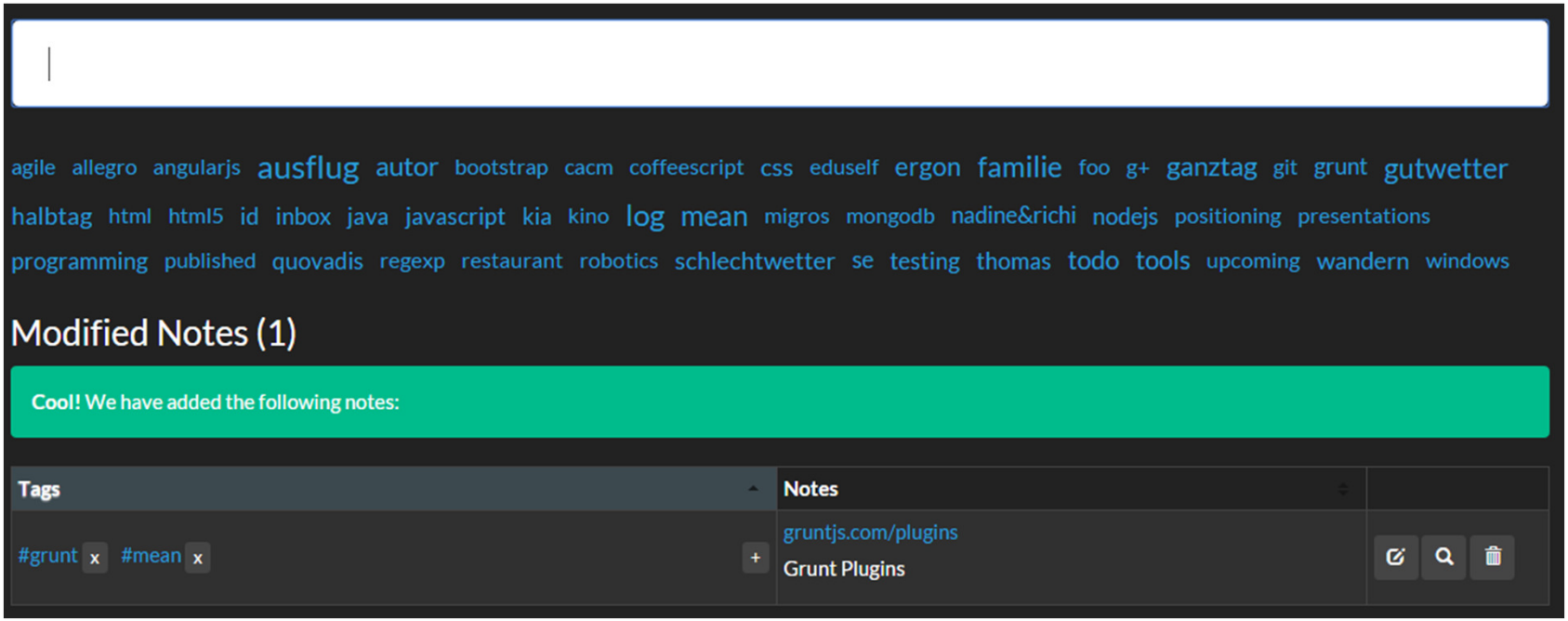


ergon

Einfache Query: Notes zu Tag-Kombination
? #mean #grunt

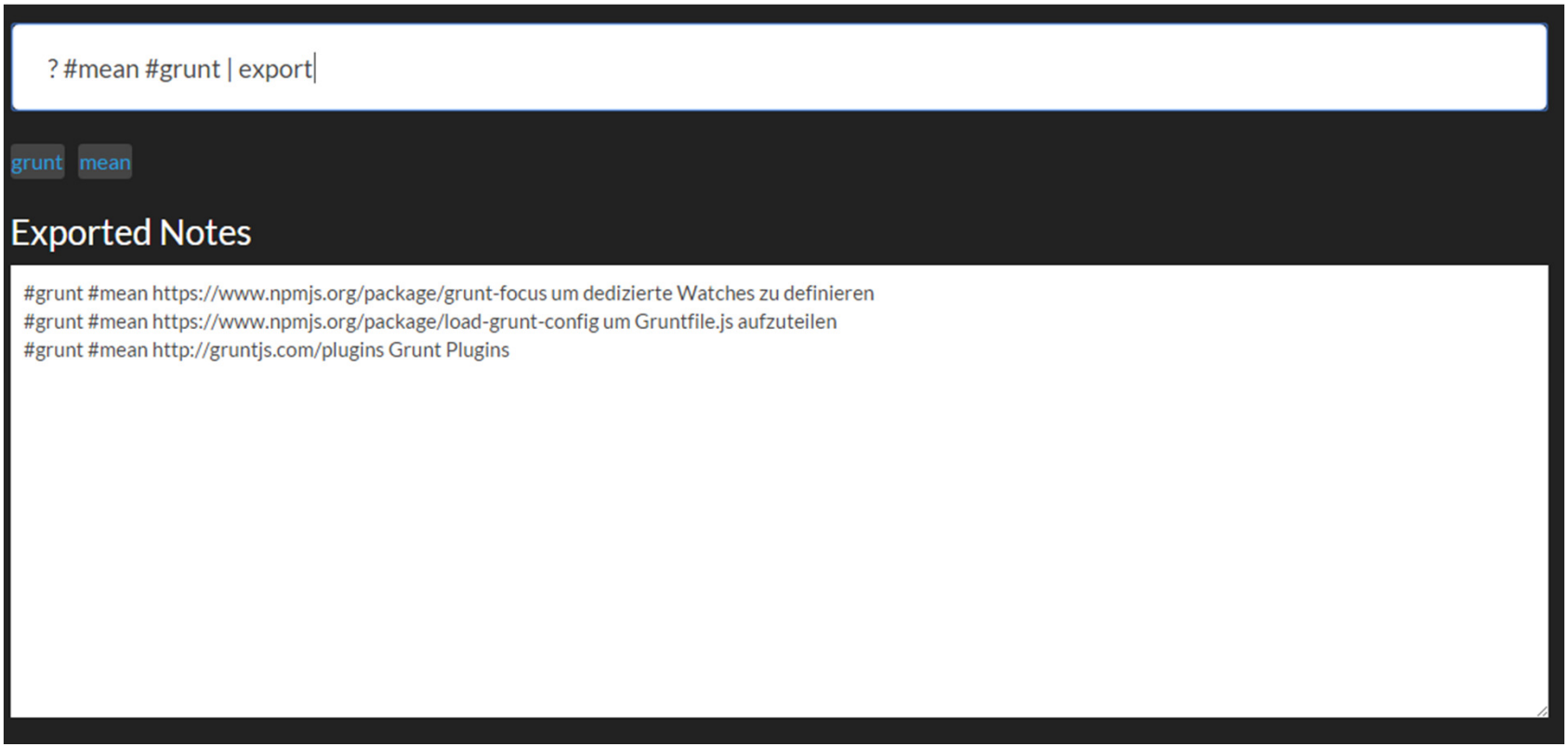


Modify: Note hinzufügen
#mean #grunt http://gruntjs.com/plugins Grunt Plugins

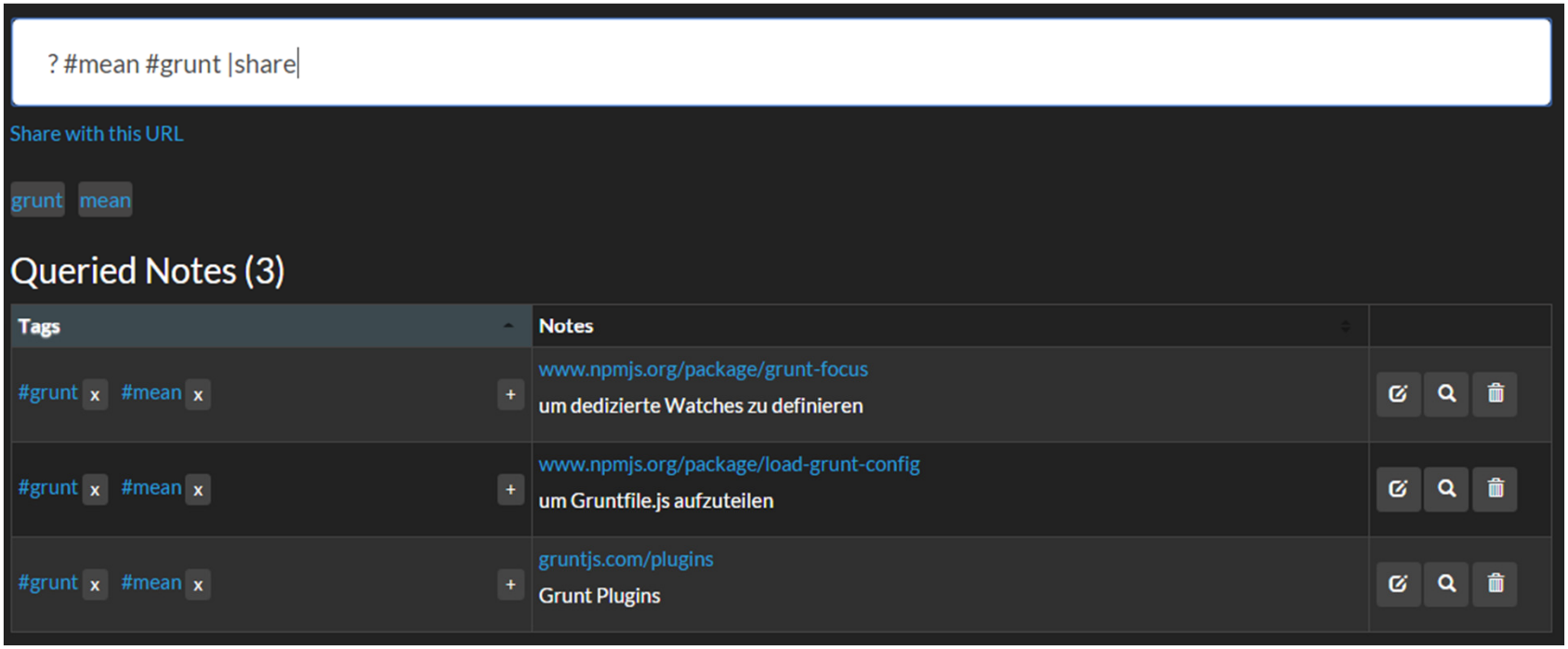


ergon

Piped Command «Export»: Notes exportieren
? #mean #grunt | export



Piped Command «Share»: Notes veröffentlichen
? #mean #grunt |share



Piped Command «Share»: Veröffentlichte Notes

notes42.herokuapp.com/#!/shared/4c54d060-34cc-4720-9424-80c24c15121d

Raimond Reichert shared 3 notes with you

Tags	Notes
#grunt #mean	www.npmjs.org/package/grunt-focus um dedizierte Watches zu definieren
#grunt #mean	www.npmjs.org/package/load-grunt-config um Gruntfile.js aufzuteilen
#grunt #mean	gruntjs.com/plugins Grunt Plugins

Weitere Query Operatoren und Kombination von Queries und Piped Commands

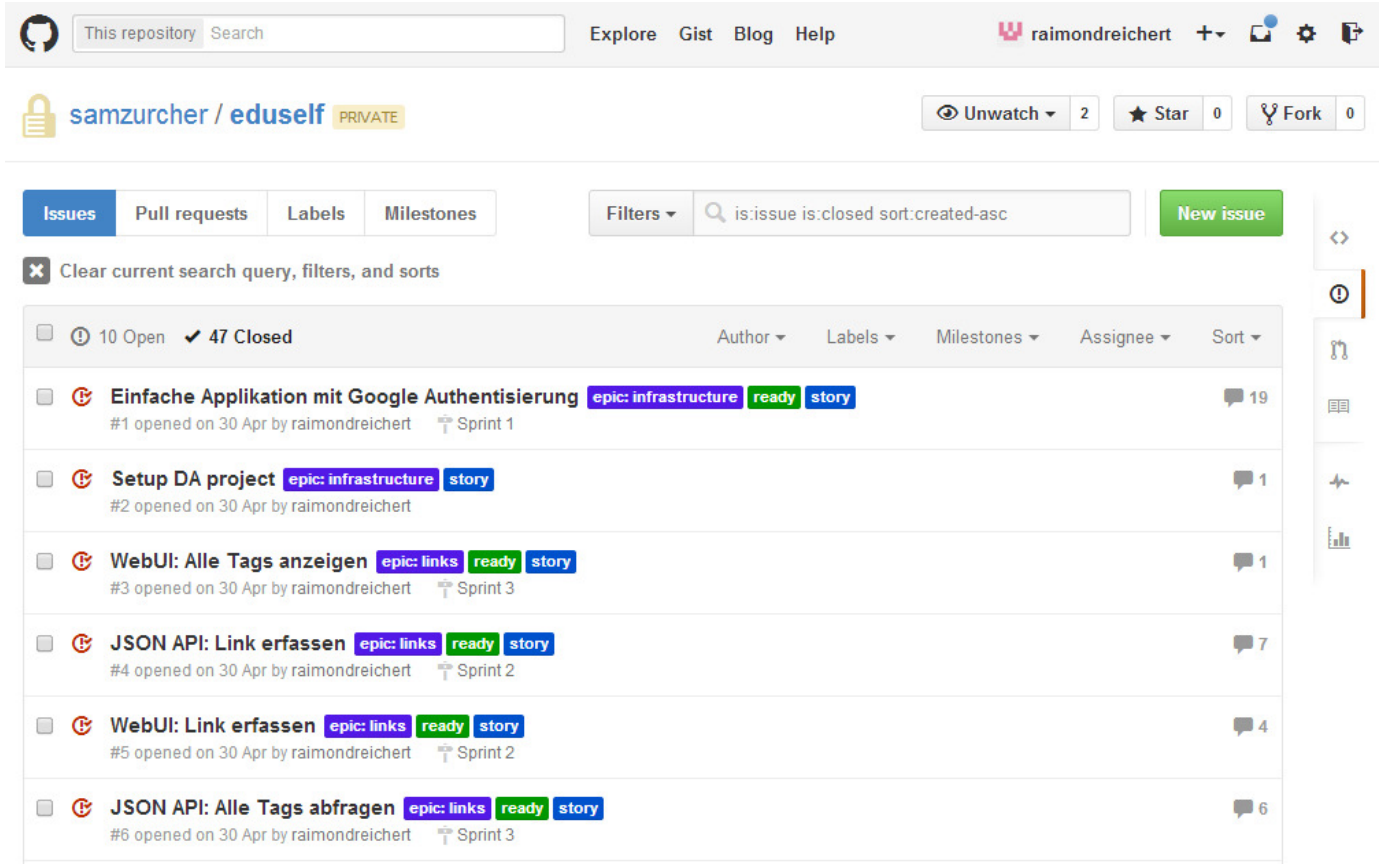
Notes mit Tag «css» ausschliessen	? #mean -#css
Notes, die in den letzten 7h erstellt oder verändert wurden	? :7h
Sortieren nach zum Beispiel URL	? | sort by url asc
