

# WEB UI

## Entwicklung JavaScript basierter Anwendungen



# Konzepte + Technologien

## ■ Web 1.0 - Klassisch

- HTML wird dynamisch am Server erzeugt und an den Browser ausgeliefert
- Technologien: JSP, JSF, XSLT ... (Urvater CGI)

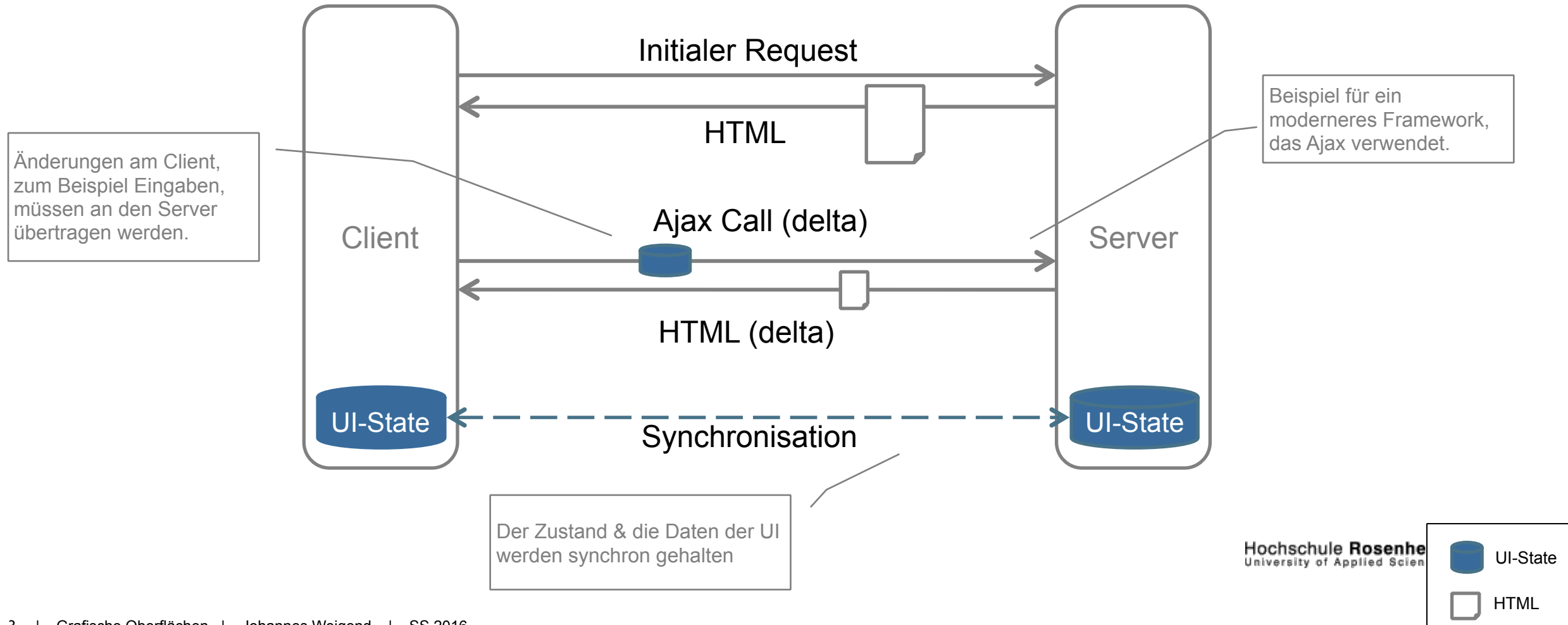
## ■ Web 2.0 - AJAX

- HTML wird dynamisch am Server erzeugt und an den Browser ausgeliefert
- Die Seite kommuniziert per JavaScript direkt dem Server
- Der Seitenablauf (Pageflow) wird vom Server gesteuert
- Technologien: JSF Komponenten wie Primefaces / MyFaces

