5.– 8. September 2011
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Good by Server...
Welcome Client!

SOFEA - ein leichtgewichtiger Architekturansatz

Sandro Sonntag

Adorsys GmbH & Co. KG

Thin Server Architektur

Web API‘s

Entwicklung des Webs

REST

Inhalt

Web-Technologien

Potenziale

Was hat HTML5
mit Architektur zu tun?

Was ist falsch an Rich
Server?

Best Practices

Über mich

Sandro Sonntag

- Java Eetwickler - 10 Jahre Java Erfahrung in Enterprise Umfeld
- Liebe zu Webtechnologien und Webscale Themen
  - Wie REST, JavaScript/Node.JS, Mongo, Couch, Appengine
- Technical Lead bei Adorsys
- Unterwegs als Berater und Softwarearchitekt (CPSA)
- https://www.xing.com/profile/Sandro_Sonntag

Ursprünge SOFEA

Ursprünge SOFEA

- **2007** Publikation des Papiers **„Live above the Service Tier“**
  - „How to Build Application Front-ends in a Service-Oriented World“
  - durch Ganesh Prasad, Rajat Taneja, Vikrant Todankar

Ursprünge SOFEA

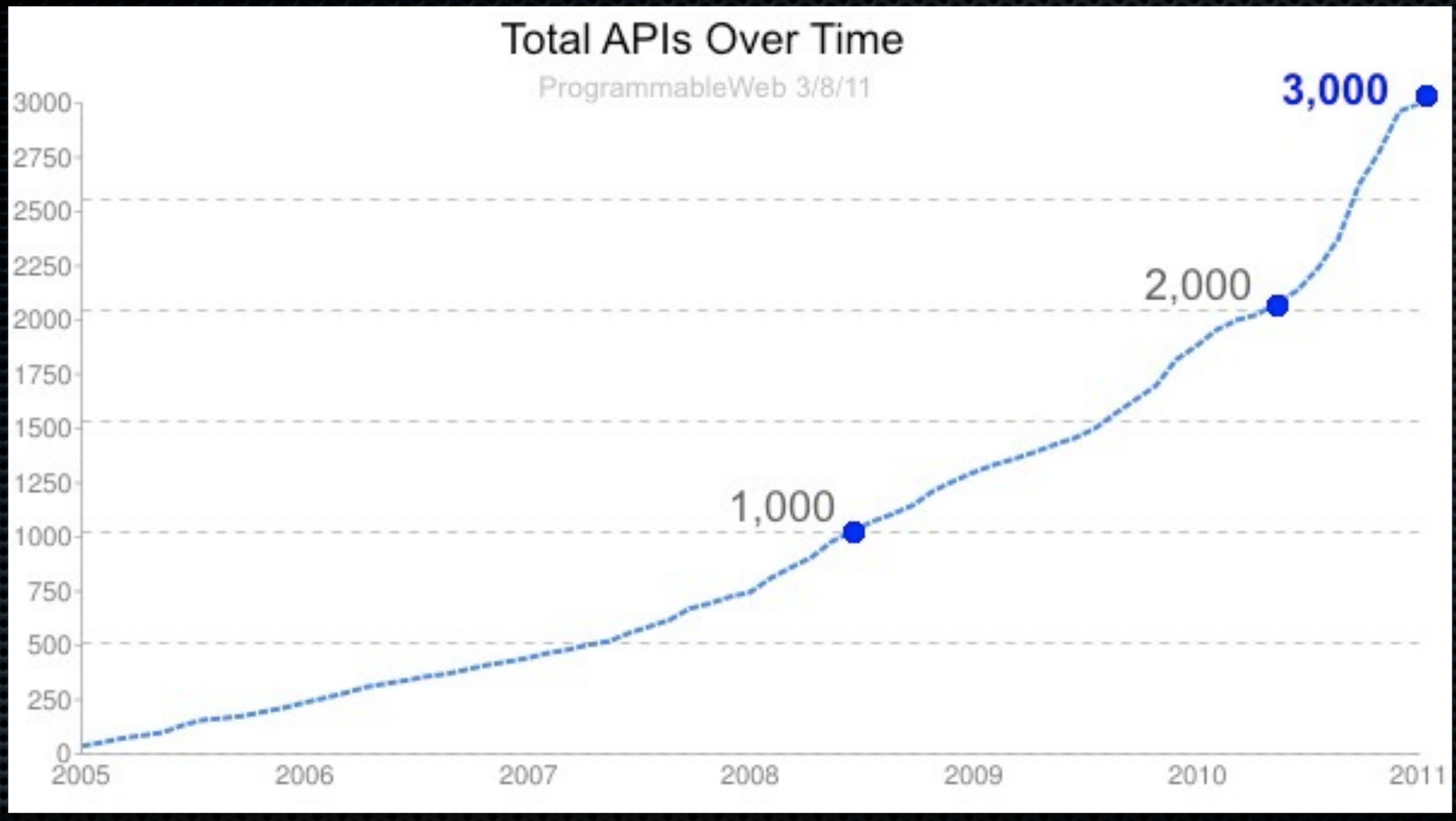
- **2007** Publikation des Papiers **„Live above the Service Tier“**
  - „How to Build Application Front-ends in a Service-Oriented World“
    - durch Ganesh Prasad, Rajat Taneja, Vikrant Todankar
- Parallel SOUI Publikation - **Service Oriented User Interface**
  - „MVC is Dead“ Norlan Wright

Ursprünge SOFEA

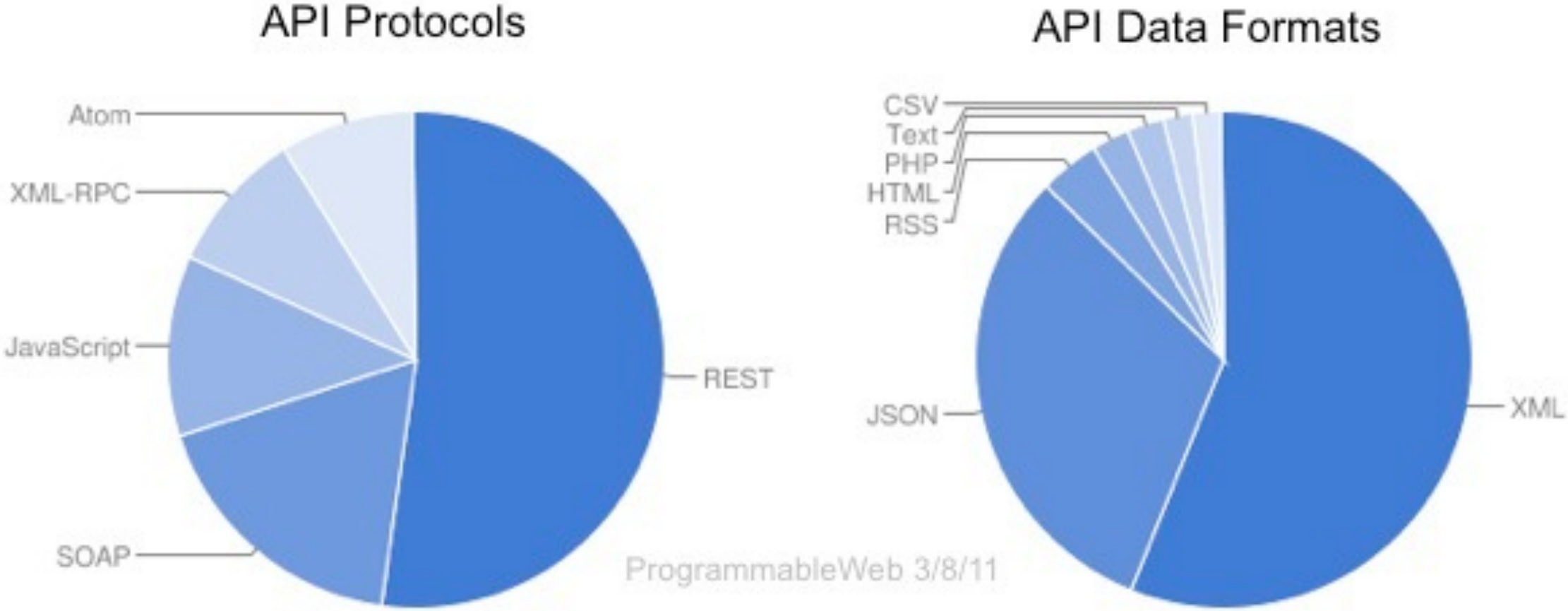
- **2007** Publikation des Papiers **„Live above the Service Tier“**
  - „How to Build Application Front-ends in a Service-Oriented World“
    - durch Ganesh Prasad, Rajat Taneja, Vikrant Todankar
  - Parallel SOUI Publikation - **Service Oriented User Interface**
    - „MVC is Dead“ Norlan Wright
  - Andere geläufige Bezeichnung: **„Thin Server Architecture“**

einige Trends

Jobtrends - HTML5 und Co.

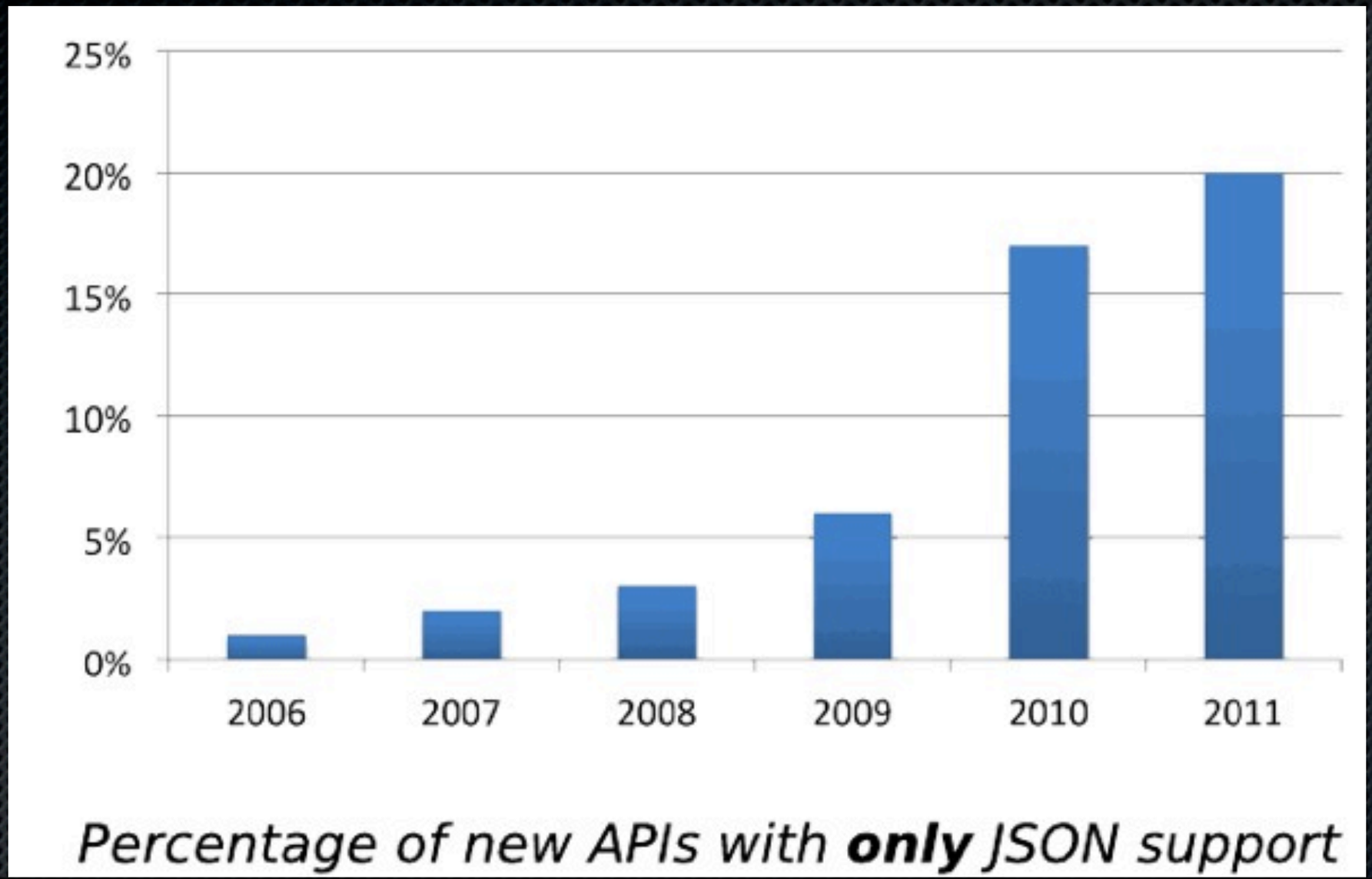


1000 APIs in 9 Monaten



Schnittstellen

REST und JSON auf dem Vormarsch



1 in 5 APIs Say “Bye XML”

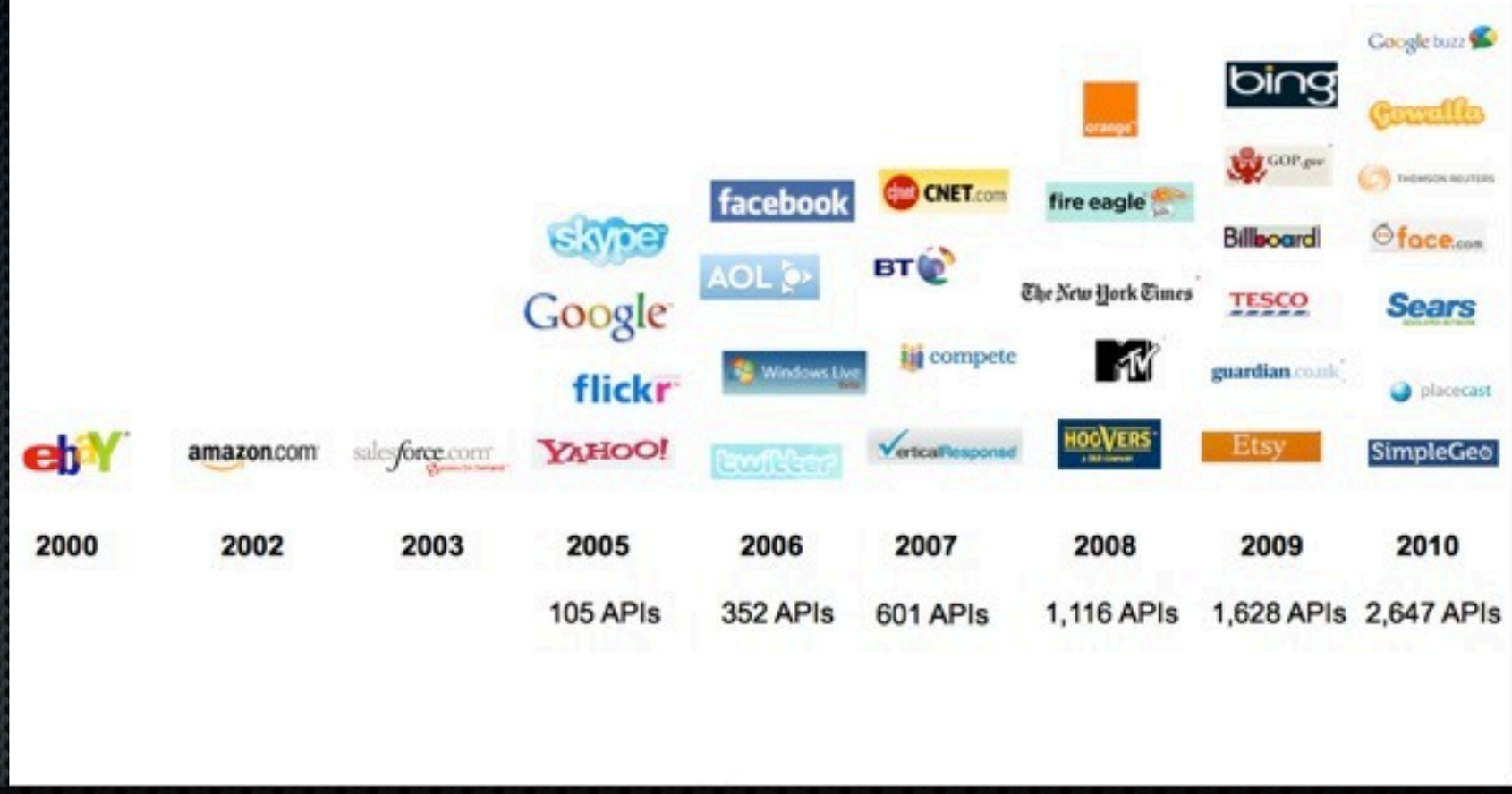
API Billionaires Club, 2011 edition

twitter	13 billion API calls / day *(May 2011)*
Google	5 billion API calls / day *(April 2010)*
facebook	5 billion API calls / day *(October 2009)*
NETFLIX	10 billion API calls / month *(January 2011)*
ebay	8 billion API calls / month *(Q3 2009)*
bing	3 billion API calls / month *(March 2009)*
npr	1.1 billion API-delivered stories / month *(March 2010)*
salesforce.com	Over 50% of all traffic via API *(March 2008)*
amazon web services	Over 260 billion objects stored in S3 *(January 2011)*

13 milliarden API calls / tag

Salesforce 50 % des Traffics durch APIs

Open API timeline



Anzahl der open API‘s wächst

Twitter Saw 600K Signups Yesterday, It Took More Than 16 Months To Reach Its First 600K

Gartner

In the next-generation client/server architecture, the **"client" is a rich application running on an Internet-connected device**, and the "**server**" is a set of application services hosted in an increasingly elastically **scalable cloud computing** platform.

Chrome OS

Der Browser als Betriebssystem

Aber was ist der Auslöser?

Mobile, Smart Pads, iTV, Social Media, Desktop Widgets and smarter Webapps

Und andere neue Technologien

Google maps
Deutschland

Was bedeutet dies für Webarchitekturen?

Services zum Server!
Präsentation zum Client!

aus „Thin Client“ wird „Thin Server“

Zurück zu den Anfängen

Es war einmal...

Aus Fat-Clients werden Thin-Clients

Thin Client Architekturen

Daten und Repräsentation sind vermischt

Browser

Server

Daten und Repräsentation sind vermischt

Name	Foo
Nachname	Bar
Str	Laufer-Str 99

➡Datenstrukturen?
➡Datentypen?

Browser

Server

Daten und Repräsentation sind vermischt

Name	Foo
Nachname	Bar
Str	Laufer-Str 99

➡Datenstrukturen?
➡Datentypen?

Browser

Server

```
<html>
   <body>
      <span>Name:  Foo</span>
```

➡Daten und Repäsentation gemischt

Daten und Repräsentation sind vermischt

Name	Foo
Nachname	Bar
Str	Laufer-Str 99

➡Datenstrukturen?
➡Datentypen?

