D3 World Wide Web (WWW)

Das WWW (World Wide Web) besteht aus WWW Servern von denen die WWW Browser (Clients) Objekte abrufen. Diese Objekte sind Daten jeglicher Art, insbesondere aber Hypertext- bzw. Hypermedia- Dokumente. Hypertext ist ein Text, der Links (verweise) auf andere Textstellen oder andere Dokumente enthält. Hypermedia enthält zusätzlich noch multimediale Anteile wie Grafik, Bilder (Bewegt- und Festbilder) sowie Audioinformation.

Die Hypermedia- Dokumente werden mittels HTML (Hypertext Markup Language) beschrieben. Ihr Lagerort auf einem Server wird durch eine URL (Universal Resource Locator) bestimmt. Die Dokumente werden mit dem HTTP (Hypertext Transfer Protocol) vom Server zum Client übertragen.

Es lassen sich drei Arten von Web-Dokumenten unterscheiden:

- Statische Dokumente: Diese werden bei der Erstellung vollständig beschrieben und auf dem Server abgelegt.
- Dynamische Dokumente: Sie werden vom Server erstellt, nachdem der Client sie angefordert hat. Sie werden durch eine Anwendungsprogramm erstellt und können je nach Anfrage unterschiedliche Inhalte haben.
  Beispiel: CGI (Common Gateway Interface) Dokument.
- Aktive Dokumente: Das Dokument enthält ein Programm, das auf dem Client ausgeführt wird.
  Beispiel: JavaScript oder Java-Applet.

URL, URI, URN

Eine URL (Universal Resource Locator) kennzeichnet den Lagerort eines Web-Dokuments. Der Aufbau einer URL ist:

```
<Schema>://<Servername>[:<Port>][/<Pfad des Dokuments>]
[;<Parameter>][#<Textmarke>][?<Anfrage>]
```

Dies ist ein Netzpfad mit einer absoluten URL.
Mögliche Schemata sind u.a.:

```
http
ftp
news
maillto
```

Man kann auch relative URL angeben. Diese beziehen sich dann jeweils auf den Standort des aktuellen Dokuments in dem der Pfad auftaucht.

```
[<Rel.Pfad des Dokuments>][;<Parameter>][#<Textmarke>]
[?<Anfrage>]
```

Um Dokumente auch dann wieder zu finden, wenn sie auf einen anderen Server gebracht werden, gibt es die URN (Universal Resource Names). Sie geben einen global eindeutigen, langlebigen logischen Namen für ein Dokument an. Der aktuelle Lagerort muss in einem Directory gespeichert sein und von dort erfragt werden. URI (Universal Resource Identifier) ist ein Obertbegriff für URL und URN.

HTTP

Das HTTP (Hypertext Transfer Protocol) ist ein Protokoll der Anwendungsschicht zur Übertragung einer Anforderung (Request) von Client zum Server und der Antwort (Response) des Servers zum Client. HTTP setzt auf TCP auf.

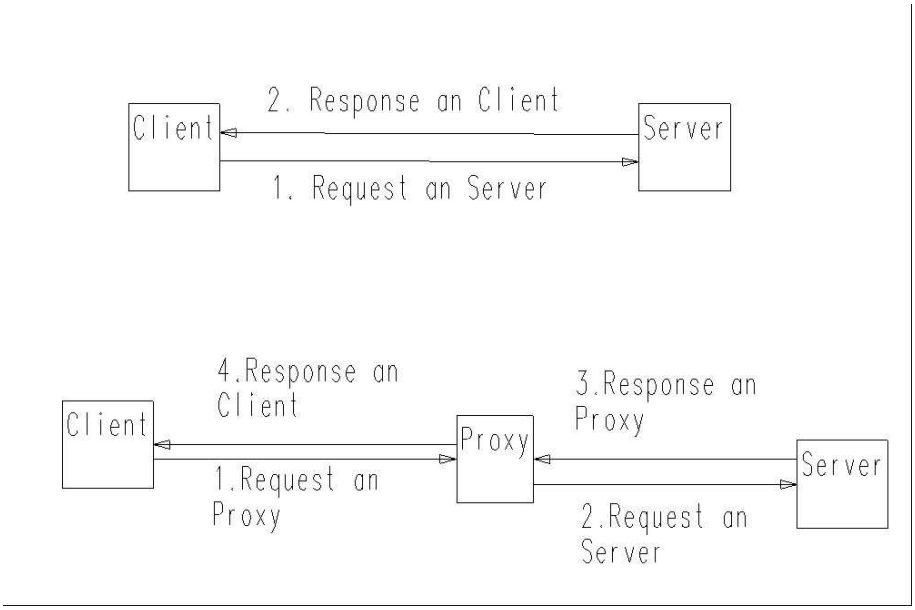
Man unterscheidet zwischen den beiden Versionen 1.0 und 1.1. Beide Versionen erstellen bei Gebrauch eine TCP- Verbindung über den (standardisierten) Port 80: Der Client sendet einen Request an den Server, worauf dieser mit einem Reply-Objekt antwortet. Daraufhin können Anfragen und Antworten (Bei HTTP 1.0 nur jeweils eine!) ausgetauscht werden. Schlussendlich wird die TCP- Verbindung wieder vollständig geschlossen.

