Tutorials schreiben

oder: Wenn sich Programmierer in Prosa versuchen

ein Tutorial von
Christian Rehn

29. September 2009

Inhaltsverzeichnis

1 Überblick

Es ist jetzt mittlerweile fast drei Jahre her, dass ich angefangen habe, Tutorials zu schreiben. Seit dem habe ich einiges gelernt. Insbesondere auch dadurch, dass ich mittlerweile fast schon regelmäßig fremde Tutorials korrekturlese. Dabei sind mir einige Dinge aufgefallen, die ich nun hier zusammenfassen möchte. Auch wenn ich meine eigenen bisherigen Tutorials wieder lese, so fällt mir einiges auf, das ich mittlerweile anders machen würde und bei so mancher Formulierung muss ich heute über meine damaligen Texte schmunzeln. Wer Lust hat, kann ja mal nach der Lektüre dieses Tutorials meine bisherigen Texte nach Fehlern und Verbesserungsmöglichkeiten durchsuchen. Es finden sich da so einige...

Ich kann nicht behaupten, dass ich eine besondere Qualifikation für so ein Tutorial vorweisen kann. Ich habe weder etwas dahingehendes studiert, noch Sekundärliteratur zum Thema gelesen. Das, was ich hier beschreibe, stützt sich ausschließlich auf meine „Erfahrungen“ mit eigenen und fremden Tutorials und ich denke, ich hätte den ein oder anderen Fehler nicht begangen, wenn mir jemand gesagt hatte, was man besser nicht macht. Nichtsdestotrotz bleibt es natürlich jedem selbst überlassen, was er von meinen Tipps hält. Kommentare jeglicher Art sind herzlich willkommen.

Dieses Tutorial richtet sich somit an alle, die gerne selbst mal ein Tutorial schreiben möchten. Hauptsächlich geht es hier um Programmier-Tutorials, jedoch gilt vieles von dem hier vorgestellten für jegliche Art Tutorial.

1.1 Zielgruppe

Ganz wichtig beim Tutorials schreiben – wie beim Schreiben von allen anderen Texten auch – ist es, die Zielgruppe zu beachten. Wer soll hinterher das Tutorial eigentlich lesen? Diese Frage ist viel elementarer, als sie auf den ersten Blick erscheint. Vorwissen, Interessen und Erwartungen der potentiellen Leser müssen berücksichtigt werden, wenn die Leserschaft weder über- noch unterfordert oder gar frustriert werden soll.

Bevor man also überhaupt mit dem Schreiben anfängt, sollte man sich also genau mit der zukünftigen Leserschaft auseinandersetzen. Es wird natürlich immer wieder Leser geben, die nicht zur gedachten Zielgruppe passen. Das ist nicht zu vermeiden, aber auch nicht weiter tragisch. Es geht hier mehr darum, die typischen Leser auszumachen. Das muss nicht zwangsweise eine ganz klar abgegrenzte Gruppe sein. Wenn man jedoch versucht ein Tutorial für Einsteiger *und* Profis, für Grundschüler, Studenten *und* Rentner zu schreiben, wird dies vermutlich schief gehen.

Die folgenden Fragen sollen helfen sich über die Zielgruppe klar zu werden:

- Welches Vorwissen bringen die Leser wohl mit?
- Allgemein: Welche Bildung haben sie? Sind es Schüler, Mathematikprofessoren oder Rentner?
- Konkreter: Was können sie nicht? Wo sind ihre Grenzen? Müssen viele (englische?) Fachbegriffe erläutert werden? Muss genauer auf mathematische Grundlagen eingegangen werden?
- Wie alt sind sie? Kinder müssen beispielsweise eher motiviert werden als Erwachsene.
- Welche Erwartungen haben sie an das Tutorial?
- Was wollen sie auf keinen Fall? Was würde sie langweilen? Was überfordern?

Wenn die Zielgruppe nicht klar ersichtlich, d. h. aus dem Thema ableitbar ist, ist es auch eine gute Idee am Anfang des Tutorials diese explizit zu benennen. Insbesondere können so erwartete Grundkenntnisse o. ä. erwähnt werden.

Oberste Grundregel: Beim Tutorials schreiben immer die Zielgruppe bedenken.

1.2 Arten von Tutorials

Im Grunde genommen gibt es drei Arten von Tutorials: Grundlagen-Tutorials, Anleitungen (auch bekannt als HowTos) und Projekt-Tutorials. Oftmals entspricht ein Tutorial genau einer dieser drei Arten. Es gibt allerdings auch diverse Mischformen.

Grundlagen-Tutorials sollen, wie der Name schon sagt, die Grundlagen zu einem bestimmten Thema vermitteln und dabei möglichst ausführlich und vollständig sein.

