VEEBIRAKENDUSTE LOOMINE

MTAT.03.230 (6 EAP)

12. Loeng

Helle Hein

Teema: AJAX
Asynchronous JavaScript + XML

- Ajax motivatsioon
- Ajax põhiprotsess
- Anonüümsed funktsioonid
- Dünaamiline sisu ja JSP
- Dünaamiline sisu ja servletid
- HTML tulemi näitamine

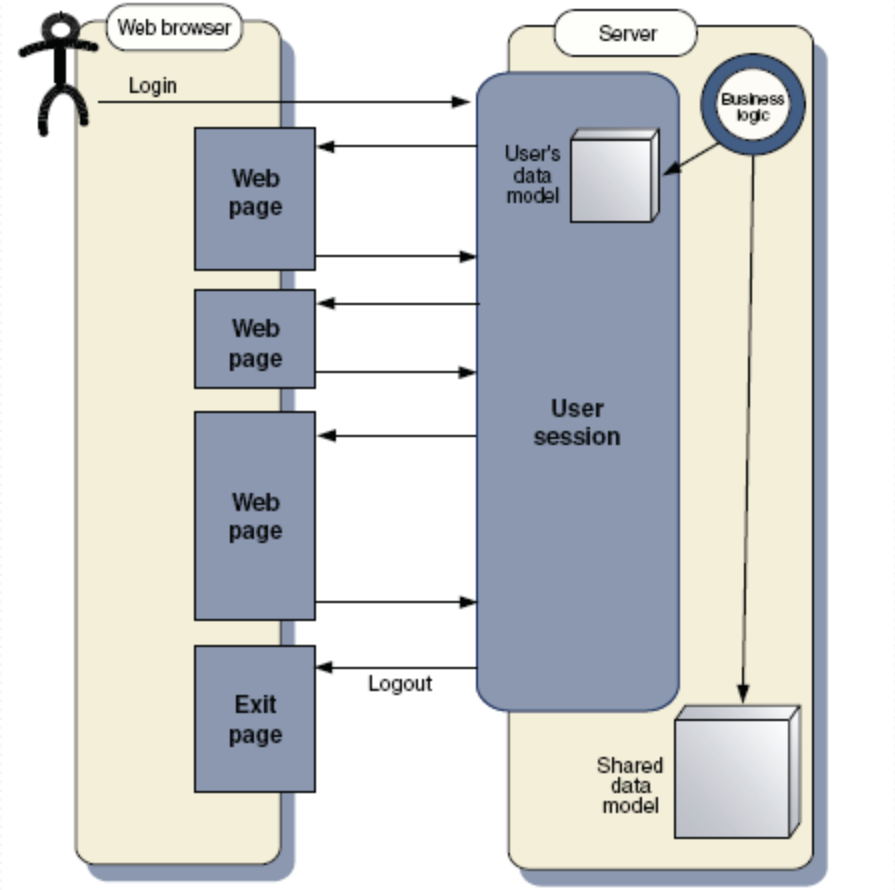
Miks Ajax?

- HTML ja HTTP ei ole interaktiivsed
- Kõik tahavad brauserit kasutada

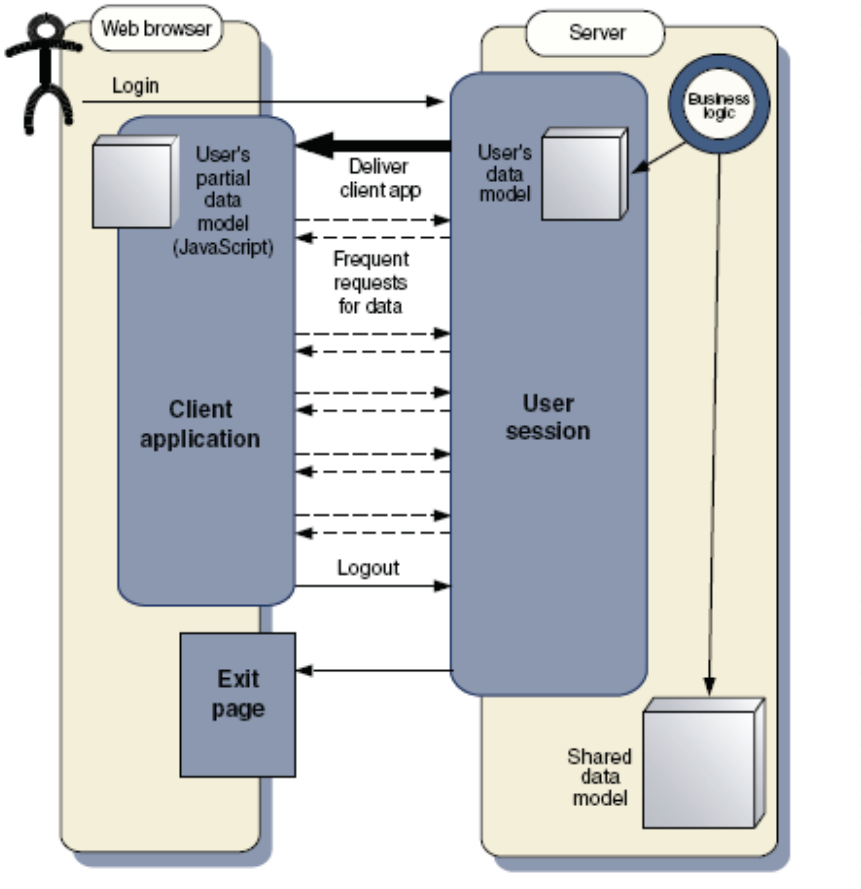
- Brauseri-põhine dünaamilisus
  - Java apletid
    - Kõik ei toeta; ei suhtle HTML-ga
  - Alternatiiv: Flash (ja Flex)
    - Pole veel universaalset toetust
  - Uued
    - Microsoft Silverlight
    - JavaFX
    - Adobe Apollo

