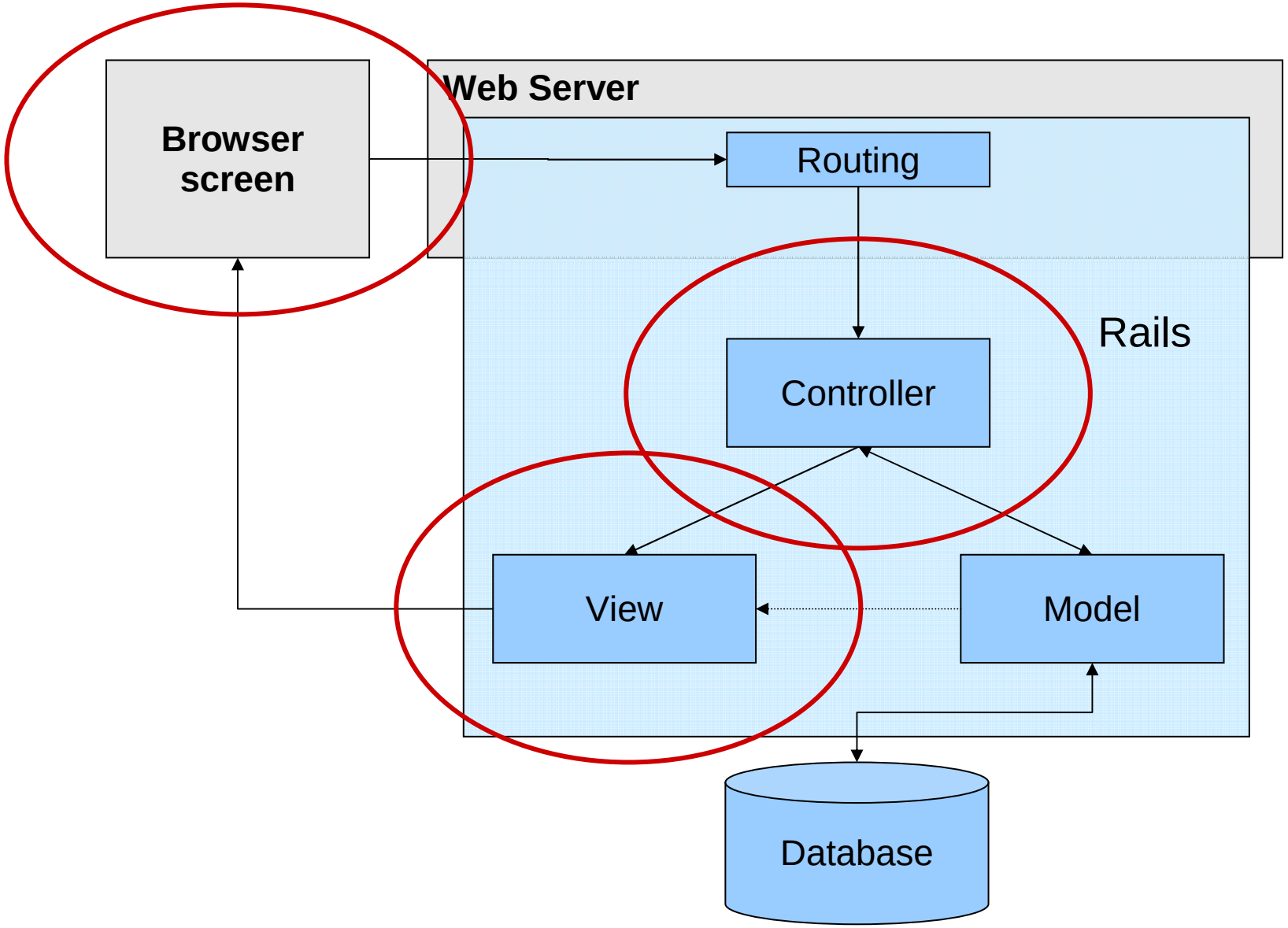
7363 - Web-basierte Anwendungen
4750 – Web-Engineering