Anleitungen sind eher dazu da, um eine ganz konkrete Aufgabenstellung Schritt für Schritt zu lösen. Beispielsweise könnte ein HowTo erklären wie ein bestimmtes Programm zu installieren oder zu konfigurieren ist. Grundlagen werden hierbei größtenteils ausgeklammert. Das heißt nicht, dass man nicht erklären sollte, was passiert. Das ist wichtig für das Verständnis und für den Fall, dass sich die gegebene Situation des Lesers in Details von der des Tutorials unterscheidet. Beispielsweise aufgrund einer anderen Programmversion. Allerdings steht hier das konkrete Problem und dessen – schnelle – Lösung im Vordergrund.

Bei Projekt-Tutorials handelt es sich um eine Anleitung zum Einüben der Grundlagen an einer konkreten größeren Aufgabenstellung. „Wie schreibe ich einen HTML-Editor?“ fiele in diese Kategorie. So sind diese Art Tutorials meist aus einer Folge von Erklärungen, Aufgabenstellungen und Musterlösungen aufgebaut.

2 Schreiben

2.1 Schreibstil und Lesefluss

Zum Schreibstil ist erst einmal zu sagen, dass sich bei Tutorials die „wir“-Form tendenziell am besten liest. Also „Wir ziehen einen Button und ein Memo aufs Formular.“ statt „Man ziehe...“ (ein Tutorial ist kein Kochrezept), „Ziehe einen Button...“ bzw. „Ziehen Sie...“.

Ansonsten sollte jeglicher Selbstbezug vermieden werden. Im Vorwort kann auch mal ein „ich“ vorkommen, denn da geht es ja gerade darum, warum man dieses Tutorial schreibt. Auch auf Sätze wie „Jetzt erkläre ich...“ sollte verzichtet werden. Ebenso ist es mit „Im nächsten Kapitel...“ und ähnlichen Bezügen auf das Tutorial selbst.

Der Grund dafür ist, dass es den Leser wieder daran erinnert, dass er ja gerade ein Tutorial liest. Mit Tutorials ist es hier wie mit Büchern. Die besten sind so geschrieben, dass der Leser ganz darin versinkt.

Aus dem selben Grund sollte man den Leser auch nicht direkt anzusprechen. In Projekt-Tutorials ist es für die Aufgabenstellungen natürlich unumgänglich und auch richtig so – hier soll der Leser ja gerade aufhören zu lesen und die Aufgabenstellung bearbeiten. An vielen anderen Stellen ist das aber eher kontraproduktiv. Formulierungen wie „Der geneigte Leser möge dazu die einschlägigen Wikipedia-Artikel lesen.“ sind ein extremes Beispiel dafür, wie man es eher nicht machen sollte. Insbesondere weil der ein oder andere hier Ironie vermuten wird. Faustregel: Den Leser nur dann ansprechen, wenn man ihn explizit auffordern will, anzuhalten und selbst nachzudenken.

Zweite Grundregel: Immer den Lesefluss im Auge behalten. Selbstbezüge, direktes Ansprechen des Lesers und alles, was des Leser aus dem Lesefluss reißt, vermeiden bzw. auf die Stellen begrenzen, wo der Leser Pause machen *soll*.

Allgemein ist ein eher lockerer Schreibstil, der nicht zu trocken wirkt, jedoch auch nicht an der Ernsthaftigkeit des Textes zweifeln lässt, angebracht. Prinzipiell ist hier natürlich Vieles persönlicher Stil des Autors, wenngleich es nicht schaden kann, sich an anderen Tutorials zu orientieren.

2.2 Typographie

Tutorials bieten auch eine Gelegenheit, sich einmal mit Typographie auseinander zu setzen. So besteht z. B. ein Unterschied zwischen Absätzen und Zeilenumbrüchen, zwischen Bindestrichen und Gedankenstrichen („Halbgeviertstrichen“), etc. Es gibt sogar unterschiedliche Leerzeichen.

So gibt es vieles, was man typographisch „falsch“ macht, weil man sich dessen überhaupt nicht bewusst ist. Ein Beispiel dafür sind unterstrichene Überschriften. Überschriften werden sehr häufig unterstrichen dargestellt. Das liegt zum einen wohl daran, dass es einfach zu viele schlechte Vorbilder gibt und zum anderen, dass Textverarbeitungsprogramme die Unterstreichung scheinbar selbstverständlich mit Fett- und Kursivschrift in eine Reihe stellen. Tatsache ist aber, dass Unterstreichung gerade *kein* anerkanntes Mittel der Textauszeichnung ist. Nicht ohne Grund finden sich Unterstreichungen zu Hauf in den bunt bebilderten Überschriftensammlungen der Boulevardblätter, wohingegen in „seriösen“ Zeitungen darauf verzichtet wird. Faustregel: Überschriften Fett und größer, Hervorhebungen im Text durch Kursivschrift.

Um Texte typographisch einigermaßen richtig zu setzen – aber auch ganz unabhängig vom Gesichtspunkt der Typographie – bietet sich LaTeX[1] als Textsatzsystem an. Als IDE sind vor allem Kile[2] (für Linux; mittlerweile gibt es aber auch eine Windows-Portierung[3]) und TeXnic-Center[4] (für Windows) zu nennen, wobei Kile vom Funktionsumfang her wohl die Nase vorn hat. LaTeX ist auf jeden Fall mal einen Blick wert. Oder auch zwei...

