

大扉のタイトル

2023-8-13 物理本 ver1 親方 Project 発行



まえがき

この本を手に取っていただき木、ありがとうございます。

あとがきあとがき

2023年8月編集長 おやかた@親方 Project 拝

免責事項

- 本書の内容は、情報提供のみを目的としております。著者一同、正確性には留意しておりますが、正確性を保証するものではありません。この本の記載内容に基づく一切の結果について、著者、編集者とも一切の責任を負いません。
- 個別の事象についてのアドバイス等はお受けできません。あくまで著者の体験を記述しているものであり、一般に適合しない事例もあります。
- 会社名、商品名については、一般に各社の登録商標です。TM表記等については記載しておりません。また、特定の会社、製品、案件について、不当に貶める意図はありません。
- 本書の一部あるいは全部について、無断での複写・複製はお断りします

目次

まえがき	2
免責事項	2
第Ⅰ部 仕事での学び	13
第1章 チーム開発におけるコミュニケーションの学び	15
1.1 はじめに	15
1.2 コミュニケーションの取り方の原則	15
1.3 リーダーからメンバーへのコミュニケーション	16
1.4 メンバーからリーダーへのコミュニケーション	20
1.5 メンバー同士のコミュニケーション	21
1.6 コミュニケーションは最初が肝心	22
1.7 さいごに	23
第2章 実務で通用する設計・プログラミングスキルの学び方	24
2.1 「実務で通用する」の定義	24
2.2 実務で通用するスキルの学び方	26
第3章 特許について	29
3.1 特許のイメージ	29
3.2 特許には何が必要か?	29
3.3 似たような課題だって特許になる	30
3.4 まとめ	30
第4章 I型人材、T型人材、π型人材とは	31
4.1 人材のカタチ	31
4.2 π型人材になるために	32
4.3 結論	33

第Ⅱ部 学校での学び	34
第5章 放送大学に入学した話	36
5.1 序文	36
5.2 放送大学入学	36
5.3 入学そして受講	37
5.4 結果	38
5.5 良かったこと	38
5.6 悪かったこと	38
5.7 まとめ	39
第6章 放送大学で認定心理士を取った	40
6.1 認定心理士とは	40
6.2 目標	40
6.3 結果と感想	41
第7章 放送大学で情報学を体系的に学ぶ	43
7.1 はじめに	43
7.2 文系エンジニアが放送大学で学び直しを決意した瞬間	43
7.3 情報学を体系的に学べる方法	43
7.4 放送大学以外の選択肢	44
7.5 放送大学で情報学を学ぶメリット	44
7.6 仕事と放送大学での学びを両立するために気をつけること	45
7.7 放送大学で勉強するにあたっての Q&A	46
7.8 さいごに	46
第8章 小中高の学びは重要	47
第9章 スポーツ嫌いを量産する方法	48
第10章 学びのために教育学の基本を勉強してみた！	49
10.1 一番「学び」を研究していくような分野を勉強すれば学びを得られるのでは？	49
10.2 本章で取り上げる教育学的なアプローチ	49
10.3 4段階教授法	49
10.4 5段階教授法	51
10.5 ツイラーの5段階教授法は4段階教授法の順当な発展形	51
10.6 ラインの5段階教授法は教師側に対する論理	52
10.7 9教授事象	53
10.8 プログラム学習の5原理	54
10.9 問題解決学習	55

10.10	まとめ：これをもってどうしていくか	56
第 11 章	大学と大学院の学びの違い	57
11.1	大学で学ぶことは基本的には専門性の向上である	57
11.2	大学院で学ぶことは「問題解決の手順」である	57
11.3	それは当然仕事でも役に立つ	57
第 12 章	学ぶ手法（資格・スクール・自習）のトレードオフについて	58
12.1	自習する	58
12.2	スクール	59
12.3	資格	60
12.4	おわりに	62
第 III 部	学びの体験談	63
第 13 章	IT エンジニアが初めての副業で得られた学び	65
13.1	やったこと (Y)	65
13.2	わかったこと (W)	73
13.3	次にやること (T)	74
13.4	そもそも、なぜ副業を始めようと思ったのか?	76
13.5	さいごに	76
第 14 章	お出かけから得られた学び（仮）	78
14.1	どうやってお出かけする機会を作るか	78
14.2	お出かけする	81
14.3	具体的なお出かけ先紹介	81
14.4	お出かけから得られた学び	83
14.5	選択肢という手札を増やそう	84
第 15 章	夏休み・春休みに高度情報試験を取った話	85
15.1	バックグラウンド	85
15.2	結果	87
第 16 章	緑内障になってから変わった心境とインプット方法	88
16.1	はじめに	88
16.2	少し変わってきた生活習慣	92
16.3	さいごに	100
第 17 章	まわりまわって繋がった話	101