Eine Vertiefungsveranstaltung

AJAX ***!=***

A*synchronous* ***J****avaScript* ***a****nd* ***X****ML*

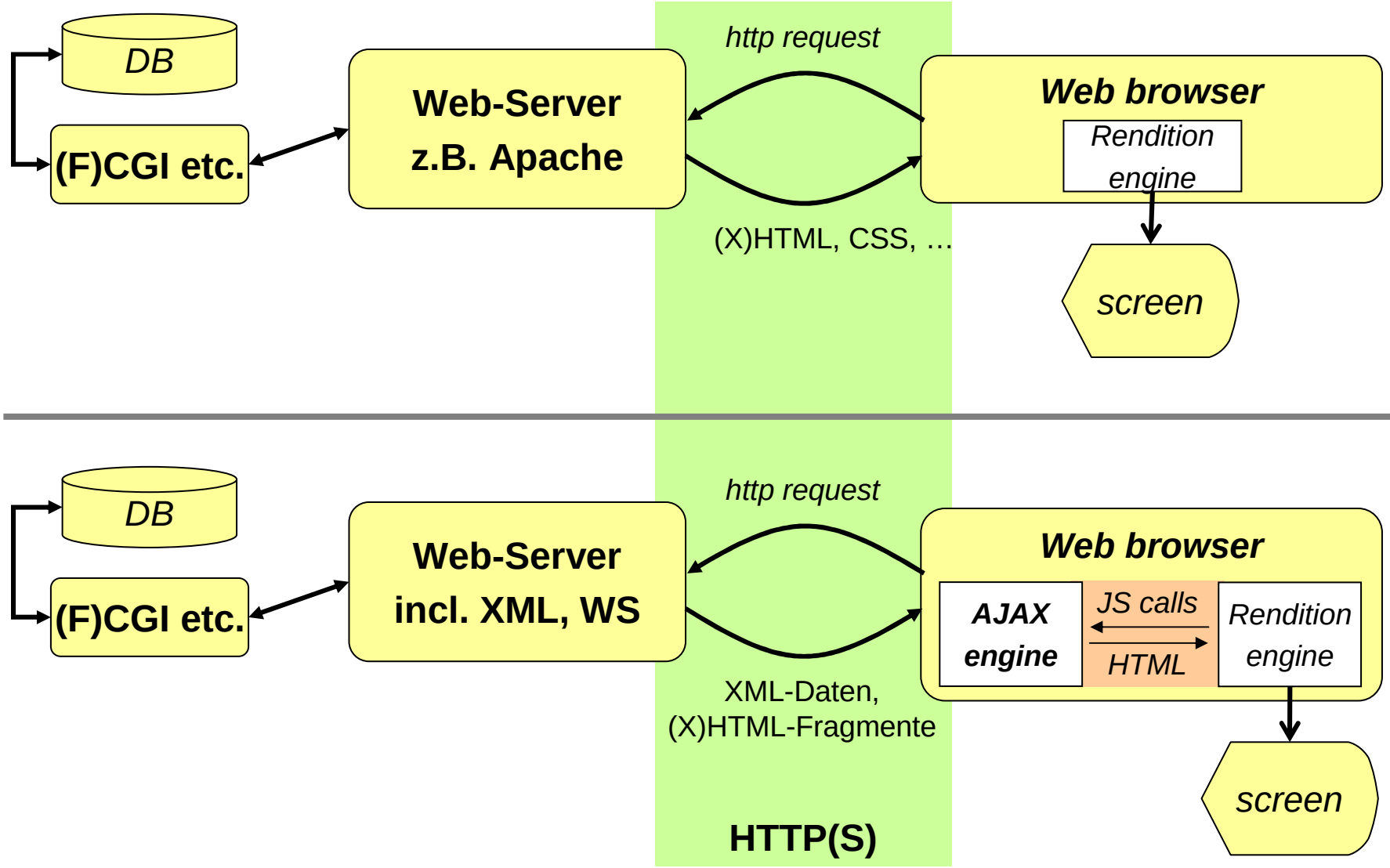
Interaktivere Benutzerschnittstellen



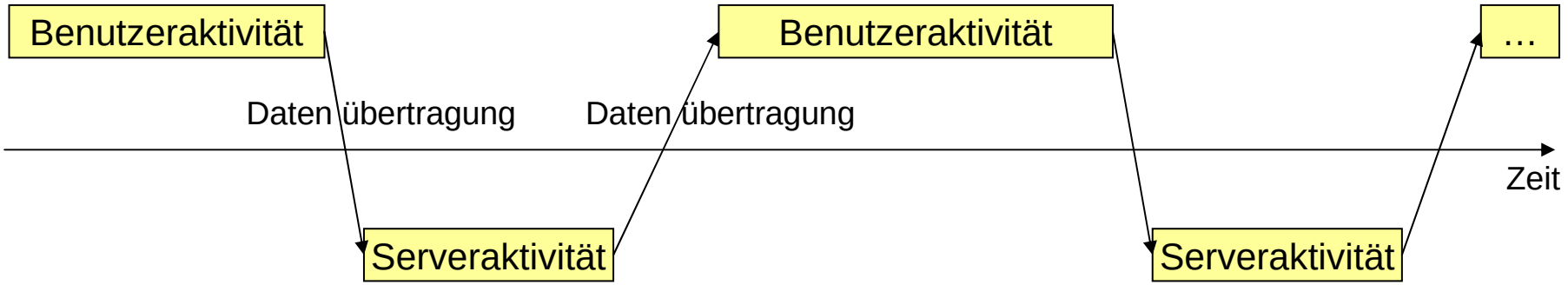
Ajax

- Ajax – Eine Mischung bekannter Techniken:
  - Präsentation von Information auf der Basis von Standard, insbesondere von **XHTML** und **CSS**
  - Dynamische Anzeigen und Interaktion mit den Inhalten mittels **DOM** (*Document Object Model*)
  - Datenrepräsentation und –transformation mit **XML** and **XSLT**
  - Asynchroner Datenaustausch mit **XMLHttpRequest**
  - und **JavaScript**, um all dies zu verbinden

