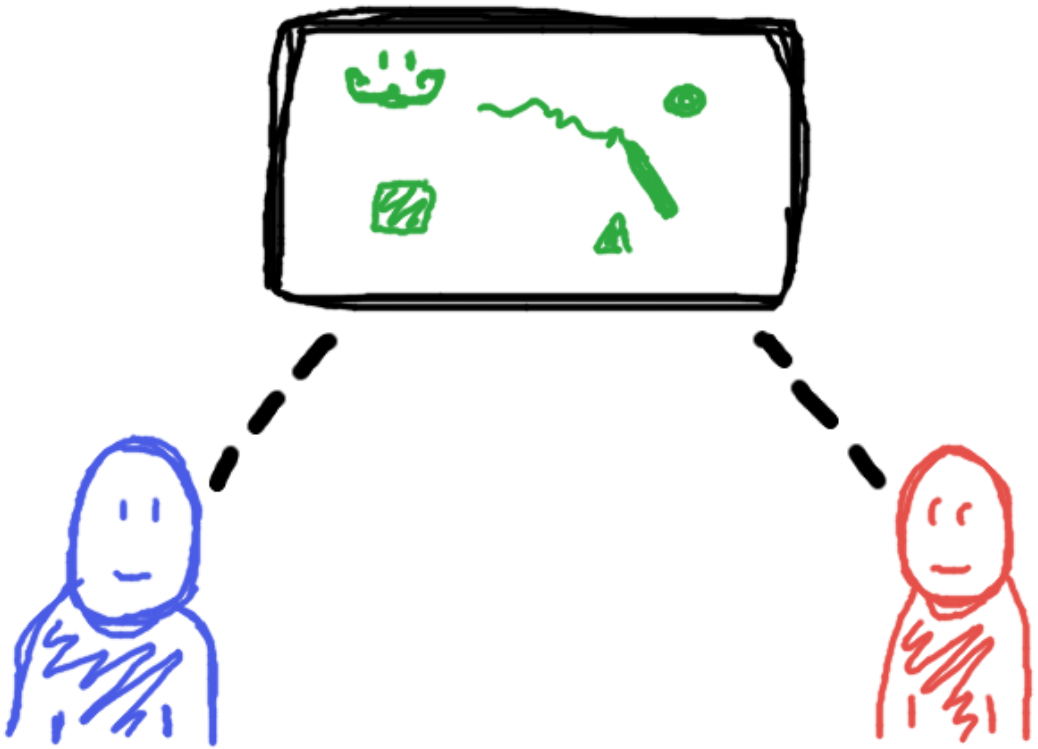
Bachelorarbeit

Browserbasiertes, kollaboratives Whiteboard

Inhalt

1. Motivation
2. Analyse
3. Design
4. Evaluation
5. Fazit

Motivation



Zusammenarbeit ermöglichen, auch wenn ein persönliches Treffen nicht möglich ist, weil ...

- Verschiedene Arbeitsplätze
- Gerade unterwegs
- Temporär immobilisiert

Analyse

- Es ist ein Whiteboard

- Geometrische Elemente: Rechtecke, Ellipsen
- Text schreiben
- Grafiken einfügen, Freihand zeichnen

