AsciiDoc Einführung

Holger Zuleger

Versionsgeschichte

21. Feb 2021	HZ

Inhaltsverzeichnis

Zusammenfassung

AsciiDoc ist ein textbasiertes Eingabeformat zur Erstellung von Artikel oder Bücher. Es basiert auf dem DocBook XML—Standard, und kann für die Ausgabe von PDF-Dokumenten, eBooks und insbesondere auch HTML Dokumenten verwendet werden. Die Eingabe besteht dabei aus reinen Textdateien (Plaintext) angereichert um einfache Formatierungszeichen.

1. Was ist AsciiDoc?

1.1. Einführung

AsciiDoc ist ein textbasiertes Eingabeformat zur Erstellung von Artikel oder Bücher. Es basiert auf dem DocBook XML—Standard, und kann für die Ausgabe von PDF-Dokumenten, eBooks und insbesondere auch HTML Dokumenten verwendet werden. Die Eingabe besteht dabei aus reinen Textdateien (Plaintext) angereichert um einfache Formatierungszeichen.

Dabei gehört AsciiDoc zu der Familie der *Markdown* Sprachen. Bei diesen geht es darum, bei der Erstellungen eines Textes sich möglichst auf die Inhalte zu konzentrieren, und nicht wie diese anschließend formatiert werden.
Trotzdem soll der Eingabetext bereits *lesbar* dargestellt sein und nicht durch Formatierungs- und Kontrollanweisungen überfrachtet werden, wie dies häufig bei Satzsystemen und z.B. auch bei Latex- oder groff-Dokumenten der Fall ist. Idealerweise sind Hervorhebungen bereits im Text als solche zu erkennen und dienen gleichzeitig als Kontrollanweisung um die Darstellung im Ausgabeformat zu perfektionieren.

Darüber hinaus bietet die Eingabe als reiner Text vielfältige Möglichkeiten bezgl. der gemeinsamen Erstellung und Änderung von Dokumenten. Hierzu gibt es eine informative Zusammenfassung bei Heise-Online [https://www.heise.de/hintergrund/Documentation-as-Code-mit-Asciidoctor-4642013.html?seite=all].

Markdown Sprachen

Die Idee, einfachen Plaintext durch ein Programm zu jagen und als Ergebnis ein ansprechendes HTML Dokument zu bekommen, und *gleichzeitig* eine gut lesbare Eingabedatei als Text zu haben, stammt von John Gruber. Von ihm kommt auch das erste in Perl geschriebene `markdown` Programm. Die syntaktische Beschreibung [https://daringfireball.net] auf seiner Webseite gilt heute noch als Maßstab für ähnliche Ansätze.

Die Möglichkeiten zur Textstrukturierung sind jedoch begrenzt und im wesentlichen auf HTML Ausgaben fixiert. Daher gibt es diverse Erweiterungen der Markdown Syntax, wie z.B. MultiMarkdown [https://multimarkdown.com/] oder Pandoc [https://pandoc.org/], aber auch Versuche der Standardisierung (CommonMark [https://commonmark.org/]).

Andere Eingabeformate, die einen ähnlichen Ansatz verfolgen sind z.B. ReStructuredText [https://www.writethedocs.org/guide/writing/reStructuredText/], oder die Eingabesyntax, die häufig bei Wikis oder bei GitHUB und Co. verwendet werden.

Alle Markdown Sprachen haben im wesentlichen HTML als Ausgabesyntax vorgesehen. Bei AsciiDoc ist der Fokus von vornherein ein anderer.

DocBook

Bei DocBook handelt es sich um eine XML Dokument Definition zur Beschreibung des formalen Aufbaus von Dokumenten wie Bücher oder Artikel. AsciiDoc ist ein Frontend zur Erstellung dieser XML-Dokumente.

Aus diesem Zwischenformat wird dann erst das Zielausgabeformat, also z.B. HTML oder PDF erzeugt. Das Aussehen des Endproduktes wird erst bei diesem Schritt erzeugt, und unterscheidet sich u.U. sehr stark in Abhängigkeit des Ausgabemediums.

Bei HTML Ausgabe wird das Aussehen durch Cascading Style Sheets bestimmt. Bei der PDF Ausgabe hängt es vom eingesetzten Backend Prozess ab. In der Regel wird über dblatex [http://dblatex.sourceforge.net/] ein Latex Dokument erzeugt, und dieses dann in PDF überführt.

Anwendungsgebiete von Markdown Sprachen

Schaut man sich die unterschiedlichen Varianten und Programme an, kann man grob folgende Kategorisierung vornehmen:

1. Markdown und Weiterentwicklungen
   Diese sind sehr gut für kurze Dokumente (z.B. Protokolle, oder Readme Dateien) von bis zu zwei, drei Textseiten geeignet. Der primäre Fokus liegt auf dem Textdokument an sich. Die Möglichkeit der „hübscheren“ Darstellung als HTML Dokument ist ein Goodie.

2. ReStructuredText
   Wird sehr gerne für die Programmdokumentation genutzt. Der Haupt Fokus liegt in der Darstellung im Web mit Inhaltsverzeichnissen. Hiermit werden durchaus auch größere Dokumente erstellt.

3. AsciiDoc (DocBook)
   Dies ist das einzige Werkzeug welches, zumindest gleichwertig zur Webausgabe, auch eine Ausgabe in anderen Formaten vorsieht. Als allererstes ist dabei PDF zu nennen aber auch ePub wäre möglich.
   Als Zieldokumente sind längerer Artikel, wissenschaftliche Veröffentlichungen und ganze Bücher im Fokus.

1.2. Integration und Tools

`Asciidoc` ist ein Python Kommandozeilenprogramm und fügt sich somit gut in die Unix Umgebung ein. Einer Installation unter Windows steht jedoch nichts im Wege.

Text Erstellung

Da zur Eingabe Textdateien verwendet werden, ist ein guter Texteditor eine Voraussetzung. Schön dabei: Jeder kann den Editor seiner Wahl benutzen. [1]
Falls ein Syntax Highlighting gewünscht ist wird dazu das GNU Programm `source-highlight` benötigt.

Für die PDF Ausgabe wird entweder `dblatex` oder `FOP` benutzt. Die erzeugten PDF Dokumente unterscheiden sich in der Darstellung durchaus. HTML Dokumente werden mit „Bordmitteln“ erzeugt.

Bei umfangreicheren Dokumenten, die sinnigerweise auf mehrere Einzeldateien aufgesplittet sind, kann ein `make` Programm nützlich sein.

