テクニカルライティング本

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。「テクニカルライティング本」の筆者、mochikoAsTech です。

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。

- エンジニア
- 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

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

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。

- ドキュメントが上手に書ける

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

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、あなたと私の間で認識を揃えておきましょう。

1.1 テクニカルライティングとは

私たちは、毎日のように誰かに何かを説明しています。

- 部下が上司に業務でつまづいている部分を報告する
- 親が子供に今日一日のお出かけスケジュールを伝える
- 病院で看護師が患者に退院の手続きを説明する

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を「エンジニア」と呼ぶように、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人たちのことは「テクニカルライター」と呼びます。

【コラム】IT 系で増えてきたテクニカルライター

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」の別名は「マニュアルライター」と紹介されています。もともとは、家電や電化製品を買うと付いてくる、あの取扱説明書を書いている人たちがテクニカルライターでしたが、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、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。

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 つとして扱われ、がっ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るようです。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 年ごろから IT 系の企業でこのテクニカルライターという仕事をしています。

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かったのか、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じで「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。

エンジニアリングとライティングの両方が好きな方には、オススメのポジションです。

1.2 「文章を書くこと」と「テクニカルライティング」の違い

前述のとおり、分かりやすい実用文を書くための技術がテクニカルライティングです。ですが、日本で生まれ育った人であれば、日本語ネイティブとして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？

まず実用文は、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。

[*1] 職業情報提供サイト - 厚生労働省 https://shigoto.mhlw.go.jp/User/Occupation/Detail/358

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「長ければ長い方がいい」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無駄になが－－－－い文章より、短い文章でサクッと理解できる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が、一方、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。

- 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す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名前だけ聞くと、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ーなテキストを書いてもらう」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物と善し悪しの物差しが異なるため、よかれと思って指摘した内容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、贅沢で美味しい食事が「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。どんなシチュエーションでも、どんな人にとっても「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を覚えておいてください。

1.3 「知らない」と書けない

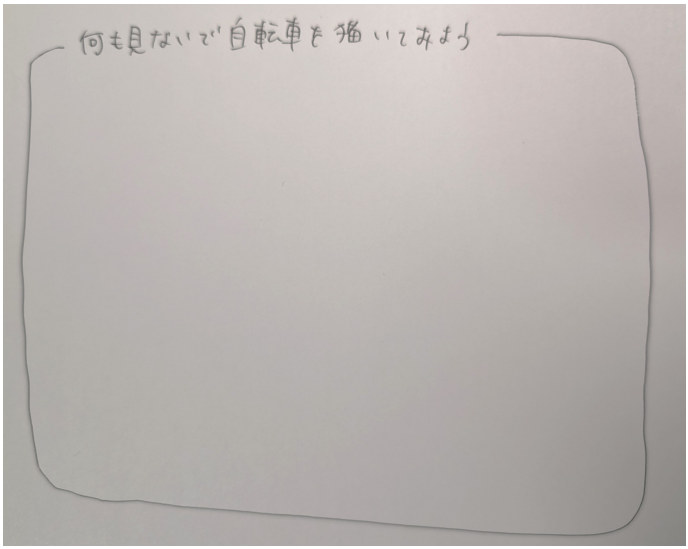
さて、話は変わりますが「自転車」を知っていますか？

何を唐突にと思われるかもしれませんが、「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。

殆どの人は「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。自転車は英語で言うと bicycle だということもきっと知っているはずです。

