



Achilles and Ajax Vase Picture from History 201 www.maxwell.syr.edu

AJAX / AJAX Libraries und Frameworks

Moderne Webtechnologien - SoSe 07 - Web 2.0

Max Bureck, Steffen Glied, Onur Kilic, Jia Xu

1. Prinzipien zu AJAX
2. Framework: JSON-RPC-Java
3. Framework: Prototype
4. Kritikpunkte und Best Practices
5. Zukunft von AJAX



Achilles and Ajax Vase Picture from History 201 www.maxwell.syr.edu

Prinzipien zu AJAX

1. Was ist AJAX?
2. Woraus besteht AJAX?
3. Was soll AJAX bezwecken?
4. Klassische Webanwendungen
5. AJAX-Anwendungen
6. Klassische Anwendungen vs. AJAX
7. XMLHttpRequest
8. Entwicklung von AJAX
9. Libraries oder Frameworks?
10. Bekannte Frameworks



Portrait of Jesse James Garrett www.jjg.net

Asynchronous Javascript and XML = **AJAX**

- „AJAX“ wurde 2003 erstmalig von **Jesse James Garret** in „**Ajax: A New Approach to Web Applications**“ benutzt
- Keine neue Technologie, sondern eine Sammlung von bereits bestehen Techniken
- Strikt ist AJAX eine abstrakte **Architektur-Idee**

2. Woraus besteht AJAX?

Eine AJAX-Architektur enthält Komponenten für

- Visuelle Darstellung von Daten
- Dynamische Modifikation von Inhalten
- Asynchroner Datenverkehr

Meistens werden

- XHTML & CSS über das **DOM**
- JavaScript als ein wichtiger Vertreter der **ECMA-Scripts**
- und als Kernkommunikationsobjekt das **XMLHttpRequest**-Objekt

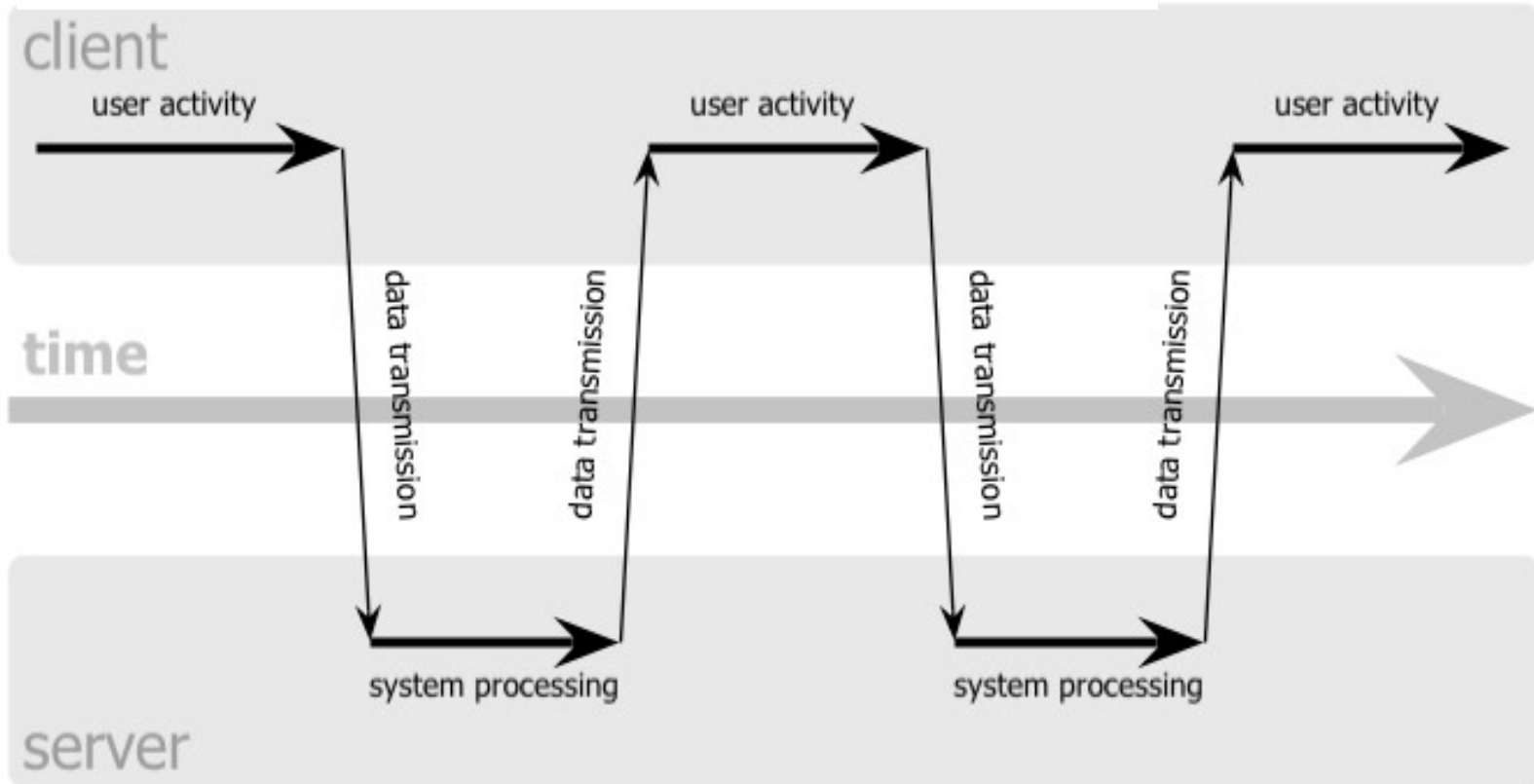
benutzt.

