Ajax

Programmering af Rige Web Applikationer

WAU 12-10-2007

Allan Hansen <fah@daimi.au.dk>

Agenda

1. Ajax Baggrund
2. Hvorfor Ajax?
3. Teknologien
   - XMLHTTPRequest Object (XHR)
   - Data (HTML, XML, JSON, ...)
4. Problemer med Ajax
5. Ajax biblioteker ("The Ajax Engine")
6. Eksempler

Ajax Baggrund

- Asynchronous JavaScript and XML
- Ajax er ikke én teknologi
- En måde at udvikle Web applikationer på
  - Baseret på en samling af teknologier
- Ajax navnet opfundet af J. J. Garrett, Adaptive Path, 2005

Ajax

J. .J. Garratt:

- Standards-based presentation using XHTML and CSS
- Dynamic display and interaction using the Document Object Model (DOM)
- Data interchange and manipulation using XML and XSLT (eller andet)
- Asynchronous (eller synkron) data retrieval using XMLHttpRequest
- and JavaScript binding everything together.

Asynkrone kald

- Ajax er ikke den første teknik
  - Men tidligere teknikker var hacks
    - Skjulte iFrames
    - <img> src, <script> src, css href,
    - ...
- Eller kræver plugins: Flash, Java, ...

Asynkront kald med iFrame

Klient

```
<html>

<script type="text/javascript">
function handleResponse(msg) {
    alert(msg);
}
</script>

  <body>
    <iframe id="frame" style="display: none" src=""></iframe>
    <a href="server.php" target="frame">Server kald</a>
  </body>
</html>
```

Asynkront kald med iFrame

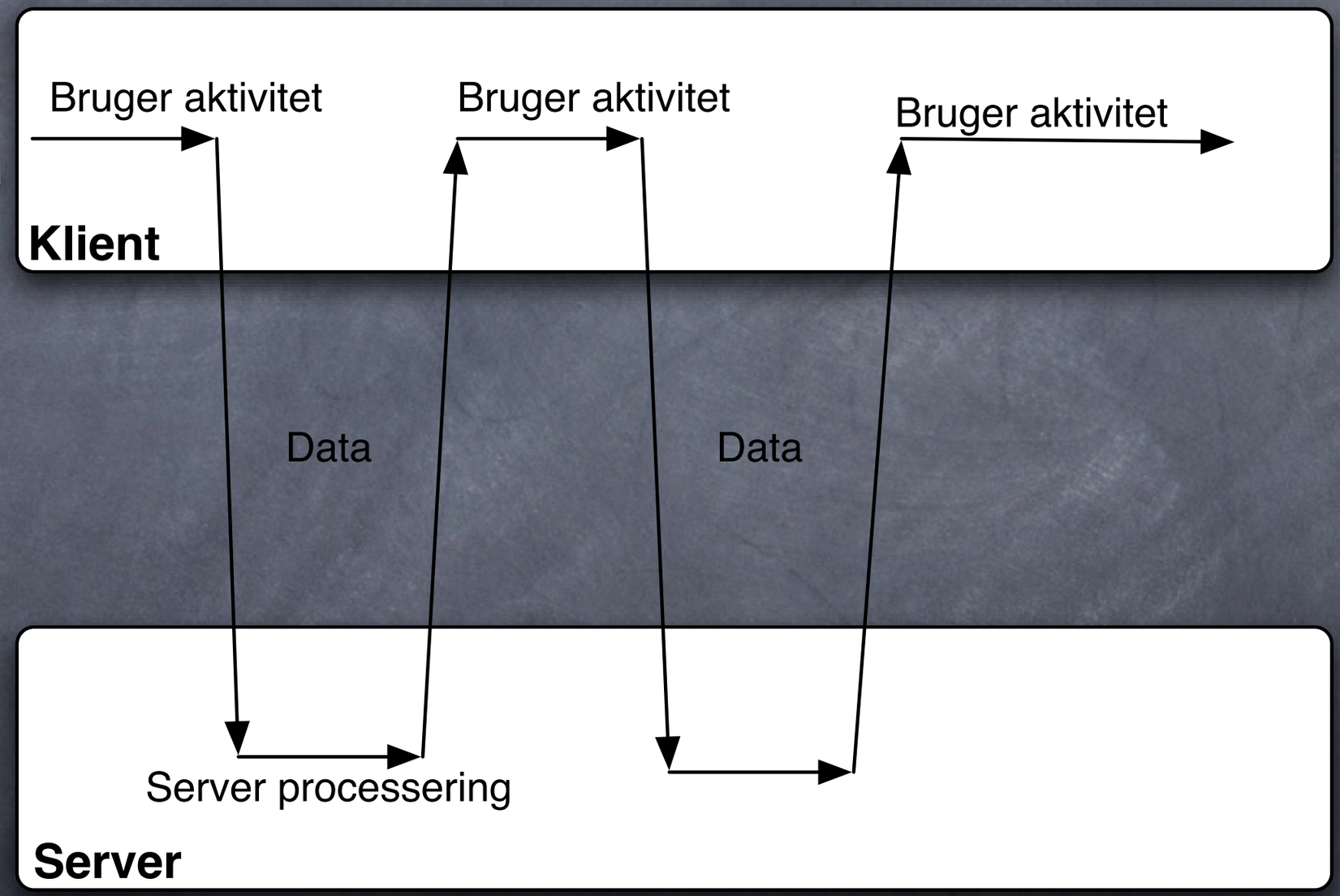
Klient

```
<html>

<script type="text/javascript">
function handleResponse(msg) {
    alert(msg);
}
</script>

  <body>
    <iframe id="frame" style="display: none" src=""></iframe>
    <a href="server.php" target="frame">Server kald</a>
  </body>
</html>
```