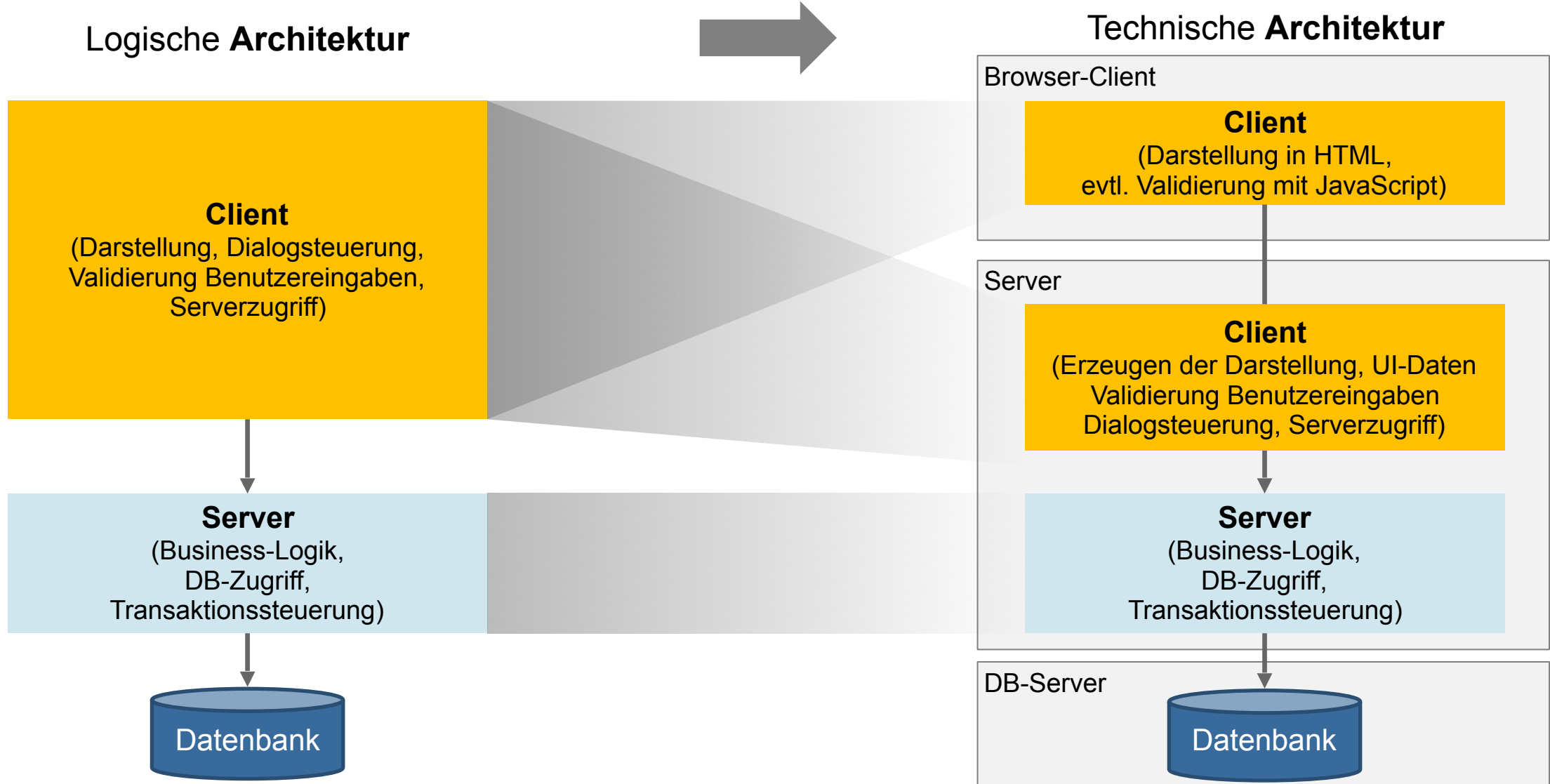
## ■ Rich-Clients

- HTML wird am Client per JavaScript zusammengebaut (Rendering)
- Die Client/Server Schnittstelle basiert auf JSON oder XML

