

Lets listen to the W3C

# HTTP: the language of Web communication

TI1506: Web and Database Technology

Claudia Hauff

[ti1506-ewi@tudelft.nl](mailto:ti1506-ewi@tudelft.nl)

Lecture 1 [Web], 2014/15

# Course overview [Web]

## **1. http: the language of Web communication**

2. HTML & Web app design

3. JavaScript: interactions in the browser

4. node.js: JavaScript on the server


5. CSS: Lets make the app pretty

6. Ajax: asynchronous JavaScript

7. Personalization: Cookies & sessions

8. Securing your application

# At the end of this lecture, you should be able to ...

- **Describe** how Web servers and clients interact with each other (via TCP/IP and HTTP)
- **Write** HTTP messages that request Web resources from Web servers and understand the responses
- **Describe** the different components of URLs and their purpose
- ~~**Understand** and **employ** basic HTTP authentication~~ 
- ~~**Explain** the difference between HTTP and HTTPS~~

# World Wide Web vs. Internet

# The Web: a brief history

**World Wide Web:** a global system of interconnected hypertext documents available via the Internet  
(envisioned already in 1945)

- **1960s:** Precursor to the Internet (ARPANET) devised by the US department of Defense
  - Initial services: electronic mail, file transfer
- Late **1980s:** Internet opened to commercial interests
- **1989:** WWW created by Tim Berners-Lee (CERN)
- **1994:** Netscape released its first Web browser
- **1995:** Microsoft released Internet Explorer v1
- **1998:** Google was founded
- **2002:** Mozilla released Firefox v1

# Key aspects of the Internet

**Internet:** interconnected computer networks (sub-networks) that span the globe; communicating through a common standard (TCP/IP)

- Sub-networks function **autonomously**
- **No centralised** (global) control instance
- Devices **dynamically** join and leave the network
- Devices interact through **agreed-upon open standards**; anyone can create a new device
- **Easy** to use: server/client software is widely available

# Two important organisations

- Internet Engineering Task Force (IETF)

“The mission of the IETF is to make the Internet work better by producing high quality, relevant technical documents that influence the way people design, use, and manage the Internet. “

Request for Comments (RFC)

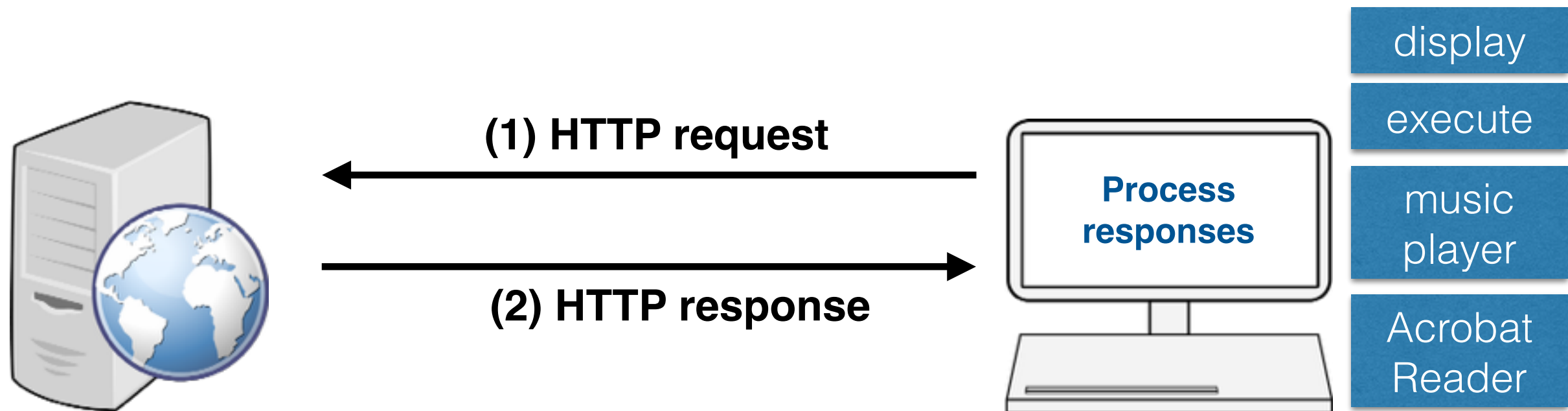
- World Wide Web Consortium (W3C)

“The W3C mission is to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure the long-term growth of the Web.”