Sortiert exportieren	? | sort by created desc | export
Sortiert veröffentlichen	? | sort by urlVisitedCount desc | share

Die Vorbereitung
Planung

April	Ideen prüfen, Stories schreiben
Mai	Lesen, lesen, lesen
Juni	Setup von VM sowie Entwicklungs- und Produktionsumgebung
Juli	10 Tage Projekt
August	Sporadische Weiterentwicklung

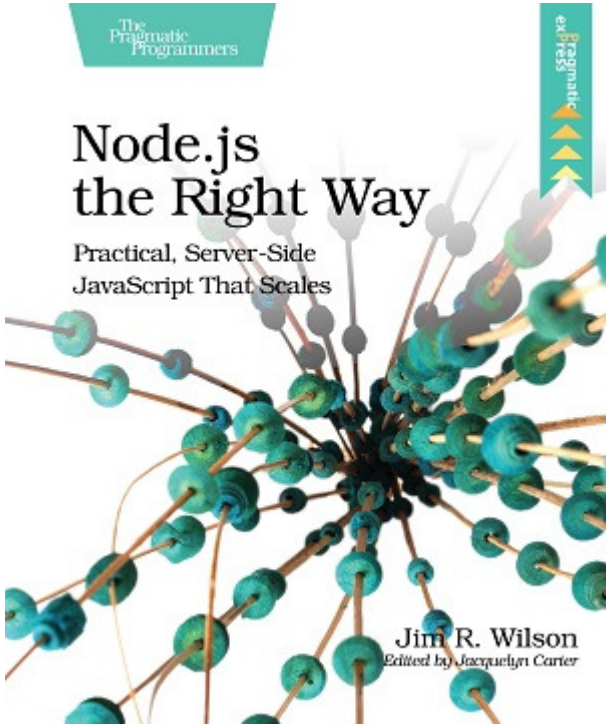
ergon

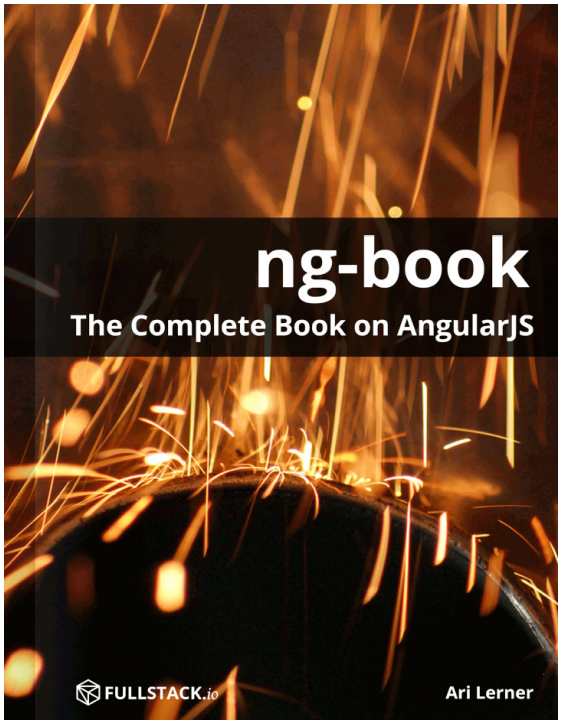
Die Vorbereitung
Stories für «minimal viable product» auf github.com



ergon

Die Vorbereitung
angularjs.org, nodejs.org, ...





ergon

Die Vorbereitung
Scripted Ubuntu VM Setup

ubuntu

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
echo 'deb http://downloads-distro.mongodb.org/repo/debian-sysvinit dist 10gen' |
sudo tee /etc/apt/sources.list.d/mongodb.list
sudo apt-get update
sudo apt-get install mongodb-10gen

sudo apt-get update
sudo apt-get -y install python-software-properties python g++ make
sudo add-apt-repository ppa:chris-lea/node.js
sudo apt-get update
sudo apt-get -y install nodejs
sudo npm install -g grunt-cli bower