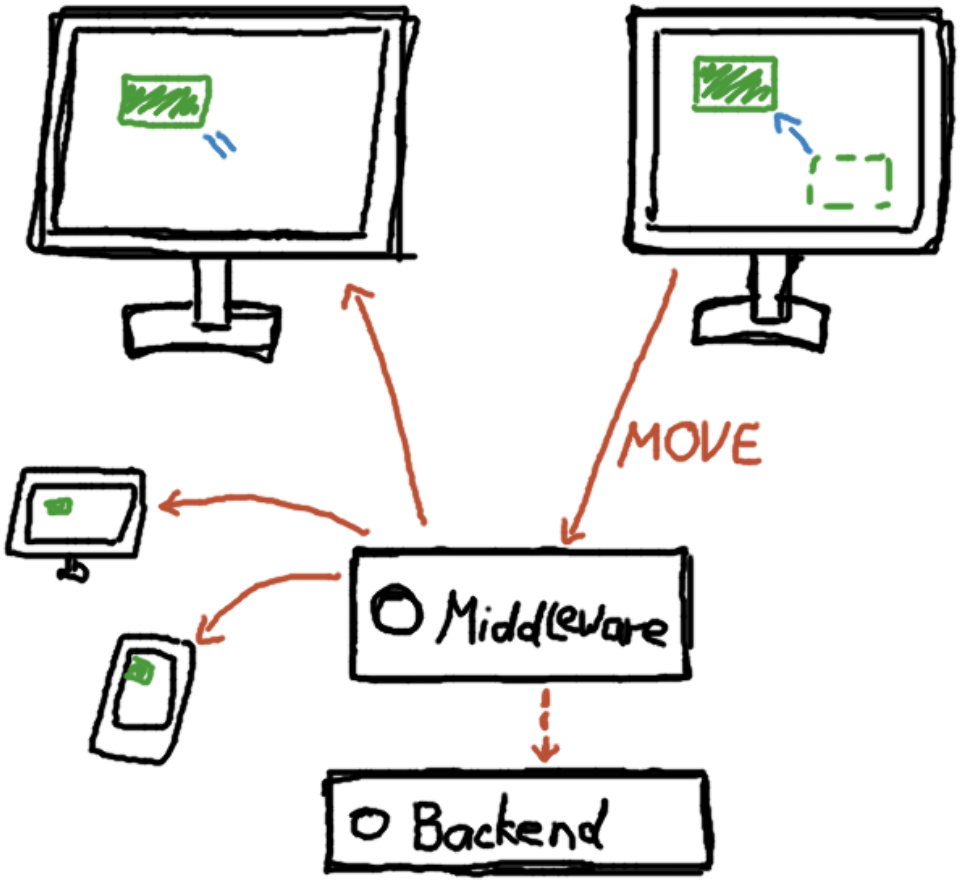
Analyse

- Es ist ein Whiteboard

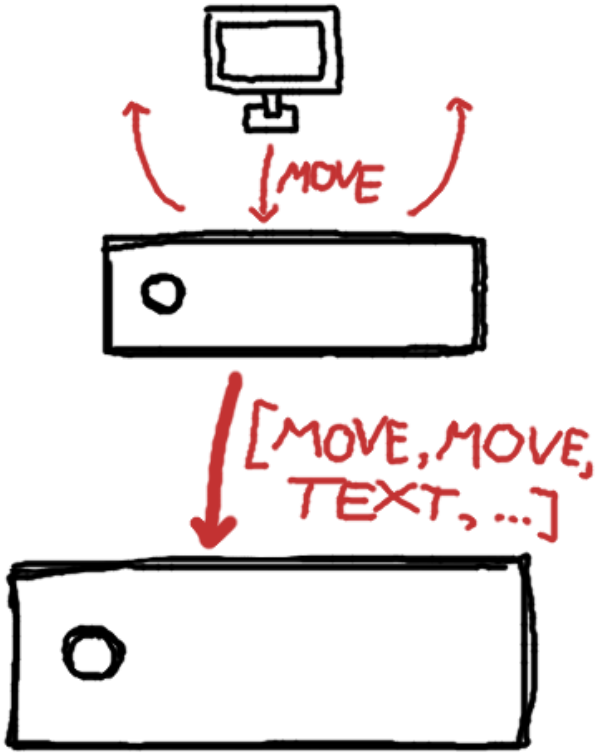
- Jeder soll das Gleiche sehen

- Geometrische Elemente: Rechtecke, Ellipsen
- Text schreiben
- Grafiken einfügen, Freihand zeichnen
- Echtzeitkommunikation zwischen Clients

Kommunikation



- Schnell unter Clients



- Langsam zum Backend

Analyse

- Es ist ein Whiteboard

- Geometrische Elemente: Rechtecke, Ellipsen
- Text schreiben
- Grafiken einfügen, Freihand zeichnen