WBA: Ajax vs. traditionelle Interaktionen



Ajax: Synchrone Datenübertragung



Traditionelle WBA wechseln zwischen Benutzer- und Server-Aktivitäten

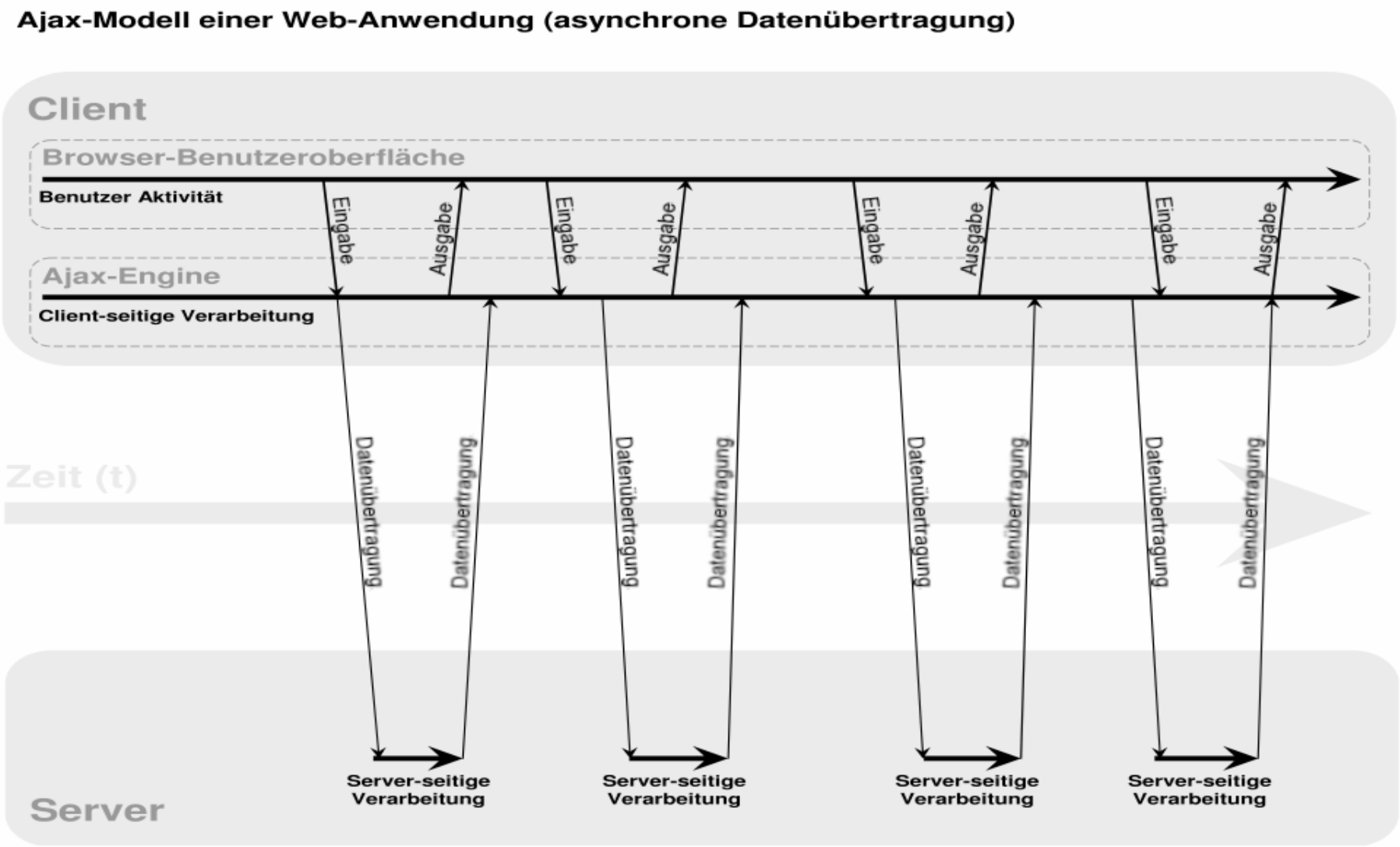
Anwender empfinden die entstehenden Wartezeiten als störende Unterbrechungen ihres Arbeitsflusses.

Ajax: Alternative Implementierungen

- Direkte Ajax-Implementierung
  - *Client* besitzt API zur XML-basierten Kommunikation mit dem Server (XMLHttpRequest, evtl. mit SOAP-Inhalten)
  - Datenaustausch effizient und flexibel,
  - Transformation erforderlich, komplexer *client*

- Indirekte Ajax-Implementierung
  - Client tauscht HTML-Fragmente mit Server aus
  - Client aktualisiert Darstellung mittels DOM
  - Beispiele:
    - Einfügen / Aktualisieren von Listeneinträgen,
    - Positionen eines Warenkorbs, einer Bestellung etc.
  - Ein solcher Mechanismus ist bereits in Rails implementiert.

Ajax



Quelle: de.wikipedia.org

Ajax

- Beispielanwendungen

  – *Google Maps:* http://maps.google.de
    - „Gleitende“ Verschiebung des Sichtbarkeitsfensters auf der Karte,
    - vorausschauendes Nachladen der nächsten Kacheln im Hintergrund
    - Ähnliche Wirkung mit Java Applets: Stadtplan Wiesbaden.
    - Gegenbeispiel: http://stadtplan.frankfurt.de

  – *Google Suggest:* http://www.google.com/webhp?complete=1&hl=de
    - Sofort angebotene Auswahlliste von Suchbegriffen allein aufgrund der bisher eingetippten Zeichen.
    - Inzwischen leistet bereits das Suchfeld im Firefox Ähliches!

