Web Engineering + Design 1

HTTP

Silvan Gehrig

HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL

FHO Fachhochschule Ostschweiz

Lernziele

Die Vorlesung soll die Teilnehmer befähigen, HTTP Request und Response im Grundsatz und mit deren wesentlichen Eigenschaften zu verstehen und dessen Konzepte fachgerecht einzusetzen.

Die Teilnehmer...

- **verstehen, wie ein Web-Request zustande kommt und kennen dessen Ablauf.**
- **können HTTP Protokoll-Probleme zwischen Client und Server interpretieren und deren Fehlerquelle einschätzen.**
- **kennen einige der wichtigsten HTTP Header mit deren Funktionalitäten.**

Table of Contents

HYPERTEXT TRANSFER PROTOCOL

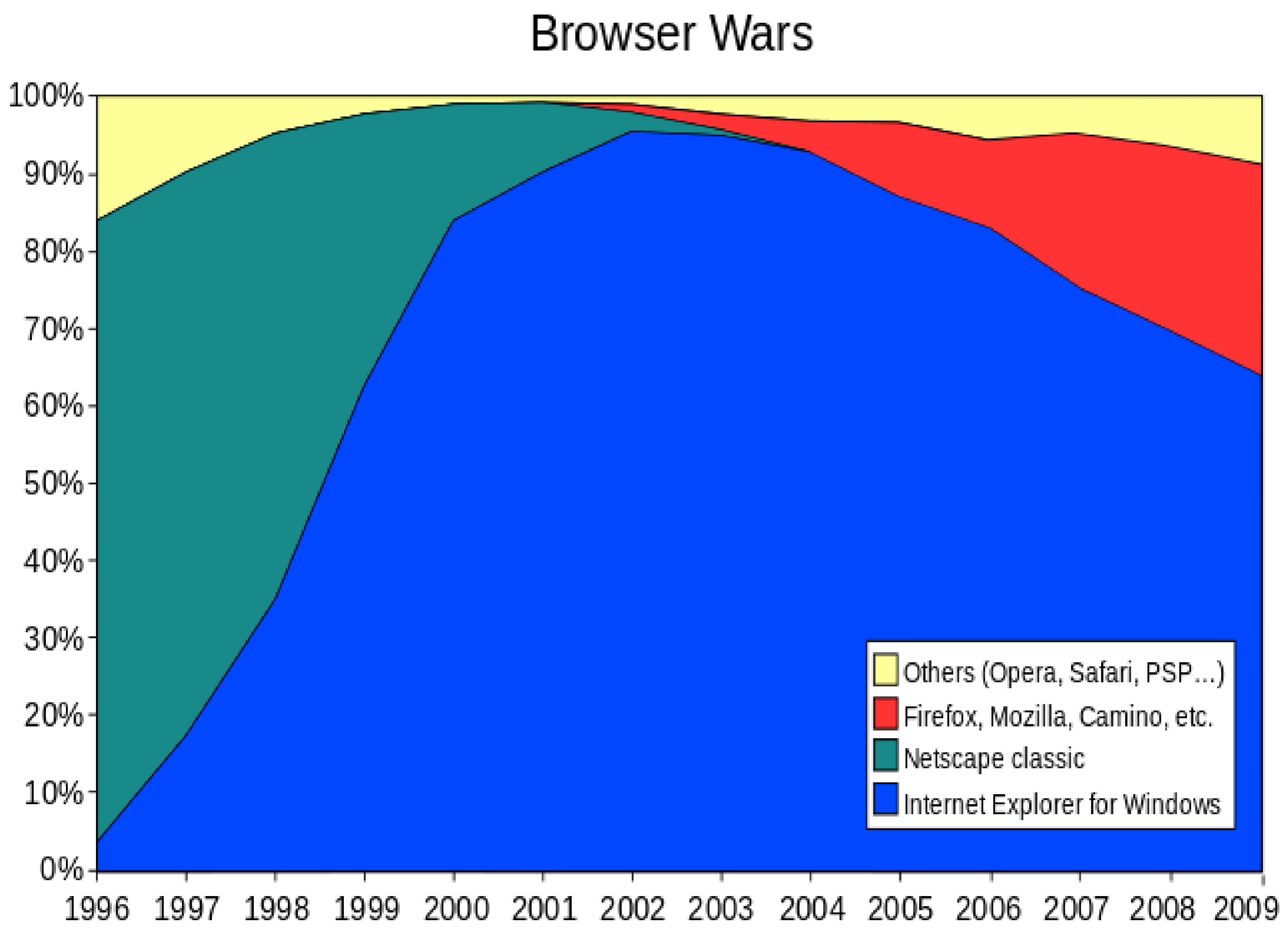
INTRODUCTION

HTTP

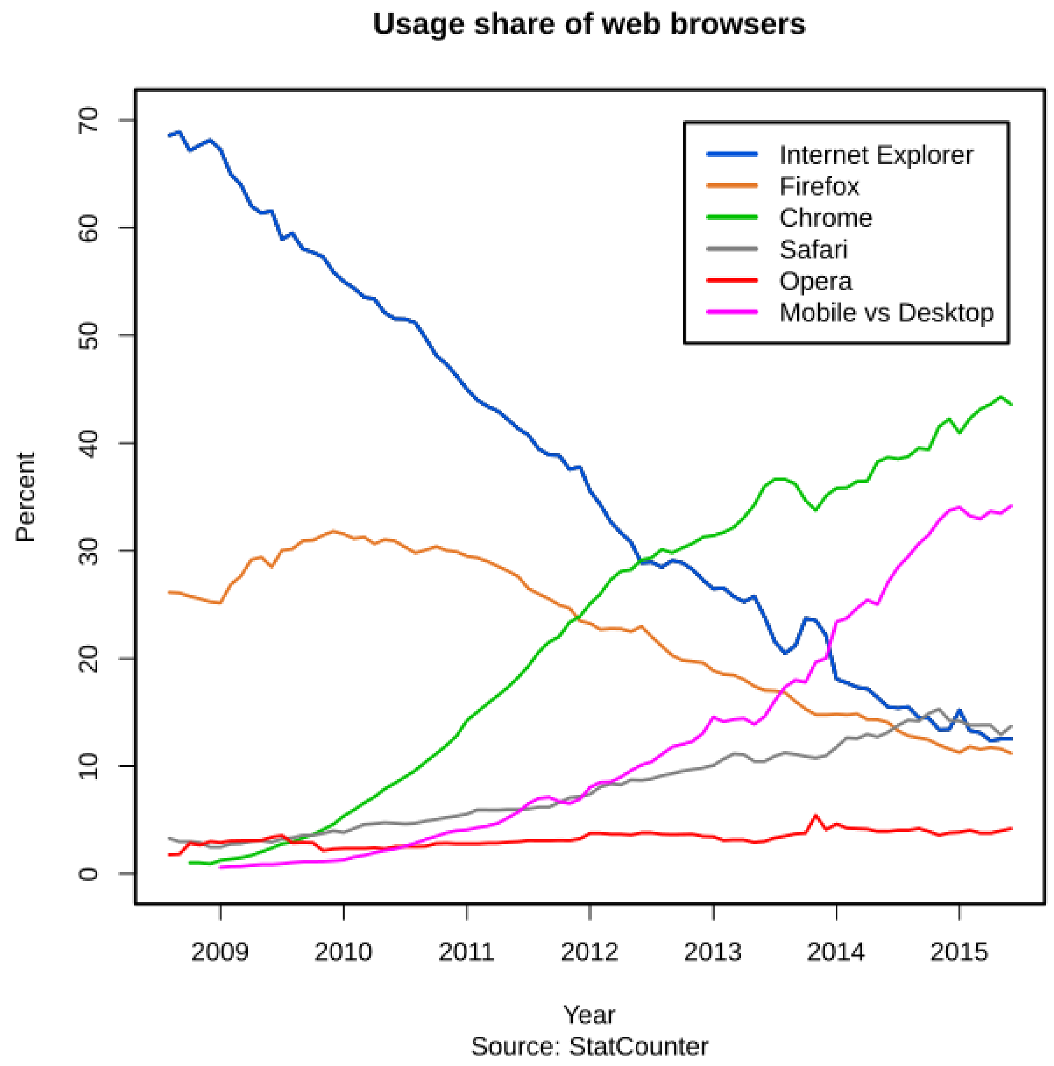
HTTP History

Year	Comment	
1962	ArpaNet	
as of 1981	TCP / IP as network protocol (→ Internet) Layer 7 Protocols: POP, SMTP, FTP, News etc.	
as of 1989	Development of HTTP at CERN Tim Berners-Lee	
1991	HTTP/0.9 Proprietary implementation Netscape, IE	
1996	HTTP/1.0 as RFC1945 der IETF	*First Browser War*
1999	HTTP/1.1 as RFC 2616 / 2617 der IETF	
2008		*Second Browser War*
2014	HTTP/1.1 as RFC 7230-7235	
ab 2012	SPDY as predecessor of HTTP/2.0	
ab 2015	HTTP/2.0 as RFC 7540	

First Browser War 1996-2009

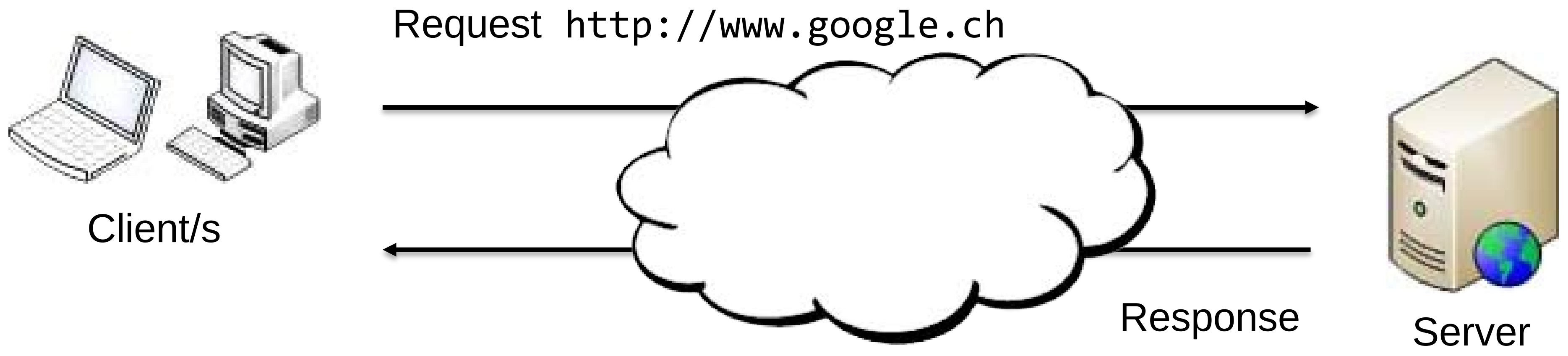


Second Browser War 2009-2015



DEMO INTRO

Client / Server Architecture



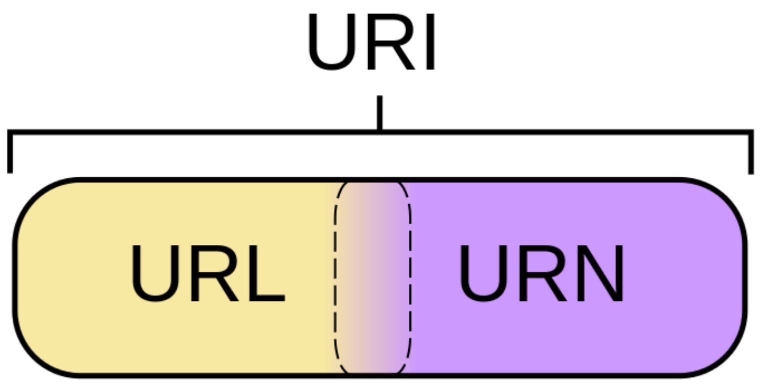
- **Client**
  - Workstation / PCs / …
  - Rely on server resources
- **Server**
  - (Powerful) Computers
  - Dedicated to mangage shared resources

HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL
FHO Fachhochschule Ostschweiz