Browser

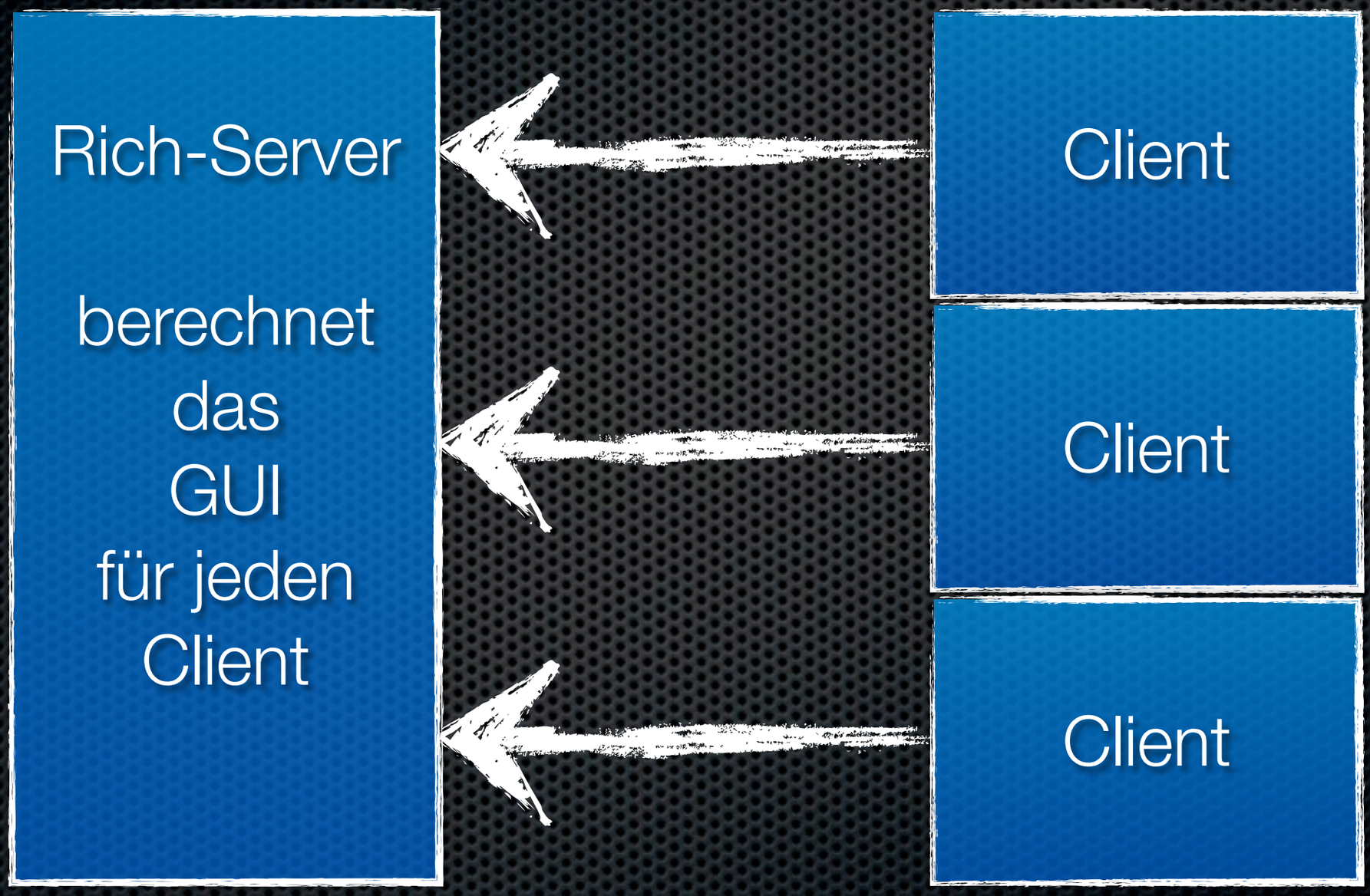
Server

➡Präsentationsflow und Datenaustausch gemischt

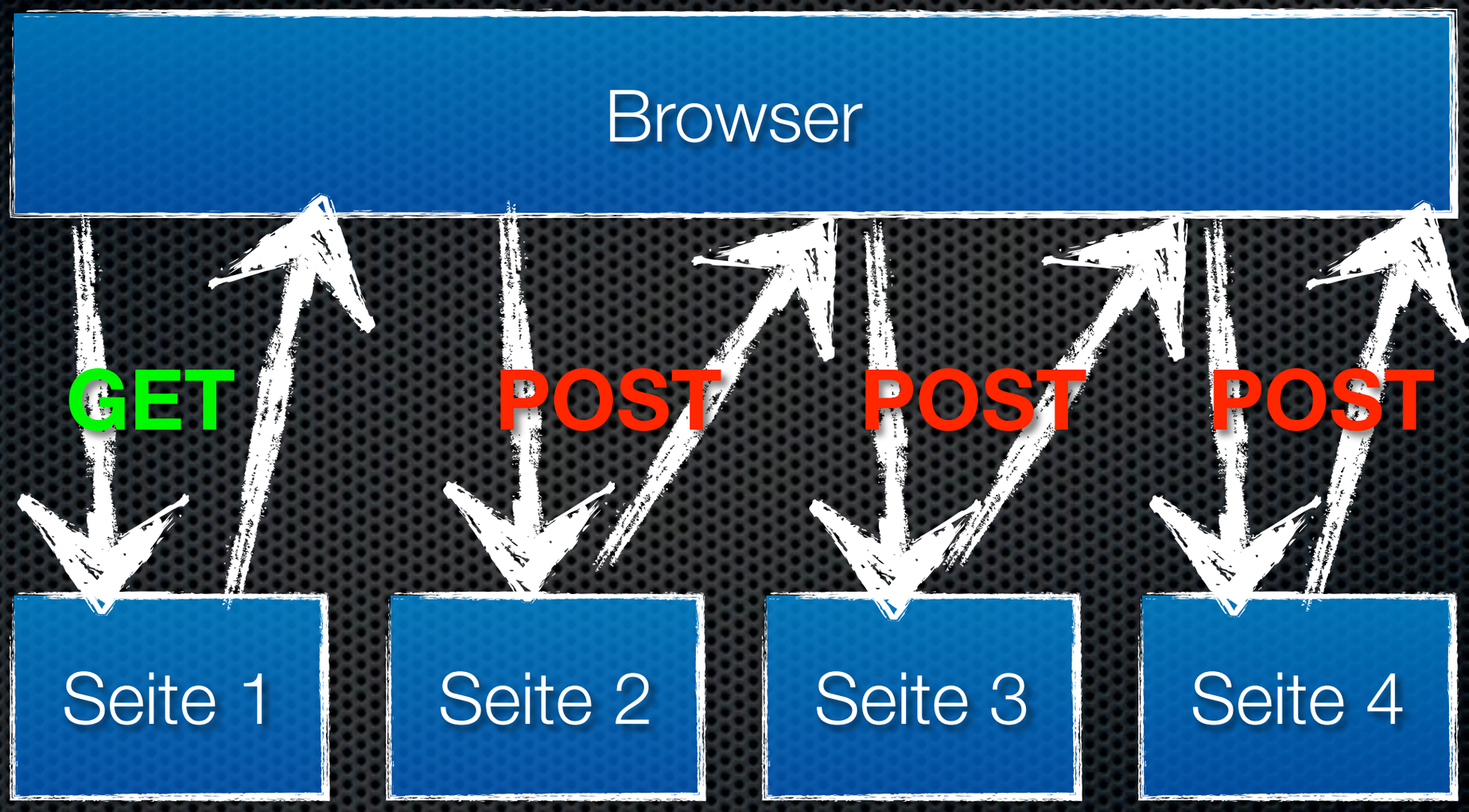
```
<span>Name: Foo</span>
```

➡Daten und Repäsentation gemischt

Schlechte Verteilung der Arbeit



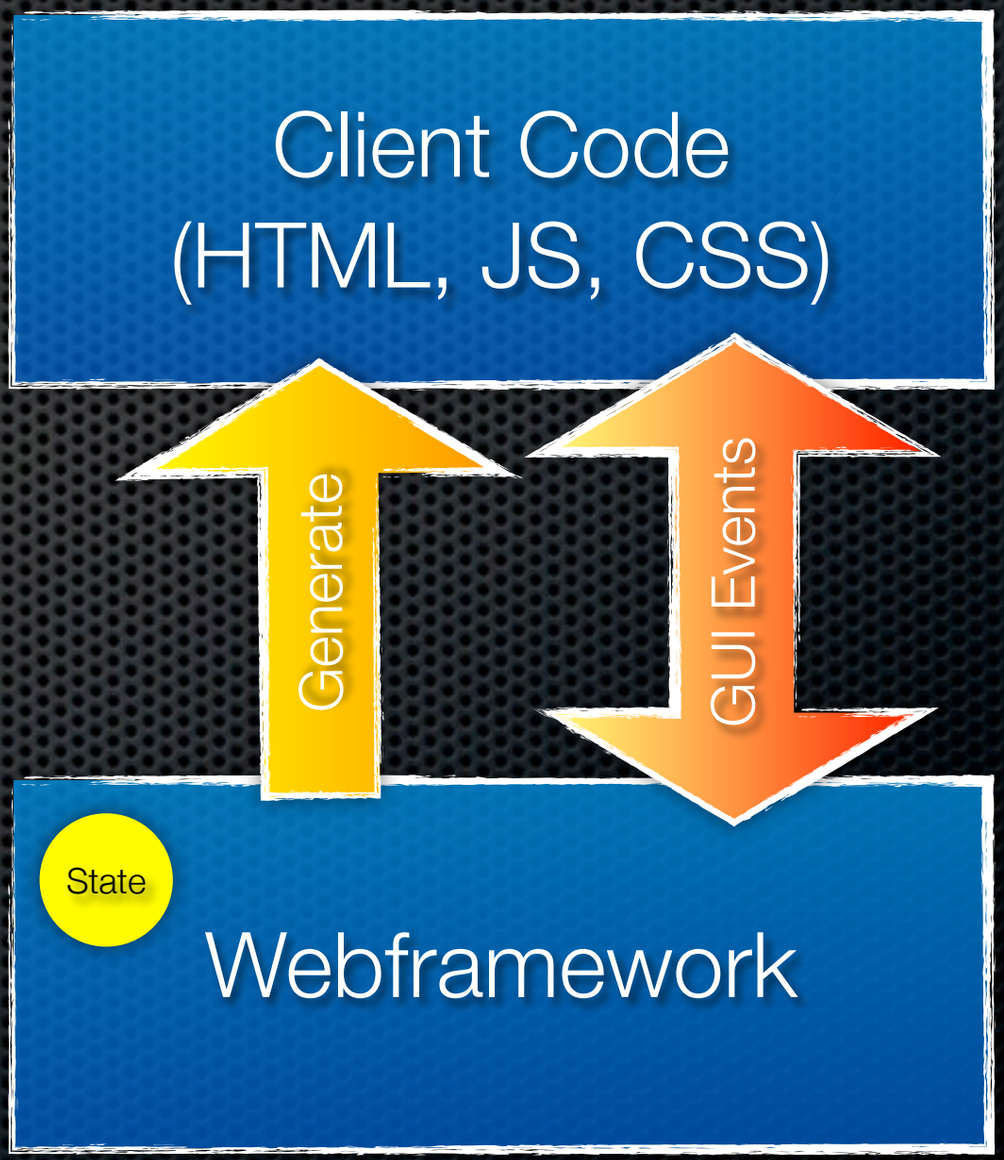
Hohe Latenz



Usability erfordert eine flüssige Bedienung.

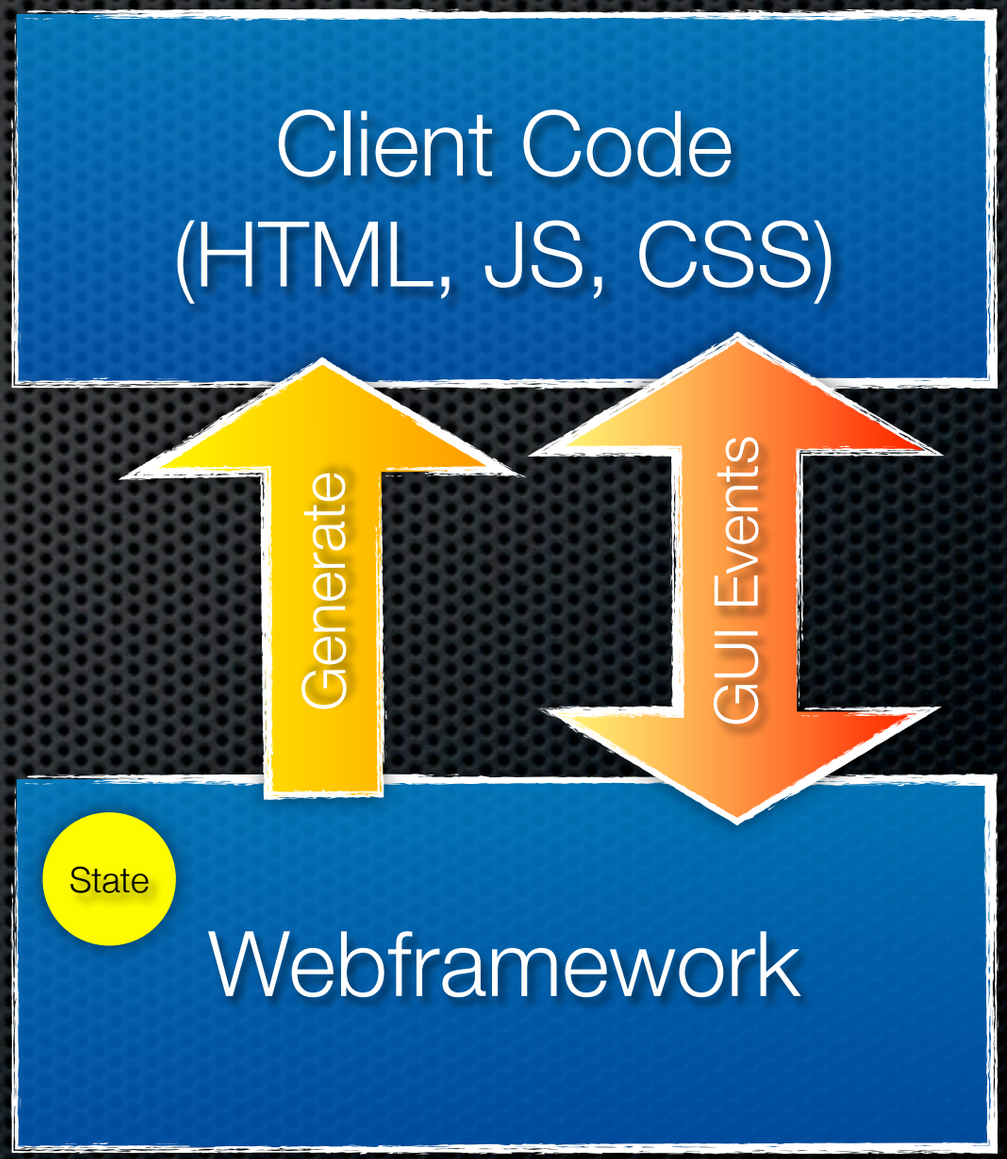
Herkömmliche Webapps reagieren zu langsam auf Benutzer Interaktionen.

Komplexität



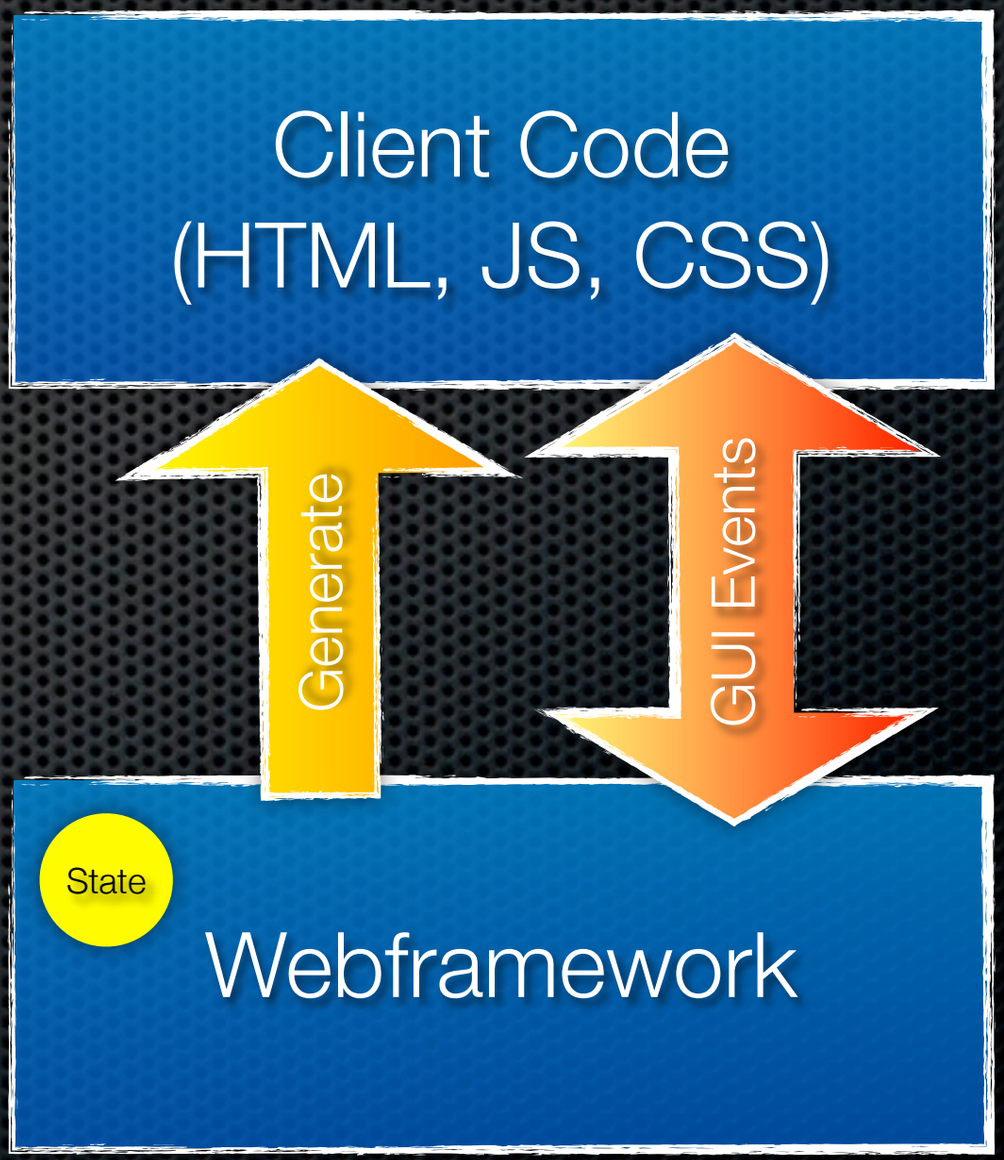
Komplexität

- Präsentationslogik ist am Client und Server



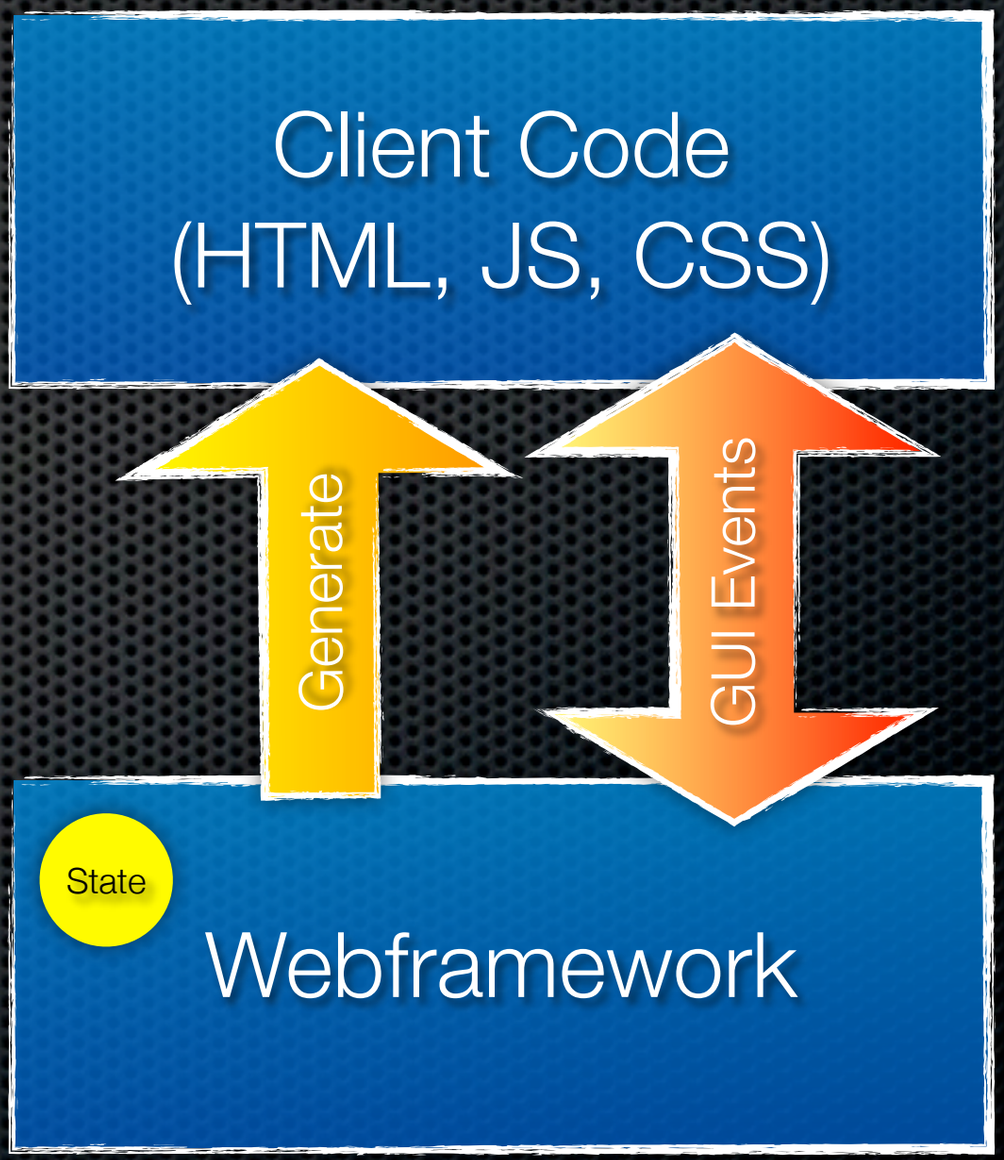
Komplexität

- Präsentationslogik ist am Client und Server
- Webframework ein Codegenerator für HTML & JS(denken in 2 Dimensionen - sehr umständlich)