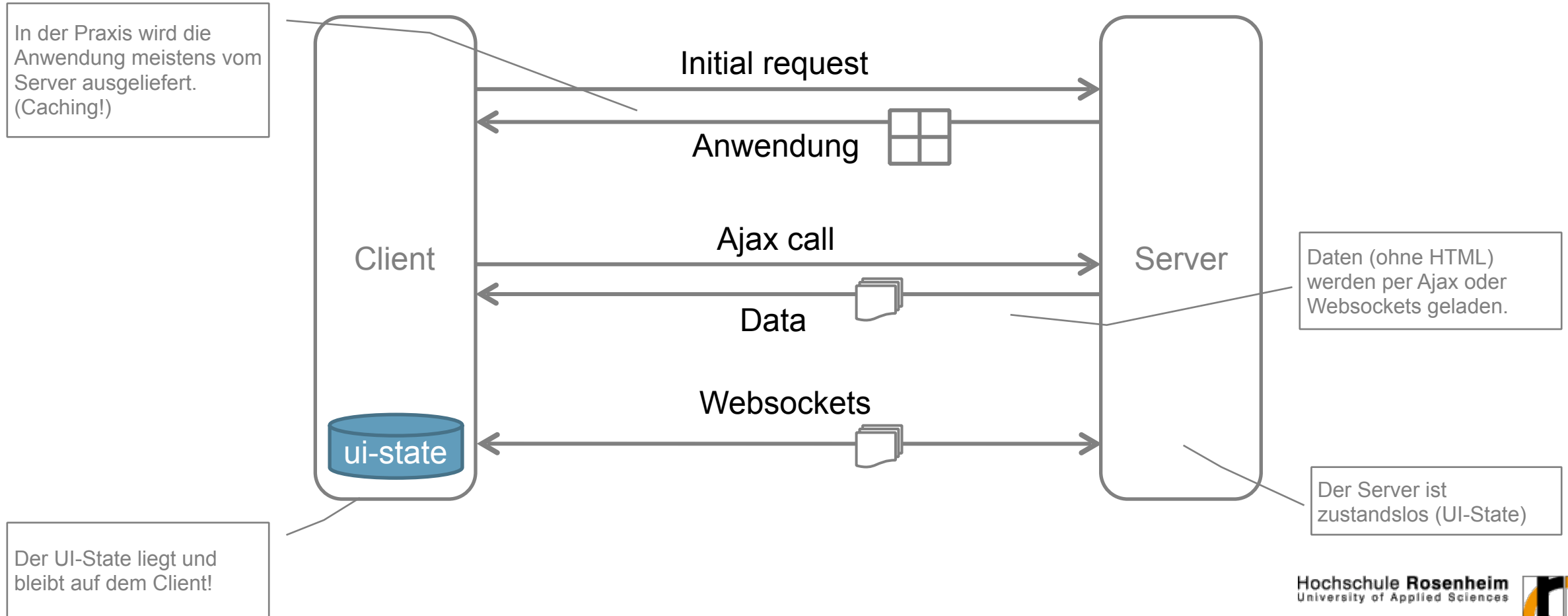
# Beim Web 2.0 Framework wird HTML auf dem Server erzeugt und an den Client ausgeliefert.



# Bei Web 2.0 Frameworks ist der Client oft unnatürlich aufgeteilt.



# Beim Web-Rich-Client liefert der Server die Daten ohne HTML-Gerüst aus, zum Beispiel im JSON-Format.



# Web-Rich-Clients haben Vor- und Nachteile.

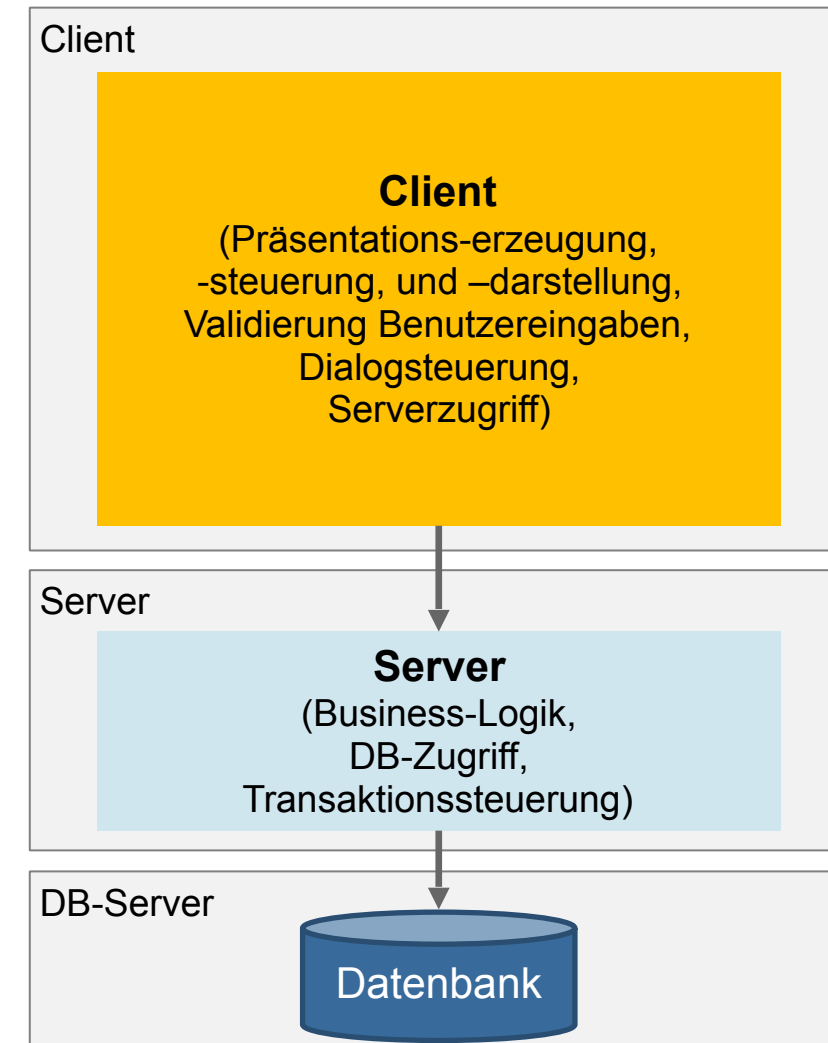
## Vorteile

- Gute User Experience
- Gute Testbarkeit des Servers (Schnittstellen & Stateless)
- Stateless Server & keine Synchronisation des Zustands
- Klarer Client-Server-Schnitt (zum Beispiel mit REST-Services)