Traditsiooniline veebirakendus vs. Ajax rakendus

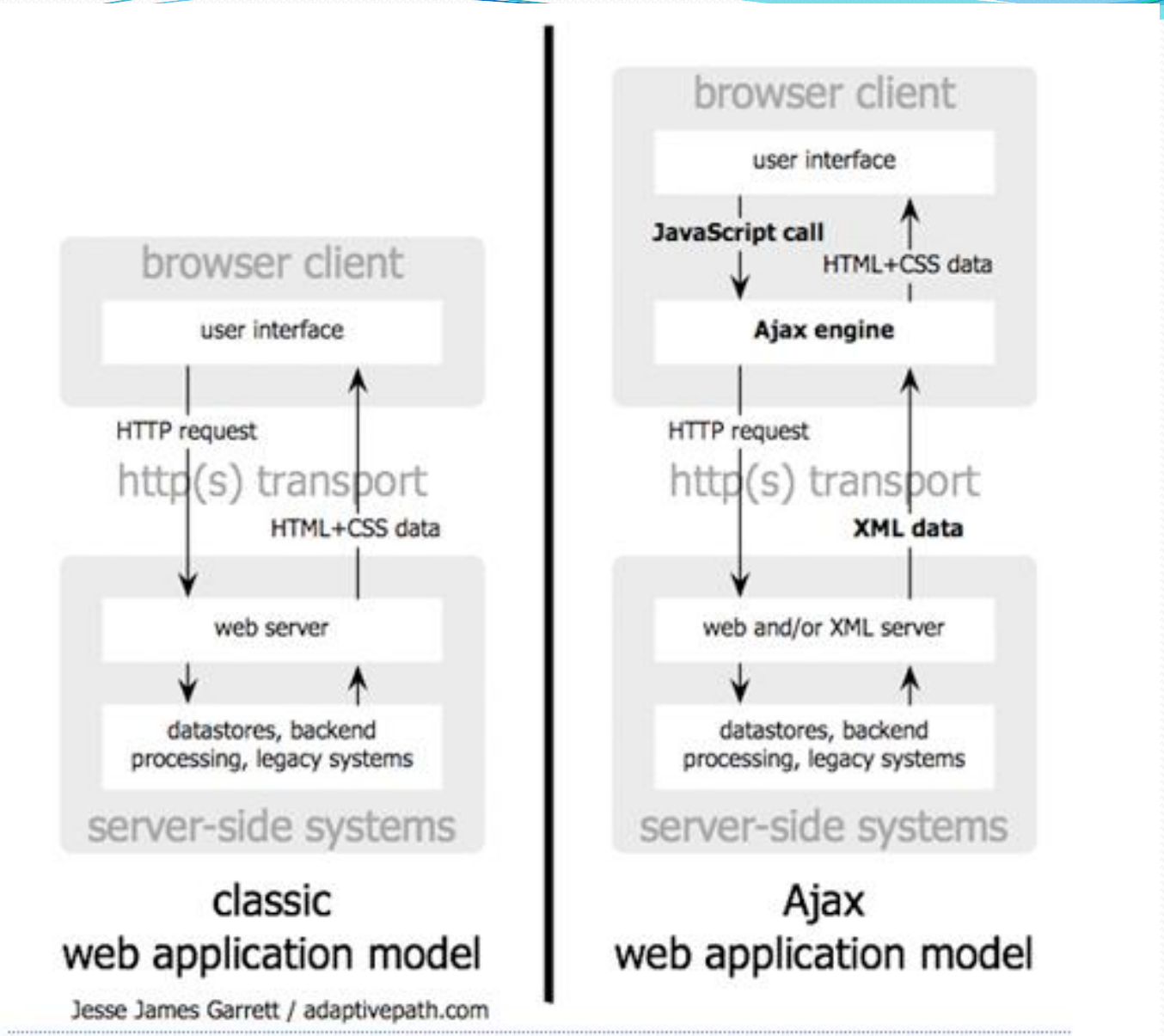
Infrequent large updates



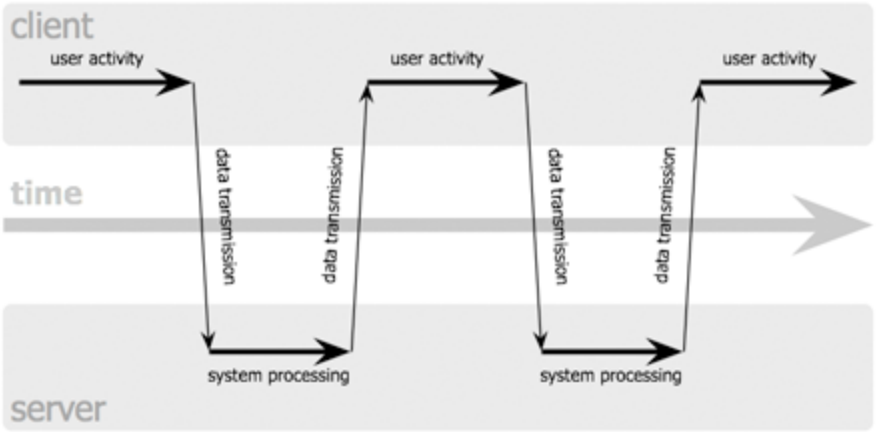
Frequent small updates



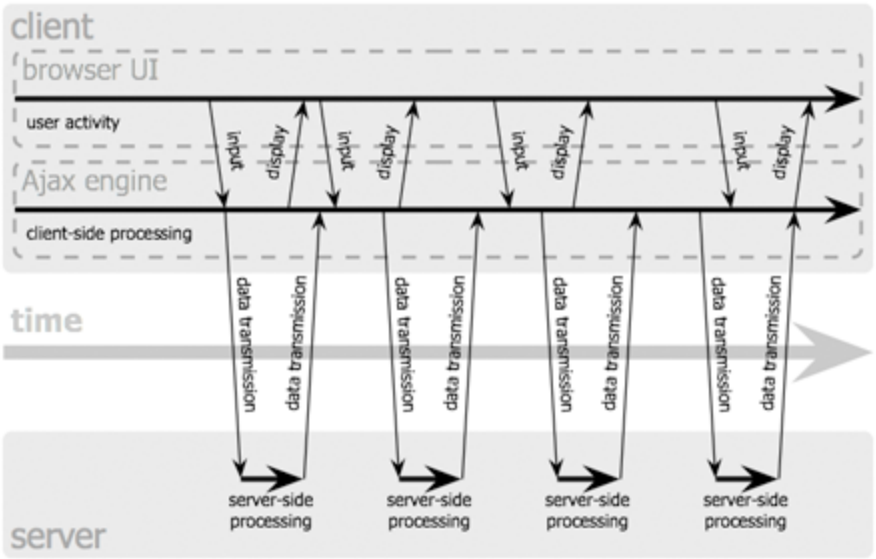
Diagrams from *Ajax in Action* by Dave Crane et al, Manning Press


browser client
user interface
HTTP request
http(s) transport
HTML+CSS data
web server
datastores, backend processing, legacy systems
server-side systems
classic web application model
Jesse James Garrett / adaptivepath.com
browser client
user interface
JavaScript call
HTML+CSS data
Ajax engine
HTTP request
http(s) transport
XML data
web and/or XML server
datastores, backend processing, legacy systems
server-side systems
Ajax web application model

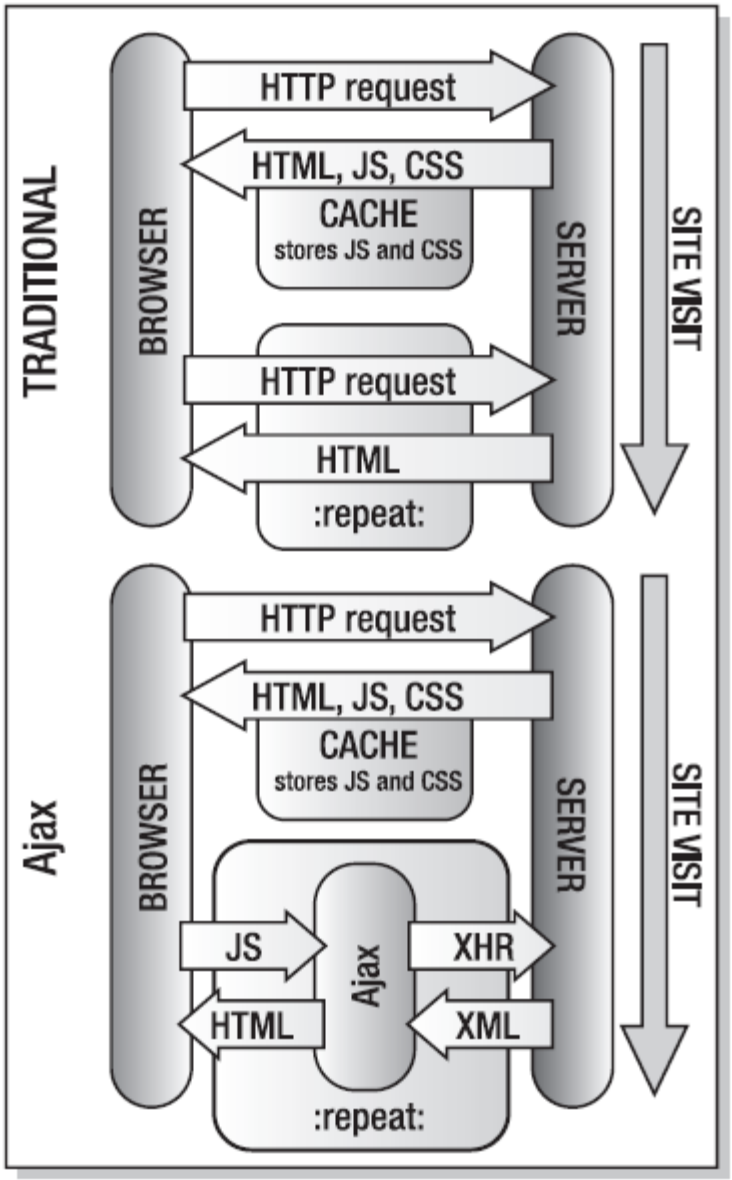
classic web application model (synchronous)



Ajax web application model (asynchronous)