Server svar

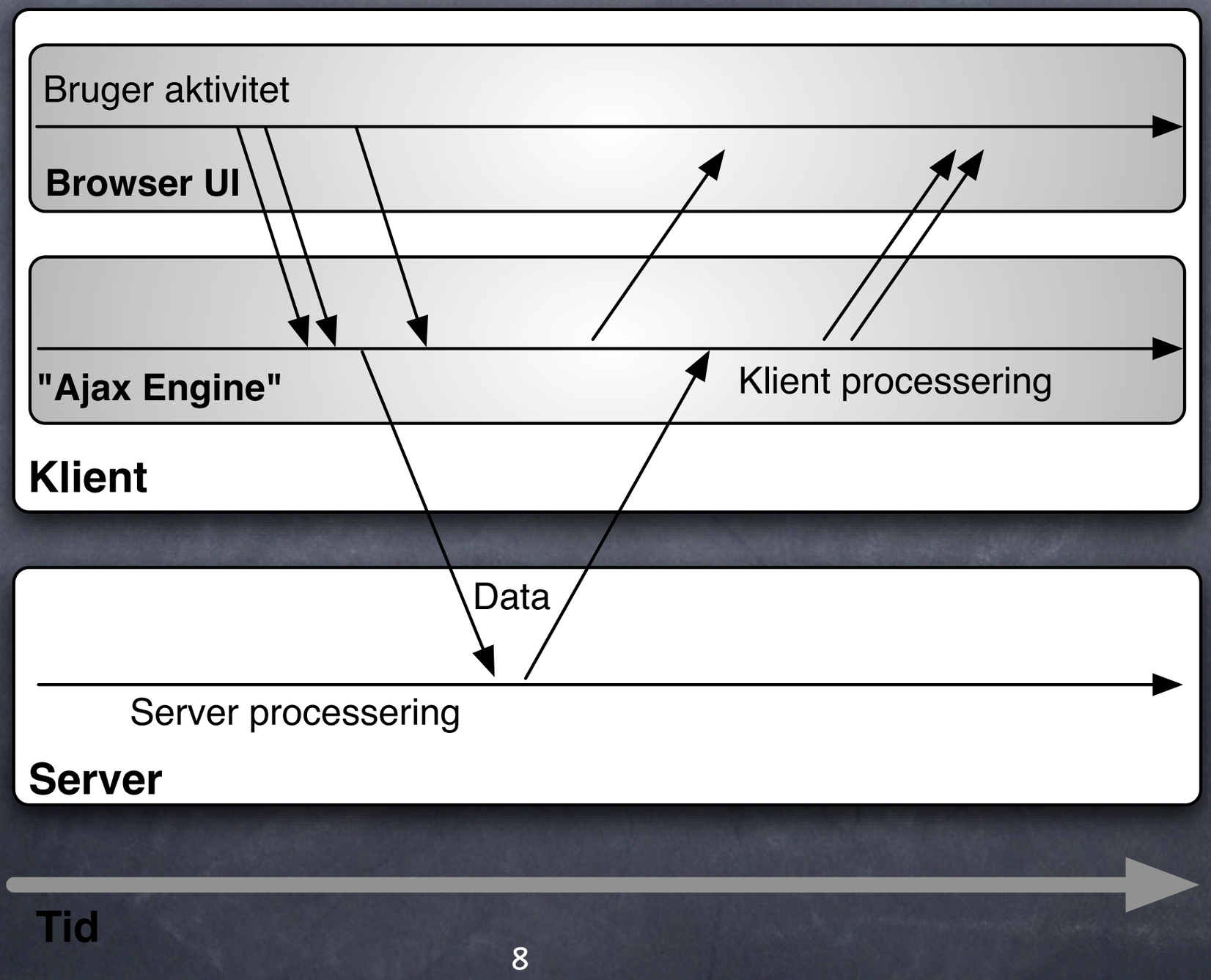
```
<script type="text/javascript"<
   window.parent.handleResponse('...')
</script>
```

Klassisk Web Applikation (Synkron)



Tid

Ajax Web Applikation (Asynkron)



Hvorfor Ajax?