3. Was soll AJAX bezwecken?

Der Sinn einer solchen Architektur:

- **Standardisierte Darstellung** von festen Datenformaten (z.B. XHTML & CSS)
- Dynamisch **Daten nachzuladen**
- **Datenmengen gering** zu halten
- **Trennung** von tiefergehender Logik und Präsentation
 - ◇ **Formalisierung** von Web-Anwendungen auf das **Niveau von Desktop-Applikationen**

4. Klassische Webanwendungen



Ajax: A New Approach to Web Applications
www.adaptivepath.com

4. Klassische Webanwendungen (II)

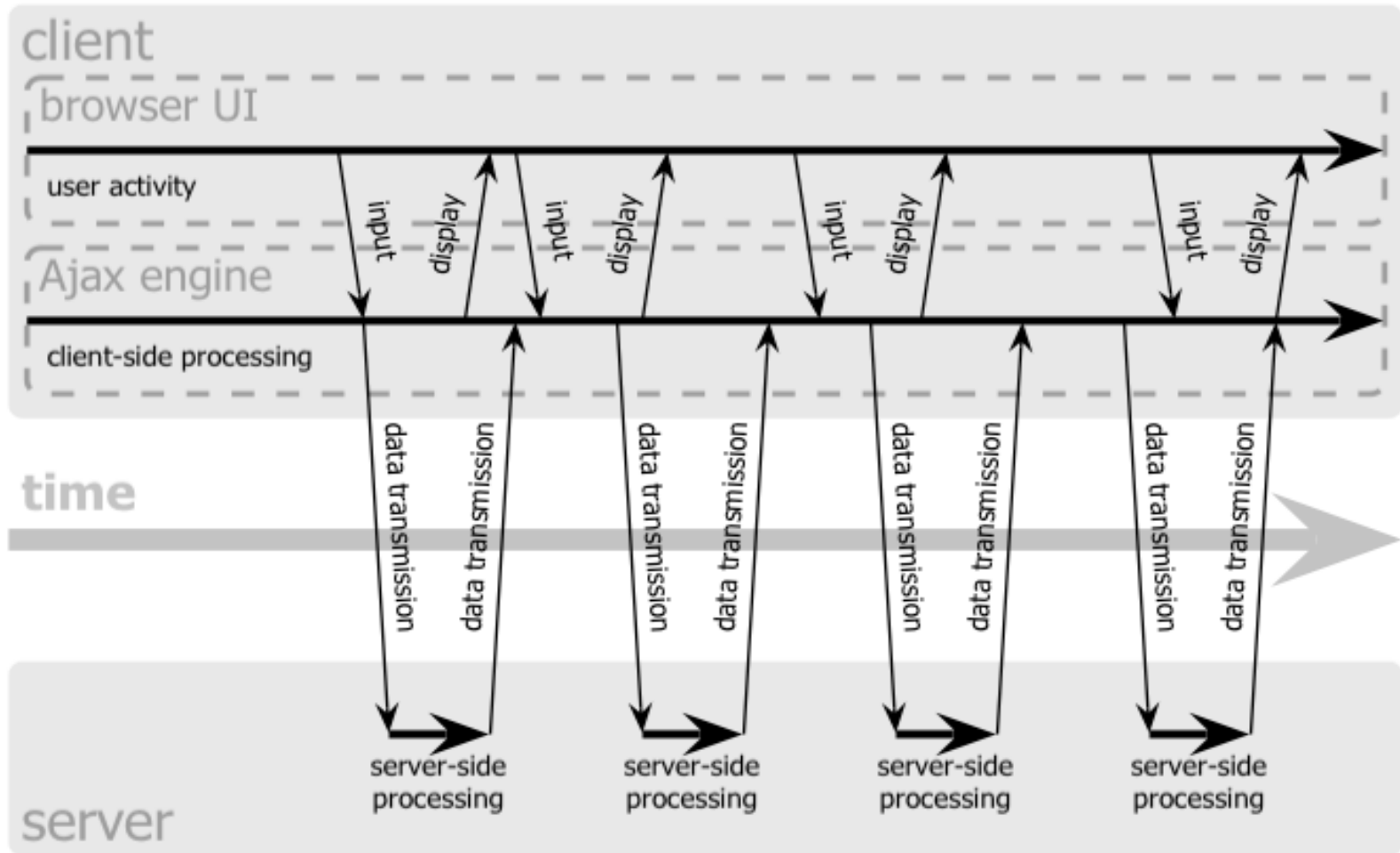
Klassisch:

- **Synchroner** Datenaustausch
- Inhaltsänderungen nur über **Neuladen** einer Seite
- **Stop-and-Go** bei Serveranfragen
- Daten und Inhalt sind **eine Einheit**
 - ◇ Reicht, um statische Daten darzustellen

Aber: Tendenz zu Web-Anwendungen mit funktionell aktivem Inhalt

- ◇ Dynamik, Benutzerfreundlichkeit und Formalismus gefragt

5. AJAX-Anwendungen



Ajax: A New Approach to Web Applications
www.adaptivepath.com

5. AJAX-Anwendungen (II)

Was man sich wünscht:

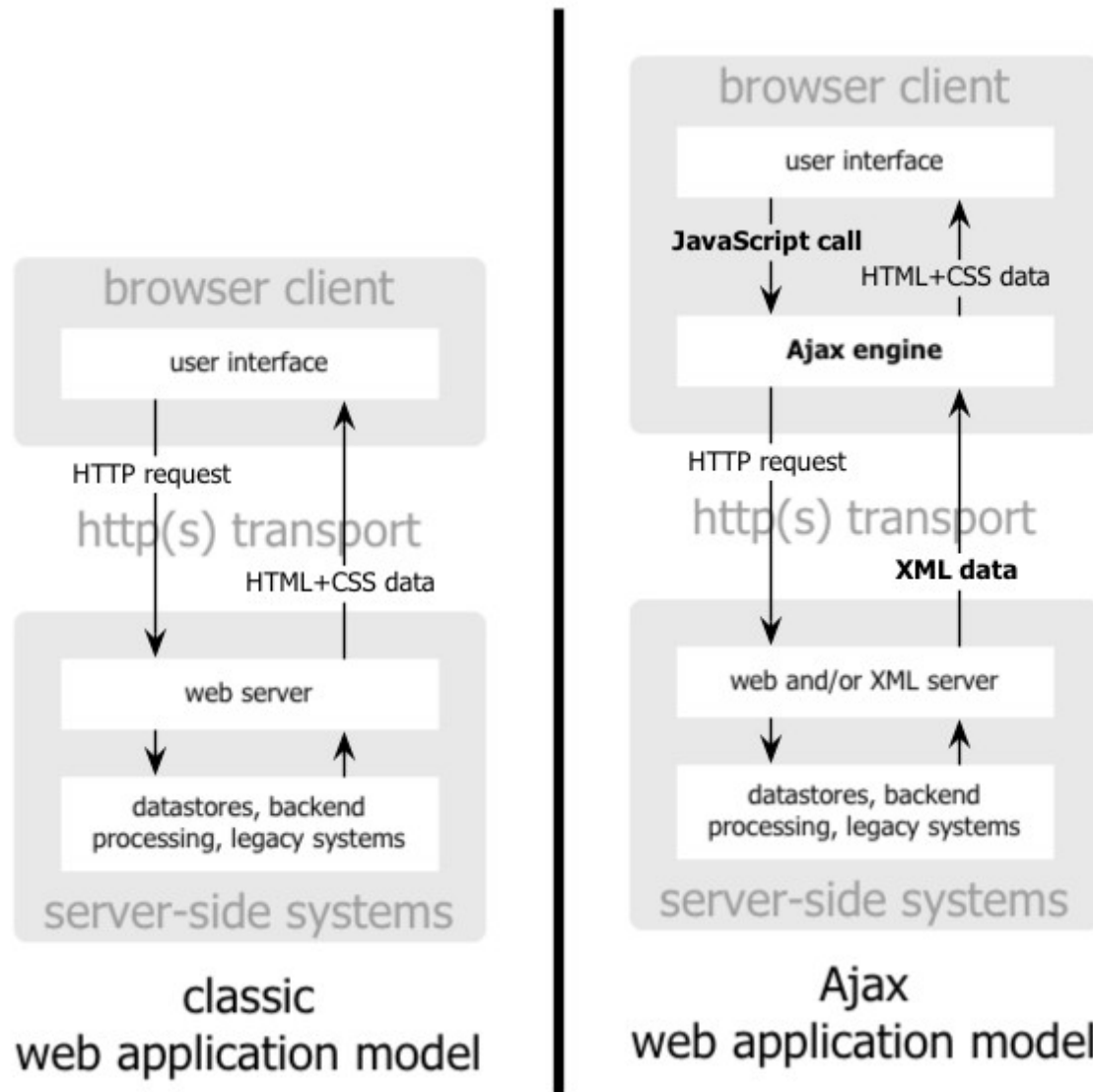
- Beliebiger asynchroner Datenaustausch, ohne „Lade-Bildschirm“
- Daten und Präsentation unabhängig voneinander

Hintergründe:

- Arbeitsfluss gestoppt, während man wartet
- Entwicklungformalismen

Hier setzt AJAX an.

6. Klassische Anwendungen vs. AJAX



Ajax: A New Approach to Web Applications www.adaptivepath.com

7. XMLHttpRequest

Entstand ursprünglich aus Microsoft-Entwicklung (2000)

- API, für ECMA-Script-Sprachen
- Text-, bzw. XML-basierende Kommunikation
- Arbeitet **parallel** zu Client-Calls
- Beinhaltet sechs Methoden
 - ◊ abort()
 - ◊ getAllResponseHeaders()
 - ◊ getResponseHeader(header)
 - ◊ open(...)
 - ◊ send(content)
 - ◊ setRequestHeader(label, value)
- Sechs Attribute
 - ◊ onreadystatechange
 - ◊ readyState
 - ◊ responseText
 - ◊ responseXML
 - ◊ status
 - ◊ statusText

8. Entwicklung von AJAX

1996

- IFRAME-Element eingeführt (IE3)

1997

- LAYER-Element eingeführt (Netscape 4)

1998

- **Microsoft Remote Scripting** (MSRS) (IE4, N4)

2000

- Erste Ansätze zu Remote Scripting (JSRS)
- Image/Cookie-Ansatz
- JavaScript on Demand
- **XMLHttpRequest** (Exchange Server, IE5+6: XMLHttp & ActiveX)

2002

- XMLHttpRequest als Quasi-Community-Standard

2003

- Garret publiziert sein AJAX-Paper (2007: nur eine **W3C WD**)

9. Libraries oder Frameworks?

Libraries

- Einzelne Funktionsbibliotheken
- **Direkter Zugriff, Steuerung von Kontrollfluss**

