Meteor
Fullstack JavaScript Development

- Retro42: Our prototype application
- Why did we choose Meteor?
- What is Meteor?
- Show me some code!
- Comparing Meteor vs. MEAN
- More about Meteor

Raimond Reichert, Samuel Zürcher, Ergon Informatik AG
The rate of innovation in the JavaScript space is still incredible. For a while, it seemed that Angular would emerge as a winner in the web application frontend wars. Now, with Meteor, there is a new contender, a full-stack development framework which promises "to allow you to build advanced apps quickly, with a small team – allowing even individuals to achieve things that used to be possible only at large companies like Google and Facebook."

We wanted to gain some experience with Meteor with a very small project (approx. 20d, two people) to form our own opinion on these promises. We were quite impressed, as our estimates of what we would be able to build were way off: We developed roughly 50% more functionality than we had estimated. Meteor seemed easy to learn, and we very quickly felt very productive with Meteor.

In this presentation, we give a short overview on Meteor (with its Distributed Data Protocol, Latency Compensation, and Reactivity) and its components for rendering (Blaze) and testing (Velocity) as well as its package system (atmosphere.js). In a "slide coding" session, we implement a "lessons learned" feature of our demo application, Retro42 (retro42.herokuapp.com) as a showcase for coding with Meteor.

Raimond Reichert, Samuel Zürcher, Ergon Informatik AG
Meteor
Fullstack JavaScript Development

- Retro42: Our prototype application
- Why did we choose Meteor?
- What is Meteor?
- Show me some code!
- Comparing Meteor vs. MEAN
- More about Meteor
Retro42: Our prototype application

Our motivation for building Retro42

- Change begins with setting goals, and tracking how you do on them.
- Retro42 lets you define questionnaires, and you can schedule them for yourself, or for your team.
- Retro42 helps you track your answers, and visualizes them for you, grouping by answers by question, or charting them as time-series.
Retro 42: Demo Screenshots

Start from an existing questionnaire template...

Daily Retrospective for a Software Engineer
A daily retrospective for a software engineer.

Five minute journal (evening)
A daily journal inspired by fiveminutejournal.com

Five minute journal (morning)
A daily journal inspired by fiveminutejournal.com
Retro 42: Demo Screenshots

... or create a questionnaire from scratch.

Fragebogen erstellen

Titel
Five minute journal (evening)

Beschreibung
A daily journal inspired by fiveminutejournal.com

Fragen

Frage
Happy?

Erläuterung
Are you happy with today?

Typ
Ja / Nein
Define your schedule for answering your questionnaire.
Retro42 will send you reminder mails according to the schedule.
Retro 42: Demo Screenshots

Submit answers to your questionnaire

<table>
<thead>
<tr>
<th>Retro42</th>
<th>Fragebogenerzeugen</th>
<th>Total: 10</th>
</tr>
</thead>
</table>

Five minute journal (evening) 10

A daily journal inspired by fiveminutejournal.com

Amazing things that happened today...
Significant events that happened today.

How could I have made today better?
If I could go back, what would I do differently?

Ihre letzten Antworten:
- I did not have a single interrupt today.
- I got a lot of work done today.
- My best friends wedding was amazing.
- I got off work early.
- The project kickoff went smoothly.

Ihre letzten Antworten:
- I would schedule a lunch meeting so that I am not alone all day.
- I would check email less often.
- I would drink less.
- I should not yell at my boss....
- I would make the work stay within the timebox.
### Five minute journal (evening)

A daily journal inspired by fiveminutejournal.com

<table>
<thead>
<tr>
<th>Frage</th>
<th>Beschreibung</th>
<th>Aktionen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazing things that happened today...</td>
<td>Significant events that happened today.</td>
<td>Ansehen</td>
</tr>
<tr>
<td>How could I have made today better?</td>
<td>If I could go back, what would I do differently?</td>
<td>Ansehen</td>
</tr>
<tr>
<td>Your interactions?</td>
<td>Rate your interactions with others today. 1: Extremely tedious. 10: Extremely energizing.</td>
<td>Ansehen</td>
</tr>
<tr>
<td>Happy?</td>
<td>Are you happy with today?</td>
<td>Ansehen</td>
</tr>
</tbody>
</table>

**Ausgefüllte Fragebogen**

<table>
<thead>
<tr>
<th>Erstellt</th>
<th>Your Interactions?</th>
<th>Happy?</th>
<th>Aktionen</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-08-11 20:43:00 vor einem Tag</td>
<td>5</td>
<td>✅</td>
<td>Bearbeiten, Ansehen</td>
</tr>
<tr>
<td>2015-08-10 20:43:00 vor 2 Tagen</td>
<td>7</td>
<td>✅</td>
<td>Bearbeiten, Ansehen</td>
</tr>
</tbody>
</table>
Retro 42: Demo Screenshots

Review your answers to a textual question

Five minute journal (evening)  10
A daily journal inspired by fiveminutejournal.com

Amazing things that happened today...
Significant events that happened today.