Hvorfor Ajax?

- Hurtigere applikationer
- Mindre krav til båndbredde
  - Hele siden reloades ikke
- Meget bedre brugbarheds oplevelse
- Kræver ikke ekstra browser-plugins
  - Fungerer i de fleste browsere
- Kan give god separeret arkitektur

To typer af anvendelser

- **Widgets**
  - Små programmer der kører som del af almindelig side
  - Forbedre brugbarheden
  - "Degradable"
  - eks.: Google suggest
- **Rich Internet Application**
  - Hele applikationer
  - eks.: Google maps

Alá anvendelserne for Flash!

Ajax Teknologien: XMLHttpRequest Object

XMLHttpRequest (XHR)

- XHR er et objekt der tillader asynkrone HTTP (POST, GET, HEADER) kald fra JavaScript
- Kaldene kan udføres i baggrunden uden reload af hele siden
- En "call-back" JavaScript funktion kan registreres og kaldes af ved hver tilstand i HTTP kaldet

XMLHttpRequest (XHR)

- XMLHttpRequest optrådte første gang i Microsoft Internet Explorer 5
  - ActiveX Component
    - Udnyttet kraftigt i Microsoft Outlook Web Access 2000 (Web udgave af Outlook)
- Implementeret som et JavaScript Objekt i Mozilla 1.0 (Netscape 7.0), Safari 1.2 og senere versioner.
- Lignende funktionalitet findes i W3Cs DOM level 3 "Load and Save" specifikation
- XMLHttpRequest er "de facto standard" men ikke helt ens på tværs af browsere

XHR Objekt

```
var xhr;

if (window.XMLHttpRequest) {
    // Mozilla, Safari, ...
    xhr = new XMLHttpRequest();
} else if (window.ActiveXObject) {
    // IE
    xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
```

XHR Object Methods

Method	Description
open("*method*", "*URL*"[, *asyncFlag* [, "*userName*"[, "*password*"]]])	Assigns URL, method, and other optional attributes of a pending request
send(*content*)	Transmits the request, optionally with postable string or DOM object data
setRequestHeader("*label*", "*value*")	Assigns a label/value pair to the header to be sent with a request
getResponseHeader("*headerLabel*")	Returns the string value of a single header label
getAllResponseHeaders()	Returns complete set of headers (labels and values) as a string
abort()	Stops the current request

XHR Objekt

- Eksempel: oprettelse af forbindelse

```
xhr.open('GET', 'http://www.example.org/server.php', true);
xhr.send(null);
```

XHR Server Response

- Vigtigt også at kunne behandle svar og data fra serveren
- Håndteres af call-back funktion
  - xhr.onreadystatechange = callback;
    hvor callback er navnet på funktionen
  - xhr.onreadystatechange = function() {...};
- Callback funktionen kaldes af XHR objektet for hver tilstand i kaldet

XHR Object Properties

Property	Description
onreadystatechange	Event handler for an event that fires at every state change
readyState	Object status integer: 0 = uninitialized 1 = loading 2 = loaded 3 = interactive 4 = complete
responseText	String version of data returned from server
responseXML	DOM-compatible document object of data returned from server
status	Numeric code returned by server, such as 404 for "Not Found" or 200 for "OK"
statusText	String message for the status code

XHR Objekt

- Eksempel: response fra server

```
xhr.onreadystatechange = xhr_callback;
xhr.open('GET', 'http://www.example.org/server.php', true);
xhr.send(null);

function xhr_callback() {
  // only if httpRequest shows "loaded"
  if (xhr.readyState == 4) {

  // only if "OK"
  if (xhr.status == 200) {
    ...
  } else {
    // Error
    alert(xhr.statusText);
  }
}
```

Ajax Teknologien: Data

Dataformater

HTML	Data	JavaScript
response+innerHTML	XML/JSON	eval()

Ajax: HTML

- Serveren returnerer et html fragment (view specifikation), som kan indsættes direkte i siden.

```
<div id="some_element"> ... </div>

<script>
  document.getElementById("some_element").innerHTML
    = xhr.responseText;
</script>
```

Ajax: XML data

- Serveren returnerer xml data som skal behandles af klienten.
- XML data kan også bruges af ikke Ajax applikationer