[1] Und ein Flamewar zwischen `vi` und `emacs` Usern bleibt aus. Alternativen dazu gibt es sowieso nicht wirklich.

Collaboration

Ein großer Vorteil von reinem Text als Eingabedatei, ist die einfache Möglichkeit Unterschiede in mehreren Dokumentenversionen über `diff` zu ermitteln. Eine Versionsverwaltung wie `git` ist dabei sehr vorteilhaft, ermöglicht sie doch das gemeinsame Arbeiten an ein und demselben Dokument ohne sich dabei zu stören.
Fügt man eine Collaboration Plattform à la GitHub [https://github.com] oder GitLab [https://gitlab.com] hinzu, hat man auch das Problem der Datensicherung gelöst.

Ausgabe Generierung

Die Erzeugung des Ausgabedokumentes erfolgt über einen Aufruf von `asciidoc` für HTML Ausgaben oder `a2x` für PDF und die anderen Ausgabeformate.

Beispiel 1. Erzeugung der Ausgabedatei

```
$ asciidoc beispiel.txt
$ ls -rt beispiel.*
beispiel.txt
beispiel.html
```

```
$ a2x -f pdf beispiel.txt
$ ls -rt beispiel.*
beispiel.txt
beispiel.html
beispiel.pdf
```

2. Dokument Gliederung

Da es sich bei AsciiDoc Dokumenten letztlich um DocBook Dokumente handelt, und diese wiederum XML-Dokumente sind, gibt es eine zugrundeliegende formale Struktur, die sich auch im Textdokument widerspiegelt.

Da allerdings alle Strukturelemente optional sind, und die meisten anderen mehrfach auftauchen können, ist von der Struktur nicht sehr viel zu sehen. Trotzdem kann man sagen, dass ein AsciiDoc Dokument aus

a. einem Kopfbereich (Header) besteht, gefolgt von
b. einem oder mehreren Kapiteln, die sich wiederum aus
c. einem oder mehreren Absätzen zusammensetzen, die ihrerseits aus
   - Sätzen
   - Tabellen
   - Bildern, Grafiken und
   - formatierten Blöcken

bestehen.

2.1. Kopfbereich (Header)

Im Kopfbereich wird im wesentlichen der Titel des Dokuments festgelegt, gefolgt von dem Autor:innen Name und Emailadresse, sowie Datum und Revisionsnummer. Alle Angaben sind optional.

Im Header kann man zusätzlich Dokumenten Einstellungen vornehmen. Diese müssen dann nicht bei Aufruf des `asciidoc` Programms über Optionen angegeben werden. Die Einstellungen werden über Variablen vorgenommen. Die Definition besteht aus einem Variablennamen, eingeschlossen in Doppelpunkte, gefolgt von dem Wert in der gleichen Zeile. Boolsche Variablen bekommen keinen Wert. Sollen diese negiert werden, schreibt man ein Ausrufungszeichen nach dem Variablennamen.

Beispiel 2. Ein Beispielheader

```
Titel des Dokumentes
====================
:Author: Bibi Bloxberg
:Email: bbx@example.de
:Revision: v0.1
:Date:  23. Januar, Corona 2nd
// Ab hier folgen die Optionsvariablen
// Können auch mit "-a opt" auf der Kommandozeile gesetzt werden
:doctype: article
:lang: de
:numbered:      // turn section numbering on
:toc:           // generate table of contents
:toclevels: 3
:data-uri:      // embed icons in html source instead of creating links
:icons:         // use icons instead of text in admonition blocks (this is the default)
//              // :icons!: switch to text instead of icons
//:theme: flask
//      Metatags for the HTML output
:keywords:      AscciiDoc, Textformatierung, DocBook, Plaintext
:description:   AsciiDoc ist ein textbasiertes Eingabeformat +
                zur Erstellung von Artikel oder Bücher.
```

Der Header endet mit der ersten Leerzeile. Danach beginnt der Inhalt.

Die im Header gesetzten Werte können auf der Kommandozeile über die Option `-a` neu spezifiziert werden.

```
$ asciidoc -a toc2 -a 'numbered!' beispiel.txt
```

Diese überschreiben die Werte innerhalb des Dokumentes!

Zugriff auf Variablen. Auf den Inhalt von Variablen wird zugegriffen, indem man den Variablennamen in geschweifte Klammern einschließt. Dadurch wird der Wert der Variablen an der aktuelle Stelle eingefügt.

Bedingte Bearbeitung. Variablen können auch in Bedingungen abgefragt werden. Soll bestimmter Eingabetext z.B. nur im PDF Backend erzeugt werden kann man dies über

```
ifdef::backend-docbook[]
[index]
== Index
endif::backend-docbook[]
```

erreichen. Im Beispiel wird bei HTML Ausgabe kein Index erzeugt.

2.2. Kapitel (Section)

Kapitelüberschriften werden am enfachsten, und das ist auch die hier empfohlene Methode, durch ein bis fünf Gleichheitszeichen am Anfang einer Zeile, gefolgt von der Kapitelüberschrift, eingeleitet. Level 0 (Ein Gleichheitszeichen) stellt den Dokumententitel dar.

Eine Alternative Syntax besteht in unterschiedlicher Art der Unterstreichung.

Darüber Hinaus kann der Level auch als Attribut angegeben werden, und durch das Attribut `leveloffset` auch angepasst werden, was beim Zusammenführen von Dokumenten aus unterschiedlichen Dateien hilfreich sein kann.

Kapitel können automatisch nummeriert werden. Kapitelnummerierung wird entweder über die Variable `:num:` im Header, oder über die Option `-n` bei Aufruf von AsciiDoc eingeschaltet.

Tabelle 1. Kapitelüberschriften

Level	Unterstrichen	Einzeilig	Verwendung
0	`==========`	`= Text =`	Dokumententitel
1	`----------`	`== Text ==`	Kapiteltitel (Level 1)
2	`~~~~~~~~~~`	`=== Text ===`	Kapiteltitel (Level 2)
3	`^^^^^^^^^^`	`==== Text ====`	Kapiteltitel (Level 3)
4	`++++++++++`	`===== Text ====`	Kapiteltitel (Level 4)

Bei den einzeiligen Titeln können die Gleichheitszeichen nach der Überschrift auch weggelassen werden.