Antworten

<table>
<thead>
<tr>
<th>Erstellt</th>
<th>Antwort</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-08-11 20:43:41 · vor einem Tag</td>
<td>I did not have a single interrupt today.</td>
</tr>
<tr>
<td>2015-08-10 20:43:41 · vor 2 Tagen</td>
<td>I got a lot of work done today.</td>
</tr>
<tr>
<td>2015-08-09 20:43:41 · vor 3 Tagen</td>
<td>My best friends' wedding was amazing.</td>
</tr>
<tr>
<td>2015-08-08 20:43:41 · vor 4 Tagen</td>
<td>I got off work early.</td>
</tr>
<tr>
<td>2015-08-07 20:43:41 · vor 5 Tagen</td>
<td>The project kickoff went smoothly.</td>
</tr>
<tr>
<td>2015-08-06 20:43:41 · vor 6 Tagen</td>
<td>I loved the lunch meal.</td>
</tr>
<tr>
<td>2015-08-05 20:43:41 · vor 7 Tagen</td>
<td>The first day of school of my youngest child went great.</td>
</tr>
<tr>
<td>2015-08-04 20:43:41 · vor 8 Tagen</td>
<td>The train was on time today.</td>
</tr>
<tr>
<td>2015-08-03 20:43:41 · vor 9 Tagen</td>
<td>My baby walked for the first time.</td>
</tr>
<tr>
<td>2015-08-02 20:43:41 · vor 10 Tagen</td>
<td>I scored five soccer goals today.</td>
</tr>
</tbody>
</table>
Retro 42: Demo Screenshots

Review your answers to a numeric question

Five minute journal (evening)  
A daily journal inspired by fiveminutejournal.com

Your interactions?
Rate your interactions with others today. 1: Extremely tedious. 10: Extremely energizing.

Statistische Werte

![Graph showing statistical values over time]
Review the percentage of positive answers to a yes/no question.
Retro 42: Demo Screenshots

Review a yes/no answers on a calendar

Don't break the chain

<table>
<thead>
<tr>
<th>Mon</th>
<th>Di</th>
<th>Mi</th>
<th>Do</th>
<th>Fr</th>
<th>Sa</th>
<th>So</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Legend:
- Green: 0
- Light Green: 1
- Red: 2
- Dk Red: 3
- Brown: 4
Retro 42: Demo Screenshots

Invite others to participate on a questionnaire

Weekly Retrospective for a Software Engineering Team

Einladung für diesen Fragebogen

Falls du diesen Fragebogen teilen willst, sende diesen Link:

http://localhost:3000/questionnaires/HNedBbiv5aKHBPk7F/invite

Zurück
Retro 42: Demo Screenshots

Review a shared questionnaire and its participants

Weekly Retrospective for a Software Engineering Team

<table>
<thead>
<tr>
<th>Frage</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describe the week in one word.</td>
<td>Just one word.</td>
</tr>
<tr>
<td>How is your motivation (1-10)?</td>
<td>1: No motivation at all. 10: Super motivated.</td>
</tr>
<tr>
<td>How is your stress level (1-10)?</td>
<td>1: Completely relaxed. 10: Super stressed out.</td>
</tr>
<tr>
<td>Ideas for improvements?</td>
<td>What could be done so that you can get your work done more efficiently?</td>
</tr>
<tr>
<td>Number one impediment?</td>
<td>What prevents you the most to get your work done?</td>
</tr>
<tr>
<td>Successful?</td>
<td>Do you consider the week to be successful?</td>
</tr>
</tbody>
</table>

Teilnehmer

- sam.zurcher+se1@gmail.com
- sam.zurcher+se2@gmail.com
- sam.zurcher+se3@gmail.com
Retro 42: Demo Screenshots