Wie bereits angedeutet schließt der Server unter HTTP 1.0 nach jeder einzelnen Anforderung die Verbindung wieder. So muss also bei Seiten, die noch irgendwelche weitere Links beinhalten, für jeden einzelnen Link die Verbindung von neuem hergestellt werden. Diese Verbindungsart nennt man "nicht-persistent"

Eine persistente Verbindung kann mittels HTTP 1.1 aufgebaut werden. Hierbei wartet der Server nach jeder Antwort darauf, dass vielleicht noch eine weitere Anfrage kommt. Sobald alles geladen ist, sendet der Client ein Signal, dass keine weiteren Seiten mehr geladen werden sollen, und der Server schließt die Verbindung.

HTTP ist zustandslos (stateless), was bedeutet, dass weder Client noch Server wissen, was in anderen, früheren HTTP- Verbindungen passierte, jede Anfrage ist völlig neu.

Die Kommunikation zwischen Client und Server kann über ein Zwischensystem einen Cache oder einen Proxy erfolgen.



Request

Die Befehle von HTTP sind in Klartext geschrieben, man sendet also Befehle wie GET, PUT, POST oder HEAD. Das Ende der Anfrage wird durch CRLF angegeben.

Die Allgemeine Struktur von HTTP- Anfragen (Requests) sieht folgendermaßen aus:

```
<Methode> " " <URL> " " <Version> CRLF
<Header-Feld> ":" <Wert> CRLF
<Header-Feld> ":" <Wert> CRLF
...
CRLF
<Datenblock>
CRLF
```

Die wichtigsten Kommandos sind:

- `GET:  Die Antwort enthält nach dem Header den gesamten Inhalt der angeforderten Seite.`
- `HEAD:  Die Antwort enthält nur den Header.`
- `POST:  Damit übermittelt man dem Server einen Datenblock, den der Server in die angegebene URL einfügen soll.`
- `PUT:  Die übermittelten Daten sollen in einer neuen oder bereits existierenden Datei an der angegebenen URL gespeichert werden.`
- `DELETE: Die angegebene Adresse soll vom Server entfernt werden.`

Request Header Felder:

Accept
: Wird zur Aushandlung des Formates der Nutzdaten verwendet und gibt an welche Medientypen der Client akzeptiert. Es können auch generische Typen oder Subtypen angegeben werden. Zusätzlich ist es möglich, zu jedem Typ eine Qualitätsstufe und die maximale Größe in Bytes die akzeptiert wird anzugeben. Der Qualitätsfaktor kann zwischen 0 und 1 gewählt werden. Je größer q, desto mehr preferiert der Client diesen Medientyp. Wird der Qualitätsfaktor weggelassen so wird für den Type q=1 als Default gesetzt.
<Accept> = "Accept" : *<media range> ","

<media ranges> = ("*/*" | <type> "/" "*" | <type> "/" <subtype>) [quality] [maxbytes]
<quality> = ";" "q" "=" ("0" | "1" | float)
<maxbytes> = ";" "mxb" "=" *DIGIT
<float> = < ANSI-C floating Point Repräsentation where(0.0 < float < 1.0)

Accept-Charset
: Gibt an, welche anderen Zeichensätze außer ISO-8859-1 oder US-ASCII der Client akzeptiert.

Accept-Encoding
: Gibt an, welche Kodierungsmethoden der Client akzeptiert. Zur Zeit sind die Kodierungen compress, base 64, gzip & quoted-printable definiert. Es sollte nie 7bit, 8bit oder binary als Kodierung angegeben werden, da sie no-encoding repräsentieren.

Accept-Language
: Gibt an, welche Sprachen der Client bevorzugt. Kann der Server die Anforderung nicht erfüllen, so sendet er das Dokument in irgendeiner Sprache.

Authorization
: Wird zur Autorisierung verwendet.

From
: Sollte die Internetadresse der Person enthalten, die den Client benutzt.