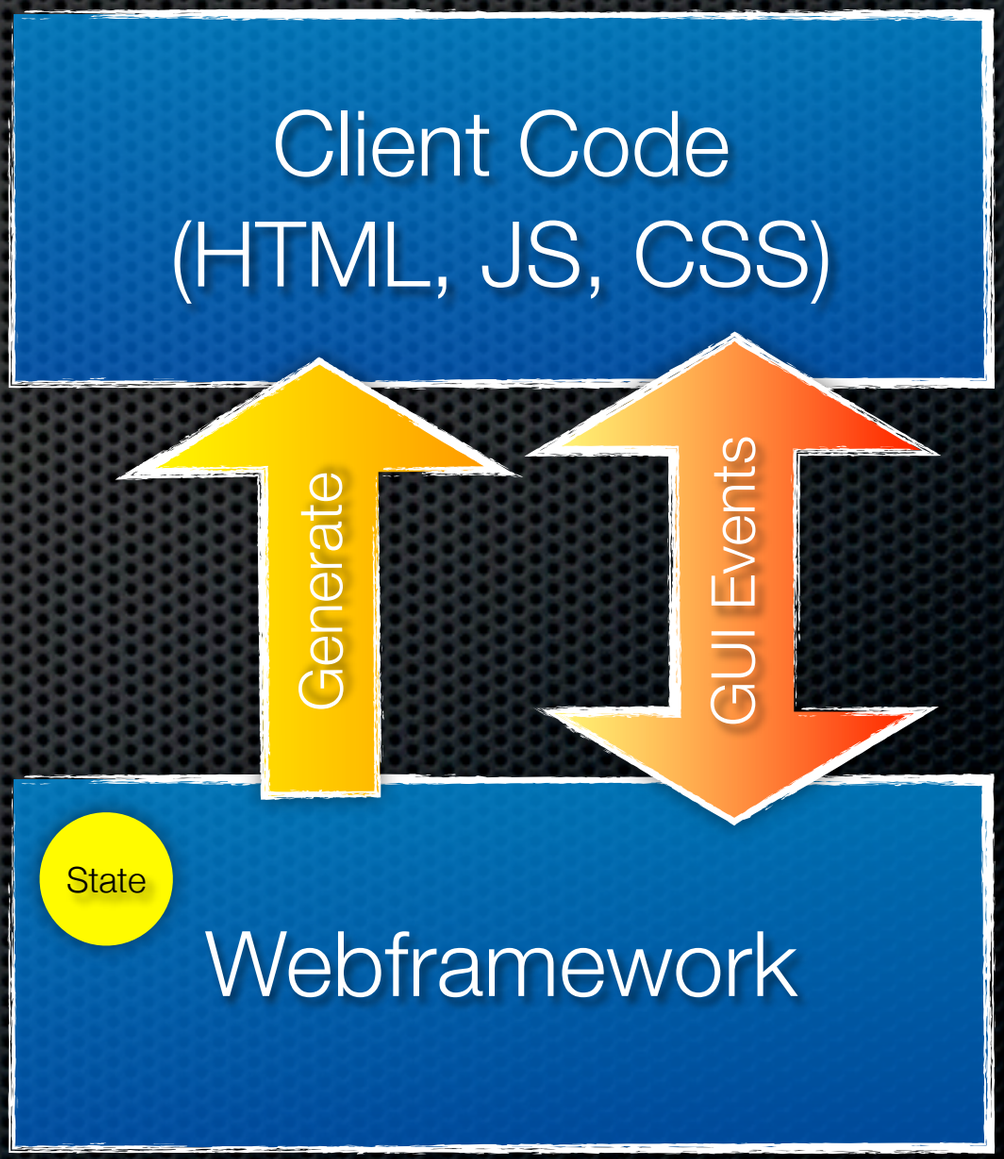
Komplexität

- Präsentationslogik ist am Client und Server
- Webframework ein Codegenerator für HTML & JS(denken in 2 Dimensionen - sehr umständlich)
- HTML muss erst in Server Templates verpackt werden



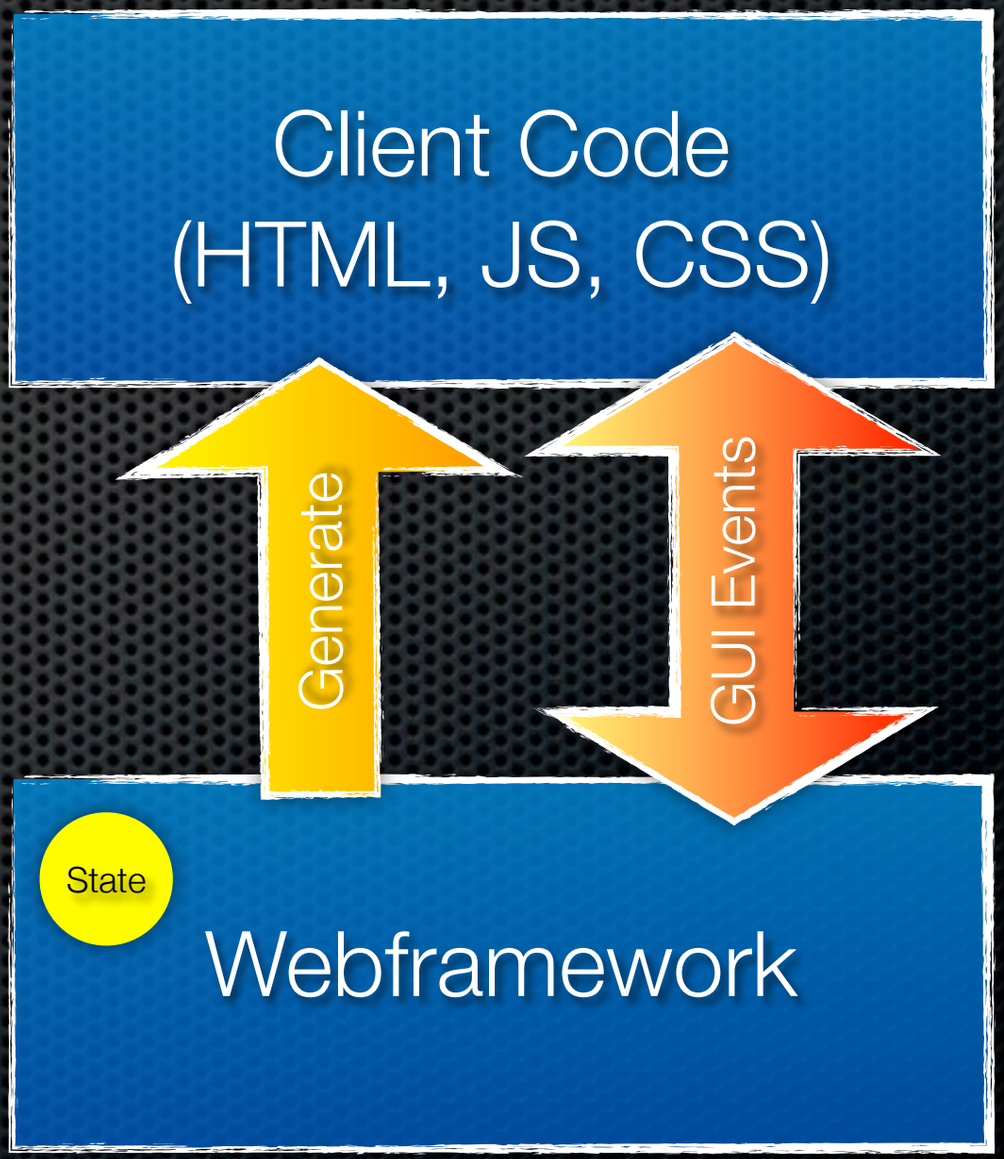
Komplexität

- Präsentationslogik ist am Client und Server
- Webframework ein Codegenerator für HTML & JS(denken in 2 Dimensionen - sehr umständlich)
- HTML muss erst in Server Templates verpackt werden
- Debugging in zwei Laufzeitumgebungen (Client/Server)



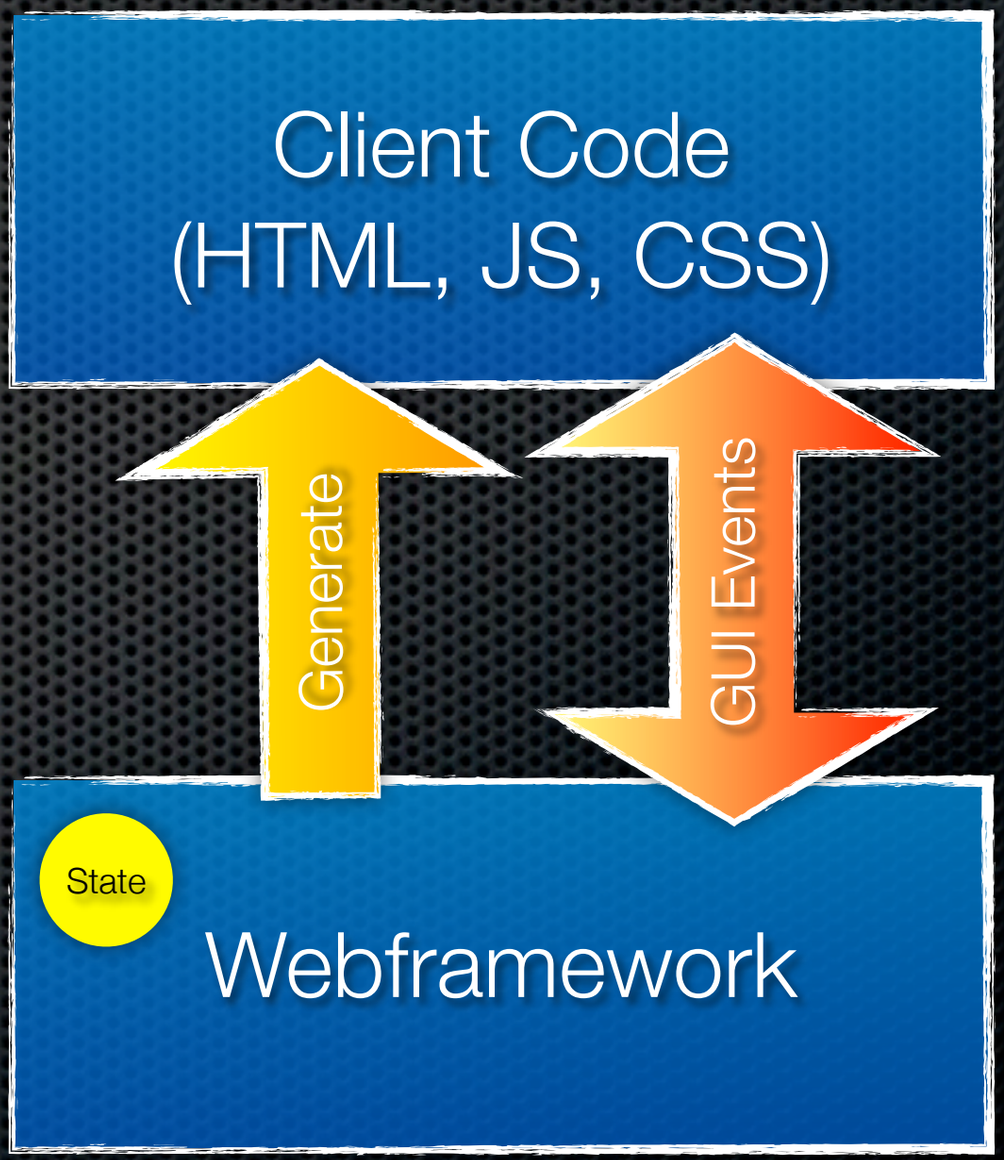
Komplexität

- Präsentationslogik ist am Client und Server
- Webframework ein Codegenerator für HTML & JS(denken in 2 Dimensionen - sehr umständlich)
- HTML muss erst in Server Templates verpackt werden
- Debugging in zwei Laufzeitumgebungen (Client/Server)
- Der meiste Code dreht sich um das Event/Request Handling (Serialisierung, Protokoll quirx)



Komplexität

- Präsentationslogik ist am Client und Server
- Webframework ein Codegenerator für HTML & JS(denken in 2 Dimensionen - sehr umständlich)
- HTML muss erst in Server Templates verpackt werden
- Debugging in zwei Laufzeitumgebungen (Client/Server)
- Der meiste Code dreht sich um das Event/Request Handling (Serialisierung, Protokoll quirx)
- Korrektes State Handling unmöglich



Fatales Statemanagement

Fatales Statemanagement

- UI State ist am Server

Fatales Statemanagement

- UI State ist am Server
- State am Client muss nicht zum Server State passen. Fehlerbehandlung hierfür ist komplex.

Fatales Statemanagement

- UI State ist am Server
- State am Client muss nicht zum Server State passen. Fehlerbehandlung hierfür ist komplex.
- Client muss immer auf den richtigen Server geroutet werden

Fatales Statemanagement

- UI State ist am Server
- State am Client muss nicht zum Server State passen. Fehlerbehandlung hierfür ist komplex.
- Client muss immer auf den richtigen Server geroutet werden
- Skalierung und Failover ein echtes Problem (und sehr teuer)

Fatales Statemanagement

- UI State ist am Server
- State am Client muss nicht zum Server State passen. Fehlerbehandlung hierfür ist komplex.
- Client muss immer auf den richtigen Server geroutet werden
- Skalierung und Failover ein echtes Problem (und sehr teuer)
- Speicherverbrauch am Server (Beispiel JSF Component Tree)

Fatales Statemanagement

- UI State ist am Server
- State am Client muss nicht zum Server State passen. Fehlerbehandlung hierfür ist komplex.
- Client muss immer auf den richtigen Server geroutet werden
- Skalierung und Failover ein echtes Problem (und sehr teuer)
- Speicherverbrauch am Server (Beispiel JSF Component Tree)
- Server behelfen sich mit Sessiontimeouts (Session bleibt meist noch 30 Minuten wenn der User bereits weg ist)

5. Offlinefähigkeit

Ein echtes Problem...

6. Fehlende Interoperabilität

6. Fehlende Interoperabilität

- **HTML Markup ist keine Schnittstelle**
  - ... GWT-RPC auch nicht

6. Fehlende Interoperabilität

- **HTML Markup ist keine Schnittstelle**
  - ... GWT-RPC auch nicht
- **Echte Services sind sind häufig nicht von „außen“ zugreifbar (z.B. Spring Services)**
  - ... nein, man kann Sie nicht einfach zum Webservice machen!
    - es ist ein fundamentaler Unterschied ob ein Service als WebService oder lokaler Service konzeptioniert wurde (Fehlerbehandlung, Call by Value, Granularität)
  - Ergo: SOA Projekte müssen von 0 starten
    - Risiko das Rad neu zu erfinden

6. Fehlende Interoperabilität

- **HTML Markup ist keine Schnittstelle**
  - ... GWT-RPC auch nicht
- **Echte Services sind sind häufig nicht von „außen“ zugreifbar (z.B. Spring Services)**
  - ... nein, man kann Sie nicht einfach zum Webservice machen!
    - es ist ein fundamentaler Unterschied ob ein Service als WebService oder lokaler Service konzeptioniert wurde (Fehlerbehandlung, Call by Value, Granularität)
  - Ergo: SOA Projekte müssen von 0 starten
    - Risiko das Rad neu zu erfinden
- **Fehlender Fokus auf qualitative Schnittstellen („die sind ja nicht von außen erreichbar“)**

6. Fehlende Interoperabilität

- **HTML Markup ist keine Schnittstelle**
  - ... GWT-RPC auch nicht
- **Echte Services sind sind häufig nicht von „außen“ zugreifbar (z.B. Spring Services)**
  - ... nein, man kann Sie nicht einfach zum Webservice machen!
    - es ist ein fundamentaler Unterschied ob ein Service als WebService oder lokaler Service konzeptioniert wurde (Fehlerbehandlung, Call by Value, Granularität)
  - Ergo: SOA Projekte müssen von 0 starten
    - Risiko das Rad neu zu erfinden
- **Fehlender Fokus auf qualitative Schnittstellen („die sind ja nicht von außen erreichbar“)**
- **Präsentation und Geschäftslogik in einem Tier fördert das verschmelzen von beidem zu einem „Brei“**

Symptome linderbar,
aber nicht fixbar!

Symptome linderbar, aber nicht fixbar!

- Zustand -> HTTPSession/Conversations

Symptome linderbar, aber nicht fixbar!

- Zustand -> HTTPSession/Conversations
- AJAX Feeling -> Partitial Page Rendering

Symptome linderbar, aber nicht fixbar!

- Zustand -> HTTPSession/Conversations
- AJAX Feeling -> Partitial Page Rendering
- Multi Windows -> Conversations

Symptome linderbar, aber nicht fixbar!

- Zustand -> HTTPSession/Conversations
- AJAX Feeling -> Partitial Page Rendering
- Multi Windows -> Conversations
- Browserhistory -> Post/Redirect/Get (PRG)

Symptome linderbar, aber nicht fixbar!

- Zustand -> HTTPSession/Conversations
- AJAX Feeling -> Partitial Page Rendering
- Multi Windows -> Conversations
- Browserhistory -> Post/Redirect/Get (PRG)
- Failover -> Session Repication = BS! (Speicher, Kosten, Network Traffic..)

Symptome linderbar, aber nicht fixbar!

- Zustand -> HTTPSession/Conversations
- AJAX Feeling -> Partitial Page Rendering
- Multi Windows -> Conversations
- Browserhistory -> Post/Redirect/Get (PRG)
- Failover -> Session Repication = BS! (Speicher, Kosten, Network Traffic..)

Symptome linderbar, aber nicht fixbar!

- Zustand -> HTTPSession/Conversations
- AJAX Feeling -> Partitial Page Rendering
- Multi Windows -> Conversations
- Browserhistory -> Post/Redirect/Get (PRG)
- Failover -> Session Repication = BS! (Speicher, Kosten, Network Traffic..)

- **Verhindert echtes Webscale!**

Hmm, Code am Server war doch immer eine gute Idee

JS sucks! Bowser Inkomaptibilität! ...und die Sicherheit!

Ja, bis 2005 war das richtig

- Für viele hat sich die Welt seither nicht weiter gedreht
- Seit 2005 gibt es viel neues
  - AJAX
  - Browser Events
  - CSS 2/3
  - JavaScript Programming Patterns (Classes,Events, DOM)
  - jQuery
  - DOM Positioning / Box Models

Und Heute? **HTML5**.

- KV/SQL-Database
- Canvas
- CSS3
- Web Workers
- Web Sockets
- Application Cache
- JS DOM Selector API
- WEB-GL

tolle neue Möglichkeiten...