# An introduction to HTTP messages

# Web servers and clients



- Servers wait for data requests
- Answer thousands of clients simultaneously
- Host **web resources**
- Clients are most often Web browsers
- Telnet

**Web resource:** any kind of content with an identity, including static files (e.g. text, images, video), software programs, Web cam gateway, etc.

# Network communication

- Conceptual model Open Systems Interconnection (OSI)
- Network protocols are matched to different “layers”
- Many network protocols exist, three are of interest to us:

**IP:** Internet Protocol

**TCP:** Transmission Control Protocol

**HTTP:** Hypertext Transfer Protocol

others include: SSH, IMAP, SMTP, FTP ....

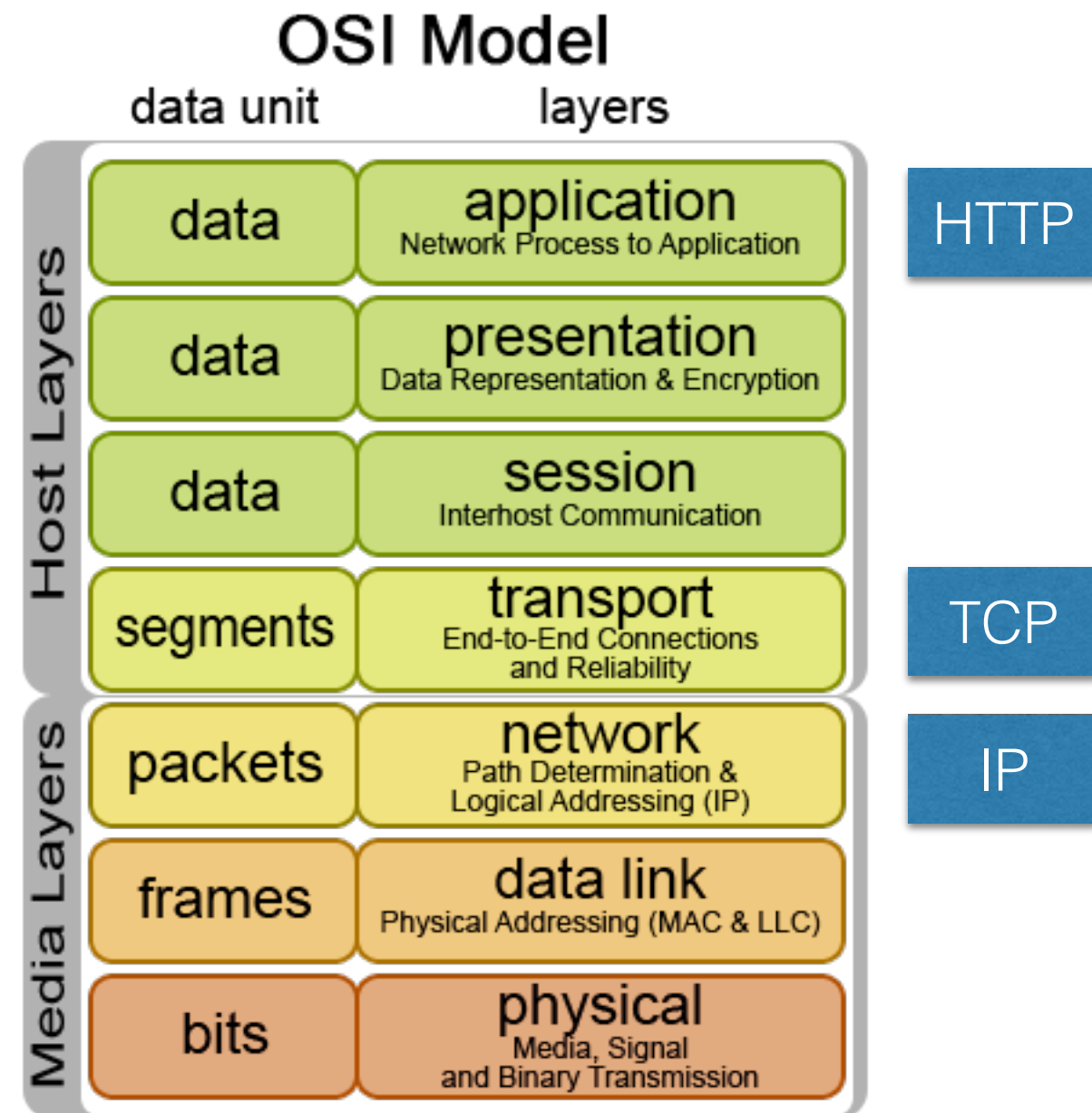


Image src: Wikipedia

HTTP uses **reliable** data-transmission protocols.

# Demo: a look at tudelft.nl

Modern web pages consist of many resources.

A cascade of HTTP transactions is required.

Different MIME types are typically found.

Firebug: essential tool for Web development.

Load tudelft.nl website and open Firebug.  
HTTP requests/responses and cookies can be viewed  
(among others).

Try a non-existing url as well to see the difference!





# HTTP request message

plain text, line-oriented character sequences

GET / HTTP/1.1

Host: www.tudelft.nl

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:31.0) Gecko/20100101 Firefox/31.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: en-gb,en;q=0.5

Accept-Encoding: gzip, deflate

DNT: 1

Cookie:

\_\_utma=1.20923577936111.16111.19805.2;utmcmd=(none);

**Question:** what is the main problem of the plain text format compared to a binary format?

# HTTP response message

HTTP/1.1 200 OK

start line

Date: Fri, 01 Aug 2014 13:35:55 GMT

Content-Type: text/html; charset=utf-8

Content-Length: 5994

Connection: keep-alive