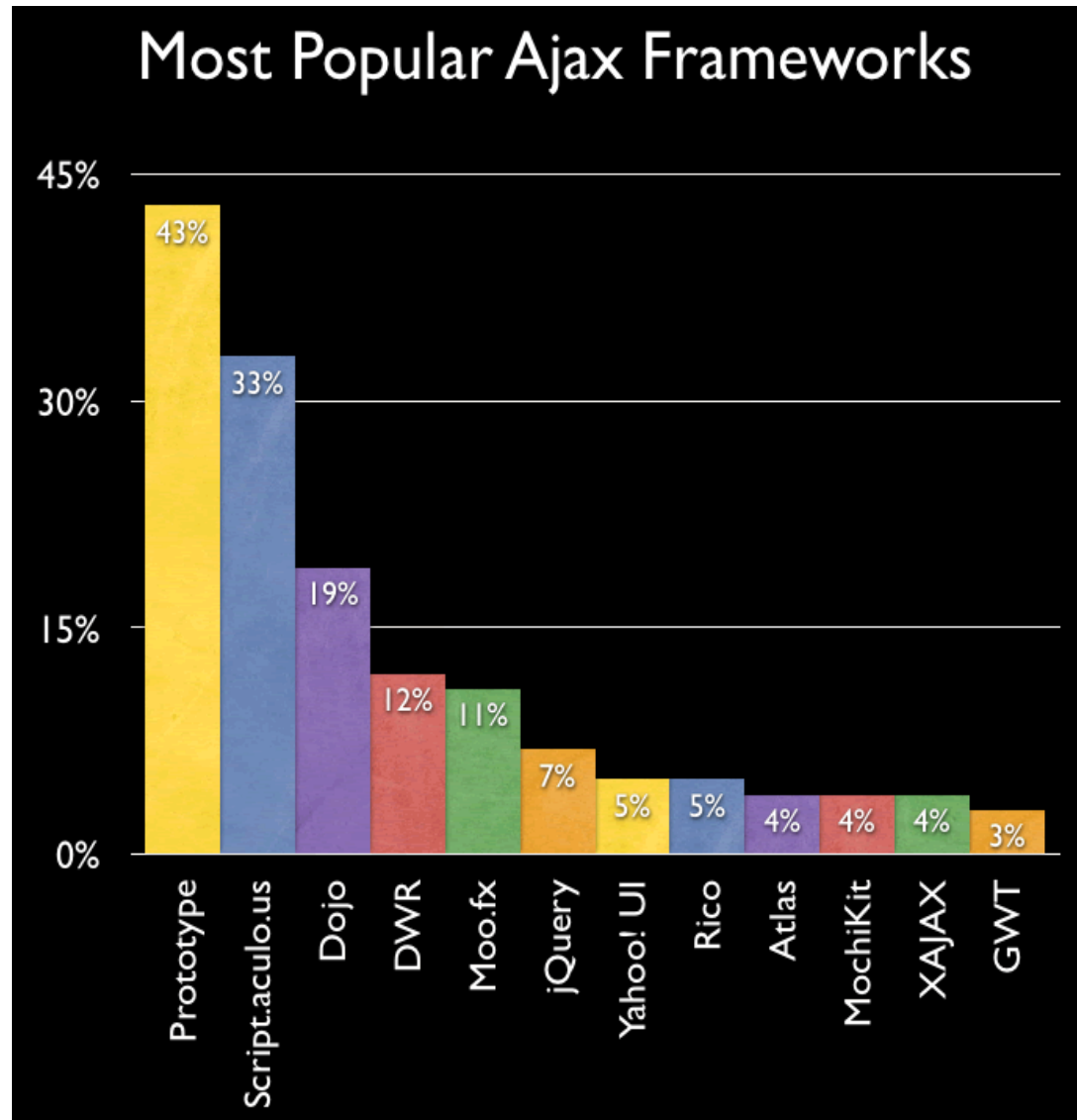
Frameworks

- **Kontrollfluss wird vom Framework bestimmt**
- Framework ruft Anwendung auf

Unterteilung von Frameworks

- „Direkte“ (Prototype, MooTools)
- Components (Dojo, openRico, script.aculo.us)
- Server-driven (DWR, ASP.NET AJAX)

◇ Einordnung bei komplexeren Frameworks teils schwierig



www.ajaxian.com (Ajaxian.com 2006 Survey Results)

- www.adaptivepath.com (Adaptive Path's Publication Archive)
 - ◇ Ajax: A New Approach to Web Applications
- www.xml.com (O'Reilly XML)
 - ◇ Very Dynamic Web Interfaces
- www.wikipedia.com (Wikipedia)
 - ◇ AJAX_(programming)
- www.well.com (The WELL)
 - ◇ Discussion: JJG – The Elements of User Experience
- www.apple.com (Apple Developer Connection)
 - ◇ Remote Scripting with IFRAME
- www.microsoft.com (Microsoft Developer Network)
 - ◇ MSDN Library Archive: Using Remote Scripting
 - ◇ Dave Massy's WebLog: [Ajax == DHTML + XMLHttpRequest](#) (Biased)
 - ◇ Internet Explorer Weblog: [Native XMLHttpRequest object](#)



JSON-RPC-Java Framework

1. Allgemeine Informationen
2. Funktionen
3. Benutzung
4. Diskussion
5. Vergleich mit DWR

1. Allgemeine Informationen

Hersteller: Metaparadigm Pte Ltd (Singapur)

Erster Release: (laut Webseite): 5.4.2004

Lizenz: Version 1.0 Apache License, Version 2.0
Version <1.0 LGPL

1. a) Grundlagen / JSON

object

{ }

{ members }

members

pair

pair , members

pair

string : value

array

[]

[elements]

elements

value

value , elements

value

string

number

object

array

true

false

null

Siehe json.org für mehr Informationen

2. Funktionalität

- JavaScript RPC zu Java Objekten
- Synchrone und asynchrone Aufrufe
- Opaque References
- Callable References

3. Benutzung (I)

Servlet in web.xml Servlet registrieren

Servletklasse: **com.metaparadigm.jsonrpc.JSONRPCServlet**

Mapping: **/JSON-RPC**

3. Benutzung (II)

In JSP Seite (Serverseitig): JSONRPCBridge in Session erzeugen & Remote-Objekt registrieren

```
<jsp:useBean id="JSONRPCBridge" scope="session"  
class="com.metaparadigm.jsonrpc.JSONRPCBridge"/>
```

```
<jsp:useBean id="myObject" scope="session"  
class="test.MyClass"/>
```

```
<% JSONRPCBridge.registerObject("greeter", myObject); %>
```