2.3. Absatz (Paragraph)

Ein Absatz besteht aus einem oder mehrerer Sätzen und endet mit einer Leerzeile. Sätze können (und sollen) in der Eingabedatei mit harten Zeilenvorschüben an beliebigen Stellen versehen werden. Es empfiehlt sich ein Editor, der bei ca. 75 Zeichen einen Zeilenvorschub einfügt. In der Formatierung durch `AsccìDoc` werden dann die harten Zeilenvorschübe entfernt und die Zeilen werden bis zur maximalen Zeilenlänge aufgefüllt (Füllmode). Je nach Konfiguration entsteht ein links- oder rechtsbündiger Flattersatz bzw. beidseitig ausgerichteter Text.[2]
Soll ein Satz in einer neuen Zeile beginnen (so wie dieser), d.h. der harte Zeilenvorschub (Hard Line Break) in die Ausgabe übernommen werden, muss in der vorangegangenen Zeile am Ende ein Leer- und Plus Zeichen (+) angehängt werden.
Das Ende eines Absatzes wird durch eine Leerzeile markiert.

Absatzüberschriften

Jeder Absatz (wie auch dieser) kann mit einer Überschrift versehen werden. Überschriften beginnen mit einem Punkt am Zeilenanfang. Absatzüberschriften sind insbesondere bei speziell formatierten Absatzblöcken, wie z.B. Zitate oder Beispielblöcke, interessant.

[2] Bei HTML Ausgabe linksbündig, bei PDF Ausgabe wird auch Blocksatz unterstützt

Text und Textauszeichnungen

Der Text des Dokumentes kann unterschiedlich hervorgehoben und dargestellt werden. So existieren Textauszeichnungen wie z.B. **Fettschrift** (*Fett*) oder *Kursiv* (_Kursiv_). Für einzelne **F**ette **B**uchstaben innerhalb eines Wortes, werden die Sonderzeichen gedoppelt (`**F**ette **B**uchstaben`).
Die Darstellung von `nicht-proportionalem` Text (`+monospace+`) kann auf zwei Arten erfolgen. Durch einschließen in +—Zeichen oder in umgekehrte Hochkommata (Backticks: ``). Der Unterschied ist subtil: Bei Backticks findet keine weitere Ersetzung statt. Sie eigenen sich daher besser für Beispiel angaben oder Kommandos (z.B. `ls -l`) innerhalb des Satzes.

Anführungszeichen werden je nach Spracheinstellung formatiert, werden sie doch im Deutschen anders als z.B. im Englischen geschrieben. Damit das klappt, müssen sie in der Eingabe entsprechend genutzt werden. Text in „Anführungszeichen“ wird in der Eingabe in doppelte Backticks am Anfang und doppelte Hochkomma am Ende (``Anführungszeichen'') d.h. nicht in "Anführungszeichen" geschrieben.[3] Verwirrt? Nun, es ist nicht so schwer…

Texte(teile) können Hoch- (`^2^`) oder Tiefgestellt (`~2~`) werden. Damit kann man $500m^2$ genauso gut darstellen wie H_2O.

Textattribute. Mindestens bei der HTML Ausgabe werden Textattribute unterstützt. Damit kann Text Größer oder kleiner oder auch unterstrichen, überstrichen, oder durchgestrichen werden.
Als Attribute gelten auch Farbangaben für Vorder- und Hintergrund.

Die Angabe der Attribute erfolgt vor dem Text in eckigen Klammern. Der Text selbst ist in # eingeschlossen. Die Attribute sind `big`, `small`, `underline`, `overline`, `line-through` sowie die 16 HTML Grundfarben (z.B. `red`) mit angehängtem `-background` für die Hintergrundfarbe (`yellow-background`).

Besondere Zeichen. Neben den Anführungszeichen, existieren noch andere Zeichenfolgen die automatisch in der Ausgabe anders formatiert werden. Pfeile rechts → (`->`) oder links ← sowie Doppelpfeile ⇒ (`=>`) werden genauso verstanden wie z.B. ein © Copyright (`(C)`) oder ® Registered (`(R)`) ™ Trademark (`(TM)`). Und zwei Minuszeichen werden zu einem em—Dash, sowie drei Punkte zu einer „ellipsis“ werden…

Sonderzeichen können als Symbol angegeben werden. Als Beispiel möge das Dagger † Symbol (`&dagger;`) dienen. Sonderzeichen können auch über `&#hex;` spezifiziert werden. Allerdings funktioniert dies nicht generell: Der bei der PDF Generierung verwendete Zeichensatz kennt wohl kein Grad Celsius # (`&#8451;`) Symbol.[4] Damit erscheint es im HTML Dokument und fehlt im PDF :-(
Und dblatex hat generell ein Problem mit der Verarbeitung der HTML Sonderzeichen.

Literale Absätze

```
Beginnt ein Absatz mit einem Leer- oder Tabulatorzeichen,
d.h. ist der Absatz eingerückt, wird er literal gedruckt.
Es wird eine monotype Schrift verwendet und der
Zeilenumbruch ist wie im Eingabetext.
```

[3] Was bei der Eingabe eher nervig ist. Da liebe ich doch mein .Q Makro in `\[gnt]roff`. :-)
[4] Nach Umstellung auf den Font *FreeSerif* funktioniert auch das

Das gleiche erreicht man indem **vor** dem Absatz eine Zeile `[literal]` eingefügt wird.

```
Hier ist ein Beispiel zu sehen.
```

Zitate

Zitatabsätze werden durch eckige Klammern in denen das Schlüsselwort `quote`, und durch Komma getrennt der Autor und die Quelle plus Erscheinungsjahr angegeben ist, eingeleitet.

> No news are good news.
>
> — H. Zuleger *Unix-Einführung (1992)*

Hinweis Absätze

Es gibt fünf unterschiedliche Hinweis Absätze. Sie alle werden durch eine Hinweisstufe wie `TIP:`, `NOTE:`, `IMPORTANT:`, `WARNING:` oder `CAUTION:` eingeleitet. In der Ausgabe erscheint entweder ein den Absatztyp beschreibendes Wort, oder eine kleine Grafik (siehe Schalter `:icons:` im Kopfheader).

Anmerkung

Der Text kann dann durchaus aus mehreren Zeilen bestehen, darf allerdings nur **ein** Absatz sein. Für Hinweis Blöcke die aus mehreren Absätzen bestehen siehe Kapitel „Absatzblöcke“.

2.4. Blockformatierung

Blöcke sind Bereiche mit besonderer Formatierung, die aus mehreren Absätzen bestehen können. Anfang und Ende sind durch jeweils eine Zeile mit mindestens vier Trennzeichen bestimmt. Die Trennzeichen legen gleichzeitig die Art des Blockes und damit seine Formatierung fest.

Beispielblöcke. Im Dokument sind bereits `Beispiele` als ein Bereich mit spezieller Formatierung gezeigt worden. Sie bekommen automatisch eine Nummerierung falls man dem Block einen Titel gibt. Der Titel ist ein Zeile vor dem Block die mit einem Punkt beginnt.