目次

第 18 章 何を学べばいいのかわからなかった自分の生存戦略	103
18.1 自分のバックグラウンド	103
18.2 なりたいものがわからなかつたので、とりあえず学ぶ	103
18.3 広く学んできたものがようやく活きてきた	104
18.4 これから何を学んでいくものがわからない人へ	105
第 19 章 少しづつでも毎日続けてみる	107
19.1 移動時間を活用する	107
19.2 注文の待ち時間を利用する	108
19.3 ブログを書く	108
19.4 まとめ	109
第 20 章 はんだ付けを学びました	110
20.1 はじめに	110
20.2 どうやって学んだか	110
20.3 実践！！	111
20.4 終わりに	111
第 21 章 キーボード布教の過程で学んだこと	112
21.1 はじめに	112
21.2 学んだこと	112
21.3 おわりに	113
第 22 章 技術同人誌の執筆を通して得られた学び	114
22.1 学びについてまとめる前に	114
22.2 継続に関する本の執筆に関する学び	114
22.3 まとめ	115
第 IV 部 コミュニティでの学び	117
第 23 章 技術書同人誌博覧会の次回案内デザイン制作で画像生成 AI を活用する	119
23.1 はじめに	119
23.2 完成イメージと素材選定	119
23.3 Adobe Firefly を活用してお目当ての画像素材を得る	120
23.4 実際にデザインを作成するプロセスに関する解説	122
23.5 まとめ	123
第 24 章 己の学びのために地域技術コミュニティ立ち上げた話	124
24.1 俺は別に東京に行きたいわけではない！！	124

24.2	立ち上げたコミュニティはどんなものか？	124
24.3	実践しているコミュニティ運営指針の紹介	125
24.4	地域コミュニティでやっていること	126
24.5	コミュニティの今後	126
第 25 章 横断的な知識をまとめる本を作つてみた。そこでの学び		128
第 26 章 思考の限界と言語化		130
26.1	私たちの思考	130
26.2	エモいのは非	130
26.3	怒りの感情を区別できない人たち	130
26.4	いい言語化とは	130
26.5	スポーツの熟達と言語化	131
26.6	権威がある人からの発言をそのまま受けることは非	131
26.7	感想	132
第 27 章 LT の実況で学びを最大化する		133
27.1	学びの最大化	133
27.2	その他のメリット	134
27.3	まとめ	135
第 28 章 イラスト制作・デザインにおける個人作品と依頼品の違い		136
28.1	違いの早見表	136
28.2	違いの大前提	136
28.3	制作工程の違い	137
28.4	制作手段の違い	140
28.5	制作をスムーズにするための工夫	141
28.6	まとめ	142
第 29 章 「ワンストップ学び」表紙作成秘話		143
29.1	依頼内容	143
29.2	ラフ提出	143
29.3	修正・ブラッシュアップ	144
29.4	キャラクターデザイン	145
29.5	CLIP STUDIO の 3D 機能を使用した画面構成	146
29.6	完成	147
第 30 章 執筆を促進するために Slack 通知をつくつてみた		148
30.1	状況を把握するためのアプローチ	149
30.2	ワイワイ感を出すためのアプローチ	149

目次

30.3 得られた学び	150
第31章 場がオープン／クローズドであることにより、得られる学びの性質は変わる	151
31.1 オープンコミュニティについて	151
31.2 クローズドコミュニティについて	152
31.3 おわりに	153
第32章 興味関心を持つために情報や人の近くにいる	154
32.1 はじめに	154
32.2 興味関心を持つことが学びのスタート地点	154
32.3 負の状態から脱却するのは新しい興味を見つけた瞬間	155
32.4 具体的に私が人や情報で学びに繋げられた事例	155
32.5 まとめ	156
第V部 学びのTips	157
第33章 メタ認知のすゝめ	159
33.1 メタ認知とは	159
33.2 メタ認知できていない人	161
33.3 さいごに	166
第34章 やる気に頼らず学ぶ、自分プログラミングという考え方	167
34.1 はじめに	167
34.2 そもそも「やる気」とは？	167
34.3 自分プログラミングの考え方	168
34.4 終わりに	169
第35章 学びの時間を最大限増やす方法	171
35.1 お風呂時間に学ぶ！	171
35.2 睡眠時間に学ぶ！	173
35.3 勤務時間に学ぶ！	175
35.4 さいごに	177
第36章 受け身でも持続的に学べる環境作り	178
36.1 受け身で気が向いたら学ぶことを良しとする割り切り	178
36.2 欲しい情報を整理し、どこで手に入るか調べる	178
36.3 受け身で情報を得られるように仕組み化する	179
36.4 仕組み導入後の効果	180
36.5 まとめ	180