3 Motivation

Kommen wir nun zum Punkt „Motivation“ also der Frage: „Wie motiviere ich meine Leser, weiter zu lesen?“. Das, was wir hierbei schon kennen gelernt haben, ist der Lesefluss. Jede Unterbrechung des Leseflusses bietet dem Leser eine ideale Möglichkeit das Tutorial langweilig zu finden und aufzuhören zu lesen.

Aber auch bei der Motivation der Leser gilt natürlich wieder, dass man je nach Zielgruppe unterschiedlich vorgehen muss. Insbesondere bei Anfängern sind z. B. schnelle sichtbare Resultate extrem wichtig. Wenn nicht binnen kürzester Zeit das Programm läuft, lesen sie erst gar nicht weiter, weil das Tutorial „zu trocken“ ist und verlieren womöglich ganz die Lust am Thema. Deshalb fängt auch so gut wie jedes Programmier-Tutorial für Anfänger mit einem Hello-World-Programm an. Schnelle Erfolge sind hier sehr wichtig.

Hat man aber eine andere Zielgruppe, beispielsweise Profis, die sich zu einem ganz bestimmten Thema informieren wollen, sind Klicki-Bunti-Beispiele eher weniger angebracht. Hier ist es wichtig die Anwendungsmöglichkeiten in den Vordergrund zu stellen und möglichst schnell zu

[1] http://de.wikipedia.org/wiki/LaTeX
[2] http://de.wikipedia.org/wiki/Kile
[3] http://windows.kde.org/
[4] http://de.wikipedia.org/wiki/TeXnicCenter

erklären, warum man sich mit dem jeweiligen Thema überhaupt beschäftigen soll. „Für was braucht man das?“ wäre hier also die zentrale Frage.

Auch die inhaltliche Herangehensweise hängt von der Zielgruppe ab. Der Leser wird besonders dann zum Weiterlesen animiert, wenn das Tutorial inhaltlich auf den Hintergrund des Lesers eingeht. Beispiel: Wenn man ein Programmiertutorial für Schüler schreibt, wird wohl erstmal Klicki-Bunti-Programmierung im Vordergrund stehen. Also eine grafische Oberfläche entwerfen und diese mit Leben füllen. Hat man aber stattdessen eine Horde Mathematik-Professoren, denen man das Programmieren näher bringen will, so ist es wohl sinnvoller mit den mathematischen Grundlagen der Programmierung anzufangen. Also konkret mit Ausdrücken und Funktionen.

So hängt also auch die Art, wie man den Leser motiviert, von der Zielgruppe ab.

4 Verständlichkeit

Einer der wichtigsten Aspekte beim Schreiben von Tutorials, wenn nicht sogar der wichtigste überhaupt, ist die Verständlichkeit. Ein unverständliches Tutorial ist nutzlos. Verständlichkeit definiert sich hierbei natürlich auch wieder über die Zielgruppe. Wenn das Tutorial zwar für Profis verständlich ist, jedoch Anfängerthemen behandelt, ist das Tutorial letztendlich unverständlich, denn Profis gehören wohl nicht zur Zielgruppe.

4.1 Überblick schaffen

Insbesondere bei Projekt-Tutorials ist es deshalb z. B. hilfreich, zuerst einmal einen groben Überblick zu geben. Das muss keine Sammlung von UML-Diagrammen sein, die der Leser womöglich eh nicht versteht, jedoch sollte man den Leser nie darüber im Unklaren lassen, was letztendlich der dahinter liegende Gedanke, das Konzept, ist. Der sprichwörtliche rote Faden muss eindeutig erkennbar sein. Bei einem Projekt-Tutorial trifft dies in doppelter Weise zu: Einmal für das Tutorial selbst und dann natürlich auch für das resultierende Ergebnis, also das „Projekt“. Nur so, kann der Leser problemlos folgen.

Ebenso kann es bei Grundlagen-Tutorials nicht schaden, anfangs einen Überblick über das Thema zu geben und die zentralen Punkte schon mal zu nennen. Ähnliches gilt auch bei HowTos, wobei sich hier der Überblick auf eins, zwei Sätze beschränken sollte.

4.2 Wiederholungen, Definitionen und Beispiele

Bei diesen grundlegenden Gedanken, gilt – ähnlich wie bei mündlichen Vorträgen: „Sag, was du sagen willst, sag es und sag, was du gesagt hast“. Nur, was wiederholt wird, bleibt auch langfristig im Gedächtnis. Besonders bei langen Tutorials, die oftmals in Etappen gelesen werden, ist es sehr wichtig die wichtigen Grundkonzepte ständig zu wiederholen. Dass der Leser „zurückblättern“ muss um etwas nochmals nach zu schlagen, ist eine eher zu vermeidende Situation. Der Leser gerät dadurch aus dem Lesefluss.

Wichtig: Nur, was wiederholt wird, bleibt auch langfristig im Gedächtnis.