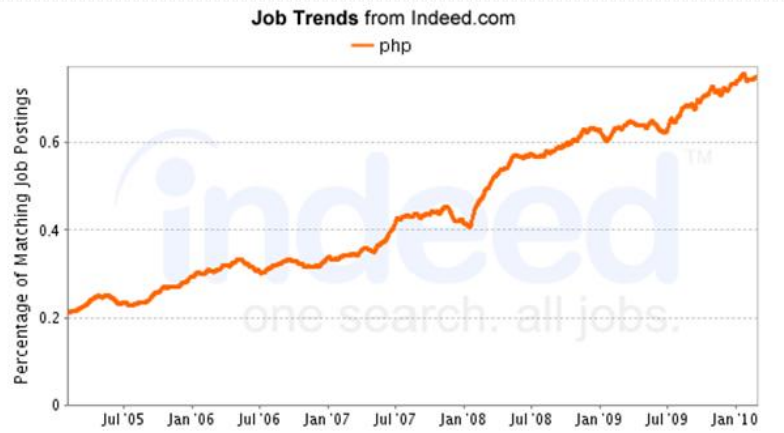
Jesse James Garrett / adaptivepath.com

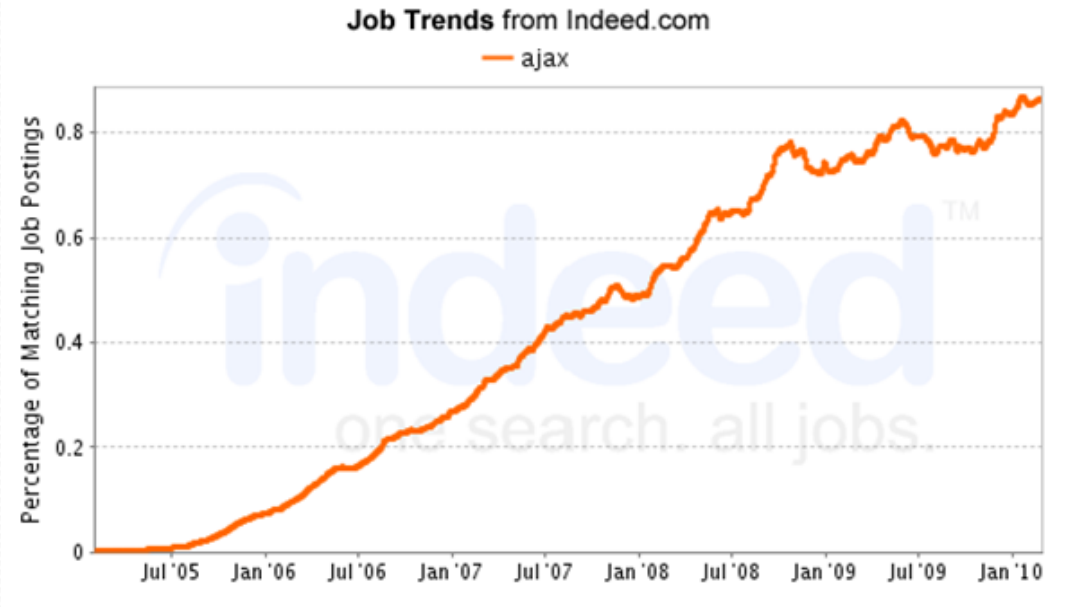

TRADITIONAL
BROWSER
SERVER
HTTP request
HTML, JS, CSS
CACHE
stores JS and CSS
HTTP request
HTML
:repeat:
SITE VISIT
Ajax
BROWSER
SERVER
HTTP request
HTML, JS, CSS
CACHE
stores JS and CSS
JS
Ajax
XHR
HTML
XML
:repeat:
SITE VISIT

Veel Ajax näiteid

- http://maps.google.com/
- http://demo.backbase.com/explorer/
- http://www.laszlosystems.com/demos/
- http://www.smartclient.com/index.jsp#_Welcome
- http://www.simplica.com/

