

読み手につたわる文章が書けるテ クニカルライティング

仮称

mochikoAsTech 著

2024-05-25 版 **mochikoAsTech 発行**

はじめに

2024 年 5 月 mochikoAsTech

この本を手に取ってくださったあなた、こんにちは、あるいははじめまして。「読み手につたわる文章が書けるテクニカルライティング」の筆者、mochikoAsTech です。

突然ですが……私たちは、毎日のように誰かに何かを説明したり、説明してもらったりしています。

- Slack で他部署の人に仕様の疑問点を伝えて質問する
- 業務でつまづいている部分を日報に書いて報告する
- 子供に LINE で今週末のお出かけスケジュールを送る
- 薬剤師が処方薬の飲み方や注意事項を紙で見せながら説明する

こんなふうに他の人に何かを理解してもらうことや、情報を齟齬なく伝達することを目的として書かれた実用的な文章のことを**実用文**と呼びます。そしてこの実用文を、相手の理解度や状況に合わせて分かりやすく書く技術が**テクニカルライティング**です。

エンジニアリングの技術を学んで、システムを開発する人たちが「エンジニア」であるように、テクニカルライティングの技術を学んで、分かりやすい実用文を書くことを生業としている人たちは「**テクニカルライター**」です。

厚生労働省による職業情報提供サイト^{*1}では、テクニカルライターは別名「マニュアルライター」とも紹介されており、もともとは家電や電化製品を買うと付いてくる、マニュアルや取扱説明書を書いている人たちがテクニカルライターと呼ばれていました。近年はそこから転じて、IT 系の企業において開発ドキュメントや API リファレンスなどを書くテクニカルライターや、ウェブサービスの UI 文言などを担当する UX ライターといったポジションの募集も増えてきています。

さらに IT 系の中でも、一概にテクニカルライターと言ってもフロントエンドエンジニアやサポートエンジニア、QA エンジニアのような「技術職」の 1 つとして扱わ

^{*1} 職業情報提供サイト - 厚生労働省 <https://shigoto.mhlw.go.jp/User/Occupation/Detail/>
358

れ、がつり開発者向けのドキュメントを書くポジションもあれば、サービスを使うエンドユーザー向けのマニュアルを書くため技術知識はそこまで問われないというポジションもあります。またガジェット系の新製品の記事を書くライターが、テックにまつわることを書くライターという意味でテクニカルライターという名前で募集されているのを見かけることもあります。

筆者はもともと Web 制作会社でインフラエンジニアをしてましたが、2020 年から IT 系の企業でこのテクニカルライターというポジションに就いて仕事をしています。ジョブチェンジをした当時はテクニカルライターを抱えている企業は片手で数えるほどでしたが、やはりドキュメントやマニュアルを作る部分で課題を抱えている企業が多くったのか、近年はスタートアップでも「1 人目エンジニア」を採用するときと同じ感覚で「1 人目テクニカルライター」を探している、という話をよく聞くようになりました。

テクニカルライターがいれば、ドキュメントを書く部分はお任せしてエンジニアは開発に専念できますが、きっとまだそうでない会社の方が多いはずです。エンジニアのみなさん、「ちゃんとドキュメントに書き残すべきだ」という気持ちはあっても、どうしても開発だけが先行してドキュメントは置いてけぼりになってしまったり、せっかく書いたのにどうもイマイチ分かりにくくて結局質問しにくる人が絶えなかったり、そんなつらい思いをしていませんか？

文章がうまくかけるようになりたい！ ごちゃつきがちな情報をスッキリ分類整理して伝えられるようになりたい！ 相手の理解度に合わせていい感じに文章で説明できるようになりたい！ 本書は、そんなあなたに向けて書かれた本です。

いわゆる「文章の書き方」を解説した入門書でよく見かける文法の話だけではなく、実際に書いていて悩みがちな部分にスポットを当てて、できるだけ実践的な内容をお伝えしていきたいと思います。

なお本書には、2019 年 4 月に発刊した「技術をつたえるテクニック～分かりやすい書き方・話し方～」の内容を一部改訂して再掲したものが含まれています。

想定する読者層

本書は、こんな人に向けて書かれています。

- 技術書や技術記事を書いている人
- 技術書の翻訳をしている人
- 文章がうまく書けるようになりたい人
- 英語ができないけど英語でドキュメントを書いている人
- ちゃんとドキュメントを残したい人
- ごちゃついた情報をスッキリ整理したい人

-
- どこに何が書いてあるのか一目で分かるような階層構造を作りたい人
 - 相手の理解度にあわせた説明ができるようになりたい人

マッチしない読者層

本書は、こんな人が読むと恐らく「not for me だった…（私向けじゃなかった）」となります。

- 魅力的な小説や詩文を書けるようになりたい人
- 広告やキャッチコピーが書けるようになりたい人
- 法務文書を書くときのコツが知りたい人

本書のゴール

本書を読み終わると、このような状態になっています。

- 文章が上手に書ける
- 悩みすぎて筆が進まない状態から早めに脱出できる
- レビュイーが受け止めやすい形で文章のレビューができる
- 英語ができないのに英語を書かなければいけないときになんとか頑張れる

免責事項

本書に記載された社名、製品名およびサービス名は、各社の登録商標または商標です。

本書に記載されている内容は筆者の所属する組織の公式見解ではありません。

また本書はできるだけ正確を期すように努めましたが、筆者が内容を保証するものではありません。よって本書の記載内容に基づいて読者が行なった行為、及び読者が被った損害について筆者は何ら責任を負うものではありません。

不正確あるいは誤認と思われる箇所がありましたら、必要に応じて適宜改訂を行いますので GitHub の Issue や Pull Request で筆者までお知らせいただけますと幸いです。

<https://github.com/mochikoAsTech/technical-writing-book>

目次

はじめに	3
想定する読者層	4
マッチしない読者層	5
本書のゴール	5
免責事項	5
第1章 「分かりやすい文章を書く」とはどういうことか？	11
1.1 テクニカルライティングとは	12
1.1.1 「文章を書くこと」と「テクニカルライティング」の違い	12
1.2 「知らない」と書けない	13
1.2.1 自転車の絵を描いてみよう	13
1.2.2 自転車を「知らない」と描けない	16
1.2.3 うまく書けないときに足りないのは文章力か技術力か	17
1.3 初心者はいっぱい読んでいっぱい書こう	18
1.4 完璧主義よりも完成主義	19
【コラム】どうやったらテクニカルライターになれますか？	19
第2章 エンジニアのためのテクニカルライティング	21
2.1 関係ない人を早めにふるい落とそう	21
2.2 読者層をはっきりさせよう	23
2.3 想定する読者層とゴールを書いておこう	23
2.4 誰に何をしてほしいのかを書こう	24
2.5 先に大枠を説明して段々詳しくしていこう	25
2.6 既知から未知に繋ごう	25
2.7 一文の長さは一口で食べられる量にしよう	25
2.8 素早く近づける	27
2.9 期待値コントロールをしよう	27
2.10 要約文ではじめよう	27

目次

2.11	1つの段落では1つのことを話そう	27
2.12	一度に把握できることは7つまで	27
2.13	まずは「分かりやすい一文」を書けるようになろう	27
2.14	分かりやすい一文を見つけるために	27
2.15	漁ぎ始めを生成AIにサポートしてもらおう	28
2.16	意味のない空白や意味のないスペースを残すな	28
2.17	単語で終わらせない	28
2.18	文章と修飾を分けて書こう	28
2.19	再利用しやすいテキストにしよう	28
2.20	並列をナカグロで書くと、後の変更で箇条書きにしたとき見た目が 変になる	29
2.21	語順を入れ替えよう	29
2.22	修飾語はかかる言葉に近づけよう	29
2.23	タイトルは「概要」にすべきか、「○○の概要」にすべきか	29
2.24	分からぬから説明を読むが、分からぬと読めない	29
2.25	年月日や対象バージョンを書いておこう	30
2.25.1	例示用のIPアドレスやドメインを使おう	30
2.26	同じものは同じ名前で呼ぼう	31
2.27	正しい名前で呼ぼう	31
2.28	箇条書きを挟んだ文章を作らない	32
2.29	主語と述語を対応させよう	32
2.30	「自分」という主語に注意しよう	32
2.31	ひらがなに開こう	33
2.32	分かったと分からぬの両方の気持ちが必要	33
2.33	リンク名称を「こちら」にしない	33
2.34	一文の中で同じことを二度言わない	33
2.35	時刻の表記はJSTか	33
2.36	その「文字数」って何文字ですか？	33
2.37	以上と以下か、より大きいと未満か	34
2.38	「教えてあげる」ことに醉わないこと	34
2.39	そこにあることを気付いて辿り着いてもらわないと意味がない	34
2.40	変わっていく「語感」を捨て置かずに拾おう	34
2.41	見つけやすくて取り扱いやすいドキュメントにしよう	35
2.42	リンクは「張る」ものか「貼る」ものか	35
2.43	「無駄なことをしたくない」から何もしないと何も書けない	36
2.44	いただくはだいたい「くださる」にできる	36
2.45	「しましょう」はLet's do it togetherかYou should do itか	36

第3章	英語を書くとか翻訳するとか	37
3.1	単語と単語の間にはスペースが1つ必要	37
3.2	単語や文の単位で翻訳すると危ない	38
3.3	前後の文脈やどこで使われるのかを知らずに翻訳はできない	39
3.4	翻訳はもとの言いたいことに立ち返って考える	40
3.5	技術文書の翻訳に必須なのは英語力や文章力よりも技術力	41
3.6	スペルミスが心配なら ATOK に頼ろう	42
3.7	例は「ex.」ではなく「e.g.」	43
3.8	機械翻訳があれば英語がまったく分からなくても訳せるか	43
3.9	翻訳しづらい文章をやめよう	44
第4章	誤解を生まないテキストコミュニケーション	45
4.1	まずは「何の話か」を先に言う	45
4.2	背景も添えて話そう	45
4.3	「聞いてほしい」のぶつかり合い	45
4.4	Slack でのコミュニケーション	45
4.5	チケットの書き方	45
4.6	チケットのサイズはできるだけ小さくする	46
4.7	見せて選んでもらう	46
4.8	いっぱい書いて Working Out Loud な働き方をしよう	46
第5章	レビューする、レビューされる	47
5.1	文章がひどいと突っ込みが多すぎて読むに堪えない	47
5.2	指摘は「後出しじやんけん」だと心得よ	47
5.3	他人が書いたものに敬意を払おう	47
5.4	頼まれていないレビューは MUST FIX でないかぎりしない	47
5.5	レビュー者が気付いたことに、書いたとき気付かないのは当然	48
5.6	レビューコメントには重要度を添えよう	48
5.7	お願いしたい観点を添えてレビューを依頼しよう	49
5.8	大量に赤を入れることがレビューの存在意義ではない	49
5.9	レビューは『相手のやることを増やす』責任を持って言おう	50
5.10	指摘は素直に受け入れる	50
第6章	どうやって学ぶか	51
6.1	テクニカルライティングの検定を受けてみよう	51
6.2	色んな本を読もう	52
6.3	「書き方」の指標を見つける	52