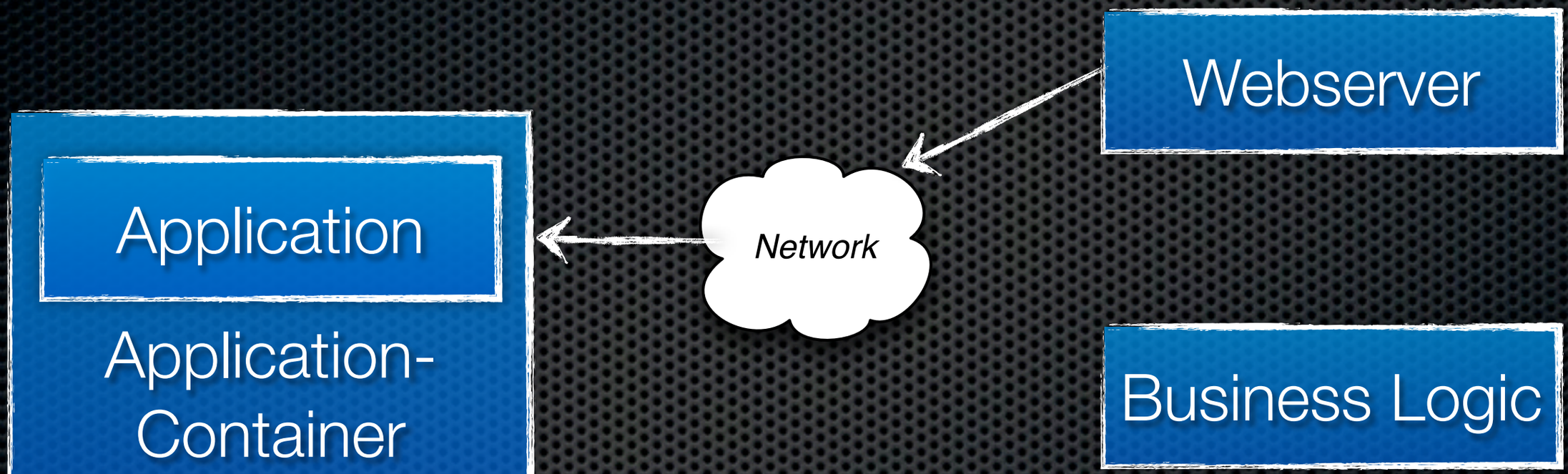
Motivation SOFEA

- Präsentation an den Client (seperation of concerns)
- Services zum Server
- SOA - Agnostische WebServices
- Multi Device (Mobile, Portal, Webclient, Richclient, APIs)
- Webscale