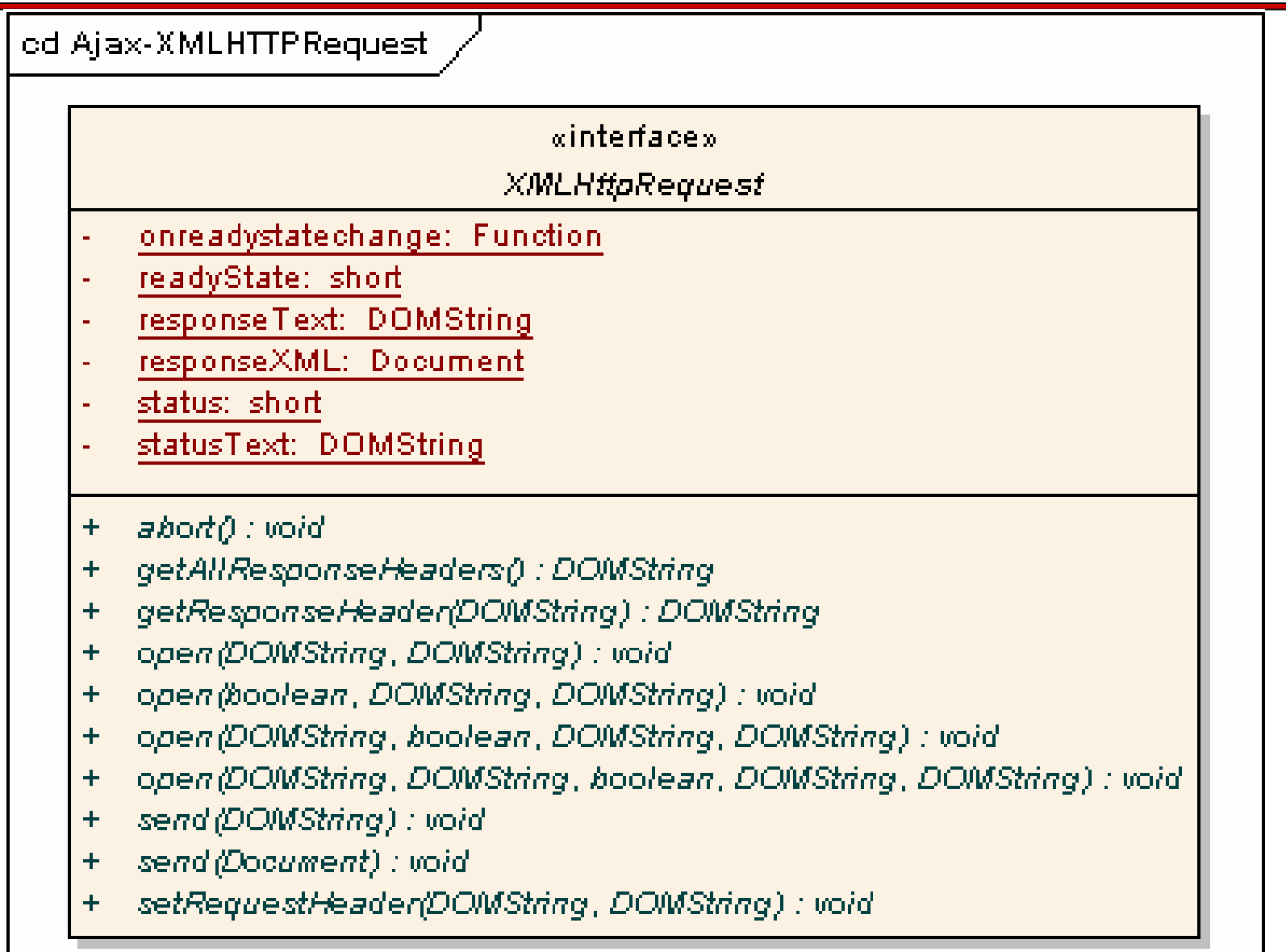
Ajax: XMLHttpRequest

- **XMLHttpRequest: Details**
  - Aktuelle Quelle: W3C-Entwurf vom 15. April 2008,
    `http://www.w3.org/TR/XMLHttpRequest/`
  - Code-Beispiel:

```
var xmlHttp = new XMLHttpRequest();
if (xmlHttp) {
  xmlHttp.open('GET', 'beispiel.xml', true);
  xmlHttp.onreadystatechange = function () {
    if (xmlHttp.readyState == 4) {
      alert(xmlHttp.responseText);
    }
  };
  xmlHttp.send(null);
}
```