- Jeder soll das Gleiche sehen

- Echtzeitkommunikation zwischen Clients

- Es gibt mobile Teilnehmer

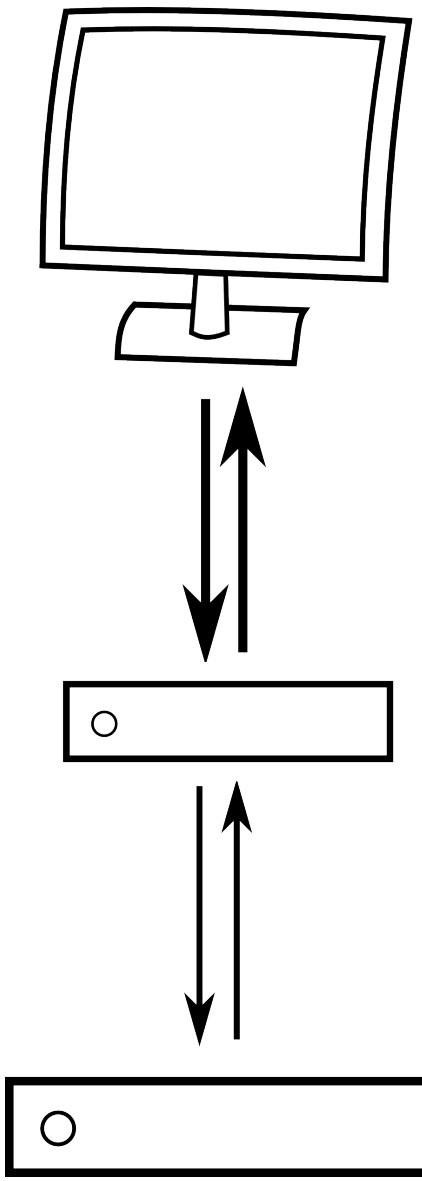
- GUI für Bedienung mit Finger und kleine Displays

Analyse

- Es ist ein Whiteboard
- Jeder soll das Gleiche sehen
- Es gibt mobile Teilnehmer
- Dient der Verständigung

- Geometrische Elemente: Rechtecke, Ellipsen
- Text schreiben
- Grafiken einfügen, Freihand zeichnen
- Echtzeitkommunikation zwischen Clients
- GUI für Bedienung mit Finger und kleine Displays
- Export und Import
- Observer-Logik für Ereignisse

Design



Browser
(Client)

- Darstellung
- Interaktion

Middleware

- Echtzeitdatenaustausch

Backend

- Anwendungslogik
- Persistenz

Kommunikation

Befehl	Bedeutung
JOIN	Beitreten
LEAVE	Verlassen
CREATE	Erstelle Element
DELETE	Entferne
MOVE	Verschiebe
CHANGE	Verändere Attribut
TEXT	Füge Text ein
…	…

Client

<canvas>

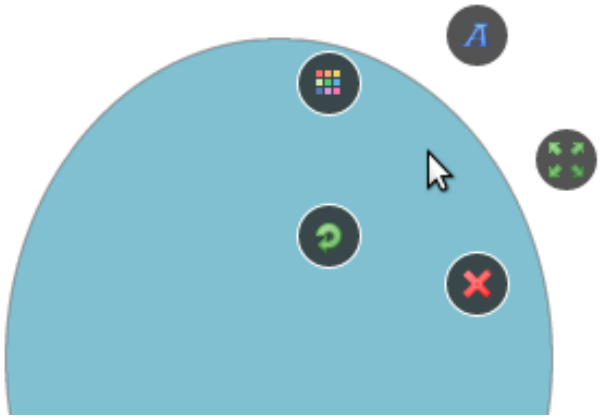
“The canvas element provides scripts with a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, or other visual images on the fly.”

- JavaScript
- Zeichnen von geometrischen Formen, Einfügen von Bildern
- Auf Gezeichnetes lässt sich nicht als Objekt zugreifen
  - Eigene Logik benötigt
- Animationen durch schnelles Neuzeichnen

http://www.w3.org/TR/html5/the-canvas-element.html#the-canvas-element

Anpassungen für mobile Geräte

- Große Finger, kleine Fläche
- Nicht die gleiche Tastatur
  - Keine Pfeiltasten
- Nicht die gleichen JavaScript-Events

