Vorlesung 4

Summary (of Muttiest Point):

Beginn

DIY Web-Server

- Wie würden Services auf anderen Ports funktionieren?
    - Quasi genau so (auf Port lauschen, auf Request reagieren), aber ohne Webseiten zu verschicken, sondern alles andere machen
- Wie "lauscht" man an einem Port?
    - Pointer auf eine Variable (z.B.: einen Character-Array char[20]) (Bibliotheken implementieren die eigentlichen Protokolle, um die eingetroffenen Daten zu verstehen)
    - Endlosschleife, um zu überprüfen, ob Daten in die Variable reingekommen ist
        - Es gibt noch geschicktere Arten als *while(1){...}*, die wir noch lernen werden
- Webserver selbst umsetzen ist heutzutage sehr leicht (aufgrund von Bibliotheken, Node.js, etc.)

Deep-Web

- Internet c DeepWeb c DarkWeb
- **Deep Web** sind alle Webseiten, die von Suchmaschinen nicht indexiert werden.
- **Opaque Web** sind alle Webseiten, die Suchmaschinen indexieren könnten, aber nicht machen (weil es sich z.B.: nicht lohnt).
- **Private Web** sind alle Webseiten, auf die man nur mit Authorisierung Zugriff erhält.
- **Proprietory Web** sind alle Webseiten, auf die man nur mit ... (hat keine große Relevanz mehr).

- **Invisible Web** sind alle Webseiten, die zum Beispiel über Web-Formulare versteckt sind.
- **Truly Invisible Web** sind alle Webseiten, die aus technischen Gründen nicht indexiert werden, weil es aus Datentypen besteht, die von der Suchmaschine nicht verstanden wird.
    - Das zeigt auch ein noch ungelöstes Grundproblem in der Informatik: Wie beschreiben wir alle möglichen Daten für Computer (möglichst automatisiert)
- **Dark Web**: Die tiefste Stelle im Internet, für die man spezielle Software/Konfigurationen benötigt (sprich man kann es mit normalen Browsern nicht erreichen
- **Tor** verbindet Rechner miteinander, über welche die Anfragen geleitet werden, um Anfragen zu anonnymisieren
    - Anonymisierung ist nicht garantiert, im (relativ unwahrscheinlichen) Fall, dass Regierungsbehörden die einzigen mit dir verundenen Rechner sind

Hauptteil

AcidTest & Browsershot

- Aufgabe 1.7
- AcidTest3 hat bei modernen Browsern nur 97/100 ergeben, weil der Test schon veraltet ist
- Browsershot zeigt Screenshots wie die Website auf unterschiedlichen Browsern & Betriebssystemen angezeigt wird. Hat bei uns leider nicht funktioniert

WWW Entities

- HTTP
    - läuft über TCP/IP
    - "in Klartext" - ASCII-encoded
    - Alle Ressourcen sind eigenständig/seperat identifizierbar und addressierbar
- Addressierung läuft über URI (Uniform Ressource Identifier)
    - Syntax für URI: <schema>:<schema-spezifischer Part>
    - Besipiele für Schemas: "mailto", "ftp", "http"

    - URI: URL+URN (L: Locator, N: Name)
    - Absolute & Relative & Parametisierte URL (Absolute alles nach "http://www.", relativ nur nach dem letzten "/"
- Regeln für URI, URL, URN
    - Encoding über ISO Latin I (ähnlich wie ASCII)
    - Kein Freizeichen(" ") - "%20"
    - Reservierte Zeichen um Sonderzeichen aszudrücken ("%", "?", etc.)

Websprachen

- HTML für Inhalt und Strukturierung der Webseite
- Jedes HTML-Dokument beginnt mit "*<!DOCTYPE html>*" und ist über Tags strukturiert
- Tag: <name>....</name>
- Jedes HTML Dokument ist in ein <head> und ein <body> geteilt (<head> für Metadaten, <body> für Inhalt der Website) (bei HTML 5 kann <head> auch ausgelassen werden, weil Browser sehr Fehlertolerant sind)
- Tags werden nicht als Text ausgegeben, sondern sind Strukturanweisungen an
- Diese Strukturierung in Tags ist typisch für Hypertext
- Tags werden in umgekehrter Reihenfolge geschlossen (Bsp.: Hier der Text
- Manche Tags haben kein schließendes Tag (z.B.:
)
- Geschichte von HTML
    - SGML, 1986 (Standard Generalized Markup Language)
        - Meta-Sprache (Sprache über Sprachen)
        - Definiert/Beschreibt die Syntax für andere Sprachen
    - HTML & HTTP wurden von Tim Burners Lee entwickelt
    - HTML (nicht HTML 5) war auf SGML basierend
    - Die meisten Standards fürs Web wurden von w3consortium entwickelt
    - HTML 1.0 (nur von Mosaic Browser unterstützt) (1992)
    - XML (1998)

        - SGML Konzepte wurden auf HTML angewandt
        - XML ist eine Untermenge von SGML
        - Heutzutage wurde XML von JSON abgelöst
        - XML ist in ASCII-Codierung, hat keine festgelegten Tags, und wurde zur Datenübertragung verwendet
    - HTML 4.1 war die letzte HTML Version (1999)
    - XHTML (2000)
        - HTML Version die auf XML basierend ist
        - lässt sich programmatisch auf Fehler überprüfen (weil es eine Untermenge von SGML ist)
        - Seiten sind auf 3 Vorlagen ("DTD" = document type definition) basierend ("Strict", "Transitional", "Frameset")
    - XHTML wurde 2009 still gelegt
    - CSS für Styling
    - CSS 3 hat eine modulare Spezifikation (und viele andere neue Features)
    - HTML 5 ist keine Weiterentwicklung von HTML 4
        - Nicht auf SGML basierend
        - HTML 5 ist also eine komplett neue Sprache
    - HTML 5 ist Standard seit (2014) (HTML 5.1 seit 2017)
    - Neue Features von HTML 5
        - Widgets (z.B.: Kalender-Widgets)
        - DOM ("Document Object Model") - alle Elemente auf der Webseite lassen sich programmatisch (z.B.: über JS) ändern
    - Webseiten werden interpretiert, nicht kompiliert
    - Body-Tag enthalten allen sichtbaren Inhalt der Website und kann auch JS Events enthalten
    - Tags können auch Attribute haben (z.B.: "id", "class")

Ende

HTML-Seite machen & kaputt machen

- Eigene HTML-Seite machen, um mit den Tags rumzuprobieren
- Schauen wie fehler-toleran moderne Browser sind, indem man die Website kaputt zu machen versucht
    - Das hat sich als äußerst schwierig ergeben