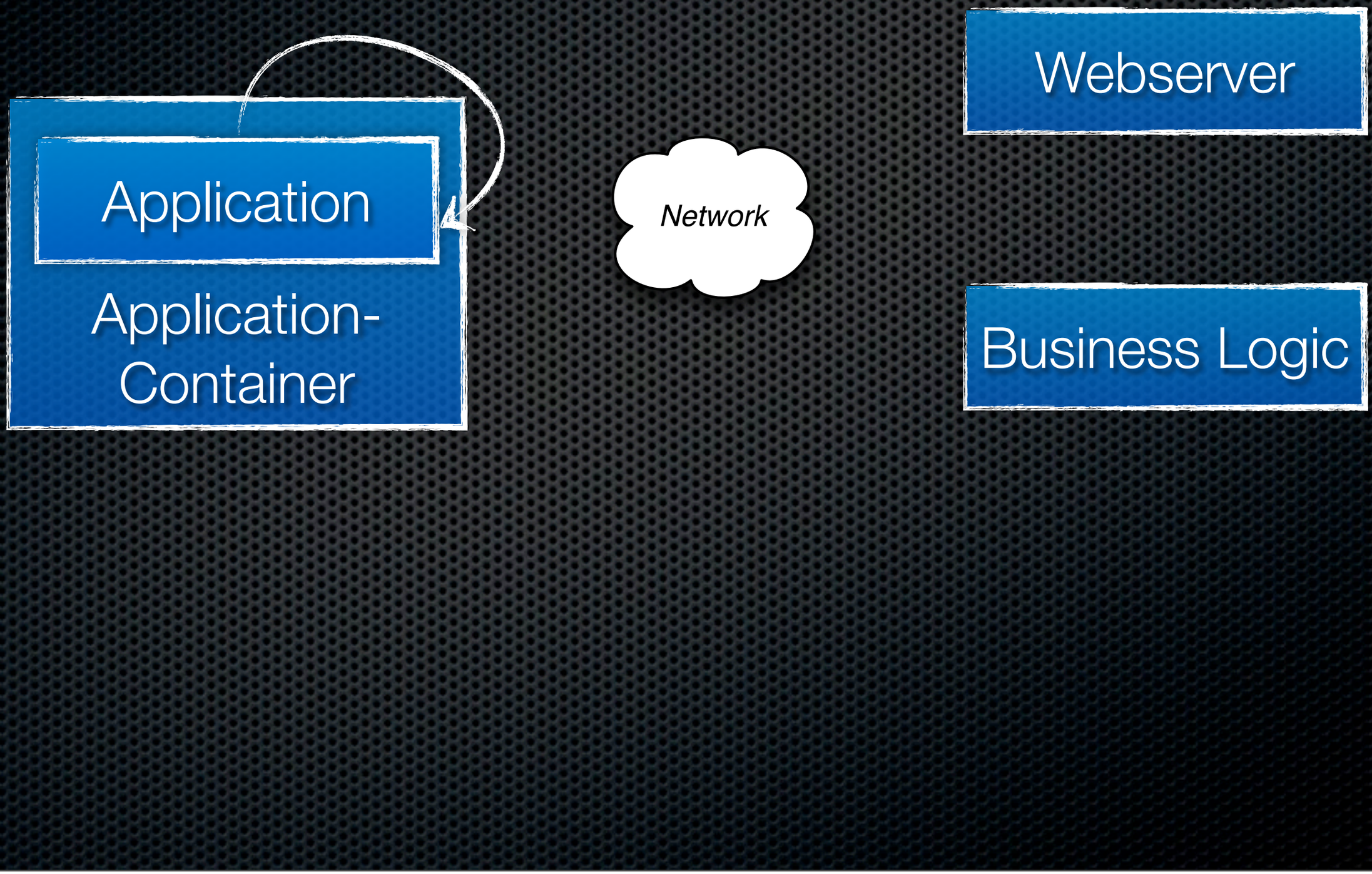
SOFEA Vogelperspektive

Zuständigkeiten eines Webframeworks

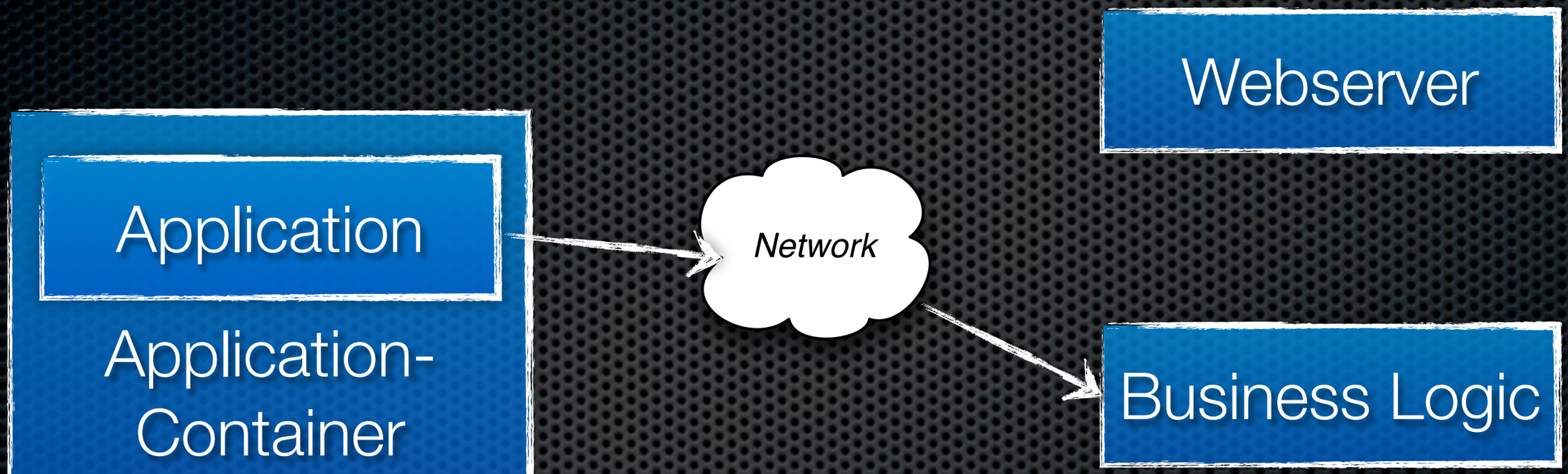
Application Download



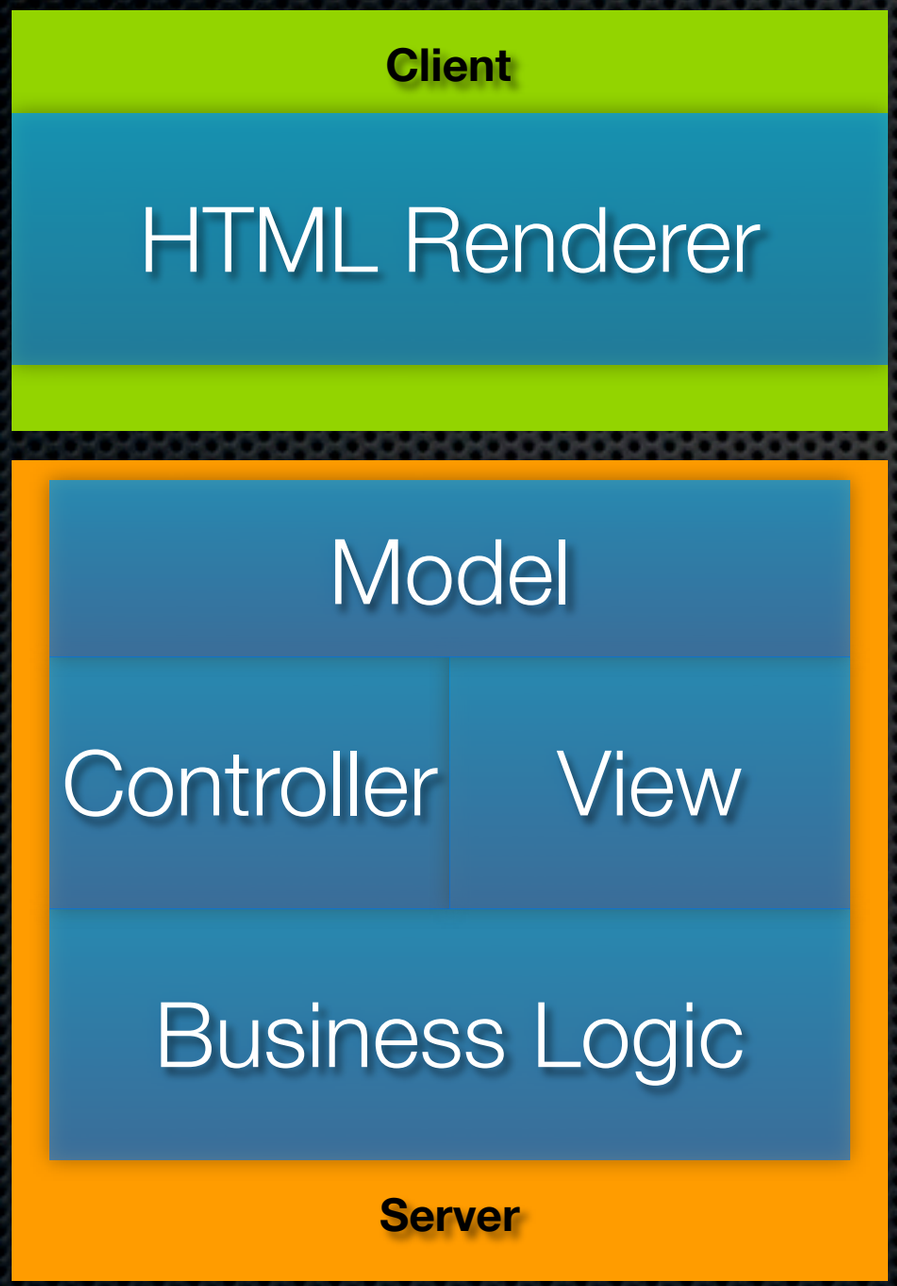
Presentation Flow



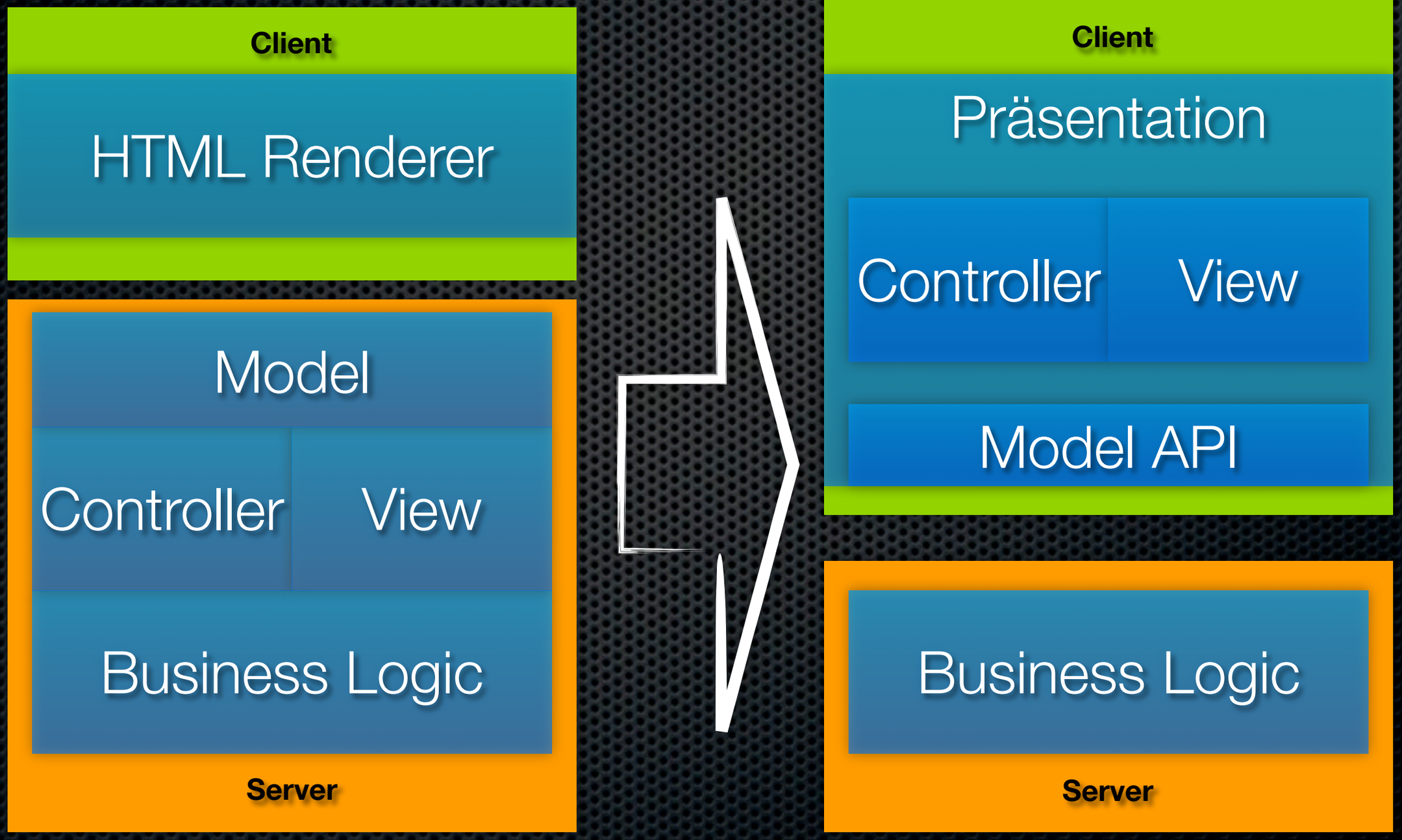
Data Interchange



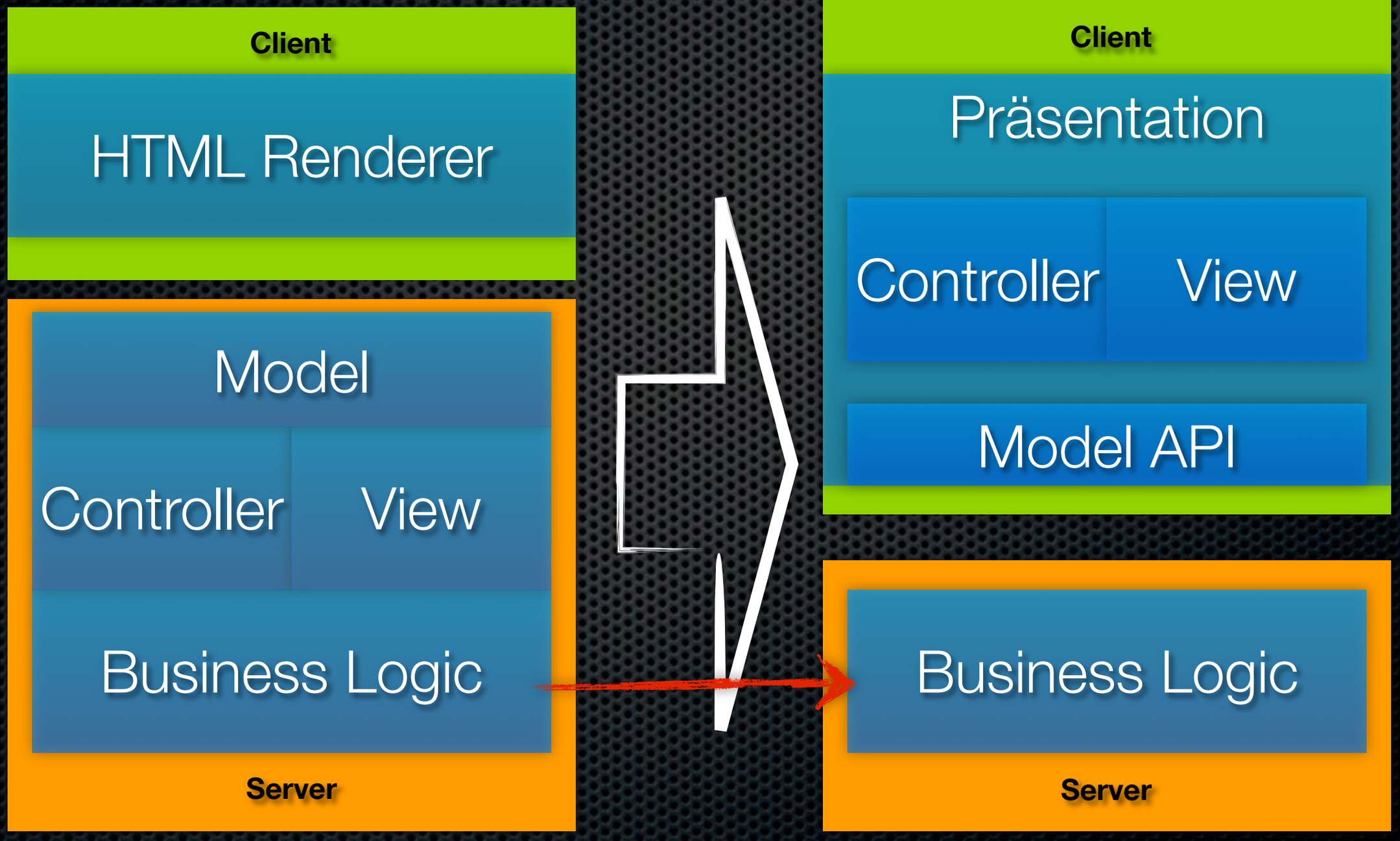
Back to basic richtiges 3 Tier



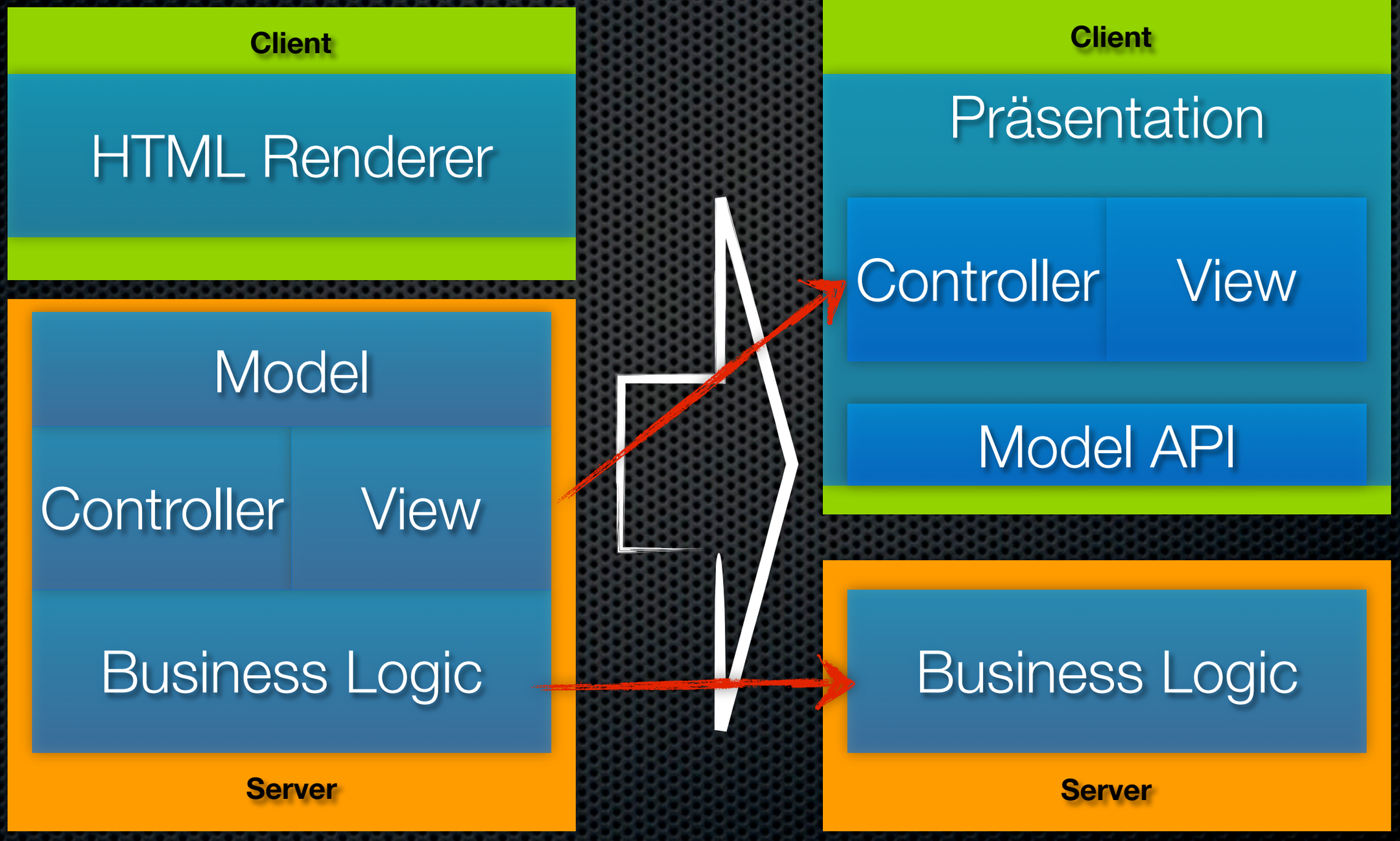
Back to basic richtiges 3 Tier



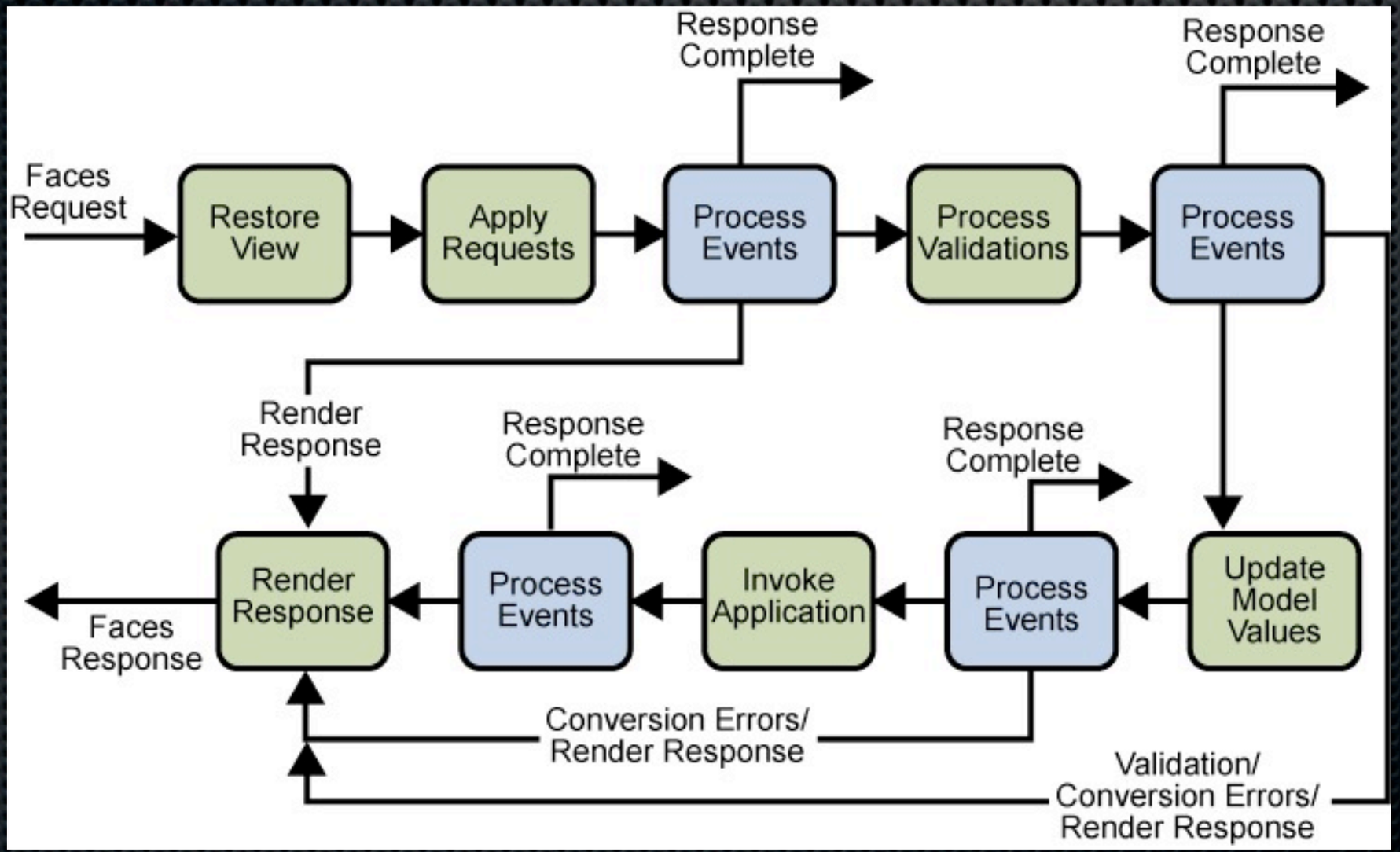
Back to basic richtiges 3 Tier



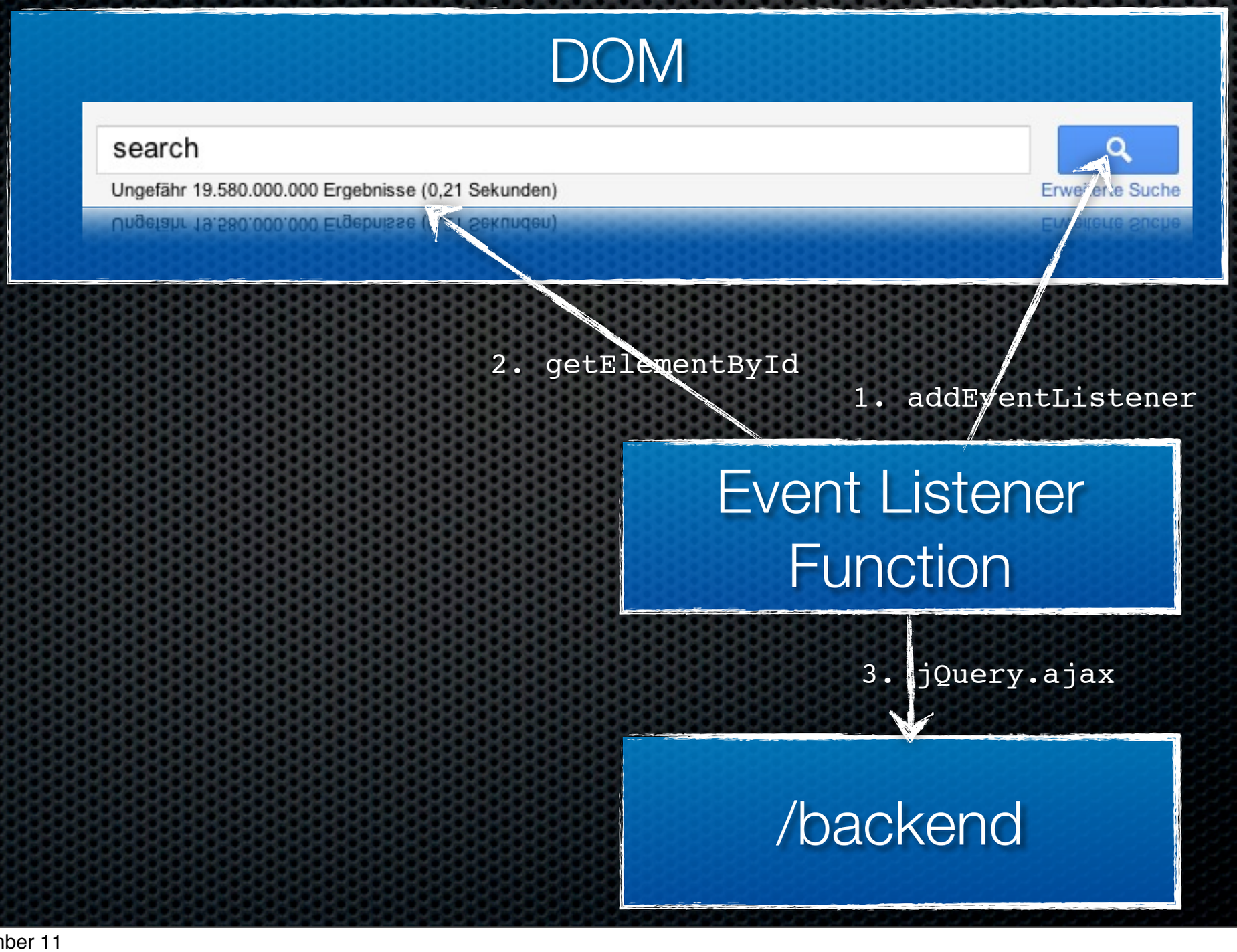
Back to basic richtiges 3 Tier



Vergleich mit JSF Lifecycle



und so in JavaScript:



Die Gretchenfrage

Thin or Rich (SPA Single Page Application) ?

SOFEA (Rich) Clients

- **Nicht RESTful**
  - SOFEA verletzt das REST Prinzip HATEOAS (Hypermedia as the Engine of Application State)
  - State ist nicht Bestandteil der URL
    - idR. ein großer Download
- **History und „Browser Back“ problematisch**
  - # URI Fragmente können helfen
  - neu: Browser History API
- **Aber: Offline fähig**

Kompromiss:
Rich Thin Clients

Kompromiss:
Rich Thin Clients

- RESTful

Kompromiss:
Rich Thin Clients

- RESTful
- viele kleine Downloads

Kompromiss:
Rich Thin Clients

- RESTful
- viele kleine Downloads
- reduzierte Komplexität

Kompromiss:
Rich Thin Clients

- RESTful
- viele kleine Downloads
- reduzierte Komplexität

Kompromiss:
Rich Thin Clients

- RESTful
- viele kleine Downloads
- reduzierte Komplexität

- Aber:
  - nicht offline fähig
  - nicht als Mobile App geeignet
  - Server Roundtripps bei Wechsel der URL/Resource

Vorteile SOFEA

Vorteile SOFEA

- Trennung zwischen Präsentation und Geschäftslogik
  - Aufgeräumtes Programmiermodell (GUI läuft am Client)

Vorteile SOFEA

- Trennung zwischen Präsentation und Geschäftslogik
  - Aufgeräumtes Programmiermodell (GUI läuft am Client)
- Keine Serverlatenz bei GUI Operationen

Vorteile SOFEA

- Trennung zwischen Präsentation und Geschäftslogik
  - Aufgeräumtes Programmiermodell (GUI läuft am Client)
- Keine Serverlatenz bei GUI Operationen
- Hoch skalierbar
- Client side state management

Vorteile SOFEA

- Trennung zwischen Präsentation und Geschäftslogik
  - Aufgeräumtes Programmiermodell (GUI läuft am Client)
- Keine Serverlatenz bei GUI Operationen
- Hoch skalierbar
  - Client side state management
- Offlineanwendungen

Vorteile SOFEA

- Trennung zwischen Präsentation und Geschäftslogik
  - Aufgeräumtes Programmiermodell (GUI läuft am Client)
- Keine Serverlatenz bei GUI Operationen
- Hoch skalierbar
  - Client side state management
- Offlineanwendungen
- Mobile Apps

Vorteile SOFEA

- Trennung zwischen Präsentation und Geschäftslogik
  - Aufgeräumtes Programmiermodell (GUI läuft am Client)
- Keine Serverlatenz bei GUI Operationen
- Hoch skalierbar
  - Client side state management
- Offlineanwendungen
- Mobile Apps
- Interoperability
  - SOFEA fördert „echte“ und qualitative Services
  - Backend kann in jeder Spache geschrieben sein (Ruby, JS, Java, .net...)

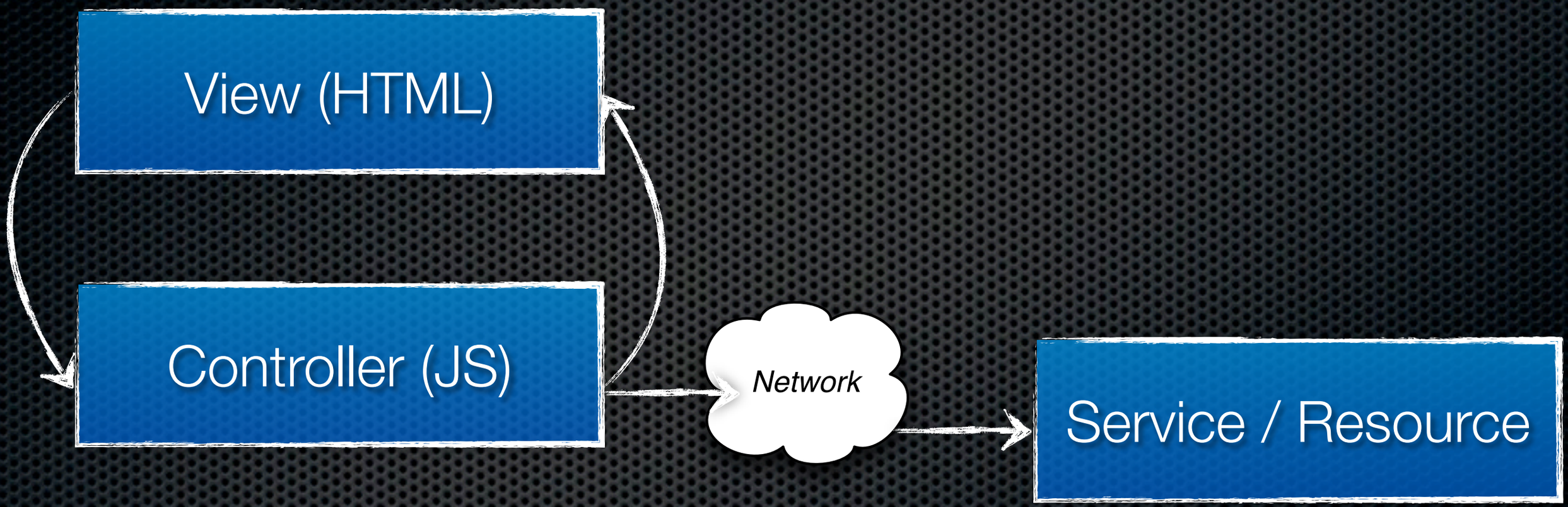
No Serverside Webframeworks

HTML5 ist das Framework!

HTML(5) hat alles was man braucht

- JavaScript für die Oberfächensteuerung
- DOM Selector API um die View zu manipulieren
- AJAX als Remote HTTP API

SOFEA Client Architecture



Architektur?!

- ein großes Javascript...
- eine HTML Seite...

Wohin mit dem State?

- JS var
- Cookie
- Browser DB / local Storage
- HTML5 History API
- URL #

Rann an die API

- REST
- SOAP
- ATOM/RSS
- Comet/Ajax Push
- Webworker

Problem: Same Origin Policy

- Reverse Proxy
- JSONP
- HTML5 - send Message
- Cross-Side XMLHTTP Request (Mozilla)

Gut genug **für einfache Apps**

oder Thin Clients...

Kontakt

Bitte verwenden Sie dieses Formular, wenn Sie Fragen an uns haben oder weitere Informationen benötigen.

Name: *

E-Mail: *

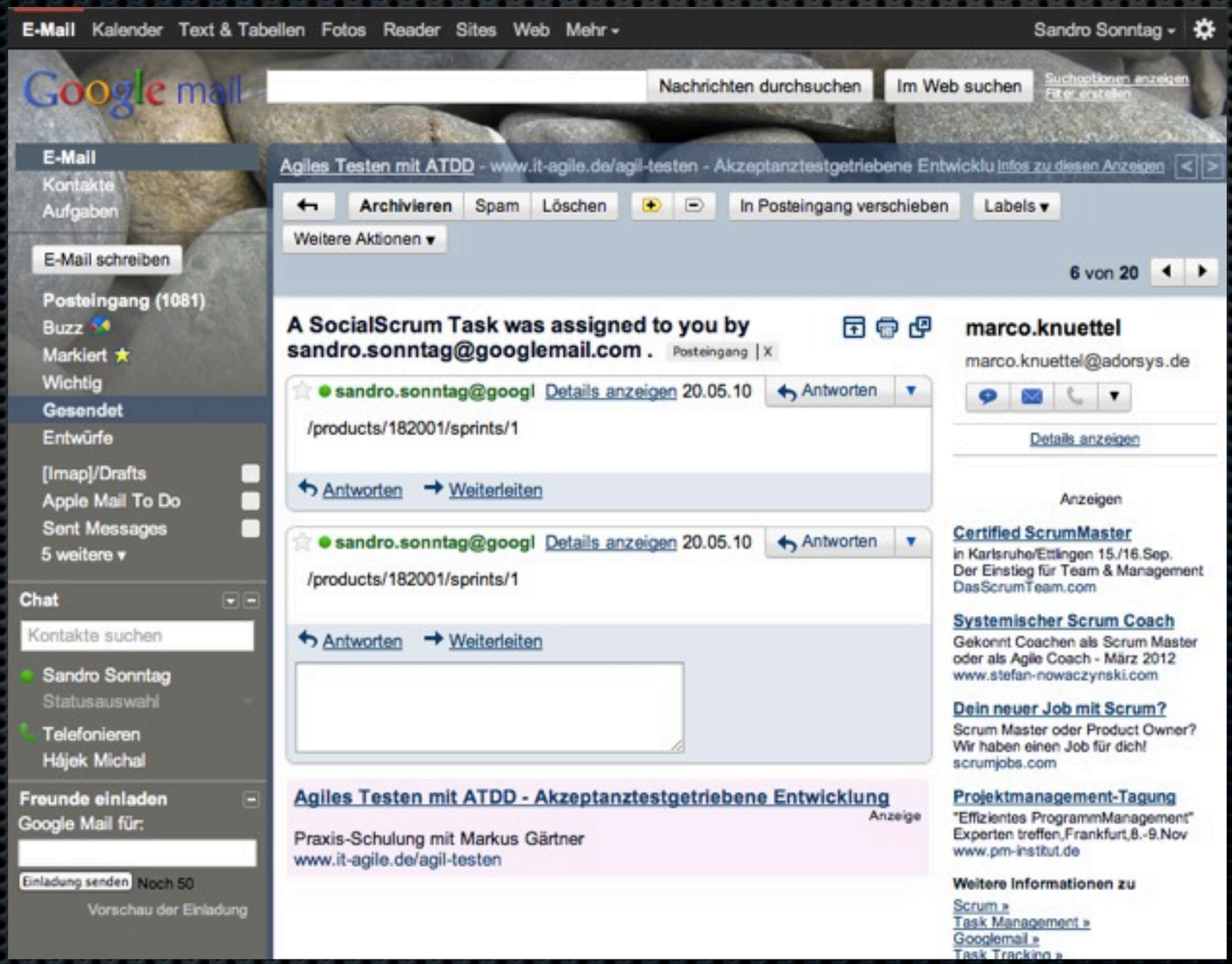
Betreff: *

Nachricht: *

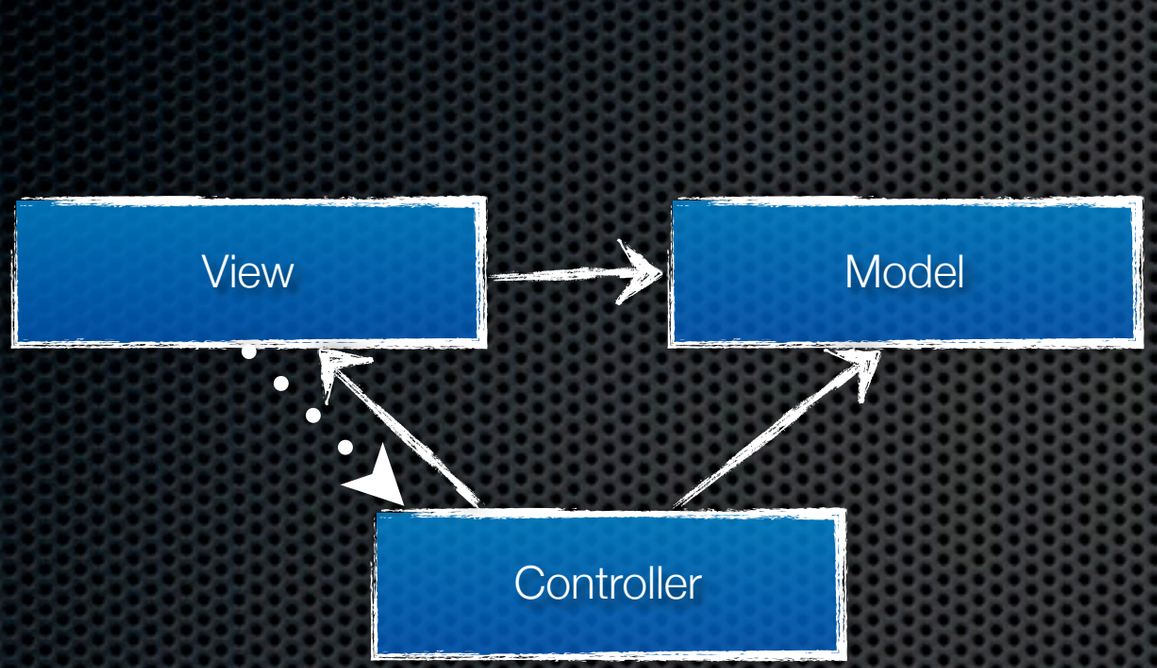
* Bitte alle Pflichtfelder ausfüllen!