目次

6.4	Technical Writing Meetup に参加しよう	52
あとがき		53
PDF 版のダウンロード	54	
Special Thanks:	54	
レビュアー	54	
参考文献	54	
著者紹介		55

第1章

「分かりやすい文章を書く」とは どういうことか？

具体的な「分かりやすい文章の書き方」を学ぶ前に、まずは「分かりやすい文章を書く」とはどういうことなのか考えてみましょう。

1.1 テクニカルライティングとは

「はじめに」に書いたとおり、テクニカルライティングとは、実用文を分かりやすく書くための技術です。

ですが日本語ネイティブであれば、もともと誰でもある程度の日本語は書けるはずです。誰にでもできる「文章を書くこと」と、「テクニカルライティング」はいったい何が違うのでしょうか？

1.1.1 「文章を書くこと」と「テクニカルライティング」の違い

文章を書くこととテクニカルライティングの違いを知るため、まずは**実用文とは何なのか**を掘り下げていきましょう。実用文は、以下のような特徴を備えています。

- 対象となる物事について説明している
- 論理的である
- 誤解を生まない
- 簡潔で明快である
- 読み手の行動を促す
- 成果物が文章である

つまり実用文とは、**対象となる『何か』について、論理的かつ簡潔明快に説明をしており、それを読むことで何をどうすればいいのかが分かるような文章**ということですね。

文章を書くというと、どうしても人の心を打つようなエモい文章や技巧的な文章、凝った小難しい文章を書ける高い文章力が必要なのでは、という方向で考えてしまいがちですが、小説や詩文とは違って、実用文においては前述のとおり「論理的かつ簡潔明快」が正義なので「流れるような文体」や「文学の香り」といったものは必要ありません。

また学生時代に宿題の読書感想文で400字詰めの原稿用紙2枚以上を求められ、大して書くこともないのになんとか長く引き延ばそうとして苦しんだ経験がある人も多いと思いますが、実用文においては「長ければ長いほどいい」という尺度は存在しません。読んだ結果、得られる情報が同じならば、無駄に長々しい文章よりも、短い文章でサクッと理解できた方が圧倒的にいいはずです。

実用文は小説や詩文とは異なる、という話は理解しやすいと思います。その一方で、一見すると実用文のようですが、テクニカルライティングの技術だけでは書けないものもあります。それが次の2つです。

- 法務文書
 - 簡潔で明快な表現よりも、論理的な整合性を優先して書かれるため、テクニカルライティングとは正解が異なります。
- 広告やキャッチコピー
 - 読み手に強烈な印象を残して購買に繋げることを重視するため、意図的に間違った言い回しや、誤読させる表現を用いるなど、テクニカルライティングでは「やってはいけない」とされていることが正解になり得ます。

テクニカルライターという職業の名前だけを聞くと、なんとなく「利用規約をレビューしてもらう」「この商品をうまく説明するキャッチャーな見出しを書いてもらう」といったことも頼めるような気がしてしまいますが、法務文書や広告は他の実用文と善し悪しの物差しが異なります。そのため、よかれと思って直したことでかえって法的な誤りを仕込んでしまったり、簡潔な言い方にしたことで印象に残らない広告にしてしまったりする可能性があります。

食を楽しみたい人にとっては美味しい食事が「いい食事」ですが、体を絞って筋肉をつけたい人にとっては脂質や糖質を抑えてタンパク質を効率よく取れる食事が「いい食事」です。つまりどんなシチュエーションでも、どんな人にとっても 100 点満点な「いい文章」というものは存在せず、「いい文章」の定義も目的によって異なるということです。

1.2 「知らない」と書けない

さて、話は変わりますが「自転車」を知っていますか？

何を唐突にと思われるかもしれません、自転車を知っているかと問われたら、恐らくあなたを含め、大半の人が「知っている」と答えるはずです。

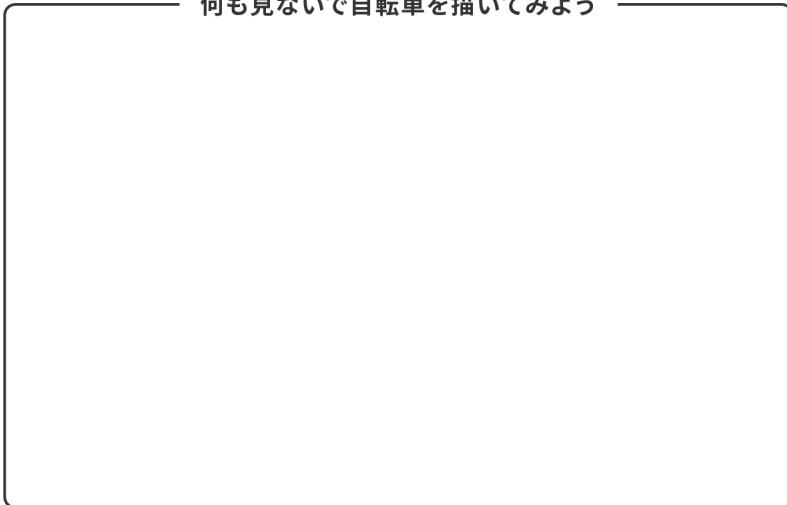
殆ど人は自転車というものを知っています。乗ったこともあるし、日々見かけるし、子供乗せ自転車^{*1}とかロードバイクとか色んな種類があることも知っています。自転車は英語だと bicycle だということをきっと知っているはずです。

1.2.1 自転車の絵を描いてみよう

ではちょっと、自転車の絵を描いてみましょう。あなたがよく知っているはずの、自転車の姿を頭に思い浮かべて、下手でもいいので自転車の絵を描いてみてください。(図 1.1)

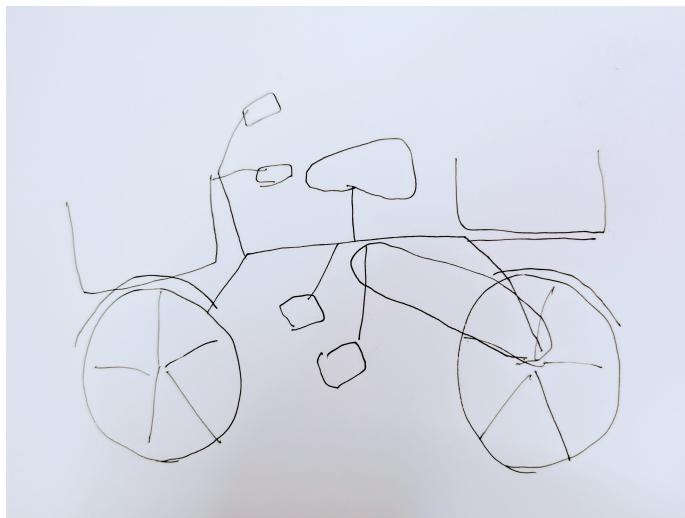
^{*1} 電動アシストの子供乗せ自転車は、うっかり充電が切れると漕ぐのがめちゃくちゃ重くてつらいことも知っている……知っているのに、なぜ人は充電を忘れてしまうのか……。

— 何も見ないで自転車を描いてみよう —



▲図 1.1 自転車の絵を描いてみよう

筆者もあなたと一緒に自転車を描いてみます。自転車にはタイヤがありますよね……大きいタイヤが 2 つ……そしてその 2 つの間になんか太いパイプみたいなのがあったはず……サドルもありますね、座るところです。あとなんか漕ぐところがある……ペダル……？ あれ、ペダル？ ペダルってどうやって回ってたでしたっけ？ 回る機構が必要なはず……どうなってましたっけ？ チェーン？ とタイヤが繋がってるはずですよね。チェーンって前輪と後輪どっちと繋がってましたっけ？ 後輪かな……あとハンドル！ ハンドルがあります！ カゴもあります……カゴってどこについてた？ あー、あとなんかタイヤの上に泥よけがあったかもしれません。我が家の中の自転車は前だけじゃなく後ろにもカゴが……あれ？ 自転車ってこんなでしだっけ？ なんか変だな。自転車って……どんな形でしたっけ？？（図 1.2）



▲図 1.2 筆者が描いた変な生き物みたいな自転車の絵

ばばーん！ そしてこんな感じの自転車の絵が生まれました。うまく描けましたか？ 筆者はうまく描けませんでした。

もともと自転車の構造をよく理解している人や、絵を描くことを生業にしていて普段から自転車を含め色々なものをよく観察している人でなければ、恐らくはこんな感じの「えー、自転車ってこんなだっけ？」という絵が生まれたことだと思います。

さて、正しい自転車の絵がこちらです。（図 1.3）



▲図1.3 正しい自転車の絵

見慣れた自転車の姿ですね。さて「自転車がどんな見た目で、どんな乗り物かなんて知っている」と私たちは思っていたはずです。でも実際は、自転車がどんな構造で、どういう理屈でできているのかという知識がないので、自転車の絵は思っていたよりうまく描けませんでした。

1.2.2 自転車を「知らない」と描けない

少し自転車の構造について解説していきます。

まず、自転車の骨組みは「ダイヤモンドフレーム」とも呼ばれており、どんな方向からの力に対しても丈夫、かつ低重心で操縦しやすい構造にするため、三角形を2つ組み合わせた斜めの菱形のような形をしています。このことを知っていれば、まず中心部に菱形の骨組みが描けます。

次に自転車に乗っているときの自分の様子を思い浮かべてみましょう。通常、あなたが椅子に座ったとき、膝はお尻の位置よりも前にあるでしょうか？それともお尻の真下にあるでしょうか？人体の構造上、椅子に座ったとき、膝は90度に曲がった状態でお尻よりも前方にあるはずです。このことから、ペダルの根元にある丸い部分、フロントギアはサドルよりも位置が少し前に来るはずだ、ということが分かります。

またハンドルと前輪は、「フロントフォーク」という部品によって繋がっています。

このフロントフォークは、前輪の中心部から真上に伸びるのではなく、乗っている人が操作しやすいようサドル側に向かって少し斜めに生えています。さらにこのフロントフォークは、名前のとおり食器のフォークのように先の方が少し持ち上がるようになっています。このことを知っていると、ハンドルと前輪の位置が正しく描けます。

舵取りをする前輪は左右に動くので、そちらにチェーンを繋ぐと自転車の構造が複雑になってしまいます。そのためペダルを漕ぐ力は後輪に繋がる、つまり自転車は後輪駆動になっています。これを理解していると、ギアやホイール、チェーンは前輪ではなく後輪と繋げて描くべきだ、ということが分かります。

こんなふうに、自転車という対象物に対する知識があれば、自ずと正しい自転車の絵が描けるようになります。必要なものは絵心やセンスではなく、知識なのです。逆に言えば、対象物を知らないと上手な絵は描けません。