Beispielblöcke werden im Programmcode durch (mindestens) vier Gleichheitszeichen in einer Zeile begonnen und beendet.

Beispiel 3. Beispielblock eingerückt (Quelltext)

```
.Beispielblock
====
        Damit dieser Beispielabsatz in Monospace gedruckt und
        nicht umgebrochen wird, muss der Inhalt eingerückt sein.
====
```

Generell wird eingerückter Text literal, also in monospace und nicht aufgefüllt dargestellt.

Beispiel 4. Beispielblock nicht eingerückt

Sonst würde der Beispiel Block so formatiert werden.

Beispielblöcke können wie Hinweis Absätze formatiert werden. Hierzu muss **vor** dem Beginn des Blockes in eckige Klammern eingeschlossen der Hinweis Typ (siehe Liste oben) angegeben werden:

Tipp

Man kann einen Hinweis Absatz natürlich auch immer als Hinweis Block eingeben. Der Absatz ist jedoch im Eingabedokument prägnanter darstellbar.

Zitatblöcke. Eine andere Art von Formatblöcken sind Zitate. Bei diesen wird der Autor und die Quelle in eckige Klammern **vor** dem Block angegeben. Der Abschnitt wird durch vier Unterstriche in einer Zeile begonnen und beendet.

Ein Zitatblock

Fürchte Dich nicht vor einem großen Schritt. Du kannst einen Abgrund nicht mit zwei kleinen Sprüngen überqueren.

— David Lloyd George *engl. Politiker (1863-1945)*

Beispiel 5. Der Zitatblock im Quelltext

```
[quote, David Lloyd George, engl. Politiker (1863-1945)]
____
Fürchte Dich nicht vor einem großen Schritt.
Du kannst einen Abgrund nicht mit zwei kleinen Sprüngen überqueren.
____
```

Listings. Quelltexte jeder Art werden in der Regel unformatiert in Monospace Schrift gesetzt. Dafür gibt es Listing Blöcke. Zusätzlich wird ein Source Code Highlighting unterstützt falls ein entsprechendes externes Programm installiert ist. Listingblöcke werden durch vier Minuszeichen in einer Zeile begonnen und beendet. Vor dem Listing wird im Beispiel durch `[source, c]` spezifiziert, dass es sich um Quelltetxt handelt und welche Programmiersprache Verwendung findet.

```c
# include <stdio.h>
# include <stdlib.h>

int     main (int argc, char *argv[])
{
        printf ("Hello %s\n", getenv ("LOGNAME"));
        return 0;
}
```

Sidebar (Kasten). Eine Sidebar ist ein optisch abgehobener Kasten um inhaltlich abgeschlossene Teilaspekte des Textes außerhalb des Fliesstextes zu platzieren. In AsciiDoc ist ein Kastenabschnitt durch jeweils eine Zeile die vier Sternchen (Bullet) enthält eingeschlossen.

Ein Sidebarblock

Ein solcher Block darf alle AsciiDoc Elemente beeinhalten (außer Sidebar Blöcke selber). Zum Beispiel einen Listing Block:

```
$ asciidoc -a toc2 beispiel.txt
$ w3m beispiel.html
```

Kommentarblock. Die verbleibenden beiden Blockarten werden nicht im Text dargestellt. Zum einen handelt es sich um einen Kommentarblock. Er dient zum Auskommentieren von Teilen

des Textes oder eben zur Kommentierung selbigens. Kommentarblöcke sind durch eine Zeile die mindestens vier Schrägstriche enthält eingeschlossen.

PassThrough Block. Der Zweite Blocktyp dient zum Durchreichen von Steueranweisungen oder Inhalten an das Backendsystem. Diese Blöcke sind durch Zeilen die mindestens vier Pluszeichen enthalten eingeschlossen.

Offener Block. Dieser Block dient lediglich dazu Textabsätze zusammenzufassen. Er wird durch jweils eine Zeile mit zwei(!!) Minuszeichen begrenzt. Die hauptsächliche Anwendung besteht darin innerhalb von Listen, Absätze unter eine bestimmte Listenebene einzustufen.

2.5. Zusammenfassung Formatblöcke

Formatierungsblöcke sind jeweils durch eine bestimmte Zeile begrenzt.

Tabelle 2. Formatierungsblöcke

Blocktyp	Begrenzungszeile	Variablen	Callouts
Kommentar	`////`	nein	nein
Listing	`----`	nein	ja
Literal	`....`	nein	ja
Beispiel	`====`	ja	nein
Zitat	`____`	ja	nein
Textkasten	`****`	ja	nein
Passthrough	`++++`	ja	nein
Offener Block	`--`	ja	nein

2.6. Listen

In Asciidoc, wie auch in vielen anderen Textauszeichnungssprachen, gibt es drei Arten von Auflistungen: Ungeordnete- (nicht nummerierte), geordnete- (nummerierte) und Definitions-Listen. Zusätzlich existiert noch die Callout-Liste die im Zusammenhang mit Beispielen und Listings verwendung findet.

Ungeordnete Listen

Eine nichtgeordnete Liste ist über Sternchen (Bullet, `*`) und/oder Bindestrich (Dash, –) zu bilden.

Zweistufige Liste (Sternchen und Bindestrich)

- Listenelement mit Sternchen
  - Listenelement mit Bindestrich (nicht eingerückt)
  - Listenelement mit Bindestrich (eingerückt)
- Listenelement mit Sternchen

Beispiel 6. Quelltext einer zweistufigen Liste

```
.Zweistufige Liste (Sternchen und Bindestrich)
* Listenelement mit Sternchen
- Listenelement mit Bindestrich (nicht eingerückt)
  - Listenelement mit Bindestrich (eingerückt)
* Listenelement mit Sternchen
```

Zweistufige Liste (Bindestrich und Sternchen)

- Listenelement mit Bindestrich
  - Listenelement mit Sternchen (nicht eingerückt)
  - Listenelement mit Sternchen (eingerückt)
- Listenelement mit Bindestrich

Bei zweistufigen Listen ist es egal mit welchem Zeichen (Sternchen oder Bindestrich) man die Liste startet. Das jeweils andere Zeichen erhöht die Einrückung der Liste.

Mehr als zweistufige Listen lassen sich mit mehreren Sternchen in Folge erzeugen.

