WBE-Praktikum 1

Entwicklungsumgebung

Einführung

In diesem ersten Praktikum geht es vor allem darum, die benötigte Infrastruktur einzurichten und ein paar Versuche damit zu machen. Es ist gut möglich, dass Sie einige der Werkzeuge bereits für andere Kurse, eigene Web-Projekte, oder im Rahmen des Vorkurses installiert und verwendet haben.

Bitte betrachten Sie die in den folgenden Abschnitten angegebenen Werkzeuge als Vorschläge und nicht als Vorgaben. Wenn Sie bereits einen Code-Editor im Einsatz haben, welcher auch die Web-Sprachen HTML, CSS und JavaScript beherrscht (das wird normalerweise der Fall sein), können Sie diesen selbstverständlich verwenden. Das gleiche gilt für die Auswahl der eingesetzten Browser. Viele Hinweise in den Folien und Praktikumsaufgaben beziehen sich allerdings auf den im folgenden vorgeschlagenen Visual Studio Code (und speziell auf Erweiterungen für dieses Programm) und den Firefox mit seinen Entwicklerwerkzeugen.

Ausnahmsweise gibt es zu diesem Praktikum einen Foliensatz mit einer Zusammenfassung der wichtigsten Tools. Am besten schauen Sie Folien und die untenstehenden Aufgaben durch und entscheiden je nach Vorkenntnissen, welche Teile neu für Sie sind. Dann können Sie sich auf diese Aufgaben beschränken.

Dokumentation Praktikumsergebnisse

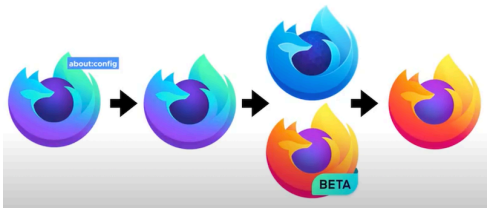
Wir empfehlen Ihnen, ein Projekt unter https://github.zhaw.ch zum WBE-Praktikum anzulegen, in welchem sie die Ergebnisse der Praktikumslektionen dieses Semesters dokumentieren. In den einzelnen Klassen kann es unterschiedliche Abgabemodalitäten für Praktikumsergebnisse geben.

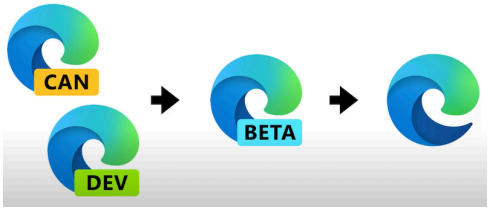
Browser mit Entwicklertools

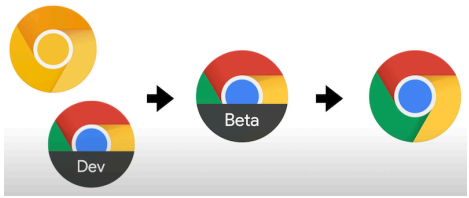
Als Web-Entwickler sollte man verschiedene Browser zum Testen zur Verfügung haben. Es bringt aber wenig, Aufwand zum Testen mit verschiedenen Browsern zu treiben, wenn diese auf der gleichen Rendering Engine basieren: Chrome, Opera, Vivaldi und der neue Edge verwenden die Rendering Engine Blink von Google, grosse Unterschiede in der Darstellung sind also nicht zu erwarten. Hier ein paar Tipps:

