**HTTP & Developer Tools**

**>** Request

**<** Response

**\*** Zusatzinfos, die curl zur Verfügung stellt

TCP\_NODELAY soll die Segmentpufferung aktivieren/deaktivieren, damit Daten so schnell wie möglich an Peer gesendet werden können (Verbesserung Netzwerkauslastung)

curl -v <http://google.de>

1. **Welche Informationen stehen im Request-Header und was bedeuten diese?**

* HTTP Anfragemethode und gewünschte Version (GET / HTTP/1.1)
* URL, an die, die Anfrage gesendet wird (Host: google.de)
* Informationen über den Client, damit z.B. browserspezifische Antworten gesendet werden können (User-Agent: curl/7.65.3)
* Zeichensätze, die der Client erwartet und verarbeiten kann (Accept: \*/\*)

curl -I <http://google.de> vs. curl -I <https://www.google.de>

1. **Wie unterscheiden sich die beiden Response-Header?**

HTTP-Version! (Achtung dafür muss unbedingt Git Bash verwendet worden sein wegen Curl-Version, nicht Windows cmd); unterschiedliche Statuscodes; unterschiedliche Cache (private, public), unterschiedliche Zeichencodierung

Cache-Control public 🡪 Response darf von jedem Cache (auch einem geteilten) gecached werden

|  |  |  |  |
| --- | --- | --- | --- |
| **Feld** | **curl –I http://google.de** | **curl –I https://www.google.de** | **Unterschied** |
|  | *HTTP/1.1 301 Moved Permanently* | *HTTP/2 200* | Http leitet zu http://www.google.de/ weiter, was der Aufruf mit www nicht braucht. Eimal wird http Version 1.1 verwendet, im anderen Fall 2 |
| Location | *http://www.google.de* |  | Wird nur bei 3xx oder 201 Antworten verwendet. Gibt die Weiterleitungsurl an |
| *Content-Type* | *text/html; charset=UTF-8* | *text/html;*  *charset=ISO-8859-1* | Verschiedene Charsets |
| *Date* | *Sun, 05 Apr 2020*  *10:35:05 GMT* | *Sun, 05 Apr 2020*  *10:36:53 GMT* | Wann die Anfrage gestellt wurde |
| *Expires* | *Tue, 05 May 2020*  *10:35:05 GMT* | *-1* | Wird ignoriert, da im cache-control eine max-age angegeben ist |
| *Cache-Control* | *public, maxage=*  *2592000* | *private, max-age=0* | Caching public, aber läuft ab im Gegensatz zum privaten Caching, das nicht abläuft |
| *Server* | gws | gws | Google Web Server |
| *Content-Length* | 218 |  | Byte-Länge des HTTP Body |
| *X-XSS-Protection* | 0 | 0 | Funktion zur Unterdrückung von XSS-Attacken |
| *X-Frame-Options* | SAMEORIGIN | SAMEORIGIN | Die Seite kann nur als Frame eingebettet werdenn, wenn beide von der gleichen Quellseite stammen. |
| *p3p* |  | *CP="This is not a P3P policy! See*  *g.co/p3phelp for more info."* | P3P-Datenschutz-Policy, wird genutzt, um z. B. Cookie-  Richtlinien durchzusetzen oder zu überprüfen |
| *set-cookie* |  | *1P\_JAR=2020-04-05-10;expires=Tue, 05-May-2020 10:36:53 GMT; path=/; domain=.google.de;*  *Secure*  NID=201=gv4j…. | Cookies werden nicht bei der Weiterleitung  gesetzt, sondern nur auf der endgültigen  Seite |
| *alt-svc* |  | *quic=":443";*  *ma=2592000;v="46,43",h3-*  *Q050=":443";ma=2592000,h3-*  *Q049=":443";ma=2592000,h3-*  *Q048=":443";ma=2592000,h3-*  *Q046=":443";ma=2592000,h3-*  *Q043=":443";ma=2592000,h3-*  *T050=":443";ma=2592000* | Alternative, um die angefragte Ressource  zu erreichen. Da es bei einer Weiterleitung  keine Ressource gibt, hat diese auch keine  Alternative. Es wird beispielweise oft auf den Port 443 verwiesen, welcher der standardisierte Port für HTTPS-Requests ist. |
| *accept-ranges* |  | *none* | Gibt die Möglichkeit von Teilanfragen an,  wird hier von beiden nicht unterstützt |
| vary |  | *Accept-Encoding* | Entscheidet, ob der Cache verwendet  werden soll oder nicht. Da die  Weiterleitung keinen Inhalt hat, wird auch  nicht explizit die Verwendung des  Caches festgelegt |
|  |  |  |  |