3. Benutzung (III a)

Die Java Klasse:

```
package test;

public class MyClass implements Serializable {

    private static final long serialVersionUID =
        4888504946382129198L;

    public String sayHello(String name){
        return "Hello "+name+" !";
    }
}
```

3. Benutzung (III)

In JSP/HTML: RPC in JavaScript

```
<script type="text/javascript" src="jsonrpc.js">
<script type="text/javascript">
<!--
    function doRPC() {
        var jsonrpc = new JSONRpcClient("/webapp/JSON-RPC");
        var result = jsonrpc.greeter.sayHello("Max");
        alert(result);
    }
-->
</script>
```

3. Benutzung (IV)

In JSP/HTML Seite: Asynchroner RPC

```
<script type="text/javascript">
<!--
    function doRPC() {
        var jsonrpc = new JSONRpcClient("/webapp/JSON-RPC/");
        var result = jsonrpc.greeter.sayHello(cb, "Max");
    }

    function cb(result) {
        alert(result);
    }
-->
</script>
```

4. Diskussion

Vorteile:

- Schnell zu erlernen
- Server und Client unabhängig
- Geringe Datengröße bei Übertragung (JSON)

Nachteile:

- Keine clientseitige Validierung
- Dezentrale Verwaltung von Objekten -> aufwendig
- Dokumentation nicht ganz aktuell

Was ist DWR?

- Ebenfalls ein RPC Framework
- Server-Seite ebenfalls Java
- Auch 2004 entstanden
- Sehr populär

5. Vergleich mit DWR (II)

DWR vs. JSON-RPC-Java

- Erzeugt Remote-Objekte über dynamisch generiertes JavaScript
- Ausführlichere Dokumentation
- Hat zentrale Konfigurationsdatei
- Feingranulare Sicherheitseinstellungen
- Hat JavaScript Utils-Klasse, für typische Aufgaben
- Reverse AJAX

JSON-RPC-Java

<http://oss.metaparadigm.com/jsonrpc/>

DWR

<http://getahead.org/dwr/>

JSON

<http://www.json.org/>

JSON-RPC

<http://json-rpc.org/>



Achilles and Ajax Vase Picture from History 201 www.maxwell.syr.edu

Framework: Prototype

Prototype

- Eigenschaften
- Funktionsumfang
 - ◇ Bibliotheken
 - ◇ ajax.js
 - ◇ base.js
 - ◇ \$() Funktion
 - ◇ dom.js

1. Prototype

Eigenschaften

- JavaScript-basierendes Framework
- Open-Source, verfügbar auf www.prototypejs.org
- Lizenz: MIT, ©Sam Stephenson
- Version: 1.5.1.1 (25.6.2007)
- Klassenbasierter Ansatz

Einbinden

- `<script src="prototype.js" type="text/javascript">
</script>`

2. Funktionsumfang

Bibliotheken

- prototype.js
- ajax.js
- base.js
- compat.js
- dom.js
- form.js
- string.js

2. Funktionsumfang

ajax.js

Klassen und Methoden

- Ajax
 - ◇ getTransport
- Ajax.Base
 - ◇ setOptions(options)
 - ◇ responseIsSuccess()
 - ◇ responseIsFailure()
- Ajax.Request
 - ◇ Request(url, options)
- Ajax.Updater
 - ◇ Updater(elem, url, options)
- Ajax.PeriodicalUpdater
 - ◇ PeriodicalUpdater(elem, url, options)

2. Funktionsumfang

Ajax.Request

- Kapselt HTTP-Request
- Aufruf:
 - ◇ `var myAjax = new Ajax.Request(url, options)`
 - ◇ `options` ist Hashfeld, default: `{ method : 'post' }`

Beispiel

```
window.onload = function() {  
    var myAjax = new Ajax.Request( "ajax.request.1.php",  
        { onComplete : showResponse, onFailure : showError } );  
}  
  
var showResponse = function(r) { alert(r.reponseText); }  
  
var showError = function(r) {  
    alert("Error: " + r.status + "/t" + r.statusText);  
}
```

2. Funktionsumfang

Ajax.Updater

- Request mit übergebenem Beschreibungselement
- Aufruf:
 - ◇ `var myAjax = new Ajax.Updater(elem, url, options);`
 - ◇ `elem` ist zu beschreibendes Element

Beispiel

Datei 1 (zeit.php), soll Zeit drucken:

```
<?php
    $d = ($_POST['t']) ? "d.m.y - h:m:s" : "d.m.y";
    print "Aktuelle Serverzeit: ".data($d);
?>
```

Datei 2: HTML-Seite, in der die Darstellung erfolgt (Script-Teil)

```
<script src="prototype.js" type="text/javascript"/>
<script type="text/javascript">
```

```
    var getServerTime = function(flag) {
        var param = (flag) ? "t=1," : " ";
        var myAjax = new Ajax.Updater ("zeit", "zeit.php",
            { parameters : param, onFailure : showError }
        );
    }
```

```
    var showError = function(r) {
        alert("Error: " + r.status + "/t" + r.statusText);
    }
```

```
</script>
```

2. Funktionsumfang

Datei 2: Body-Teil

```
<a href="#" onclick="getServerTime(0); return false;">
  Datum
</a>
<a href="#" onclick="getServerTime(1); return false;">
  Datum + Zeit
</a>

...

<div id="zeit"/>
```

- Klicken der Links bewirkt einen Remote Call

2. Funktionsumfang

base.js

- Literal-Objekte als Klasse darstellen
- Number-Objekt dient zur Arbeit mit numerischen Werten
- Methode toColorPart () ermittelt für einzelne Integerwerte den jeweiligen hexadezimalen Wert

Beispiel

```
alert("Rot: #" + Nummer(255).toColorPart() +  
      Nummer(0).toColorPart() +  
      Nummer(0).toColorPart() )
```

- Als Ergebnis bekommt man Rot: #ff0000

2. Funktionsumfang

\$() Funktion

- Verkürzte Schreibweise für den Zugriff auf Elemente der Dokumentenstruktur

Beispiel

```
var e = $("zeit")  
alert(e.style.color);
```

- Beispiel legt Referenz e auf Element mit CSS-ID „zeit“
- Üblicherweise würde man Referenz wie so bekommen:
var e = document.getElementById("zeit");

2. Funktionsumfang

dom.js

- Für Arbeit mit DOM bietet Prototype eine Reihe nützlicher Methoden an, z.B.