Töökohad

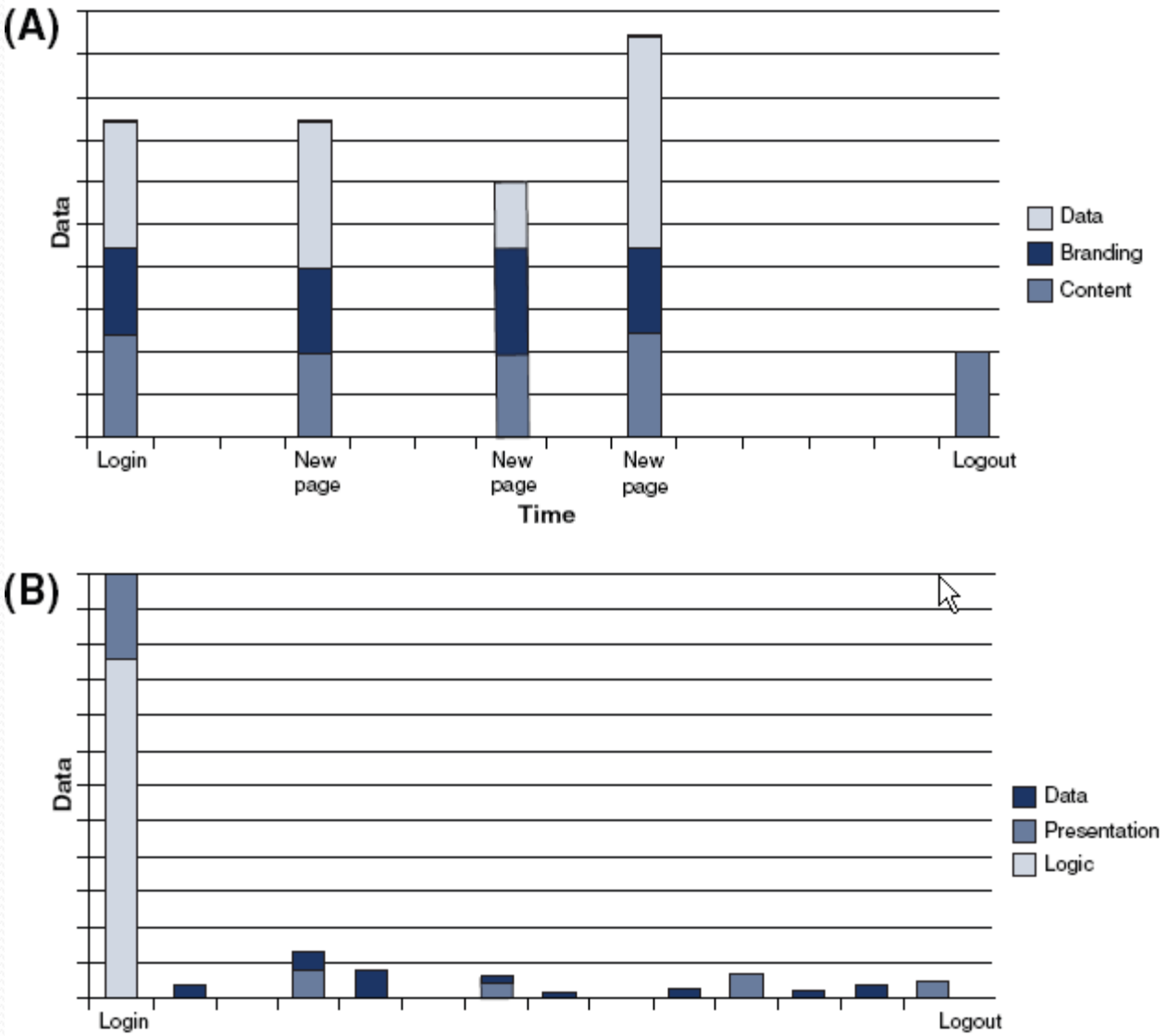


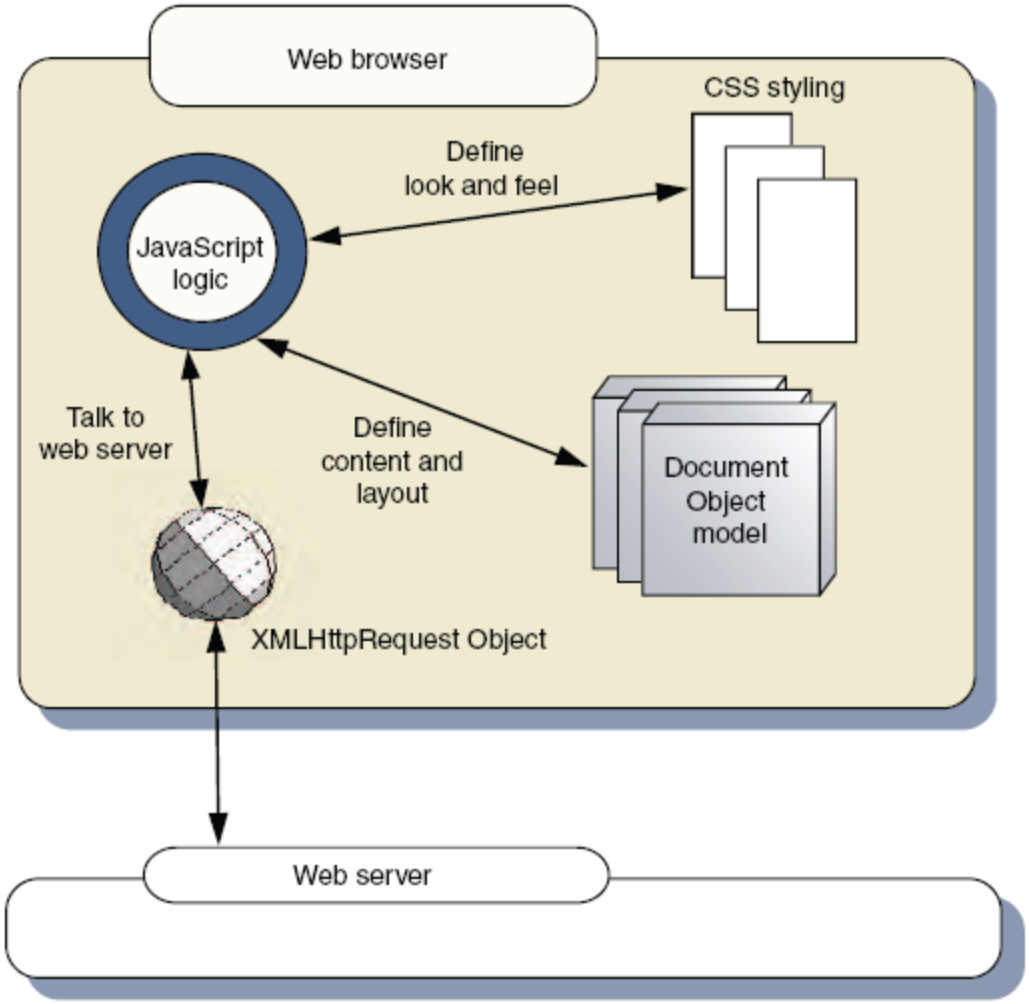


Juuli 05 – Jaanuar 10

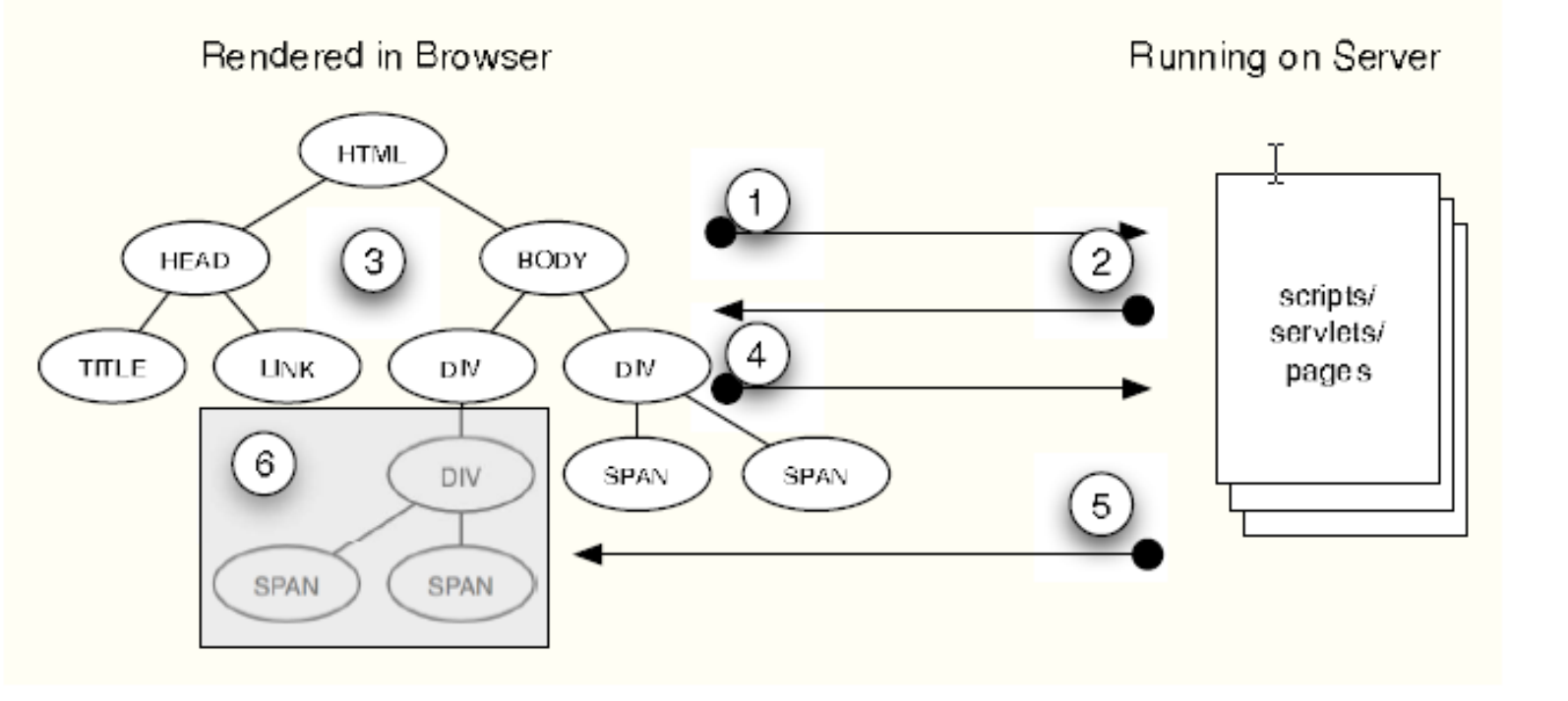
Töökohad


Job Trends from Indeed.com
java
ajax
php
Percentage of Matching Job Postings
3
2
1
0
Jul '05
Jan '06
Jul '06
Jan '07
Jul '07
Jan '08
Jul '08
Jan '09
Jul '09
Jan '10


(A)
Data
Login
New page
New page
New page
Logout
Time
Data
Branding
Content
(B)
Data
Login
Logout
Data
Presentation
Logic


Web browser
CSS styling
Define
look and feel
JavaScript
logic
Talk to
web server
Define
content and
layout
Document
Object
model
XMLHttpRequest Object
Web server

