Meteor

- Javascript Framework für
- Realtime…
- Single-Page Apps…
- “Simplicity”
- Sehr neu (Vorstellung: 11. April 2012)

Inhalt

1. (Langer) Exkurs: Simple vs Easy, oder: “Warum ich mich überhaupt mit Software-Architektur beschäftige”
2. Meteor – technische Seite
3. Diskussion: Macht das alles Sinn?

Simple vs. Easy

Rich Hickey: “Simplicity matters” / “Simple made easy”

- http://www.youtube.com/watch?v=rI8tNMsozo0 / http://www.infoq.com/presentations/Simple-Made-Easy
- Rich Hickey hat Closure entwickelt. Philosoph, Mathematiker

“Easy”

- Naheliegend, bereits bekannt (“familiar”)
- Capability/Skill
- Gegenteil: Schwer, schwierig
- Bsp: “Schnell installiert”
- Relativ, subjektiv

“Simple”

- Ein-fach (Flechten: Sim-Plex)
- Gegenteil: Komplex
- “Simple” ist oft gar nicht “Easy” zu erreichen

- Elephant: Bisher geschriebener Code dominiert Projekt
- => Agilität durch (architekturelle) Simplizität
- => Simple Komponenten
- => Substitute, run on different box, Re-Combine, Re-Purpose

- Rich Hickey: “Good Design is about taking things apart”
- Rich Hickey: “We can be creating the exact same programs out of significantly simpler components”

- Edsger W. Dijkstra: “Simplicity is the prerequisite for reliablity”
- Antoine de Saint-Exupery: “*Perfection is achieved* perfection not when *there is nothing* left to add, but when *there is nothing left to take away*”
- Leonardo daVinci: “Simplicity is the ultimate sophistication.”

Was heißt das praktisch?

- Datenstrukturen: Hash Maps statt Objekte für alles (vgl. Interfaces / APIs von Servern)
- Values statt Objects
- Functions/Namespaces statt Methods
- Set functions statt loops
- Queues statt actions
- Declarative Data Manipulation statt ORM
- Weniger Order! (Ordnung, Reihenfolge, Sortierung)
    - Sets statt Listen
    - Named arguments / Map statt Positional Arguments
    - Data statt Syntax

    - Deklarativ statt imperativ
    - Queues statt call chains
    - Frage: “Can you move it?” Daten statt Klassen mit Verben die von Services konsumiert werden
    - <=> Warum macht man andere Interfaces wenn man verteilte Anwendungen programmiert?
- Was fällt dem WSA noch ein?
    - Diskussion: “Complexity is easy to get: Gem install hairball”

Bret Victor

- “Easy” geht aber auch noch besser
- “Inventing on Principle” http://vimeo.com/36579366
- http://worrydream.com/LearnableProgramming/

Meteor

- “The Big Plan”: http://www.meteor.com/about/mission
    - Die ersten beiden Paragraphen
    - Die Leute

- Recap: Client-Side Javascript, Single-Page WebApps
  - **Slides (Architektur.odp) oder Whiteboard?**
- The key pieces of Meteor could be built yourself using these pieces: http://stackoverflow.com/a/10623051/1181063

Slide: Meteor 7 Principles

- http://docs.meteor.com/#sevenprinciples

Data on the Wire

- DDP
- vgl. Comet, aber mit WebSockets: Packete, Push
- “Kein HTML over the network”: Fallback für Non-JS Clients geplant (auch wegen Crawlern)

One language

- JS auf Client und Server, Dokumentdatenbank (kein SQL, was eine weitere Sprache wäre)

Database everywhere

- Aktuell MongoDB API
- “Minimongo” auf Client-Seite
  - Nicht sehr weit, “Skip” und “Limit” gehen zB noch nicht

- Ist das **“Smart” oder “Clever” ? Exkurs!**
- Hilft Code nicht mehrmals schreiben zu müssen
- Ich hab ein komisches Gefühl: Ungewohnt? Oder objektiv schlecht? Was denkt der WSA?
- Joel Spolsky: “Law of leaky abstractions”
  - Diskutieren!
  - Beispiele: Autofahren im Regen, TCP/IP, C++ String-Klasse

- Joel Spolsky: “Abstractions save us time working, but they don't save us time learning.”
  - Abstraktionen helfen uns, immer komplexere Dinge zu erschaffen (GUI, network programming)
  - Aber eines Tages gibt es ein Problem mit der Abstraktion und man braucht zwei Wochen um den Fehler zu finden

Latency Compensation

- View zuerst aktualiseren
- Gleichzeitig Daten zum Server schicken
- Wenn der Server einen Konflikt erkennt, rück-abwickelt der die Änderung (schickt die Daten zum Client, der macht ein Update)

Full Stack Reactivity

- “Reactive Programming”
  - Excel-Beispiel
  - Datenzentrizität!
  - “Write your code in a simple, imperative style, and the result will be automatically recalculated whenever data changes.”
  - Data-Flow Programming
    - Evtl. Exkurs: Tolle Abstraktion!
    - Bsp: Unix-Pipelines, PD (Pure Data), Click Modular Router
- Make realtime the default
  - Reaktivität nicht nur in der GUI, sondern über Server zu anderen Clients hinweg
- Viel viel Observer-Pattern / Event-Driven
- (Meteor.autosubscribe() und Meteor.deps() erklären?)

Embrace the Ecosystem

- Open Source, MIT Lizenz

Simplicity Equals Productivity

- So simpel, wie die tun, wirkt es auf mich erst mal nicht
- “So it's actually rather straight forward,” http://stackoverflow.com/a/10293760/1181063

It's Demozeit

- Hammertime.

Smart vs. Clever

- Website: http://books.google.com/ngrams/graph?content=clever,smart&year_start=1800&year_end=2000&corpus=0&smoothing=3
  - via http://english.stackexchange.com/a/46460
    - “Although there is a significant difference between BrE and AmE I believe that in recent years the meanings are getting closer to each other and this particularly due to many new clever gadgets and especially smart devices.”

Smart	Clever
Von Sharp: (Scharfsinnig?) -> wit (Geist, Verstand)	von: handy (praktisch, geschickt), dexterous (fingerfertig, geschickt)
educated, trained,	does not always implies

experienced, seasoned, implies intelligence	intelligence
"high-tech": great deal of time, effort, and money	good, cheap, and simple solution
Conventional wisdom often produces conventional results	Clever takes smart a step farther, adding insight and a dash of the unexpected.
	Necessity is the mother of cleverness
smart devices	clever gadgets
Für mich: ausschließlich positiv (außer wie in “Smart ass”)	Für mich: Negative Konnotation (Ein krasser Hack, eine “Kludge”, eine Abkürzung ist für mich Clever, nicht Smart)