Aber nicht nur grundlegende Gedanken müssen erläutert werden. Auch Begrifflichkeiten sollten unbedingt geklärt werden. Der Leser soll nicht noch Lexika wälzen müssen um das Tutorial zu verstehen. Auch hier ist natürlich wieder die Zielgruppe wichtig. Zu viele Erklärungen langweilen den Leser, zu wenige frustrieren ihn.

Man kann Begrifflichkeiten in einem separaten Abschnitt – vorzugsweise weit vorne – erklären und so eine grundlegende Basis schaffen, Glossare am Ende sind wohl eher etwas für sehr lange

Werke und dann besteht natürlich noch die Möglichkeit Begriffe bei der ersten Verwendung zu erklären. Keine dieser Herangehensweisen ist falsch, jedoch gilt auch hier, was für die Grundgedanken gilt: Nur, wenn sie wiederholt und wiederholt erklärt werden, merkt sich der Leser die Begriffe. Dabei sollte man natürlich vermeiden den Leser zu langweilen, weswegen es geschickt ist, Begriffsdefinitionen implizit zu wiederholen. Beispielsweise könnte in einem Satz der Fachbegriff verwendet werden und im darauf folgenden könnte man statt diesen zu wiederholen einfach die „Übersetzung“ des Begriffs verwenden. Somit wird die Begriffsdefinitionen wiederholt, ohne dass es der Leser explizit als solche wahrnimmt. Beispiel:

„Ein *Stack* (auch *Stapelspeicher* oder *Keller* genannt) ist eine Datenstruktur, die nach dem LiFo-Prinzip (engl. *last in first out*) arbeitet. Das, was man als letztes auf den Stapel gelegt hat, wird später als erstes wieder herunter genommen.“

Das war die Definition. Im Folgenden können die Begriffe Stack, Stapel, Keller und LiFo-Speicher abwechselnd verwendet werden. So wiederholt der Leser implizit die Begriffe, erkennt sie als Synonyme und merkt sich ihre Bedeutung. Würde im Folgenden nur von Stack gesprochen, hätte der Leser recht schnell vergessen, dass man auch Stapel oder Keller sagen kann und legt er das Tutorial mal beiseite und liest später weiter, hätte er womöglich auch vergessen, dass ein Stack nach dem LiFo-Prinzip arbeitet. Zudem ist es sprachlich geschickter Wortwiederholungen durch Synonyme zu umgehen.

Somit sind wir auch schon beim nächsten Punkt: bei den Beispielen. Beispiele sind fürs Verständnis essentiell wichtig. Oftmals wird der Fehler begangen nur eine formale Definition zu geben, jedoch keine Beispiele zu geben, die dem Leser veranschaulichen, ob er die Definition richtig verstanden hat. Zudem prägen sich Beispiele besser ein als trockene Definitionen. Man kann dies weiter unterstützen, indem man Beispiele interessant und lustig gestaltet und nicht ständig immer nur von „Foo“ und „Bar“ spricht. Dabei sollten die Beispiele immer noch einigermaßen natürlich sein. Macht man sie zu lustig und zu gekünstelt, so kann es passieren, dass der Leser sie nicht mehr ernst nimmt, d. h. nicht glaubt, dass das, was für das künstliche Beispiel gilt, auch für andere Situationen – eben solche in der Praxis – zutrifft.

Wichtig: Zum Verständnis sind immer Definitionen *und* Beispiele nötig.

Beispiele sind also sehr wichtig, jedoch ist es ebenso falsch sich ausschließlich auf Beispiele zu stützen. Allgemeine Definitionen und Erklärungen müssen ebenfalls sein, sonst muss diese der Leser erraten, was u. U. auch zu falschen Interpretationen führen kann. Zu einem guten Tutorial gehört immer beides: knallharte Definition und anschauliches Beispiel.

4.3 Hintergrundwissen, Aufgaben und Detailgrad

Neben Faktenwissen und Beispielen ist es auch wichtig, dem Leser etwas an Hintergrundwissen mitzugeben. Bei Grundlagen-Tutorials ist dieses Hintergrundwissen ja gerade der eigentliche Inhalt, aber auch bei HowTos sind kurze Erklärungen wichtig, damit der Leser bei etwaigen Problemen oder kleinen Änderungen der Situation – beispielsweise neue Programmversionen – schnell selbst eine Lösung erarbeiten kann. Entsprechendes gilt natürlich auch bei Projekt-Tutorials.

Bei eben solchen kommt allerdings noch hinzu, dass es hier sinnvoll ist, dem Leser Aufgaben zu stellen, die er lösen soll. Seine Ergebnisse kann er dann leicht mit dem vergleichen, was im Tutorial vorgestellt wird. Wie schon erläutert bieten sich solche Aufgaben an, den Leser wirklich einmal anzusprechen und so den Lesefluss zu unterbrechen, damit der Leser die Aufgabe bearbeiten kann. Beispiel:

Aufgabe: Untersuche, wie in Abschnitt 4.2 Definitionen und Beispiele verwendet werden.