Set-Cookie: fe\_typo\_user=d5e20a55a4a92e0;  
path=/; domain=tudelft.nl

[...]

Server: TU Delft Web Server

header fields

name:value

body  
(optional)

...

...

# HTTP headers dissected

# Primary entity header fields

<b>Content-Type</b>	Entity type
<b>Content-Length</b>	Length/size of the message
Content-Language	Language of the entity sent (e.g. English)
<b>Content-Encoding</b>	Data transformations applied to the entity
Content-Location	Alternative location of the entity
Content-Range	For partial entities, range defines the pieces sent
<b>Content-MD5</b>	Checksum of the content
<b>Last-Modified</b>	Date on which this entity was created/modified
<b>Expires</b>	Date at which the entity will become stale
Allow	Lists the legal request methods for the entity

**Important:** Entity bodies only contain raw data, header information required to interpret the data.



# Content-Type

- **MIME** types are attached to all HTTP object data

Multipurpose Internet **Mail** Extensions

historic reasons

- MIME type determines clients reaction to the received data

- Pattern: [primary object type]/[subtype]

e.g. text/plain, text/html, image/jpeg,  
video/quicktime, application/vnd.ms-  
powerpoint

# Content types are diverse

Most popular
text/html
image/jpeg
text/xml
application/rss+xml
text/plain
application/xml
text/calendar
application/pdf
application/atom+xml
unknown/unknown

Least popular
application/pgp-keys
application/x-httpd-php4
chemical/x-pdb
model/mesh
application/x-perl
audio/x_mpegurl
application/bib
application/postscript
application/x-msdos-program

# Content-Length

- Indicates the **size** of the entity body in the message
- Necessary to detect premature message truncation (e.g. due to a server crash, faulty proxy)
- Essential for **persistent connections** to discover where one HTTP message ends and the next one begins