- **Chrome** (Google) ist heute der am weitesten verbreiteten Browser. Wer Bedenken hat, den Browser einer Firma zu verwenden, welche den grössten Teil ihres Umsatzes mit Werbung macht (und damit zwangsläufig auch mit dem Sammeln von Daten über Sie), kann **Chromium** verwenden, das Open-Source-Projekt, auf dem der Chrome basiert. Chrome und Chromium haben sehr gute Entwicklertools und sind sehr schnell darin, neue Web-Standards zu unterstützen (welche nicht selten von Google-Entwicklern selbst vorangetrieben werden).
- **Firefox** (Mozilla) ist unter Windows praktisch die einzige verbliebene Alternative zu Browsern, die auf Chromium aufbauen. Gute Unterstützung aktueller Web-Standards. Ebenfalls mit sehr guten Entwicklertools und einer grossen Sammlung von verfügbaren Erweiterungen. Empfehlung: Firefox mindestens als Ergänzung zu anderen Browsern installieren.
- **Edge** (Microsoft) verwendet mittlerweile auch die Rendering Engine des Chromium-Projekts. Als Web-Entwickler sollte man aber auch mit älteren Edge-Versionen testen. Für WBE ist das nicht so wichtig, da Kompatibilität mit älteren Browsern hier nicht im Fokus ist.
- **Safari** (Apple) ist unter iOS und MacOS vorinstalliert und dementsprechend auf diesen Plattformen ein wichtiger Browser. Unter iOS ist die Webkit-Engine des Safari sowieso Vorschrift, das heisst auch Chrome oder Firefox verwenden unter iOS Apple's Webkit-Engine.

Neben der Hauptversion der Browser gibt es auch noch experimentelle Vorabversionen, die neue Features implementieren aber noch Fehler enthalten können. Um das eine oder andere neue Feature ausprobieren zu können, kann es nötig sein, eine solche Vorabversion zu installieren. Fast alles, was wir in WBE behandeln, wird aber in den Hauptversionen der aktuellen Browser funktionieren.[1]







Erweiterungen: Nicht viele nötig, da die Entwicklertools bereits sehr gute Werkzeuge enthalten. Die Erweiterung *Web Developer*[2] ist schon etwas älter, wird aber noch gepflegt und enthält ein paar nützliche Hilfsmittel. Heutzutage kommt man im Web ausserdem kaum um einen Content-Blocker herum. Empfehlenswert hier *uBlock Origin*[3].

[1] Bilder aus: Why are there Four Firefoxes? https://www.youtube.com/watch?v=qQ1oQJn1nQ

[2] https://chrispederick.com/work/web-developer/

[3] https://github.com/gorhill/uBlock#ublock-origin

Code-Editor mit Erweiterungen

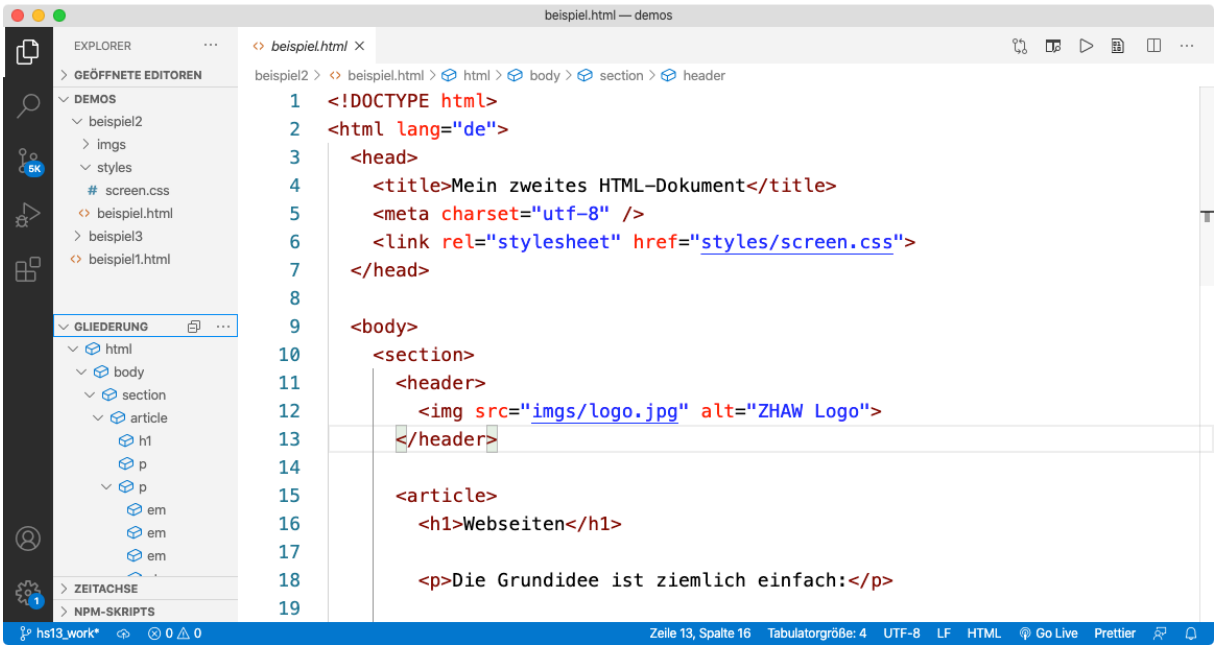
Selbstverständlich benötigen Sie zum Bearbeiten Ihrer JavaScript-Programme und von HTML- und CSS-Code einen geeigneten Code-Editor. Im Prinzip kann jeder Texteditor dafür eingesetzt werden. Mit Vorteil bietet der Editor aber zusätzliche Unterstützung zur Code-Bearbeitung:

- Syntax-Elemente geeignet einfärben. So behält man den Überblick und findet Fehler schneller.
- Unterstützung beim Bearbeiten – etwa Vorschlagen passender Attribute oder Schliessen von Tags beim Bearbeiten von HTML-Code, oder Vervollständigen von CSS-Eigenschaften.
- Plugin-System, um den Editor bei Bedarf erweitern zu können: Module zum Überprüfen oder Erweitern von Code, Syntax zusätzlicher Sprachen, Live-Vorschau der aktuellen Seite, Funktionen zur Verwaltung des Projekts in einem Git-Repository, …

Sie können in WBE gerne Ihren Lieblings-Code-Editor verwenden. Sollten Sie noch keinen haben, empfehlen wir den **Visual Studio Code** (https://code.visualstudio.com, Microsoft). Der VSCode ist mittlerweile stark verbreitet und selbst mit Webtechnologien aufgebaut: Basis für VSCode (und für viele andere Applikationen heutzutage) ist das Electron-Projekt, und dieses baut auf dem Chromium-Browser auf.

Es gibt natürlich diverse Alternativen: Atom (GitHub, jetzt Microsoft, nicht mehr weiterentwickelt), Sublime Text (kostet was, Stern etwas am sinken), Komodo Edit (gratis, auch nicht mehr sehr verbreitet), WebStorm (JavaScript IDE von JetBrains) und noch ein paar mehr. Die Beispiele in WBE werden sich aber meistens auf VSCode beziehen.

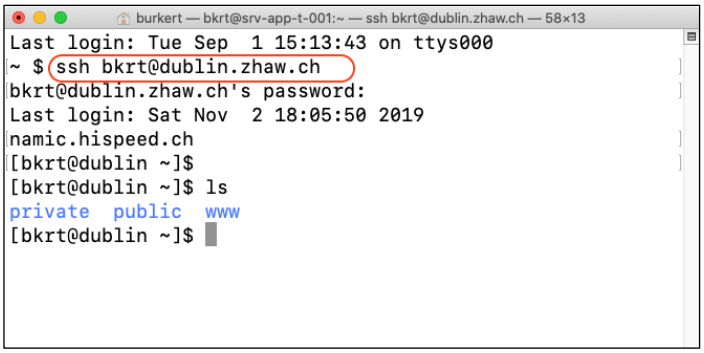
Erweiterungen für den VSCode: HTML, CSS und JavaScript bereits gut unterstützt. Ein paar Tipps zu Erweiterungen gibt's auf den Begleitfolien und nach Bedarf in den Lektionen.