URI SCHEMA

HTTP URI

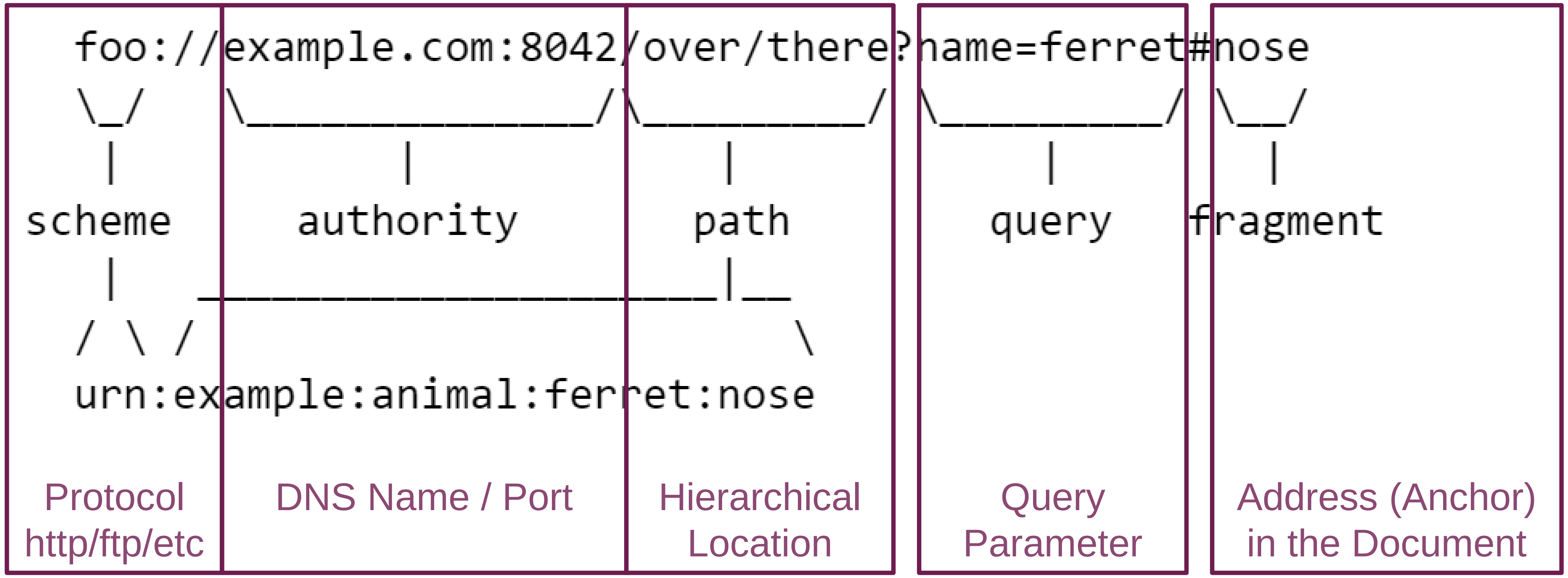
- **URI = Unified Resource Identifier (URI)**
  - is a string of characters used to identify a name of a resource
  - URL and URN are URIs
- **URL = Unified Resource Locator (URL)**
  - is a reference to a resource that specifies the location of the resource on a computer network and a mechanism for retrieving it
  - URLs may contain a URN
  - Limited schema range
- **URN = Unified Resource Name (URN)**
  - is a string of characters used to identify a name of a resource.
  - Independent from location / persistence / …
  - Limited to schema “URN”



URL Schema

- **<scheme name> : <hierarchical part> [? <query>] [# <fragment>]**

- **Example:**



```
  foo://example.com:8042/over/there?name=ferret#nose
  \_/   \______________/\_________/ \_________/ \__/
   |           |            |            |        |
scheme     authority       path        query   fragment
   |   _____________________|__
  / \ /                        \
  urn:example:animal:ferret:nose
```