そして自分が対象物を思っていたより「知らない」ということに、我々は気付いていないことが多いのです。上手な絵は、対象物のことを知らないと描けません。同じように、**分かりやすい文章も対象物をよく知らないと書けない**のです。

1.2.3 うまく書けないときに足りないのは文章力か技術力か

あなたも「割と知っている」「結構分かっている」はずの技術について本やブログを書こうとしたとき、書く前はラクに書ける気がするのに、実際書き始めるとなぜ書けないのか……と悩んだことがありますか？^{*2}既に使ったことがあり、分かっている技術であっても、理解の解像度が高くないと筆がなかなか進みません。

たとえば「ハッシュドビーフ作りの手順書を作ってください」という依頼があったとして、ハッシュドビーフについて「食べたことがあるけど料理は未経験」という理解度の人と、「食べたことがある。要は肉と野菜とルウを買って煮ればいいんだろうな」という理解度の人と、「牛肉とタマネギときのこを買って、切って炒めて煮込んでルウを入れる料理だな。赤ワインを入れることもある。こびりつくから後片付けのときは油汚れをお湯でよく流して洗わないといけないな」という理解度の人だと、手順書をすらすら書けるかどうかや、書かれた手順の分かりやすさはかなり違ってくるはずです。

分かっているはずの技術なのに、自分は言葉にするのが下手でなかなかうまく伝えられない、と思っている人にとって、実は不足しているのは出力の部分ではなく、入力や理解の部分なのかもしれません。「書こうとすると、なんかふわっとした説明になってしまう……」と思ったら、その「もやもや」っとした部分だけ理解の解像度が

^{*2} 筆者はよくあります！ 本書も最初はさらさら書けると思っていたのに、全然うまく書けなくて苦しみました。

低くなっていることを疑って、改めて対象の理解に努めてみましょう。

私たちは対象物について「分かった！」の山と、「何も分かっていなかった……」の谷を越えて、少しずつ理解していきます。どうか自分が**対象物を知っているつもりでも意外と知らないこと**、そして**知らないものについて分かりやすい文章は書けない**ことを最初に知っておいてください。実際は知らないまま、知っているつもりで体裁の整った文章を書いても、それは分かりやすい文章にはなりません。

美味しい料理を作るには、材料がなければいけません。うまく料理が作れないとしたら、腕よりも前に材料が足りていないことを疑いましょう。出力するためには先に入力が必要であり、そして材料が不足していることに気付けるのは大抵キッチンに立って料理を作り始めてからです。「書けると思ったのに……全然書けない……何も分かっていなかった……」という挫折はある意味必要なステップで、これがなければ対象を理解した分かりやすい説明には辿りつけないです。

1.3 初心者はいっぱい読んでいっぱい書こう

「もっと文章をうまく書けるようになりたいです！ どんなことに気をつけて書いたら、文章力が上がりますか？」と相談された場合、まだ経験の浅い方であれば細かいライティングのテクニックについて学ぶより先に、まずはなんでもいいからいっぱい読むことをお勧めしています。

本屋さんで買える技術書も、インターネット上にある公式リファレンスも、個人の技術ブログや Zenn や Qiita も、いっぱいいっぱい読むと「この説明分かりやすいな」とか「この言い方だと自分みたいな初心者には分かりにくいいな」というように、参考にしたいお手本のような表現も、使わない方がいいと感じる表現もいっぱい自分の中に溜まっています。

レゴブロックと同じで、少ないパーツで何か作れと言われると難易度が高くなります。「この言い回し」とか「こういう表現」とか「こういう説明の仕方」といったいいパーツ、あるいは「こういう構成」や「こういう順番」といったいい組み合わせの例が自分の中にいっぱいあると、それだけ「いい文章」が組み立てやすくなります。

そしてたくさん読んでパーツを集めただけでなく、その溜め込んだパーツを実際に使ってどんどん組み立てて作ってみると上達への近道です。つまり、技術ブログや技術記事をいっぱい書いてみましょう。書くことで「あ、こういう系統のパーツが全然足りてない！」ということに気付いて、また読む作業に対して意欲的になれます。

文章を書くことについてまだ初心者であれば、私のできるアドバイスは「細かいことは気にせず、とにかくいっぱい読んでいっぱい書くといいよ」です。

1.4 完璧主義よりも完成主義

そしていっぱい書くときに大事なのは、「完成させること」です。色々書いたとしても、中途半端な書きかけがブログの下書きにいっぱい溜まっていたり、完成させられないメモ書きや構想ばかりが積まれていたり、という結局誰にも読んでもらえない状態ではうまくなりません。

完成度を求めた結果、いつまでもできあがらない、完成させられない完璧主義はやめましょう。完成度にこだわりすぎず、とにかく完成させること、完成して世に出すことを優先しましょう。一度できあがって世に出さないと、その次の改善するというステップにも進めません。

そのためには「〆切」を先に得ることが重要です。〆切もないのに何かを完成させられるほど我々の精神は成熟していません。あそこもここもおかしいし、本当は直したいし、もっとこういうことにも触れたかった、最初の構想と違って何もかも全然足りてないんだけどこれを完成とする！と開き直るためには、自分では動かせないどうにもならない〆切が必要なのです。

それから「〇〇についてもっと詳しくなったら書こう」と思っていると一生書けません。なぜなら書きはじめてようやく「あ、この辺の知識が抜けている」という不足に気付くからです。人間は書くことで初めて「〇〇について詳しくなる」という階段を上り始めるのです。

ビルの10階に行きたいのなら、あなたに出来ることはまず1階から2階への階段を上ることです。「2階とか3階とかそんな低い場所には居られない。俺はもっと高みに……10階に行きたいんだ！」と意識の高いことを言われても、ヘリコプターでもない限り2階や3階を経ずに10階にはたどり着けません。それと同じで、100点満点の完璧な文章が書ける自分になるためには、まずは10点や20点のひどい出来でも文章を何度も完成させるステップを踏まなければなりません。

最初に高すぎる目標を立てて、「このクオリティが出せるまでは何も書かないし、書いたものも世に出さない！」としているといつまでもうまくなりません。

まずは〆切を手に入れて、完成させて、世に出しましょう。

【コラム】どうやったらテクニカルライターになれますか？

「いまは他の仕事をしているんですが、書くことにも興味があって……IT系の会社でテクニカルライターになるにはどうしたらいいですか？」という質問をいただくことがあります。そもそもテクニカルライターというポジ

ション自体が職業としてまだあまり認知されていないので、そんな中で存在を知ってもらい、さらに「なりたい」と思ってもらえるのは大変有難いことです。

書いたものを見せてもらうのがいちばん早いので、そんなときは「過去に書いたドキュメントやブログなどはありますか？」ZennでもQiitaでもなんでもいいんですが……」と聞くようにしています。これは「プロ野球選手になるにはどうしたら？」と聞かれたときに、普段の練習メニューを教えてもらったり、過去の練習試合の動画を見せてもらったりするような感覚です。

そういうときに「特に何か書いてはいないので、まだ見せられるようなものは何もなくて……」と言われると、お気持ちは分かるものの、正直こちらから言えることが殆どなくなってしまいます。

たとえば毎日毎日ラーメンを食べる仕事があったとして、その仕事に向いているのはどんな人かと言われたら、恐らく「仕事でもないのに毎日食べてしまうくらいラーメンが好きな人」だと思います。「ラーメンって美味しいけど、今まで殆ど食べたことがない。だけど毎日ラーメンを食べる仕事に就きたいと思っている」という希望に対しては、私からは「取りあえず頻繁に食べてみて、結構いけそうか、それとも毎日吃べるのは思ってたよりしんどいのか試してみたらどうでしょう」と勧めるようにしています。

テクニカルライターになると、寝ても覚めても書くことに向き合う生活になります。筆者がこの職業を知ったときは「ドキュメントを書いて給料がもらえる？ そんな夢のような仕事がこの世に？」と思いましたが、ある人にとっては夢のような楽しいことが、他の人にとっては苦行のように感じられることだったりします。好きな技術について頼まれてもいらないのにあれこれ書いてしまう、という人にとってはテクニカルライターはきっと天職です。どんなことでもそですが、適性があるのは「〇〇をやろうと思っている／〇〇に興味がある」という人よりも、「〇〇をやってみた」という人です。^{*3}

^{*3} 大学教員8年目やってるとワナビーとモノづくり好きの区別がつくようになってくる→「へえ、〇〇がやりたくて大学に入ってきたんだ、でなんで今までやってないの？」(次週)「え、どうして今週できなかったの？」 | 落合陽一 <https://note.com/ochyai/n/n781fb2209af4>

第2章

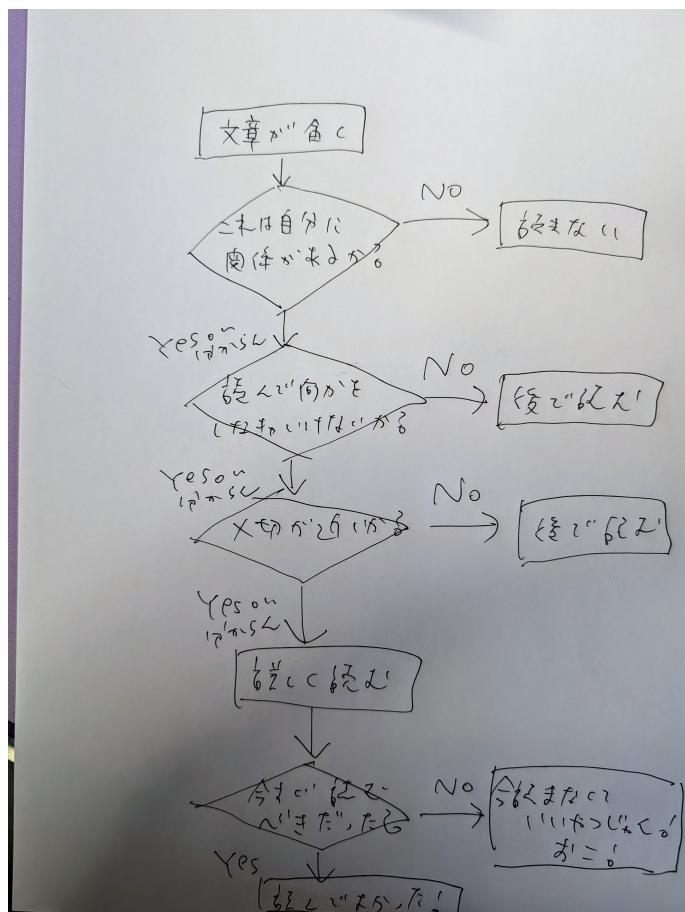
エンジニアのためのテクニカルライティング

仕様書、設計書、ドキュメント、オンボーディング資料、技術ブログ、技術記事など、技術に関する文章を書く場面はたくさんあります。分かりやすい文章を書くために、筆者が実践していることを紹介します。