- Kleine JS/DOM-Demo

Ajax: XMLHttpRequest



Quelle: de.wikipedia.org

Ajax: XMLHttpRequest

```
interface XMLHttpRequest {
                attribute EventListener onreadystatechange;
    readonly    attribute unsigned short readyState;
    void        open(in DOMString method, in DOMString url);
    void        open(in DOMString method, in DOMString url,
                     in boolean async);
    void        open(in DOMString method, in DOMString url, in boolean async,
                     in DOMString user);
    void        open(in DOMString method, in DOMString url, in boolean async,
                     in DOMString user, in DOMString password);
    void        setRequestHeader(in DOMString header, in DOMString value);
    void        send();
    void        send(in DOMString data);
    void        send(in Document data);
    void        abort();
    DOMString   getAllResponseHeaders();
    DOMString   getResponseHeader(in DOMString header);
    readonly attribute DOMString responseText;
    readonly attribute Document responseXML;
    readonly attribute unsigned short status;
    readonly attribute DOMString statusText;
};
```

Quelle: http://www.w3.org/TR/XMLHttpRequest/

Ajax

- Problemgebiete
  - Verletzung des seitenorientierten Aufbauprinzips
    - Durch das dynamische Verhalten von AJAX-Anwendungen funktionieren *Back button* und Lesezeichenverwaltung des Browsers nicht mehr (bzw. nicht mehr wie erwartet).
    - Problem ist analog zu früheren Problemen mit *frames*
    - Wahrnehmung dynam. Änderungen innerhalb einer Seite durch die Anwender??
    - Aktueller Artikel dazu: Frank Puscher, Klarheit trotz Ajax, c't 2/2007.

    Auswege:
    - Beschränkung von AJAX-Funktionen auf (kleine) Funktionsgruppen innerhalb einer nach wie vor als „Seite" wahrgenommenen Einheit
    - Verwendung von *back button* und Lesezeichen zwischen diesen „Seiten", Verzicht auf diese Elemente innerhalb einer Gruppe.
    - Verlagern von Ajax-Aktivitäten in unsichtbare „iframes" in statischer HTML-Seite
    - Dynam. geänderte Seitenbestandteile (vorübergehend) farblich kennzeichnen.
  - Barrierefreies Internet?
    - Auch mit Ajax angereicherte Seiten sollten sich z.B. vorlesen lassen können

Ajax

- Problemgebiete (Forts.)
  - *Polling*-Problem
    - Web Server unterliegen dem C/S-Modell – sie können den Client nicht „zurückrufen“!
    - Asynchrones Verhalten des XMLHttpRequest-Objekts wird durch Nebenläufigkeit (*multi-threading*) erreicht. Diese zusätzlichen Threads existieren länger als bei normalen C/S-Anfragen und binden Ressourcen auf Client-Seite entsprechend länger.
    - Clients können durch ungeschicktes Vorgehen (häufiges *polling*) neue, erhebliche Serverlasten verursachen. Auch kann sich die Anzahl gleichzeitig offener TCP-Verbindungen des Servers erhöhen.
  - Gelegentlich störend:
    - Download der für Ajax benötigten JS-Bibliotheken: Dauer hinderlich?
    - JS aktiviert? Benötigte JS-Funktionen durch *Client* freigegeben?
    - Code-Weichen für Client-Abhängigkeiten, insb. MSIE vs. Firefox & Co.