URI Encoding

- **Special Characters must be encoded**
- **Important for control characters**

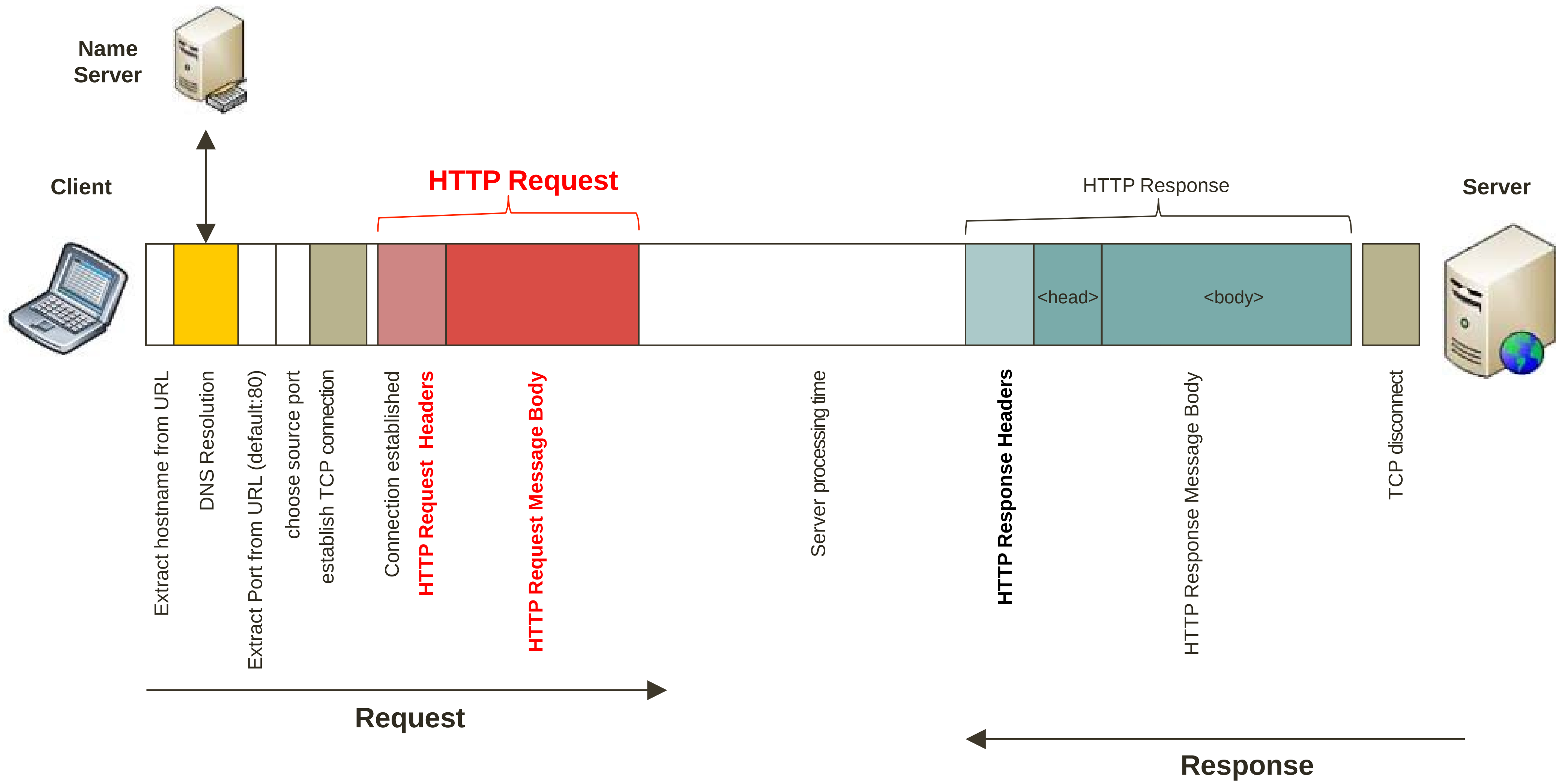
- **Example:**

'/' must be converted	%2F
' ' (space)	%20
'&'	%26
'#'	%23
'?'	%3F

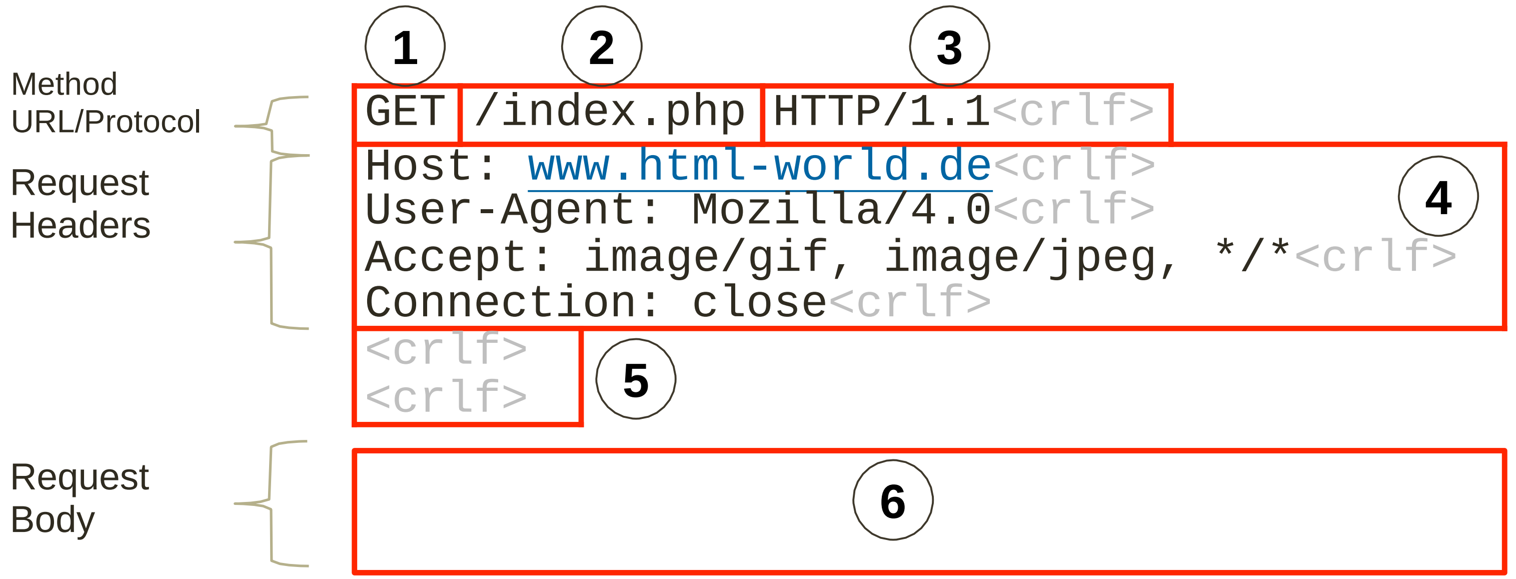
REQUEST

HTTP

Establish an HTTP Connection



HTTP Request



```
GET /index.php HTTP/1.1<crlf>
Host: www.html-world.de<crlf>
User-Agent: Mozilla/4.0<crlf>
Accept: image/gif, image/jpeg, */*<crlf>
Connection: close<crlf>
<crlf>
<crlf>
```

1. Method
2. Address (URL Path)
3. Protocol Version (HTTP/0.9 | HTTP/1.0 | HTTP/1.1. | HTTP/2.0)
4. Request Headers (optional, depends on the application)
5. Header/Body Separator (2 x crlf)
6. Request Body (used for example in <form> Data Transmitions)

HTTP Methods

Method	Description
GET	The GET method is used to **retrieve information** from the given server using a given URI. Requests using GET should only retrieve data and should have **no other effect on the data**.
HEAD	Same as GET, but transfers the **status line and header section** only.
