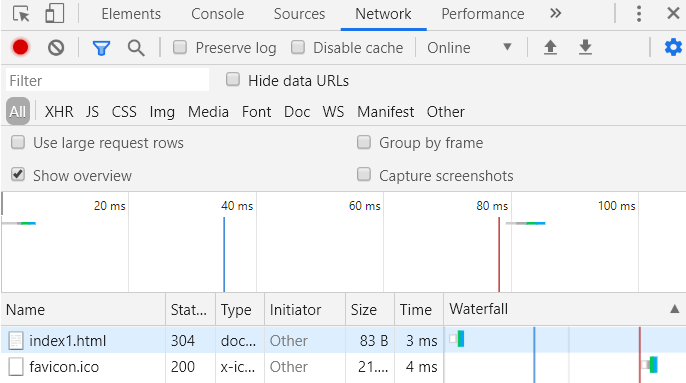
The HTTP Protocol

*Before you start, you should create a document (or a just a piece of paper) where you should write down the Status Code generated by each of the following exercises (You need this for exercise 4-c)*

1) Monitoring HTTP Headers 1



Filnavn index1

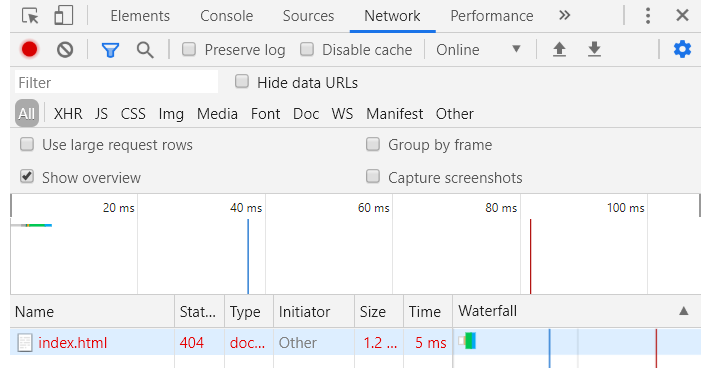
Stat 304, er kode for at ok.

type = document

size = netværktranfered

time = ping

Waterfall er en visuel præsentation med for f.eks. css bliver loaded.



Filnavn index

Stat 404, er kode for at at filen ikke findes. Hvilket er klart, da jeg ændrede navnet.

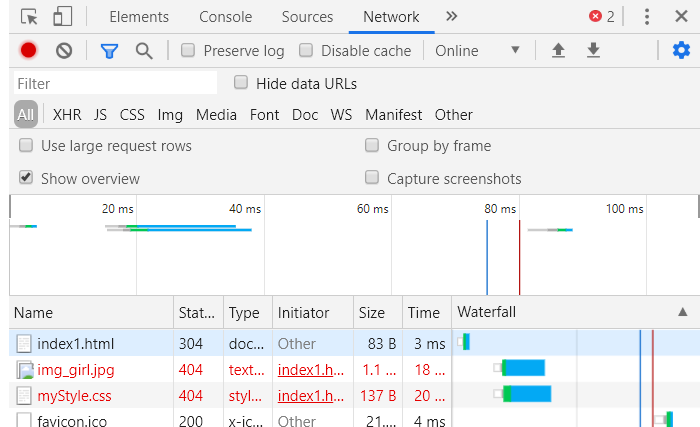
type = document

size = netværktranfered

time = ping

Waterfall er en visuel præsentation med for f.eks. css bliver loaded.

2) Monitoring HTTP Headers 2



får en kode det hedder 404 fordi filen ikke filen findes. Grunden til at jeg også får det på billedet er fordi jeg ikke har sat billedet ind i den rigtige placering. Min ping og waterfall stiger pga større indhold.

3) Monitoring HTTP Headers 3  (Response-codes 3xx)

Redirect får type 302

r.html type 200

vi har sat den i serlvleten til at redirecte til ” r.html.

3a) Redirecting to HTTPs instead of HTTP

Type 301 og type 200

Type 301 he HTTP response status code **301** Moved Permanently is used for permanent URL redirection, **meaning** current links or records using the URL that the response is received for **should** be updated. ... The **301** redirect is considered a best practice for upgrading users from HTTP to HTTPS.

Type 200

This class of **status codes** indicates the action requested by the client was received, understood and accepted. **200** OK. Standard response for successful HTTP requests.