# ...
```

ergon

Die Vorbereitung
MEAN-Projekt-Scaffolding mit mean.io

```
> sudo npm install -g meanio@latest
> mean init notes42
> cd notes42 && npm install
> grunt
```

... und schon hat man eine einfache CRUD-Applikation inklusive Authentifizierung.

Die Vorbereitung
Infrastruktur-Setup abgeschlossen

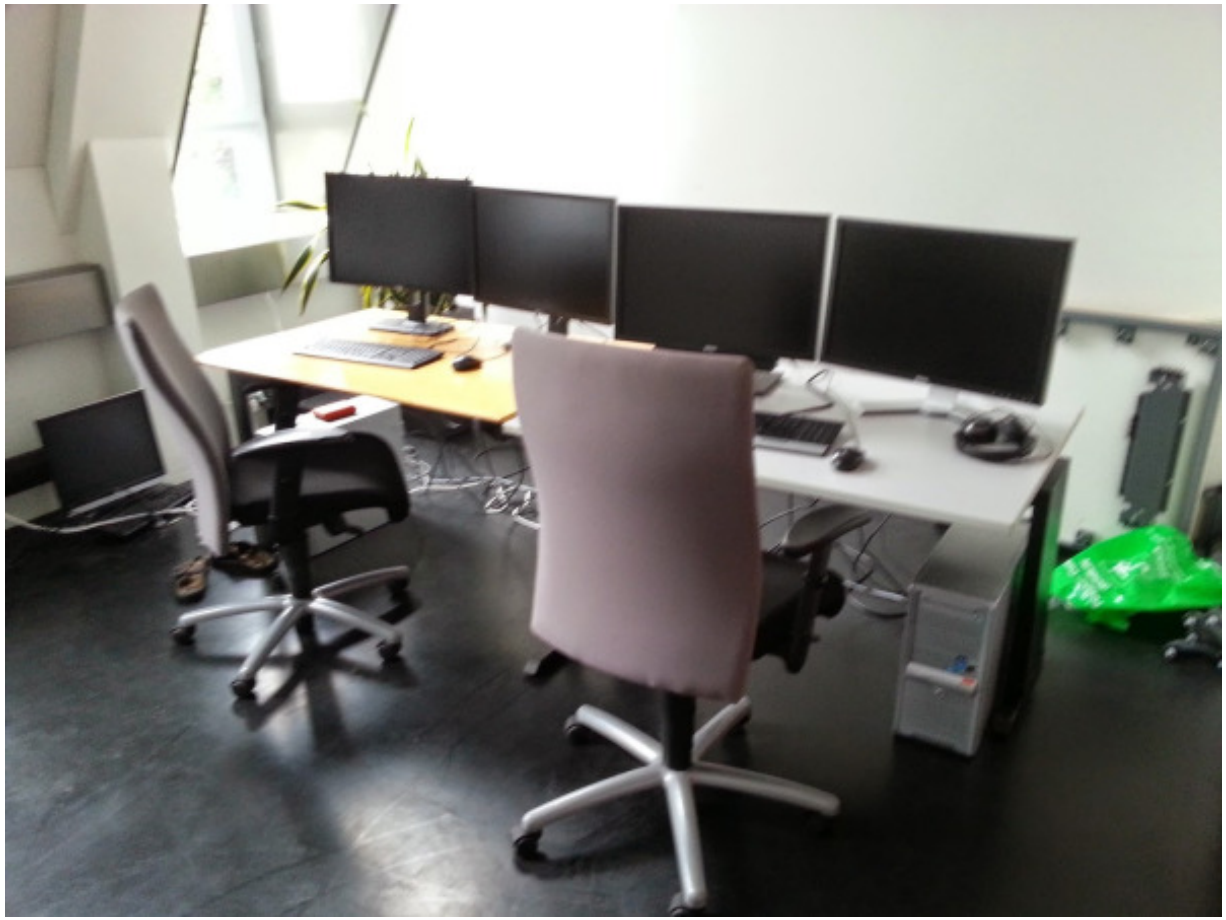
- ✓ «Setup VM und Entwicklungsumgebung»
- ✓ «Setup Projekt»
- ✓ «Einfache Applikation mit Google Authentisierung»
- ✓ «Setup Produktionsumgebung»

Die Umsetzung
5 Sprints à 2 Tage

April	Ideen prüfen, Stories schreiben
Mai	Lesen, lesen, lesen
Juni	VM Setup, Development and Production Environment Setup
Juli	**10 Tage Projekt**
August	Sporadische Weiterentwicklung

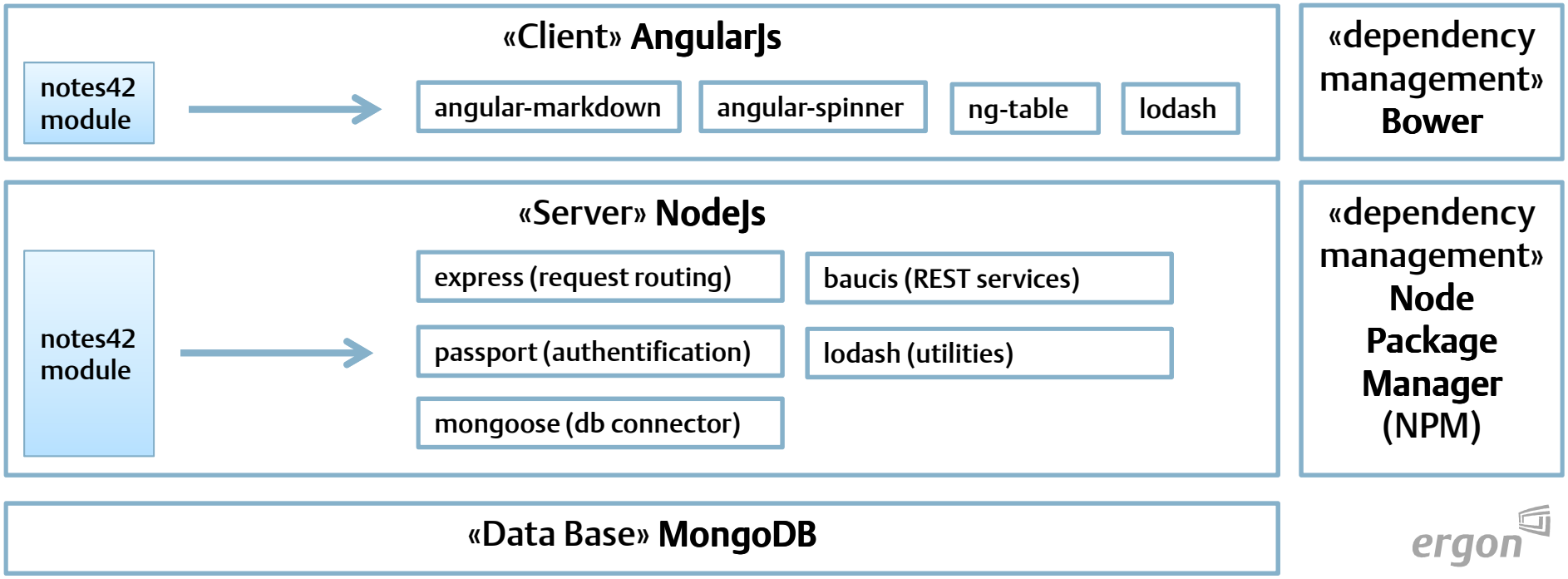
ergon

Der Arbeitsplatz
Setup für «Extreme pair programming»



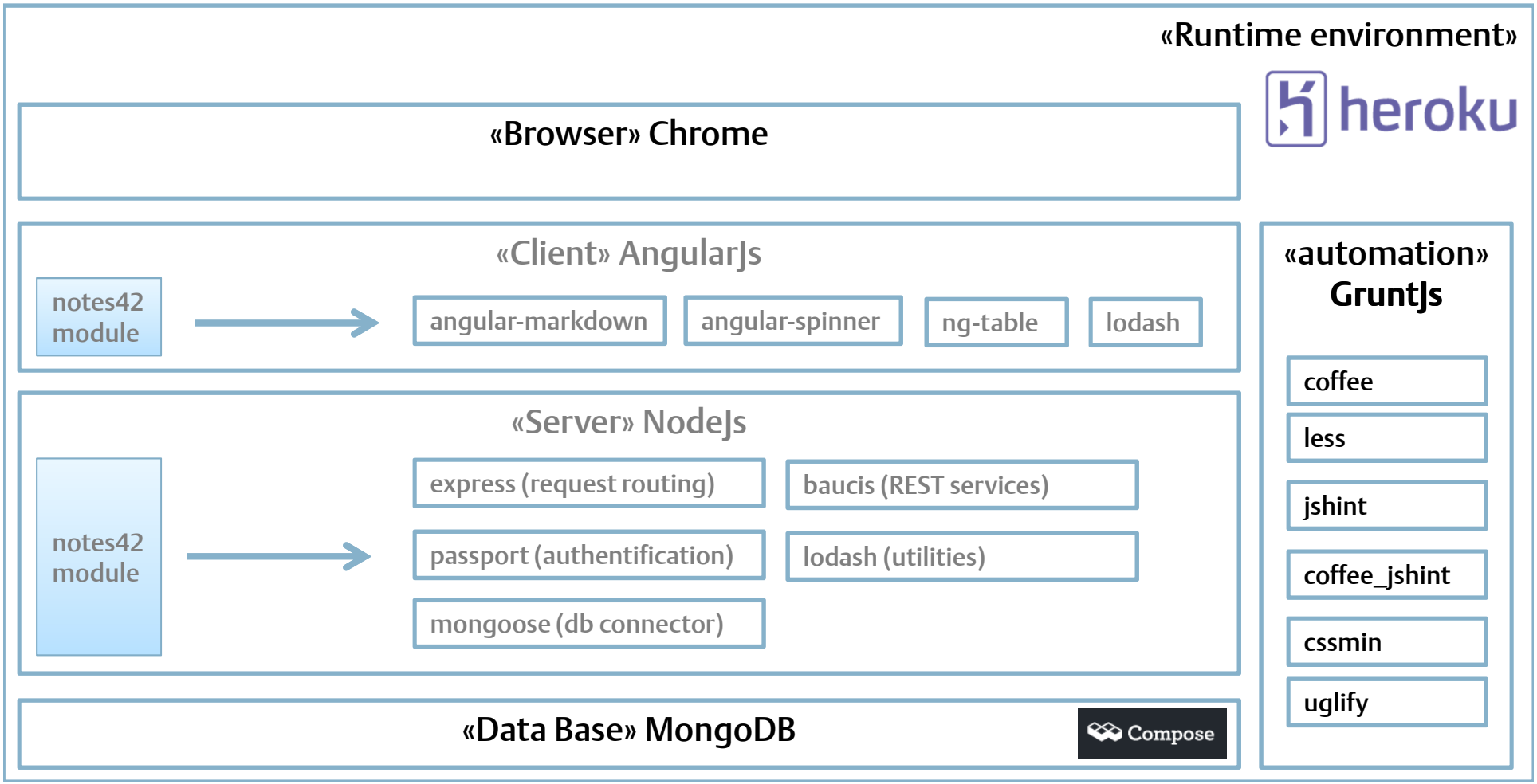
ergon

Komponenten der Applikation
Kleine, fokussierte Plugins für so ziemlich alles

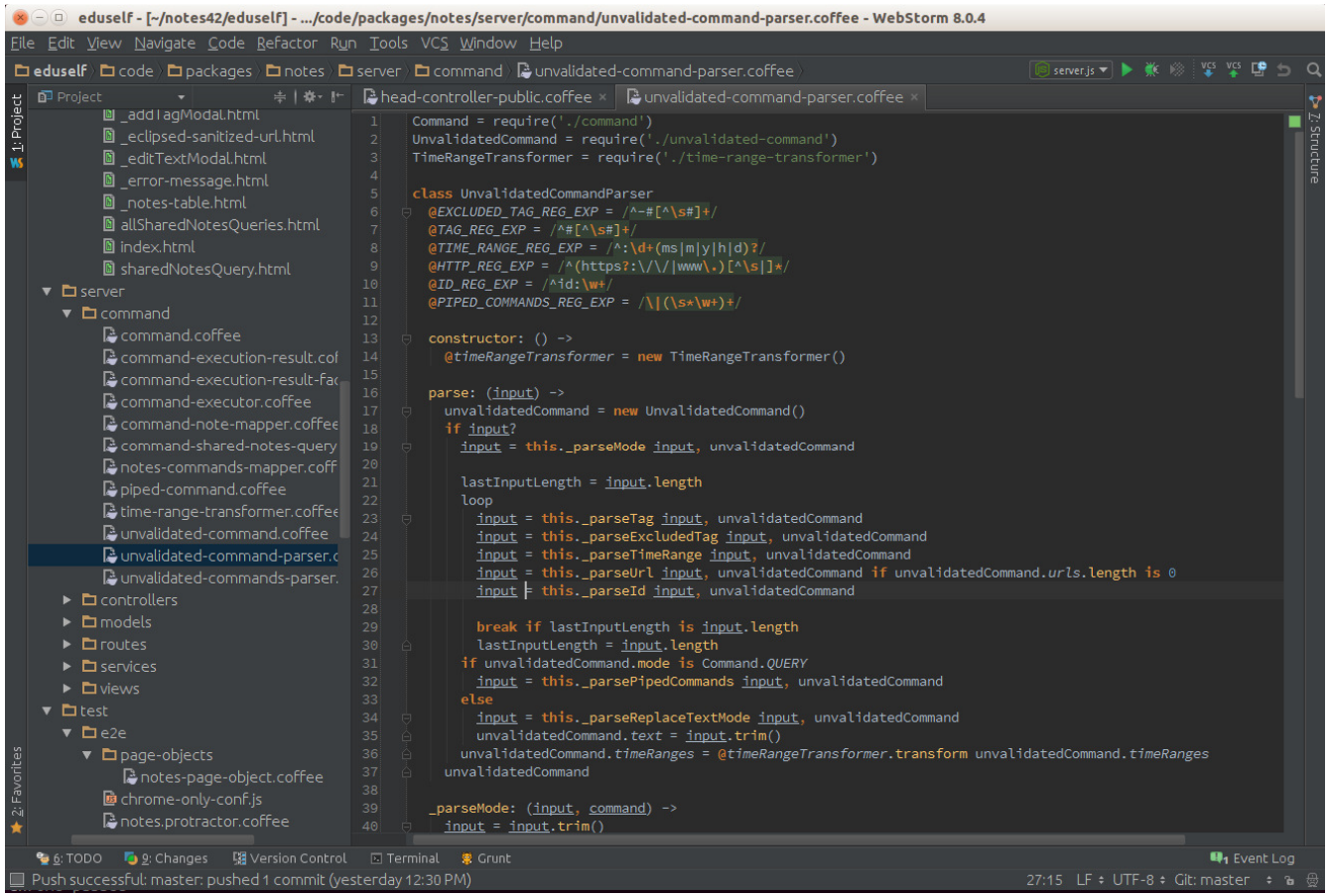


ergon

Produktionsumgebung: «git push heroku master»



Development
Webstorm IDE – empfehlenswert!



ergon

Development
GruntJs: CoffeeScript und LESS «on save» übersetzen

«Zero Turnaround» Live Reload

«Browser» Chrome

«Debugging» Batarang

«Client» AngularJs

notes42 module

angular-markdown

angular-spinner

ng-table

lodash

«Server» NodeJs

notes42 module

express (request routing)

baucis (REST services)

passport (authentification)

lodash (utilities)

mongoose (db connector)

«Data Base» MongoDB

«automation» GruntJs

coffee

less

jshint

coffee_jshint

Development
GruntJs: Unit Tests «on save» ausführen

«Browser» Chrome

«Debugging» Batarang

«Client» AngularJs

notes42 module

angular-markdown | angular-spinner | ng-table | lodash

«Server» NodeJs

notes42 module

express (request routing) | baucis (REST services)

passport (authentification) | lodash (utilities)

mongoose (db connector)

«Data Base» MongoDB

«automation» GruntJs

- mocha
- jasmine
- *protractor*

Development
JavaScript: Funktionaler Programmierstil

«functions are first class citizens»
Closures und (haufenweise) Callbacks zentral.
Gewöhnungsbedürftig: Kein Compiler, der vor Scope-Problemen in Closures warnt.