Ajax: XML data

- Server data:
  <root><data>...</data></root>
- Klient:

```
var xml_doc = xhr.responseXML;
var element = xml_doc.getElementsByName("root").item(0);
// ... flere dom metoder her
```

Ajax: JSON

- JSON (JavaScript Object Notation) er datastrukturer udtryk i ECMAScript (Object Literals)
- JSON Objekter:
  - object: {member, member, ...}
  - member: <String>name: value
- JSON Arrays:
  - [value, value, ..., value]

Ajax: JSON

- JSON

```
{"menu": "File",
"commands": [
   {
      "title": "New",
      "action":"CreateDoc"
   },
   {
      "title": "Open",
      "action": "OpenDoc"
   },
   {
      "title": "Close",
      "action": "CloseDoc"
   }
  ]
}
```

XML

```
<root>
  <menu>File</menu>
  <commands>
    <item>
      <title>New</value>
      <action>CreateDoc</action>
    </item>
    <item>
      <title>Open</value>
      <action>OpenDoc</action>
    </item>
    <item>
      <title>Close</value>
      <action>CloseDoc</action>
    </item>
  </commands>
</root>
```

Ajax: JSON + JS

- JSON er skræddersyet til Ajax og JavaScript
- Kan fortolkes direkte af JavaScript fortolkeren (Det samme gælder for ren JavaScript kode):
  - var doc = eval('(' + xhr.responseText + ')');
- JSON kan dog forholdsvis nemt oversættes til andre formater som f.eks. XML til brug for ikke Ajax applikationer
- JavaScript kode er dog tæt bundet til Ajax applikationen

Ajax: Dataformater

- XML: DOM modifikation er standard
  - Men lidt besværlig at bruge
  - Mange metodekald for at oprette knuder
- HTML: innerHTML
  - Modificerer html
  - innerHTML er ikke W3C standard
- JSON: eval()
  - Dynamisk evaluering af data, ofte på simpel model

Problemer med Ajax

Ajax Problemer

- Dynamiske ændringer registreres ikke som en unik url:
- Browser History fungerer ikke
- "Back" funktion fungerer ikke
- Man kan ikke linke og bookmarke til bestemte tilstande i applikationen, da urlen er den samme for alle tilstande.

Ajax Problemer

- Browser history
  - Nogle browsere registrerer ændringer i frames, så ændringer kan skrives til en skjult iFrame (f.eks. IE)
- Men ikke understøttet af alle browsere
  - Firefox registrere ikke ændringer i frames foretaget fra DOMen...

Ajax Problemer

- Bookmarks
  - Alle tilstande har samme url
  - Dvs. man kan ikke bookmarke en enkelt tilstand
  - FragmentIdentifieren i sidens url (efter #) kan dog ændres dynamisk (i de fleste browsere) fra JavaScript og dermed beskrive tilstanden

Ajax Problemer

- Network Latency
  - Forsinkelser i netværket
  - Kan have stor betydning for brugsoplevelsen
    - Vigtig med god feedback (ex. throbbers) ved netværksoperationer.
  - Kan også have betydning for rækkefølgen af call-back funktioner.

Ajax Problemer

- Søgemaskiner
  - Søgemaskiner indekserer siders indhold
    - Men Ajax sider oprettes dynamisk i JavaScript, som ikke fortolkes af søgemaskinerne
- Ikke nødvendigvis problem for RIAs (applikationer)

Ajax Problemer

- JavaScript
  - Forskellige browsere
  - Forskellige Datamodeller
  - Forskellige JavaScript fortolkere
- JavaScript scripts skal ofte skrives i flere udgaver til forskellige browsere
- JavaScript Cross-domain requests
- ...En del af dette løses ved at bruge Ajax bibliotker...

Ajax Biblioteker

Ajax Biblioteker

- J. J. Garretts "Ajax engine"
- Mange fordele fremfor at skrive det hele manuelt og re-implementere framework funktionaliteten
- Cross-browser interface til XMLHTTPRequest objektet
- Hurtigere udvikling
- Fejlhåndtering
- Håndtering af mange requests
- Logging
- kryptering
- Avanceret UI biblioteker

Ajax biblioteker

- Prototype
  - Brugt som basis for mange andre toolkits RubyOnRails, script.acul.us, m. fl.
- Dojo
  - Basis for Open Ajax, m. fl.
- Adobe Spry
  - Letvægts, begrænset, rettet mod designere
- YUI: Yahoo User Interface
  - Veldokumenteret og understøttet. Se også Yahoo Pipes.
- Google Web Toolkit
  - Open Source. Programmer udvikles i Java som oversættes til JavaScript.
- Microsoft ASP.NET AJAX
  - Client side, server side, Ajax UI Controls

Ajax Eksempler