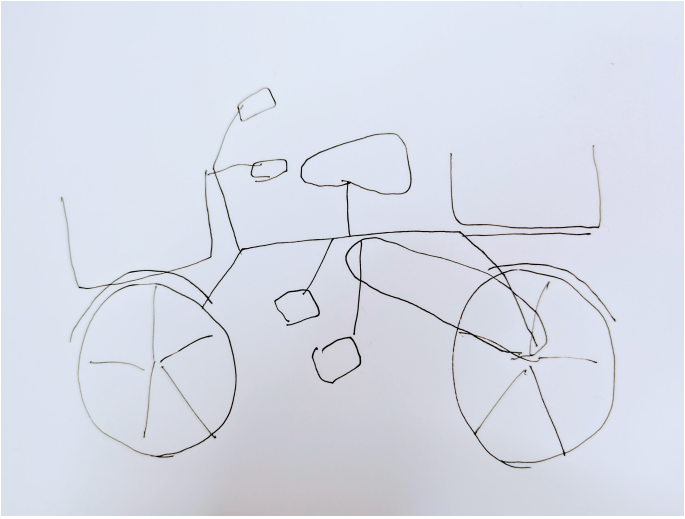
ではちょっと、自転車の絵を描いてみましょう。あなたがよく知っているはずの、

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）



▲図 1.1　自転車の絵を描いてみよう

自転車にはタイヤがある……大きいタイヤが 2 つ……その間をつなぐ。あとなんか太いパイプみたいなのがある……サドルがある……座るところ。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んだっけ？ 回る機構が必要だよな……どうなってたっけ？ チェーン？ とタイヤが繋がってる。チェーンって前輪と後輪どっちと繋がってたっけ？ 後輪かな……あとハンドル！ ハンドルがある。カゴもある……カゴどこについてたっけ？ あー、あとなんかタイヤの上に泥よけがあったかもしれない。後ろにもカゴが……あれ？ 自転車ってこんなだっけ？ 自転車って……どんな形だっけ？？（図 1.2）



▲図 1.2　筆者が描いた自転車の絵

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。うまく書けましたか？

ロードバイクが好きで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んなものをよく観察している人でなければ、大体こんな感じの「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、私たちは自転車の絵が思っていたよりうまく描けませんでした。

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 つ組み合わせた平行四辺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。

次に椅子に座ったときの自分の様子を思い浮かべてみましょう。椅子に座ったとき、膝は座った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、操作が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曲がっており、それによって舵取りをしやすくしています。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。

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。絵は、対象物をよく知らないと描けません。同じように、文章も、対象物をよく知らないと書けないのです。

「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せんか？ 描けると思ったのに書けないのは、あなたが実際は「知らない」からです。知らないまま、知ったかぶりをして、あるいは知っているつもりできれいな文章を書いても、それは分かりやすい文章にはなりません。人体の構造を知らない人が作った服は、ぱっと見がすてきでも実際はとても着用に堪えないように、実用文としては役には立たない文章なのです。

上手な料理を作るには、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、腕より先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、先に入力が必要なのです。「全然分からない……何も分からない……」という混乱の森を抜けなければ、理解にはたどり着けず、対象を理解していない人間が分かりやすい説明など書けないのです。

1.4 理解の解像度が低いとやっぱり書けない

既に使ったことがあり、分かっている技術であっても、理解の解像度が低いとやはり筆は進みません。

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、作業に対する理解度が「料理なんかやったことないけど、ハッシュドビーフは食べたことある」という理解度の人と、「ハッシュドビーフは知らないけど、要は材料を買って調理して洗い物をするんだろうな」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ゥを入れる料理だな。赤ワインを入れることもある。後片付けのときは油汚れをお湯でよく流して洗わないな」という理解度の人だと、最終的に出てくる手順書の精度はかなり違ってきそうです。

十分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うまく伝えられない、と思っている人は、実は機能として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。

ここを書こうとすると、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」っとした部分だけど恐らく理解の解像度が低くなっています。ライ

ティングの技術を磨くよりも、対象の理解に努めることが正解かもしれません。

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「知らないものについて、分かりやすい文章は書けない」ということと、自分が「知っている」つもりでも意外と対象物を知らないこと、そして知らなかったり、理解の解像度が低かったりすると書けなくなってしまうということを最初に知っておいてください。

1.5 まずはいっぱい読むことが大事

文章がうまく書けない場合、細かいライティングのテクニックを覚えるより前に、まずは「いっぱい読むこと」をオススメしています。

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な」のように、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きます。

レゴブロックと同じで、少ないパーツでなん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。

1.6 いっぱい書くことも大事

たくさん読んでパーツを集めるだけでなく、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ことが上達への近道です。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、こういうパーツが全然足りてない！」に気付いて、また読む作業に意欲的になれます。