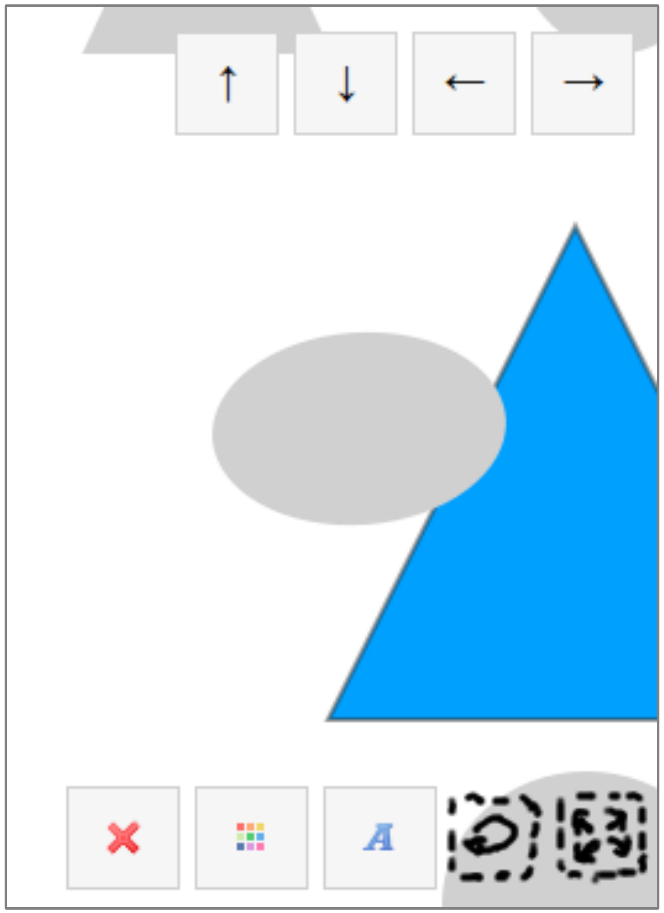


```
canvasContainer.addEventListener(
    "mousedown", getElement, false
);
canvasContainer.addEventListener(
    "touchstart", getElement, false
);
```

Anpassungen für mobile Geräte

- Große Finger, kleine Fläche
- Nicht die gleiche Tastatur
  - Keine Pfeiltasten
- Nicht die gleichen JavaScript-Events

```
canvasContainer.addEventListener(
    "mousedown", getElement, false
);
canvasContainer.addEventListener(
    "touchstart", getElement, false
);
```



Middleware

Echtzeitkommunikation

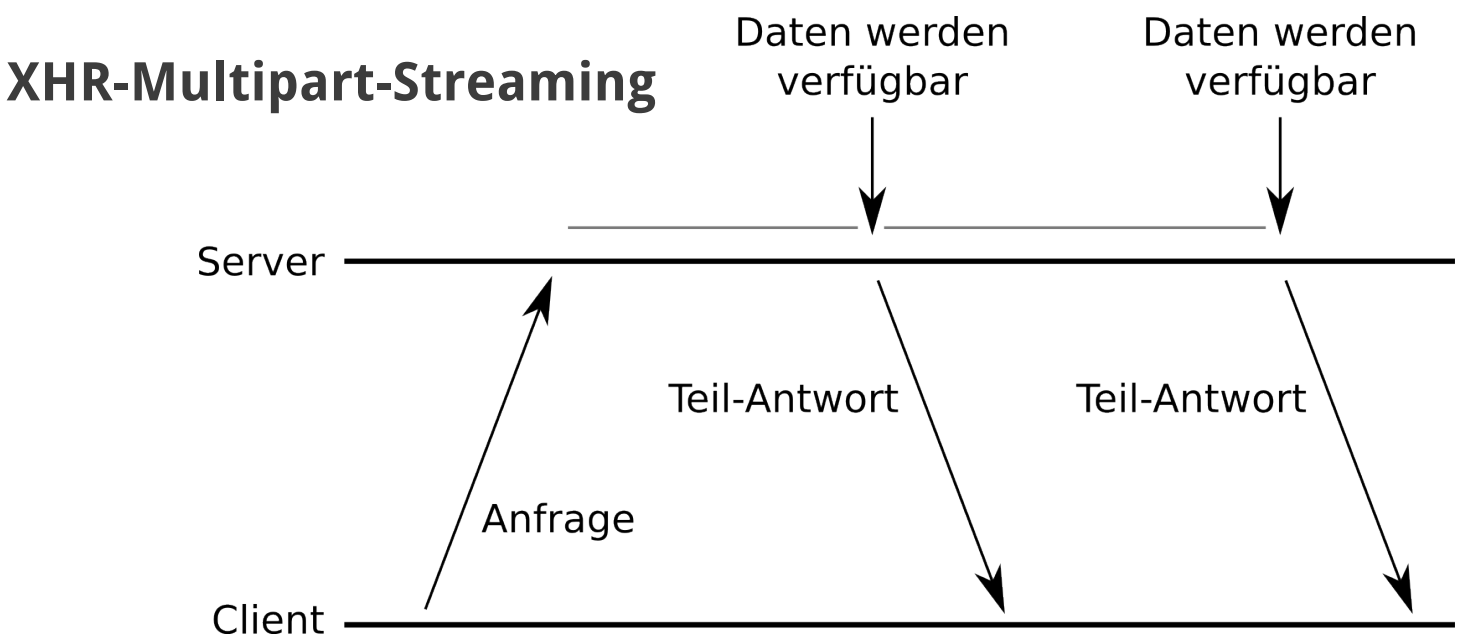
Ajax Push Engine-Projekt (APE)