Mehrstufige Liste mit Sternchen

- Ein Sternchen
- Ein (in der Eingabe) eingerücktes Sternchen
  - Zwei Sternchen
    - Drei Sternchen
      - Vier Sternchen
    - Noch mal drei Sternchen
  - Zwei Sternchen
- Ein Sternchen

Wichtig

Generell kann man sagen, dass Einrückungen bei Listenelementen in der Quelle vielleicht die Lesbarkeit im Textdokument erhöhen, aber keine Auswirkungen auf die Tiefe der Liste haben.

Geordnete Listen

Geordnete Listen werden entweder automatisch von AsciiDoc nummeriert, oder man gibt die Nummern im Text an. Werden Nummern manuell gesetzt findet keine automatische Neunummerierung bei Änderungen statt. Als „Nummern“ sind sowohl Arabische, als auch Römische Zahlen erlaubt, genau wie Groß- und Kleinbuchstaben.

Es gibt sehr vielvältige Automatismen bei geordneten Auflistungen in AsciiDoc. Im folgenden wird genau eine generische Art der Definition eschrieben. Wer sich für die Details interessiert, sei auf den AsciiDoc UserGuide [https://asciidoc-py.github.io/userguide.html] verwiesen.

Geordnete Listen Elemente werden wie ungeordnete Listen formuliert, allerdings setzen man anstelle des Sternchens oder des Bindestrichs, einen Punkt. Bei manuell formatierten Listen auch das Nummerierungszeichen selber.

Falls die Liste nicht mit 1 beginnen soll, setzt man den Startwert über `[start=4]` **vor** der Liste. Die Art der Nummerierung wird durch den „Numberstyle“ bestimmt. Diesen schreibt man mit in die eckigen Klammern, wobei der Defaultwert arabische Zahlen sind. Die Schlüsselworte zur Kennzeichnung sind `arabic`, `lowerroman`, `upperroman`, `loweralpha` und `upperalpha`.

Mit Großbuchstaben nummerierte Liste, ab dem 4. Listenelement

D. Viertes Listenelement

E. Fünftes Listenelement

Mehrstufige nummerierte Listen. Werden ähnlich zu den ungeordneten Listen, mit mehreren Punkten vor dem Listenelement spezifiziert. Bei jeder Einrückungstiefe wird ein anderer Stil verwendet. Am einfachsten wird auch hier der Nummerierungsstil vorgegeben.

Beispiel 7. Quelltext einer mehrstufigen nummerierten Liste

```
.Ausgabe einer dreistufigen Liste
[arabic]
. Erstes Element
[loweralpha]
.. Erstes Element, erste Schachtelung (Kleinbuchstaben)
.. Zweites Element, erste Schachtelung
[lowerroman]
... Erstes Element, zweite Schachtelung (Kleine römische Ziffern)
... Zweites Element, zweite Schachtelung
.. Drittes Element, erste Schachtelung
   * Ungeordnete Liste innerhalb der nummerierten
   * Zweites Element der ungeordneten Liste
. Zweites Element, oberste Ebene
```

Ausgabe einer dreistufigen Liste

1. Erstes Element

    a. Erstes Element, erste Schachtelung (Kleinbuchstaben)

    b. Zweites Element, erste Schachtelung

        i. Erstes Element, zweite Schachtelung (Kleine römische Ziffern)

        ii. Zweites Element, zweite Schachtelung

    c. Drittes Element, erste Schachtelung

        - Ungeordnete Liste innerhalb der nummerierten

    - Zweites Element der ungeordneten Liste

2. Zweites Element, oberste Ebene

Definitions- oder Wort-Listen

Definitions- oder Wortlisten werden für Aufzählungen benutzt, bei denen die Listenelemente durch Worte eingeleitet werden. Ein gutes Beispiel dafür sind die Begriffsdefinitionen am Ende dieses Dokumentes. Formatiert werden die Definitionslisten über den Wortteil, gefolgt durch zwei Doppelpunkte. Anschliessend kommt der eingerückte Definitionstext.

Beispiel einer Definitionsliste für die Optionen des AsciiDoc Programms

`-a` *asciidoc-attribut*

AscciDoc Attribut zur Beeinflussung der generierten DocBook Datei. Die Option `-a` kann mehrfach angegeben werden.

`-d` *dokumenttyp*

Gibt den Typ des Eingabedokumentes an. Kann sein `article`, `book` oder `manpage`.

Bei kurzem Text, sieht eine Definitionsliste eleganter aus, wenn der Definitionstext in der gleichen Zeile beginnt. Durch ein `[horizontal]` vor der Definitionsliste ist dies zu erreichen.

-a Option a

-d Option d

Listenelemente mit mehreren Absätzen

Eine Leerzeile, als Ende eines Absatzes, beendet gleichzeitig auch das Listenelement und damit auch die entsprechende Formatierung. Das gleiche gilt für die bereits vorgestellten Blockelemente. Möchte man ein kurzes Listing z.B. als Element einer Liste formatieren, muss das Listenelement fortgeschrieben werden. Dies wir durch eine Zeile, die lediglich ein Pluszeichen beeinhaltet erreicht.

- Listenelement

  Mit einem zweiten Absatz

- Listenelement mit diesem Beispiel als eingebettetes Listing

```
* Listenelement
+
Mit einem zweiten Absatz

* Listenelement mit diesem Beispiel als eingebettetes Listing
+
----
!Achtung Rekursion!
----
+
```

```
 * Letztes Listenelement
```

- Letztes Listenelement

Callout Listen

Ein Callout Liste wird immer in Kombination mit einem Literal- oder Listingblock verwendet. In dem Listing werden nummerierte Markierungen gesetzt (sog. Callouts), die dann im Anschluss in Form einer Liste erläutert werden. Dies eignet sich besonders gut bei Quelltext oder zur Erläuterung von Kommandos oder ähnlichem.

Die Markierung erfolgt durch in Kleiner- und Größerzeichen eingschlossene Nummern (`<1>`) im Listing. Danach werden diese mit dem Erläuterungstext versehen aufgezählt.

Callout Liste (Quelltext).

```
====
        $ who           <1>
        $ who am I      <2>
====
<1>    Ausgabe wer auf dem Rechner angemeldet ist
<2>    Ausgabe wer man selber ist
```

Beispiel 8. Callout Liste (Ausgabe)

```
$ who           1
$ who am I      2
```

1 Ausgabe wer auf dem Rechner angemeldet ist
2 Ausgabe wer man selber ist

2.7. Tabellen

Die Formatierung von Tablellen ist sicherlich für alle Textverabeitungs Programme eine Herausforderung. Gibt es doch eine ausgesprochen große Menge an Formatierungsmöglichkeiten der Spaltenüberschriften, der einzelnen Zellen und der Tabelle als Ganzes.
Noch dazu soll die Eingabe einfach, und im Eingabetext als Tabelle erkennbar sein.