If-Modified-Since
: Wird verwendet, um die Methode GET vom Zustand des Dokumentes abhängig zu machen. Der Server sendet das Dokument nur dann zurück, wenn es auch wirklich verändert worden ist. Dies verringert den Overhead bei der Übertragung deutlich.

Pragma
: Definiert Direktiven die alle Server auf dem Weg der Anfrage einhalten müssen. Dies soll typischerweise Proxy Server davon abhalten, die Art der Anfrage zu ändern. Zur Zeit ist nur eine Direktive definiert und das ist die no-cache Direktive. Sie weist einen cachenden Proxy Server an, das Dokument neu zu laden auch wenn es im Cache vorhanden ist.

Referer
: Gibt dem Client die Möglichkeit die URL anzugeben, von der aus er das neue Dokument anfordert. Dies dient dem Server für Statistiken, zur Optimierung der Cache-Strategien, für Rückwärtslinks usw.

User-Agent
: Gibt den Namen und die Version des Clients an. Dies dient zu statistischen Zwecken und ermöglicht Protokollverletzungen leichter zu identifizieren und in neueren Versionen zu beheben.

Daneben gibt es noch folgende Genral Header, die im Request und im Response auftreten können.

Date
: Gibt das Datum an, an dem die Nachricht erzeugt wurde. Es hat die selbe Semantik wie das orig-date im RFC822.

Forwarded
: Wird von Proxies verwendet um die Schritte zwischen Client und Server anzuzeigen. Das Feld ist analog dem *Received* Feld von RFC822.

Message-ID
: Dieses Feld ist identisch zu denen, die von Internet Mail und USENET Nachrichten verwendet wird. Normalerweise ist es in HTTP Nachrichten nicht enthalten, aber Gateways sollten es erzeugen wenn sie die HTTP Nachricht über ein anderes Protokoll versenden, das dieses Feld benötigt.

MIME-Version
: Sollte nur verwendet werden, wenn die Nachricht wirklich 100% MIME konform ist. HTTP selbst ist nicht 100% MIME konform, so dass Gateways dafür sorgen müssen, dass die Nachricht konform ist, wenn sie sie in eine MIME konforme Umgebung exportieren. Zurzeit verwenden Client und Server dieses Feld leider, obwohl ihr Nachrichten nicht 100% MIME konform sind.

Die Entity Header Felder definieren für Request und Response optionale Metainformationen über die übertragenen Daten, oder falls keine Daten vorhanden sind (bei HEAD Request) über die Resource selbst.

Allow
: Beschreibt die Methoden die für diese Resource möglich sind. Muß vorhanden sein, wenn der Status Code *405 Method Not Allowed* gesendet wird.

Content-Encoding
: Wird benutzt um die zusätzlichen Kodierungsmethoden anzuzeigen, die auf die Resource angewandt wurden. Dies wird verwendet um die Daten zu komprimieren ohne die Identität des zugrundeliegenden Medientyps zu verlieren.

Content-Language

Beschreibt die Sprache in der das Dokument verfaßt wurde.

Content-Length
: Gibt die Länge des Dokuments in Bytes an.

Content-Transfer-Encoding
: Gibt an welche Kodierungsmethoden angewandt werden mußten um eine sichere Übertragung zu gewährleisten.

Content-Type
: Dient zur Typisierung der Nutzdaten. Das Format bzw. der Typ der Nutzdaten wird MIME-konform angegeben. Bsp.: *Content-Type : audio/basic* oder C*ontent-Type : text/html*

Derived-From
: Gibt an welche Version die Resource hatte bevor sie verändert wurde. Wird zusammen mit dem *Versions* Feld verwendet um einer Gruppe von Leuten die gleichzeitige Bearbeitung eines Dokumentes zu ermöglichen.

Expires
: Gibt an, ab welchem Datum dieses Dokument abgelaufen bzw. nicht mehr aktuell ist.

Last-Modified
: Gibt an, an welchem Datum die Resource zum letzten Mal verändert wurde. Die genaue Bedeutung dieses Feldes ist nicht vorgegeben. Sie hängt vom jeweiligen Sender und von der Art der Resource ab.

Link
: Gibt die Möglichkeit eine Beziehung zwischen den Daten und anderen Ressourcen herzustellen. Diese Links geben typischerweise Beziehungen wie die hierarchische Struktur von Dokumenten oder Navigationswege an. Das *Link* Feld ist semantisch equivalent zu dem HTML Tag <*link*>.

Location
: Dieses Feld ist eine frühe Version des URL-Headers und ist eigentlich überflüssig. Es sollte nur aus Kompatibilitätsgründen noch verwendet werden.