POST	A POST request is used to **send data to the server**, for example, **new** customer information, *file upload*, etc. using *HTML forms*.
PUT	**Replaces all current representations** of the target resource with the uploaded content.
DELETE	**Removes all current representations** of the target resource given by a URI.
CONNECT	Establishes a tunnel to the server identified by a given URI.
OPTION	Describes the communication options for the target resource.
TRACE	Performs a message loop-back test along the path to the target resource.

HTTP GET vs POST Methods

- **GET data is placed into the URL (as query string)**

```
GET /search?query=abc HTTP/1.1<crlf>
Host: www.html-world.de<crlf>
User-Agent: Mozilla/4.0<crlf>
Accept: image/gif, image/jpeg, */*<crlf>
Connection: close<crlf>
<crlf>
<crlf>
```

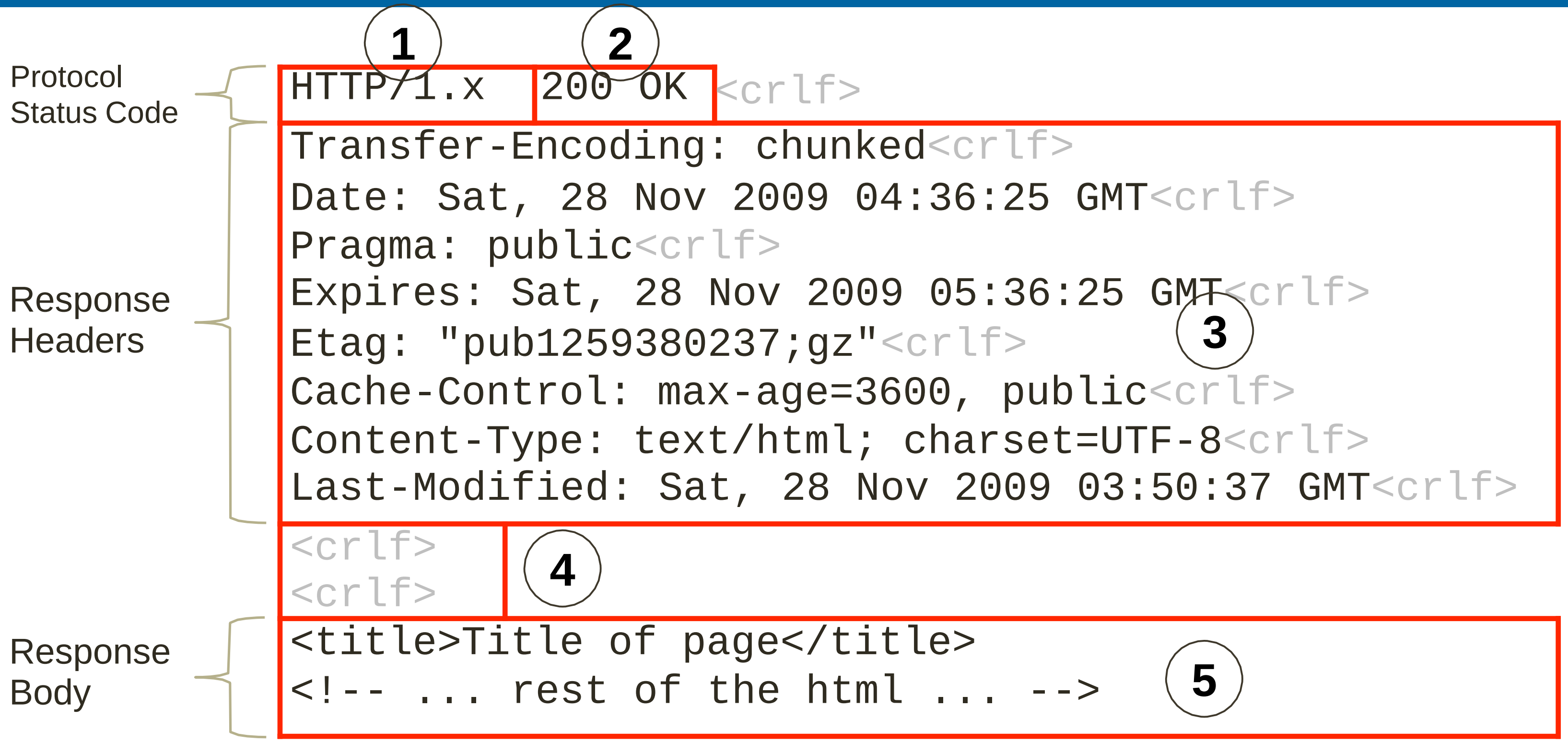
- **POST data is placed into the request body**

```
POST /search HTTP/1.1<crlf>
Host: www.html-world.de<crlf>
User-Agent: Mozilla/4.0<crlf>
Accept: image/gif, image/jpeg, */*<crlf>
Connection: close<crlf>
<crlf>
<crlf>
query=abc
```