In AsciiDoc beginnt und endet eine Tabelle mit einer Zeile die einen senkrechten Strich (Pipe, `|`) gefolgt von drei (oder mehr) Gleichheitszeichen beeinhaltet. Wie bei den meisten anderen Gestaltungselement auch können **vor** dem Tabellenstart, in eckige Klammern eingeschlossen, die Attribute der Tabelle gesetzt werden.

Attribute können die Anzahl der Spalten sein, aber insbesondere Formatierungsanweisungen für die jeweilige Spalte. Dazu zählt z.B. links bzw. rechtsbündige Darstellung oder der Schrifttyp, z.B. Fett (Strong), Monospaced oder Kursiv (Emphasis). Ein Beispiel einer Tabelle mit drei Elementen (Wir nehmen das Optionsbeispiel von oben)

Option	Argument	Beschreibung
`-a`	*attribute*	Die Option -a kann mehrfach angegeben werden
`-d`	*doctype*	Der Typ des Dokumentes

Generiert wurde die Tabelle aus der folgenden Eingabe:

```
[cols="1>m,2^e,7<",width="70%",align="center",frame="topbot",options="header"]
|===
| Option| Argument      | Beschreibung
| -a    | attribute     | Die Option -a kann mehrfach angegeben werden
| -d    | doctype       | Der Typ des Dokumentes
|===
```

Entscheidend ist dabei die `cols=` Angabe. Für jeden durch Komma getrennten Formatstring wird eine Spalte angelegt.

Die Elemente darin werden entweder rechtsbündig (`>`), linksbündig (`<`) oder zentriert (`^`) ausgerichtet. Optional kann, durch Punkt getrennt, eine vertikale Ausrichtung spezifiziert werden. Das Kleinerzeichen steht für unten, das Größerzeichen für oben und das Caret-Zeichen (Zirkumflex) für mittige Darstellung innerhalb der Zelle.

Im Beispiel wird für die linke Spalte ein Monospace (`m`) Font und für die daneben liegende eine Kursivschrift (`e`) verwendet. Als Textauszeichnung gibt es noch Strong (`s`), Literal (`l`) bzw. Verse (`v`) und `h`, was zur selben Darstellung wie eine Headerzeile führt.

Normalerweise wird der gesamte verfügbare Raum (im obigem Beispiel 70% der Zeilenbreite) gleichmäßig auf alle Spalten aufgeteilt, was selten gut aussieht. Daher wurden die Spaltengrößen im Verhältnis zueinander festgelegt. Die Zahlen geben an, dass die letzte Spalte 7 Teile bekommt, die davor zwei Teile und die erste nur noch ein Teil der Gesamtbreite. Prozentuale Angaben sind gleichfalls möglich.

Durch `options="header"` wird die erste Zeile der Tabelle als Kopfzeile interpretiert und entsprechend hervorgehoben. Ohne diese Angabe würden alle Zeilen als Datenzeilen ausgewertet werden. Hier kann zusätzlich noch `footer` für eine entsprechende Fußzeile angegeben werden. Alle anderen möglichen Optionen funktionieren entweder nur im HTML Ausgabekontext oder nur bei Docbook (XSL-FO) Ausgabe und werden daher hier nicht weiter erläutert.

Spannend ist noch `grid=` zur Erzeugung der Linien zwischen allen Tabellenfeldern (default, `all`), respektive nur zwischen Zeilen (`rows`), Spalten (`cols`) oder eben gar nicht (`none`).
Durch `frame=` wird ein hervorgehobener Rahmen an allen Seiten (default, `all`) um die Tabelle gezogen. Alternativen sind nur oben und unten (`topbot`), links und rechts (`sides`) oder gar nicht (`none`).

Anmerkung

Alle Angaben hinter dem Gleichheitszeichen bei den Formatierungsangaben sind **immer** in Anführungszeichen zu schreiben. Ansonsten werden sie stillschweigend ignoriert, was die Autoren*innen mitunter zur Verzweiflung treiben könnte.

Tabellen Eingabefelder. Die Daten für die Spalten einer Tabelle werden im Normalfall durch ein Pipe-Symbol am Anfang des Feldes voneinander abgegrenzt. Diese Art der Abgrenzung wird als **P**refix **S**eparated **V**alues (`psv`) bezeichnet und sorgt für eine Tabellenähnliche Darstellung bereits im Eingabetext.

Die Datenseparierung kann auf **D**elimter **S**eparated **V**alues (`dsv`) umgestellt werden. Der Separator ist dann Trennzeichen *zwischen* Feldern. Das Standard Trennzeichen bei diesem Format ist ein Doppelpunkt oder Newline.

Als letztes gibt es noch `csv` (**C**omma **S**eparated **V**alues) welches gut geeignet ist um Excel Dateien direkt einzulesen.

Das Eingabeformat einer Tabelle wird über `format="psv|dsv|csv"` gesetzt.

Beispiel einer aus einer Datei eingelesenen Tabelle (Die Format Zeile ist hier umgebrochen was in der Eingabe nicht erlaubt ist).

```
.Die ersten 6 Einträge aus /etc/passwd
[format="dsv",cols="2<,1^,1>m,1>m,2<,3<m,5<m",align="center",
      width="90%",grid="none",frame="topbot",options="header"]
|===
User : PW : UID : GID : GCOS : Homedirectory : Shell
sys::[head -6 /etc/passwd]
|===
```

Tabelle 3. Die ersten 6 Einträge aus /etc/passwd

User	PW	UID	GID	GCOS	Homedirectory	Shell
root	x	0	0	root	/root	/bin/bash
daemon	x	1	1	daemon	/usr/sbin	/usr/sbin/nologin
bin	x	2	2	bin	/bin	/usr/sbin/nologin
sys	x	3	3	sys	/dev	/usr/sbin/nologin
sync	x	465534		sync	/bin	/bin/sync
games	x	5	60	games	/usr/games	/usr/sbin/nologin

3. Textliches Beiwerk

3.1. Links

Zu Links zählen alle Formen von URLs, also auch Email Adressen. Diese werden automatisch als aktive Links in den Text eingefügt, so sie denn als solche erkennbar sind. Für Web URLs muss dafür die komplette Syntax inkl. `http://` angegeben werden. Beispiel: http://www.heise.de funktioniert, `www.heise.de` nicht.