Review your group's questionnaire passes

<table>
<thead>
<tr>
<th>Teilnehmer</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:sam.zurcher+se1@gmail.com">sam.zurcher+se1@gmail.com</a></td>
<td></td>
</tr>
<tr>
<td><a href="mailto:sam.zurcher+se2@gmail.com">sam.zurcher+se2@gmail.com</a></td>
<td></td>
</tr>
<tr>
<td><a href="mailto:sam.zurcher+se3@gmail.com">sam.zurcher+se3@gmail.com</a></td>
<td></td>
</tr>
<tr>
<td><a href="mailto:sam.zurcher+se4@gmail.com">sam.zurcher+se4@gmail.com</a></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ausgefüllte Fragebogen</th>
</tr>
</thead>
<tbody>
<tr>
<td>% beantwortet</td>
</tr>
<tr>
<td>100%</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>100%</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>100%</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>100%</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>100%</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Retro 42: Demo Screenshots

Review all answers in a group questionnaire pass

### Weekly Retrospective for a Software Engineering Team

A weekly retrospective for a software engineering team.

<table>
<thead>
<tr>
<th>Frage</th>
<th>Beschreibung</th>
<th>Antworten</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describe the week in one word.</td>
<td>Just one word.</td>
<td>• Great</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fantastic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Superb</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Enjoyable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Relaxing</td>
</tr>
<tr>
<td>How is your motivation (1-10)?</td>
<td>1: No motivation at all. 10: Super motivated.</td>
<td>Mittelwert 9.2, Median 9. Minimum 9, Maximum 10</td>
</tr>
<tr>
<td>How is your stress level (1-10)?</td>
<td>1: Completely relaxed. 10: Super stressed out.</td>
<td>Mittelwert 2, Median 2. Minimum 1, Maximum 3</td>
</tr>
<tr>
<td>Ideas for improvements?</td>
<td>What could be done so that you can get your work done more efficiently?</td>
<td>• Successful release!</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Successful retrospective!</td>
</tr>
<tr>
<td>Number one impediment?</td>
<td>What prevents you the most to get your work done?</td>
<td>• Successful planning!</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Successful project kickoff!</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Vacation time!</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Release more often!</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Use Retro42 more often!</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Use planning poker consistently!</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Start more smaller projects!</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• No vacations when everyone else is on vacation:)</td>
</tr>
</tbody>
</table>
Weekly Retrospective for a Software Engineering Team

How is your motivation (1-10)?

1: No motivation at all 10: Super motivated.

Statistische Werte
Weekly Retrospective for a Software Engineering Team

A weekly retrospective for a software engineering team.

**Successful?**
Do you consider the week to be successful?

**Erfolgsquote**

![Graph showing performance over time](image-url)
Meteor

Fullstack JavaScript Development

- Retro42: Our prototype application
- Why did we choose Meteor?
- What is Meteor?
- Show me some code!
- Comparing Meteor vs. MEAN
- More about Meteor
Why did we choose Meteor?

In 2015, what would you choose?

So many web and mobile app development options, yet still no clear winner emerges... making each option a risk with regards to long-term maintainability.

We chose Meteor because it implements a different approach which is quite different from what we've gotten used to.
Why did we choose Meteor?

Integrated development platform

Why did we choose Meteor?

Principles of particular interest

- Full-stack JavaScript (or CoffeeScript, for us), for both backend and frontend
- Meteor's **database everywhere** principle makes it easy to build apps on top of databases
- We wanted to see Meteor's **latency compensation** principle and reactivity in action.
- Meteor advertises **fast and fun development**, and who wouldn't like that:-)  
- It promises one code base for web and mobile platforms. However, we did not look into Meteor's multi-platform capabilities in our project.
Meteor

Fullstack JavaScript Development

- Retro42: Our prototype application
- Why did we choose Meteor?
- What is Meteor?
- Show me some code!
- Comparing Meteor vs. MEAN
- More about Meteor
What is Meteor?

**Command line tool**

- `meteor add <package name>`
  Add a package to your Meteor project.

- `meteor run`
  Serve the current app at http://localhost:3000 using Meteor's local development server.
  Also continuously runs tests if Velocity package has been added to your project.

- `meteor reset`
  Reset the current project to a fresh state.
  Removes all local data.
What is Meteor?