Auch bei Grundlagen-Tutorials sind Aufgaben und Verständnisfragen angebracht. Es bietet sich an, solche am Ende eines Kapitels zu stellen. Lösungen sollten natürlich auch verfügbar sein. Wichtig ist hierbei aber natürlich, dass der Leser die Lösung nicht sofort schon sieht. Also hier sollte gerade der Lesefluss unterbrochen werden – etwa indem die Lösung erst durch Anklicken eines Links erscheint. Bei Tutorials in PDFs oder auf Papier ist eine separate Seite mit Lösungen im Anhang sinnvoll. Das auf den Kopf Stellen der Lösung, was man ja öfters in Zeitschriften sieht, ist bei Tutorials meist nicht die beste Lösung, weil diese ja meist am Rechner gelesen werden.

Zu den angesprochenen Hintergrundinfos gehört auch, dass alles, was verwendet wird und nicht als Grundwissen vorausgesetzt werden kann, erklärt werden muss. Dazu gehören insbesondere auch Methodenparameter und ähnlicher Kleinkram. Ansonsten muss der Leser nachschlagen, was den Lesefluss wieder unterbricht. Andere Leser werden einfach weiter lesen, haben dann aber womöglich einen Teil des Tutorials nicht richtig verstanden. Man darf natürlich auch nicht vom Hölzchen aufs Stöckchen kommen, aber das, was gezeigt wird, sollte der Leser schon verstehen können.

Aber auch das Thema selbst hat einen Einfluss darauf, was erklärt werden muss. So sollte das Tutorial wirklich alles Wichtige zum Thema abdecken. Das gilt hauptsächlich, aber nicht ausschließlich, für Grundlagen-Tutorials. Der Leser soll sich umfassend informiert fühlen. Passen bestimmte Informationen des Themas nicht ins Tutorial, so ist das ein Hinweis darauf, dass der Titel vielleicht unpassend ist. Dieser sollte nicht mehr und nicht weniger versprechen, als auch wirklich gehalten wird.

Wir haben ja schon festgestellt, dass man den Leser normalerweise nicht aus dem Lesefluss reißen soll. Dazu gehört auch, dass man Vorwärts-Referenzen vermeidet. Das heißt so etwas wie „Das wird später noch erklärt“ sollte vermieden werden. Stattdessen ist es besser den Sachverhalt direkt an Ort und Stelle zu erklären. Darauf kann später immer noch genauer eingegangen werden. Das hat auch noch den Vorteil, dass wieder etwas Wichtiges wiederholt wird. So kann sich das der Leser besser behalten.

Wenn der Leser aber auf später vertröstet wird, nützt ihm das momentan auch nichts und er wird unweigerlich zuerst einmal etwas nicht verstehen. Das ist unbefriedigend und fördert nicht gerade das Verständnis. Manchmal ist es recht schwer, ohne Vorwärts-Referenzen aus zu kommen, aber man sollte sie dennoch vermeiden, wo es geht.

4.4 Verständliche Sätze

Nicht nur Aufbau und Inhalt sollen die Verständlichkeit fördern, auch die Sprache, die einzelnen Sätze sollen verständlich sein. So sollten komplizierte Schachtelsätze, sowie Einschübe in Klammern und Gedankenstrichen eher vermieden werden. Auch Negationen können problematisch sein. Statt „Das ist nicht gut.“, sollte man also besser „Das ist schlechter Stil.“ schreiben. Das Wörtchen „nicht“ wird einfach relativ schnell überlesen, was den Leser leicht verwirren kann. Noch schlimmer sind doppelte Verneinungen. Wenn sich die positive Formulierung zu künstlich anhört, kann man zumindest die Negationen vor Fehlinterpretationen etwas „schützen“, indem man ein „aber“ dazu packt. So muss der Leser schon zwei Worte überlesen um den Satz falsch zu verstehen. Beispiel: „Das ist aber typographisch gesehen nicht so gut.“

4.5 Grafiken und Code

Text ist nicht das Einzige, was gute Tutorials enthalten. Oftmals sind noch Grafiken, Bilder, Screenshots und natürlich Code enthalten. Dabei muss man sich jeweils fragen, was Sinn und Zweck dessen ist. Letztendlich unterbricht das nämlich wieder den Lesefluss. Der Leser springt vom Text in die Grafik und wieder zurück. Deshalb sollten Grafiken nur dort eingesetzt werden, wo sie die Verständlichkeit erhöhen. Oftmals sind sie aber in solchen Fällen dann überaus hilfreich.

Bei Grafiken ist es wichtig, diese im Text zu erläutern und auch aktiv darauf Bezug zu nehmen. Der Leser soll nicht noch lange raten müssen, was genau gemeint ist und für was die Grafik überhaupt da ist. Ein Satz wie „Grafik 4.2 zeigt das Innere eines Regenwurms, wobei der Kopf auf der linken Seite abgebildet ist.“ macht die Grafik verständlicher und bietet eine definierte Stelle, an der der Leser sich die Grafik ansehen soll. Am besten sind solche Hinweise am Ende eines Abschnitts aufgehoben, weil dann das wiederfinden der Textstelle einfacher ist.