<http://google.de> mit dem Browser aufrufen:

1. **Wo finden Sie Informationen über dieses Request und dessen Response? Vergleichen Sie die Angaben zu denen, die curl ausgegeben hat.**

In der Netzwerk-Analyse unter Headers. Die Angaben unter Kopfzeilen sind nahezu identisch, es haben sich nur browserspezifische Dinge verändert (User-Agent, Accept, etc.) und Anfragezeit/Ablaufdatum.

1. **Beschreiben Sie, was passiert.**

Zuerst wird auf http://www.google.de weitergeleitet (301, anhand des Location Headers sieht man wohin die Ressource verschoben wurde). Dort erhält man einen 302 Status der auf https://www.google.de/?gws\_rd=ssl weiterleitet. Auf dieser Seite werden nun einige GET Anfrage gesendet, die Bilder und Funktionalitäten laden.

1. **Die Developer-Tools bieten zwei Möglichkeiten der Darstellung der Response. Welche sind es?**

Formatiert und unformatiert

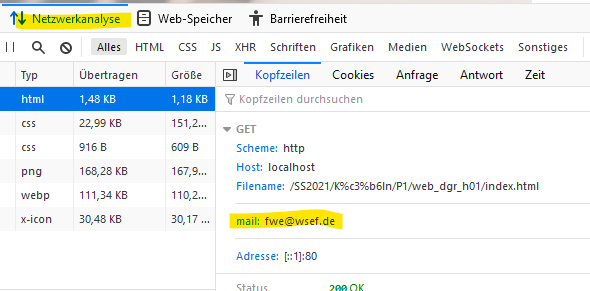
1. **Woran können Sie erkennen, dass es sich um GET-Request handelt und wo kann man in den Developer Tools mit geschickten Query-Parameter einsehen?**

Eine der Spalten im Network-Tab enthält die Methode. Query Parameter befinden sich in der Übersicht unter Datei hinter dem Fragezeichen oder beim Klick auf einen Request unter Parameter

GET:

Chrome: Tab Headers, Reiter: Query String Parameters

~~Firefox: Tab Parameter, Reiter: Suchbegriff~~



1. **An welcher Stelle in den Developer Tools (Tab, Reiter/Stichwort) kann man die gesendeten Daten nun einsehen?  In welchem Format werden die Parameter des POST-Requests übertragen?**

POST:

Netzwerkanalyse,

Chrome: Tab Headers, Reiter: Form Data

Firefox: ~~Tab Parameter, Reiter: Formulardaten~~  Tab Anfrage, Reiter: Formulardaten/Anfrageinhalt

Format: key value Paare: key1=value1&key2=value2

1. **Was bedeutet HTTP/2 Server Push?**

Der Server übernimmt die Initiative zusätzlich Responses zu schicken, ohne dass der User diese angefragt haben muss. (Zum Beispiel Bilder, CSS-Sheets, etc.)

1. **Was bewirkt die Option keep-alive? (Betrachten Sie dabei die HTTP Versionen 1.0, 1.1 und 2.0)**

keep-alive: es wird keine neue TCP-Verbindung aufgemacht. Es wird eine TCP-Verbindung für alle Requests benutzt (Persistent Connection).