2.1 関係ない人を早めにふるい落とそう

小説であれば、読みはじめてすぐに「もういいや」と読むのをやめてしまうつまらない話よりも、最後の1文字まで読者を惹きつけて離さない面白い話の方がいいはずです。ですが実用文においては、読んだ人が「私は何をどうすればいいのか」という答えを見つけて、やるべき作業に早く進めた方がいいので、最後まで読んでもらえないのはいいことです。

100人の読み手がいて、書いてある情報を必要としている人はそのうち1人しか居なかった場合に、100人が最後まで読んでもらうと相当な労力が無駄になります。たとえばマネージャ以上の管理職に対して組織の目標設定を促すお知らせがあった場合、最初に「これはマネージャ以上の管理職に向けたお知らせです」と書いてあれば、その時点で管理職ではない殆どの人が離脱できます。(図2.1)



▲図 2.1 読む読まないの離脱フロー

一通り読んだ後で、ようやく自分には関係のない話だと気付いたときの「早く言ってくれよ！」という徒労感には誰しも覚えがあるはずです。このまま読み進むべきか、自分には関係ないので読まなくていいのかを判断できる情報を冒頭で提示するようにしましょう。

みんなの時間を無駄に使わないうことはコスト削減でもあります。組織全体の「読む」労力ができるだけ少なくなるよう、早めにできるだけたくさん的人が該当する分岐で読み手をふるい落としましょう。

2.2 読者層をはっきりさせよう

万人に最適な文章というものはありません。文章を読む人が変われば、適した文章も変わります。

たとえば DNS に関する技術書を書くとしても、対象となる読者層が「インターネット？ ほぼ使ってないです。インスタは使ってますけど」というレベルの大学生なのか、それとも「A レコードは登録したことあるけどフルリゾルバは知らないです」というレベルのエンジニアなのかによって、書くべき内容や説明方法は大きく異なります。

あなたがこれから書く文章は誰に向けたものなのか、を最初にしっかり決めておかないと、のちのち「どこまでさかのぼって説明しないとだめなんだ……？ HTTP ステータスコードの概念からか……？」と頭を抱えることになったり、あるいは読者に「こんな簡単なことはもう知っている。もっと踏み込んだ内容が読めると期待してたのに」と不満を持たれたりします。

文章を読み終わったときに、文句を言いたくなったり低評価を付けたくなったりするいちばんの動機は「思っていたものと違った……」です。読み終わってから残念な思いをさせないよう、ミスマッチを防ぐため読者層ははっきりさせておきましょう。

2.3 想定する読者層とゴールを書いておこう

関係ない人を早めに振り落としてミスマッチを防ぐ簡単な方法は、本文の前に「想定する読者層」と「ゴール」を書いておくことです。たとえば筆者が以前書いた「SSL をはじめよう」という書籍では、想定する読者層を次のように定義していました。

本書は、こんな人に向けて書かれています。

- * よく分からぬままネットの手順通りにSSLを設定している人
- * 「サイトをHTTPS化したいな」と思っている人
- * 証明書の購入や設置の流れがいまいち分かっていない人
- * SSLとTLSの関係性がよく分からない人
- * SSL証明書が一体何を証明しているのか知らない人
- * これからシステムやプログラミングを学ぼうと思っている新人
- * ウェブ系で開発や運用をしているアプリケーションエンジニア
- * 「インフラがよく分からぬこと」にコンプレックスのある人

想定読者を書くことで、ここに当てはまらない人は「自分が期待している内容ではないかもしない」と判断できます。さらにもう一步踏み込んで「マッチしない読者層」も書いておくと、「あ、これは自分向けではないんだな」に気付いて、よりミス

マッチが防げるようになります。

本書は、こんな人が読むと恐らく「not for meだった…（私向けじゃなかった）」となります。

- * SSL/TLSの通信をC言語で実装したい人
- * 「プロフェッショナルSSL/TLS」を読んで完全に理解できた人

また「SSL をはじめよう」ではゴールを次のように定義していました。

本書を読み終わると、あなたはこのような状態になっています。

- * SSL証明書がどんな役割を果たしているのか説明できる
- * 証明書を買うときの手順が分かっている
- * 意図せず「保護されていない通信」と表示されてしまったときの対処法が分かる
- * 障害が起きたときに原因を調査できる
- * 読む前よりSSLが好きになっている
- * SSL/TLSと併記されている「TLS」の意味が分かっている

このようにゴールを書いておくことで、読み手は「読むことで何が得られるのか」を事前に把握できます。ざっと概要だけ知りたいのか、手を動かして実践的な知識を得たいのか、文章を読む目的は人によって異なります。

最初に「想定する読者層」および「マッチしない読者層」、そして「ゴール」を書いておくことで、不幸なミスマッチを防ぐとともに、著者自身が文章の方向性を見失いかけたときに「誰に向けた文章なんだっけ？」「読み終わったらどうなって欲しいんだっけ？」と振り返る拠り所にもなります。

2.4 誰に何をしてほしいのかを書こう

ある日、あなたに「棚卸しのお願い」というお知らせが届きます。お知らせには棚卸しの対象となる資産や、自分が会社から貸与されている資産の一覧を見る方法などが詳しく書いてありますが、結局いつまでに何をすればいいのかや、そもそも自分がこの棚卸しという作業をしなければいけない対象者なのか否かはまったく分かりません。色々と人に聞いたり調べたりした結果、もうすぐ年1回の棚卸しの時期がくるので事前告知として概要を先出しているだけで、現時点はやるべきことやできることは何もない、ということが分かり、あなたには徒労感だけが残りました……。

こんなお知らせ、実際によくありますよね。文章を書くときは、これを誰に読んでもらって、いつまでに何をして欲しいのかを最初の方に書きましょう。

このお知らせが届いた人全員に今すぐ何かをしてほしいのか、それともリンクを踏

んで貸与されている資産があった人にだけ読んでもらって何かをしてほしいのか、はたまた今やれることは何もないから時間のある人に事前告知として聞くだけ聞いておいてほしいのか、いったいこの文章を「誰がどういう気持ちで読めばいいのか」を先に説明してあげましょう。

何だか言わずにはいきなり「食べて！ ほら食べて！」とスプーンを差し出されると、「え、怖い。なになになに？」となって、とても素直に口を開く気にはなれませんし、食べたところで猜疑心で味もよく分かりません。初めて作ったプリンが思いのほか美味しくできたので一口食べて感想を教えてほしい、というように、「どういう気持ちで何をして欲しいのか」を先に説明してあげる必要があります。

文章も同じで「読んで！ さあ読んで！」と要求する前に、これを読んで誰に何をして欲しいのかを最初に書いておく必要があります。

2.5 先に大枠を説明して段々詳しくしていこう

文章を書くときには、先に大枠を説明して、段々詳細にしていくという順番を意識しましょう。

たとえばパパ抜きの説明を書くのであれば、「トランプを使うゲーム」「2人以上でやる」といった概要をまず書きます。その上で、「カードの中に1枚だけジョーカーがある」「同じ数字のカードは2枚を1組にして場に捨てられる」「順番に隣の人のカードを引いていき、最後まで手元にジョーカーが残った人が負け」という詳細を書き、最後に「どれがジョーカーか悟られないようにポーカーフェイスを保つことが大事」「パパ抜きから派生したゲームでジジ抜きもある」のような補足情報を書きます。

この大枠から詳細へという順番を無視して、先に細かい部分から書いてしまうと、読み手は全体像が分かっていないので混乱します。

2.6 既知から未知に繋ごう

意識すべきもう1つの順番が「既知から未知へ」です。知っていることから、知らないことへ、順番に負荷を上げていきましょう。

2.7 一文の長さは一口で食べられる量にしよう

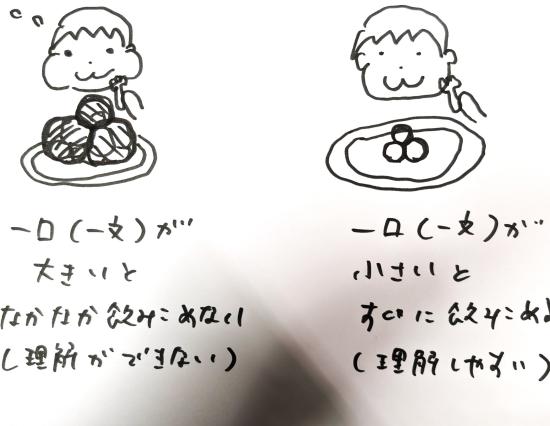
まずはこの「長くて分かりにくい文章の例」を読んでみてください。

まず、事前に確認しておきたいのですが先週金曜の16時の定例のミーティングで話に上がっていた方針で間違いはないと思うのですが、興味深い内容がありましたのでその際にもお伝えしたとおりその件につきまして理解したとおりの見解を共有させていただければという意図です。既存仕様と異なると考えられる理由としては殆どの機

能がメインケースとサブケースを切り分けて動作していないので意図と異なる動きになりユーザーが混乱すると考えているため、リジェクトの可能性もあるので回避のために僭越ながら先方への回答は迅速に行いたいと思っています。

我ながらすばらしくひどい例ができてしまいました。目が滑って全然頭に入ってきません。理解しようと3回くらい読み返しましたが、読み返してもなお何を言っているのかさっぱり分かりません。

文章の「一文」は、食事の「一口（ひとくち）」と同じようなものです。句点（。）までの一文は、食事における一口分と同義なので、一口があまりに大きいと口の中いっぱいに食べ物を詰め込んでいるような状態になります。あまりにも大きい一口は、口の中で延々ともぐもぐ咀嚼する必要があり、いつまでたっても飲み込めません。（図2.2）



▲図2.2 一文を短くすれば理解しやすい

一文の長さは、一口で食べられるくらいの量（短さ）にしてください。具体的に言うと、以下のような接続詞が出てきたらそこを「です」や「ます」にして文章を切りましょう。

- ~ですが
- ~ので
- ~し

- ~して
- ~ため

2.8 素早く近づける

言いたいことと、「これを言いたいのかな？」と相手が察することを素早く近づける

2.9 期待値コントロールをしよう

文章に対する一番のヘイトが生まれる原因は、その文章の善し悪しとは別に「期待していた内容じゃなかった」であるどれだけ美味しいめんつゆでも、麦茶だと思って飲んだ人からすると吹き出すくらいひどい味に感じられる

2.10 要約文ではじめよう

最初に要約文を書いておく

2.11 1つの段落では1つのことを話そう

1つの段落に2つの情報を詰めない

2.12 一度に把握できることは7つまで

短期記憶は7つまで

2.13 まずは「分かりやすい一文」を書けるようになろう

タマネギのみじん切りができないのに、いきなりハンバーグに挑戦するのはやめろ
段階的に難しくなっていくので、段階的にできるようになっていこう

2.14 分かりやすい一文を見つけるために

早くたくさんのパターンを書いて、最善手を見つける 100点を思いつくまで書かないのではなく、思いつくままにたくさんのパターンを書いていく改善案を思いついたら直前の案をコピペして別の案として書くとにかくナンパリングしながら全部の案を並べていく 1番良い物を見つける