Title
: Gibt den Title des Dokumentes an.

URL-Header
: Der URL-Header hat mehrere Verwendungsmöglichkeiten. Im einfachsten Fall gibt er eine URL an mit der der Ursprung der Daten identifiziert werden kann. Wenn der Client eine POST Anfrage an den Server stellt kann der URL-Header dazu dienen eine URL vorzuschlagen unter der das Dokument gespeichert werden soll. Der Server antwortet dann mit der URL im URL-Header unter der das Dokument gespeichert wurde. Falls eine URL auf eine Menge von Varianten eines Dokumentes zeigt, so wird die Dimension dieser Varianten angegeben. Als Dimensionen sind zur Zeit *type, language, version, encoding, charset* und *user-agent* definiert. Die Anfrage nach dieser URL wird von den Einträgen in den korrespondierenden Request Header Feldern wie Accept- Language, Accept, usw. abhängen.

Version
: Gibt die Version des Dokumentes an. Dies wird zusammen mit dem *Derived-From* Feld verwendet um einer Gruppe von Leuten die gleichzeitige Bearbeitung eines Dokumentes zu ermöglichen.

Response

Beim Response wird zuerst eine Statuszeile, die die HTTP Version, einen dreistelligen Statuskode und eine Begründung im Klartext enthält gesendet. Daran schließt sich ein Response Header an, gefolgt vom Datenteil.

```
<status line> <CR><LF>
<response header> <CR><LF>
<DATA>
```

```
<status line> = <http version> <status code> <reason line>
<CR><LF>
<status code> = 3 * DIGIT
<http version> = "HTTP/" DIGIT "." DIGIT
```

Der Status Code ist ein 3 ziffriger Zahlencode als Statusmeldung. Der erste Ziffer gibt die Anwortklasse an, die letzten beiden Ziffern sind nicht kategorisiert. Zur Zeit sind folgende Klassen definiert:

1xx: Informational
: Wird zur Zeit nicht benutzt ist aber für die Zukunft reserviert.

2xx: Succes
: Die letzte Aktion wurde erfolgreich empfangen, verstanden und akzeptiert.

3xx: Redirection
: Zusätzliche Aktionen sind nötig um die Anfrage zu vervollständigen.

4xx: Client Error
: Die Anfrage war entweder syntaktisch falsch oder konnte nicht erfüllt werden.

5xx: Server Error
: Der Server war nicht in der Lage die Anfrage zu erfüllen.

Die wichtigsten Status Codes sind 200 OK, 404 Seite nicht gefunden und 500 Internal Server Error.

Die Reason Line gibt eine kurze textuelle Beschreibung des Status Codes. Während der Status Code für den Client zur automatischen Auswertung gedacht ist, so ist die textuelle Darstellung für den Anwender gedacht der sich nicht alle Codes merken möchte. Die textuelle Darstellung ist nicht zwingend vorgeschrieben, sie kann auch lokal verändert werden, ohne dass dies einen Einfluss auf die Standardkonformität hätte.

Die Felder im Response Header

Die Felder im Response Header ermöglichen es dem Server, Informationen an den Client zu senden, die in der Status Zeile keinen Platz mehr haben. Diese Informationen sind nicht dazu da die Daten in der Antwort näher zu spezifizieren, sondern um nähere Informationen zum Server selbst zu erhalten.

Public
: Dieses Feld spezifiziert eine Liste von vom Server unterstützten, nicht standardisierten Methoden Zum Beispiel MGET oder MHEAD.

Retry-After
: Kann benutzt werden um im Zusammenhang mit dem Status Code *503 Service Unavailable* anzugeben wie lange der Dienst für den Client nicht verfügbar ist. Es kann entweder ein volles Datum oder eine ganzzahlige Anzahl von Sekunden enthalten.

Server
: Gibt die Serversoftware und ihre Version an und ist vergleichbar mit dem User-Agent Feld im Request Header.

WWW-Authenticate
: Dieses Feld muß eingefügt werden, wenn der Server ein *401 Unauthorized* als Status Code zurückgibt. Es beschreibt das Authentifizierungsverfahren und Parameter für die angeforderte URL.

Beispiel einer HTTP Session

```
GET /~whity/index.html HTTP/1.0
Accept: text/plain, text/ps; q=0.8; mxb=32767, text/html

HTTP/1.0 200 OK
Server: NCSA/1.3
Content-type: text/html
Last-Modified: Wed Apr 6 12:56:23 1995
Content-Length: 12344
<html>
<title> Whity´s Heimat </title>
...
```