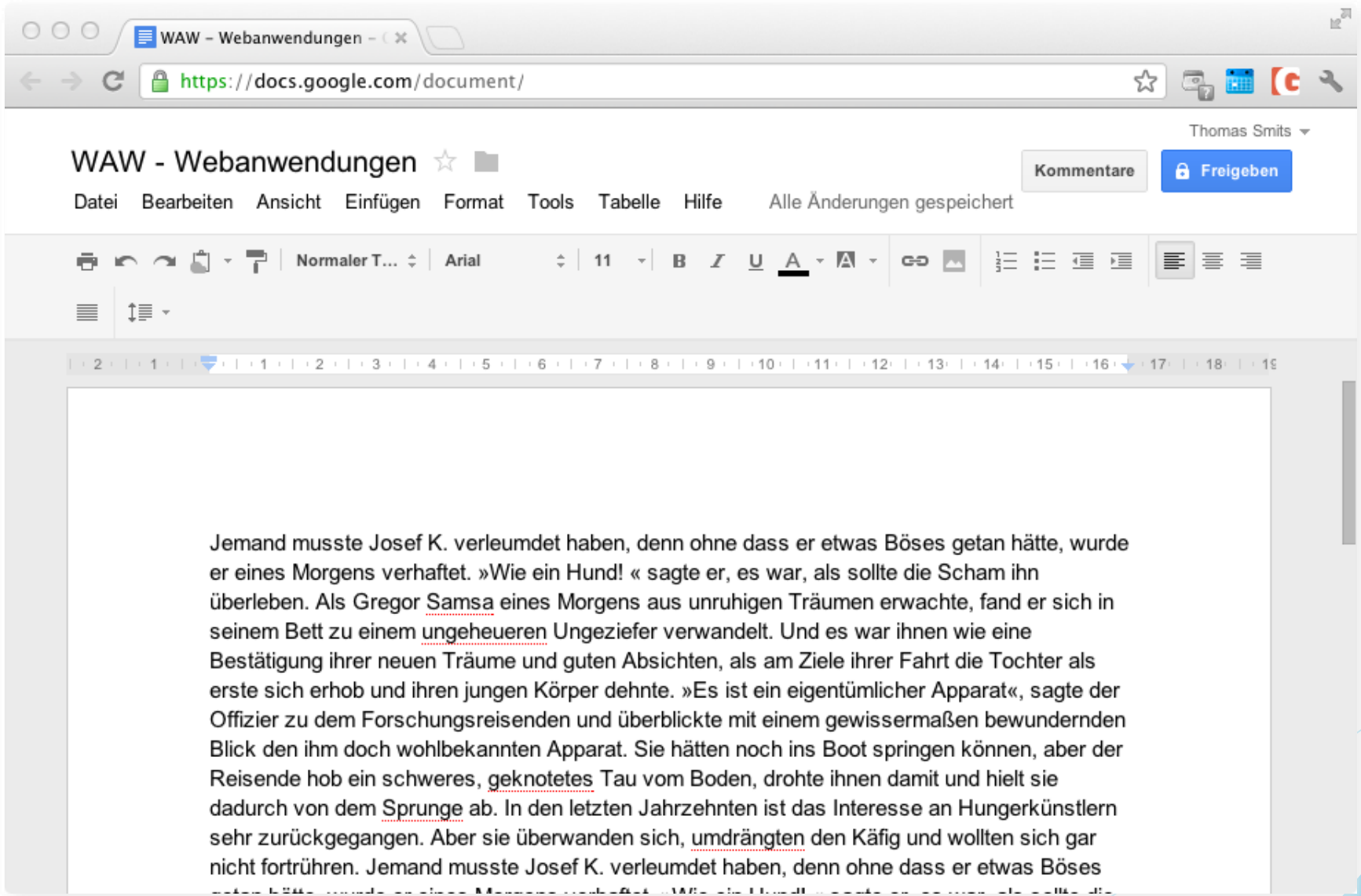
Webanwendungen
Vorlesung – Hochschule Mannheim

AJAX

Einführung

Rich Internet Applications (RIA)



AJAX

AJAX – Asynchronous JavaScript and XML

Ziel: Web-Anwendungen sollen wie Desktop-Apps verhalten

- schnelle Antworten auf Benutzereingaben, interaktiv
- kein Flackern beim Seitenaufbau
- ohne Neuladen der Seite für jede Kleinigkeit
- auch offline benutzbar