AJAX lehe elutsükkel

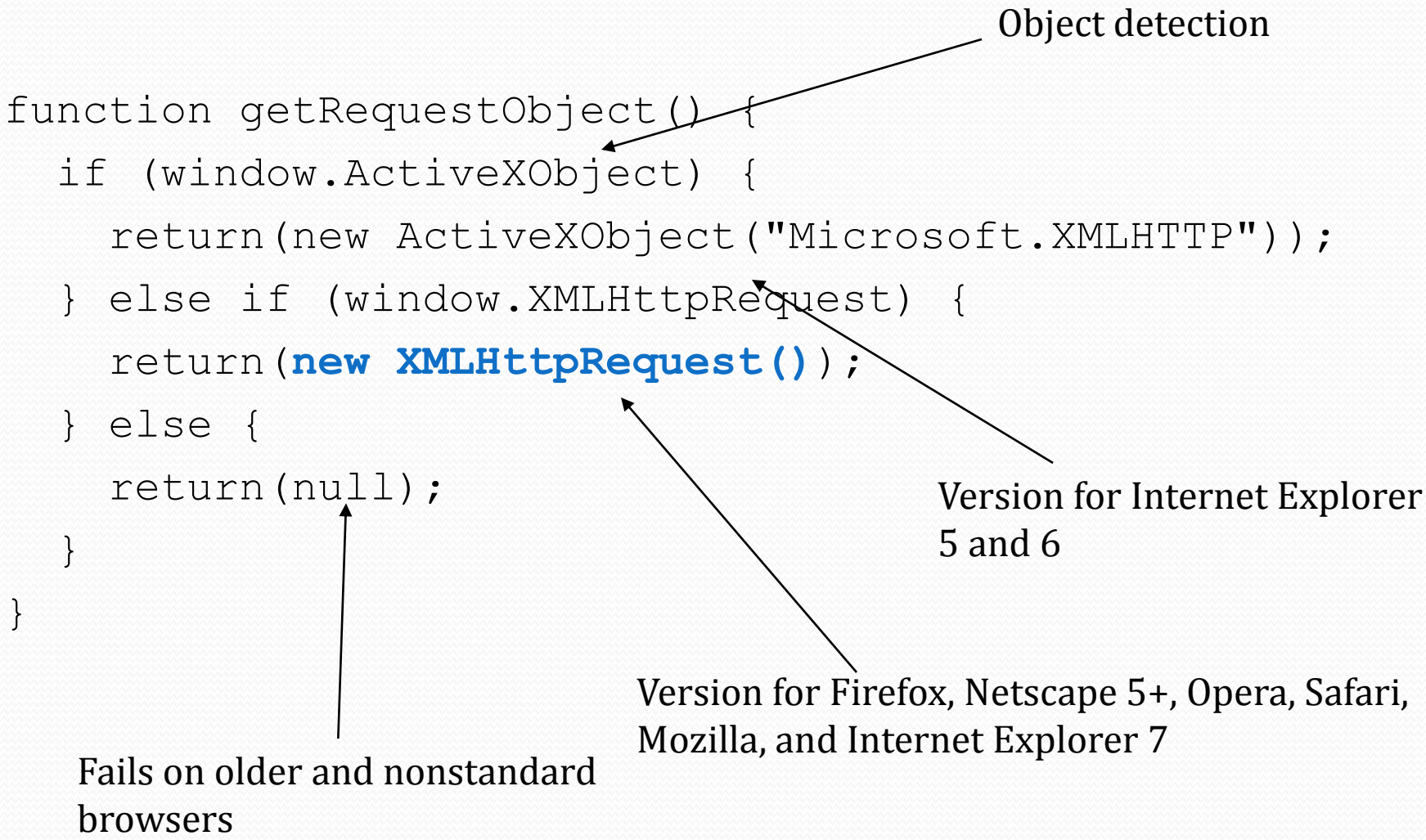


Justin Gehtland, Ben Galbraith, Dion Almaer

Ajax põhiprotsess

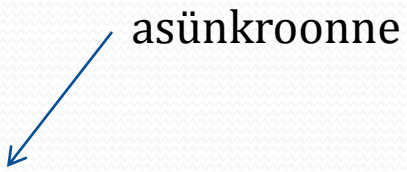
- JavaScript
  - Defineerime objekti HTTP päringu saatmiseks
  - Päringu initsialiseerimine
    - Päringu loomine
    - Määrata anonüümne vastuse töötlemise funktsioon
      - **`onreadystatechange`** atribuut
    - Määrata GET või POST
    - Saada andmed
  - Vastuse töötlemine
    - Oodata **`readyState 4`** ja HTTP staatus 200
    - Eraldada vastuse tekst (`responseText` või `responseXML`)
- HTML
  - Laadida JavaScript
  - Anda juhtimine sellele, mis initsialiseerib päringu
  - Lisada **`id`** sisendelementidele ja väljundi kohatäitjatele

Päringuobjekti defineerimine



Päringu saatmine

```
function sendRequest(url, params, HttpMethod){
  if (!HttpMethod){
      HttpMethod = "POST";
  }
  var req = getRequestObject();
  if (req != null){
      req.open(HttpMethod, url, true);
      req.setRequestHeader("Content-Type",
      "application/x-www-form-urlencoded");
      req.send(params);
  }
}
```



Päringu saatmine ja vastuse töötlemine

```
function sendRequest() {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { handleResponse(request) };
  request.open("GET", "message-data.html", true);
  request.send(null);
}
```

Code to call when server responds

URL of server-side resource. Must be on same server that page was loaded from.

POST data (always null for GET requests)

Don't wait for response (Send request asynchronously)

Vastuse töötlemine

```
function handleResponse(request) {
  if (request.readyState == 4) {
    alert(request.responseText);
  }
}
```

Response from server is complete (handler gets invoked multiple times)

Text of server response

Pop up dialog box

Funktsioonidest JavaScriptis

- JavaScript lubab funktsioone eraldi defineerida
  - `function doSomethingWithResponse() { code }`
  - `request.onreadystatechange = doSomethingWithResponse;`
  - Java seda ei luba

- JavaScript lubab anonüümseid funktsioone:
  - `var request = getRequestObject();`
  - `request.onreadystatechange = function() {` kood, mis kasutab muutujat `request};`

  - Javas on anonüümsed klassid, kuid pole anonüümseid funktsioone.

Mittekorrektne lähenemine (globaalne muutuja `Request`)

```
var request;

function getRequestObject() { ... }

function sendRequest() {
  request = getRequestObject();
  request.onreadystatechange = handleResponse;
  request.open("GET", "...", true);
  request.send(null);
}

function handleResponse() {
  if (request.readyState == 4) {
    alert(request.responseText);
}
```

See lähenemine on raamatutes
Foundations of Ajax,
Ajax in Practice,
Ajax in Action,
JavaScript the Definitive Guide

Probleem

- Stsenaarium
  - Olgu meil kaks xhtml nuppu, esimest töötleb `function1` ja teist töötleb `function2`
  - `function1` töötlemine võtab 5 sekundit serverilt tulemuse saamiseks
  - `function2` töötlemine võtab 1 sekundit serverilt tulemuse saamiseks

Probleem

- Oletame, et kasutaja klikkis `button1` ja seejärel `button2`.
  - Kui `function1` otsib `request.responseText`, siis ta saab `function 2` vastuse teksti!
  - Funktsioonil, mis töötleb `onreadystatechange`, on null argumenti.
- Lahendus
  - Kasutada anonüümset funktsiooni koos *lokaalse* koopiaga päringuobjektist.

Korrigeeritud lähenemine (lokaalne päringumuutuja)

```
function getRequestObject() { ... }

function sendRequest() {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { handleResponse(request); };
  request.open("GET", "...", true);
  request.send(null);
}

function handleResponse(request) {
  ...
}
```

Terviklik JavaScript kood (show-message.js)

```
function getRequestObject() {
  if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else {
    return(null);
  }
}
```

```
function sendRequest() {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { handleResponse(request); };
  request.open("GET", "message-data.html", true);
  request.send(null);
}
```

```
function handleResponse(request) {
  if (request.readyState == 4) {
    alert(request.responseText);
  }
}
```

HTML Code

- Use xhtml, not HTML 4
  - In order to manipulate it with DOM
    - <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    - "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    - <html xmlns="http://www.w3.org/1999/xhtml">...</html>
    - Due to IE bug, do not use XML header before the DOCTYPE
- Load the JavaScript file
    - <script src="*relative-url-of-JavaScript-file*"
    - type="text/javascript"></script>
    - Use separate </script> end tag
- Designate control to initiate request
    - <input type="button" value="*button label*"
    - onclick="*mainFunction*()"/>

Internet Explorer XHTML Bugs

- Can't handle XML header
  - XML documents in general are supposed to start with XML header:
    - <?xml version="1.0" encoding="UTF-8"?>
      <!DOCTYPE html ...>
      <html xmlns="http://www.w3.org/1999/xhtml">...</html>

    Omit this!
  - XHTML specification recommends using it
  - *But*... Internet Explorer will switch to quirks-mode (from standards-mode) if DOCTYPE is not first line.
    - Many recent style sheet formats will be ignored
    - So omit XML header
- Needs separate end tags in some places
  - Scripts will not load if you use <script .../> instead of <script...></script>

Don't do this.

Do this.