Auch Code sollte man nicht im Übermaß liefern. Nur so viel, wie zum Verständnis notwendig ist. Den vollständigen Code kann man ja beispielsweise zum Download anbieten. Der Detailgrad des Codes hängt dann auch wieder von Thema und Zielgruppe ab. Bei Anfängertutorials ist ein hoher Detailgrad und ausführliche Kommentierung wichtig. Auch sollte hier der Code unbedingt kompilierbar sein.

Ist das Thema abstrakter und die Zielgruppe fortgeschrittener, kann auch der Code abstrakter und grobschrittiger sein. Oftmals bietet sich auch Pseudocode an, da sich dieser besonders leicht im Detailgrad variieren lässt. So kann man wirklich nur das Notwendige zeigen und den programmiersprachlichen Ballast beiseite lassen. Man sollte sich immer fragen, was man konkret zeigen will.

Bei besonders langen Projekt-Tutorials kann es auch geschickt sein, in regelmäßigen Abschnitten „Code-Checkpoints“ einzuführen, d. h. dem Leser nicht nur den vollständigen Quellcode zum Download anzubieten, sondern auch in mehreren Zwischenstadien. So kann der Leser das, was er selbst programmiert hat, mit dem Code-Checkpoint, also quasi der „Musterlösung“, vergleichen. Zudem verhindert das, dass der Leser nicht mehr folgen kann, weil er etwas nicht ganz hinbekommen oder anderes gelöst hat. Wichtig ist hierbei dann natürlich wie im gesamten Tutorial, dass der Quellcode auch wirklich mustergültig ist.

4.6 Hervorhebungen

Manchmal möchte man besonders wichtige Stellen hervorheben. In einigen Tutorials wird das gemacht, indem einzelne Begriffe fett gedruckt werden. Das ist aber typographisch gesehen nicht so gut. Besser sind hier separat hervorgehobene Abschnitte am besten noch mit einem Symbol gekennzeichnet, wie hier im diesem Tutorial gezeigt. Das erleichtert die Wiederauffindbarkeit von wichtigen Infos.

Auch um Zusatzinformationen, Regeln, Tipps, Aufgaben und Lösungen hervorzuheben sind solche hervorgehobenen Abschnitte sinnvoll. Mit Fußnoten sollte man hingegen vorsichtig sein. Diese können, wieder den Lesefluss unterbrechen. Deshalb eignen sie sich mehr für Informationen, die fürs Lesen erstmal unwichtig sind, beispielsweise Links zu weiterführenden Informationen.

4.7 Vorbildfunktion

Wenn man ein Tutorial schreibt, so sollte man sich immer im Klaren darüber sein, dass einem der Leser meist einfach glauben wird. Und zwar mehr, als einem vielleicht lieb ist. Man wird für den Leser implizit zu einer Art „Autorität“. Man tut ja so, als wüsste man, wovon man schreibt. Deshalb ist es auch wichtig, dass man es auch wirklich weiß.

Im Allgemeinen ist es so, dass der Autor eines Tutorials am meisten dabei lernt. Er muss nämlich jede Formulierung genau überdenken, ob sie auch genau so richtig ist und demnach genau über das recherchieren, was er schreibt. Alles muss er selbst durchdenken. Ein Leser kann sich damit zufrieden geben, wenn er nur 95% versteht. Ein Autor nicht. Der Leser wird einem glauben und im Zweifel das Tutorial wörtlich nehmen, ja mehr noch: Der Leser liest auch zwischen den Zeilen. Deshalb sollte man mit Übertreibungen, Späßen und Ähnlichem – so schön das auch ist und so schön es auch den Text auflockert – sehr vorsichtig sein. Der Leser könnte da etwas herauslesen, was gar nicht so gemeint ist.

Nun ist ein Tutorial keine wissenschaftliche Arbeit, von daher sind Bibliographien, etc. nicht nötig, jedoch sollte alles, was man schreibt, Hand und Fuß haben. „Das stimmt so irgendwie oder so ungefähr“ reicht hier nicht.

Wichtig: Alle Beispiele, Definitionen und Aussagen müssen *für die Zielgruppe* mustergültig sein.

Auch hier ist es natürlich mal wieder wichtig, die Zielgruppe zu bedenken. Alle Beispiele, Definitionen und Aussagen müssen *für die Zielgruppe* mustergültig sein. In einem Anfängertutorial den Leser mit einem vollständigen objektorientierten Entwurf zu konfrontieren, ist wohl eher nicht sinnvoll. Hier muss der einigermaßen saubere Anfängeransatz als mustergültig angesehen werden. So bringt es nichts, abstrakte Klassen zu definieren, wenn man dem Leser gerade einmal beigebracht hat, was Funktionen sind.