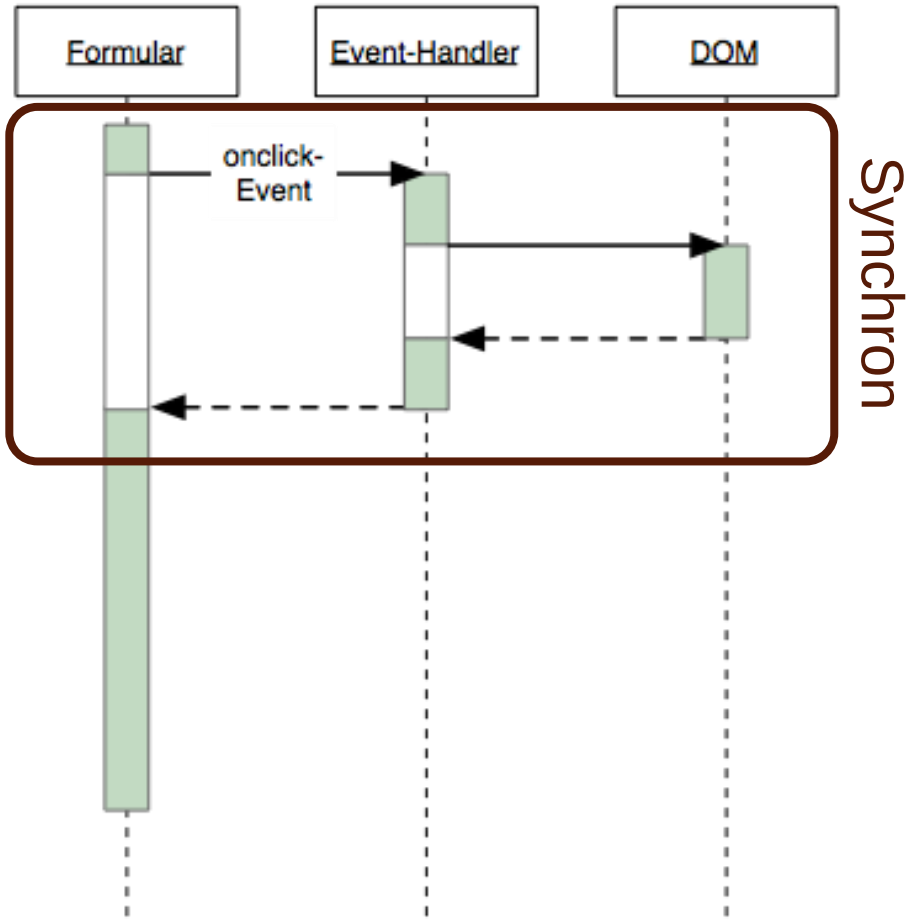
AJAX – Grundidee

- Funktionsprinzip
  - HTML-Seite wird nicht neu geladen
  - Daten werden asynchron (im Hintergrund) vom Server nachgeladen
  - JavaScript manipuliert das DOM und reagiert auf Ereignisse
- Benötigte Komponenten
  - HTML, CSS, JavaScript, DOM, (XML)
  - XMLHttpRequest zum Laden im Hintergrund
  - Frameworks zur Erleichterung (z. B. jQuery, prototype.js)

Beispiel: DOM-Manipulation

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JavaScript DOM-Manipulation</title>
<script type="text/javascript">
  function manipuliere() {
    var div = document.getElementById("ajaxDiv");
    div.firstChild.nodeValue = "Neuer Text (dynamisch eingesetzt)";
  }
</script>
</head>
<body>
  <div id="ajaxDiv">Alter Text</div>
  <button onclick="manipuliere()">Drück mich</button>
</body>
</html>
```

Ablauf der DOM-Manipulation



AJAX

Erweiterung des Beispiels

Eingefügte Daten sollen vom Server kommen

- Server-Aufruf direkt aus dem Event-Handler per JavaScript
- Browser blockiert, bis Antwort vom Server eintrifft

Lösung: XMLHttpRequest-Objekt

- Festlegen einer Abfrage an den Server (als URL)
- Registrieren eines *Callback-Handlers*
- Absenden der Anfrage (asynchron)
- Callback-Handler wird gerufen, wenn Daten eintreffen

Beispiel: AJAX

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JavaScript DOM-Manipulation</title>

<script type="text/javascript">
  ...
</script>
</head>
<body>
  <div id="ajaxDiv">Alter Text</div>
  <button onclick="manipuliere()">Drück mich</button>
</body>
</html>
```

Beispiel: AJAX

```
var request = new XMLHttpRequest();

function manipuliere() {
    // URL für Request festlegen
    request.open("GET", "/php/time.php");

    // Callback-Handler zuordnen
    request.onreadystatechange = callbackHandler;

    // Request abschicken
    request.send();
}
```