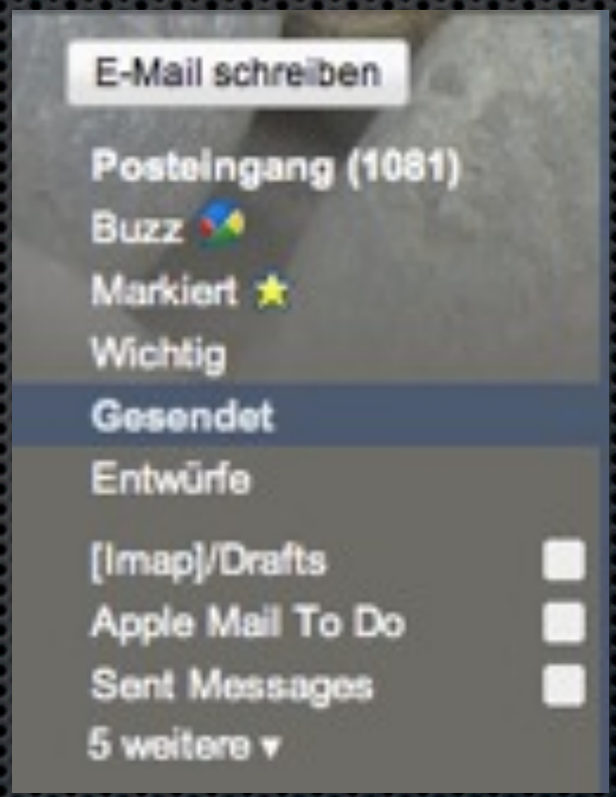
Formular absenden

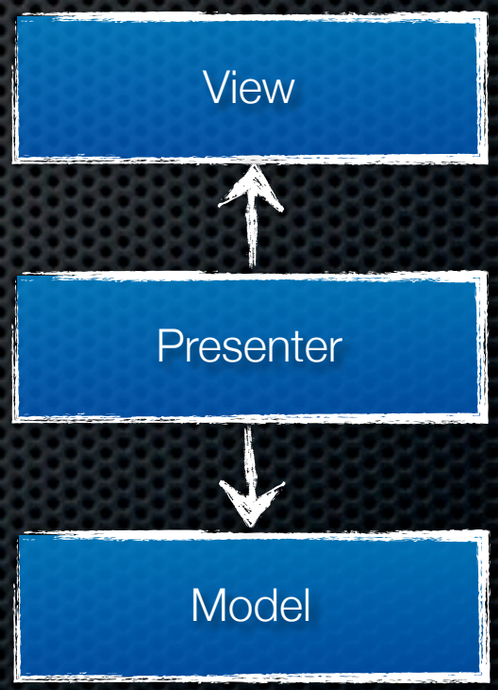
Aber wie würden wir Applikationen wie GMail realisieren?



MVP (Model View Presenter) Pattern

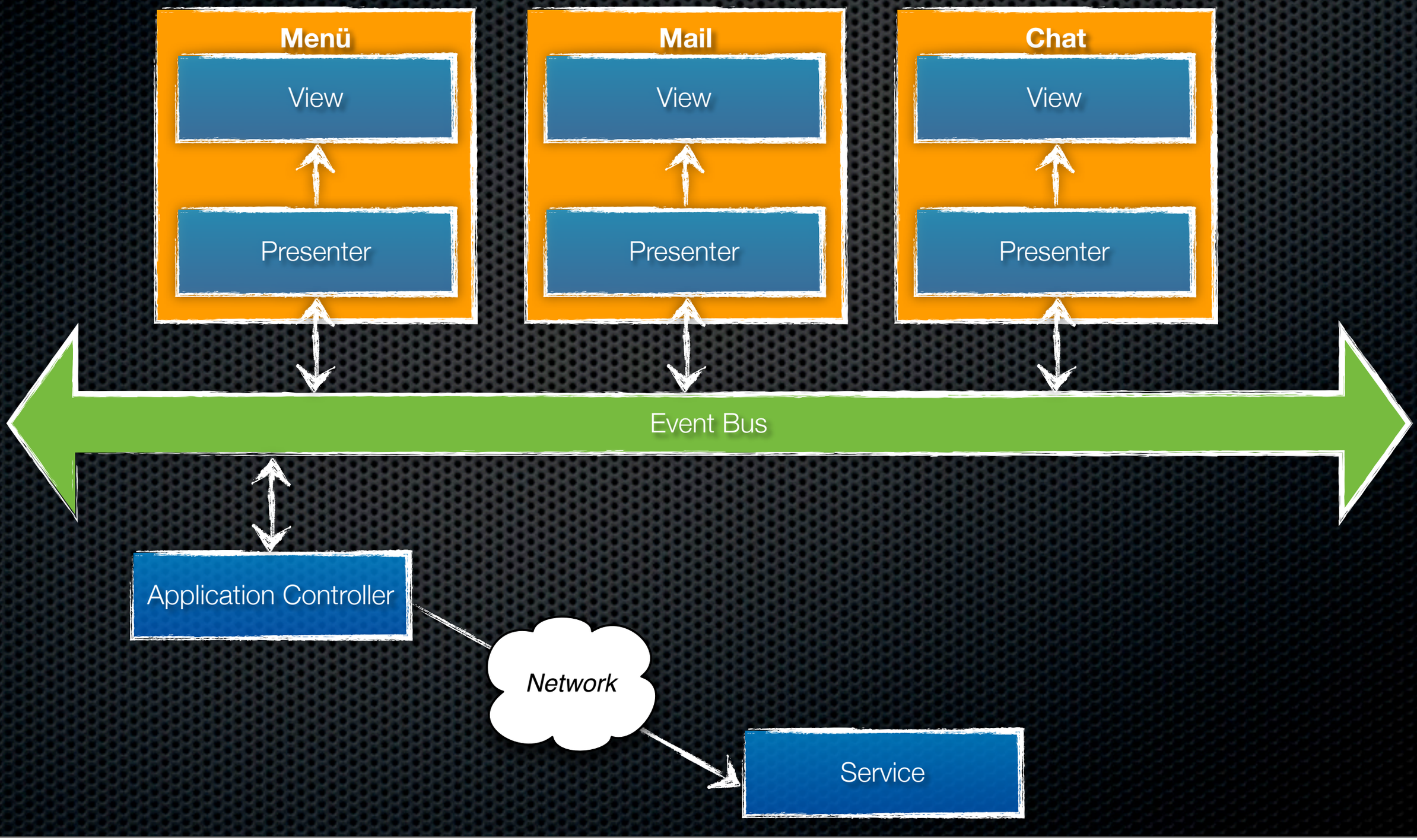






SOFEA Architecture 2.0

MVP + EventBus



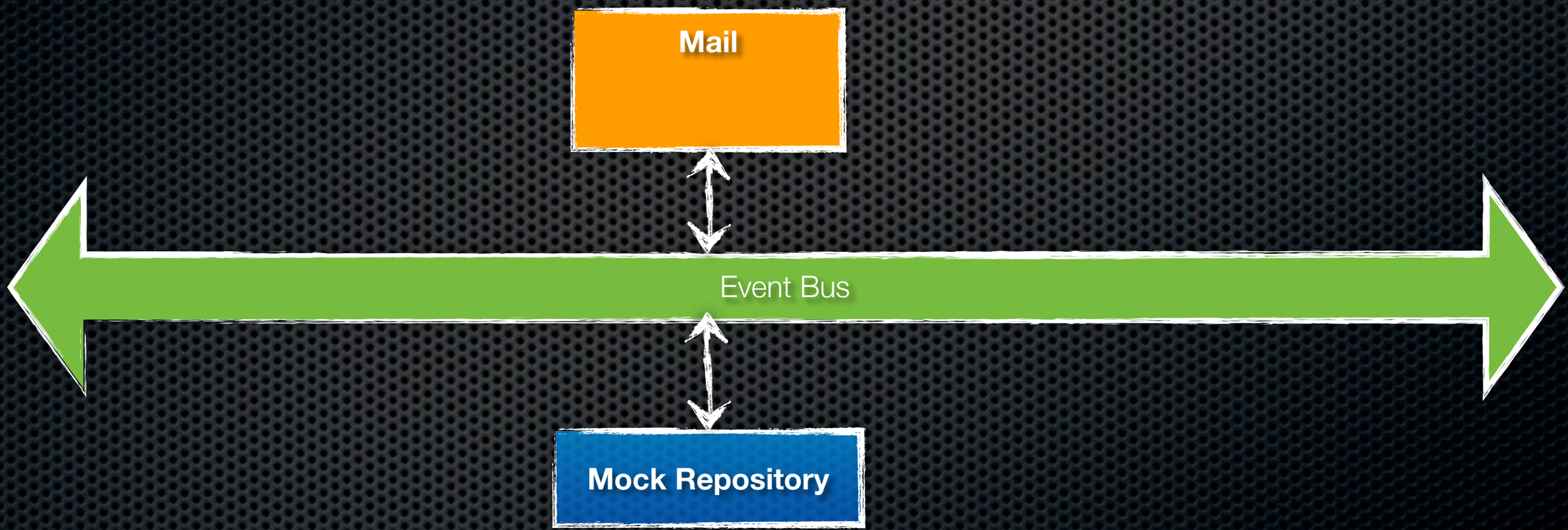
OpenAJAX Hub

- Standard Client Side JavaScript Hub
- publish/subscribe
- zum schreiben von Secure Mashups
  - Sandboxing von Mashups
- aktuell in Version 2

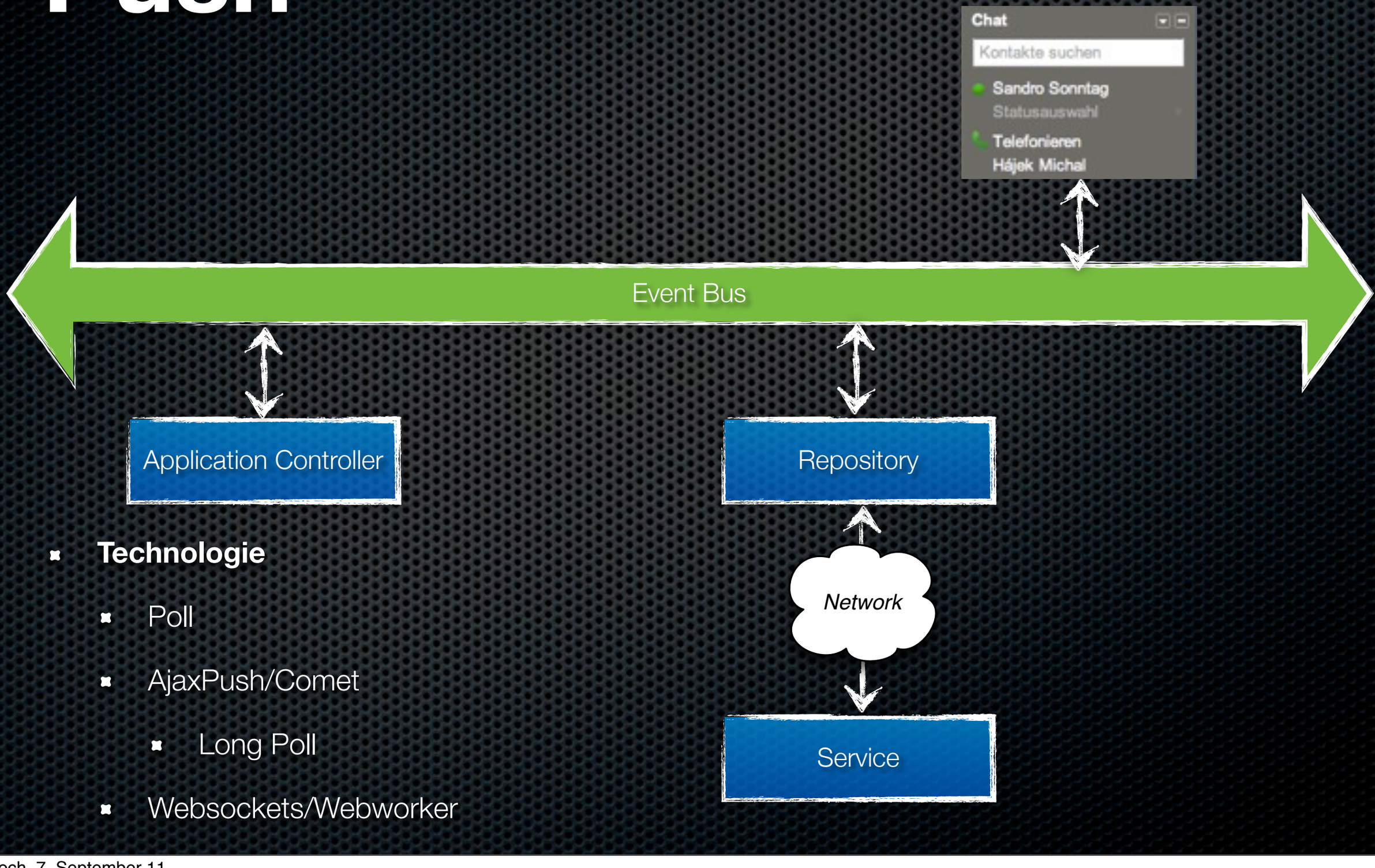
```
OpenAjax.hub.publish("de.adorsys.urlaubsantrag.new", "2007.08.05");

OpenAjax.hub.subscribe("de.adorsys.urlaubsantrag.*", function() {
    //hello
}, this);
```

SOFEA Architecture 2.0
Tests mit Mocks

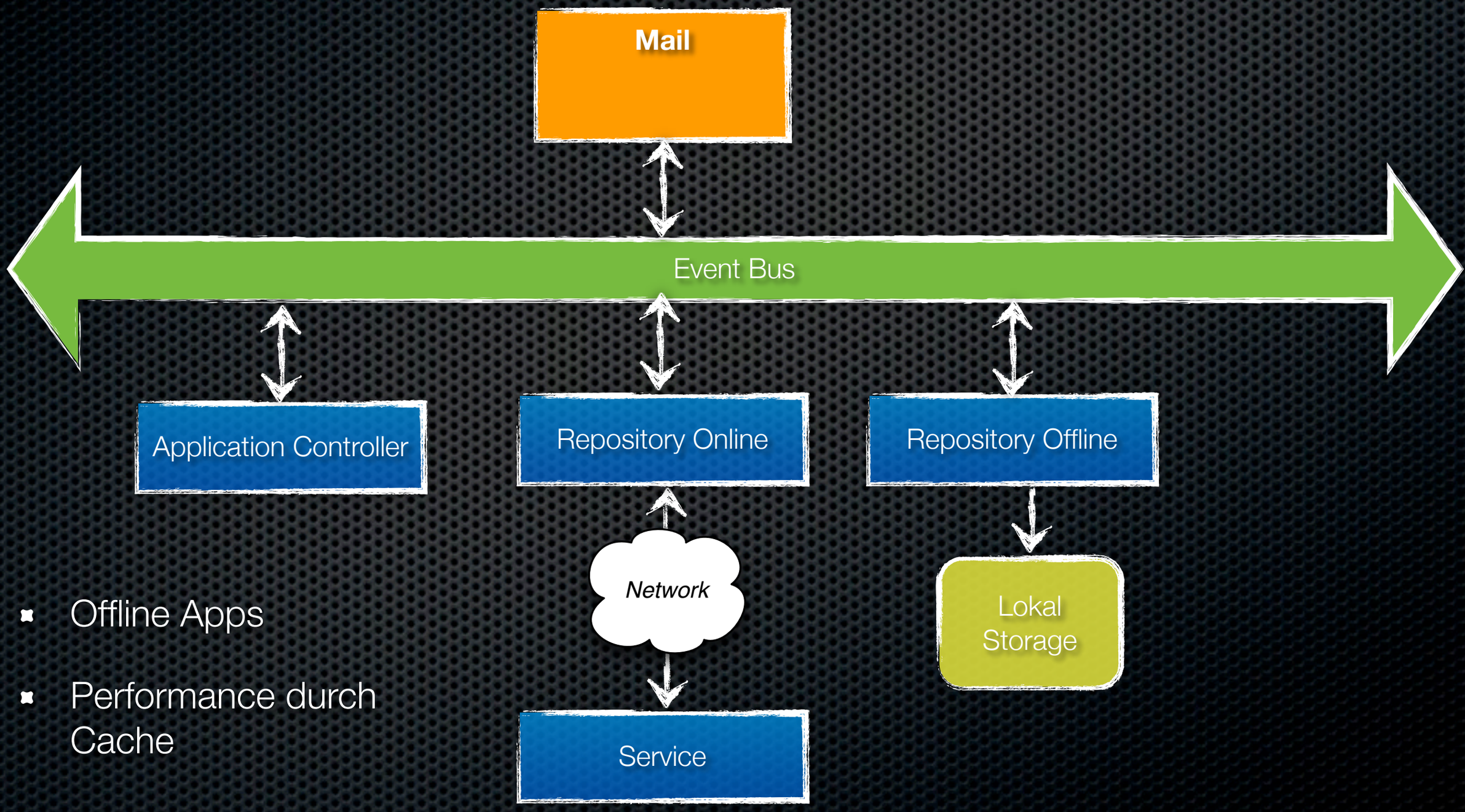


SOFEA Architecture 2.0
Push



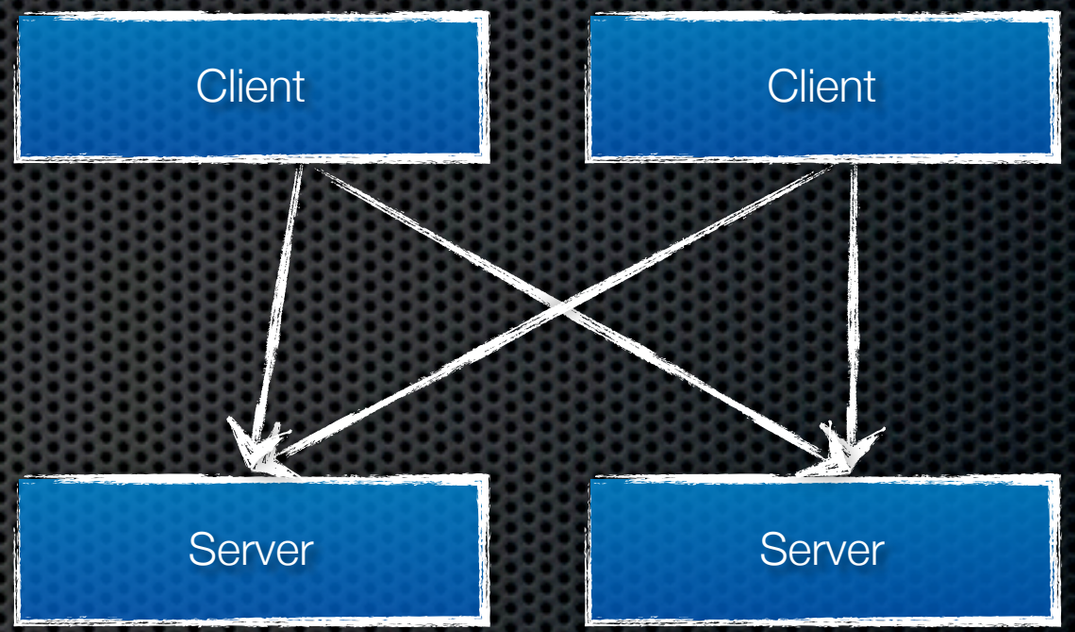
- **Technologie**
  - Poll
  - AjaxPush/Comet
    - Long Poll
  - Websockets/Webworker