4a) Status Codes (5xx)

# HTTP Status 500 – Internal Server Error

4b) Status Codes (4xx)

http status 404

4c) Status Codes - Ranges

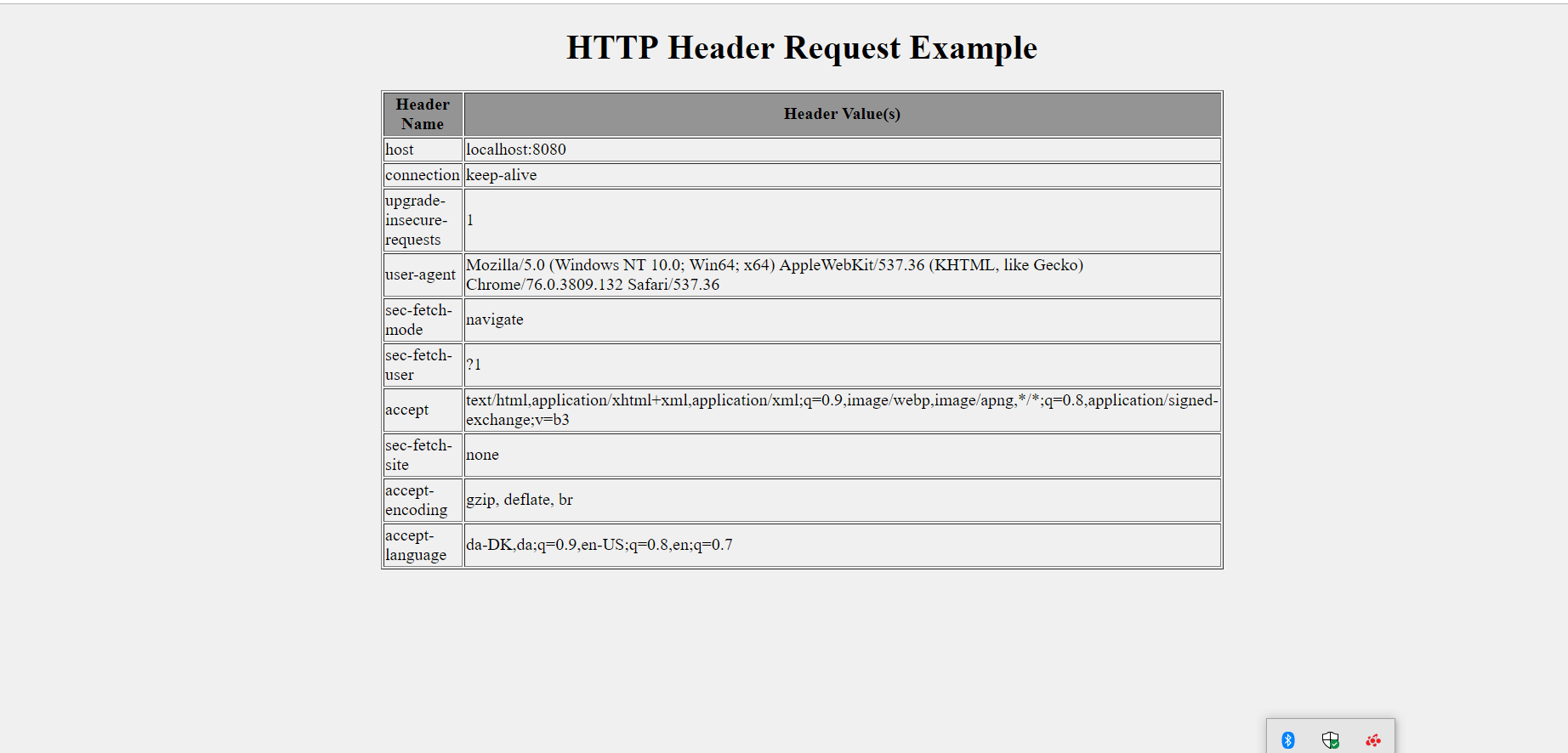
200 means ok

300  Multiple Choices redirect status response code indicates that the request has more than one possible responses

400 Bad Request **error** is an **HTTP** status code

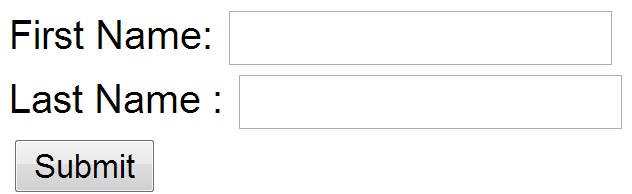
500 Internal Server **Error** is a very general **HTTP**status code that **means** something has gone wrong on the website's server, but the server could not be more specific on what the exact problem is

5) Get HTTP Request Headers on the Server



Kotankt hvis kode ønskes. artin-4000@hotmail.com

6) Get/Post-parameters

Create a new HTML-file in the web-project made in exercise 1.

Add a form to the file, including two text input boxes and a submit button as sketched below:

Add an extra input field to the form with type=”hidden”, name=”hidden” and value=12345678.

Add the value “#” for the forms action attribute.

Set the forms method-attribute to the value “GET” (actually the default value) and test the form. Observe what happens in your browser's address field.

Change the forms method-attribute to the value “POST” and test the form. Observe the change in your browsers address field. Figure out (using Chrome Developer Tools), how parameters are passed in, for a POST request.