第 37 章 ChatGPT を使った文章の書き方の一例	181
37.1 ChatGPT は文書作成の強力なツールである	181
37.2 AI 良い文章になる	182
第 38 章 学びと言語化	184
38.1 言語化とは何か	184
38.2 言語化の重要性：プログラミングの例から	185
38.3 言語化をやってみよう	186
38.4 言語化の困難さと限界	187
38.5 学びのコミュニティにおける言語化の役割	188
38.6 言語化のさらなる練習方法：ブログを始めよう	188
38.7 まとめ	190
第 39 章 学びのプロセスを詳細に分解する	191
39.1 学びのサイクルを理解する	191
39.2 メタ学び：学び方を学ぶ	193
39.3 例をもとに学びのプロセスを理解する	193
39.4 学びの共有と発信：知識の循環	194
39.5 まとめ	195
第 40 章 学び hack	196
40.1 学びの複雑さを解きほぐす	196
40.2 あたりをつけて進める	197
40.3 経験と本質の両立	198
40.4 経験から学びを深める	198
40.5 モチベーションを持続させる	199
40.6 実践的な学びの例	199
40.7 失敗からの学び	199
40.8 コミュニティでの学び	200
40.9 振り返りの重要性	200
40.10 まとめ	200
第 41 章 自己変容を受け入れる	201
41.1 学びにおける自己変容	201
41.2 自己変容のプロセスと段階	201
41.3 自己変容の個人差を理解する	202
41.4 メタ認知と自己変容	203
41.5 自己変容を阻む要因	203
41.6 大きな自己変容を受け入れる	204

目次

41.7	自己変容を促進する環境づくり	205
41.8	自己変容の多様性を理解する	205
41.9	自己変容がうまくいかないとき	206
41.10	まとめ	207
第 42 章 生成 AI を活用する		208
42.1	生成 AI とは	208
42.2	対話型 AI	209
42.3	プロンプト	209
42.4	ChatGPT, Claude, Gemini	210
42.5	対話型 AI に文章を書かせてみる	212
42.6	活用事例: 文章を読ませる	216
42.7	活用事例: ドキュメントを元に音声を作る	220
42.8	活用事例: 問題を作ってもらう	220
第 43 章 自分をメンテナンスするために月に 1 回有給を取ろう		224
43.1	土日が家族対応でつぶれる現実	224
43.2	平日は仕事で忙しい	224
43.3	自分のために月に 1 回、有給を取ろう	225
43.4	月 1 回の休暇がもたらす効果	225
43.5	最後に	226
第 44 章 HP と MP を別々に回復させよう		227
44.1	精神力 (MP) の回復と推し活	227
44.2	体力 (HP) の回復には何が必要か?	227
44.3	何もしない日も大切	228
44.4	ちゃんと寝る	228
第 45 章 幅を広げる学習のススメ		230
45.1	深く学ぶか広く学ぶか	230
45.2	なぜ広く学ぶのか	230
45.3	知識の幅を広げるきっかけづくり	232
45.4	知識を広げたあとは?	233
第 46 章 学べる環境に身をおいてしまえば学ばざるを得ず、強制的に学べる		234
46.1	はじめに	234
46.2	持株会の何がそんなに良かったのか	234
46.3	「これから上がる株」みたいな本一杯あるけどさあ	235
第 47 章 資格を学びに組み込むと段階的に学びやすい		236