```
// returns list of books with at least 'threshold' copies sold
function bestSellingBooks(threshold) {
  return bookList.filter(
    function (book) { return book.sales >= threshold; }
  );
}
```

Development
JavaScript: Prototypen statt Klassen

```
// Prototyp-Definition
var Car = function Car (make, model, year) { // Konstrukor
  this.make = make;
  this.model = model;
  this.year = year;
}

Car.prototype.toString = function() {  // Instanz-Methode
  return this.make + ' ' + this.model + ' ('  + this.year + ')';
}

console.log(new Car("Eagle", "Talon TSi", 1993).toString());
// prints “Eagle Talon Tsi (1193)”
```

Development
JavaScript: Prototypen statt Klassen

```
// Beliebigen existierenden Objekt-Prototyp erweitern

String.prototype.dasherize = function() {
  return this.replace(/_/g, "-");
};

console.log("no_more_underscores".dasherize());
// prints “no-more-underscores”
```

Development
JavaScript: Duck Typing statt Interfaces

- JavaScript ist dynamisch typisiert – Duck Typing statt Interfaces:
  «When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck.”
- Gerade im Zusammenhang mit Callbacks angenehm, da man nicht überall Ein-Methoden-Interfaces definieren muss.
- Nachteile der dynamischen Typisierung:
  - Objekt-Datenstruktur ist im Wesentlichen ein assoziativer Array. Es gibt kein «Design by Contract» in Form von Klassen oder Interfaces, um Objekte verbindlich zu definieren.
  - Automatisierte Refactorings trotz IDE-Unterstützung schwierig.

JavaScript

Development
JavaScript: «Sicher» entwickeln

Software Engineering «Best Practices» werden zur zwingenden Voraussetzung, um in der dynamisch-typisierten JavaScript-Welt sicher entwickeln zu können.

- Testing: Automatisierte Tests auf allen Ebenen: Unit, Integration, Application – gutes Tooling!
- Design: «Law of Demeter» einhalten: «Objekte interagieren nur mit ihrer unmittelbaren Umgebung»
- Static Analysis: Tools wie JsHint, «a program that flags suspicious usage in programs written in JavaScript»

ergon

Development
CoffeeScript: «A little language that compiles into JS»

	CoffeeScript	JavaScript

Functions	square = (x) -> x * x	square = function(x) {
		  return x * x;
		};

Array comprehension	cubes = (math.cube num for num in list)	cubes = (function() {
		  var _i, _len, _results;
		  _results = [];
		  for (_i = 0, _len = list.length; _i < _len;
		_i++) {
		  num = list[_i];
		  _results.push(math.cube(num));
		  }
		  return _results;
		})();

Development
CoffeeScript: «syntactic sugar» für OOP