2.15 滑り始めを生成 AI にサポートしてもらおう

使えるなら ChatGPT でも GitHub Copilot でもなんでも使う。自転車はこぎはじめがしんどい。文章も同じで、初案を書く作業がしんどい既にある文章を読んで、あーだこーだ文句を付けながら手直ししていく方がラク

ゴミみたいな初案を自分で作れるならそれでもいいけど、「こんな話を書きたい」というのを言って ChatGPT にゴミみたいな初案を作ってもらうのもあり滑り始めだけやってもらうと、スピードがのった状態で手直しから入れるからラク

2.16 意味のない空白や意味のないスペースを残さない

「なぜここに空白？ 敢えて？」となるので、意味のない謎の改行やスペースを残さない「なんでここはこうなの？」と聞かれたら全部理由が言えるようにしておこう

2.17 単語で終わらせない

「確認」じゃなくて「確認した」なのか「自分が確認する」なのか「相手に確認してもらう」なのか「事前告知」じゃなくて「事前告知した」なのか「事前告知をする」なのか最後まで書く

「説明がうまい人」とかどういう人のことか？「話したいこと」を「話したい順番」で好きに話す人手ではなく、「相手の聞きたいこと」を「相手の関心度が高い順番」で話して、早く疑問を解消してあげる人

2.18 文章と修飾を分けて書こう

Markdown 記法のように、文章そのものと、文章の修飾を分けて書こう。Word とか CMS みたいに、文章そのものと修飾が分かれがたく 1 つになっていると後で再加工が辛い。

2.19 再利用しやすいテキストにしよう

再利用しやすい、再加工しやすいテキストにしよう。箇条書きに・をつかっていると、「マークダウンに変換しよう」と思ったとき、箇条書きでないところ「りんご・バナナ・いちご」も誤変換されてしまう。

2.20 並列をナカグロで書くと、後の変更で箇条書きにしたとき見た目が変になる

・生命・身体・健康に影響を与えるもの・投資・資産運用に関連するもの・クラウドファンディング・寄付・投げ銭に類するもの・ヒーリング・セラピーに類するもの・自己啓発に類するもの・メンタルヘルスに関連するもの・宗教・スピリチュアルに関連するもの・政治に関連するもの・アカウントと関連のないプランを提供するもの

2.21 語順を入れ替えよう

より分かりやすい語順がないか考えてみよう。

2.22 修飾語はかかる言葉に近づけよう

修飾語は近づけよう。

誤解の少ない文章にする特効薬は、一文を短くすること。複数の修飾が延々と続くような説明はやめて、一文ごとに句点で終わらせる。

一文を考える一段落を考える一ページを考える一章を考える一冊を考えるドキュメント全体を考える段々と難しくなっていく

2.23 タイトルは「概要」にすべきか、「○○の概要」にすべきか

上位のタイトルを見れば補完されて分かるけど、私はタイトルだけで分かるようにしたいので「○○の概要」派です

2.24 分からないから説明を読むが、分からないと読めない

「返ってきたパケットが TLS/SSL record ではないってことですよ」「え、どういうこと…？（しばらく調べる）あー！返ってきたパケットが TLS/SSL record ではないってことか！」すごい！さっきまで分からなかった文章が分かるぞ！分からないと説明読むんだけど、説明は分かんないと読めなくて、分かると読めるんだよな。

2.25 年月日や対象バージョンを書いておこう

書いたドキュメントは、本人が書いたことを忘れるくらい時間が経ってから突然参照されることがあります。その際、「いつ書かれたものか」という情報がないと、非常に古い情報をいま現在の仕様だと思って読んでしまう可能性がありますので、文章を書くときは必ず「その文章が書かれた年月日」を記載しておきましょう。

ブログであれば、その記事を投稿した年月日が自動で表示されるようにしておきましょう^{*1}。技術書であれば奥付^{*2}に書いておけばよいですが、それ以外に文中でも「今年の技術書典」や「5月26日の技術書典」ではなく「2024年5月26日(日)の技術書典」のように、**数年経ってからその文章を読んでもいつのこと**を指しているのか分かるようにしておくとさらによいでしょう。

またミドルウェアやソフトウェアであれば、**どのバージョンを対象とした内容**なのかも記載しておきましょう。

2.25.1 例示用のIPアドレスやドメインを使おう

たとえば「ブラウザで www.example.com を開くと、名前解決が行われてウェブサーバの 203.0.113.222 という IP アドレスが返ってきます」というように、技術の説明をしていると具体的な IP アドレスやドメインを書きたくなることがあります。このようなときは例示用のドメインや IP アドレスを使いましょう。

実はインターネットでは「例示やテストで使っていいドメインや IP アドレス」というものが定められています^{*3}。

例として記載する URL、メールアドレスなどでは次のものをつかいましょう。

- 例示として使えるドメイン
 - example.com
 - example.net
 - example.co.jp
 - example.jp
- 例示として使える IP アドレス
 - 192.0.2.0/24 (192.0.2.0～192.0.2.255)

^{*1} クラスメソッドさんの DevelopersIO は、1年以上前の記事には「この記事は公開されてから 1 年以上経過しています。情報が古い可能性がありますので、ご注意ください。」という案内が表示されるところが素晴らしいなと思います。

^{*2} 書籍や雑誌の巻末にある著者名・発行者・発行年月日などが書かれている部分。本著にもあります。

^{*3} 例示用のドメインは RFC2606 や JPRS のサイト、IP アドレスは RFC5737 で確認できます。

- 198.51.100.0/24 (198.51.100.0~198.51.100.255)
- 203.0.113.0/24 (203.0.113.0~203.0.113.255)

例示であっても自分の持ち物でないドメインや IP アドレスを勝手に使うことはトラブルの元になります^{*4}。必ず例示用のドメインや IP アドレスを使いましょう。

2.26 同じものは同じ名前で呼ぼう

今までのものが「A」に変わったら、「今までのもの」をなんて呼ぶのか考えよう

同じものを安否確認サービスと安否確認システムと緊急時安否確認アラートみたいに色んな名前で呼ばない。検索したときに、完全一致じゃないと引っかからない検索システムでも引っかかるように。あとバラバラの名前だと修正時に漏れる。

2.27 正しい名前で呼ぼう

ソフトウェアやハードウェアの名称は、自分がなんとなく使っている通称や誤った表記ではなく、正しい名称で書くようにしましょう。(表 2.1)

▼表 2.1 通称や誤記ではなく正しい表記で書こう

通称や誤記	正しい名称
VSCode	Visual Studio Code
Github	GitHub
Word Press	WordPress
JAVA Script	JavaScript
iphone	iPhone

多少でも名前が間違っていると読者も混乱しますし、間違えられた側も決していい気分はしません^{*5}。特にスペースの有無や大文字小文字などは意識していても間違えやすいので、筆者は公式サイトや公式ドキュメントの表記をコピーペーストして使うようにしています。

^{*4} 実際にどんなトラブルになるのか？は「DNS をはじめよう」の P110 「<トラブル> test@test.co.jp を使って情報漏洩」で紹介しています。

^{*5} 以前所属していた会社で MVP として壇上に呼ばれた際、社長に名前を間違えられて「表彰相手の名前くらいは把握しておいてもらえると嬉しい……」と思ったことがあります。相手に興味がなくてもいいのですが、それを悟らせて得るものは何もないのに、せめて興味があるように見える最低限の準備は大事だなと思います。

また英数字の羅列だと覚えにくいけれど何の略なのか分かれば理解しやすくなる、という側面もありますのではじめは正式名称で紹介して、以降は略称にするという形もよいでしょう。読み方が分からずにひそかに悩んでしまう^{*6}のも初心者あるあるですでの、次のようにカタカナで読み仮名も添えるとなお親切です。

AWSではサーバはAmazon Elastic Compute Cloudの略で「EC2」(イーシーツー)と呼ばれています。

2.28 箇条書きを挟んだ文章を作らない

私がやりたいことは

- A
- B
- C

の3つです。

みたいに書かない。

2.29 主語と述語を対応させよう

主語と述語のねじれを見つけよう。

2.30 「自分」という主語に注意しよう

カメラマンから「写真を撮るときは鏡に映った自分の目を見てください」と指示されたら、あなたは誰の目を見ますか？この指示文は、次の2つの解釈が可能です。

- 写真を撮るときは、鏡に映ったカメラマンの目を見てほしい
- 写真を撮るときは、鏡に映った自分自身の目を見てほしい

自分という一人称が指すものは、「私」であることも「相手」であることもあります。

^{*6} k8s は Kubernetes の略でクバネティスと読むとか、nginx と書いてエンジンエックスと読むとか、誰かに教えてもらわないと筆者は想像もつかなかったです。密かに「んぎっくす…？」と思つていました。

2.31 ひらがなに開こう

一方で、漢字にしないと意味が即座に取れずに返って分かりにくいものもある。

2.32 分かったと分からぬの両方の気持ちが必要

分かんない人の気持ちも分からなきゃいけないし、分かんない人に教えられるだけの理解度もいる

2.33 リンク名称を「こちら」にしない

可能なら URL は全部書くコピペしたときに、テキストとして貼り付けると情報が失われることがあるあとリンク先が間違っていたときに、リンクテキストが書いてあることで間違いに気づけるリンクを開く前に、なにに飛ばされるのか判別できないので、見るべきか見なくていいのか分からぬ

たとえば利用規約など同意必須の内容を「こちら」をリンクにして参照させていた場合、何かで A タグが外れたり、プレーンテキストとしてコピペして別の場所で使われた際に意図せずリンク情報が消えることがある。URL をそのまま書いていれば、少なくとも自力で URL を開いて参照できる。

2.34 一文の中で同じことを二度言わない

「俺が避けるべきだと思う実装は、こういうのやああいうのは避けるべき」みたいに、文章として崩壊していないか確認しよう。

2.35 時刻の表記は JST か

時刻の表記は JST ? UTC ?