47.1	簿記	236
47.2	リテールマーケティング（販売士）	238
47.3	技術的な国家資格についても確認してみよう	239
47.4	おわりに	241
第 48 章 情報処理技術者試験を俯瞰する		242
48.1	ITSS の各レベルと情報処理技術者試験の対応	243
48.2	ITSS レベル 1 の試験	243
48.3	ITSS レベル 2 の試験	244
48.4	ITSS レベル 3 の試験	245
48.5	ITSS レベル 4 の試験	246
48.6	おわりに	247
第 49 章 聴いて学ぼう		248
49.1	聴いて学ぶ際のメリット・デメリット	248
49.2	聴いて学ぶための手段	248
49.3	おわりに	250
第 50 章 資格試験を受け続けて気付いた事		251
50.1	資格試験を申し込んでから受験対策を考えても結構間に合う	251
50.2	いっぱい〆切ができると結構学べる	251
50.3	色々な資格試験を受けると自分の適性が見えやすいかも	251
50.4	色々な資格試験を受けると、1+1 は 2 ではなくなる	252
50.5	資格を色々拾って嬉しかった事	252
50.6	最後に、イテレーティブに受験し続ける際の最大のコツをお教えします	253
50.7	おわりに	254
第 51 章 良いアウトプットのためのインプットとノート整理		255
51.1	はじめに	255
51.2	進める際の個人的ポイント	255
51.3	実践したメリット	256
51.4	まとめ	260
第 52 章 認知のベンチマークをしよう		261
52.1	他者の認知と比較する	261
52.2	まとめ	262
第 53 章 効率よく学ぶには資格を取る		263
第 54 章 「読みたい」と「書ける」は違う		264

目次

第 55 章 2ミリの学びを捉える	265
55.1 はじめに	265
55.2 『学び』とは何を指すのか	265
55.3 『差分』の解像度を上げる	265
55.4 終わりに	266
第 56 章 学びと言ってもいろいろありすぎる～	267
あとがき	269
著者紹介	270
編集長	270
著者	270
表紙イラスト・デザイン	272

第Ⅰ部

仕事での学び

毎日やっている「仕事」は、これまで学んだことを発現する場であります。その一方で新しい刺激を受け、日々学ぶ場であります。

新しく学ぶチャンスでもあり、学びをより洗練させるチャンスであります。

そんな仕事における学びについて、集めてみましょう。

本書の読者は、今「働いている人」が多いでしょう。この本を読んで、明日から、明後日から、何か役に立つたら幸いです。

チーム開発におけるコミュニケーションの学び

1.1 はじめに

FORTE(フォルテ)@FORTEgp05

どうも、本章の執筆者の FORTE(フォルテ) です。私は IT エンジニアとして 16 年目に入り、さまざまな現場でチーム開発を経験してきました。その中でチーム開発におけるコミュニケーションの取り方について学ぶところが多かったので、今回は「チーム開発におけるコミュニケーションの学び」と題して 1 章書きたいと思います。

なお、本章の内容は個人的に Zenn にアップした以下の記事を元に大幅に加筆・訂正した内容となっています。余談ですが、この記事をアップした際のツイートは私の中で最大のバズりとなりました。

<https://zenn.dev/forte/articles/822c75cbb53519>

業務でシステム開発をするとなると、個人で開発するケースもあるかとは思いますが、多くはチーム開発になるでしょう。そこで避けては通れないのが、チームメンバーとのコミュニケーションです。このコミュニケーションは大きく次の 3 つに分かれると思います。

- リーダーからメンバーへのコミュニケーション
- メンバーからリーダーへのコミュニケーション
- メンバー同士のコミュニケーション

ここではリーダーと表現しましたが、マネージャー、エンジニアリングマネージャー、プロジェクトマネージャーと置き換えて大丈夫です。

それぞれのパターンにおいて私が経験してきた気をつけるべきポイントと、どうすれば良いのかを紹介していきたいと思います。

1.2 コミュニケーションの取り方の原則

まず前提としてどのパターンでも通じる原則があります。それは謙虚 (Humility)、尊敬 (Respect)、信頼 (Trust) の 3 つの精神です。これは HRT(ハート) と呼ばれ、Team Geek という Google でリーダーを務めた IT エンジニアが書いた本に登場した概念です。それぞれ次のような概念となっています。

- 謙虚 (Humility)
 - 驕り高ぶらず、自分が常に正しいとは限らないことを念頭に置き、常に自分を改善する姿勢でいること。
- 尊敬 (Respect)
 - 他のチームメンバーを心から思いやろう、相手はコンピューターできなく尊敬すべき一人の人間であり、その能力や功績を高く評価すべきだ。
- 信頼 (Trust)
 - 他のチームメンバーは有能であり、正しいことをすると信じることで、積極的に権限以上を行おう (ただし無能な人には気をつけて)