Ajax

- Ajax und Mitbewerber um RIA-Technologien
  - Für die Umsetzung sogenannter *Rich Internet Applications* (RIAs) gibt es neben Ajax auch weitere Optionen:
  - Flash
    - Proprietäre, heute weit verbreitete Technik von Adobe Systems, Inc.
    - „ActionScript“ eröffnet ähnliche Möglichkeiten und Probleme
  - Silverlight
    - Microsofts RIA-Technologie, direkte Konkurrenz zu Adobe Flash
- Am Rande erwähnenswert:
  - Mozilla XUL
    - Eine XML-basierte GUI-Sprache für RIAs
  - Konkurrierender W3C-Standard:
    - DOM Level 3 Load & Save Specification, 7. April 2004 (Status: REC), siehe http://www.w3.org/TR/DOM-Level-3-LS, s.o.
    - Noch wenig verbreitet. Unterstützung durch Browser?

Ajax: Quellen

[1] http://de.wikipedia.org/wiki/Ajax_(Programmierung)

- Sehr kompetenter, gut verständlicher und weiterführender Übersichtsartikel!

[2] Jesse James Garrett: *Ajax: A New Approach to Web Applications.* Adaptive Path LLC, 18. Februar 2005, http://www.adaptivepath.com/publications/essays/archives/000385.php

- Der Artikel, der die Bezeichnung AJAX nachhaltig prägte.

[3] http://www.openajax.net/

- Beispiel- und Linksammlung

[3] Drew McLellan: *Very Dynamic Web Interfaces.* 9. Februar 2005, http://www.xml.com/pub/a/2005/02/09/xml-http-request.html

- Ein Artikel u.a. mit Details zum Umgang mit XMLHttpRequest

AJAX und Rails

Ajax und Rails

- Ajax-Unterstützung durch Rails
  - Integration der JS-Bibliotheken „Prototype“ und „Script.aculo.us“
    - Mehr oder weniger aktuelle Versionen dieser Bibliotheken sind Bestandteil von Rails-Projekten
      - Ihre Auslieferung an Clients muss gesondert freigegeben werden
    - Clientseitig werden deren APIs für Ajax u.a. genutzt
    - Serverseitig wird JS-Codegenerierung mit Ruby gekapselt
    - Erzeugung von JS-Code wird als dritte „Säule“ neben XHTML und XML vom Template-System unterstützt
  - Rails unterstützt AJAX i.w. indirekt
    - d.h. durch Manipulation des (X)HTML-Seitenaufbaus
  - Direkte Unterstützung leicht realisierbar
    - XML-Daten generieren (etwa mit „Builder“), XSLT-Code analog CSS

Ajax und Rails

- Vorbereitungen für den AJAX-Einsatz

  – Auslieferung der JS-Bibliotheken konfigurieren
    - In app/views/**layouts/application**.html.erb (oder spezifischer):

```
<%= javascript_include_tag "defaults" %>
```

(liefert alle (beide) JS-Bibliotheken aus), oder:

```
<%= javascript_include_tag "prototype" %>
```

(liefert nur die Prototype-Bibliothek aus, etc.)

  – Bitte Bandbreiten und Client-Last beachten – am besten nur laden, wenn benötigt

Ajax und Rails

- Zwei typische Beispiele für AJAX-Einsatz:
  - *auto-completion*, vgl. *Google suggest*-Beispiel
  - Kontext-gerechte *selection box*-Inhalte

- Kern-Idee beider Lösungen
  - Die ausgelieferte HTML-Seite besitzt ein HTML-Element, dessen Inhalt per DOM ausgetauscht werden soll
  - Der neue Inhalt wird von einem AJAX-Call angefordert und einer geeigneten Server-Aktion geliefert (mit oder ohne Schablone)
  - Die Triggerung eines AJAX-Calls geschieht Client-seitig situationsgerecht:
    - *Auto-completion*: Eingabe der ersten Zeichen in Textfeld
    - *Selection box*: Auswahl des Eintrags einer vorgeschalteten *selection box*