## Nachteile

- Ggf. unterschiedliche Technologien in Client <-> Server (zum Beispiel Java & JavaScript).
- Das Bootstrapping bis zur ersten Anzeige ist oft langsamer.
- Die Werkzeugkette ist noch nicht so ausgereift (Build, Metriken, etc.).
- **Die Frameworks sind oft nicht standardisiert und ändern sich!**  
(„Wilder Westen“ - JWE)

**Kenne die Vor-/Nachteile und treffe für dein Projekt eine begründete Entscheidung!**



# Alle heutigen Rich Client Web Technologien basieren auf JavaScript.

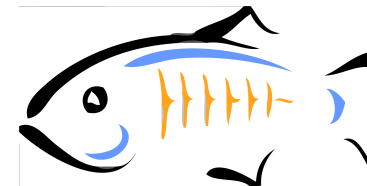
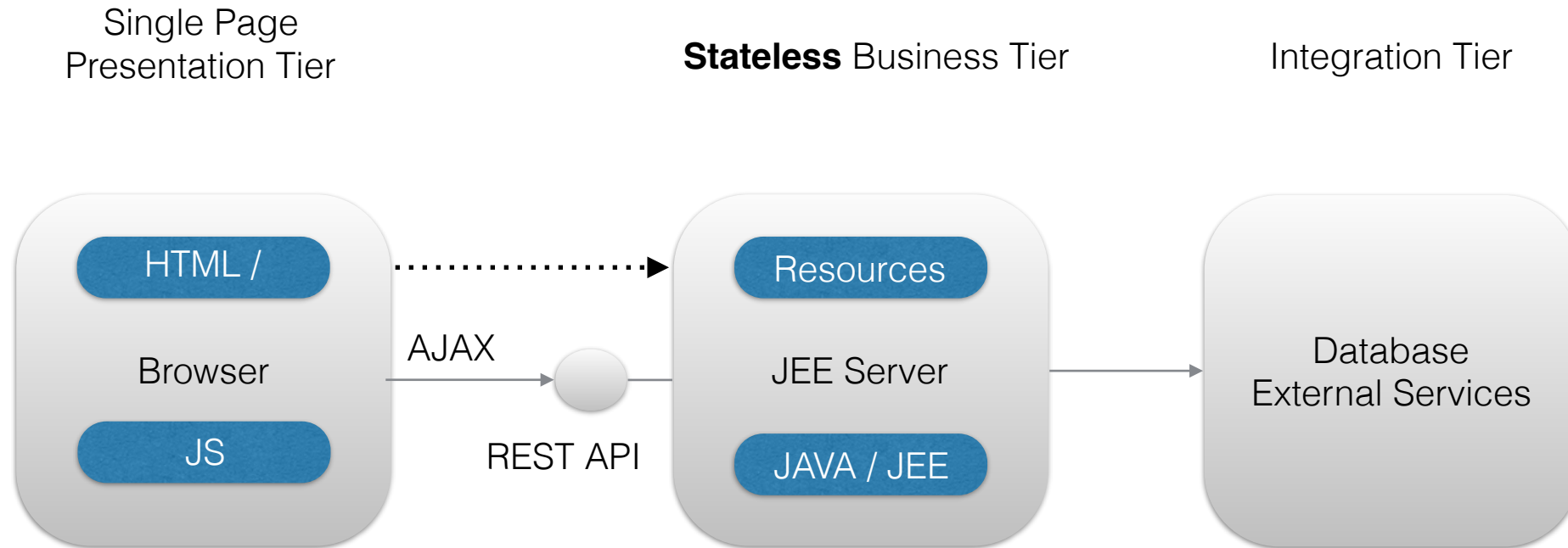
- Lebendige Technologien (HTML5 / JS only)
  - AngularJS u.a.
  - GWT, Vaadin
- Tote Technologien (Plugin notwendig)
  - Applets, .NET Smart Client
  - Adobe Flash / Flex
  - JavaFX per Webstart