これらを念頭に置いて行動することで多くの問題は解決できますが、良かれと思って行動したことが裏目に出たり、他のメンバーが HRT を尊重しない場合もあります。原則はあくまで原則であり、実際の現場で起きていることに適応しないと活かせないのです。次から 3 つのパターンごとに実際に起きた事例と対処方法を紹介していきます。

1.3 リーダーからメンバーへのコミュニケーション

1.3.1 メンバーを否定しないで肯定する

例え内心（それはないな…）と感じても表に出さず一度肯定してから伝えることが重要です。当たり前に思えて意外とできない人も多いと思います。特に意識していないと自分の率直な感想が出がちですが、そうするとメンバーが否定された!と感じてしまうかもしれません。

例えば年次目標に 3 冊技術書を読むと言ったメンバーがいるとします。この 1 年で 3 冊は普段から本を読む人からすると正直少ないと私は思います。ですが、ここで素直に「少ないですね」と言ってしまうとメンバーが否定された!と感じてしまうかもしれません。「まずは今年 3 冊確実に、来年から量を増やしていきましょう」などのように表現すると良いでしょう。

他にも否定から入るのが口癖のようになってしまっている人もいると思います。メンバーから何か言われた時に「いや、それは…」とか、「そうじゃなくて、これは…」とかのように言ってしまうケースです。このような言動を繰り返しているとメンバーは何を言っても否定されると思い、そのうち何も言わなくなってしまいます。もし心当たりがある人は「確かにそうですね。でも他の可能性として…」の様に言い換えることを意識してみると良いでしょう。

1.3.2 聞かれない限りアドバイスしない

否定しないに関連して、メンバーが否定された!と思う行動のひとつにリーダーからの押し付けがましい身勝手なアドバイスがあります。オススメの本を紹介する、こういう行動を取ると良い、こういう考え方をすると良いとアドバイスするのも、メンバーにとっては自分の選んだ本を否定された!自分の考えを否定された!となる可能性があるので、関連した話題が出てもメンバーから明確に聞かれない限り、または質問されない限りアドバイスしない方が賢明です。

ちなみに私の経験としてアドバイスしたら自分の考えを否定された!と感じたメンバーはチームリーダーになりたいというキャリア志向の持ち主であり、上長からリーダーについてアドバイスをして欲しい、次のキャリアとして後進育成もやって欲しいと言われた中でアドバイスをした結果、メンバーには否定された!と感じられました。このように上昇志向がありキャリアアップを望んでいる人に対するアドバイスでもメンバーからすると否定された!となってしまう可能性があるのです。そのため、上長からの依頼であっても、良かれと思っても、本人が望んでいても、明確に求められない限りアドバイスはするべきではないでしょう。

これは後にも書きますが、チーム結成時、そして定期的に希望を確認すると回避できるでしょう。本人がアドバイスを望むのか?望むのならどんなアドバイスを望むのか?頻度やきっかけは?こういった項目を事前にメンバーと擦り合わせておくとメンバーの否定された!という思いを防ぐことができます。

1.3.3 こちらに非があれば謝罪する

これも当たり前に思えて、会話の流れで謝罪せずに終わらせてしまうと後々問題になることがあります。また、リーダーが「そんな細かいこと…」と思っていてもメンバーは気にしているかもしれないで、きちんと謝罪する癖をつけておきましょう。

とはいっても謝罪しても仕方ないので、ポイントとしては指摘をした際が重要です。リーダーとはいっても人間ですので勘違いもあれば、ミスもあります。その際、間違った指摘をしてしまったときはきちんと謝罪をするようにしましょう。

ここで最初に書いた会話の流れで謝罪せずに終わらせてしまうケースが発生し得ます。例えばメンバーが書いたコードを見ながら会話していて「ここ間違ってるよ」と指摘したのに設計書通りで正しかったという場合に「あ、私のミスだった。で、次は～」「ごめんね、で次は～」「すまん、違った。で、次は～」のように明確に謝罪しなかったり、しても流れで謝罪しているとリーダーが非を認めなかった!謝罪がなかった!とメンバーが感じてしまう可能性があります。ここはきちんと「すいません、私が間違ってました。」と謝罪するようにしましょう。