```
class Animal
  constructor: (@name) ->

  move: (meters) ->
    alert @name + " moved #{meters}m.“

class Snake extends Animal
  move: ->
    alert "Slithering..."
    super 5

sammy = new Snake "Sammy the Python“
sammy.move()
```

ergon

Development
CoffeeScript: Nachteile

- Build-Infrastruktur wird komplexer, da CoffeeScript kompiliert werden muss.
- JavaScript-Fehlermeldungen zur Laufzeit nicht immer einfach nach CoffeeScript rückzuübersetzen, im Zweifelsfall muss generiertes JavaScript angeschaut werden.
- Tools wie Coffee-JsHint haben noch Kinderkrankheiten wie unbrauchbare Fehlermeldungen.

Development
lodash: Unverzichtbare Utilities

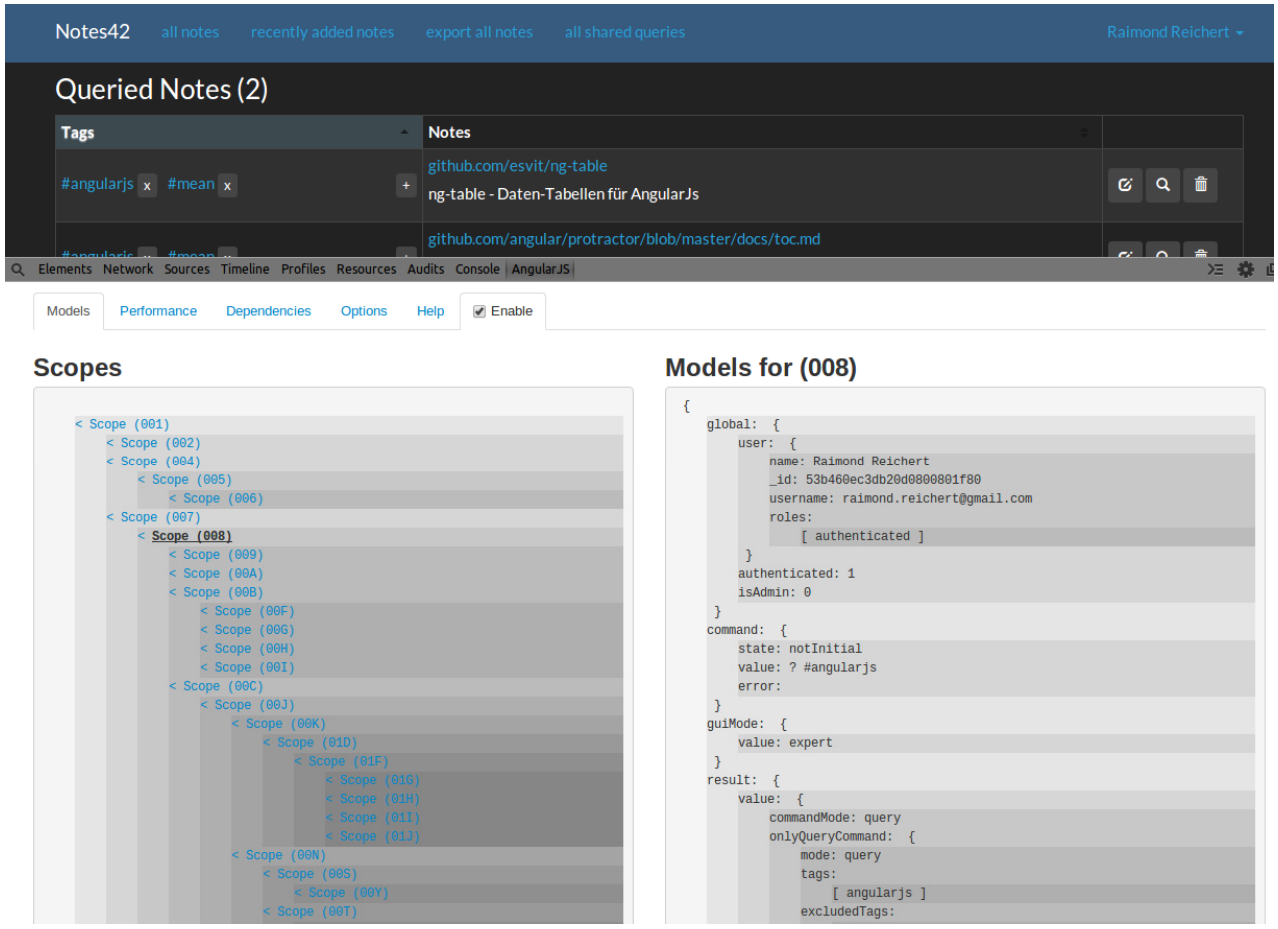
```
_removeTags: (command, tagArray) ->
 tagsToRemove = _.intersection command.excludedTags, tagArray
 _.xor tagArray, tagsToRemove

_addTags: (command, tagArray) ->
 _.union tagArray, command.tags

_getTagStringValue: (notes) ->
 tagStrings = _.map notes, (note) -> note.tagString
 sortedTagString = _.sortBy tagStrings, (tagString) -> tagString
 tagStringValue = _.reduce tagStrings, (result, value) -> result + ' ' + value

# ... und vieles mehr!
```

Development
AngularJs : Data Binding – HTML + Expressions + Scopes

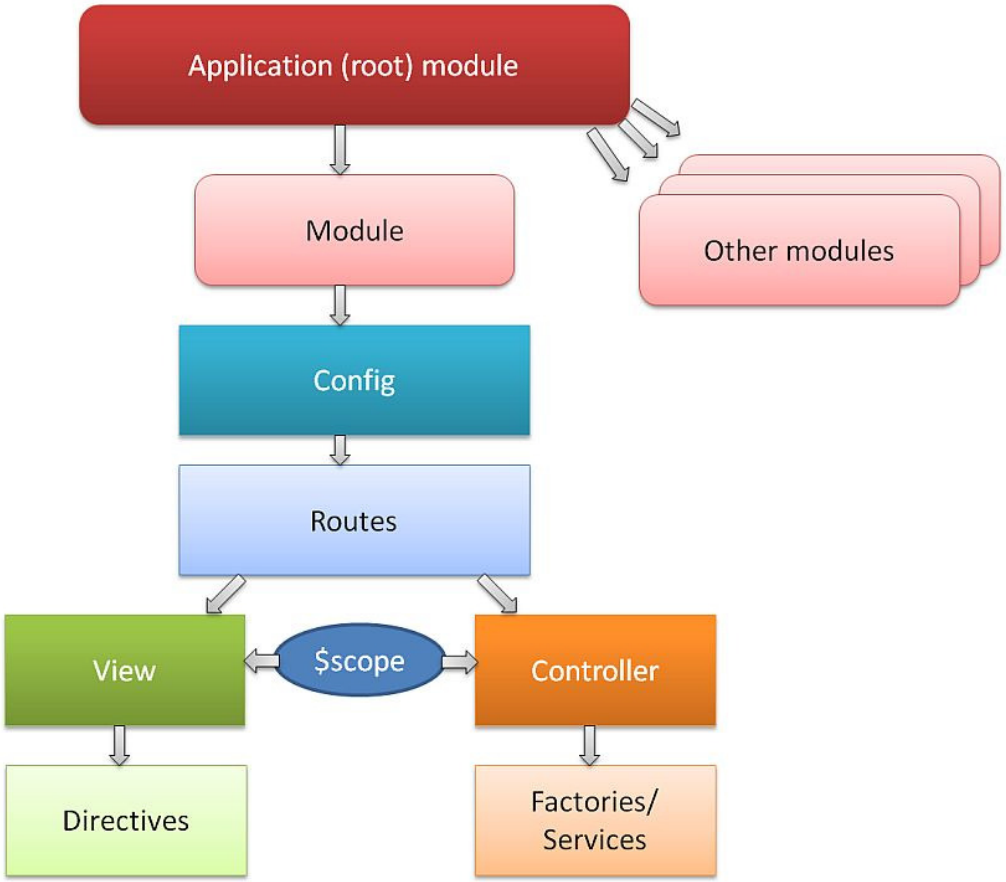
```
<html ng-app="phonecatApp"> <!-- head-Element weggelassen -->
<body ng-controller="PhoneListCtrl">
 <ul>
  <li ng-repeat="phone in phones">{{phone.name}}: {{phone.snippet}}</li>
 </ul>
</body>
</html>
```