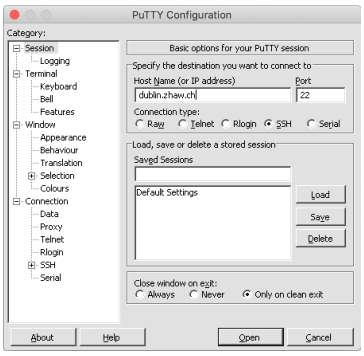


SSH und SFTP

Bei der Entwicklung von Web-Applikationen ist es häufig nötig (oder mindestens hilfreich), sich auf einem Server einloggen und dort auf der Kommandozeile arbeiten zu können. Dazu kann das Kommandozeilen-Tool **ssh** verwendet werden. Es baut eine verschlüsselte Verbindung zu einem Server auf und öffnet dort eine *Shell*, welche Kommandozeilenbefehle entgegennimmt.

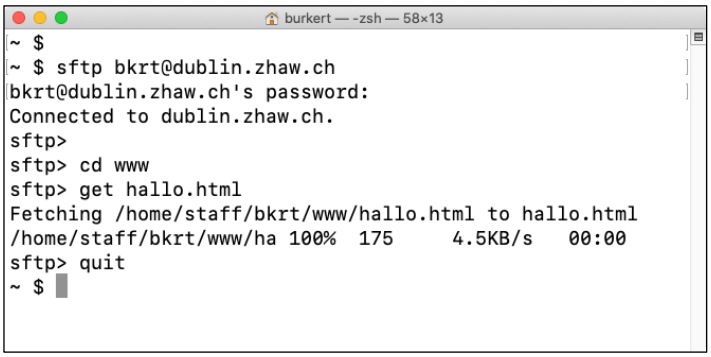
Unter MacOS und Linux steht *ssh* als Befehl im Terminal zur Verfügung. Unter Windows kommt es auf die Version an. Gegebenenfalls müssen Sie *ssh* dort erst aktivieren. Alternativ können Sie unter Windows auch Putty[4] verwenden.





Bei der ersten Verbindungsaufnahme mit einem Server müssen Sie den Fingerabdruck des Servers bestätigen, um sicherzustellen, dass Sie mit dem korrekten Server verbunden sind.

Während SSH dazu dient, auf einem Server remote zu arbeiten, wird SFTP verwendet, um Dateien vom oder zum Server zu übertragen. Auch dafür gibt es ein Kommandozeilen-Tool: **sftp**.



Alternativ können Programme mit grafischer Oberfläche verwendet werden, z.B. **FileZilla**[5].

Wichtiger Hinweis zu FileZilla: Unter Windows enthält der Installer möglicherweise zusätzliche Angebote – *«This installer may include bundled offers»* – achten Sie darauf, nur den FileZilla-Client zu installieren.

[4] https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

[5] Es gibt sicher Programme mit schönerem GUI als FileZilla, aber FileZilla hat sich als zuverlässig und robust erwiesen.

Node.js

Node.js ist eine browserunabhängige JavaScript-Laufzeitumgebung basierend auf der V8-Engine, die auch im Chrome-Browser verwendet wird. Sowohl Node.js als auch die JavaScript-Umsetzungen der Browser basieren auf der von der Organisation ECMA definierten Spezifikation der Programmiersprache JavaScript. Die unterstützten JavaScript-APIs im Browser und in Node.js unterscheiden sich jedoch beträchtlich.