Für Mailadressen reicht hingegen die einfache Form hugo@example.de [mailto:hugo@example.de], nicht jedoch `mailto:hugo@example.de`. Auch wenn das merkwürdig ist, entspricht es doch unseren Gewohnheiten.

Erkannt werden `http` , `https`, `ftp`, `mailto`, und `file` URLs. Für das `mailto:` Beispiel (Hugo’s mail [mailto:hugo@example.de]) oben, muss jedoch die vollständige Syntax genutzt werden, d.h. mit angehängten eckigen Klammern `mailto:hugo@example.de[Hugo's mail]` zur Angabe des Textes im Dokument.

TODO: Interne Links & Querverweise erklären

3.2. Bilder

Das Einfügen fon Bildern wird im Abschnitt "Einfügen von Datei" beschrieben.

TODO: Querverweis einfügen!

3.3. Fussnoten

.

3.4. Indizes

.

3.5. Glossare

.

4. Dokumentkontrolle

Wer es bisher gewohnt war Dokumente in Word oder LibreOffice (oder meinetwegen auch in Pages) zu verfassen, hat dies meist innerhalb eines Dokumentes getan. Die Arbeit mit `groff(1)`, Latex und auch mit AsciiDoc erlaubt und unterstützt aktiv die Verwendung unterschiedlicher Eingabedateien, die beim Build-Prozess zu einer Ausgabedatei zusammengeführt werden.

So können die Kapitel eines längeren Dokuments in eigenen Dateien verfasst sein. Auch Grafiken und insbesonders Auszüge aus Programm Quelltexten liegen in der Regel in separaten Dateien vor. Um diese zu einem Dokument hinzuzufügen gibt es unterschiedliche Anweisungen.

4.1. Einfügen von Dateien

Das Include Makro fügt eine Datei an der Stelle im Dokument ein an der das Makro steht. Das Makro beginnt mit dem Schlüsselwort `include` am Anfang der Zeile, direkt gefolgt von zwei Doppelpunkten, gefolgt von dem Pfadnamen der einzufügenen Datei, gefolgt von eckigen Klammern, in denen Attribute angegeben werden können.

```
include::kapitel2.txt[]
```

Eine Variante des Makros, genannt `include1::`, fügt ebenfalls eine Datei an dieser Stelle ein. Allerdings wird dabei keine Ersetzung von Text oder eine ähnliche Bearbeitung durchgeführt. Die Datei wird sozusagen „roh“ eingefügt.

Include Attribute

- Mit `tabsize=<n>` wird die Expansion von Tabulatorzeichen in der eingefügten Datei festgelegt.
- Durch `depth=<n>` wird die maximale Schachtelungstiefe von Include Makros festgelegt. Der Standardwert ist 10.
- Mit dem `lines` Attribut wird ausgewählt welche Zeilen der Datei eingefügt werden. Das obige Beispiel mit den ersten 8 Zeilen der Unix *password* Datei wäre über `include::/etc/passwd[lines="1..8"]` zu erreichen gewesen. Es können auch mehrere Zeilenbereiche durch Komma getrennt spezifiziert werden.

Achtung

Leider funktioniert dies aktuell nicht. Warum ist noch unklar, evtl. ein Fehler im AsciiDoc Programm.

4.2. Einfügen von Kommandoausgaben

Anstatt einer Datei, kann auch die Ausgabe eines Kommandos in den Text übernommen werden. Das Makro dazu heißt `sys::[]` und bekommt in den eckigen Klammern das Kommando angegeben dessen Ausgabe im Text eingefügt werden soll.

```
sys::[date +"Q%q/%Y"]
Q1/2022
```

Tipp

Das `sys` Makro gibt es nicht nur in der oben angegeben Form um die Ausgabe innerhalb eines Blocks zu präsentieren, sondern auch als Inline Makro innerhalb des Textes. Über `{sys:date +"Q%q/%Y"}` (Q1/2022) kann die Ausgabe im laufenden Text dargestellt werden.

Das Makro `sys2::[]` dient dazu nicht nur die Standardausgabe, sondern zusätzlich auch die Standardfehlerausgabe im Text einzufügen.

4.3. Einfügen von Bildern

Bilder im PNG oder JPG Format werden mit dem `image::<datei>[<attribute>]` Makro in den Text eingefügt. Zwischen `image` und der Dateipfadangabe stehen zwei Doppelpunkte wenn das Bild in einen Block eingefügt werden soll, und lediglich ein Doppelpunkt wenn es ein Inline Image ist (siehe Beispiel unten).

Als <*attribute*> können, durch Komma getrennt, folgende Optionen gesetzt werden:

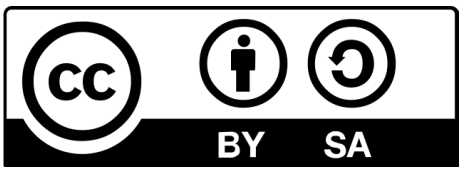
- Ein in Anführungszeichen eingeschlossener Text wird als Alt Attribute verwendet, d.h. falls die Bilddatei nicht dargestellt werden kann erscheint der angegebene Text.
- Ein Bildtitel in wird über das `title=` Attribut in den Text übernommen (auch als Mouseover Text).
- Höhe und Breite werden jeweils durch `height=` resp. `width=` Attribute spezifiziert. Die Einheit sind Pixel(!). Beispiel: 
- Für die DocBook Ausgabe erfüllt `scaledwidth=` einen ähnlichen Effekt. Allerdings handelt es sich bei dem Wert um eine Prozentangabe.
- Über `align=center|left|right` findet eine horizontale Ausrichtung des Bildes statt.
- Ein Link zu einem alternativen Bild wird über `link=` angegeben.

Beispiel Image. Die gleiche Bilddatei wie oben, nur jetzt freistehend, zentriert, etwas größer und mit Bildunterschrift.

```
image::640px-CC-BY-SA_icon_white.svg.png[height="{hsize}",
```

```
        align="center",title="CC-BY-SA Logo", alt="CC-BY-SA-Logo"]
```

Abbildung 1. CC-BY-SA Logo



4.4. Bedingtes Einfügen von Text

In obigen Beispiel sollte die Höhe der Grafik in Abhängigkeit des Ausgabegerätes spezifiziert werden. Daher wurde für das `height=` Attribute der Wert einer Variablen eingesetzt.

Diese Variable wurde ihrerseits in Abhängkeit der verwendeten Ausgabe (backend) entsprechend angepasst:

```
ifdef::backend-docbook,backend-docbook45,backend-docbook5[:hsize: 60]
ifdef::backend-html5,backend-xhtml11[:hsize: 100]
```

Mit `ifdef::` wird geprüft ob eine Variable definiert ist, wohingegen mit `ifndef::` auf das Gegenteil getestet wird.