**Persistent connections** reuse the same TCP connection for multiple HTTP request/response messages.

# Content-Encoding

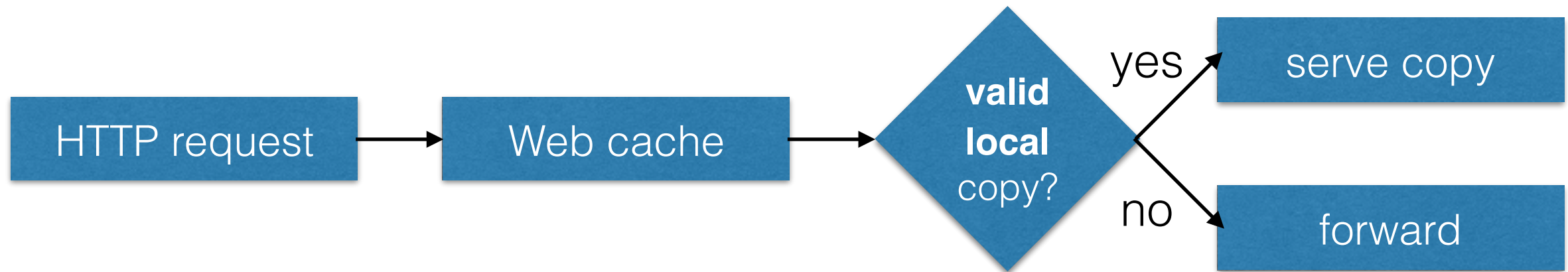
- Commonly either gzip, compress (Unix compression), deflate (zlib compression) or identity (no encoding)
- Servers aim to use **encodings** that clients understand
  - Clients send a list of acceptable encodings in the `Accept-Encoding` request header
- **Compression** saves network bandwidth (fewer bits to transmit) but increases processing costs to decompress

# Content-MD5

- HTTP messages are sent via TCP/IP (ensuring reliable transport)
- **But:** the Internet is huge, many servers interact to transport a message with different implementations of established protocols (which may be buggy)
- Sanity check: Sender generates a checksum (MD5) of the content to detect unintended modifications of the content

# Expires

- **Web caches** keep copies of *popular* documents



- Advantages of Web caches
  - A. Reduction of redundant data transfer
  - B. Reduction of network bottlenecks
  - C. Reduced demand on origin servers
  - D. Reduced distance delay
- **Expires** indicates when fetched resource is no longer valid and needs to be retrieved from the origin server

# Expires & Cache-Control

- Content on the origin server can change
- Caches need to ensure that their copies are **in sync** with the origin server
- Caches can revalidate their copies at any time with any frequency (**but**: this would not be efficient)
- **Expires** in HTTP response header indicates a **document's expiration date** in **absolute** terms — determines how long the content is fresh; cache revalidates at that date
- Alternative to **Expires** is **Cache-Control**: indicates a document's expiration date in **relative** terms (number of seconds since being sent)

# Last-Modified

- Contains the date when the document was last **altered** (in HTTP response)
- No indication about the amount of changes in the document
- Often used in combination with `If-Modified-Since` for cache revalidation requests — origin server only returns the document if it changed since the given date (returned is `304 Not Modified` response)
- Last-Modified dates are **not always reliable** (e.g. manipulated by origin servers to ensure high cache revalidation rates)



# Remember: HTTP response message

```
HTTP/1.1 200 OK
```

start line

```
Date: Fri, 01 Aug 2014 13:35:55 GMT
```

```
Content-Type: text/html; charset=utf-8
```

```
Content-Length: 5994
```

```
Connection: keep-alive
```

```
Set-Cookie: fe_typo_user=d5e20a55a4a92e0;  
path=/; domain=tudelft.nl
```

```
[...]
```

```
Server: TU Delft Web Server
```

header fields

name:value

```
....
```

```
....
```

body  
(optional)

# Common status codes

<b>1xx</b>	Informational
<b>2xx</b>	Success (200 OK)
<b>3xx</b>	Redirected
<b>4xx</b>	Client error (404 Not Found)
<b>5xx</b>	Server error

In practice only a few codes per category are supported

[A more detailed overview.](#)

# HTTP methods

# Common HTTP methods

<b>GET</b>	Get a document from the Web server.
<b>HEAD</b>	Get the header of a document from the Web server.
<b>POST</b>	Send data from the client to the server for processing.
<b>PUT</b>	Save the body of the request on the server.
<b>TRACE</b>	Trace the message through proxy servers to the server.
<b>OPTIONS</b>	Determine what methods can operate on a server.
<b>DELETE</b>	Remove a document from a Web server.