RESPONSE

HTTP

HTTP Response



```
HTTP/1.x 200 OK<crlf>
Transfer-Encoding: chunked<crlf>
Date: Sat, 28 Nov 2009 04:36:25 GMT<crlf>
Pragma: public<crlf>
Expires: Sat, 28 Nov 2009 05:36:25 GMT<crlf>
Etag: "pub1259380237;gz"<crlf>
Cache-Control: max-age=3600, public<crlf>
Content-Type: text/html; charset=UTF-8<crlf>
Last-Modified: Sat, 28 Nov 2009 03:50:37 GMT<crlf>
<crlf>
<crlf>
<title>Title of page</title>
<!-- ... rest of the html ... -->
```

1. Protocol Version (HTTP/0.9 | HTTP/1.0 | HTTP/1.1. | HTTP/2.0)
2. Status Code
3. Response Headers (optional, depends on application)
4. Header/Body Separator (2 x crlf)
5. Response Body (The returning HTML Code)

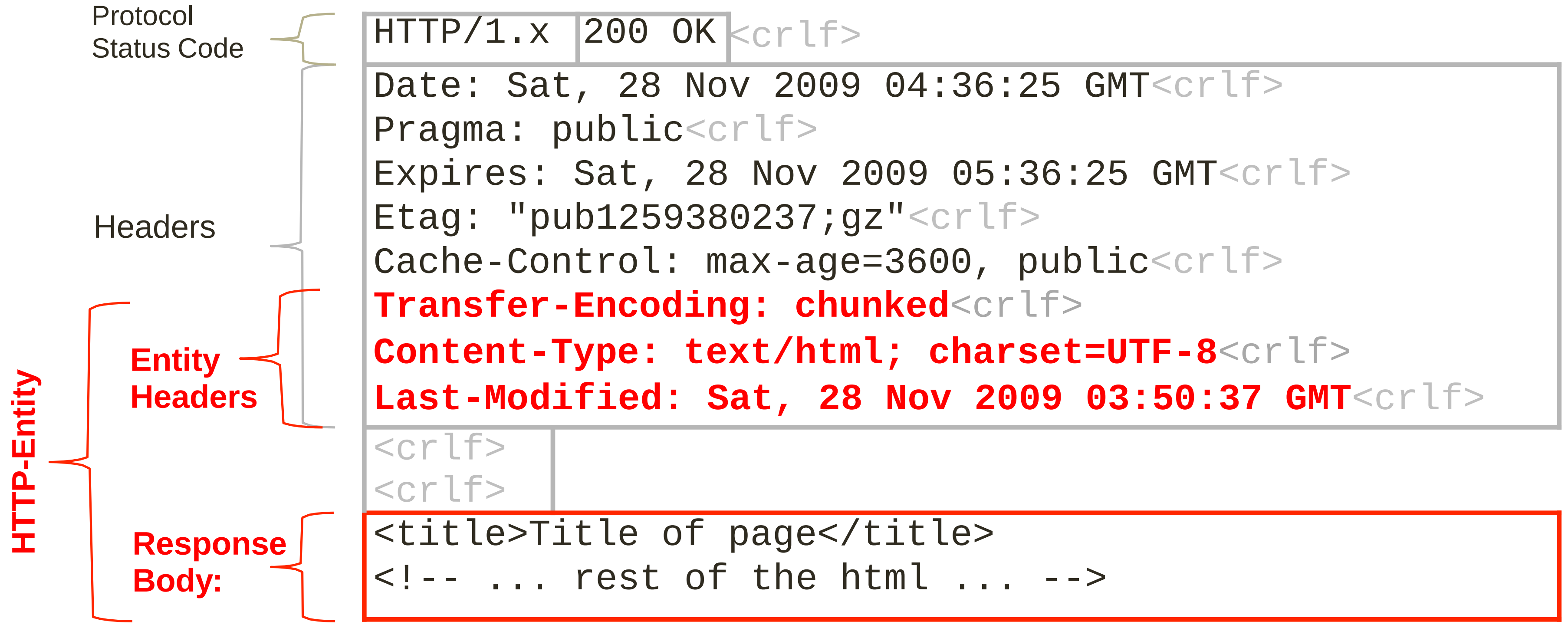
HTTP Response Status Codes

CODE	
1xx	Informational
2xx	Successful 200 OK 201 Created 204 No Content
3xx	Redirection 301 Moved Permanently
4xx	Client Error 400 Bad Request 401 Unauthorized 403 Forbidden 404 Not Found
5xx	Server Error 500 Internal Server Error 505 HTTP Version Not Supported
9xx	Non-Standard Codes