“APE is a full-featured OpenSource solution designed for Ajax Push. It includes a web-server and a Javascript Framework. APE allows to implement any kind of real-time data streaming to a web browser, without having to install anything on the client-side.”

- Eigener Server (Port 6969), der Daten zwischen den Clients austauscht
- Verschiedene Techniken implementiert
- JavaScript in Client und Server

http://www.ape-project.org/

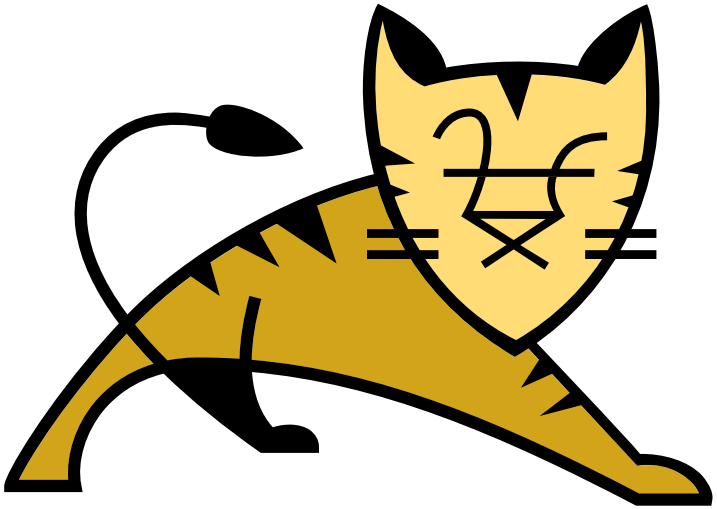
Middleware



1. Anfrage schicken
2. Erhält „Teil“-Antwort
3. Erhält „Teil“-Antwort
4. Erhält „Teil“-Antwort

- Nur noch wenige Anfragen
- Geleg. Verbindungs-Reset
  - Speicher freigeben

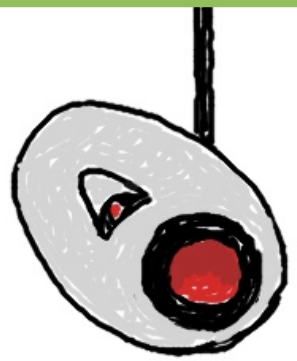
Backend



Aufgaben

- Anwendungslogik
- Persistenz: Halten von Zuständen

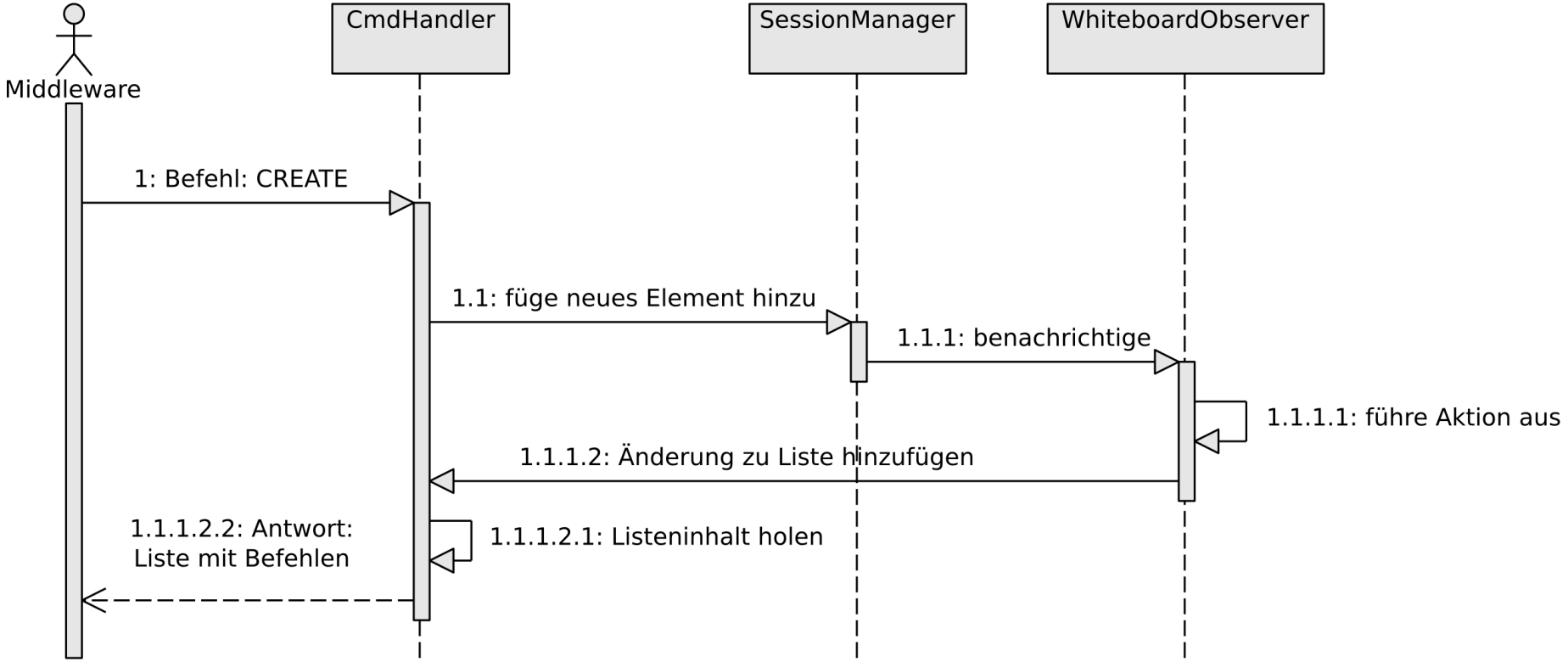
- Persistenz: Whiteboard und Elemente
- Observer: Reagieren auf Events
- Export/Import: Schnittstelle, XML-Implementierung
- Java EE (Java Server Pages, Java Servlets)



Observer

1. Eigene Klasse **erbt** von `WhiteboardObserver`
   - Implementieren von Methoden