`document.getElementsByClassName`

- Methode erweitert Umfang des `document`-Objekt des DOM
- Syntax:
 - ◇ `document.getElementsByClassName(class);`
- `class` ist gewünschter Klassenname
- Liefert Array von Objekten aller gefundenen Elemente zur gesuchten CSS-Klasse



Achilles and Ajax Vase Picture from History 201 www.maxwell.syr.edu

Framework: script.aculo.us

script.aculo.us

- Eigenschaften
- Funktionsumfang
 - ◇ Drag & Drop

1. script.aculo.us

Eigenschaften

- JavaScript-basierendes Framework
- Open-Source, verfügbar auf <http://script.aculo.us>
- Prototype als Basis
- Hauptmerkmal: spektakuläre Effekte

Einbinden

```
<script src="lib/prototype.js" type="text/javascript">
  </script>
<script src="src/script.aculo.us.js"
  type="text/javascript"> </script>
```

2. Funktionsumfang

Drag & Drop

Mit Maus einzelne Elemente einer Webseite bewegen

Klassen und Methoden

- Draggable
 - ◇ Draggable()
 - ◇ destroy()
- Droppable
 - ◇ add()
 - ◇ remove()
- Sortable
 - ◇ create()
 - ◇ destroy()
 - ◇ serialize()

2. Funktionsumfang

Draggable

- Elemente erweitern und bewegen
- Syntax:
 - ◇ `new Draggable(element, options)`

Beispiel (Script Einbettung)

```
...  
<script src="lib/prototype.js"  
        type="text/javascript"></script>  
<script src="src/script.aculo.us.js"  
type="text/javascript"></script>  
...
```

2. Funktionsumfang

```
<script type = "text/javascript">

  onload=function() {
    d = new Draggable( 'bsp',
      { change : check,
        handle : innen } );
  }

  var check = function() {
    if (parseInt($('bsp').style.left) >= 300)
      this.constraint = 'vertical';
    if (parseInt($('bsp').style.top) >=400)
      d.destroy();
  }

</script>
```

2. Funktionsumfang

...

```
<div id="bsp" style="position : absolute;
                    color : #ff0000;
                    white-space : pre;">
```

```
+-----+
|         |
|         |
+-----<span id "innen"
style="color:#0000ff;">+</span>
</div>
```

...

- J. Gamperl: Ajax – Web 2.0 in der Praxis, Galileo Computing, 2006, ISBN 978-3-89842-764-7
- www.ajaxian.com/resources
- www.prototypejs.org
- <http://wiki.script.aculo.us/scriptaculous/show/Prototype>
- <http://script.aculo.us>
- <http://www.sergiopereira.com/articles/prototype.js.html>



Achilles and Ajax Vase Picture from History 201 www.maxwell.syr.edu

AJAX – Kritikpunkte und Best Practices

1. Kritikpunkte aus Nutzersicht

- Benötigt aktiviertes JavaScript
- Ungewohntes Anwendungsverhalten
- Eingeschränkte Zugänglichkeit / Barrierefreiheit

2. Technische Kritikpunkte

- Funktionalität des Zurück-Knopfes
- Funktionalität von Lesezeichen
- Fehlende Rückmeldung bei asynchronem Nachladen
- Einbindung in Suchmaschinen
- Evaluation der Seitennutzung
- Sicherheitslücke bei Datentransfer über JavaScript

3. Best Practices

4. Pluspunkte

1. Kritikpunkte aus Nutzersicht

I. Benötigt aktiviertes JavaScript im Browser

- Ohne JavaScript keine AJAX-Anwendung
- Benötigt XMLHttpRequest-Objekt (<IE7 zusätzlich aktiviertes ActiveX)
- Problem für Plattformen wie ältere Browser, mobile Geräte und Text-only-Browser

Lösung

- Aufklärung des Nutzers über AJAX und Scripting
- Alternative Seite/Anzeige (noscript-Tag) ohne JavaScript
- Alternativ: keine kritische Funktionalität mit AJAX

1. Kritikpunkte aus Nutzersicht

II. Ungewohntes Anwendungsverhalten für Weboberfläche

- Statusanzeige der Anwendung direkt auf Seite anstatt im Browser
- Visuelles Feedback anwendungsspezifisch
- Validierung und Speicherung von Eingaben ohne Absenden-Knopf
- Aktualisierung der Seite nicht sofort ersichtlich
- Eingeschränkte Funktionalität des Zurück-Knopfes
- Eingeschränkte Funktionalität der Lesezeichen

Lösung

- Aufklärung des Nutzers über Anwendungsverhalten
- Leicht erfassbares Feedback
- Hinweise und leichter Zugriff auf Änderungen
- Zu eingeschränkter Funktionalität: siehe technische Kritikpunkte

1. Kritikpunkte aus Nutzersicht

III. Eingeschränkte Zugänglichkeit / Barrierefreiheit

- Anwendungen meist für grafische Browser entworfen
- Screen Reader für Sehbehinderte nur bedingt für dynamisches Nachladen geeignet

Lösung

- Alternative Seite anbieten
- Kompatibilität zu Screen Readern gewährleisten
 - ◇ Hinweis und einfacher Zugriff auf Änderungen
 - ◇ Kontrolle über das Fokusverhalten in der Anwendung
 - ◇ Kontrolle über die Häufigkeit der dynamischen Aktualisierung