HEADERS

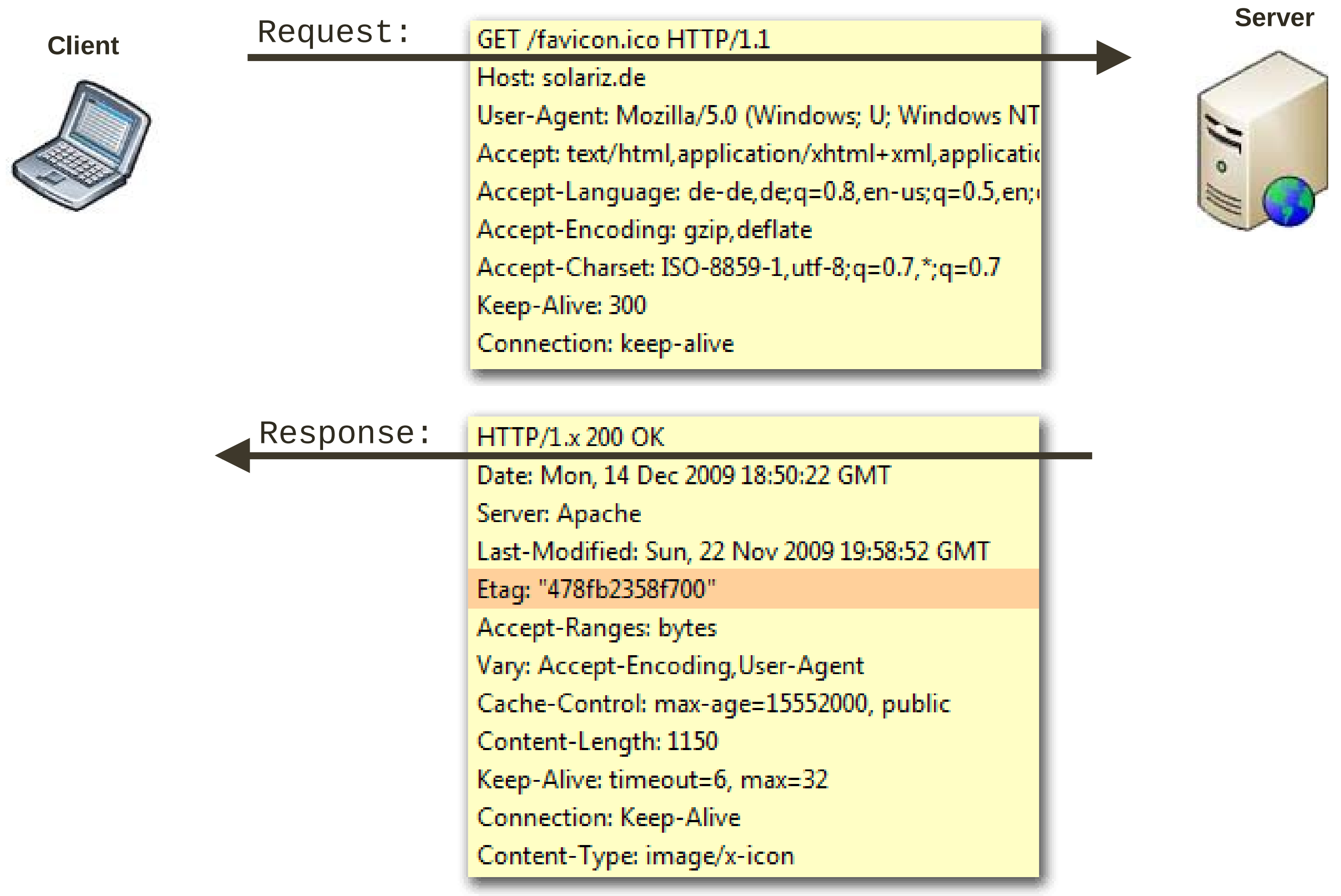
HTTP

Entity (Response)



```
HTTP/1.x 200 OK<crlf>
Date: Sat, 28 Nov 2009 04:36:25 GMT<crlf>
Pragma: public<crlf>
Expires: Sat, 28 Nov 2009 05:36:25 GMT<crlf>
Etag: "pub1259380237;gz"<crlf>
Cache-Control: max-age=3600, public<crlf>
Transfer-Encoding: chunked<crlf>
Content-Type: text/html; charset=UTF-8<crlf>
Last-Modified: Sat, 28 Nov 2009 03:50:37 GMT<crlf>
<crlf>
<crlf>
<title>Title of page</title>
<!-- ... rest of the html ... -->
```

Response handling with Etag 1/2



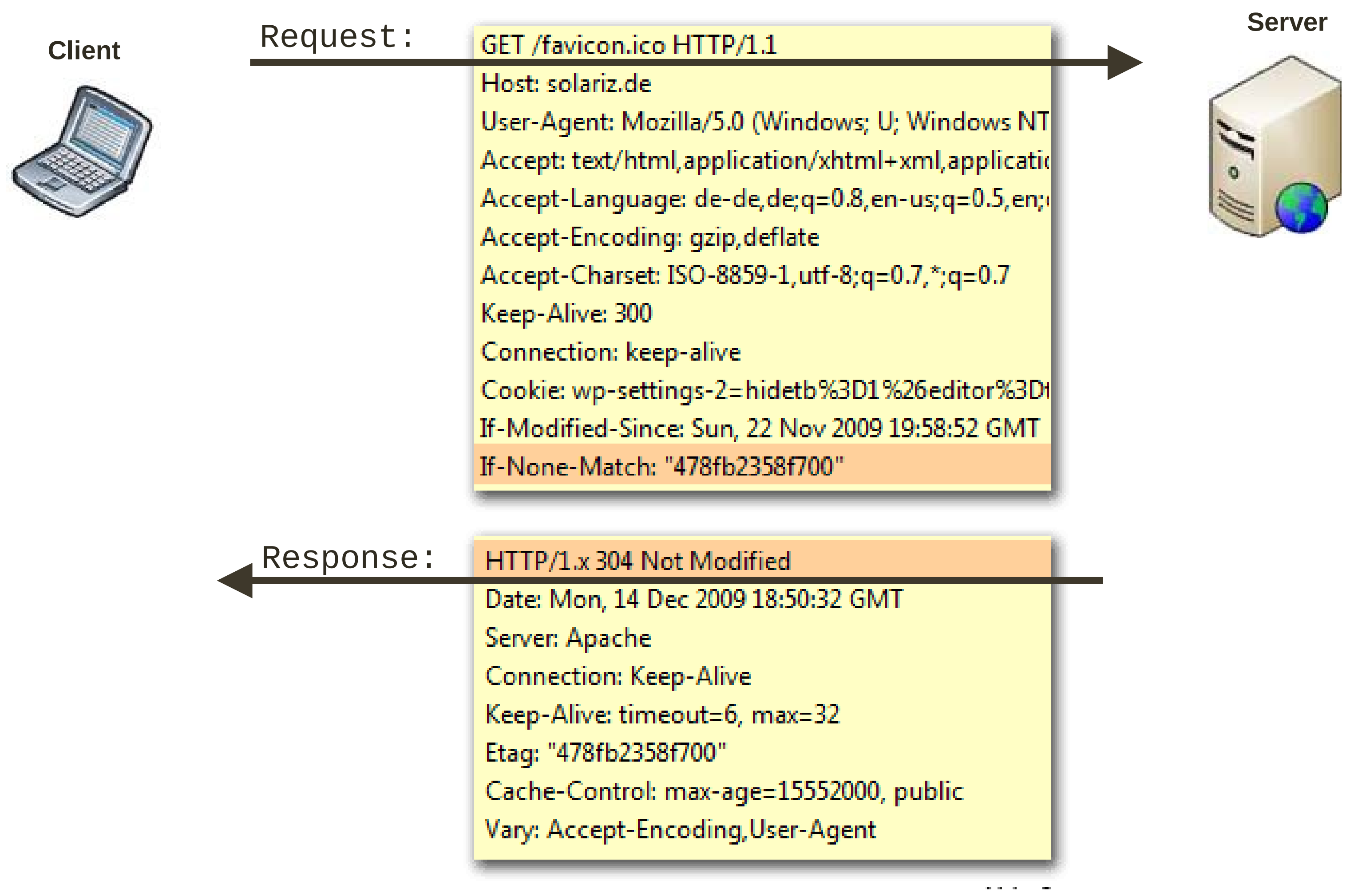
```
GET /favicon.ico HTTP/1.1
Host: solariz.de
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
Accept: text/html,application/xhtml+xml,applicati
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