**Question:** in what circumstances might HEAD be useful?

# Demo: telnet

Telnet opens a **TCP connection** to a Web server; chars are typed directly into the port. The server treats telnet as Web client, the returned data is displayed onscreen.

A number of HTTP requests may be required to end up at the final (wanted) page.

Often Web servers treat Web-browser requests differently from machine-generated requests.

Open a terminal and use telnet.  
Run through a few GET/HEAD examples.

PUT/POST/DELETE will be practiced in the assignment.

# TCP & HTTP: displaying a simple HTML resource in the browser

**domain name**                      **path**

`http://www.microsoft.com:80/index.html`

**port**

- (1) Browser extract's domain name from URL
- (2) Browser converts domain name into IP address
- (3) Browser extracts port number (default: 80)
- (4) Browser establishes a TCP connection with the Web server
- (5) Browser sends an HTTP request
- (6) Web server sends an HTTP response
- (7) The connection is closed, browser displays the document

**HTTP is stateless. Each request/response occurs in isolation.**

# TCP & HTTP: displaying a simple HTML resource in the browser

TCP delivers

- error-free data transportation
- in-order delivery
- an unsegmented data stream

`http://www.microsoft.com:80/index.html`



`http://65.55.57.27:80/index.html`

32-bit IP address: four 8-bit numbers (0-255)

**domain name**

**port**

**path**

**IP address**

# Uniform Resource Locators (URLs)



# Question: which of the following URLs are valid?

A. `mailto:c.hauff@tudelft.nl`

C. `ftp://anonymous:mypass@ftp.csx.cam.ac.uk/  
gnu;date=today`

E. `http://www.bing.com/?scope=images&nr=1#top`

G. `https://duckduckgo.com/html?q=delft`

I. `http://myshop.nl/comp;typ=c/  
apple;class=a;date=today/index.html;fr=delft`

K. `http://правительство.рф`

**All are syntactically valid!**

# URL syntax

- Uniform resource locators offer a standardised way to point to any resource on the Internet
- Not restricted to the `http` scheme, syntax slightly varies from scheme to scheme
- General format (adhered to by most schemes):

**<scheme>**://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>

determines the protocol to use when connecting to the server.

# URL syntax

- Uniform resource locators offer a standardised way to point to any resource on the Internet
- Not restricted to the `http` scheme, syntax slightly varies from scheme to scheme
- General format (adhered to by most schemes):

`<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>`

the username/password (may be necessary to access a resource)

# URL syntax

- Uniform resource locators offer a standardised way to point to any resource on the Internet
- Not restricted to the `http` scheme, syntax slightly varies from scheme to scheme
- General format (adhered to by most schemes):

`<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>`

domain name (host name) or numeric IP address of the server

# URL syntax

- Uniform resource locators offer a standardised way to point to any resource on the Internet
- Not restricted to the `http` scheme, syntax slightly varies from scheme to scheme
- General format (adhered to by most schemes):

`<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>`

the port on which the server is expecting requests for the resource

# URL syntax

- Uniform resource locators offer a standardised way to point to any resource on the Internet
- Not restricted to the `http` scheme, syntax slightly varies from scheme to scheme
- General format (adhered to by most schemes):

`<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>`

the local path to the resource

# URL syntax

- Uniform resource locators offer a standardised way to point to any resource on the Internet
- Not restricted to the `http` scheme, syntax slightly varies from scheme to scheme
- General format (adhered to by most schemes):

`<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>`

Additional input parameters applications may require to access a resource on the server correctly. Can be set per path segment.



# URL syntax

- Uniform resource locators offer a standardised way to point to any resource on the Internet
- Not restricted to the `http` scheme, syntax slightly varies from scheme to scheme
- General format (adhered to by most schemes):

`<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>`

Parameters passed to gateway resources, i.e. applications [identified by the path] such as search engines.