Its own build system

- For example, `meteor add coffeescript` adds CoffeeScript support to your project – that's it.
- Client and server: Packages work seamlessly on both client and server.
- Asset building and bundling: The package system integrates with Meteor's asset building/bundling system.
- "We feel strongly that every package in your app should be always pinned to a specific version, and those version pins should be checked into source control."

see quora.com/Node-js/Why-does-Meteor-use-its-own-package-system-rather-than-NPM
What is Meteor?

Full-stack package system

See atmospherejs.com
What is Meteor?

**Integrated MongoDB support**

The same database API works on both client and server. The API is compatible with the Mongo database API.

```javascript
# common code on client and server declares a DDP-managed mongo collection
Messages = new Mongo.Collection "messages"

# return array of my messages (here, in client code)
myMessages = Messages.find({userId: Meteor.userId()}).fetch()

# create a new message
Messages.insert {text: "Hello, world!"

# mark my first message as "important"
Messages.update myMessages[0]._id, {$set: {important: true}}
```

see [docs.meteor.com/#/full/mongo_collection](https://docs.meteor.com/#/full/mongo_collection)
What is Meteor?

Integrated MongoDB support

- On the server, when you call methods on a collection, they translate directly into normal Mongo operations (after checking that they match your access control rules).
- On the client, Minimongo is essentially an in-memory, non-persistent implementation of Mongo in pure JavaScript. It serves as a local cache that stores just the subset of the database that this client is working with.
- When you write to the db on the client, the command is executed locally immediately, and, simultaneously, it's sent to the server and executed there too.

see docs.meteor.com/#/full/mongo_collection
What is Meteor?

Mini-MongoDb limitations

In its current release, Minimongo has some limitations:

- $pull in modifiers only accepts certain kinds of selectors.
- findAndModify, **aggregate functions, and map/reduce aren't supported.**
- Minimongo doesn't currently have indexes. It's rare for this to be an issue, since it's unusual for a client to have enough data that an index is worthwhile.

All of these will be addressed in a future release.

see docs.meteor.com/#/full/mongo_collection
What is Meteor?

From event-driven-programming...

```javascript
var spaceship = {
    _temperature: 0, _listeners: [],

    getTemperature: function() {
        return this._temperature;
    },

    setTemperature: function(value) {
        this._temperature = value;
        this.changed();
    },

    // to be continued
};
```

from stephenwalther.com/archive/2014/12/05/dont-do-react-understanding-meteor-reactive-programming
What is Meteor?

From event-driven-programming...

```javascript
var spaceship = { // code from previous slide
    changed: function() {
        for (var i=0; i< this._listeners.length; i++) {
            this._listeners[i]();
        }
    },

    onChanged: function(func) {
        this._listeners.push(func);
    }
};

// observer updates dashboard when temperature changes
spaceship.onChanged(function() {
    var temp = spaceship.getTemperature();
    gauge.setValue(temp);
});
```

from stephenwalther.com/archive/2014/12/05/dont-do-react-understanding-meteor-reactive-programming
What is Meteor?
... towards reactive programming

```javascript
var spaceship = {
    _temperature: 0,
    _temperatureDepend: new Tracker.Dependency,

getTemperature: function() {
    this._temperatureDepend.depend();
    return this._temperature;
},

setTemperature: function(value) {
    this._temperature = value;
    this._temperatureDepend.changed();
}
};
```

*from stephenwalther.com/archive/2014/12/05/dont-do-react-understanding-meteor-reactive-programming*
What is Meteor?
... towards reactive programming

// Tracker updates dashboard when temperature changes
Tracker.autorun(function() {
  var temp = spaceship.getTemperature();
  gauge.setValue(temp);
});

The function passed to Tracker.autorun() is rerun automatically whenever any of its dependencies change.

from stephenwalther.com/archive/2014/12/05/dont-do-react-understanding-meteor-reactive-programming
What is Meteor?

Meteor Reactivity is Transparent

You rarely work with Tracker Dependency directly. Instead, you work with Meteor's reactive data sources such as:

- Session object
- Reactive variables
- Minimongo
- Meteor.user(), Meteor.status()

from stephenwalther.com/archive/2014/12/05/dont-do-react-understanding-meteor-reactive-programming
What is Meteor?

Livequery: Latency Compensation

from info.meteor.com/blog/optimistic-ui-with-meteor-latency-compensation
What is Meteor?