```
var phonecatApp = angular.module('phonecatApp', []);
phonecatApp.controller('PhoneListCtrl', function ($scope) {
 $scope.phones = [
  {'name': 'Nexus S', 'snippet': 'Fast just got faster with Nexus S.'},
  {'name': 'Motorola XOOM', 'snippet': 'The Next, Next Generation tablet.'} ];
});
```

Development
AngularJs : Hierarchische Scopes – Debugging in Chrome



Development
AngularJs : Gibt App-Architektur vor – flache Lernkurve



Konzepte

- Template
- Directive
- Model
- Scope
- Expression
- Compiler
- Filter
- View
- Data Binding
- Controller
- Dependency Injection
- Module
- Service

Development
AngularJs: Google als Sponsor, starke Community

Google als Sponsor

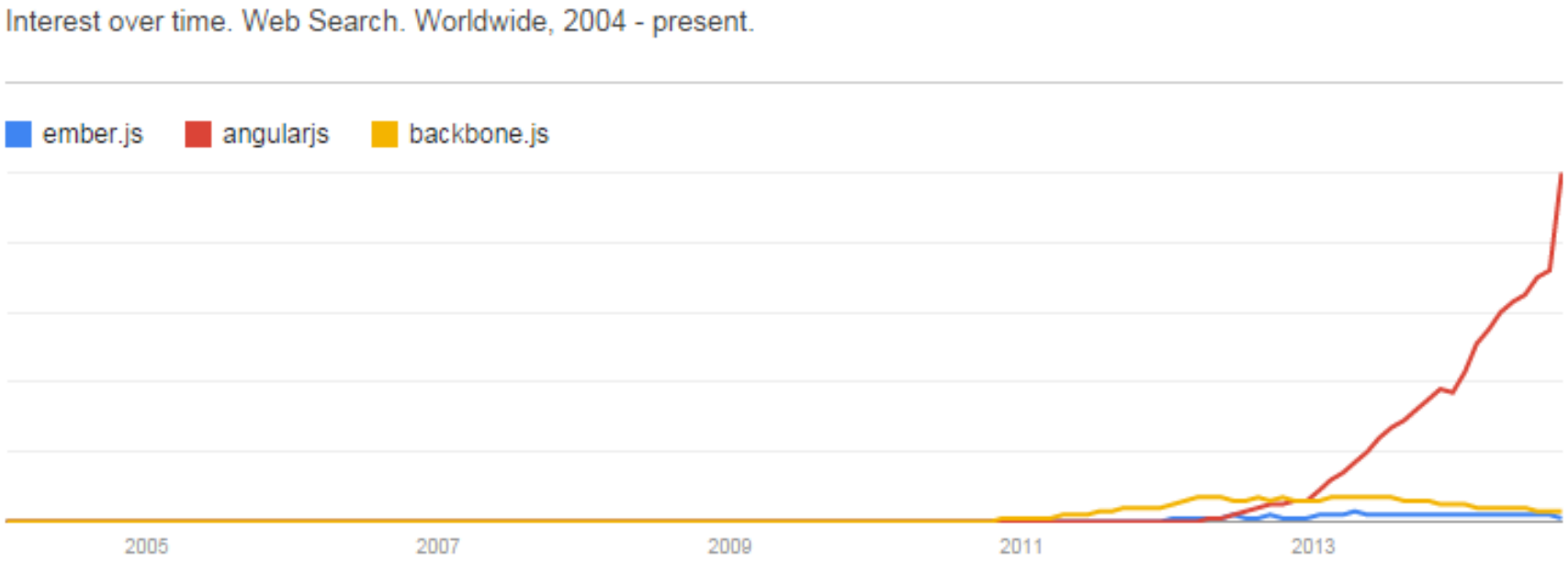
“Misko Hevery [...] managed to recreate a web application that consisted of 17 thousand lines of code and took 6 months to develop in a mere 3 weeks using just GetAngular. Reducing the size of the application to just about 1,000 lines of code convinced Google to start sponsoring the project, turning it into the open-source AngularJS we know today.“

Starke Community

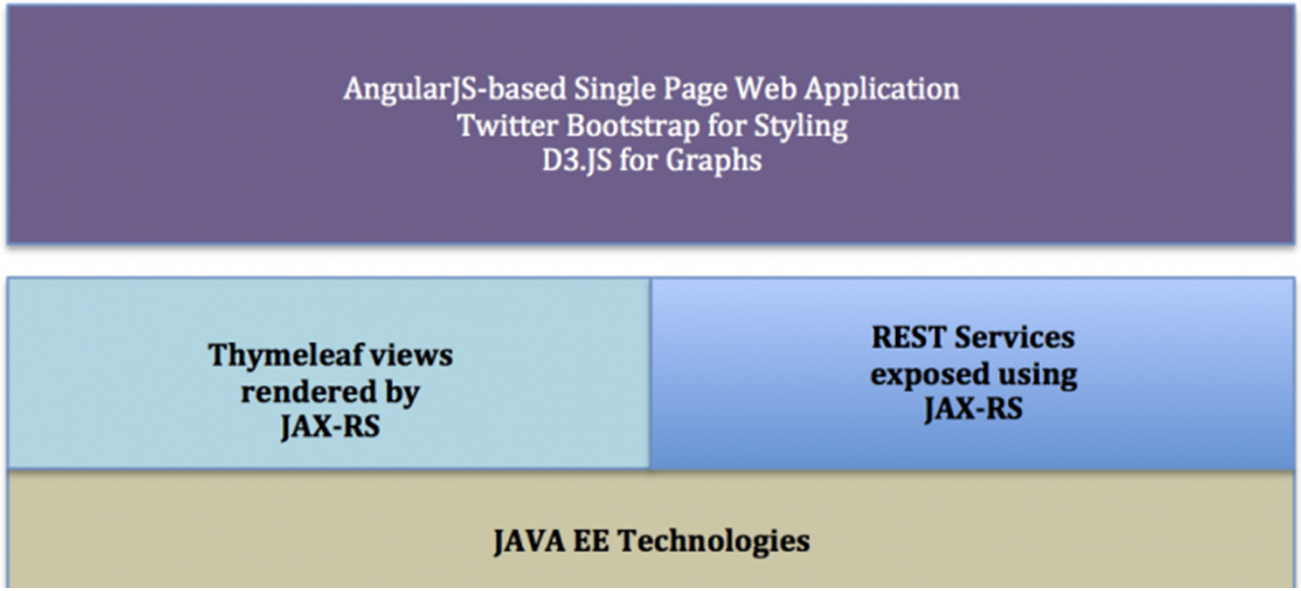
AngularJs hat über 800 Module auf ngmodules.org.

www.airpair.com/js/javascript-framework-comparison

Development
AngularJs: Verbreitung



Development
AngularJs: «Java EE 7 Real World Experience»



...

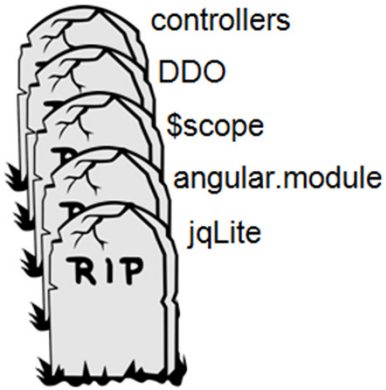
JEE7 – AngularJs als empfohlene Frontend-Technologie:
blog.arungupta.me/2014/09/log-your-miles-and-community-runs-java-ee-7-real-world-experience/

Development
AngularJs: P.S. Browser Kompatibilität...

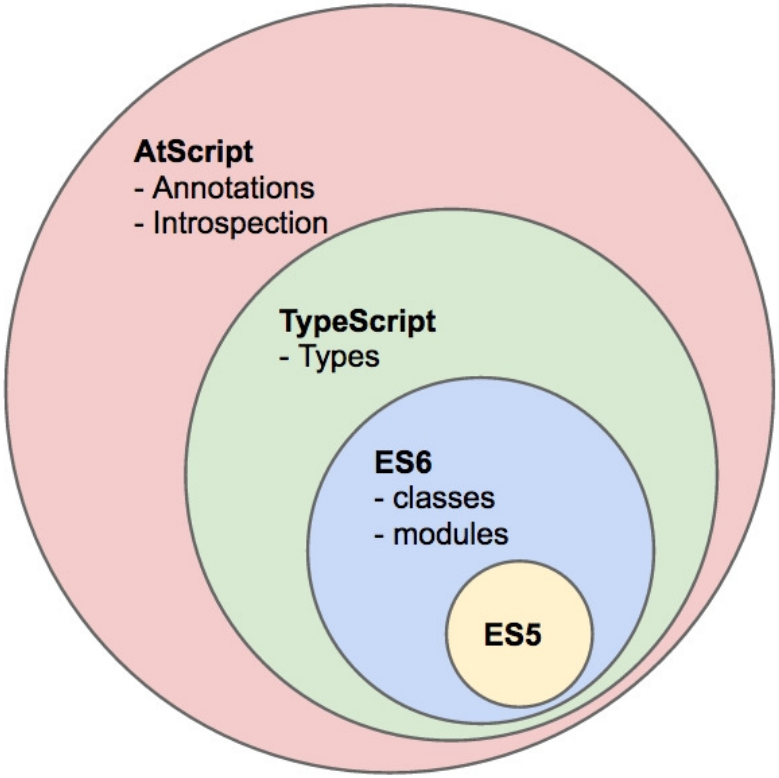
Unterstützt: Safari, Chrome, Firefox, Opera, IE9/10/11 und Mobile Browser (Android, Chrome Mobile, iOS Safari).

Nicht unterstützt: IE8. Da Microsoft im April 2014 offiziell den Support für Windows XP einstellte und Internet Explorer 8 vor allem darauf lief, unterstützt AngularJs ab Version 1.3 IE 8 nicht mehr.

AngularJs 2
Grundlegende Änderungen – kompletter Rewrite von 1.3



- generic binding syntax
- DI query mechanism
- benchpress
- WTF instrumentation



Development
NodeJs: Angenehm steile Lernkurve

example.js

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');
```

> node example.js

```
Server running at http://127.0.0.1:8124/
```

ergon

Development
NodeJs + ExpressJs: Routing einfach gemacht

```
var express = require('express');
var app = express();
app.get('/hello.txt', function(req, res) {
  res.send('Hello World');
});
var server = app.listen(3000, function() {
  console.log('Listening on port %d', server.address().port);
});
```

Development
NodeJs + ExpressJs: Server-side Rendering mit Jade

```
app.engine('jade', require('jade').__express);
app.get('/', function (req, res) {
  res.render('index', {
    title: 'Hey',
    message: 'Hello there!'
  });
})
```

index.jade

```
html
  head
    title!= title
  body
    h1!= message
```

Development
NodeJs: Leichtgewichtiger Container, einfach erweiterbar

- Im Vergleich zu JEE-Containern
  - ist Node **extrem leichtgewichtig** und startet in Sekundenbruchteilen,
  - gibt es keine Standards, aber eine sehr aktive Community.