SOFEA Architecture 2.0
Offline Cache

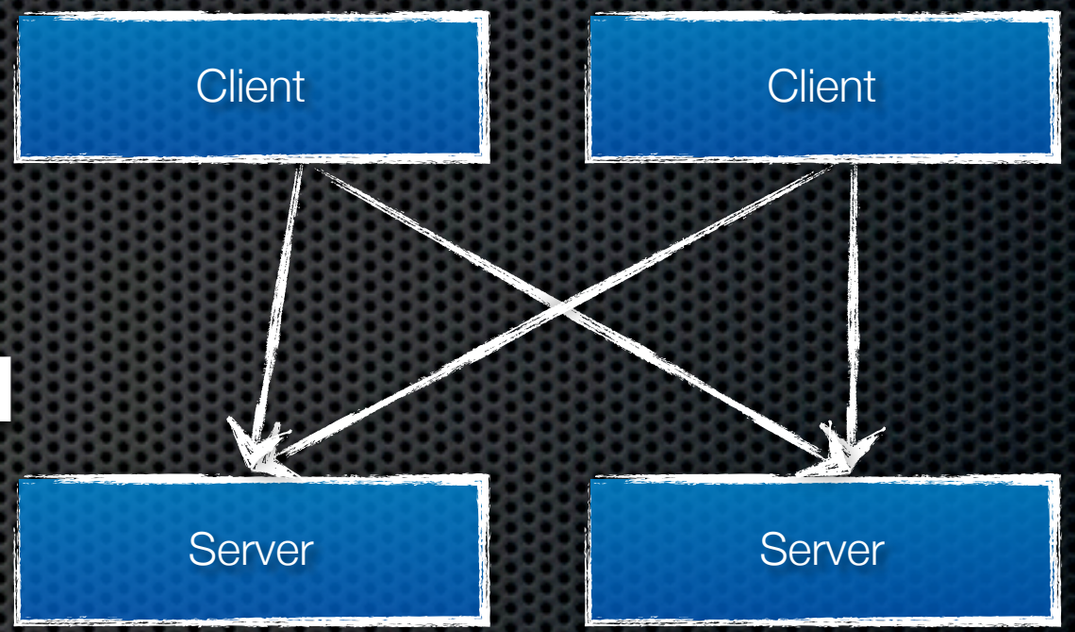


- Offline Apps
- Performance durch Cache

Scaling SOFEA



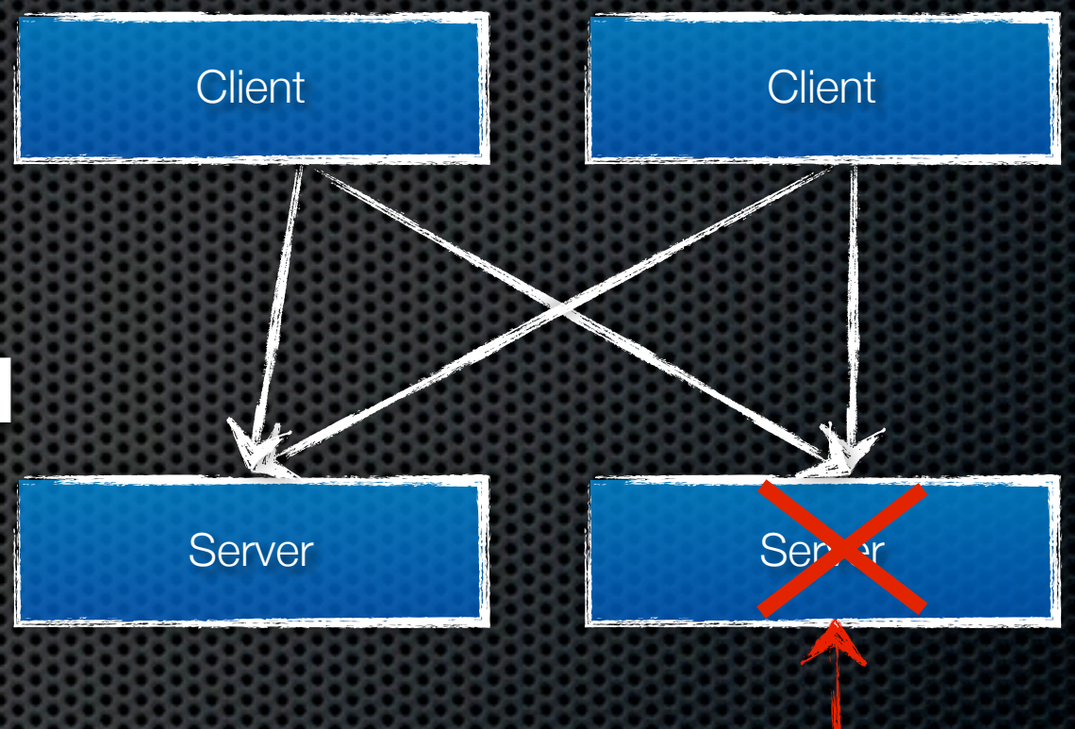
Scaling SOFEA



- GUI Rendering am Client
- **Konzept: Stateless Backend**
  - REST
  - Caching (Client/Server)
  - einfache und „billige“ Skalierung durch zusätzliche Hardware
  - keine (sticky) Sessions
    - transparent Failover

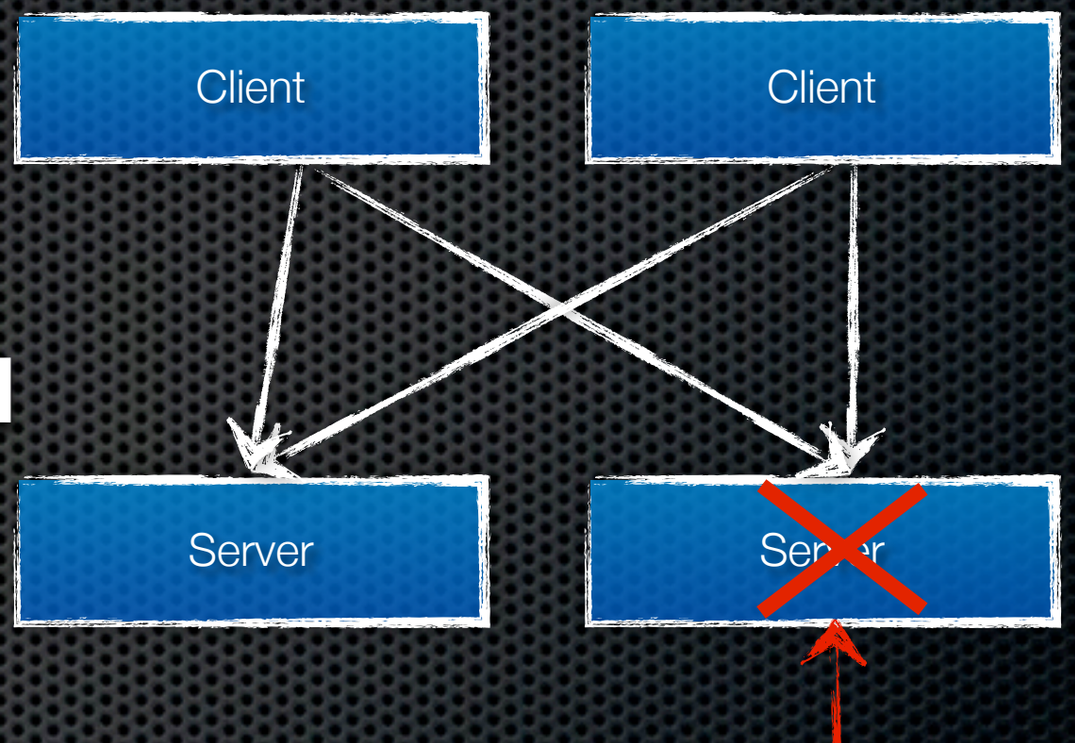
Scaling SOFEA

- GUI Rendering am Client
- **Konzept: Stateless Backend**
  - REST
    - Caching (Client/Server)
    - einfache und „billige“ Skalierung durch zusätzliche Hardware
    - keine (sticky) Sessions
      - transparent Failover



Scaling SOFEA

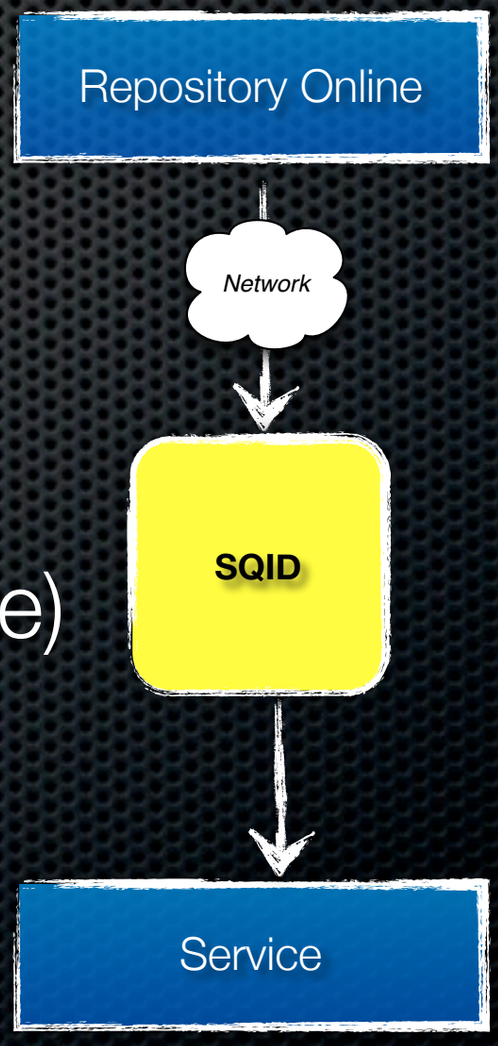
ein neues Problem die Datenbank?



- GUI Rendering am Client
- **Konzept: Stateless Backend**
  - REST
  - Caching (Client/Server)
  - einfache und „billige“ Skalierung durch zusätzliche Hardware
  - keine (sticky) Sessions
    - transparent Failover

Caching

- RESTful HTTP ist der Schlüssel
- HTTP Client Cache (expires, E-TAG)
- durch Infrastruktur (Reverse Proxy Cache)
- Serverside Service Cache



Authentication in SOFEA

Authentication in SOFEA

- **Die meisten Appserver präferieren Session Auth**
  - Dadurch ist Server stickiness erforderlich

Authentication in SOFEA

- **Die meisten Appserver präferieren Session Auth**
  - Dadurch ist Server stickiness erforderlich
- **Lösung: Security Tokens**
  - Token basierte Protokolle
    - Eigene Protokolle
    - OAuth
    - OpenID

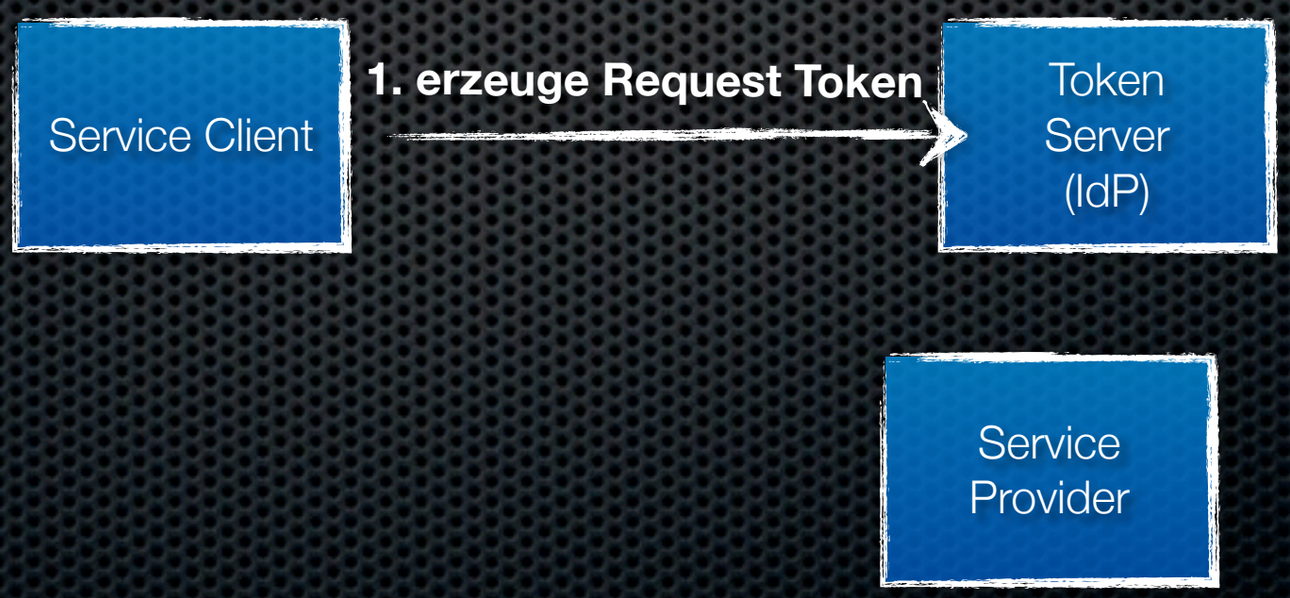
Funktionsweise - **OAuth**

Service Client

Token Server (IdP)

Service Provider

Funktionsweise - **OAuth**



Funktionsweise - **OAuth**

2. Authentisiere request Token

Service Client

1. erzeuge Request Token

Token Server (IdP)

Service Provider

Funktionsweise - **OAuth**

2. Authentisiere request Token

3. redirect request Token param

1. erzeuge Request Token

Service Client

Token Server (IdP)

Service Provider

Funktionsweise - **OAuth**

2. Authentisiere request Token

3. redirect request Token param

Service Client

1. erzeuge Request Token

4. erzeuge Access Token

Token Server (IdP)

Service Provider

Funktionsweise - **OAuth**

2. Authentisiere request Token

3. redirect request Token param

Service Client

1. erzeuge Request Token

4. erzeuge Access Token

Token Server (IdP)

5. API Call

Service Provider

Funktionsweise - **OAuth**

2. Authentisiere request Token

3. redirect request Token param

Service Client

1. erzeuge Request Token

4. erzeuge Access Token

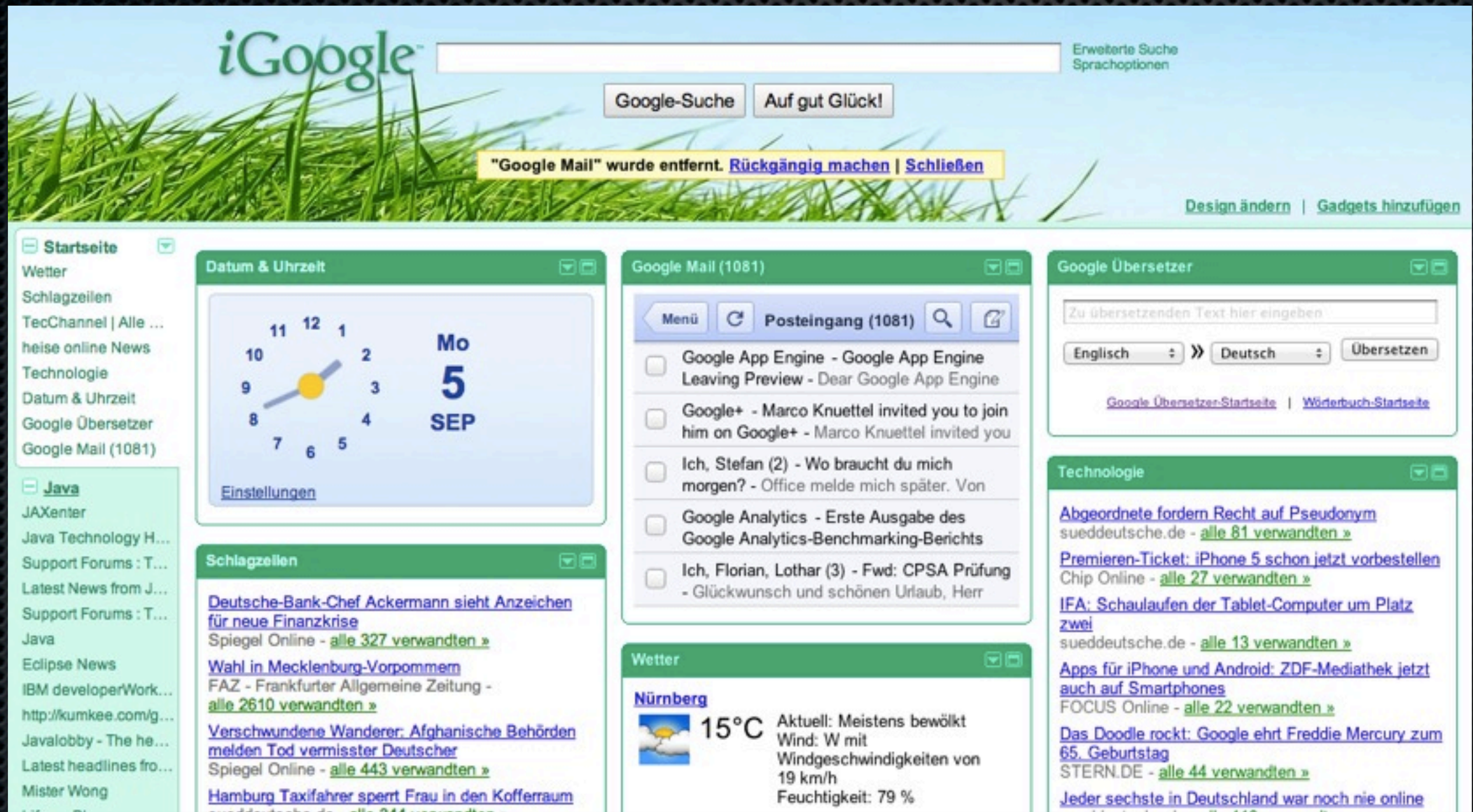
Token Server (IdP)