HTML kood (show-message.html)

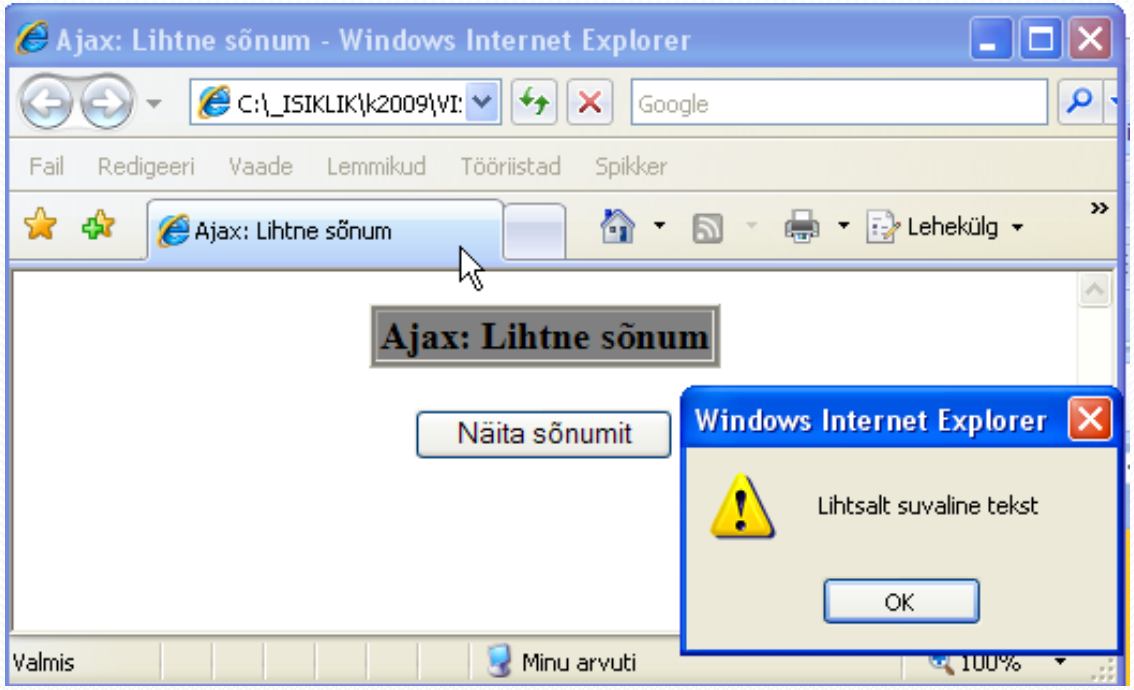
```
<!DOCTYPE html PUBLIC "..."
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Ajax: Lihtne sõnum</title>
<script src="show-message.js"
        type="text/javascript"></script>
</head>
<body> <center>
<table border="1" bgcolor="gray">
  <tr><th><big>Ajax: Lihtne sõnum</big></th></tr>
</table>
<p/>
<form action="#">
 <input type="button" value= "Näita sõnumit"
        onclick="sendRequest()"/>
</form> </center></body></html>
```

HTML kood (`message-data.html`)

Lihtsalt suvaline tekst

- Märkus: seda näidet võibvaadata otse brauseriga, sest tal on staatiline sisu
  - Edasised näited juba Tomcati peal.

The Basic Process: Results



Dünaamiline sisu JSPst

Esimeses näites:

- Sisu sama igal päringul
  - Sama tulemus kui `alert` JavaScript abil
  - Selle asemel kaasata JSP leht serverilt
- Ressursi aadress on JavaScriptis
  - Takistab funktsiooni kasutamist eri olukordades
  - Selle asemel teha üldine funktsioon
- JavaScript fail oli samas kaustas nagu HTML
  - Raske kasutada JavaScripti eri lehtedel
  - Selle asemel teha kaust JavaScripti failide jaoks
- Stiililehti ei kasutatud
  - Kasutada CSS

Sammud

- JavaScript
  - Defineerida objekt HTTP päringute saatmiseks
  - Initsialiseerida päring
    - Teha päringuobjekt
    - Määrata anonüümne vastuse töötlemise funktsioon
      - `onreadystatechange` atribuut
    - Initsialiseerida GET **või POST** päring **JSP lehele**
    - Saata andmed
  - Vastuse töötlemine
    - Oodata `readyState` 4 **ja HTTP status 200**
    - Eraldada vastuse tekst `responseText` **või `responseXML`**
    - Töödelda
- HTML
  - Laadida JavaScript **kesksest kaustast. Kasutada stiililehti.**
  - Määrata kasutajaelement, mis algatab päringu
  - **Anda `id` väljundi kohahoidjale**

Defineerida päringu objekt

```
function getRequestObject() {
  if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else {
    return(null);
  }
}
```

Muutusi võrreldes eelmise näitega ei ole

Algatada päring

```
function ajaxAlert(address) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseAlert(request); };
  request.open("GET", address, true);
  request.send(null);
}
```

Relatiivne serveripoolne ressursi URL (Selles näites me edastame JSP lehe aadressi)

Vastuse töötlemine

```
function showResponseAlert(request) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    alert(request.responseText);
  }
}
```

Serveri vastus tuli tagasi veata (HTTP status code 200).

Terviklik JavaScript kood (ajax-basics-1.js)

```
function getRequestObject() {
  if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else {
    return(null); }}
```

```
function ajaxAlert(address) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseAlert(request); }
  request.open("GET", address, true);
  request.send(null);}
```

```
function showResponseAlert(request) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    alert(request.responseText);
  }
}
```

HTML kood

- Laadida JavaScript

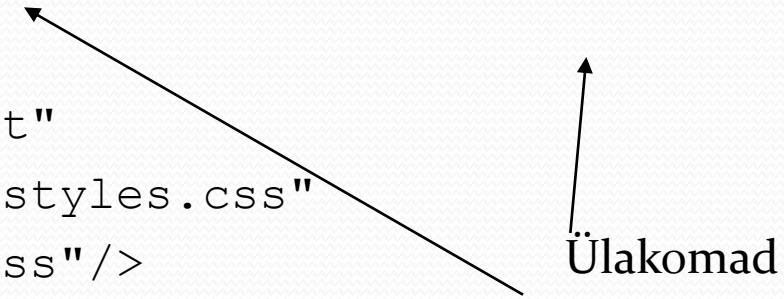
```
<script src="./scripts/ajax-basics.js"
  type="text/javascript"></script>
```

- Edastada JSP aadress funktsioonile

```
<input type="button" value="Show Server Time"
            onclick='ajaxAlert("show-time.jsp")'/>
```

- Kasutada stiililehti

```
<link rel="stylesheet"
        href="./css/styles.css"
        type="text/css"/>
```



HTML kood

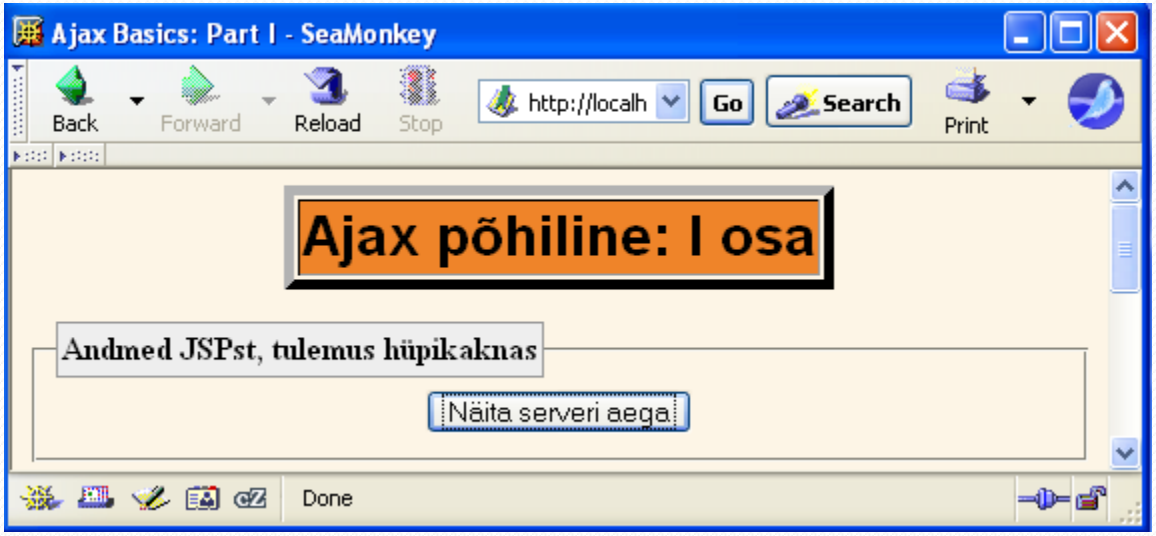
```
<!DOCTYPE html PUBLIC "...">
<html xmlns="http://www.w3.org/1999/xhtml">...
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<script src="./scripts/ajax-basics.js"
        type="text/javascript"></script>...
<body>...
<fieldset>
  <legend>Andmed JSPst, tulemus hüpikaknas </legend>
  <form action="#">
    <input type="button" value= "Näita serveri aega"
           onclick='ajaxAlert("show-time.jsp")'/>
  </form>
</fieldset>
...
```