2. Technische Kritikpunkte

I. Eingeschränkte Funktionalität des Zurück-Knopfes

- Stellt nicht vorherigen Zustand wieder her
- Lädt stattdessen vorherige Anwendung/Seite
- Löscht Zustand der aktuellen Anwendung

Lösung

- Bereitstellung von einem Rückgängig-Knopf in der Anwendung
- Mischung aus asynchr. Aufbau und „normalem“ Seitenwechsel
- AJAX History Library: arbeitet mit fragment identifier/hash (#)
- Framework

II. Eingeschränkte Funktionalität von Lesezeichen

- Seitenaktualisierung geht nicht einher mit URL-Änderung
- Lesezeichen auf ursprünglichen Zustand der Anwendung
- Sichtbare Seite (Zustand) nicht als URL weiterreichbar

Lösung

- AJAX History Library
 - ◇ Lesezeichen nutzt Anker nach Raute (#) als Zustandsinformation
 - ◇ Anker durch JavaScript dynamisch aktualisierbar
- Bookmark-Link
 - ◇ Problematisch: kompletter Zustand muss in URL kodiert werden
 - ◇ Seiten mit sensitiven Daten nicht zu speichern
- Framework

III. Fehlende Rückmeldung bei asynchronem (Nach)laden

- Asynchrones (Nach)laden gibt keine automatische Rückmeldung
- Zustand der Anwendung teilweise nicht ersichtlich
- Hohe Latenzzeiten erzeugen Gefühl einer trägen Anwendung
- Aktualisierter Inhalt wird möglicherweise nicht wahrgenommen

Lösung

- Implementierung visueller Rückmeldungen
- Wechseln des Fokus auf Aktualisierungen
- Information des Benutzers über Update
- Erleichterte Navigation/Übersicht über zu aktualisierten Inhalt
- Framework

IV. Eingeschränkte Einbindung in Suchmaschinen

- Suchmaschinen werten keine Scripte aus
- Dynamisches Laden verhindert somit vollständige Indexierung

Lösung

- Lightweight Indexing: Meta-Tags und Überschriften
- Extra Link Strategy: unsichtbare Links zu indexierbaren Seiten
- Secondary Site Strategy: eine zweite, voll zugängliche Seite
- Aber: Extra Link und Secondary Site könnten als Täuschungsversuch gewertet werden
- Scripting nur für Nebenfunktionen mit wenig Einfluss auf wichtigen Seiteninhalt

V. Andere Evaluation des Seitenerfolgs / der Seitennutzung

- Klassische Nutzungsstatistik durch höhere Anzahl an Http-Transaktionen pro Seite und asynchronem Nachladen verfälscht
- Seitenzugriffe liefern auch keine repräsentativen Werte mehr, da Nutzer nicht von Seite zu Seite navigiert

Lösung

- Stetige Überwachung der Nutzeraktionen möglich
- Verweildauer und Fehleingaben genauer messbar
- Funktionen fehlen, um Werte auf Qualität der Anwendung abzubilden

VI. Sicherheitslücke bei Datentransfer über JS (Hijacking)

- Schwerwiegend, wenn vertrauliche Daten mit JS transferiert werden
- Lücke:
 - ◇ JS-Datenobjekt-Verweise (z.B. in JSON) werden in Angreiferseite eingebunden
 - ◇ Browser fordert Objekte mit hinterlegten Authentifizierungsdaten für die entsprechende Seite an
 - ◇ Angreiferseite wertet geladenes JS mit eigenem Objekt-Konstruktor aus und gelangt so an die Daten

Lösung

- Direkte Ausführung von JavaScript-Antworten unmöglich machen

3. Best Practices (I)

- Aufklärung des Benutzers
- Option, um automatische Seitenaktualisierung zu deaktivieren
- Benachrichtigung bei Seitenaktualisierung
- Keine automatische Veränderung des Fokus bei Aktualisierung etc.
- Übersichtliche Navigation zu sich ständig ändernden Abschnitten
- Vorhandene Elemente aktualisieren, anstatt neue Elemente einzufügen
- Alternative Seite bereitstellen, die ohne JavaScript funktioniert

3. Best Practices (II)

- Pattern
 - ◇ Refreshing HTML segments
 - ◇ Avoiding browser cache
 - ◇ Reading JavaScript data
 - ◇ Reading XML data
- Anti-Pattern
 - ◇ Doing too much on the server
 - ◇ Passing XML when you should pass JavaScript code
 - ◇ Passing complex XML when HTML would be better
 - ◇ Not inspecting the return results in the callback

4. Pluspunkte

- Nachladen neuer Inhalte anstatt Seitenreload
- Plattformunabhängigkeit
- Keine Plugins nötig
- Kostenlos
- Viele Frameworks, die Probleme abschwächen

- IBM:
 - ◇ [AJAX Accessibility](#)
 - ◇ [AJAX and XML: Five Ajax anti-patterns](#)
 - ◇ [AJAX and XML: Five common Ajax patterns](#)
- Fieldexpert.com:
 - ◇ [AJAX Best Practices: Bookmarks](#)
 - ◇ [AJAX Best Practices: Back Button](#)
- Galileocomputing: [JavaScript / AJAX Open Book](#)
- namics: [AJAX und Web-Grundprinzipien](#)
- Fortify Software: [JavaScript Hijacking](#)
- Softwareas.com: [Current Concerns with AJAX](#)
- Webaim.org: [AJAX Accessibility](#)
- xul.fr: [AJAX Drawbacks](#)
- t3n: [AJAX](#)
- Wikipedia: [AJAX \(Programmierung\)](#)