# Das JavaScript Ökosystem

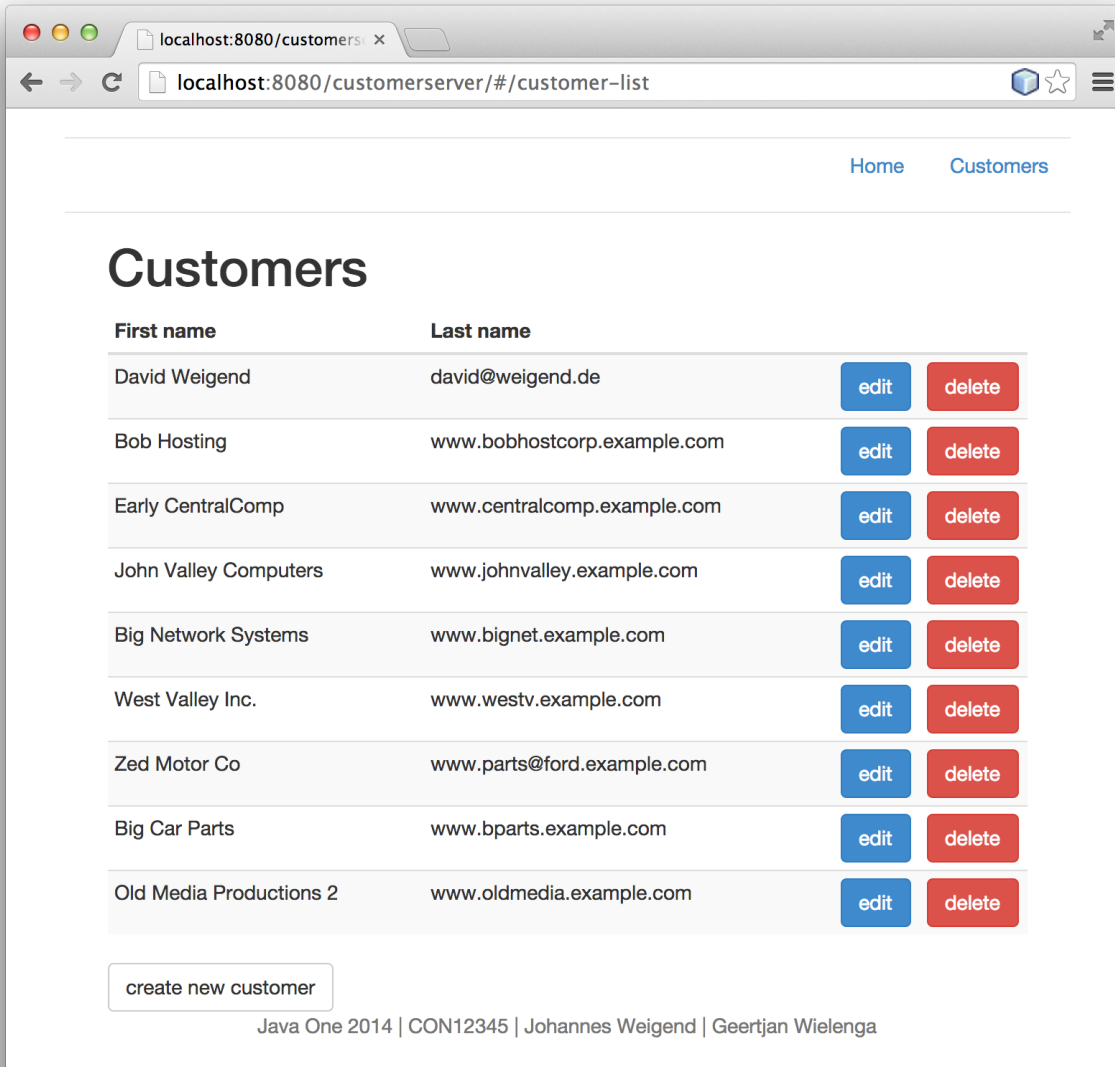
<i>JavaScript</i>	<i>Java</i>	<i>Beschreibung</i>
<i>Grunt, Gulp</i>	<i>Ant, Maven, Gradle</i>	<i>Build-Automation</i>
<i>Bower</i>	<i>Maven Repository</i>	<i>Dependency- Management</i>
<i>Jasmine</i>	<i>JUnit</i>	<i>Unit-Testing</i>
<i>NodeJS</i>	<i>JSE / JEE</i>	<i>Platform</i>
<i>AngularJS</i>	<i>JavaFX</i>	<i>UI Platform</i>