「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？」と聞いて来た人が、もしまだライティングに関して初心者だったら、私の答えは「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。

1.7 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で、「完成させること」です。中途半端に書きかけの下書きがブログにいっぱい溜まっていて、構想ばかりで完成させられない状態だとうまくなりません。

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、「これでよし！」と完成させられるほど我々の精神は成熟していません。あれ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、全然足りないんだけどこれを完成とする！ と開き直るためには、自分ではどうにもならない〆切が必要なのです。

「○○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。なぜなら書くことではじめて「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。

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 10 階に行きたいんだ」と言われても、2 階や 3 階を経ずに 10 階にはたどり着けないのと同じで、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。

第 2 章

エンジニアのためのテクニカルライティング

文章を書いているエンジニアのみなさんへ仕様書、設計書、技術ドキュメント、会社やチームに新しい人が入ってきたときのための技術研修資料、オンボーディング資料会社あるいは個人の技術ブログ、Qiita、Zenn、技術書や技術雑誌の記事みなさん、やっぱり技術に関する文章は、色んな場面で書いている

文章を書いているエンジニアからテクニカルライターによくいただく質問：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？

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

2.1　先に大枠を説明して段々詳しくしていこう

まずは大枠を説明しましょう。先に細かい部分を話し出すと、全体像が分かっていないので混乱します。

ババ抜きの説明をするなら、「トランプというカードを使って 2 人以上でやるゲーム」辺りから説明をはじめます。いきなり「順番にカードを抜いていく」とか、「ババを引いたらだめ」といった細部から話すと、ババ抜きを知らない人は混乱します。

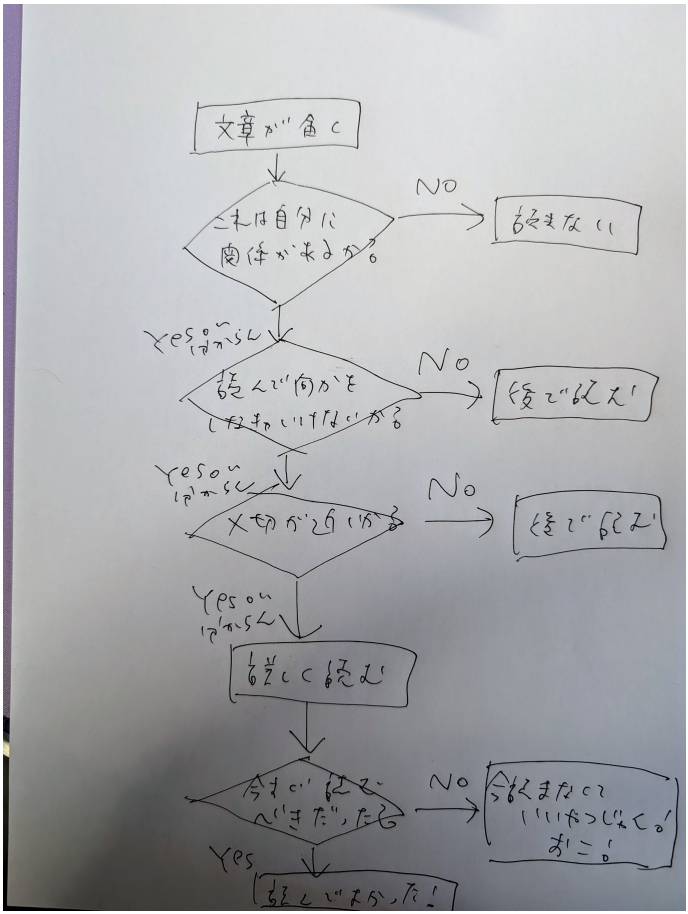
2.2　最後まで読んでもらえない方がいい、こともある

その人が問題を解決できればいい、答えを見つけて手を動かす作業に戻れればいいので、全文を最後まで読まれないことが正義のこともある。

最短経路で「ここには欲しい情報がない」「自分はこれを読まなくていい」を判断できるようにしておこう。

2.3 早めにふるい落とそう

100 人の読み手がいて、ここに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しまうと相当な労力が無駄になる。(図 2.1)