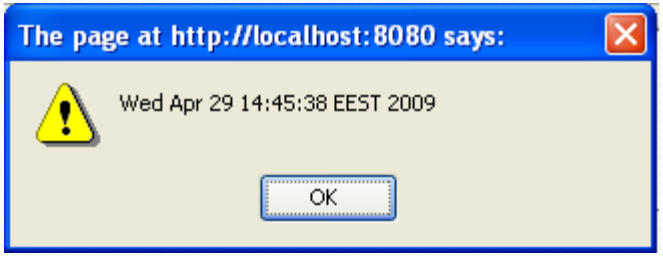
JSP kood (`show-time.jsp`)

```
<%= new java.util.Date() %>
```

- Käivitada Tomcatil

Tulemus JSPst





Dünaamiline sisu servletist
JSP näide: disaini puudused

- Puhverdamise probleemid
  - URL on sama, kuid väljund muutub
  - Kui brauser puhverdab lehe, siis tulemuseks vale aeg
  - Lahendus: saata päised `Cache-Control` ja `Pragma`

- Aeg ei ole formaaditud
  - kasutati `Date` meetodit `toString`

  - Lahendus: kasutada `String.format`
  - JSP on parim, kui palju HTML ja vähe loogikat/Javat
  - Hetkel meil on loogika, kui pole HTML
  - Lahendus: kasutada servlette

Etapid

- JavaScript
  - Defineerida objekt HTTP päringute saatmiseks
  - Initsialiseerida päring
    - Teha päringuobject
    - Määrata anonüümne vastuse töötlemise funktsioon
      - `onreadystatechange` atribuut
    - Initsialiseerida GET **või POST** päring **servletile**
    - Saata andmed
  - Vastuse töötlemine
    - Oodata `readyState` 4 **ja HTTP status 200**
    - Eraldada vastuse tekst `responseText` **või `responseXML`**
    - Töödelda
- HTML
  - Laadida JavaScript **kesksest kaustast. Kasutada stiililehti.**
  - Määrata kasutajaelement, mis algatab päringu
  - **Anda `id` väljundi kohahoidjale**

Defineerida päringuobjekt, initsialiseerida päring,töödelda vastust

```
function getRequestObject() {
  if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else {
    return(null);
  }
}

function ajaxAlert(address) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseAlert(request); }
  request.open("GET", address, true);
  request.send(null);
}

function showResponseAlert(request) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    alert(request.responseText);
  }
}
```

Muutus ainult aadress, mis tuleb HTML lehelt

HTML kood

```
...
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<script src="./scripts/ajax-basics.js"
        type="text/javascript"></script>
...
<fieldset>
  <legend>
     Andmed servletist, tulemus hüpikaknas
  </legend>
  <form action="#">
    <input type="button" value="Show Server Time"
           onclick='ajaxAlert("show-time")'/>
  </form>
</fieldset>
...
```

Servleti aadress (`servlet-mapping url-pattern`).

Servleti kood

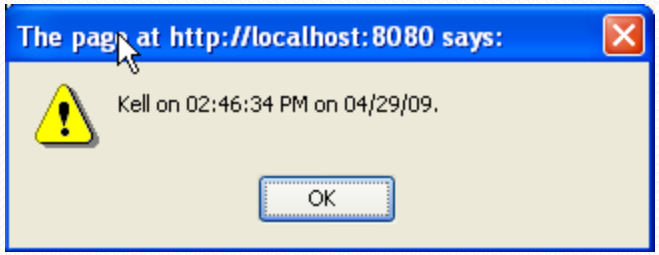
```
package aservlet;
import ...

public class ShowTime extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    PrintWriter out = response.getWriter();
    Date currentTime = new Date();
    String message =
      String.format("Kell on %tr on %tD.",
                    currentTime, currentTime);
    out.print(message);
  }
}
```

web.xml

```
...
  <servlet>
    <servlet-name>ShowTime</servlet-name>
    <servlet-class>aservlet.ShowTime</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ShowTime</servlet-name>
    <url-pattern>/show-time</url-pattern>
  </servlet-mapping>
...
```

Tulemused


Ajax Basics: Part I - SeaMonkey
Back
Forward
Reload
Stop
http://localh
Go
Search
Print
Ajax põhiline: I osa
Andmed JSPst, tulemus hüpikaknas
Näita serveri aega
Andmed servletist, tulemus hüpikaknas
Näita serveri aega
Done


The page at http://localhost:8080 says:
Kell on 02:46:34 PM on 04/29/09.
OK

HTML väljundi genereerimine
`ShowTime` servlett: disaini puudused

- Tulemusi näidatakse hüpikaknas
  - Tavaliselt tehakse seda vigade korral
  - Kasutajad eelistavad normaalseid tulemusi lehe sees
  - Lahendus: kasutada dünaamilist HTML et korrigeerida tulemust
    - HTML + CSS stiilid on DOM-is
    - JavaScript oskab panna elemente DOM-i
      - Leida element antud id-ga:
        - `someElement = document.getElementById(id);`
      - Lisada HTML-i
        - `someElement.innerHTML = "<h1>tekst</h1>";`
    - JavaScript oskab lugeda DOM-ist
      - Nt tekstielemendi sisu
        - `document.getElementById(id).value`

Teksti paigutamine dünaamiliselt

- HTML

```
<div id="results-placeholder"></div>
```

- JavaScript

```
resultRegion =
     document.getElementById("results-placeholder");
```

  - `resultRegion.innerHTML = "<h2>Wow!</h2>";`

- Tulemus

```
<div id="results-placeholder"><h2>Wow!</h2></div>
```

- Ettevaatust
  - Kontrollida, et tulemus on legaalne xhtml
    - Kasutada õiget registrit

Vahekokkuvõte

- HTML
  - Defineerida algselt tühi `div` element

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

- JavaScript vastusetöötlejas
  - Varustada `id` (`resultRegion`), leida selle `id`-ga element, ja lisada vastuse tekst `innerHTML` omadusse

```
document.getElementById(resultRegion).innerHTML
  = request.responseText;
```

Sammud

- JavaScript
  - Defineerida objekt HTTP päringute saatmiseks
  - Initsialiseerida päring
    - Luua päringuobjekt
    - Määrata anonüümne vastuse töötlemise funktsioon
      - `onreadystatechange` atribuut
    - Initsialiseerida GET või **POST** päring **servletile**
    - Saata andmed
  - Töödelda vastust
    - Oodata `readyState` 4 ja HTTP status 200
    - Eraldada vastuse tekst `responseText` **või `responseXML`**
    - **Kasutada `innerHTML` tulemuse paigutamiseks**
- HTML
  - Laadida JavaScript, kasutada stiililehti.
  - Määrata kasutajaelement, mis initsialiseerib päringu
  - **Anda `id` väljundipiirkonnale**

Defineerida päringuobjekt

```
function getRequestObject() {
  if (window.ActiveXObject) {
    return(new
  ActiveXObject("Microsoft.XMLHTTP"));
  } else if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else {
    return(null);
  }
}
```

Muutusi ei ole

Initsialiseerida päring

```
function ajaxResult(address, resultRegion) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseText(request,
                                  resultRegion); };
  request.open("GET", address, true);
  request.send(null);
}
```

Töödelda vastust

```
function showResponseText(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    document.getElementById(resultRegion).innerHTML
  = request.responseText;
  }
}
```