# Single Page JEE Application



# Customer CRUD HTML5 Application



localhost:8080/customers x

localhost:8080/customerserver/#/customer-list

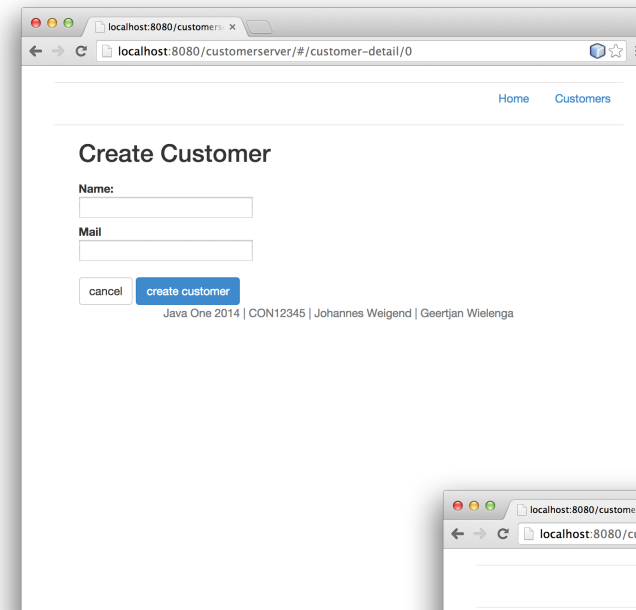
Home Customers

## Customers

First name	Last name		
David Weigend	david@weigend.de	edit	delete
Bob Hosting	www.bobhostcorp.example.com	edit	delete
Early CentralComp	www.centralcomp.example.com	edit	delete
John Valley Computers	www.johnvalley.example.com	edit	delete
Big Network Systems	www.bignet.example.com	edit	delete
West Valley Inc.	www.westv.example.com	edit	delete
Zed Motor Co	www.parts@ford.example.com	edit	delete
Big Car Parts	www.bparts.example.com	edit	delete
Old Media Productions 2	www.oldmedia.example.com	edit	delete

create new customer

Java One 2014 | CON12345 | Johannes Weigend | Geertjan Wielenga



localhost:8080/customers x

localhost:8080/customerserver/#/customer-detail/0

Home Customers

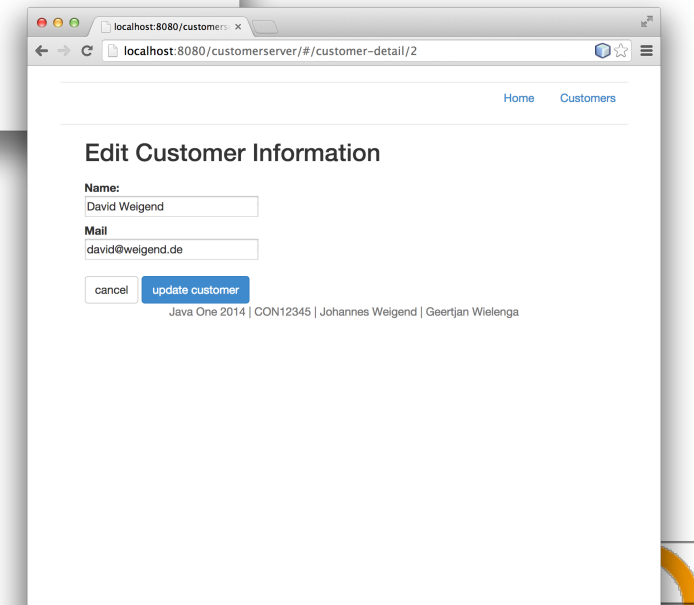
## Create Customer

Name:

Mail:

cancel create customer

Java One 2014 | CON12345 | Johannes Weigend | Geertjan Wielenga



localhost:8080/customers x

localhost:8080/customerserver/#/customer-detail/2

Home Customers

## Edit Customer Information

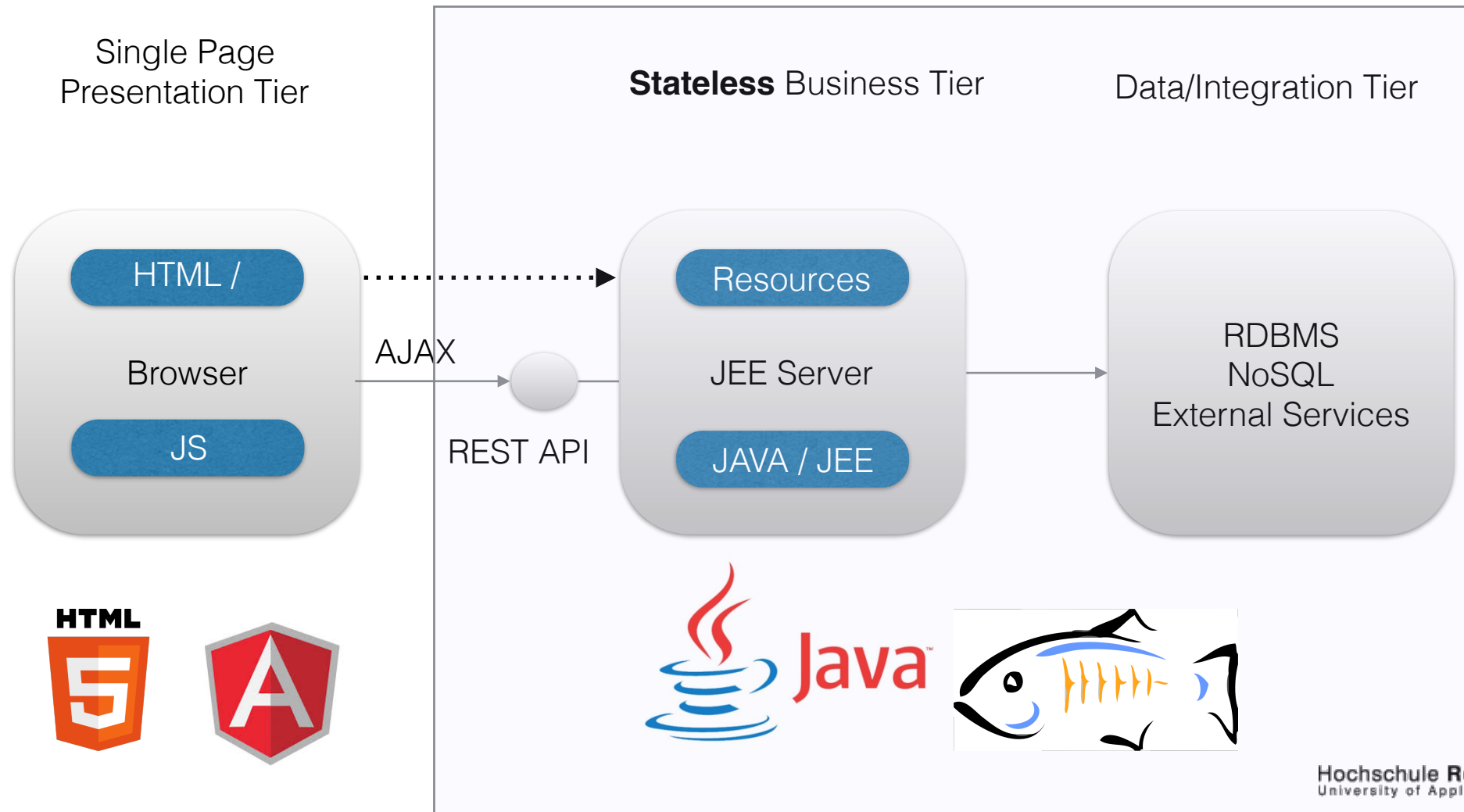
Name:

Mail:

cancel update customer

Java One 2014 | CON12345 | Johannes Weigend | Geertjan Wielenga

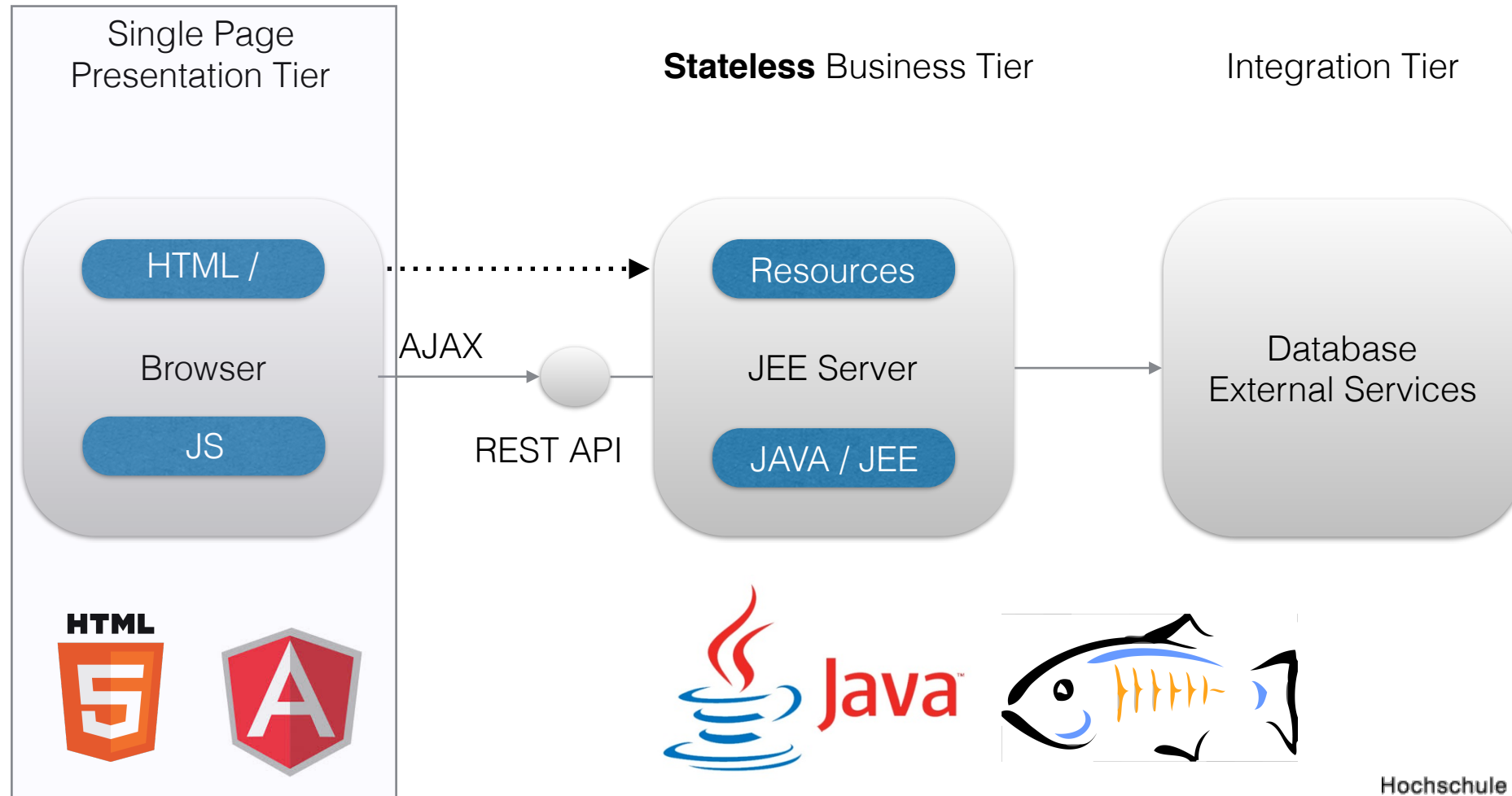
# Step 1: Implementing a REST CRUD API with JEE7