2. Instanz **registriert** sich auf bestimmte Ereignisse
   - z.B. Element erstellt/verändert/entfernt
3. Änderungen werden Middleware bei nächster Gelegenheit **mitgeteilt**
   - Middleware leitet weiter an Clients

Observer: Ablauf



Evaluation

Profile fürs Senden von MOVE und TEXT

- Langsam: 1000, 3000 ms
- Normal: 150, 600 ms
- Schnell: 20, 60 ms

Verschickte Datenmengen

- MOVE-Größe: 558 Byte (inkl. HTTP-Header)
  - bei 4 Clients, 150 ms:
    - 14.5 kb/s, 25.5 MB in 30 min.

Server-CPU-Auslastung

- i3-Prozessor, 4 Clients, 150 ms
  - Auslastung nur minimal zwischen 1-3%

Fazit

- Mit aktuellen Web-Standards (HTML5) ist es möglich Rich Internet Applications zu bauen, die in Echtzeit ablaufen
  - keine Plugins mehr notwendig

Fazit

- Mit aktuellen Web-Standards (HTML5) ist es möglich Rich Internet Applications zu bauen, die in Echtzeit ablaufen
  - keine Plugins mehr notwendig

- Denkbare Erweiterung:
  - Backend gibt Middleware Filter-Anweisungen
  - „Weg“ vom Whiteboard und hin zu einem Framework, mit dem sich auch andere Anwendungen erstellen lassen