```
HTTP/1.x 200 OK
Date: Mon, 14 Dec 2009 18:50:22 GMT
Server: Apache
Last-Modified: Sun, 22 Nov 2009 19:58:52 GMT
Etag: "478fb2358f700"
Accept-Ranges: bytes
Vary: Accept-Encoding,User-Agent
Cache-Control: max-age=15552000, public
Content-Length: 1150
Keep-Alive: timeout=6, max=32
Connection: Keep-Alive
Content-Type: image/x-icon
```

HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL

FHO Fachhochschule Ostschweiz

Response handling with Etag 2/2



```
GET /favicon.ico HTTP/1.1
Host: solariz.de
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
Accept: text/html,application/xhtml+xml,applicati
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: wp-settings-2=hidetb%3D1%26editor%3D
If-Modified-Since: Sun, 22 Nov 2009 19:58:52 GMT
If-None-Match: "478fb2358f700"
```

```
HTTP/1.x 304 Not Modified
Date: Mon, 14 Dec 2009 18:50:32 GMT
Server: Apache
Connection: Keep-Alive
Keep-Alive: timeout=6, max=32
Etag: "478fb2358f700"
Cache-Control: max-age=15552000, public
Vary: Accept-Encoding,User-Agent
```

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Headers Overview (Request & Response)

Request

Request Headers

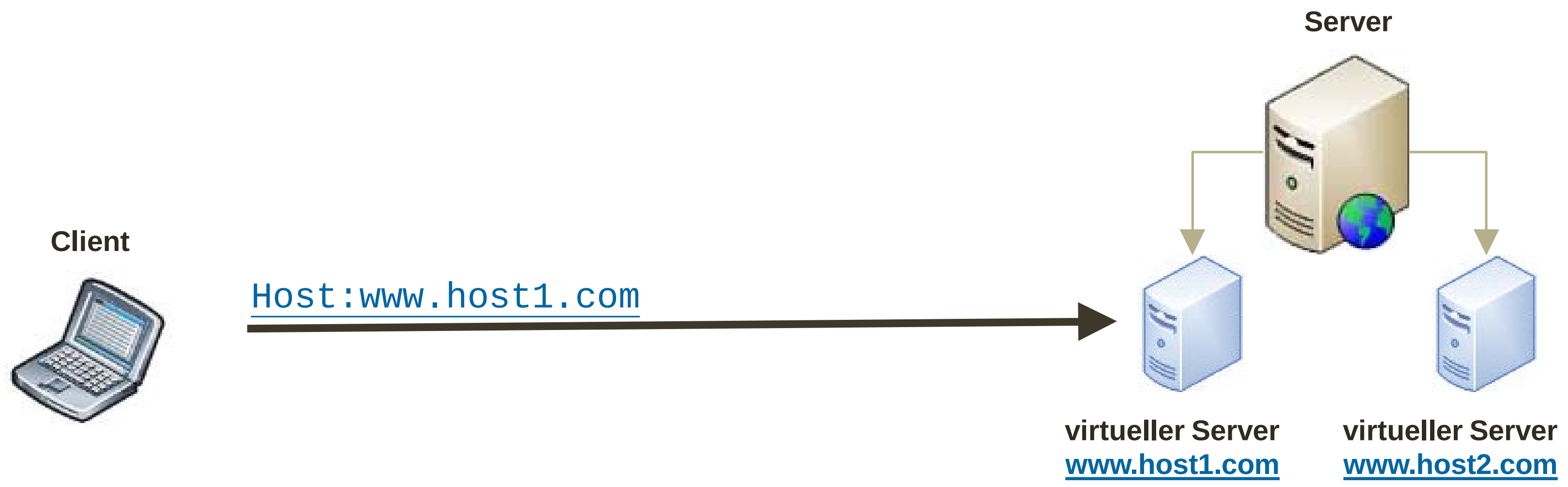
```
GET /index.php HTTP/1.1<crlf>
Host: www.html-world.de<crlf>
User-Agent: Mozilla/4.0<crlf>
Accept: image/gif, image/jpeg, */*<crlf>
Connection: close<crlf>
<crlf>
<crlf>
```

Response

Response Headers

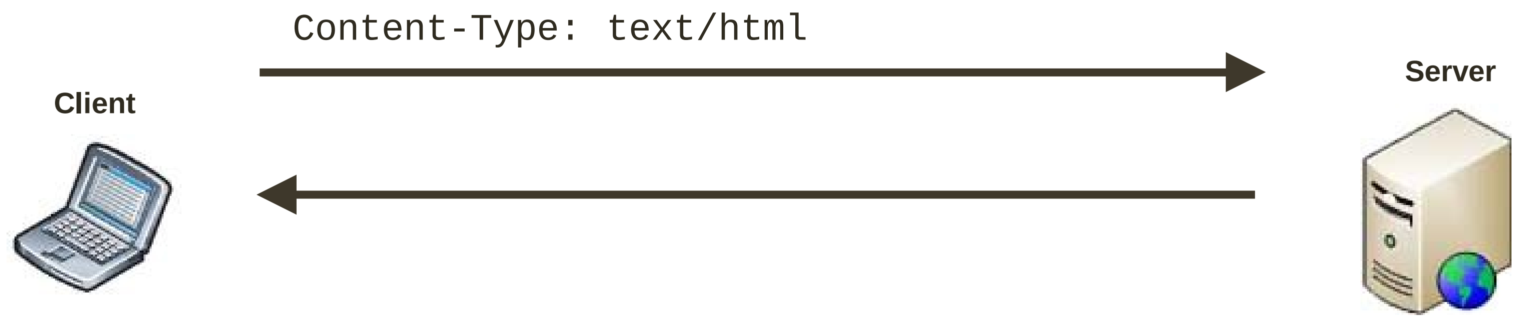
```
HTTP/1.x 200 OK<crlf>
Transfer-Encoding: chunked<crlf>
Date: Sat, 28 Nov 2009 04:36:25 GMT<crlf>
Pragma: public<crlf>
Expires: Sat, 28 Nov 2009 05:36:25 GMT<crlf>
Etag: "pub1259380237;gz"<crlf>
Cache-Control: max-age=3600, public<crlf>
Content-Type: text/html; charset=UTF-8<crlf>
Last-Modified: Sat, 28 Nov 2009 03:50:37 GMT<crlf>
<crlf>
<crlf>
<title>Title of page</title>
<!-- ... rest of the html ... --
>
```

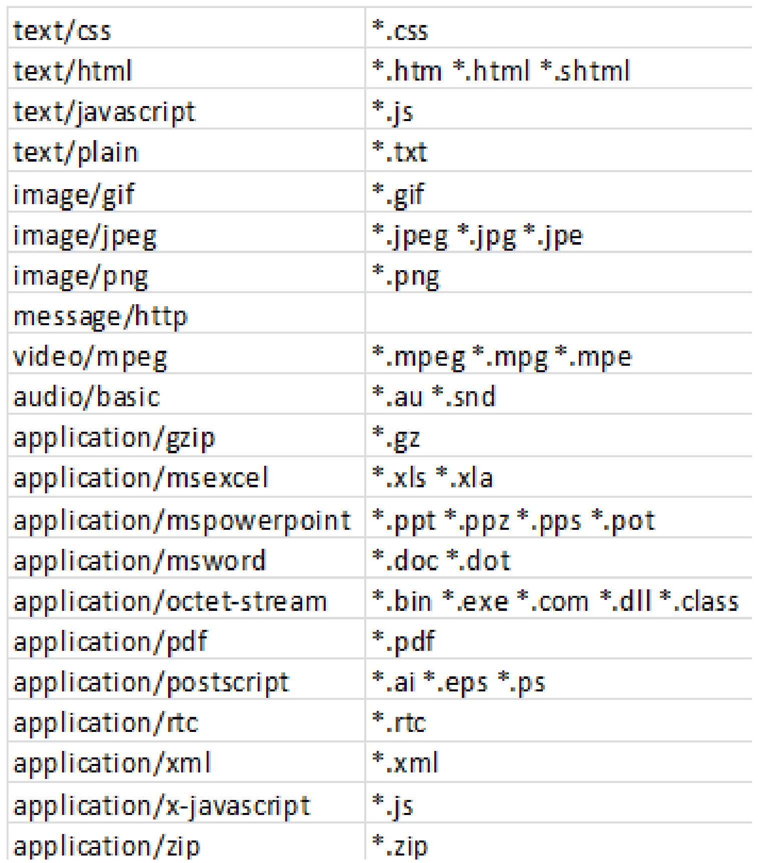
Host Request Header



- **The same IP address can be used for multiple hosts**
- **To determine the virtual server the host name is by-passed**

Content-Type Entity Header



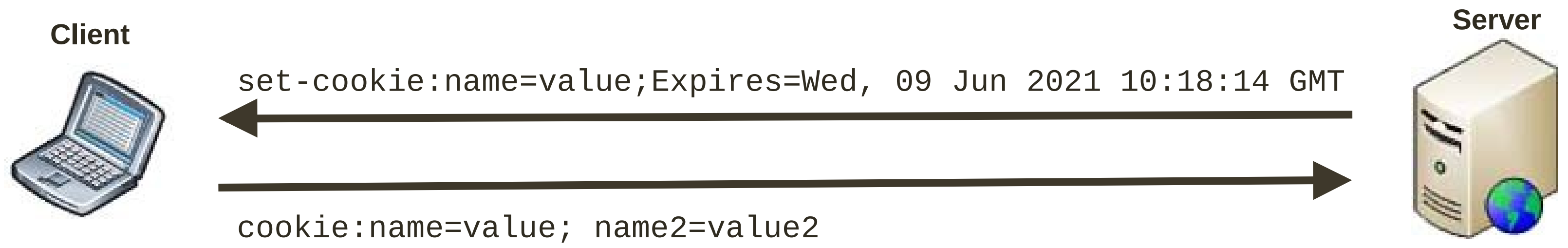


text/css	*.css
text/html	*.htm *.html *.shtml
text/javascript	*.js
text/plain	*.txt
image/gif	*.gif