2.36 その「文字数」って何文字ですか？

文字数の上限を示すときは、カウント方法をちゃんと書こう。半角は 1 文字分？全角なら 2 文字分？ それとも 1 文字は 1 文字？ 絵文字は何文字？

2.37 以上と以下か、より大きいと未満か

上限や下限の数字を示すときは、「以上」なのか「より大きい」なのか、「以下」なのか「未満」なのかを明示しよう。

Textlint

文章を書き始める前に、構造を考えよう

「要はなんだ？」格好良く、美しく、ちゃんとした日本語で、でも意味は1つもわからん、ということはある。<https://www.itmedia.co.jp/news/articles/2205/10/news123.html>

2.38 「教えてあげる」ことに酔わないこと

「教えてあげる」「話を聞いてもらう」のは基本的に気持ちがいいことなので、酔わないように気をつける。色々アウトプットをしているように見えて、実際はここ10年同じことを繰り返し言っているようだと、苦しみながら新しいものを学ぶという楽しい時間を逃しているかもしれない。

2.39 そこにあることを気付いて辿り着いてもらわないと意味がない

いくら分かりやすいドキュメントがあっても、どこにあるのか分かりにくいと意味がない。届かなければ意味がない。

よく読めば分かるでしょ、というのは提供する側の傲慢さで、人はできれば文章なんか読みたくない。だから「今北産業」があるので。

最初の2、3行を読んで「ここにはなさそう」と思われたら、そこで大方の人間が離脱する。

困っている人は困っているので割と最後の方まで熱心に読んでくれるけど、熱心に読んだ挙げ句に欲しい情報が手に入らないと怒り狂う。

2.40 変わっていく「語感」を捨て置かずに拾おう

若い世代に「1時間弱」は何分ぐらいか？と聞くと、「1時間が60分、60分とちょうどだから70分くらい？」と答えるらしい。まさか、と思って息子に聞いたたら、その通りに答えた。

その理屈でいくと「1時間強」は何分くらいになるのか？と聞いたところ、「60分と結構たくさんくらいなので、85分くらいとか？」らしい。なるほど。同じ理屈で

2.41 見つけやすくたどり着きやすいドキュメントにしよう

大さじ 1 杯弱も「大さじ 1 杯十ちょっと」だと思っていたとのこと。

本来の解釈は伝えたが、こういう「口に出さない誤解」は周囲も誤解に気付かないで解くのが難しい。

言葉の本来の意味は辞書に載っていますが、その言葉から受け取る「こんな感じかな」という語感は、人や世代によって移り変わっていきます。

学校で先生に当てられて答えを言ったとき、先生から「はい。結構です」と止められたら、「満足しました。そこまで十分ですよ」というプラスの意味で受け取るか、「もういいです。それ以上聞きたくありません」というマイナスの意味で受け取るかは、人に寄って異なると思います。

正しい意味はこれなんだ！ 誤解した意味で受け取る側が悪い！ 日本語をちゃんと勉強しろ！ と怒る気持ちも分かりますが、誤解する人が 2 割、3 割を増えていってもなお、書き手が頑なにその存在を無視し続けるのはよくありません。

誤解する人が多いと分かった時点で、「1 時間弱というのではなく 1 時間より少し少ない」という意味です」というような補足を入れてあげるか、曖昧な言い方をやめて「45 分～60 分」のように誤解されない言い方に直してあげましょう。

2.41 見つけやすくたどり着きやすいドキュメントにしよう

「メンテナンスされていないものも含めて、ドキュメントがとにかく大量にある」状態は、「ドキュメントがまったくない」状態よりも人を混乱させることができます。ドキュメントはあればあるほどいいというものではなく、メンテナンスを怠ればコードと同じようにドキュメントも負債化します。

お知らせや告知、社内資料があまりにも多すぎて、とてもたどり着けない、アクセシビリティが低い状態では、ドキュメントはその良さを発揮できません。

良い文章を書くことと同じくらい、その文章を見つけやすくたどり着きやすい状態にしておくことが大切です。

2.42 リンクは「張る」ものか「貼る」ものか

蜘蛛がこちらからあちらへ糸を張るように、他の情報と繋ぐためのリンクなので、個人的にはリンクは「張る」を使っています。

URL をコピーペーストするイメージで「貼る」を使いたくなるかもしれません、Slack でリンクを教えてあげるときは「URL 貼っときますね」だし、「商品一覧から詳細ページにリンクを張る」とときは「張る」だと思っています。

https://twitter.com/mainichi_kotoba/status/1769229909283778901

- 2.43 「無駄なことをしたくない」から何もしないと何も書けない
- 2.44 いただくはだいたい「くださる」にできる
- 2.45 「しましょう」は Let's do it together か You should do it か

第3章

英語を書くとか翻訳するとか

英語ができない日本語話者のエンジニアが、なんとか頑張って英語をひねり出したときにやりがちな失敗と、書いた文章が変な言い回しになっていないか確認する方法を紹介します。

3.1 単語と単語の間にはスペースが1つ必要

たとえば You can use this API to get weather information という英文を見ると分かるように、単語と単語の間にはスペースが1つあります。英語が不得手な人でも、そこまではなんとなく認識できていると思います。

ですが、括弧やカンマやピリオドといった記号が入ると、この**単語と単語の間にはスペースが1つ入る**という原則を忘れがちです。たとえば、以下のような自己紹介文を書いてしまったことはありませんか？

```
mochiko(nickname)
Living in Japan(Tokyo)
Presented at Tech Seminar vol.22,vol.23
Loves : Flutter,Nuxt.js,Kubernetes
```

この自己紹介文は、どれも単語と単語の間にはスペースが1つ入るという原則を守れていません。重要なのは**記号はスペースの代わりにはならない**ということです。間違いに気付くため、括弧やカンマやピリオドといった記号を消してみましょう。

```
mochikonickname
Living in JapanTokyo
Presented at Tech Seminar vol22vol23
```

Loves FlutterNuxt.jsKubernetes

あちこちで単語が繋がってしまったり、逆にスペースが2つ続いてしまったりしています。では単語と単語の間に1つだけスペースを入れた上で記号を元に戻して、正しい英文に直してみましょう。

```
mochiko (nickname)
Living in Japan (Tokyo)
Presented at Tech Seminar vol. 22, vol. 23
Loves: Flutter, Nuxt.js, Kubernetes
```

できました！括弧書きの手前には半角スペースが入りますし、volume の略である vol. と数字の間や、区切り文字であるカンマの後ろにもスペースが入ります。逆に Loves とコロンの間にあったスペースはなくなりました。

これは日本語話者が英語を書いたときに非常によくやる失敗^{*1}です。

3.2 単語や文の単位で翻訳すると危ない

OSS などで英語を日本語訳するとき、前後の文脈や原文で言いたいことを汲まずに単語や文の単位で翻訳すると、おかしな日本語になることがあります。たとえば、AWS が提供している日本語のドキュメント^{*2}に、以前こんな一文がありました。

インスタンスを削除することは、実質的には、そのインスタンスを削除するということです。いったん終了したインスタンスに再接続することはできません。

削除するということは、実質的には削除するということ……進次郎構文^{*3}みたいになっています。どうしたのでしょうか。

^{*1} かくいう私もよくこの失敗をしていましたが、同僚が「日本人が英語でやりがちな失敗 / Common mistakes in English that Japanese people tend to make」というイベントで説明してくれてようやく理解できました。感謝！ <https://www.youtube.com/watch?v=2nXUkXmFfZI&t=943s>

^{*2} チュートリアル: Amazon EC2 Linux インスタンスの開始方法 https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/EC2_GetStarted.html

^{*3} 「今まではいけないと思います。だからこそ、日本は今まではいけないと思っている」のように前半と後半で同じことを繰り返す謎構文のこと。 <https://dic.nicovideo.jp/a/%E9%0%B2%E6%AC%A1%E9%83%8E%E6%A7%8B%E6%96%87>

3.3 前後の文脈やどこで使われるのかを知らずに翻訳はできない

原文と思われる英語^{*4}を当たってみると、どうやら Terminate と Delete を両方とも「削除」と訳したことで生まれてしまった悲しい翻訳文だったようです。

```
Terminating an instance effectively deletes it;  
you can't reconnect to an instance after you've terminated it.
```

「インスタンス」は AWS におけるサーバのことです。なので恐らく原文はこう言いたかったのだろう、という意図を汲んで日本語にすると、「サーバの『終了』とは、単に電源を落とすシャットダウンではなく、サーバそのものを『削除』してしまうことを意味します。そのため『終了』させたサーバには二度と接続できないので注意してください」という注意喚起の文章だったのではないかと思われます。^{*5}

Terminate を英和辞典で引くと「終わらせる」「解除する」という訳が出てきます。「Terminating an instance」で「インスタンスを終わらせる」ということなら、確かに「インスタンスを削除する」と訳してもおかしくはありません。もう 1 つの Delete も「削除する」「消す」なので、どちらも「正しく」訳した結果、「削除する」ということは、実質的には削除するということ」という進次郎構文が生まれてしまったのだと思察しています。

この注意喚起文によって何を食い止めたかったのか、という背景や意図を理解せず、単に単語や文の単位で「正しく」翻訳すると、こんなふうに実際は意味を為さない文が生まれてしまうことがあります。

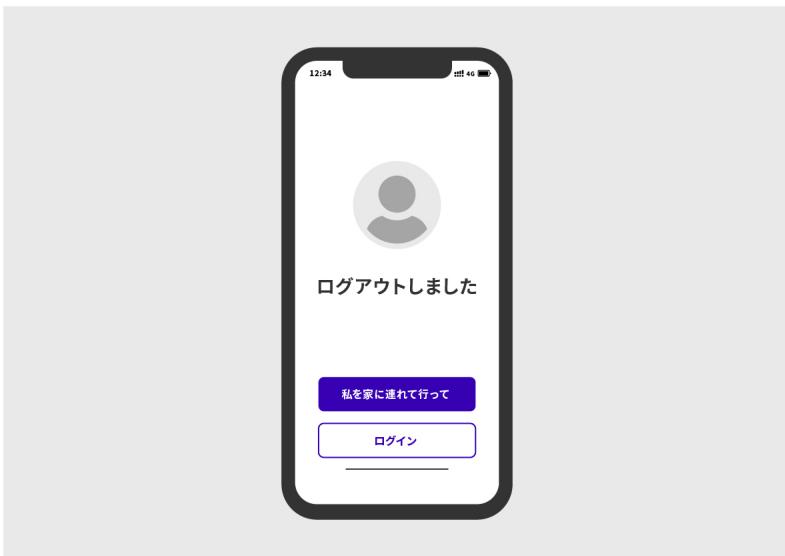
3.3 前後の文脈やどこで使われるのかを知らずに翻訳はできない

とても簡単な英語を「正しく」翻訳しても、おかしくなってしまうことがあります。たとえば「Take Me Home」を「私を家に連れて行って」と訳すのは、英語がそこまで得意でない人が見ても恐らく正しいと思うはずです。

ですが、これがアプリでログアウトした後の画面に表示されていたらどうでしょう？（図 3.1）

^{*4} Tutorial: Get started with Amazon EC2 Linux instances https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

^{*5} AWS？ インスタンス？ その辺をもっと詳しく教えてくれ！ と思ったら「AWS をはじめよう 改訂第 2 版～AWS による環境構築を 1 から 10 まで～」をどうぞ。 <https://booth.pm/ja/items/1032590>