-> Keine zusätzlichen 3-Way-Handshake & keine zusätzlichen TCP-Slow Starts

Es existieren viele verschiedene Implementierungen für HTTP 1.0 und HTTP 1.1

1.0: nicht vorgesehen und ursprünglich nicht Teil vom Standard, manche Implementierungen berücksichtigen es aber: https://tools.ietf.org/html/rfc2068#section-19.7.1

1.1: alle Verbindungen sind per default persistent, manche Browser setzen trotzdem keep-alive flag

2.0: benutzt andere im Protokoll integrierte Mechanismen, immer persistente Verbindungen -> keep-alive Header obsolet

(Chrome setzt das keep-alive flag nur bei HTTP 1.1 (ab HTTP 2 gibt es das nicht mehr). Firefox setzt immer keep alive)

1. **Welche Information enthält das Header-Feld User-Agent? Gehen Sie genauer auf die einzelnen Angaben ein.**

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90

The user agent application is Mozilla version 5.0.

The operating system is NT version 10.0 (and is running on a Windows(64-bit) Machine).

The engine responsible for displaying content on this device is AppleWebKit version 537.36 (KHTML, an open-source layout engine, is present too).

The client is Chrome version 77.0.3865.90.

**Mozilla/5.0** is the general token that says the browser is Mozilla compatible, and is common to almost every browser today. --> hat historische und Kompatibilitäts-Gründe

Infos über das Betriebssystem, Browser und Browser-Engine

1. **Was beinhaltet der Content-Type? Welche Content-Types werden beim Aufruf von https://www.fh-aachen.de/ angefordert?**

Angabe im Response Header, in dem dem Browser mitgeteilt wird, um was für einen Typ Ressource es sich bei der übertragenen Datei handelt. 🡪 Browser kann so schneller arbeiten

Auch MIME-Type genannt

* text/html
* text/css
* application/javascript
* application/font-woff
* image/svg+xml
* image/png
* image/jpeg

**TODO:**

**~~Sie sollen schreiben welchen Browser sie benutzt haben~~**

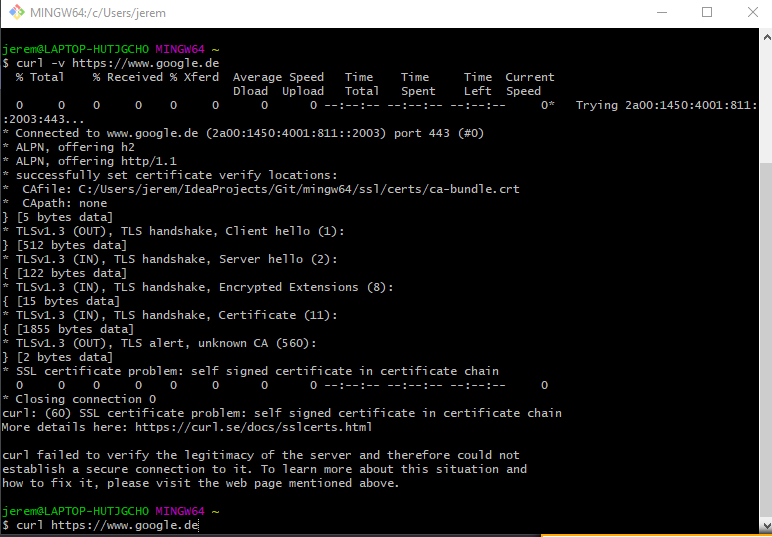
**~~A10 genauer stellen, dass die auf jedes einzelne eingehen~~**

**~~Neue A11: Was beinhaltet der Content-Type? Bzw. welche content-types finden Sie auf fh-aachen.de~~**

**A7: ~~die sollen zusätzlich die Frage beantworten wie das Format der übergebenen Form-Data ist~~ wo es übergeben wird**

**~~Studis sollen Screenshot vom Diner anfügen~~**

**Aufgetretene Probleme:**



**Lösung: Kaspersky schaltet sich bei TLS Anfragen dazwischen! Die Google-Seite hatte ein Kaspersky-Zertifikat! Sah man im Browser! Curl hat dann das Kaspersky Zertifikat als self signed Zertifikat erkannt**