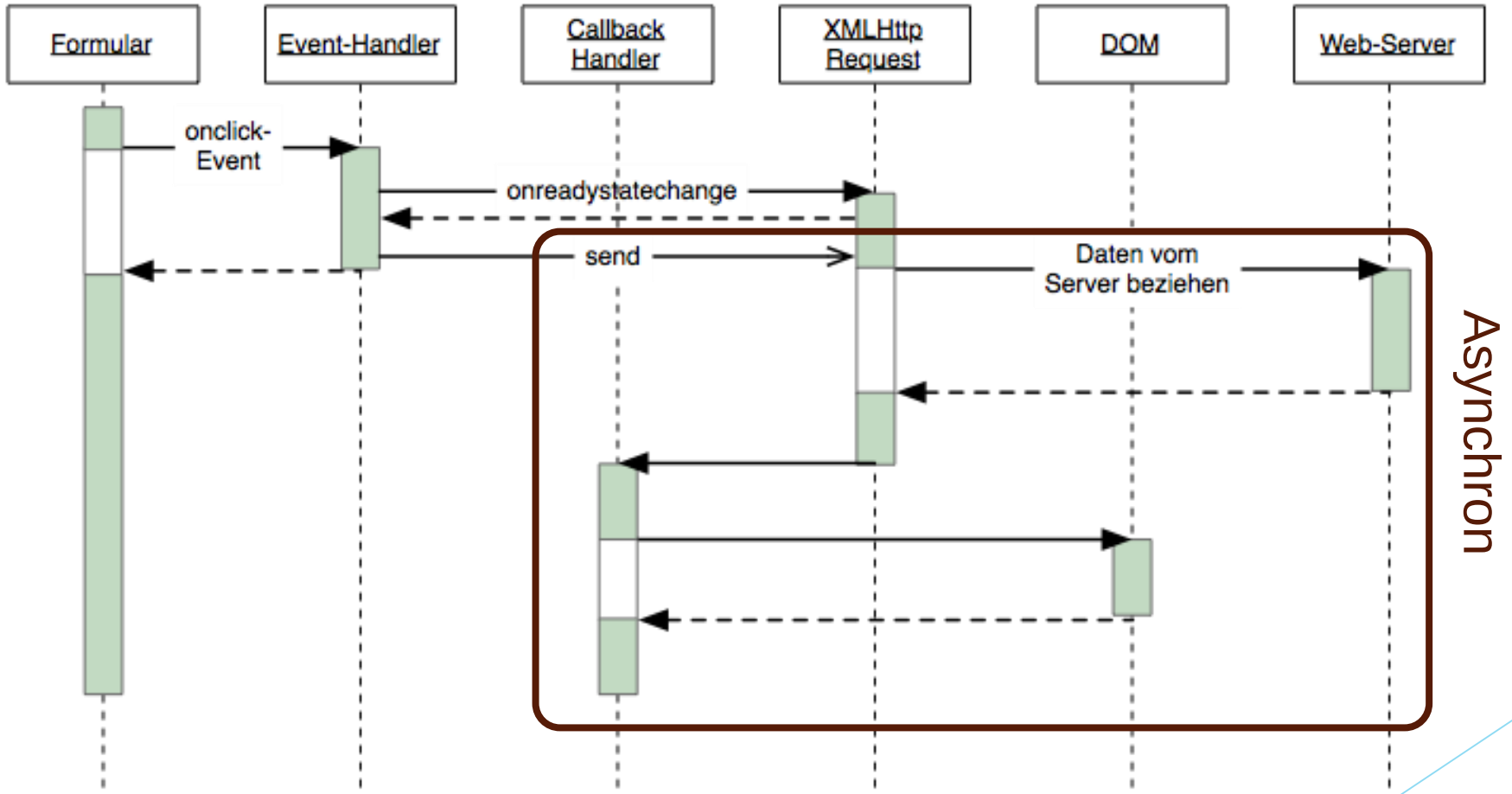
Beispiel: AJAX

```
function callbackHandler() {

    if ((request.readyState == 4)
            && (request.status == 200)
            && (request.responseText != null)) {

        // Daten in das Div-Tag eintragen
        var div = document.getElementById("ajaxDiv");
        div.firstChild.nodeValue = "Vom Server: "
            + request.responseText;
    }
}
```