Trotzdem sollte man keine groben Fahrlässigkeiten unkommentiert vorzeigen. So kann man z. B. nicht auf notwendige Ressourcenschutzblöcke verzichten, nur, weil man noch nicht erklärt hat, dass man Ressourcen auch wieder freigeben muss. Hier also besser vorbildlichen Code liefern und in eins, zwei Sätzen erläutern, was man da tut.

Ebenso ist es unumgänglich notwendig, dass Begriffe richtig verwendet werden. Wenn man Vereinfachungen annimmt, sollte man explizit darauf hinweisen, sonst wird das der Leser nicht erkennen. „Ich dachte das gilt ganz allgemein so. Im Tutorial wirds ja so gemacht.“ Zudem prägen sich falsche Begriffe recht schnell ein und Murphys Gesetz zufolge richtige fast überhaupt nicht. Besser also den Leser gleich an die richtigen Begriffe gewöhnen.

Tipp: Es kann hier auch sinnvoll sein, vermeintlich „bekannte“ Begriffe nochmals nach zu schlagen. Vielleicht haben diese ja eine leicht andere oder breitere Bedeutung.

5 Wie fängt man an?

5.1 Titel

Wie fängt man an? Nun, mit dem Titel natürlich. Der Titel eines Tutorials hat zweierlei Aufgaben. Erstens soll er neugierig machen und interessant klingen und zweitens soll er etwas über den Inhalt verraten. Klingt er langweilig, wird das Tutorial keinen Neugierig machen, der es nicht sowieso lesen wollte und wenn er nichts über den Inhalt verrät, macht sich der Leser womöglich falsche Vorstellungen und ist im schlimmsten Fall unnötigerweise enttäuscht.

Diese beiden Funktionen können beliebig auf Titel und Untertitel aufgeteilt werden. Nur sollte beides vorhanden sein.

5.2 Inhaltsverzeichnis

Wie aber fängt man nun inhaltlich an? Am besten mit dem Inhaltsverzeichnis. Um Struktur in seinen Text zu bekommen, sollte man sich frühestmöglich über diese Gedanken machen. So

ist es ratsam wirklich das Inhaltsverzeichnis zuerst zu erstellen. Dabei kann man ruhig sehr detailliert vorgehen und zu den einzelnen Abschnitten Stichpunkte, die den späteren Inhalt beschreiben, hinzufügen. Diese Stichpunkte können hinterher dann wieder gelöscht bzw. in LaTeX zu Kommentaren gemacht werden.

Ein Problem beim Tutorials Schreiben ist es nämlich auch beim Thema zu bleiben, aber auch nichts zu vergessen. Leicht passiert es, dass man sich in unwichtigen Details verliert, in andere Themen abdriftet oder Wichtiges einfach vergisst. Das Inhaltsverzeichnis und die Stichwörter können hier helfen. Sie zwingt einen dazu, beim Thema zu bleiben und verringert die Gefahr, dass man etwas vergisst.

Damit das auch funktioniert, ist es natürlich notwendig, dass die Überschriften „sprechend“ sind. Das kommt auch dem Leser zugute, denn so kann er sich leichter zurecht finden. So sollte etwa jeder Abschnitt von der Größe einer halben A4-Seite eine sprechende Überschrift bekommen. Eine detailliertere Gliederung ist unnötig und lässt im schlimmsten Fall den Text „abgehackt“ erscheinen; längere Abschnitte hingegen schrecken den Leser ab.

Faustregel: Alle halbe Seite eine Überschrift.

5.3 Die Einführung

Und wie fängt man den Text an? Manche Tutorials fangen einfach direkt an ohne irgendwelche einführenden Worte. Oftmals ist es jedoch besser, eine kleine Einführung ins Thema zu geben. Was hat einen dazu bewogen, das Tutorial überhaupt zu schreiben? Hier ist mehr oder weniger die einzige Stelle, an der ein „ich“ vollkommen OK ist. Hier kommt der Autor zu Wort.

Es gibt aber auch andere Möglichkeiten, wie man so eine Einführung gestalten kann. So könnte man z. B. auch mit einer Anekdote oder einem Zitat anfangen. Ziel ist es einfach den Leser nicht gleich ins kalte Wasser zu werfen, sondern ihm langsam das Thema näher zu bringen.

Tipp: Manchmal kann es einfacher sein, die Einführung zum Schluss zu schreiben, weil man sich dann schon ins Thema eingearbeitet hat.

6 Wie hört man auf?

Nachdem wir uns angesehen haben, wie man anfängt zu schreiben, stellt sich natürlich noch die Frage, wie man aufhört. Man kann natürlich „einfach so“ aufhören. Eine andere, etwas elegantere, Möglichkeit ist, quasi ein Gegenstück zur Einführung zu schreiben, vielleicht einen Ausblick auf andere Themen zu geben oder, im Falle eines Projekt-Tutorials, mögliche Erweiterungen des Projektes aufzuzeigen.