▲図 2.1　読む読まないの離脱フロー

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の

人が該当する分岐」でふるい落とそう。

読み進むべきか、読まなくていいのか、早めに判断させてあげよう。

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 9 割が離脱できる。

みんなの時間を無駄に使わない。

2.4 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

2.5 何をしてほしいのか最初に書こう

これを読んで、いつまでに何をして欲しいのかを最初に書こう。

「急に知らせてびっくりしないよう、事前告知で概要を先出ししてるだけなので、現時点でやって欲しいことは特にないよ」を最初に書いておく。

2.6 要約文ではじめよう

最初に要約文を書いておく

2.7 既知から未知に繋ごう

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こう。

2.8 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

2.9 勘違いを捨てよう

次のような勘違いを捨てて、

- 私が読んで分かるんだから、他の人が読んでも分かるはずだ
- ちゃんと読んでくれないから伝わらないんだ

2.10　一度に把握できることは 7 つまで

短期記憶は 7 つまで

2.11　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

2.12　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バ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

2.13　漕ぎ始めを生成 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漕ぎ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

2.14　意味のない空白や意味のないスペースを残すな

「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

2.15　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「事前告知をした」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

2.16 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ちがたく 1 つになっていると後で再加工が辛い。

2.17 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。

2.18 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

2.19 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。

2.20 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

2.21 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

2.22 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！ 分からないから説明読むんだけど、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。

2.23 年月日を書け

すべてのドキュメントは、文脈とか前後とかを把握されずに突然そこだけ読まれると思った方がいい順を追って読んでほしい、前後をコントロールしたいなら「本」にする

誰に向けた文章なのか、読むとどうなれるのか、冒頭にゴールを書いておく

2.24 同じものは同じ名前で呼ぼう

いままでのものが「A」に変わったら、「いままでのもの」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。

2.25 正しい名前で呼ぼう

iPhone を iphone と書くとか、JavaScript を java script と書くとか、

2.26　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。

2.27　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。

2.28　「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。

2.29　一文を短くしよう

「〜ので」「〜し」
文章が長くなるときは、箇条書きを使って分けてみよう。表を使ってもいい。

2.30　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。

2.31　分かったと分からない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

2.32 結論を先に言おう

結局何をすれば良いの？ 棚卸しのお願いというメールが来て、対象となる資産の一覧や、自分が貸与されている資産の一覧を見る方法は書いてあるが、結局いつまでになにをすればいいのか、なにをやったら終わりなのか、自分は対象者なのかが分からない。

今すぐ何かをして欲しいのか、今やれることは何もないから事前告知として聞くだけ聞いておいてほしいのか、何かしら準備をして欲しいのか、これを「どういう気持ちで読めばいいのか」を先に説明して欲しい。なんだか言わずにいきなり「食べて！ほら食べて！」とスプーンを差し出されると、「え、なになになに？」となって怖い。

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ない

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」があるのだ。

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 分とちょびっとだから 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

大さじ 1 杯弱も「大さじ 1 杯＋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ので解くのが難しい。

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 時間弱というのは本来は 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あります。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が、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」ときは「張る」だと思っています。

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

第 3 章

英語を書くとか英訳するとか

日本語話者のエンジニアが英語を書くときにやりがちな失敗

3.1 例は「ex.」か

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。

3.2 単語や文の単位で翻訳すると危ない

AWS が提供している日本語のドキュメント[*1]に、こんな一文がありました。

> インスタンスを削除するということは、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[*2]みたいになっています。どうしたのでしょうか。

原文と思われる英語[*3]を当たってみると、どうやら Terminate と Delete を両方と

[*1] 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

[*2] 「今のままではいけないと思います。だからこそ、日本は今のま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%80%B2%E6%AC%A1%E9%83%8E%E6%A7%8B%E6%96%87

[*3] Tutorial: Get started with Amazon EC2 Linux instances https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