- In NodeJs sind nur eine **überschaubare Anzahl APIs integriert.** Das wiederum hält die Lernkurve steil.
  - Für die Entwicklung von NodeJs-Webapplikationen kommt sehr oft auch **ExpressJs** (Web-App-Framework) zum Einsatz.

- Mit dem **Node Package Manager**. Via **www.npmjs.org** steht eine riesige Anzahl von Modulen zur Auswahl (Stand Sept 2014: 92'000).
  - "The **ecosystem** around JavaScript as a serious application platform continues to evolve. Many interesting new tools for testing, building, and managing dependencies in both server- and client-side JavaScript applications have emerged recently." (Thoughtworks Tech Radar 2014)

ergon

Development
NodeJs: Sinnvolle Szenarien für Einsatz von NodeJs

“Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.“ (nodejs.org)

Gut geeignet für …	Nicht geeignet für …
▪ Single-Page-Applikationen Backends / JSON-basierte REST-Services ▪ Web-Realtime-Applikationen ▪ Streaming	▪ rechenintensive Anwendungen

heise.de/developer/artikel/2x-Nein-4x-Ja-Szenarien-fuer-Node-js-2111050.html

Development
NodeJs: Produktiv einsetzbar und eingesetzt

- **LinkedIn** hat von Ruby on Rails auf NodeJs umgestellt, siehe etwa cacm.acm.org/magazines/2014/2/171684-node-at-linkedin.
  - NodeJs als Server-seitiger Integrationslayer für diverse Backend-Systeme, die grösstenteils in JEE entwickelt wurden.
  - Beeindruckende Performance: «we discovered that Node was roughly 20 times faster than what we had been using and its memory footprint was smaller as well.»
- Microsoft, VMWare, Ebay, Yahoo und viele andere setzen NodeJs ein.

Development
MongoDB: Dokument-basierte Datenbank

Eigenwerbung von mongodb.org
MongoDB (from "humongous") is an open-source document database, and the leading NoSQL database. Written in C++, MongoDB features:

- **Document-Oriented Storage:** JSON-style documents with dynamic schemas offer simplicity and power.
- **Full Index Support:** Index on any attribute, just like you're used to.
- **Querying:** Rich, document-based queries.
- Map/Reduce, GridFs und vieles mehr

Development
mongoosejs: «elegant mongodb object modeling»

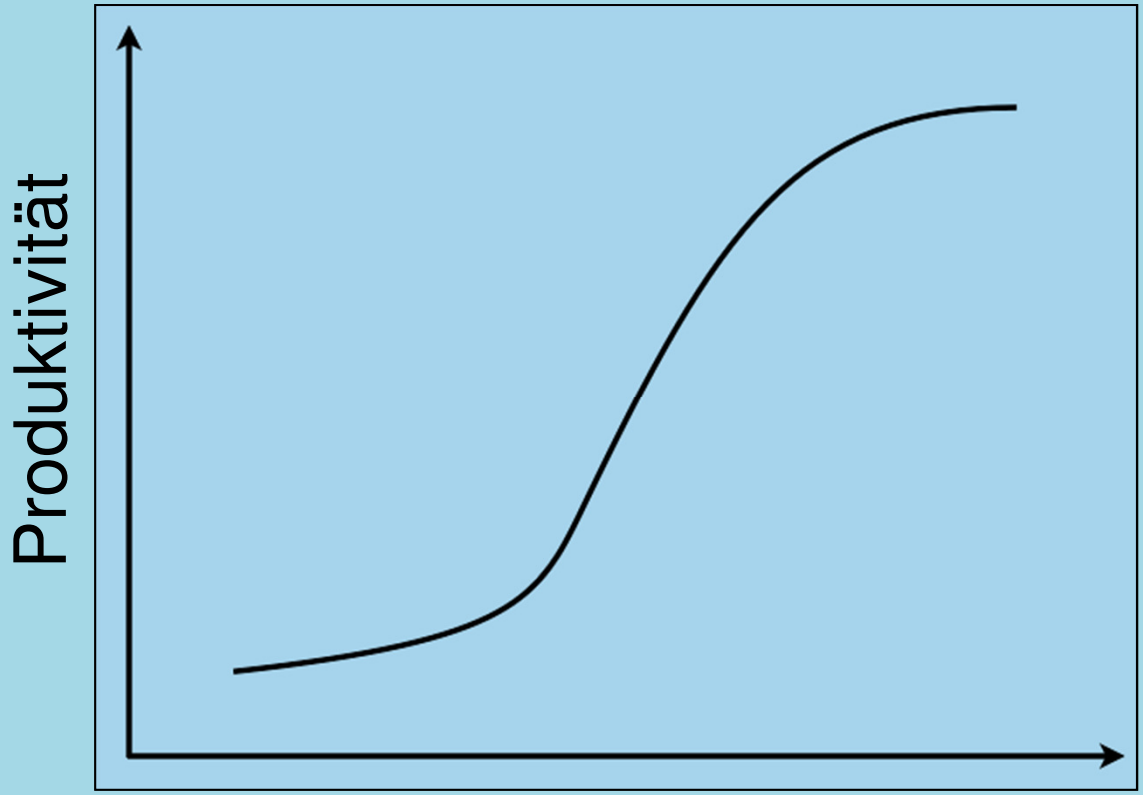

```
mongoose = require 'mongoose'
Schema = mongoose.Schema
timestamps = require 'mongoose-times'

NoteSchema = new Schema {
 text: String
 tagArray: [
  type: String
  index: true
 ]
 tagString:
  type: String
  default: ''
  trim: true
 url:
  type: String
  default: ''
  index: true
 ownedBy:
  type: Schema.Types.ObjectId
  ref: 'User'
  index: true
}
NoteSchema.plugin timestamps
```

ergon

Development
MongoDb + mongoosejs: Aggregation Pipeline-Beispiel

```
class TagsFinder
 # constructor not shown
 find: (command, successCallback, failureCallback) ->
 Note.aggregate([
  {$match: @matchConditionFactory.queryMatchCondition(command)}
  {$unwind: '$tagArray'}
  {$group: {_id: '$tagArray', number: {$sum: 1}}}
  {$sort: {_id: 1}}
 ]).exec (err, tags) ->
  mappedTags = _.map tags, (tag) -> { value: tag._id, count: tag.number  }
  successCallback mappedTags
  # error handling not shown
```

Der Durchbruch
Nach 7 Tagen gefühlt sehr produktiv



ergon

Der Rückblick

ergon

Lessons Learned
Methodisch

- **Zur Vorbereitung**
  - **Unverzichtbar, um in der kurzen Zeit produktiv entwickeln zu können**
  - **Zeit für Infrastruktur-Aufbau: Der Aufwand zahlt sich rasch wieder aus**
  - **Einlesen in die Technologien: Spezialisierung hilfreich**
  - **Stories auf Github halfen, Projekt zu strukturieren und uns zu fokussieren**
- **Zur Durchführung**
  - «Extrem pair programming»: Deutlich effizienter gelernt als alleine
  - **Kurze Feedback-Zyklen: abends «lessons learned» notieren, morgens umsetzen**
  - **Time boxing: Manchmal müssen pragmatische Hacks genügen**
  - **«Continuous Deployment»:** Von Anfang an regelmässige Verifikation auf herokuapp.com
  - **Das Erfolgserlebnis: Nach 7 Tagen auch individuell produktiv**

Lessons Learned
Was noch fehlt für «produktives Projekt»

- **Continuous Integration**
  - Zum Beispiel mit Jenkins: blog.dylants.com/2013/06/21/jenkins-and-node
  - Strider (stridercd.com) für Selbstbetrieb; auch als Hosted Service zu haben
  - Travis-Ci (travis-ci.org) als Hosted Service für Github-Projekte
  - CodeShip (www.codeship.io) als Hosted Service