# URL syntax

- Uniform resource locators offer a standardised way to point to any resource on the Internet
- Not restricted to the `http` scheme, syntax slightly varies from scheme to scheme
- General format (adhered to by most schemes):

`<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>`

The name of a piece of a resource. Only used by the client - the fragment is not transmitted to the server.

# URL syntax: query

(most important to us)

**<scheme>**://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>

**https** : // **duckduckgo.com** / **html** ? **q=delft**

- Query component is passed to the application accessed at the Web server (“gateway resource”)
- Necessary to enable interactive applications (like our ToDo Web application!)
- Common convention: `name1=value1&name2=value2&...`

# Schemes: more than just http

```
http://<host>:<port>/<path>?<query>#<frag>
```

**https** is analogous to **http** with **https** as scheme name (end-to-end encryption of HTTP connections)

```
mailto:<valid-email-address>
```

```
ftp://<user>:<password>@<host>:<port>/<path>;<params>
```

```
file://<host>/<path>, e.g.
```

```
file:///Users/claudiahauff/tmp.html
```

# Relative vs. absolute URLs

## base url

absolute

```
http://www.st.ewi.tudelft.nl/~hauff/new/index.html
```

```
<h1>Visualizations</h1>
```

```
<ol>
```

```
  <li><a href="vis/trecvis.html">TREC</a></li>
```

```
  <li><a href=" ../airsvis.html">AIRS</a></li>
```

```
</ol>
```

relative



Not trivial: URLs can be complex,  
RFC 3986 governs conversion

```
http://www.st.ewi.tudelft.nl/~hauff/new/vis/trecvis.html
```

```
http://www.st.ewi.tudelft.nl/~hauff/airsvis.html
```

**Question:** what is the main advantage of relative URLs?

# Pointer: <http://httpbin.org/>

- A **HTTP request & response service**
- A very useful site to play around with the different features of the http protocol
- You will use it in Assignment 1

# Summary

# HTTP ensures that content

- A. can be correctly identified.
- B. can be unpacked properly.
- C. is fresh.
- D. meets the user's needs.
- E. arrives complete and untampered with.

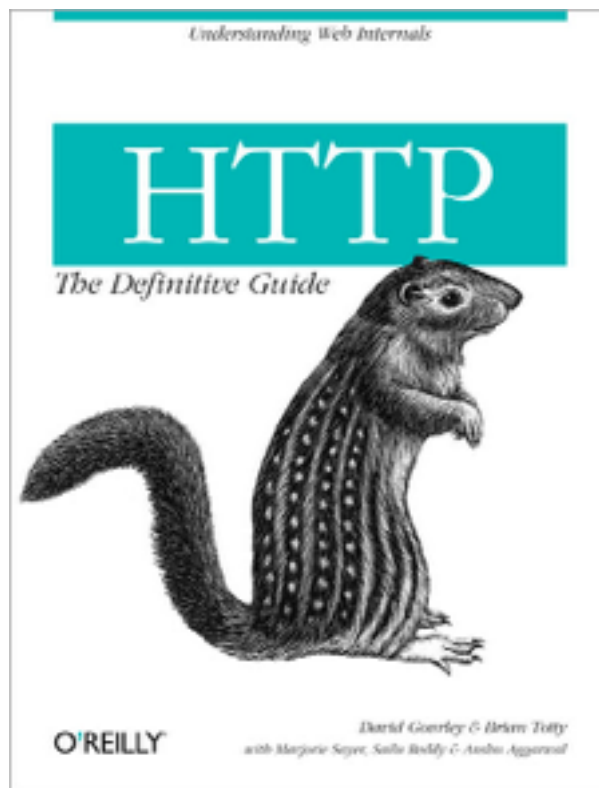
# Today we covered

- World Wide Web vs. the Internet
- Introduction to HTTP messages
- HTTP messages dissected
- Uniform resource locators



# Readings

- **No required reading** (the lecture is self-contained)
- Recommended: *HTTP: The Definitive Guide* (O'REILLY, 2002)



Chapters 1, 2, 3 and 12  
(the lecture material is largely based  
on those chapters)

**End of Lecture**