Ablauf der DOM-Manipulation



XMLHttpRequest-Objekt

<table>
<tr><th>Attribut</th><th colspan="3">Bedeutung</th></tr>
<tr><td>onreadystatechange</td><td colspan="3">Methode, die bei Eintreffen der Antwort aufgerufen werden soll</td></tr>
<tr><td rowspan="8">readyState</td><td colspan="3">Bearbeitungsstatus (manche Browser unterstützen nur 0, 1 und 4)</td></tr>
<tr><td>Wert</td><td>Bezeichnung</td><td>Bedeutung</td></tr>
<tr><td>0</td><td>Uninitialized</td><td>open() noch nicht aufgerufen</td></tr>
<tr><td>1</td><td>Open</td><td>open() aufgerufen, send() noch nicht</td></tr>
<tr><td>2</td><td>Sent</td><td>Anfrage abgesendet (d.h. send() aufgerufen)</td></tr>
<tr><td>3</td><td>Receiving</td><td>Antwort wird gerade vom Server empfangen</td></tr>
<tr><td>4</td><td>Loaded</td><td>Antwort des Servers liegt vollständig vor</td></tr>
</table>

XMLHttpRequest-Objekt

<table>
<tr><th>Attribut</th><th colspan="3">Bedeutung</th></tr>
<tr><td>responseText</td><td colspan="3">Serverantwort der AJAX-Anfrage als String</td></tr>
<tr><td>responseXML</td><td colspan="3">Serverantwort als DOM Level 2 Node (für XML-Antworten)</td></tr>
<tr><td rowspan="7">status</td><td colspan="3">HTTP-Statuscode der Antwortnachricht</td></tr>
<tr><td>Wert</td><td>Bezeichnung</td><td>Bedeutung</td></tr>
<tr><td>200</td><td>OK</td><td>Anfrage war erfolgreich</td></tr>
<tr><td>401</td><td>Unauthorized</td><td>Kein Zugriff ohne Login und Passwort</td></tr>
<tr><td>403</td><td>Forbidden</td><td>Login und / oder Passwort falsch</td></tr>
<tr><td>404</td><td>Not Found</td><td>Angeforderte Datei nicht verfügbar</td></tr>
<tr><td>405</td><td>Internal Server Error</td><td>Serverfehler</td></tr>
<tr><td>statusText</td><td colspan="3">Statusmeldung im Klartext, z.B. Internal Server Error</td></tr>
</table>

XMLHttpRequest-Objekt

Methode	Bedeutung
open(method, url, asyncFlag [,userName [,password]])	Anfrage erstellen
send(content)	Anfrage absenden
abort()	aktuelle Anfrage abbrechen
setRequestHeader(name, value)	Header für Request setzen
getResponseHeader(name)	Header aus Response lesen
getAllResponseHeaders()	Alle Response-Header lesen

XML in AJAX

- XML für komplexe Datenübertragung per AJAX
  - XMLHttpRequest erlaubt auch Übertragung von XML
  - Zugriff auf XML per request.responseXML
    statt request. responseText
  - XML-Dokument wird als eigenes DOM zur Verfügung gestellt
  - Zugriff erfolgt mit normalen DOM-Methoden – also z. B.
    request.responseXML.getElementsByTagName(…);
- Verwendung von XML ist allerdings umständlich
- Alternative: JSON

JSON als Alternative zu XML (AJAJ)

JSON lässt sich deutlich einfacher als XML nutzen

- Einfache und schlanke Notation zur Darstellung von Listen, Strings, Zahlen und assoziativen Arrays
- gängige Programmiersprachen bieten Bibliotheken zum Serialisieren von Objekten nach JSON
- der JavaScript Befehl eval() deserialisiert ein übergebenes Argument und liefert das Ergebnis als JavaScript-Objekt (z.B. assoziatives Array)

Vorteile für Anwender

Ähnlichkeit mit Desktop-Anwendungen

- kurze Antwortzeiten
- auch während des Nachladens bedienbar
- sehr interaktiv

Bedienung mit Standard-Web Browser

- aktiviertes JavaScript ausreichend
- keine Browser-Plugins nötig
- plattformunabhängig
  - keine lokale Softwareinstallation nötig

Nachteile für Anwender

- Rückwärtsbutton und Browserhistorie funktionieren nicht
- Bookmarks funktionieren nicht
- Abhängigkeit von Online-Verbindung (offline-Modus möglich)
- Verzögerungen beim Nachladen von Daten
- Sicherheitsprobleme beim Nachladen von fremden Webservern

Vorteile für Entwickler

- geringere Serverbelastung als bei ständigem Nachladen kompletter Seiten
- zentrale Softwareinstallation und -wartung
- nur eine Version für sämtliche Plattformen nötig
- Kombination bestehender Anwendungen zu Software Mashups
  - Wiederverwendung
  - Software-Mietmodelle

Nachteile für Entwickler

erhöhter Programmier- und Testaufwand

- schwer zu debuggen
- Tricks, um Rückwärtsbutton, Browserhistorie und Bookmarks zu ermöglichen
- Tricks, um Suchmaschinen zu unterstützen
- Browserunterschiede müssen berücksichtigt warden

JavaScript keine vollwertige objektorientierte Programmiersprache