Für die JavaScript-Einführung im ersten Teil des Semesters werden wir Node.js verwenden. Für die Installation gibt es verschiedene Varianten. Empfohlen wird üblicherweise, Node.js mit einem Tool für die Node-Versionsverwaltung zu installieren, z.B. *nvm* oder *n* unter Linux oder MacOS, *nodist* oder *nvm-windows* unter Windows. Das hat den Vorteil, dass Sie Tests mit verschiedenen Node-Versionen machen können. Informationen dazu:
https://docs.npmjs.com/downloading-and-installing-node-js-and-npm

Falls das nicht klappt, können Sie auch das nötige Installationsprogramm direkt hier herunterladen:
https://nodejs.org/en/download/

Noch eine Variante: unter Linux und auf dem Mac können Sie Node.js auch mit einer Paketverwaltung installieren, auf dem Mac zum Beispiel mit *Homebrew* oder *Macports*:

```
$ # Homebrew
$ brew install node

$ # Macports
$ port install nodejs20
```

Mit der Installation von Node.js kommt normalerweise auch das Kommandozeilenprogramm *npm* für die Node.js-Paketverwaltung. Überprüfen Sie es durch Aufruf von *npm* auf der Kommandozeile. Es gibt dann einfach einen Hilfetext aus. Falls *npm* nicht installiert ist, installieren Sie es separat.

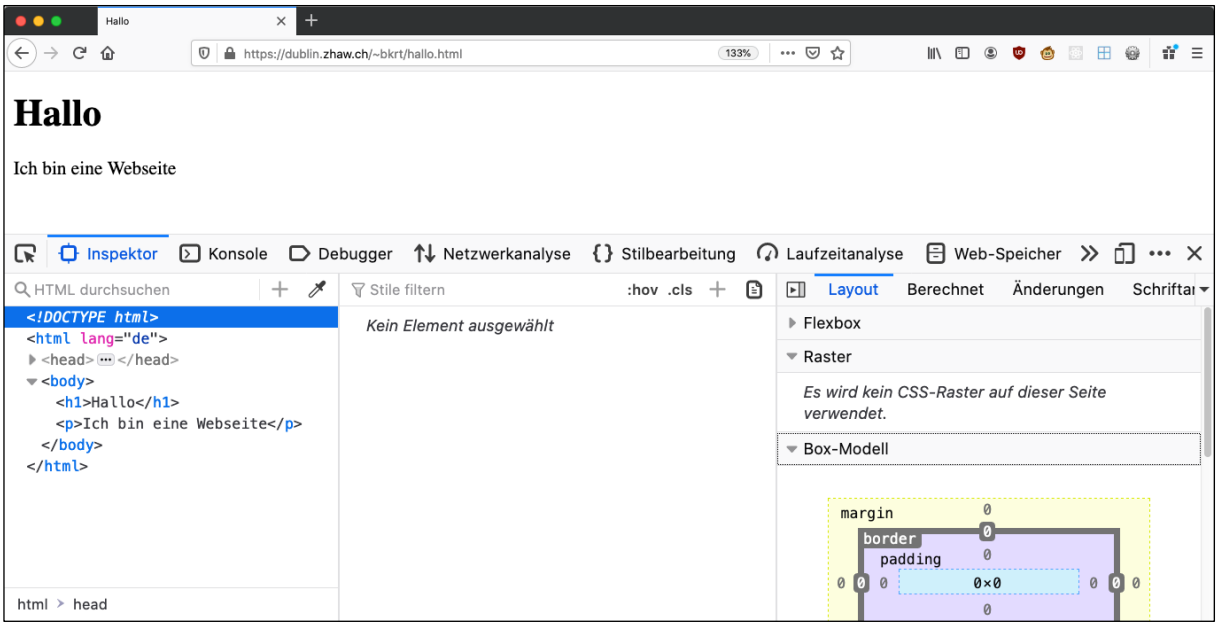
Aufgaben

Es folgen einige Aufgaben zur Infrastruktur, welche Sie nach Bedarf und je nach Vorkenntnissen bearbeiten sollten, wenn Sie die nötigen Installationen abgeschlossen haben. In diesem Praktikum muss nichts abgegeben werden.