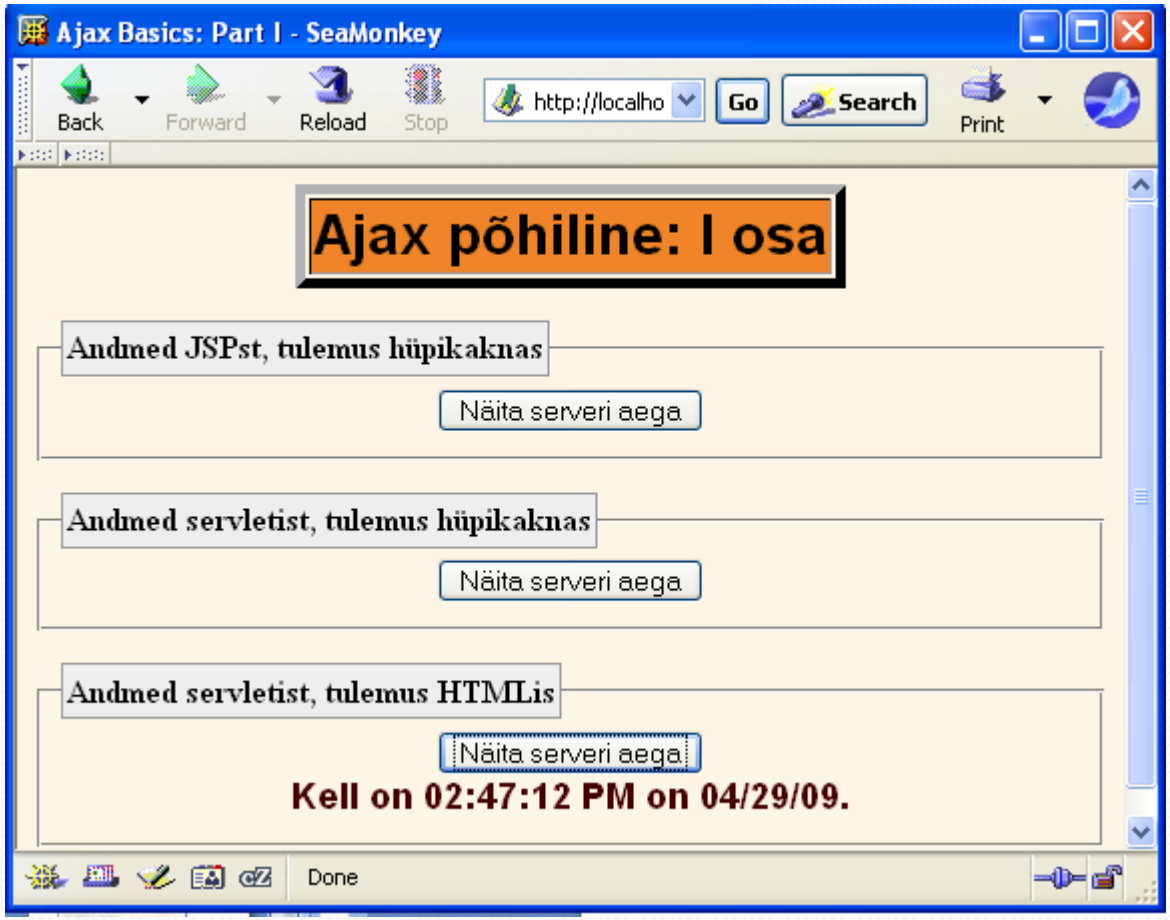
HTML kood

```
...
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<script src="./scripts/ajax-basics.js"
        type="text/javascript"></script>
...
<fieldset>
  <legend>Data from Servlet, Result Shown in HTML</legend>
  <form action="#">
    <input type="button" value="Show Server Time"
           onclick='ajaxResult("show-time",
                               "timeResult1")'/>
  </form>
  <div id="timeResult1" class="ajaxResult"></div>
</fieldset>
...
```

Stiililehed (`css/styles.css`)

```
.ajaxResult { color: #440000;
              font-weight: bold;
              font-size: 18px;
              font-family: Arial, Helvetica, sans-
  serif;
}
```

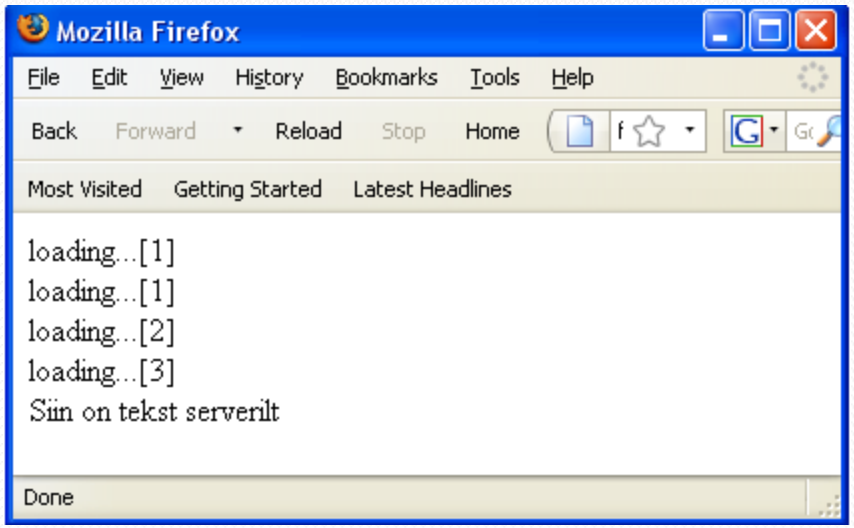
HTML väljund: Tulemused



```
<html>
<head>
<script type='text/javascript'>
var req=null;
var console=null;
var READY_STATE_UNINITIALIZED=0;
var READY_STATE_LOADING=1;
var READY_STATE_LOADED=2;
var READY_STATE_INTERACTIVE=3;
var READY_STATE_COMPLETE=4;
function sendRequest(url,params,HttpMethod){
if (!HttpMethod){
HttpMethod="GET";
}
req=initXMLHTTPRequest();
if (req){
req.onreadystatechange=onReadyState;
req.open(HttpMethod,url,true);
req.setRequestHeader
("Content-Type", "application/x-www-form-urlencoded");
req.send(params);
}
}
```

```
function initXMLHTTPRequest(){
var xRequest=null;
if (window.XMLHttpRequest){
xRequest=new XMLHttpRequest();
} else if (window.ActiveXObject){
xRequest=new ActiveXObject
("Microsoft.XMLHTTP");}
return xRequest;}

function onReadyState(){
var ready=req.readyState;
var data=null;
if (ready==READY_STATE_COMPLETE){
data=req.responseText;
}else{
data="loading...["+ready+"]“;}
toConsole(data);}
```

```
function toConsole(data){
if (console!=null){
var newline=document.createElement("div");
console.appendChild(newline);
var txt=document.createTextNode(data);
newline.appendChild(txt);
}
}
window.onload=function(){
console=document.getElementById('console');
sendRequest("data.txt");
}
</script>
</head>
<body>
<div id='console'></div>
</body>
</html>
```


Mozilla Firefox
File Edit View History Bookmarks Tools Help
Back Forward Reload Stop Home
Most Visited Getting Started Latest Headlines
loading...[1]
loading...[1]
loading...[2]
loading...[3]
Siin on tekst serverilt
Done

Kokkuvõte

- JavaScript
  - Defineerida päringuobjekt
    - Kontrollida Microsoft ja mitte-MS objekte.
  - Initsialiseerida päring
    - Luua päringuobjekt
    - Määrata anonüümne vastuse töötlemise funktsioon
    - Initsialiseerida GET päring
  - Töödelda vastust
    - Oodata readyState 4 ja HTTP status 200
    - Eraldada vastuse tekst responseText
    - Teha midagi tulemusega
      - Kasutada innerHTML tulemuse paigutamiseks elementi
- HTML
  - Anda id elemendile (tavaliselt div). Algatada protsess.
- Java
  - Kasutada JSP ja/või servlette
  - Takistada brauseril puhverdamist.

Viited

- Tutorials:
  - http://www.w3schools.com/ajax/default.asp
  - http://www.ajaxtutorial.net/
  - http://www.xul.fr/en-xml-ajax.html

James Garret:
http://www.adaptivepath.com/ideas/essays/archives/000385.php

62