Post gets type 200

Get gets type 300

Write down your observations

Session and Cookies

*For the next two exercises/demos you should create a new Maven web-project. Both the demos use a Servlet.*

7)      Sessions (Session Cookies)

In your web project use the wizard to generate a  new Servlet

1. Enter ***SessionDemo*** as the name of the Servlet and *servlets* as package name
2. Right-click the file and select Run to see “what is does”
3. Change the generated processRequest(..) method as sketched below.

protected void processRequest**(**HttpServletRequest request**,**

                              HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

    String name **=** request**.**getParameter**(**"name"**);**

**if** **(**name **!=** **null)** **{**

      request**.**getSession**().**setAttribute**(**"name"**,** name**);**

**}** **else** **{**

      name **=** **(**String**)** request**.**getSession**().**getAttribute**(**"name"**);**

**}**

    response**.**setContentType**(**"text/html;charset=UTF-8"**);**

**try** **(**PrintWriter out **=** response**.**getWriter**())** **{**

      out**.**println**(**"<!DOCTYPE html>"**);**

      out**.**println**(**"<html>"**);**

      out**.**println**(**"<head>"**);**

      out**.**println**(**"<title>Servlet SessionDemo</title>"**);**

      out**.**println**(**"</head>"**);**

      out**.**println**(**"<body>"**);**

**if** **(**name **!=** **null)** **{**

        name **=** **(**String**)**request**.**getSession**().**getAttribute**(**"name"**);**

        out**.**println**(**"<p> Welcome " **+** name  **+** " !</p>"**);**

**}** **else** **{**

        out**.**println**(**"<h2>Please enter your name, and submit</h2>"**);**

        out**.**println**(**"<form action='SessionDemo'>"**);**

        out**.**println**(**"<input type='input' name='name'>"**);**

        out**.**println**(**"<input type='submit'></form>"**);**

**}**

      out**.**println**(**"</body>"**);**

      out**.**println**(**"</html>"**);**

**}**

**}**

1. Enter your name and press submit, copy the URL in the browser into your clipboard, close the tab (but not the browser) and load the page again in a new tab using the URL in the clipboard.
2. While doing the things in step d, you should monitor the content of your local cookies and the HTTP requests being sent, using the development tools in Chrome.
3. **Most import part of this exercise:**

Explain (on paper) using both words and images how the Server can maintain state between subsequent calls even when the state is not transmitted from the client to server.

Write down your observations

8)     Persistent Cookies

1. In your web project, use the wizard to generate a new servlet
2. Enter *CookieDemo* as the name of the Servlet and *servlets* as package name
3. Change the generated processRequest(..) method as sketched below.

protected void processRequest**(**HttpServletRequest request**,** HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

String name **=** request**.**getParameter**(**"name"**);**

**if** **(**name **!=** **null)** **{**

Cookie cookie **=** **new** Cookie**(**"username"**,** name**);**

cookie**.**setMaxAge**(**60 **\*** 60 **\*** 24 **\*** 365**);**

response**.**addCookie**(**cookie**);**

**}**

Cookie**[]** cookies **=** request**.**getCookies**();**

**if** **(**cookies **!=** **null)** **{**

**for** **(**Cookie cookie **:** request**.**getCookies**())** **{**

**if** **(**cookie**.**getName**().**equals**(**"username"**))** **{**

name **=** cookie**.**getValue**();**

**}**

**}**

**}**

response**.**setContentType**(**"text/html;charset=UTF-8"**);**

**try** **(**PrintWriter out **=** response**.**getWriter**())** **{**

/\* TODO output your page here. You may use following sample code. \*/

out**.**println**(**"<!DOCTYPE html>"**);**

out**.**println**(**"<html>"**);**

out**.**println**(**"<head>"**);**

out**.**println**(**"<title>Servlet CookieDemo</title>"**);**

out**.**println**(**"</head>"**);**

out**.**println**(**"<body>"**);**

**if** **(**name **!=** **null)** **{**

out**.**println**(**"<p> Welcome " **+** name **+** " !</p>"**);**

**}** **else** **{**

out**.**println**(**"<h2>Please enter your name, and submit</h2>"**);**

out**.**println**(**"<form action='CookieDemo'>"**);**

out**.**println**(**"<input type='input' name='name'>"**);**

out**.**println**(**"<input type='submit'></form>"**);**

**}**

out**.**println**(**"</body>"**);**

out**.**println**(**"</html>"**);**

**}**

**}**

1. Enter your name and press submit, copy the URL in the browser into your clipboard, close the tab (but not the browser) and load the page again in a new tab using the URL in the clipboard.
2. Now close your browser (you could even close your laptop, but don’t ;-) , open it again and load the page again using the URL in the clipboard
3. While doing the things in step e, you should monitor the content of your local cookies and the HTTP requests being sent, using the development tools in Chrome.
4. **The most import part of this exercise:**

Explain (on paper) how Cookies can be used to maintain “state” on the client between subsequent calls to a server, even when a browser has been closed down.

Write down your observations