Meteor Data Flows

from meteor.hackpad.com/Meteor-speaker-kit-uaPe3zDDH8z
What is Meteor?

Meteor Data Flows

from meteor.hackpad.com/Meteor-speaker-kit-uaPe3zDDH8z
What is Meteor?

Meteor Data Flows

from meteor.hackpad.com/Meteor-speaker-kit-uaPe3zDDH8z
What is Meteor?

Meteor Data Flows

from meteor.hackpad.com/Meteor-speaker-kit-uaPe3zDDH8z
What is Meteor?

Meteor Data Flows

from meteor.hackpad.com/Meteor-speaker-kit-uaPe3zDDH8z
What is Meteor?

Meteor Data Flows

from meteor.hackpad.com/Meteor-speaker-kit-uaPe3zDDH8z
What is Meteor?

Meteor Data Flows

from meteor.hackpad.com/Meteor-speaker-kit-uaPe3zDDH8z
What is Meteor?

Meteor Data Flows

from meteor.hackpad.com/Meteor-speaker-kit-uaPe3zDDH8z
What is Meteor?

Meteor Data Flows

from meteor.hackpad.com/Meteor-speaker-kit-uaPe3zDDH8z
What is Meteor?

Meteor Data Flows

from meteor.hackpad.com/Meteor-speaker-kit-uaPe3zDDH8z
What is Meteor?

Meteor Data Flows

from meteor.hackpad.com/Meteor-speaker-kit-uaPe3zDDH8z
Spacebars is simply HTML, with the addition of three things:

- **Inclusions** use the `{{> templateName}}` syntax, and simply tell Meteor to replace the inclusion with the template of the same name.

- **Expressions** such as `{{title}}` either call a property of the current object, or the return value of a template helper as defined in the current template’s helper.

- **Template helpers** are special tags that control the flow of the template, such as `{{#each}}` ... `{{/each}}` or `{{#if}}`...`{{/if}}`. 
What is Meteor?

View with Blaze and Spacebars

In this example, **url** and **title** come from a **post** object, whereas **domain** is a call to a template helper function:

```html
<template name="postItem">
  <div class="post">
    <div class="post-content">
      <h3><a href="{{url}}">{{title}}</a><span>{{domain}}</span></h3>
    </div>
  </div>
</template>
```

```
Template.postItem.helpers
  domain: () ->
    a = _createLink this.url
    a.hostname
```
Blaze is reactive: The template will update automatically, with no additional work on your part. This works even if the template runs arbitrary JavaScript code to compute its data.

Blaze's simplicity is made possible by Tracker, an extremely lightweight (one kb) system for transparent reactivity.

By wiring all of this up for you, Blaze removes an enormous amount of boilerplate from your app, and allows you to say what you want once without repeating yourself.

from meteor.com/blaze
What is Meteor?

Integrated development platform

from meteor.hackpad.com/Meteor-speaker-kit-uaPe3zDDH8z
Meteor

Fullstack JavaScript Development

- Retro42: Our prototype application
- Why did we choose Meteor?
- What is Meteor?
  - Show me some code!
- Comparing Meteor vs. MEAN
- More about Meteor
Implementing "lessons learned"

Let user add lessons learned to a questionnaire:

How is your admin stress level? 1
A questionnaire regarding your admin stress level.

<table>
<thead>
<tr>
<th>Question</th>
<th>Explanation</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stressed?</td>
<td>Something that stressed you out today.</td>
<td>View</td>
</tr>
<tr>
<td>Relaxed?</td>
<td>Something that relaxed you today.</td>
<td>View</td>
</tr>
<tr>
<td>Stress Level (1-10)</td>
<td>Numeric</td>
<td>View</td>
</tr>
<tr>
<td>Are you less stressed today than yesterday?</td>
<td>Boolean</td>
<td>View</td>
</tr>
</tbody>
</table>

Lessons Learned

Add a lesson learned

Answered questionnaires

<table>
<thead>
<tr>
<th>Created at</th>
<th>Stress Level (1-10)</th>
<th>Are you less stressed today than yesterday?</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-09-06 21:12 12 hours ago</td>
<td>2</td>
<td>✗</td>
<td>Edit</td>
</tr>
</tbody>
</table>
Show me some code!

Implementing "lessons learned"

Display lessons learned of a questionnaire:

How is your admin stress level? 1
A questionnaire regarding your admin stress level.