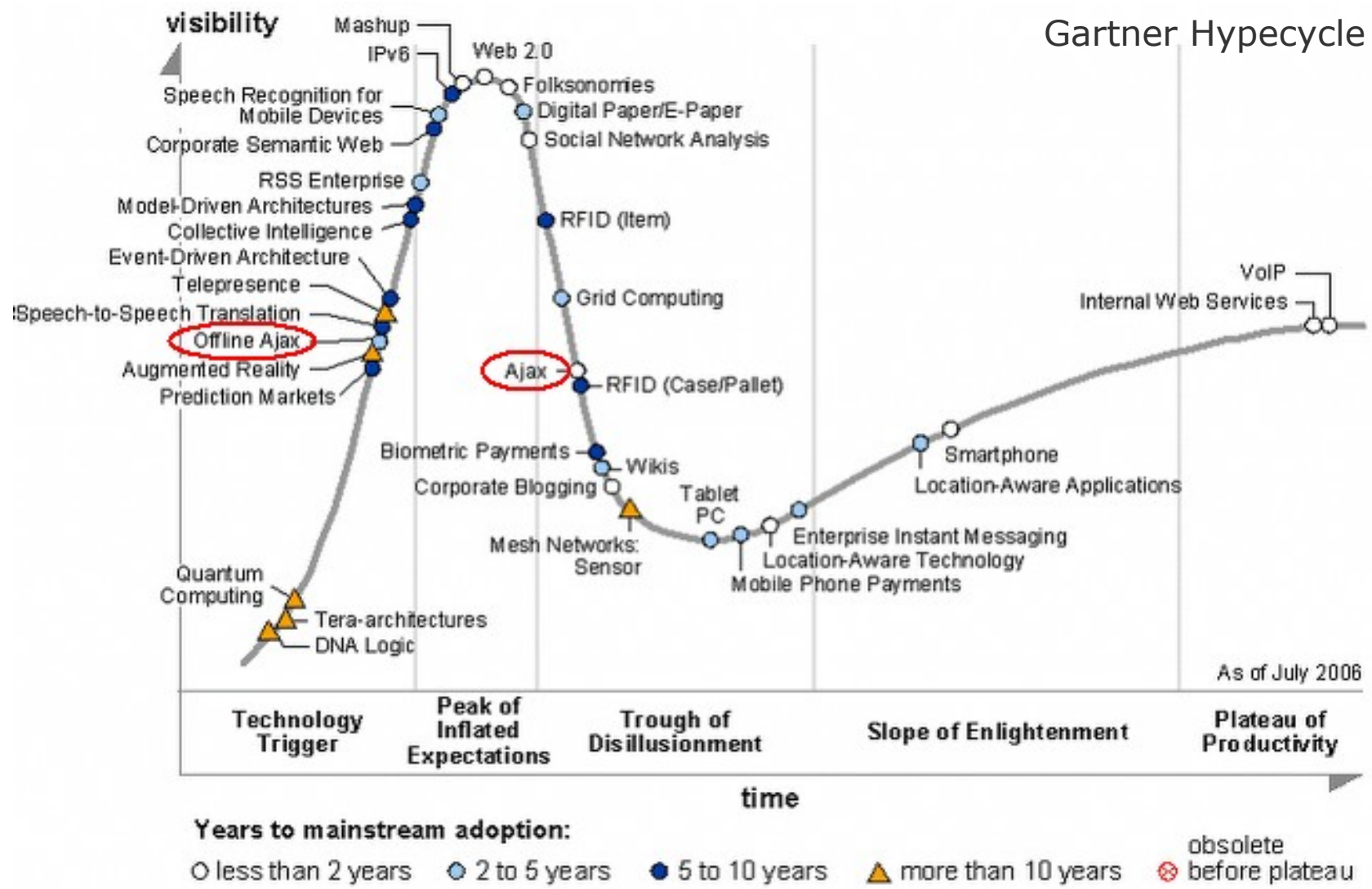


Achilles and Ajax Vase Picture from History 201 www.maxwell.syr.edu

Zukunft von AJAX

1. Derzeitige Entwicklungen
2. Mögliche Entwicklugen
3. Trends
4. Konkurrenten

1. Derzeitige Entwicklungen (I)



Quelle: http://atomiq.org/archives/2006/08/web_20_tops_the_gartner_hype_cycle.html

1. Derzeitige Entwicklungen (II)

- Integration in bestehende Web (1.0) Technologien
 - ◇ Ruby on Rails
 - ◇ JSF : Ajax4jsf ...
 - ◇ ASP : ASP.NET AJAX (aka Atlas) ...
- Reimplementierung von Desktop-GUI Technologien auf Web-Basis
 - ◇ Java SWT : Eclipse RAP & Java2Script
 - ◇ Delphi/C++ VCL : VCL for the Web (aka IntraWeb)

2. Mögliche Entwicklungen

- AJAX Techniken verschwinden hinter Kulissen (Komponenten)
- Komponentenmodelle, die als Web- oder Desktop-Applikationen „deployed“ werden können

3. Trends

- Weg von XML hin zu JSON oder REST
- Weg von SOAP hin zu POX
- Offline AJAX
(Dojo Offline, Google Gears, Adobe AIR, Zimbra Desktop)

4. Konkurrenten

Konkurrenten aus dem RIA Lager:

- Adobe Flash (Flex 3)
- Microsoft Silverlight / Mono Moonlight
- Java (Consumer JRE & JavaFX)
- XUL

Quellen

Heise Newsticker

<http://www.heise.de/newsticker/>

Golem Newsticker

<http://www.golem.de/ticker/>

Ajax4jsf

<http://jboss.org/projects/jbossAjax4jsf>

ASP.NET AJAX

<http://ajax.asp.net/>

Google Gears

<http://gears.google.com/>

Dojo Offline

<http://dojotoolkit.org/offline>



Achilles and Ajax Vase Picture from History 201 www.maxwell.syr.edu

Ende

Vielen Dank!

Noch Fragen?