Hat man dann das Tutorial dann endlich fertig gestellt, strotzt es meist noch vor Fehlern. Das ist vollkommen normal und kein Grund zur Beunruhigung. Man sollte es am besten mehrmals selbst lesen, Tippfehler und Formulierungen ausbessern etc. Dadurch, dass man mit dem Inhaltsverzeichnis angefangen hat, kann es auch passiert sein, dass die einzelnen Teile zusammenhanglos wirken. Oft fehlen Überleitungen zwischen den einzelnen Themen. Auch das sollte in diesem Schritt korrigiert werden.

Da man aber selbst meist die eigenen Fehler nicht so leicht bemerkt, ist es sehr sinnvoll das Tutorial jemand anderem zum Kontrolllesen vorzulegen. Meist kommen da noch die ein- oder anderen Schwächen zu Tage. Auch solche, die hier vielleicht nicht angesprochen wurden...

7 Anhang: Checkliste

Die folgende Checkliste soll helfen, beliebte Schwachstellen im eigenen Tutorial zu finden. Das ist keine Garantie für ein perfektes Tutorial und umgekehrt muss auch nicht alles „falsch“ sein, nur, weil es hier als „Fehler“ klassifiziert wird. Diese „Regeln“ hier sind keineswegs in Stein gemeißelt, sondern sollen ausschließlich als Orientierung dienen.

7.1 Allgemein

1. Ist die Zielgruppe klar definiert und passt sie zum Thema des Tutorials?
2. Wird die Zielgruppe über- oder unterfordert? Beispielsweise durch zu viele oder zu wenige Erklärungen?
3. Um welche Art Tutorial handelt es sich? Grundlagen-Tutorial, HowTo, Projekt-Tutorial oder eine Mischform? Passt das zum Thema?

7.2 Schreibstil und Lesefluss

1. Wird überall die „wir“-Form verwendet? Konkret: Taucht irgendwo ein „du“ oder „Sie“ auf? Oder ein „ich“ an einer anderen Stelle, als in der Einleitung?
2. Sind Selbstbezüge wie „im nächsten/vorherigen *Kapitel*“ vorhanden? Ein „Wie schon erläutert“ das eine Wiederholung kennzeichnet, ist dabei kein Problem.
3. Wird der Leser nur dann angesprochen, wenn er explizit aufgefordert werden soll, beim Lesen inne zu halten und sich etwas zu überlegen?
4. Ist der Schreibstil zu trocken? Wirkt das Tutorial langweilig?
5. Ist der Schreibstil zu locker? Lässt er an der Ernsthaftigkeit des Textes zweifeln?

7.3 Motivation

1. Wird die Zielgruppe zum Weiterlesen motiviert? Wird klar, warum man das Tutorial lesen soll?
2. Geht das Tutorial inhaltlich auf die Zielgruppe ein?
3. Wenn es ein Anfänger-Tutorial ist: Werden schnell sichtbare Resultate präsentiert?

7.4 Verständlichkeit

1. Ist das Tutorial *für die Zielgruppe* verständlich?
2. Wird dem Leser ein Überblick über das Thema, die Herangehensweise bzw. die Idee gegeben? Ist der „rote Faden“ erkennbar?
3. Werden die wichtigen Inhalte wiederholt?
4. Werden alle für die Zielgruppe unbekannten Begriffe erklärt?
5. Werden diese Begriffe, sofern sie wichtig sind, wiederholt?
6. Existieren immer jeweils Definitionen *und* Beispiele?

7. Sind die Beispiele „natürlich“ oder wird immer nur von „Foo“ und „Bar“ gesprochen? Kann man den Beispielen glauben, dass sie auch in der Praxis funktionieren?
8. Werden nötige Hintergrundinformationen geliefert?
9. Bei HowTos: Kann der Leser das Tutorial verwenden, wenn seine Situation leicht anders ist?
10. Bei Projekt- und Grundlagen-Tutorials: Sind Aufgaben für den Leser vorhanden? In ausreichender Zahl?
11. Deckt das Tutorial alles, was zum Thema gehört, ab? Hält das Tutorial, was der Titel verspricht?
12. Werden Vorwärts-Referenzen vermieden?
13. Sind die einzelnen Sätze verständlich? Oder sind sie vielleicht zu lang und zu kompliziert? Werden Negationen vermieden?
14. Werden Grafiken und Code sinnvoll eingesetzt? Nimmt der Text Bezug darauf?
15. Bei großen Projekt-Tutorials: Gibt es „Code-Checkpoints“? Genügend?
16. Sind alle Definitionen, Beispiele, Codes und Aussagen „mustergültig“ (für die Zielgruppe)? Oder stimmen sie vielleicht nur „ungefähr“?
17. Werden Begriffe richtig verwendet?

7.5 Aufbau

1. Macht der Titel neugierig? Beschreibt er das Thema genau?
2. Sind genügend Abschnitte vorhanden? Aber auch nicht zu viele? Faustregel: Eine Überschrift für jede halbe Seite.
3. Ist der Text durch Absätze gegliedert?
4. Gibt es eine Einführung ins Thema?
5. Gibt es einen „Ausblick“ oder ein „Fazit“ am Ende? Oder wird das Tutorial auf andere Weise abgeschlossen?