も「削除」と訳したことで生まれてしまった悲しい文章だったようです。

```
Terminating an instance effectively deletes it; you can't reconnect to an insta
```

「インスタンス」は AWS におけるサーバのことです。なので恐らく英語で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ね」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。[*4]

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」と書い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推察します。

この注意喚起文によって何を食い止めたかったのか、という背景を理解せずに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 前後の文脈やどこに書かれる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] 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。[*5]

3.4　スペルミスが心配なら ATOK や辞書に頼ろう

ATOK つかうか辞書を引こう。人間は information をうっかり複数形のつもりで informations にしたりする。

[*5] 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うのであった。 https://twitter.com/mochikoAsTech/status/1779009150426767775

3.5 英単語や数字の後ろに半角スペースは入れるべきか

3.6 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、しれっと真逆の意味に翻訳したり、間の文章を一文落としてきたり、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）。

3.7 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」

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。どれかひとつしか手に入らないなら、絶対に技術力を優先して翻訳者を選ぶべき。

翻訳する人は「文脈が分かっている」と強い。そもそも何の話か分かっているから、原文が多少説明不足でもピントの合った訳ができる。技術ドキュメントを書くとき、あるいは翻訳するとき、「何を言っているのか」が分からないと、文法があっているだけで正しいことを何も伝えない、きれいで誤った文章になる。

3.8 翻訳は「もとの言いたいことに立ち返ってから」

伝えたいことと、実際に発露した文章の違い発露した文章を元に翻訳しようとすると元のデータ量が減ってるので誤訳が生まれる

言いたいことがあって、そこから原文が生まれる、その時点で情報量は減っている。JPEG みたいに、保存のたびに情報量が減衰していくので、元の「言いたいこと」に立ち返らずに翻訳をすると、原文から翻訳文になるときさらに情報量が減衰する。文章としての体を為すため、その減衰を補おうとして、結果嘘が混ざる。

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

よい README を書く

4.1 Markdown の記法を知って書こう

たとえば GitHub には、注釈を書くための Markdown 記法が存在します。

> [!NOTE] > Useful information that users should know, even when skimming content.

> [!TIP] > Helpful advice for doing things better or more easily.

> [!IMPORTANT] > Key information users need to know to achieve their goal.

> [!WARNING] > Urgent info that needs immediate user attention to avoid problems.

> [!CAUTION] > Advises about risks or negative outcomes of certain actions.

https://docs.github.com/ja/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax#alerts

第 5 章

誤解を生まないテキストコミュニケーション

5.1　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

5.2　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、「なぜならば今晩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

5.3　「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。

5.4　Slack でのコミュニケーション

5.5　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

5.6 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。

5.7 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 どっちがいいですか？」

5.8 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 でどんどんアウトぷっとしていく。実況中継しながら作業したり、分からないことを「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 6 章

レビューする、レビューされる

6.1　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。

6.2　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。

6.3　他人が書いたものに敬意を払おう

色ん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

6.4　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。

6.5 レビュアーが気付いたことに、書いたとき気付かないのは当然

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 人いる幼児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ったんだ……」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。

6.6 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、このままでもいいけど気になったので一応伝えておくもの

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。

6.7 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なてにおはの誤りがないか
- 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

6.8 大量に赤を入れることがレビュア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ア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。

6.9 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ない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ない、という無限レビューはレビュイーの心を折ります。[*1]

レビューをするとき、レビュアーは、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。

6.10 指摘は素直に受け入れる

レビュイーは指摘を聞くと、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 7 章

どうやって学ぶか

7.1 TC 検定を受けよう

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=N9OS4BtDHik&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）

7.2 色んな本を読む

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。

7.3 「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/bunkashingikai/kok
公用文作成の考え方（建議）（PDF）https://www.bunka.go.jp/seisaku/bunkashingikai/kokug

7.4 Technical Writing Meetup に参加する

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

おわったら書く

2024 年 5 月
mochikoAsTech

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

- 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ター協会
  - `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―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`

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

挿絵デザインを担当。

Shinya Nagashio

表紙、章扉、目次デザインを担当。

テクニカルライティング本
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