Aufgabe 1: Browser und Entwicklertools

Öffnen Sie eine einfache Website im Browser. Ein Beispiel finden Sie hier:[6]
https://dublin.zhaw.ch/~bkrt/hallo.html

Öffnen Sie die Entwicklertools im Browser. Im Firefox sieht das folgendermassen aus:



Da die Seite kein JavaScript und keine Stylesheets enthält, sind die Möglichkeiten in den Entwicklertools für dieses Beispiel gering. Trotzdem eignet sich das Beispiel, um sich einen ersten Überblick über die Entwicklertools zu verschaffen.

- Überprüfen Sie im *Inspektor* den Aufbau des Dokuments. Fahren Sie mit dem Mauszeiger über die einzelnen Elemente (Überschrift und Absatz). Klappen Sie testweise den Kopfbereich <head> auf.
- Klicken Sie mit der rechten Maustaste auf den Absatz <p> (Paragraph) und wählen Sie aus dem Kontextmenü *HTML bearbeiten*. Passen Sie den Text an.
- Klicken Sie mit der rechten Maustaste auf den Absatz <p> (Paragraph) und wählen Sie aus dem Kontextmenü *Knoten klonen*. Prüfen Sie das Ergebnis. Wie können Sie den neuen Knoten wieder löschen? Was geschieht bei *Neuen Knoten einfügen*?
- Testen Sie ein paar weitere Möglichkeiten, die das Kontextmenü auf den Elementen des Dokuments zur Verfügung stellt. Es gibt auch verschiedene Varianten zum Kopieren und Einfügen.

[6] Der dublin.zhaw.ch ist leider nur noch aus dem ZHAW-Netz oder via VPN erreichbar.

- Klicken Sie mit der rechten Maustaste auf den Absatz <p> (Paragraph) und anschliessend im mittleren Unterfenster auf das Pluszeichen. Es wird eine Stylesheet-Regel eingefügt. Ergänzen Sie diese folgendermassen und prüfen Sie das Ergebnis im Browserfenster:

```
p {
    color: goldenrod;
}
```

Alle Änderungen haben wir natürlich nur lokal in der Repräsentation der Seite im Speicher des Browsers durchgeführt. Ein erneutes Laden der Seite setzt sie wieder in den Ausgangszustand zurück.
Noch ein paar Bemerkungen zu den Entwicklertools:

- Neben dem *Inspektor* befindet sich der Eintrag *Konsole*. Die Konsole erlaubt es, JavaScript-Ausdrücke und -Anweisungen einzugeben und auf diese Weise mit der Seite zu interagieren. Wir werden dies in späteren Lektionen noch benötigen. Für den Moment können Sie einmal folgendes ausprobieren:

```
> 2 ** 256
> 2n ** 256n
```

- Die Entwicklertools können an verschiedenen Stellen im Browserfenster platziert oder in einem eigenen Fenster angezeigt werden: Menü mit den drei Punkten auf der rechten Seite.
- Dort finden Sie auch eine Möglichkeit zum Öffnen der Dokumentation zu den Entwicklertools
- Ebenfalls in diesem Menü können die Einstellungen geöffnet werden. Schauen Sie die möglichen Einstellungen durch. Aktivieren Sie mindestens noch das Werkzeug DOM (falls es nicht bereits aktiviert ist), denn auch dieses werden wir in einer späteren Lektion benötigen.

Aufgabe 2: Web-Editor

Legen Sie eine neue leere HTML-Datei im Code-Editor an, zum Beispiel *demo.html*. Die folgenden Angaben beziehen sich auf den Editor Visual Studio Code.