▲図 3.1 ログアウト画面

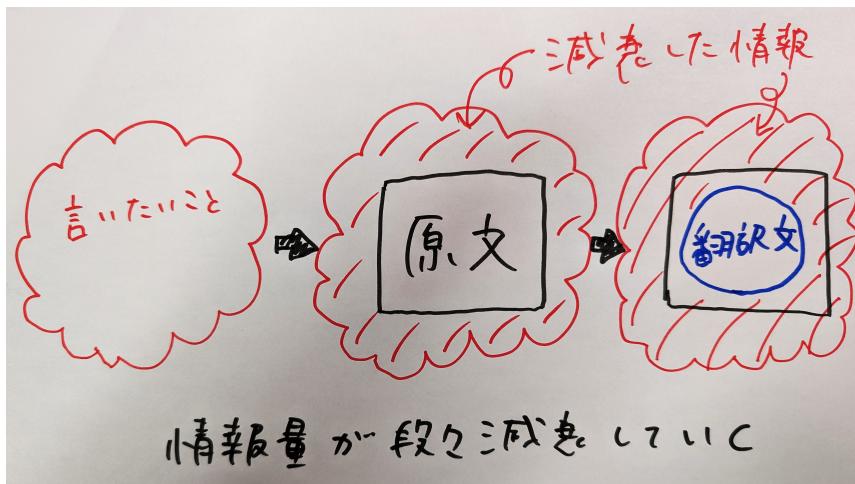
恐らく「Take Me Home」の訳は、「ホームに戻る」が適切だったはずです。その文章がどこに配置されるのか分からずに訳すと、やはり「正しい」のにとても違和感のある UI 文言になってしまうことがあります。^{*6}

3.4 翻訳はもとの言いたいことに立ち返って考える

英語から日本語に翻訳するとき、あるいは英語から日本語に翻訳するときに、原文をしっかり読んでも誤訳してしまうことがあります。

もともと頭の中に何か「言いたいこと」があって、そこから原文が生まれているので、原文になった時点で情報量は減っています。JPEG のように、文章として保存されるたびに情報量は減衰していくので、元の「言いたいこと」に立ち返らずに、情報量の減った原文だけを読んで翻訳すると、原文から翻訳文になるときさらに情報量が減衰します。そして文章としての体を為すためにその減衰を補おうとして、ものの「言いたいこと」にはなかった想像や嘘が混ざることがあります。(図 3.2)

^{*6} 実在した事例です。ただこういう翻訳を見ると、ひどい訳だと笑う前に、もはや日本が「丁寧に翻訳するコストをかけるほどの市場規模じゃない」と相手にされていない可能性を考えてしまってうーん……。 <https://twitter.com/mochikoAsTech/status/1779009150426767775>



▲図 3.2 言いたいことを書いたり翻訳したりすると情報量が減衰していく

元々言いたかったことと、実際に発露した文章はイコールではなく、そこには必ず差分があります。翻訳するときは、原文を書いた人に意図や背景や、元の言いたかったことを質問して、できるだけ元の「言いたいこと」に立ち返って訳しましょう。

3.5 技術文書の翻訳に必須なのは英語力や文章力よりも 技術力

前述のとおり、OSS や技術書など英語から日本語に翻訳する場合、元の「言いたいこと」や前後の文脈を汲めるかどうかが重要です。そのため、こと IT の分野においては翻訳に必須な力は対象技術の知識、実際に触ったことがある深度の深い理解であって、日本語の文章力や、英語の文章力はその次だと個人的には考えています。もし技術力はあっても、英語が不得手だからと OSS の翻訳協力に尻込みをしていたら、ぜひ一歩踏み出してみてください。

翻訳する人は「文脈が分かっている」と非常に強いです。そもそも何の話か分かれているから、原文が多少説明不足でも元の「言いたいこと」を推察してピントの合った訳ができます。

ドキュメントや技術書を翻訳するとき、英語が読めても「何を言っているのか」が分からないと、文法があつてはいるだけで正しいことを何も伝えない、きれいで誤った翻訳文になってしまいます。

3.6 スペルミスが心配なら ATOK に頼ろう

筆者は自分が記憶している英単語のスペルや意味にまったく自信がないので、ATOK^{*7}という有償の日本語入力システムを使っています。

ATOKは、たとえば「ほうこくはいじょうです」という入力を、「報告は以上です」という漢字とかなが交ざった文に変換する部分を担うツールです。もともとWindowsやMacにはデフォルトでMicrosoft IMEなどのIME^{*8}が入っていますが、性能や精度はそこそこので、平気で「報告は異常です」といった意図しない変換をかましてきます。ストレスなく文章を書くための投資だと思って、その辺りの変換が非常に賢いATOKにお金を払うと思っていた以上のリターンが得られます。

ATOKは日本語を入力するときにも便利なのですが、英語を書くときにもしっかりとサポートしてくれます。たとえばATOKで「ろくがつ」と入力すると、変換結果に「June」(図3.3)が出てきます。あるいは「ひつよう」や「ねせさりー」と入力すると、「necessary」が出てきます。



▲図3.3 日本語を入力すると変換候補に英語も出てくる

*7 Windows、Mac、iOS、Androidに対応。ATOK Passport プレミアムなら月600円（税抜き）で最大10台までインストールできる。登録している単語や自分の入力のクセなどが複数台のPCとスマホすべて同期されるので非常に便利。<https://atok.com/>

*8 Input Method Editorの略。英語であれば物理キーボードに書かれた文字を直接叩くことで文字が入力できるが、日本語のようにすべての文字に対応するキーがあるわけではない言語では、このIMEという入力システムを使って変換することで文字入力を実行している。

また単語の意味や使い方を確認したいときは、調べたい言葉を選択して **ctrl** キーを 2 回押すことで、すぐに広辞苑、大辞林、ウィズダム英和辞典、ウィズダム和英辞典といった最新の辞書が引けます。(図 3.4)



▲図 3.4 単語を選んで **ctrl** キーを 2 回叩くと複数の辞書が引ける

人間はミスをする生き物なので、自信のない分野は積極的にツールに頼りましょう。お金かかるのはちょっと嫌なので毎回ネットで検索すればいいや、と思っていても、ちょっとした面倒くささを乗り越えて毎回調べるのは大変なので結局やらなくなってしまう可能性が高いです。生産性のためによいキーボードやよいシェアを選ぶように、ぜひよい IME を使いましょう。

3.7 例は「ex.」ではなく「e.g.」

日本だと例示 (Example) の略で ex. って書く人が多いけど、ex だと「前の彼女」を「ex-girlfriend」って言ったりするときの「前の」の意味なので e.g. の方がよさそう。

3.8 機械翻訳があれば英語がまったく分からなくても訳せるか

機械翻訳も ChatGPT も、しつこく真逆の意味に翻訳したり、間の文章を一文落としてきたたり、逆に 2 回繰り返して訳してきたりする。

機械翻訳はあくまで電動自転車みたいなもので、自力で漕げないときには乗れない

(=乗ると事故るので乗ってはいけない)。

3.9 翻訳しづらい文章をやめよう

なんか言ってるけど、実際のところ何も言っていない、翻訳しづらい文章はやめよう。

第4章

誤解を生まないテキストコミュニケーション

4.1 まずは「何の話か」を先に言う

「〇日ヒマ？」は、ヒマと言ったらどうなるのか分からないので返事がしにくい何が欲しいのかの後に、どういう背景でこれを申し出ているのか、自分がどういう立場なのかを添えると、よりよい案を出してもらえることがある

4.2 背景も添えて話そう

「真鯛がほしい」だけだと「ありません」になるけど、「なぜならば今晚鯛しゃぶにしたいから」を添えると「真鯛はないけど、いい金目鯛があるよ」と言われるかも

4.3 「聞いてほしい」のぶつかり合い

こちらの言いたいことを聞いてもらうには、先に相手の懸念を解消してあげないといけない、というケースがあります。

4.4 Slackでのコミュニケーション

4.5 チケットの書き方

このチケットでやること、このチケットのスコープ外とすること、やることになった背景、スケジュール、誰がOKしたらリリースできるのか、リリースのタイミングを合わせなければいけない他の何かがあるか

4.6 チケットのサイズはできるだけ小さくする

チケットクローズやリリースは健康によいので、できるだけサイズを小さくして1工程が終わったら即閉じられるようにしておこう。

4.7 見せて選んでもらう

実物を見ないでした議論は、実物を見るとあっさりひっくり返る質問は「どんなんのがいいですか？」とオープンで聞かず、現物を用意して「AとBどちらがいいですか？」

4.8 いっぱい書いて Working Out Loud な働き方をしよう

作業が途中でも、みんなの目に触れる times でどんどんアウトプットしていく。実況中継しながら作業したり、分からぬことを「これってどういうことだ？」と口に出していく。(文字に、だが)

質問が必要になったら質問すればいいと思うかもしれないが、「やらなくてもいい無駄な工程をやっている」などは、そもそも自分で無駄だと気付いていないので「これって無駄な工程ですか？」という質問も出てこない。よって「あ、こいつ無駄な方向に歩いていくこうとしている」が目に見えないと「そっち行かなくていいよ」というアドバイスがもらえない。

<https://blog.studysapuri.jp/entry/2018/11/14/working-out-loud>

第5章

レビューする、レビューされる

5.1 文章がひどいと突っ込みが多すぎて読むに堪えない

ひどい文章は、本当に、読むに堪えない。ざらざらした音の悪いポッドキャストを聞くに堪えないのと同じで、「情報を伝える」という目的を阻害する。

伝われば文法とか何でもいいでしょと言うが、なんと最低限のルールを守らないと伝わらない。

5.2 指摘は「後出しじゃんけん」だと心得よ

モンティ・ホール問題と同じで、既に誰かが書いた技術記事に対して「もっとこうすれば」「技術的に誤りが」と指摘するのは、基本的に後だしじゃんけんなので、最初の記事を書いた人より有利な立場であることを自覚したうえで指摘すべき。

5.3 他人が書いたものに敬意を払おう

色々な事情があってこうとしか書けなかったのかもしれないそのときの精一杯なので、馬鹿にしても何もいいことはない

5.4 頼まれていないレビューは MUST FIX でないかぎりしない

人には間違えて素っ転んで痛い思いをする権利がある。そこにあるのが取り返しのつかないような明らかな間違いで、誰がどう見ても MUST FIX なものであり、こちらから to-be 案も提供できる、というものでなければ、世に出る前のものに頼まれていないレビューはしない。

5.5 レビュアーが気付いたことに、書いたとき気付かなければ当然

レビュアーは読むだけなので、リソースの 100% を「注意深く読むこと」に使えます。

一方、書く側は「情報全体を把握し」「どう伝えるか考え」「キーボードを叩いて文章を組み立てながら」「読み返し」「修正案を再び考え」「どちらがよいか判断する」というように、リソースを思考や判断、出力といった作業に振り分けています。

わらわらと 10 人いる幼稚の面倒を同時並行で見なきゃいけない保育士と、1 人だけにつきっきりになれる保育士だったら、そりゃあ後者の方が「襟元にカレーのシミがあるな」とか「昨日より少し元気がない」とか、色んなことに気付けるはずです。レビュアーは、自分が気付いたことにライターが気付かないのは当然であると心得ましょう。

一方、書き手はレビュアーから指摘を読んで、「俺はなぜ……これを書いたときに気付かなかつたんだ……」と己の目の節穴さに落ち込む必要はありません。

レビュアーのときは気付くし、ライターのときは気付かない。これはもうそういうものなのです。

5.6 レビューコメントには重要度を添えよう

前述のとおり、レビューでの指摘は「相手のやることを増やす」ものであり、公開を遅らせるブロッカーになり得るものです。

レビュアーは軽い気持ちで指摘したのに、ライター（レビュイー）は「これを全部直さないと公開できないんだ」と思い込んでしまい、必要以上にコストをかけて全部を直すことになった、というようなすれ違いは悪感情を生むし、勿体ないものです。