image/jpeg	*.jpeg *.jpg *.jpe
image/png	*.png
message/http	
video/mpeg	*.mpeg *.mpg *.mpe
audio/basic	*.au *.snd
application/gzip	*.gz
application/msexcel	*.xls *.xla
application/mspowerpoint	*.ppt *.ppz *.pps *.pot
application/msword	*.doc *.dot
application/octet-stream	*.bin *.exe *.com *.dll *.class
application/pdf	*.pdf
application/postscript	*.ai *.eps *.ps
application/rtc	*.rtc
application/xml	*.xml
application/x-javascript	*.js
application/zip	*.zip

- **Declares the parsing / interpreting format for the Client/Server**
- **The Content-Type is declared in MIME-Type format**
  - Specifies the media type (text, video, audio, application, …) and a subtype (html)

HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL
FHO Fachhochschule Ostschweiz

Cookie Request/Response Headers


Client
Server
set-cookie:name=value;Expires=Wed, 09 Jun 2021 10:18:14 GMT
cookie:name=value; name2=value2

- **A cookie represents a small piece of data**

- **Server writes the Cookie to the Client (`set-cookie:…`)**
- **Client transmits all Cookies for the current site back to the Server (`cookie:…`)**

SESSION MANAGEMENT

Introduction

- **Invented by Netscape as HTTP Header**
  - Originally a non-standard header
  - Today standardized in RFC 6265
- **Cookie is a small data unit stored on the Client and transmitted to Server on every Request**
  - Max 4096 Bytes
  - Max 50 Cookies per domain (varies by browser)
  - Max 3000 Cookies overall (varies by browser)
- **Cookie Expiration Time can be declared**
  - If a Cookie has no expiration, the Cookie is valid until the browser gets closed
  - Memory only Cookies are treaded as **Session**
- **Cookie can be declared as HTTPOnly to use the cookie only for HTTP / HTTPS requests**

Scope

- **Domain and Path declare the Scope of the Cookie**

- **Domain attribute**
  - **Cookie/Session is also present for subdomains**
  - **If no domain specified, the current domain is set**
- **Path attribute**
  - **Defines the scope inside a domain**
  - **The cookie is valid for the given path (and paths below)**

- **Example**
  - sessionId=AYQEVnDKrdst; Domain=.foo.com; Path=/access; HttpOnly

QUESTIONS?

Sources

- **Slides**
  - Markus Wirrer, Namics
  - http://en.wikipedia.org/