<table>
<thead>
<tr>
<th>Question</th>
<th>Explanation</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stressed?</td>
<td>Something that stressed you out today.</td>
<td>View</td>
</tr>
<tr>
<td>Relaxed?</td>
<td>Something that relaxed you today.</td>
<td>View</td>
</tr>
<tr>
<td>Stress Level (1-10)</td>
<td>Numeric</td>
<td>View</td>
</tr>
<tr>
<td>Are you less stressed today than yesterday?</td>
<td>Boolean</td>
<td>View</td>
</tr>
</tbody>
</table>

Lessons Learned

- **Created at**: 2015-09-07 09:03 - a few seconds ago
  - **i should take more breaks**

- **Add a lesson learned**

Answered questionnaires

<table>
<thead>
<tr>
<th>Created at</th>
<th>Stress Level (1-10)</th>
<th>Are you less stressed today than yesterday?</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-09-06 21:12 - 12 hours ago</td>
<td>2</td>
<td>✗</td>
<td>Edit View</td>
</tr>
</tbody>
</table>
Show me some code!

Include with Blaze, Spacebars

```html
<template name="questionnairePasses">
  {{> _questionnaireTitleDescription}}
  <!-- left out: display questionnaire questions -->
  {{> _questionnaireButtons}}

  {{> lessonsLearnedTemplate questionnaireId=_id}}

  <!-- left out: display questionnaire passes -->
</template>
```

- We include the `lessonsLearnedTemplate` in the `questionnairePasses` template.
- We pass this questionnaire's `_id` as `questionnaireId`. 
The template name `lessonsLearnedTemplate` is the key which we will use in our CoffeeScript code.

`{{_ 'key'}}` is i18n with Meteor package tap:i18n.

We left out some of the table markup (thead).
Show me some code!

Display with Blaze, Spacebars

```handlebars
{{#each lessonsLearned}}
  <tr>
    <td class="createdAt">{{createdAtAbsolute}} {{createdAtRelative}}</td>
    <td class="text">{{text}}</td>
    <td class="actions">
      <a href="#"
         class="deleteLessonLearned btn btn-default btn-sm">
        {{_ 'lessonsLearnedTemplate.delete'}}
      </a>
    </td>
  </tr>
{{/each}}
```

- Examples of calling a template helper function: `lessonsLearned`, `createdAtAbsolute`.
- Example of looking up a value in current scope: `text`.
- Iterating with `{{#each lessonsLearned}}` creates a new `this` scope for each document in the collection.
Show me some code!

**Template helper functions**

```javascript
Template.lessonsLearnedTemplate.helpers

lessonsLearned: () ->
  LessonsLearned.find({questionnaireId: this.questionnaireId},
  {sort: {createdAt: -1}}).fetch()

createdAtAbsolute: () ->
  moment(this.createdAt).format('YYYY-MM-DD HH:mm')

createdAtRelative: () ->
  moment(this.createdAt).fromNow()
```

- Get the questionnaire id for which to show lessons learned: `this.questionnaireId`.
- Use livequery to read from `LessonsLearned` collection.
- Note that the `this` scope for the helper functions `createdAt*` is a lesson learned document.
Create a new collection with `new Mongo.Collection 'lessons_learned'`. On the server, this sets up a MongoDB collection called my-collection; on the client, this creates a cache connected to the server collection.

Define conditions which allow insertion and removal of lessons learned documents.
# server-side only code
Meteor.publish 'lessons_learned', (questionnaireId) ->
    check questionnaireId, String
    LessonsLearned.find(
        {questionnaireId: questionnaireId, userId: this.userId},
        {sort: {createdAt: -1}}
    )

# server- and client-side code
Router.route '/questionnaires/:_id/passes', {
    name: 'questionnairePasses'
    waitOn: () -> [  
        Meteor.subscribe 'questionnaire', this.params._id
        Meteor.subscribe 'lessons_learned', this.params._id
        Meteor.subscribe 'questionnaire_passes', this.params._id
    ]
    data: () -> Questionnaires.findOne this.params._id
}
Show me some code!

Auto-publish: Turn it off!

from Discover Meteor
Show me some code!

Publish full collections

from Discover Meteor
Show me some code!

Publish partial collections

```javascript
Meteor.publish('somePosts', function() {
  return Posts.find({ author: 'Tom' });
});
```

*from Discover Meteor*
Show me some code!