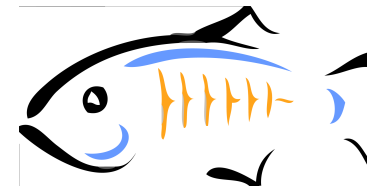
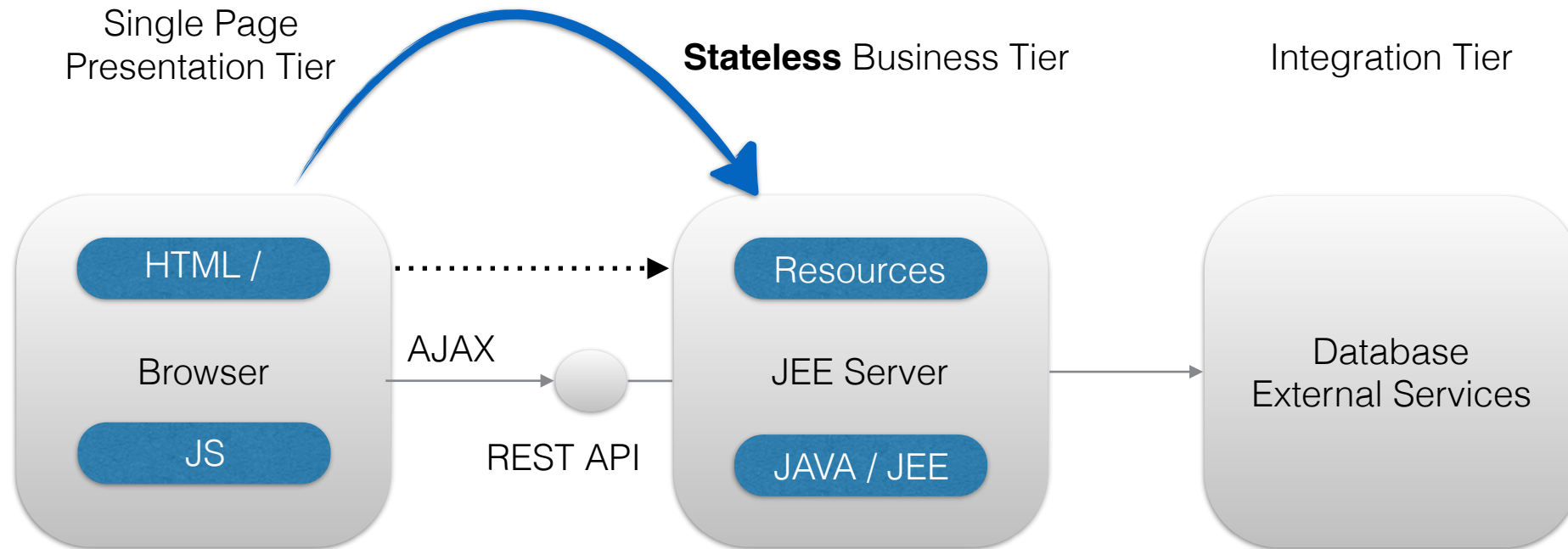
Hochschule Rosenheim  
University of Applied Sciences



# Step 2: Implementing a AngularJS CRUD Client



# Step 3: Deliver as single WAR



http://www.screencast.com/t/BIGU0kuj

https://github.com/jweigend/JEEAngularCRUD/  
@JohannesWeigend