これを防ぐため、私がドキュメントや技術書のレビューをするときは、レビューコメントに以下のような「重要度」を添えています。

- MUST FIX
 - 明らかな誤りで、このままでは重大な問題があるので必ず直してほしいもの
- Nice-to-have
 - 絶対ではないが直した方がよくなるのでできれば直してほしいもの
- Nitpicking
 - 軽微な誤りや好みの問題、このままでいいけど気になったので一応伝えておくもの

5.7 お願いしたい観点を添えてレビューを依頼しよう

これにより、ライターは「早く公開することが最優先なので MUST FIX だけ対応しよう」や「〆切まで時間があるし、読みやすくすること最優先にして全部丁寧に対応していこう」というように、どこまで直すのかを判断しやすくなります。

5.7 お願いしたい観点を添えてレビューを依頼しよう

どういう観点でレビューしてもらいたい、何について指摘をして欲しい、という認識がレビュアーとレビュイーの間で合っていないと、たまに「ざっと見て欲しいだけだったのに、頼んでもいない細かいところまで何度も指摘されて嫌な思いをした」や「すごくたくさん指摘するところがあってこちらも大変だったのに、言い訳ばかりされて結局全然直してもらえなかった」という地獄のような状態が生まれます。

レビューを頼むときは以下のように、どんな観点で何について指摘をして欲しいのか、あるいはどういう観点の指摘は不要なのか、といった情報をきちんと伝えるようにしましょう。

- 技術仕様に誤りがないか
- 初心者向けでも分かりやすい構成になっているか
- 日本語として明らかに誤りがないか
- より分かりやすくするためにできることがないか
- 他の人が書いた章と文体が揃っているか
- 不足している情報がないか
- 説明に不親切なところがないか

5.8 大量に赤を入れることがレビューの存在意義ではない

稀に、レビューを頼まれたときに「たくさん赤を入れないと仕事をしたことにならない」と思ってしまう人がいます。そういう思い込みのままレビューをすると、「たくさん指摘をすること」が最優先になっているため、文章をよくするためのレビューのはずなのに、逆に誤りを混入させるような修正指示や、簡潔で分かりやすい文章を長く曖昧にさせる提案をしてしまいます。

大量に赤を入れることがレビューの存在意義ではありません。よい文章を早く世に出すことが、レビュアーとレビュイーの共通目的であると心得ましょう。

レビューで指摘するところが少なかった、ということは、「レビュアーが仕事をしていない」ということを意味しません。ライターが書いた元の文章の出来がもともとすごくよかったのかもしれないし、レビュアーの方が知識が不足していて、誤りを見

つけられていないだけかもしれません。

どちらにせよ、指摘することが少なかったときに、粗探しをして無理矢理指摘をひねり出す必要はありません。

5.9 レビューは『相手のやることを増やす』責任を持って言おう

「分かりにくいです」と感想を伝えるだけのレビューや、直した方がいいかもしれません箇所を増やす指摘は簡単です。逆に「これでいいよ」と責任を持ってゴーサインを出す方がずっと難しいことです。

書いてある内容について自分もきちんと理解していれば「あとこれだけが足りない」「ここは明確に間違っているので、○○に直しましょう」がはっきり言えるけれど、理解していない状態で「なにか言わないと」とひねり出そうとすると、「できればこれも書いておいた方が分かりやすいんじゃないでしょうか」と無責任に宿題を増やすだけの人になってしまいます。

直して持っていくとまた新しいことを言われて、何をクリアしたら出せるのか分からぬ、という無限レビューはレビューの心を折ります。^{*1}

レビューをするとき、レビューは、その修正指示が「早く出る」という価値を毀損してまでいまやるべきことなのかを自分にきちんと問いましょう。

レビューには「文章をよくする」といういい面だけでなく、「相手のやることを増やす」「早く出るという価値を損なう」というように足を引っ張る面もあります。なんでもかんでも気軽に放言せず、私は今から相手のやることを増やすのだ！公開を遅らせてでも直した方が絶対にいいんだ！という責任を持って指摘を伝えるか、前述のような重要度を添えて、取り込む取り込まないを相手が選べるようにしてあげましょう。

5.10 指摘は素直に受け入れる

レビューは指摘を聞くと、なんだかんだ言い訳をして取り込まない方向に持っていきたくなりますが、素直に受け入れてさっと直すと大抵その方がずっとよくなります。

^{*1} 作品を持ち込んでは編集者にダメ出しされることの繰り返しで、何をどう直したら掲載されるのか分からず心が折れてしまう漫画家や小説家の話を読むと、ジャンルは全然違うけどたぶん同じような状態なんだろうなと思ってしまう。

第6章

どうやって学ぶか

6.1 テクニカルライティングの検定を受けてみよう

テクニカルライターになって2年半、テクニカルライティング技術を改めて体系立てて学んでみたい！という訳で、2022年7月24日(日)に「TC技術検定3級」(正式名称はテクニカルコミュニケーション技術検定試験3級 テクニカルライティング試験)を受けてきました！

試験の詳細はLINE Technical Writing Meetup vol. 15というイベントでお話ししましたので、なにそれ？テクニカルライティングの試験なんてあるの？どんな試験なの？という方はこちらのスライドと動画をご覧ください。

https://speakerdeck.com/line_developers/my-story-about-taking-the-technical-writing-exam

<https://www.youtube.com/watch?v=N90S4BtDHik&t=856s>

公開されている分析データによると、今回の合格率は71%だったようです。ひとつ前の回（2022年2月）は合格率58%だったので、私の「想定より難しかった…」という体感とは異なり、実際は従来より問題が易しめだったのかもしれません。（あるいは受験者のレベルが軒並み高かったとか）

そして2022年9月2日(金)に、郵送で合否判定通知書と合格証書も届きました。合否判定通知書に載っていた点数は選択問題が92点、記述問題が87点だったので、なんとか面目を保てる点数でほっとしました。（毎日「どうもどうも！テクニカルライターです！」という顔で仕事をしているのに落ちたらまじで洒落にならんと思っていた）

6.2 色んな本を読もう

近年、技術の1ジャンルとして確立されてきたのか、テクニカルライティング技術を題材にした本がたくさん出版されるようになりました。

6.3 「書き方」の指標を見つける

文化庁が2022年1月に出した「公用文作成の考え方」という資料がある。

「公用文作成の考え方」について(建議)<https://www.bunka.go.jp/seisaku/bunkashinkai/koki>

公用文作成の考え方(建議)(PDF)<https://www.bunka.go.jp/seisaku/bunkashinkai/kokugaku>

6.4 Technical Writing Meetupに参加しよう

テクニカルライティングをテーマにしたMeetupを開催しているので、ぜひオンラインで参加しよう。

過去のアーカイブ動画も見られるよ。

あとがき

本著はテクニカルライティングをテーマにした本ですが、あとがきはあくまであとがきであってあとがきでしかないのでここからは自由にまいりましょう！ いえーい！ 実用文とかテクニカルライティングとかを気にしないときの、私の自由な文体を形成した作家さんは野梨原花南です。

さて、東京ディズニーランドにおいて、コロナ禍真っ只中の2021年4月1日から始まった「ミッキーのマジカルミュージックワールド^{*1}」という常設のステージショーがあります。開演からちょうど3年が経った2024年4月1日、このショーの演出や出演者の人数が予告なく大きく変わって、恐らく「これってもともとはこういうショーだったんだ！」という完全版のような状態になりました。

精神が舞浜で育った結果、ミッキーとミニーが顔を見合わせて頷く姿を見ると、すぐに涙ぐんでうんうん頷いてしまう筆者ですが、このショーは本当にお勧めです。やっぱりいい店と美味しいご飯の組み合わせで最高の一品っていうか、舞台のよさと演者のよさが組み合わさって最高のショーが！ あー！ 観てくれー！

ところで最近、セカンドハウスを買いました。ベランダから海と山が見えて、部屋のお風呂でじゃぐちをひねると温泉が出るヴィンテージのリゾートマンションです。いわゆる「別荘」の響きとは裏腹にかなり安かったので、おうちオフィスの別拠点としてうっかり買ってしました。わいわい。たのしいね。

2024年5月
mochikoAsTech

^{*1} <https://www.tokyodisneyresort.jp/tdl/show/detail/895/>

PDF 版のダウンロード

本書（紙の書籍）をお買い上げいただいた方は、下記の URL から PDF 版を無料でダウンロードできます。

- ダウンロード URL
 - https://mochikoastech.booth.pm/items/*****
- パスワード
 - ****

Special Thanks:

- 「だめだよ」と言われると「かわいいのに？？」という顔をするねこ

レビュアー

- やさしい誰か

参考文献

- センスは知識からはじまる - 水野学
 - <https://publications.asahi.com/product/15849.html>
- 日本語スタイルガイド（第3版）-一般財団法人テクニカルコミュニケーション協会
 - https://jtca.org/learn-tc/publication/guide_jsg/
- 論理が伝わる 世界標準の「書く技術」 - 倉島保美
 - <https://bookclub.kodansha.co.jp/product?item=0000194754>
- ずかん自転車一見ながら学習調べてなっとく - 森下昌市郎
 - <https://direct.gihyo.jp/view/item/000000003092>
- ことばから誤解が生まれる 「伝わらない日本語」見本帳 - 飯間浩明
 - <https://www.chuko.co.jp/ebook/2013/07/514091.html>
- Software Design 2024年4月号 - Software Design 編集部
 - <https://gihyo.jp/magazine/SD/archive/2024/202404>

著者紹介

mochiko / @mochikoAsTech

テクニカルライター。元 Web 制作会社のインフラエンジニア。ねこが好き。「分からない気持ち」に寄り添える技術者になれるように日々奮闘中。技術書典で頒布した「DNS をはじめよう 改訂第 2 版」「AWS をはじめよう 改訂第 2 版」「SSL をはじめよう」の「はじめようシリーズ 3 部作」は累計で 12,000 冊を突破。

- <https://twitter.com/mochikoAsTech>
- <https://bsky.app/profile/mochikoastech.bsky.social>
- <https://mochikoastech.booth.pm/>
- <https://note.com/mochikoastech>
- <https://mochikoastech.hatenablog.com/>
- <https://www.amazon.co.jp/mochikoAsTech/e/B087NBL9VM>

Hikaru Wakamatsu

表紙、章扉、目次デザインを担当。

Shinya Nagashio

挿絵デザインを担当。

読み手につたわる文章が書けるテクニカルライティング 仮称

2024 年 5 月 25 日 技術書典 16 初版

著 者 mochikoAsTech

デザイン Hikaru Wakamatsu / Shinya Nagashio

発行所 mochikoAsTech

(C) 2024 mochikoAsTech