- Geben Sie ein Ausrufezeichen auf der leeren Seite ein. Emmet schlägt daraufhin vor, das Gerüst einer HTML-Seite einzufügen (s. Bild). Emmet kann das Eingeben von HTML- (und teilweise CSS-) Code deutlich beschleunigen, wenn man die passenden Abkürzungen kenn. In VSCode ist es vorinstalliert, in den meisten anderen Code-Editoren lässt es sich nachrüsten (https://emmet.io).
- Ergänzen Sie noch den Titel des Dokuments und fügen Sie im Inhaltsbereich (<body>) noch eine Überschrift und einen Absatz hinzu. Wenn Sie gar kein HTML kennen, können Sie sich am Beispiel in der letzten Aufgabe orientieren. Speichern Sie das Dokument.
- Betrachten Sie noch die Leiste ganz unten im Editor. Dort ist eingestellt, wie beim Drücken der Tab-Taste eingerückt werden soll (im Bild: *Leerzeichen: 4*), welche Zeichencodierung eingestellt ist (*UTF-8*), welches Zeilenendezeichen eingestellt ist (*LF* für Linefeed) und welche Syntax für das aktuelle Dokument verwendet wird (hier: *HTML*).
- In der Symbolleiste auf der linken Seite finden Sie ganz oben den *Explorer* mit den Dateien des Projekts und weiter unten *Erweiterungen*, wo Sie sich über installierte Erweiterungen informieren und zusätzliche Erweiterungen suchen und installieren können.

Eine grundlegende Einführung in den VSCode ist an dieser Stelle nicht möglich. Es gilt: «Learning by doing». Für Interessierte gibt es hier einige einführende Artikel:
https://www.microsoft.com/de-de/techwiese/know-how/visual-studio-code-01-die-grundlagen.aspx

Aufgabe 3: SSH, SFTP

Beim ersten Einloggen auf dem *dublin.zhaw.ch* werden einige Einstellungen und Berechtigungen gesetzt. Daher wird in den folgenden Schritten zunächst SSH und dann SFTP verwendet.

- Loggen Sie sich via SSH auf dem *dublin.zhaw.ch* ein. Selbstverständlich müssen Sie statt „bkrt" in den folgenden Schritten Ihr Kurzzeichen eingeben. Es öffnet sich eine Kommandozeile, in der Sie Unix-Befehle eingeben können.

```
~ $ ssh bkrt@dublin.zhaw.ch
bkrt@dublin.zhaw.ch's password:
Last login: Thu Sep  3 16:49:13 2020
[bkrt@dublin ~]$
[bkrt@dublin ~]$ ls
private  public  www
[bkrt@dublin ~]$
[bkrt@dublin ~]$ exit
```

Bei der ersten Verbindungsaufnahme mit dem Server müssen Sie den Fingerabdruck des Servers bestätigen, um sicherzustellen, dass Sie mit dem korrekten Server verbunden sind. Dieser Fingerabdruck kann in verschiedenen Codierungen angezeigt werden.

- Kopieren Sie die in der letzten Aufgabe erstellte HTML-Datei ins Verzeichnis *www* in Ihrem Home-Verzeichnis auf dem dublin-Server. Verwenden Sie *sftp* auf der Kommandozeile oder einen grafischen SFTP-Client (wie FileZilla, Hinweis dazu s.o.).
- Öffnen Sie Ihre HTML-Datei via HTTP im Browser.

Aufgabe 4: SSH

Auf dem Server Dateien und Verzeichnisse anlegen oder löschen, Dateien anlegen, bearbeiten, verschieben, umbenennen, oder deren Berechtigungen ändern, das sind Aufgaben, die Sie mit einer Remote Shell erledigen können.

- Öffnen Sie erneut eine SSH-Verbindung zum *dublin.zhaw.ch*.
- Wechseln Sie ins Verzeichnis *www* und prüfen Sie, ob die Datei, welche Sie in der letzten Aufgabe angelegt haben, auffindbar ist. Die wichtigsten Kommandos auf der Unix-Shell sollte jeder Informatiker kennen. Bei Bedarf finden Sie im Internet zahlreiche Tutorials. Ein Cheat Sheet mit einigen Kommandos ist als Datei *unix101.pdf* bei den Praktikumsunterlagen zu finden.
- Öffnen Sie Ihre HTML-Datei mit dem vi, einem Editor, der auf der Kommandozeile verwendet werden kann (tatsächlich wird die verbesserte Variante *vim* geöffnet):
  `vi demo.html`
  Machen Sie ein paar Änderungen in der HTML-Datei und laden sie diese neu im Browser.