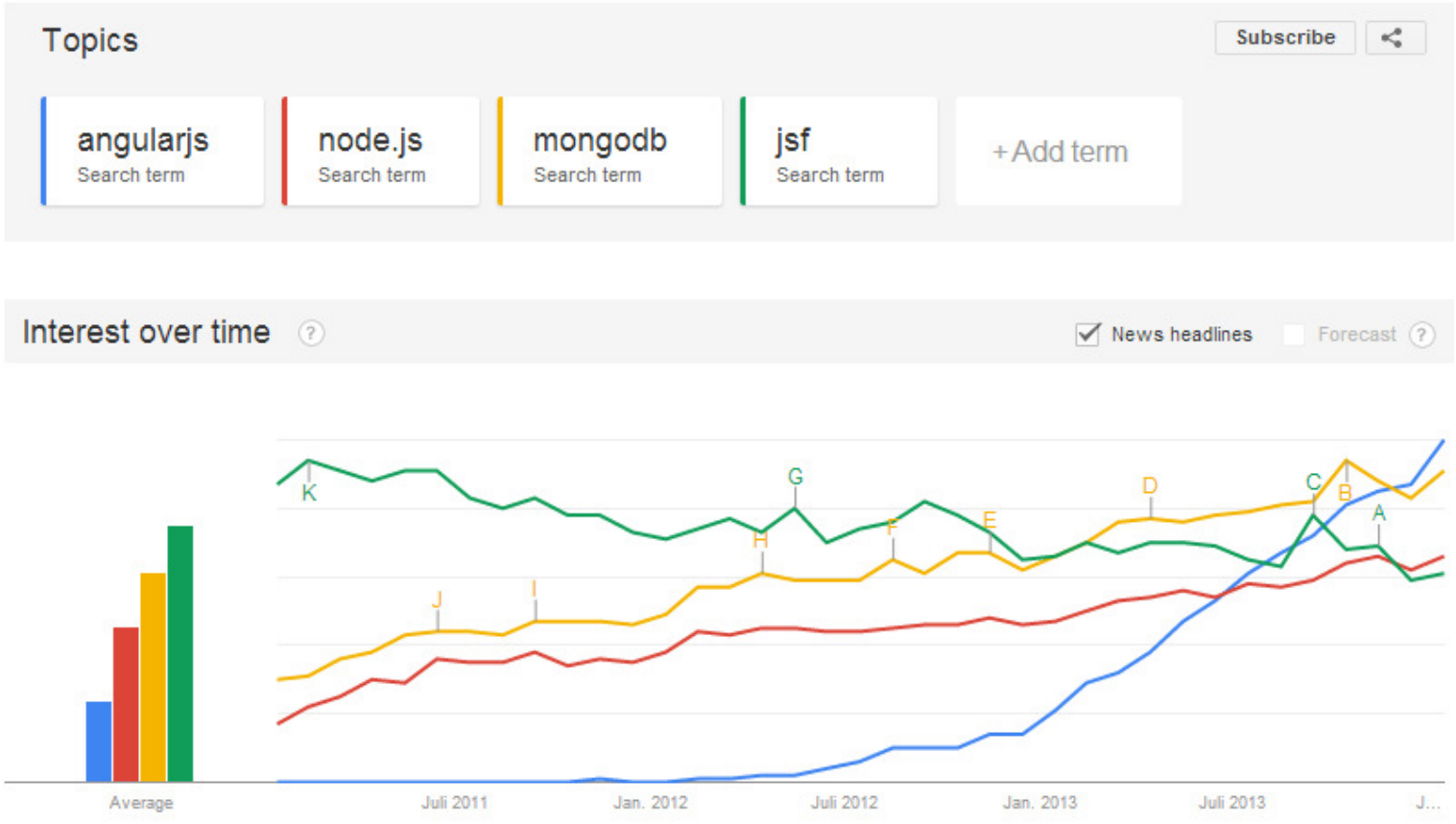
- **Betrieb von NodeJs-Server**
  - Einzelne NodeJs-Server: Sehr einfach mit Forever (github.com/nodejitsu/forever)
  - Clusters zum Beispiel mit Strongloop (strongloop.com/node-js/controller)

Lessons Learned
Was noch fehlt für «produktives Projekt»

- **Code Coverage Tools**
  - Istanbul (github.com/gotwarlost/istanbul) wird von Karma Test Runner verwendet
  - BlanketJs (blanketjs.org)
  - JsCover (tntim96.github.io/JSCover)
- **Performanz-Analyse-Tools**
  - Strongloop (strongloop.com/node-js/monitoring/)
  - Weitere Möglichkeiten siehe www.clock.co.uk/blog/easy-cpu-profiling-in-node-js

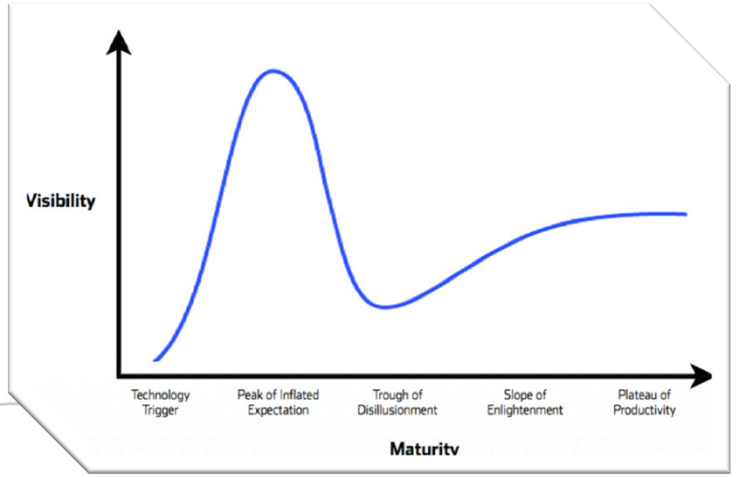

The Future
?
NEXT EXIT

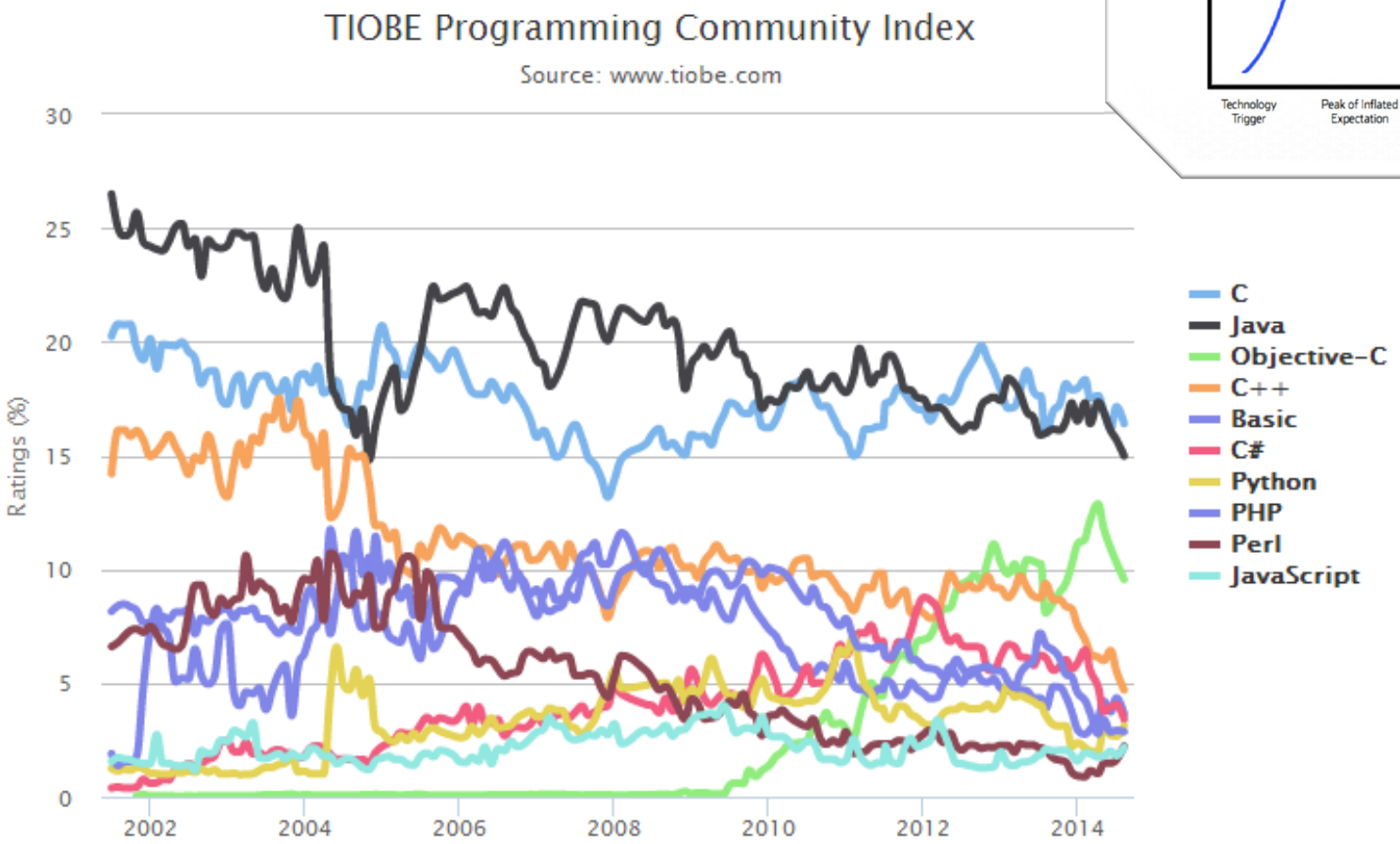
Lessons Learned
JavaScript macht Spass und ist voll im Trend ...



ergon

Lessons Learned
... der noch sehr jung ist!





ergon

notes42.herokuapp.com

«notes for two» oder «notes fortytwo»?

Raimond Reichert, Samuel Zürcher
Ergon Informatik AG