Ajax und Rails

- *Selection boxes* mit dynamischem Inhalt
  - Situation: Frühe Eingaben (mit *radio buttons, selections, text fields*, …) auf einem Formular verändern oder bestimmen die Wahlmöglichkeiten in später zu bedienenden *selection boxes*
  - Beispiel: Download von Bedienungsanleitungen
    - Festlegung Sprache
    - Festlegung Produktgruppe

    →Liste der auswählbaren Produkte

    →Liste der verfügbaren Dokumente zu einem Produkt (Benutzerhandbuch, Kurzanleitung, technisches Handbuch, Zeichnungen, …)

    →Eigentlicher Download
  - Eingesetzte Mittel:
    - JS-Bibliothek „Prototype“
    - Rails-Helper „form_remote_tag“, „remote_function“ (in „select“)
    - div-Container mit id-Attribut im HTML-Code, wo Inhalte zu ersetzen sind
    - XHR *action* in Controller zur Generierung von HTML-Code und JS-Code zu dessen clientseitigem Einbau via DOM, hier: „option“-Elemente in „selection“

Einzelheiten: Siehe AoR-Demo

Ajax und Rails

- *Auto-completion*
  - Situation: Analog „*Google Suggest*“ – Ein Texteingabefeld reagiert bereits auf erste Zeicheneingaben mit einer Liste möglicher vervollständigter Texte. Statt vollständiger Eingabe genügt Auswahl per Anklicken.
  - Beispiel: Suche nach Dozent(en), siehe AoR-Demo
  - Eingesetzte Mittel:
    - JS-Bibliothek „Prototype“ mit „Scipt.aculo.us“-Ergänzungen
    - Rails-Helper „form_remote_tag“, „text_field_with_auto_complete“ oder „auto_complete_field“
    - div-Container mit id-Attribut im HTML-Code, wo Inhalte zu ersetzen sind
    - XHR *action* in Controller zur Generierung von HTML-Code und JS-Code zu dessen clientseitigem Einbau via DOM. Hier: Aufbau eines „ul“-Elements
    - Zusätzlich meist CSS-Code zur besonderen Gestaltung derartiger „ul“-Elemente

Ajax und Rails

- Dokumentation
  - Noch sehr dürftig!
    - Leider noch kein „*guide*“ (guides.rubyonrails.org)
    - api.rubyonrails.org: Unvollständig (?). Siehe ActionView::Helpers::X, mit X aus {JavaScriptHelper, PrototypeHelper*, ScriptaculousHelper}
  - Noch am besten: Agile Web Dev. With Rails (2nd ed.), ch. 23 „The Web 2.0“
  - Ruby on Rails 2: Kap. 13 (Ajax on Rails)

Ajax und Rails

- Sicherheitsfragen
  - XmlHttpRequest-Aufrufe können vom Anwender unbemerkt ablaufen
  - JS kann dabei auch Einfluss auf HTTP *header* nehmen, z.B.
    - nicht nur GET, sondern auch POST-Requests ausführen
    - Cookies auslesen bzw. setzen
  - Insgesamt:
    - JS-Code in einem Browser kann dank XHR beliebige komplexe Angriffe vai HTTP organisieren und ohne Notiz durch den Anwender ausführen
    - Einschleusung von JS-Schadcode wird durch XHR noch kritischer

- Am Rande erwähnt
  - Bei Flash ist die Situation ähnlich …

Ajax-Einsatz im Projekt

- Mögliches Einsatzgebiet
  - *Ergebnislisten-Interface, Suche nach Name*
    - Eingabefeld für Name, „Suchen“-Button
    - Tabelle oder Textfeld mit den ersten ca. 10 Treffern
    - Mit jedem eingetippten Zeichen wird die Trefferliste aktualisiert:
    - Anzeige = Die ersten 10 Namen, die mit den eingetippten Zeichen beginnen!
  - *Ergebnislisten-Interface, Suche nach Verein/Ort*
    - Analog zur Namenssuche
- Hinweise
  - Realisieren Sie mindestens einmal eine AJAX-Funktionalität!
  - Selbst die großen Stadtmarathon-Seiten bieten diesen Komfort bisher nicht!