5. API Call

Service Provider

6. check Token

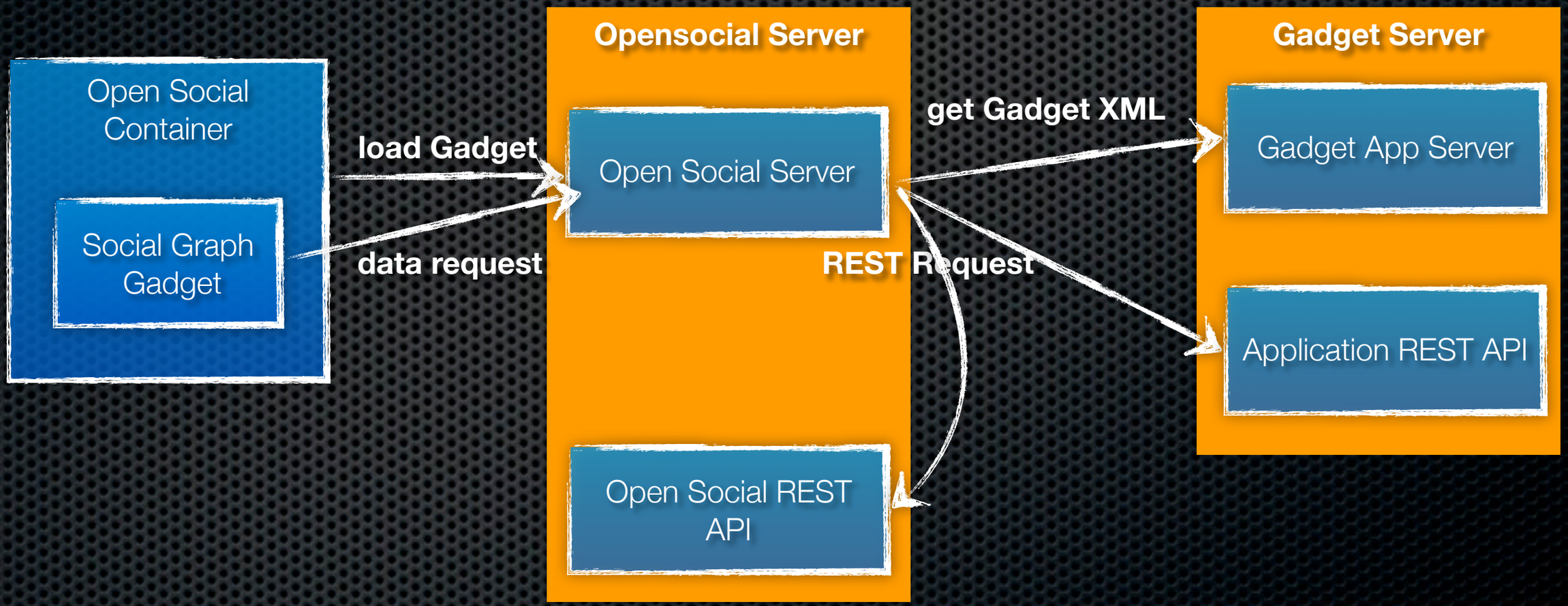
OpenSocial


iGoogle
Erweiterte Suche
Sprachoptionen
Google-Suche
Auf gut Glück!
"Google Mail" wurde entfernt. Rückgängig machen | Schließen
Design ändern | Gadgets hinzufügen
Startseite
Wetter
Schlagzeilen
TecChannel | Alle ...
heise online News
Technologie
Datum & Uhrzeit
Google Übersetzer
Google Mail (1081)
Java
JAXenter
Java Technology H...
Support Forums : T...
Latest News from J...
Support Forums : T...
Java
Eclipse News
IBM developerWork...
http://kumkee.com/g...
Javalobby - The he...
Latest headlines fro...
Mister Wong
Datum & Uhrzeit
Mo
5
SEP
Einstellungen
Schlagzeilen
Deutsche-Bank-Chef Ackermann sieht Anzeichen für neue Finanzkrise
Spiegel Online - alle 327 verwandten »
Wahl in Mecklenburg-Vorpommern
FAZ - Frankfurter Allgemeine Zeitung - alle 2610 verwandten »
Verschwundene Wanderer: Afghanische Behörden melden Tod vermisster Deutscher
Spiegel Online - alle 443 verwandten »
Hamburg Taxifahrer sperrt Frau in den Kofferraum
Google Mail (1081)
Menü
Posteingang (1081)
Google App Engine - Google App Engine Leaving Preview - Dear Google App Engine
Google+ - Marco Knuettel invited you to join him on Google+ - Marco Knuettel invited you
Ich, Stefan (2) - Wo braucht du mich morgen? - Office melde mich später. Von
Google Analytics - Erste Ausgabe des Google Analytics-Benchmarking-Berichts
Ich, Florian, Lothar (3) - Fwd: CPSA Prüfung - Glückwunsch und schönen Urlaub, Herr
Wetter
Nürnberg
15°C
Aktuell: Meistens bewölkt
Wind: W mit Windgeschwindigkeiten von 19 km/h
Feuchtigkeit: 79 %
Google Übersetzer
Zu übersetzenden Text hier eingeben
Englisch
Deutsch
Übersetzen
Google Übersetzer-Startseite | Wörterbuch-Startseite
Technologie
Abgeordnete fordern Recht auf Pseudonym
sueddeutsche.de - alle 81 verwandten »
Premieren-Ticket: iPhone 5 schon jetzt vorbestellen
Chip Online - alle 27 verwandten »
IFA: Schaulaufen der Tablet-Computer um Platz zwei
sueddeutsche.de - alle 13 verwandten »
Apps für iPhone und Android: ZDF-Mediathek jetzt auch auf Smartphones
FOCUS Online - alle 22 verwandten »
Das Doodle rockt: Google ehrt Freddie Mercury zum 65. Geburtstag
STERN.DE - alle 44 verwandten »
Jeder sechste in Deutschland war noch nie online

OpenSocial

- User Profile
- Social Graph
- Acivity Streams
- Gadgets

OpenSocial Gadget



Write **Gadgets**

- Google Docs
- Gmail - Kontext bezogene sidebar gadgets.
- Calendar - Gadgets für die Calendar sidebar.
- iGoogle - Gadgets für das Google „Portal“.

SOFEA für Mobile Apps

- Native Clients
  - Betriebssystemabhängig (Kosten, Skills)
- Mobile Websites (optimierte Websites für Mobile)
- **Hybrid Mobile Apps**

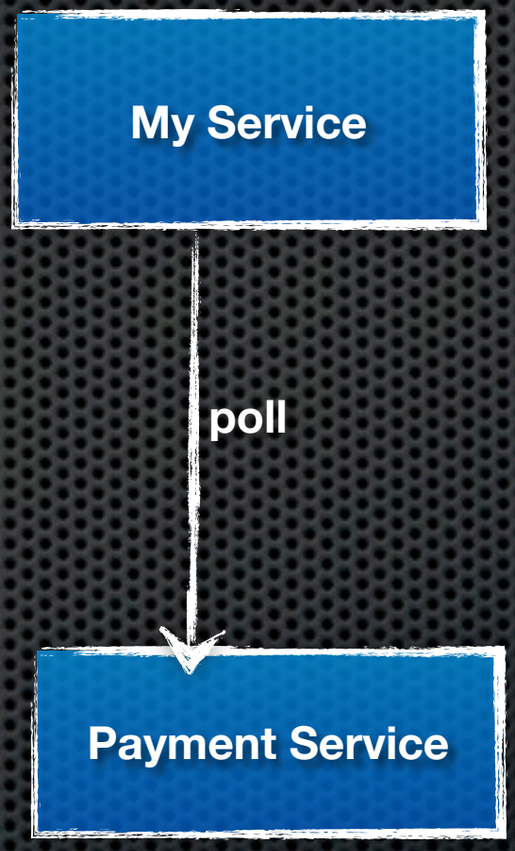
Single Source - Hybird Apps

HTML

Sencha

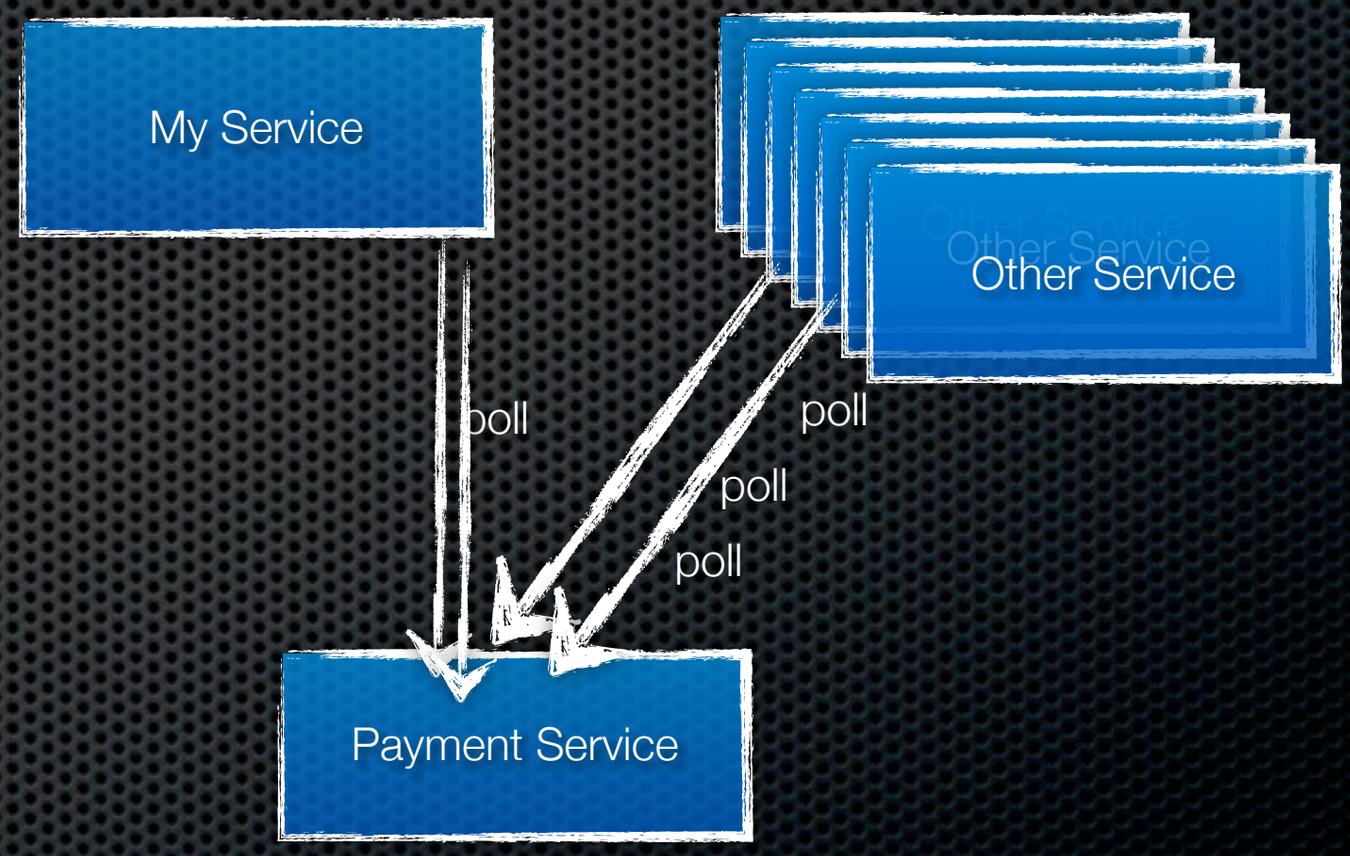
jQuery
mobile framework

WebHooks
Callbacks im Web

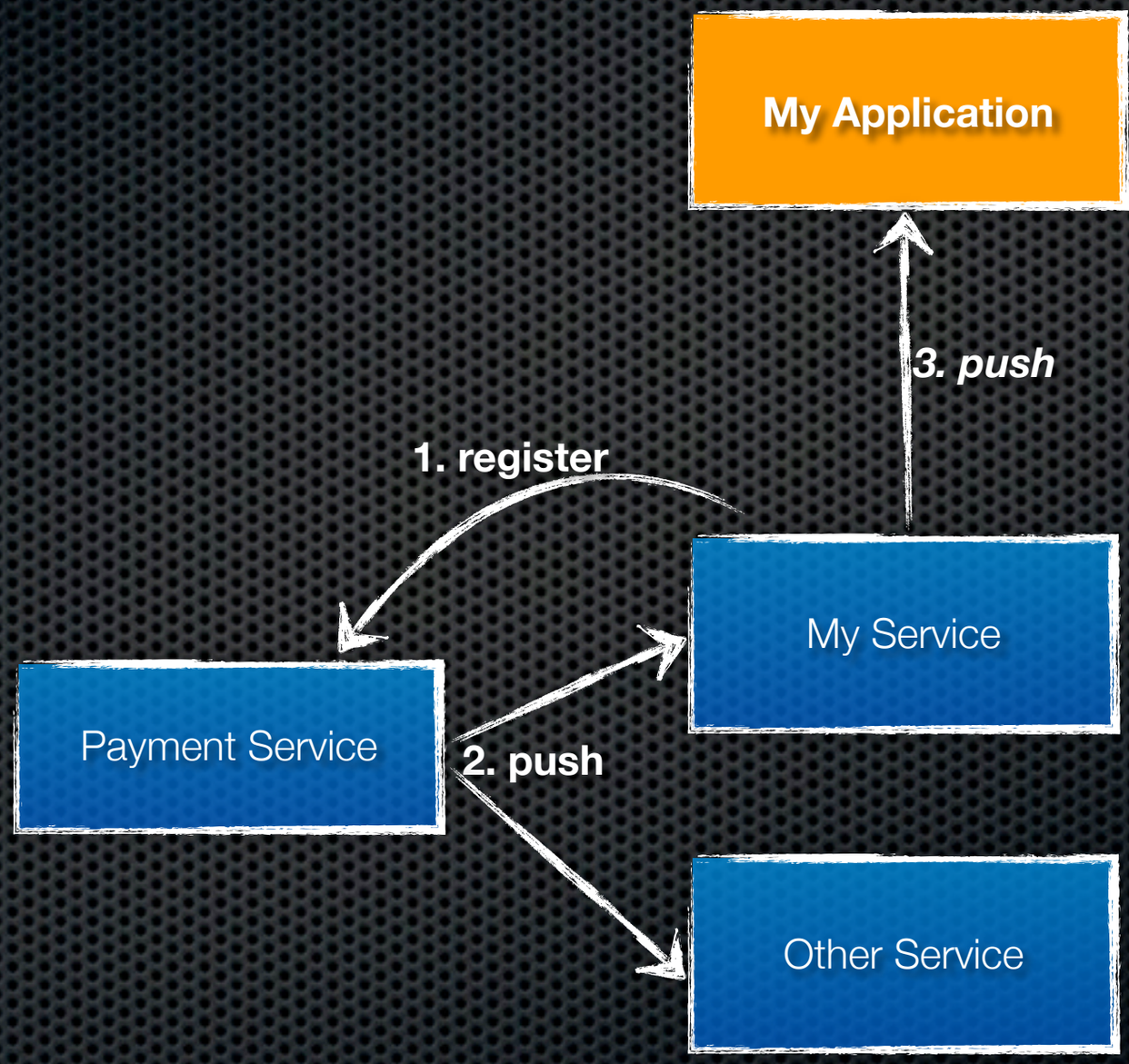


Webhooks Das Problem...

- Poll ist ineffizient
- hohe Latenz

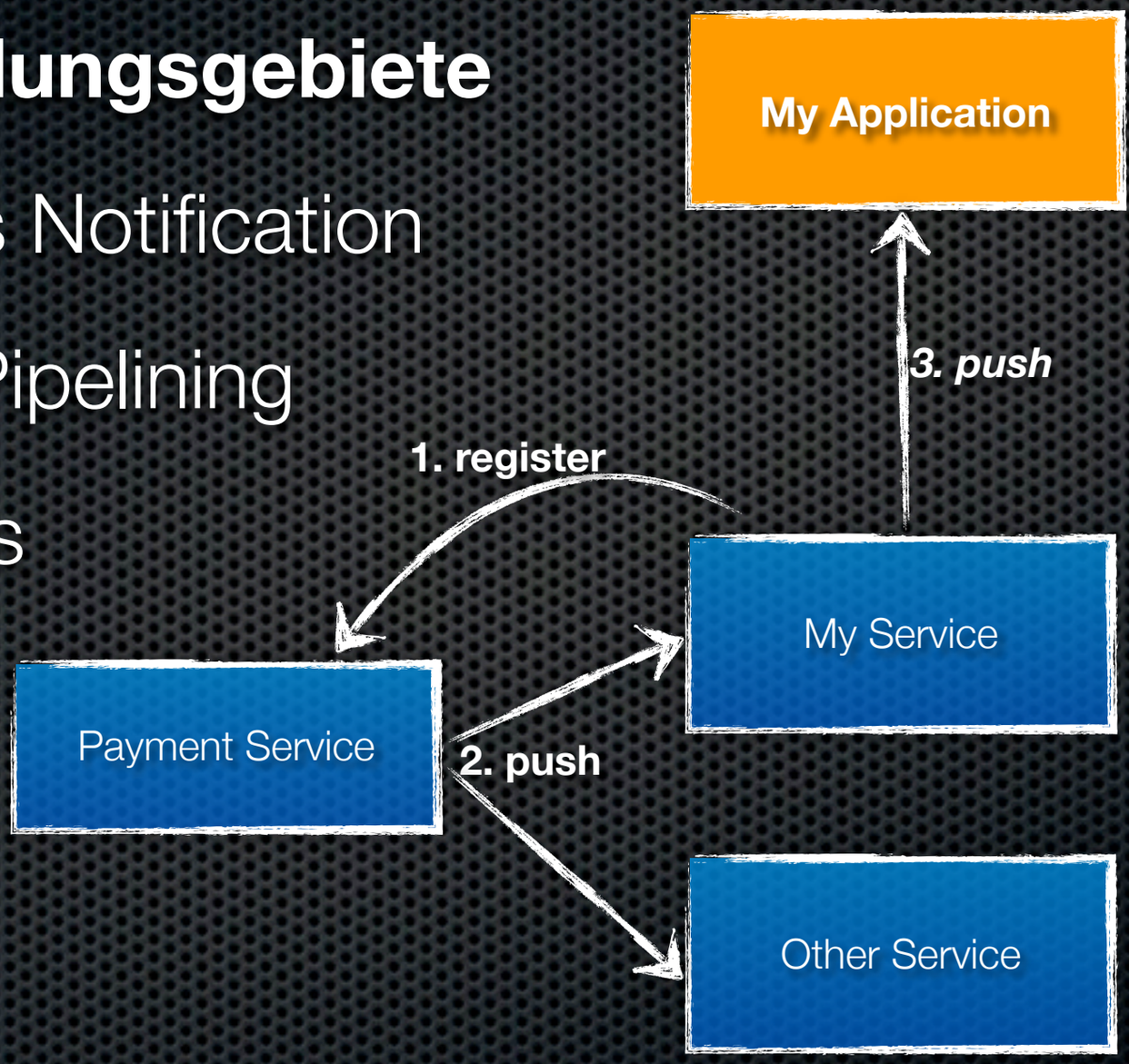


Webhooks - Die Lösung



Webhooks - Die Lösung

- **Anwendungsgebiete**
  - Events Notification
  - Data Pipelining
  - Plugins



Web Intends

My Application

Bild auswählen

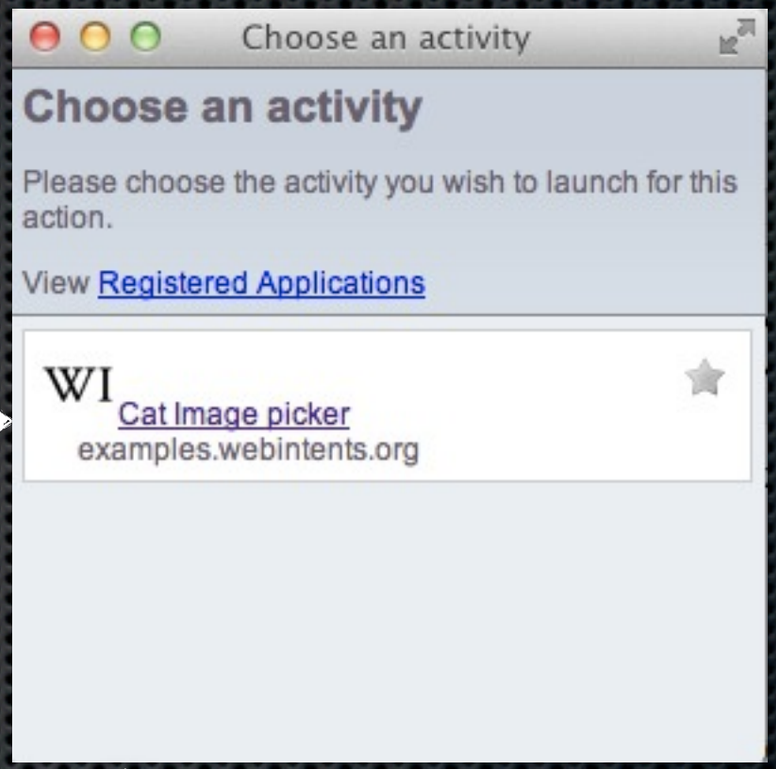


Bild aus Flicker übernehmen

Code Beispiel
Web Intends

Registrieren

```
<intent
  action="http://webintents.org/share"
  type="image/*"
  href="share.html"
/>
```

Feuern:

```
var intent = new Intent("http://webintents.org/share",
  "text/uri-list",
  "http://news.bbc.co.uk");

window.navigator.startActivity(intent);
```

High Scalable
Full JavaScript Stack

Mein Favorit :)

5.– 8. September 2011
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vielen Dank!

Sandro Sonntag

Adorsys GmbH & Co KG