Der vi gilt als schwierig zu verwenden, eine Handvoll Kommandos genügen aber für die meisten Fälle: Mit der Taste **i** wechseln Sie in den Einfügemodus und können nun Text eingeben, den Cursor platzieren Sie dazu mit den Pfeiltasten, mit **ESC** wechseln Sie zurück in den Kommandomodus, mit **:w** speichern Sie die Datei und mit **:q** beenden Sie den Editor. Auch hier gilt: man sollte mindestens die benötigten Kommandos kennen, um schnell ein paar Änderungen an einer Datei auf dem Server machen zu können. Kleine Hilfe: *vi_cheat_sheet.pdf* bei den Unterlagen.

Aufgabe 5: HTTP

Abschliessend noch zwei weitere Kommandos, *curl* und *telnet*. Wenn diese im Terminal Ihres Systems nicht zur Verfügung stehen, können Sie die folgenden Schritte einfach auf dem *dublin* ausführen, zu dem Sie ja problemlos eine SSH-Verbindung herstellen können.

- Stellen Sie mit host die IP-Adresse des Servers *dublin.zhaw.ch* fest:
  `host dublin.zhaw.ch`
- Verwenden Sie *curl*, um Ihre HTML-Datei *demo.html* via HTTP vom Server zu laden:
  `curl https://dublin.zhaw.ch/...`

Mit *curl* können Sie nicht nur Dateien via HTTP laden: neben GET lassen sich auch andere HTTP-Kommandos verwenden, etwa auch Formulardaten zum Server schicken. Zudem spricht curl auch zahlreiche andere Protokolle (https://ec.haxx.se).

Nun noch *telnet*: es erlaubt, eine TCP-Verbindung zu einem Server auf einem bestimmten Port herzustellen und textbasiert mit dem Server zu kommunizieren. So können wir direkt mit dem Webserver per HTTP sprechen. Da der dublin.zhaw.ch keine Verbindungen auf Port 80 mehr erlaubt, verwenden wir für diese Übung einen anderen Server:

- Stellen Sie eine Verbindung zu Port 80 des *waikiki.zhaw.ch* her:
  `telnet waikiki.zhaw.ch 80`
- Senden Sie ein HTTP-Kommando an den Server:
  `GET /~bkrt/hallo.html HTTP/1.0`
  Zum Abschluss müssen Sie zweimal ENTER drücken. Der Server antwortet mit dem HTTP-Header, dann folgt eine Leerzeile und dann der Inhalt der Anfrage, die HTML-Datei. Anschliessend wird die Verbindung unterbrochen.


burkert — bkrt@srv-app-t-001:~/www — ssh bkrt@dublin.zhaw.ch — 90×18
[bkrt@dublin www]$ telnet dublin.zhaw.ch 80
Trying 160.85.67.138...
Connected to dublin.zhaw.ch.
Escape character is '^]'.
GET /~bkrt/hallo.html HTTP/1.0
HTTP/1.1 200 OK
Date: Thu, 03 Sep 2020 18:02:46 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16
Last-Modified: Thu, 03 Sep 2020 10:56:54 GMT
ETag: "fa-5ae669c32de02"
Accept-Ranges: bytes
Content-Length: 250
Connection: close
Content-Type: text/html; charset=UTF-8
<!doctype html>
<html lang="de">

Soweit die Einführung in einige wichtige Werkzeuge und Programme, die in der Web-Entwicklung wichtig sind. Im Laufe des Semesters wird es immer wieder Ergänzungen und weitere Hinweise zu dieser Sammlung geben.