Publish partial properties

```javascript
Meteor.publish('allPosts', function() {
  return Posts.find({}, {fields: {date: false}});
});
```

from Discover Meteor
Show me some code!

Using events for adding, deleting

```html
<template name="lessonsLearnedTemplate">
  <h2>{{_ 'lessonsLearnedTemplate.title'}}</h2>
  <!-- display code from previous slides -->
  <form id="addLessonLearned"> <!-- we left out some Bootstrap markup -->
    <input type="text" name="text">
  </form>
</template>

Template.lessonsLearnedTemplate.events
  'submit #addLessonLearned': (event) ->
    event.preventDefault()
    LessonsLearned.insert
      questionnaireId: this.questionnaireId
      userId: Meteor.userId()
      text: event.target.text.value
    event.target.text.value = ''

  'click .deleteLessonLearned': () ->
    LessonsLearned.remove this._id
```
Using reactivity

Making our relative time stamps react to language change:

```coffee
Template.lessonsLearnedTemplate.helpers
createdAtRelative: () ->
  share.languageDependency.depend()
  # make this computation dependent on a shared dependency
  moment(this.createdAt).format('YYYY-MM-DD HH:mm')

# client/startup.coffee
Meteor.startup () ->
  share.languageDependency = new Deps.Dependency

# client/templates/header.coffee
Template.header.events
  'click .tap-i18n-buttons button': () ->
    newLanguage = TAPi18n.getLanguage()
    accountsUIBootstrap3.setLanguage newLanguage
    moment.locale newLanguage
    share.languageDependency.changed()
  # forces re-computation of all dependent computations
```
Show me some code!

Did you notice? No callbacks!

We've implemented the "lessons learned" feature without any callbacks! In many cases, Meteor takes care of the asynchronicity and the callbacks necessary to deal with it.

If you need to explicitly call a server-side function, use `Meteor.call`. Using a `ReactiveVar`, it's easy to update the UI reactively.

In our project, we only used `Meteor.call` nine times, some due to Mini-MongoDb limitations (no aggregations).
Show me some code!

**Meteor.call, ReactiveVar example**

```javascript
_numericAnswersChartVar = new ReactiveVar null

Template.questionnaireQuestionView.helpers
    numericAnswersChart: () -> _numericAnswersChartVar.get()

Template.questionnaireQuestionView.onRendered () ->
    questionnaire = this.data
    Meteor.call 'questionnaireQuestionAnswers', Router.current().params._id,
        Router.current().params._questionUuid, (error, result) ->
            if error then return share.showErrorMessage error.reason
            questionnaire = share._question questionnaire
            if question.type is 'numeric'
                timeSeries = _.map result,
                    (answer) -> [answer.createdAt.getTime(), parseFloat(answer.value)]
                numericChart = _numericChart timeSeries.reverse(), questionnaire
                _numericAnswersChartVar.set numericChart # update ReactiveVar
```
Meteor

Fullstack JavaScript Development

- Retro42: Our prototype application
- Why did we choose Meteor?
- What is Meteor?
- Show me some code!
- Comparing Meteor vs. MEAN
- More about Meteor
Comparing Meteor vs. MEAN

All-inclusive vs. do-it-yourself

Meteor = MEAN stack + Socket.IO + grunt/gulp + Cordova + hot code reload + a lot more.

from wiki.dandascalescu.com/essays/meteor_js_vs_the_mean_stack,
see also wiki.dandascalescu.com/essays/why_meteor
Comparing Meteor vs. MEAN

What we liked better

- Less code, more functionality
- It took us much less preparation time to get productive with Meteor than with the MEAN stack
- It also felt like we were much more productive once we were up and running: we overestimated our effort by 50%
- Blaze easier to learn, whereas Angular can be overwhelming with all its concepts
- More straight-forward code because we rarely had to deal with keeping track of callbacks and/or promises.
Comparing Meteor vs. MEAN

What we liked better

- No time spent on build system (e.g. Grunt, Gulp)
- Simple dependency management, with versions always automatically fixed by Meteor
- Only one dependency management system, not two (i.e. npm, bower)
- Much less boiler-plate code to publish data to client
- Great and easy to add packages, e.g. to add authentication for platforms such as Google, Facebook, run `meteor add accounts-google`, then add `{loginButtons}` in your template to show the login UI, and you're done.
Comparing Meteor vs. MEAN