Im Beispiel wird eine ganze Liste von Varablen auf „definiert“ abgefragt. Je nach Ergebnis wird dann der in eckige Klammern angegeben Teil ausgeführt.

Bei größeren Textblöcken, die in Abhängigkeit der Bedingung stehen, werden diese in `ifdef::`, `endif::` Zeilen eingeschlossen.

5. Einschränkungen & Vorbehalte

5.1. Fehlende Elemente und Funktionen

- Farbige oder unterstrichene Textauszeichnungen sind nicht nur merkwürdig in der Eingabe, sondern funktionieren nur bei der HTML Ausgabe.

- Es fehlt die Möglichkeit einen Textblock oder z.B. Tabellen in der Schriftgröße kleiner zu machen.

- Keine Möglichkeit ein Soft-Trennzeichen einzugeben um lange Worte zu trennen. Ok, dies funktioniert, leider abbhängig vom Backend: Das Softtrennzeichen ist `&shy;` und wird bei HTML Ausgabe und bei dem FOP Backend erkannt. Latex muss da leider passen.
  Der Trennalgorithmus sollte an die Sprache angepasst sein.
  Hierzu muss die Datei `fop-hyph.jar` aus dem Internet geladen [5] und in einem Verzeichnis deponiert werden. Der Pfadname der Datei wird dann in der Umgebungsvariablen `FOP_HYPHENATION_PATH` hinterlegt welche in der `fop(1)` Resource Control Datei `~/.foprc` gesetzt wird. CAUTION: Das Softrennzeichen wird von dem dblatex Backend als **nicht** valides zeichen angesehen.

[5] Siehe hierzu http://offo.sourceforge.net/

- Keine Worttrennungen bei HTML Ausgabe.

- Die „Revisionsliste“, die bei PDF Ausgabe erzeugt wird, enthält immer nur einen (den letzten) Eintrag. Möchte man tatsächlich eine Revisonsliste haben, muss man diese explizit über eine XML Datei (`*-docinfo.xml`) anlegen. In diesem Fall stören jedoch die Angaben im Header, da sie zusätzlich erzeugt werden.
  Eventuell ist es einfacher (und schöner) die Liste als Tabelle im Text zu definieren. Dann kann die Ausgabe leicht an Bedingungen geknüpft werden.

- Keine mehrspaltige Ausgabe.
  Dies ist in dieser Form nicht richtig. Eine mehrspaltige Ausgabe des gesammten Dokumentes kann mittels XSL-FO über den Parameter `column.count.body 2` erreicht werden.
  Allerdings fehlt eine Möglichkeit dies z.B. nur für eine Seite innerhalb des Dokumentes zu steuern.

5.2. Merkwürdigkeiten

- Die Ebenen bei geschachtelten Listen sind schwer festzulegen. Insbesondere bei Absätzen am Ende ist es nicht leicht zu sagen zu welcher Ebene er gehört.

  Tipp

  Eine Zeile mit Plus-Zeichen am Anfang verbindet zwei Absätze. Bereiche können in einem offenen Block zusammengefasst werden (Mit `--` Zeilen geklammert).

- Oben gesagtes gilt generell für Absätze innerhalb von Listenelementen. Dies kann über Fortsetzungszeichen (Pluszeichen in der Zeile) zwar geregelt werden, sieht allerdings in der Eingabe *unschön* aus da die Einrückung aufgehoben werden muss.

- Bei `sys::`, `include::` und `image::` Makros müssen zwei Doppelpunkte als Trenner angegeben werden. Bei inline Images (`image:`) jedoch nur einer. Soweit noch ok, aber bei *image* steht der Dateiname hinter dem Doppelpunkt und bei *include* in den eckigen Klammern. Das macht das Erlernen der Syntax nicht unbedingt einfacher.

- Bei der DocBook Druckaufbereitung über XSL-FO lässt sich eine Sortierung der Glossareinträge erreichen. Bei der HTML-Ausgabe funktioniert dies leider nicht.
  Hier ist man u.U. doch auf Skripting angewiesen, oder man lässt das Glossar in der HTML Ausgabe gleich weg.

- Ein Kapitel auf Ebene zwei oder höher mit der Überschrift „Zusammenfassung“ führt zu einer XML Fehlermeldung. Die Fehlermeldung hängt davon ab, wo das Kapitel steht und welche Ebene es hat.
  Strange!

Begriffsdefinitionen

Hard Line Break	Ein erzwungener Zeilenvorschub. Sorgt dafür, dass der nächste Satz in einer neuen Zeile beginnt.
em—Dash	Ein Minuszeichen der Breite des kleinen *m*. Dies wird, anders als ein Minuszeichen in der Eingabe, nicht als Trennzeichen (Bindestrich), sondern als Gedankenstrich verwendet. AsciiDoc setzt zwei Minuszeichen in der Eingabe automatisch als Gedankenstrich.
Casccading Style Sheets (CSS)	Es handelt sich hierbei um Formatierungsanweisungen für HTML5 Webseiten. Die Darstellung eines HTML Elementes wird daher durch das Style Sheet und nicht mehr durch das Endgerät (den Browser) bestimmt. Durch unterschiedliche Style Sheets kann eine Webseite gleichen Inhalts völlig anders dargestellt werden.
Plaintext	Reine Textdateien, meist mit der Dateikennung `.txt` angelegt. Früher bestanden diese Dateien ausschließlich aus 7-bit Ascii Zeichen. Später dann aus 8-bit Extended Asciizeichensätzen was zu dem Problem führte, dass man die Encodierung der Textdatei vorab kennen musste um zu einer korrekten Ausgabe zu kommen. Heutzutage nutzt man unter Unix Systemen für Textdateien in der Regel eine UTF-8 Encodierung.
Git	Siehe Versionsverwaltung.
Versionsverwaltung	Eine Versionverwaltung dient dazu mehrere Versionen eines Dokumentes platzsparend zu verwalten. Bei den meisten Versionsverwaltungssystemen wird neben dem Original Dokument, nur noch die Änderungen gespeichert, die notwendig sind um eine bestimmte Version zu erzeugen. Ausserdem wird durch eien Versionsverwaltung das gemeinsame Arbeiten an ein und demselben Dokument (meist handelt es sich um Progammcode) ermöglicht. Eine der ersten verfügbaren Versionsverwaltungen ist `SCCS`. Weitere bekannte Implementierungen sind `RCS` oder `Mercury`. Heute ist `git` sehr weit verbreitet.

Index

P

V