What we liked less

- Best practices on how to organize the code?
- Live reload in browser is slower because app must be assembled and precompiled by Meteor
- Testing framework Velocity was a late addition to Meteor, and there is little support from Meteor for writing tests
- For webtests, Velocity uses the Robot Framework which utilizes the keyword-driven testing approach: however, we would have preferred a more Protractor-like approach
Comparing Meteor vs. MEAN

What others have to say

"This isn’t your daddy’s JavaScript"

"Week 3: Tasks were being completed faster than ever before. We did struggle occasionally, but we were able to find plenty of documentation and support from our super smart lead developer who we call Google. At the end of one week, we had completed more in Meteor than the entire previous month in .NET."

see info.meteor.com/blog/from-.net-to-meteor-in-30-days (2015-09-09)
Meteor

Fullstack JavaScript Development

- Retro42: Our prototype application
- Why did we choose Meteor?
- What is Meteor?
- Show me some code!
- Comparing Meteor vs. MEAN
- More about Meteor
More about Meteor

Meteor Roadmap

- Meteor 1.2, Summer 2015
  - ES2015 (ES6)
  - 1st class Angular and React support
- Future Directions
  - Full-stack reactive SQL
  - REST and microservices
  - large app patterns
  - ES2015 modules, and more
  - Galaxy: managed production-quality "meteor deploy"

see info.meteor.com/blog/whats-coming-in-meteor-12-and-beyond
More about Meteor

Who is behind Meteor?

- Active development by full-time employees
- Revenue product: Galaxy, a high-availability / large scale Meteor hosting platform
- Meteor platform is MIT-licensed: host anywhere; you own the code

from meteor.hackpad.com/Meteor-speaker-kit-uaPe3zDDH8z
More about Meteor

Learn Meteor

see discovermeteor.com, guide.meteor.com, and meteor.com
Conclusion

What's great: Productivity

Initial backlog

- At schedules intervals, send reminder mails
- Allow user to schedule his questionnaires
- When answering questions, show previous answers to the current question
- Allow yes/no questions
- For numeric questions, show graph of all answers to this question
- Show all answers for a selected question
- Show questions & answers for a selected session
- Show list of sessions for a selected questionnaire
- Answer questions for a selected questionnaire
- Configure questionnaire for current user
- Show questions (question-text only) for a selected questionnaire
- Show list of all questionnaires (user-independent) after login
- Simple application with Google auth
- Deployment Environment Setup
- VM Setup including Development Environment

Additional backlog (extract!)

- Intro page for non-logged in users for email UI
- Questionnaire Passes View: percentage answered groups
- Group Question View groups
- Group Pass View groups
- Add group users to questionnaire groups
- Questionnaire passes: Show some boolean and numeric answers in overview
difficult duplicate UI
- Display questionnaire passes in a table
- Markdown Support for Text Questions
- Add demo data to fixture for email
- Create questionnaire from scratch
- Show user notification on missed questionnaire passes
- Allow user to delete questionnaire (and its passes)
- Allow user to edit questionnaire
- Enhance list of sessions for a selected questionnaire with scheduling info

Meteor has been called "embarrassingly easy-to-learn": We definitively agree.
Conclusion

Should be improved: Test support
Conclusion

Should be improved: Test support

What we missed most about Meteor was in-built support for unit and integration tests, client- and server-side.

Also, we would like to see alternatives to the Robot framework for web tests.
Conclusion

On a tech radar, give it a "trial"
Any Questions?
Backup Slides
What about Performance?

Comparing Blaze, React Meteor, Angular-Meteor, Angular 2 with Meteor, with "Waldo Finder".

Backup Slides

What about Performance?

Backup Slides

What about Performance?

What about Performance?


- Static type systems help use a new set of classes – an effective form of self-documentation
- We believe the most important result is that the static type systems showed a clear tendency in class identification tasks, and that we found a first indicator that this is caused by a reduced navigation effort.
- Static type systems make it easier for humans and reduce the effort to fix type errors.
- For fixing semantic errors, we observed no difference with respect to development times, and static type systems may not be helpful in preventing semantic errors.
Isogaba maware: *When you are in hurry, take your time*

- Testing: Automated tests at all levels
- SOLID design, and remember the Law of Demeter
- Dependencies: Use a consistent namespaces scheme
- Documentation: Document your APIs
- Continuous Integration, ...