1.3.4 指摘をする際にはエビデンスを用意してから指摘する

一つ前の話の防止策にもなるのですが、メンバーに対して何かしらの指摘をする際にはコード、設計書、テスト結果などのエビデンスを用意しましょう。なぜならば、その指摘内容に誤りがあると謝罪が必要になり、うまく謝罪できないとメンバーの不信感につながります。そのため、指摘する前にエビデンスを用意してから指摘しましょう。もし会話の最中などで気づいた場合は「確認したい点がある」と言って一緒にエビデンスを確認してから指摘するようにしましょう。

1.3.5 無理難題を言われても断らず、かと言って確約もしない

チームメンバーからこうして欲しいという要望があっても、様々な事情でそれを叶えられない場合があります。そういった場合に詳細に理由を説明し、難しい、できないと回答しても、メンバー

からするとリーダーがこちらの要望を聞いてくれない!という感情につながりかねません。

かといって、できもしないことを約束するのは悪手です。その場は収まっても後で約束が守れないとなった場合に問題が悪化するからです。また、過去にこの件についてTwitterで聞いてみたところ、リプくださった2名の方はできもしないことを約束されるより、詳細に理由を説明してほしいとのことでした。のことからも空約束するよりも詳細に理由を説明した方が無難ではあります。

そこでまずは「理由があって難しいですが、実現できるように努力します。その理由としては○個あって1つ目は~」のようにメンバーの意に沿う用にしつつ、できない理由を説明しましょう。そして最後に「できれば要望を叶えたいと思っているので努力しますが、どうしても難しい場合はご理解ください」とまで言えばメンバーの納得度も高まるでしょう。さらにこの説明は口頭で行った場合は文章にしておく(メールやチャットで送っておく)と安心です。あとで蒸し返された際に説明を行ったエビデンスとなります。

1.3.6 プラクティスを押し付けない、正当化しない

システム開発には様々なプラクティスがあり、不確実性の高いシステム開発に対応するには日々やり方を工夫しないとうまくいきません。ですが、チームメンバーはリーダーから振られたタスクをただこなせば良いと思っているケースがあります。そういったメンバーに、効率がいいから、あなたのため(教育、スキルアップ)になるから、などと言っても響かないですし、プラクティスに割く体力、時間を無駄だと感じことがあります。

そのため、リーダーが良いと思ったプラクティスでも、いくら現在のタスクをこなすために効率が良いプラクティスでも、メンバーが嫌がりそうなら多少効率が悪いやり方でも従来のやり方で進めるのが良いでしょう。

実際にメンバーから不評がられたプラクティスとしてはペアプロ、モブプロ、アイスブレイクがあります。

ペアプロは疲れるから嫌

これはリーダー側から定期的に休憩を取るようにすべきだったのを作業が進みすぎてしまったために休憩が取れず、拘束時間が長くなってしまったため、メンバーが疲れてしまったケースです。

リマインダーやタイマーでもセットして強制的に作業を区切ればよかったです、そういった工夫をしなかったためメンバーの疲労が溜まってしまう原因となりました。定期的に休憩を取れば解決する話ですが、ペアプロ、モブプロは進捗が良い分、作業を区切りづらいです。必ず最初に何時間やったら休憩というのを決めておき、タイマーなどをセットしておきましょう。一度疲れると認識されると二度とペアプロやモブプロをやりたくないと思われてしまいます。

モブプロで汗をかくから同じキーボードを使いたくない、実装中のコードを見られたくない

これはメンバーが手汗をかくため、同じキーボードを使い回したくないということがありました。それぞれがキーボードを用意してドライバー交代時にキーボードも交換することで解決します

が、本人は手汗を書くことを気にしているので根本的な解決にはなっていません。

また、メンバーによっては実装中のコードを見られたくないという人もいました。レビューはどうするのか?と聞くと我慢していると言っていたので、コードを見られるのが嫌という人もいるようです(正直チーム開発に向いてないんじゃないかと思いましたが、まぁそういう人もいるということでしょう)

早く作業したいのにアイスブレイクが長い

私のチームでは毎日の朝会で最初にアイスブレイクとしてハピネスレーダーをやっています。この時間が長いため、朝会が長くなり、作業時間が削られて嫌だ感じたメンバーがいました。こちらはリモート開発で雑談が少ないとへの対策、ルーティンとして毎朝行うことでモードを仕事モードに切り替えるなどの意味を持ってやっているわけですが、無駄だと思われているようです。

アイスブレイクにかかる時間としては5分以内、長くとも10分以内に終わっています。朝会自体も長くて30分です。メンバーが不満を感じた際のチーム人数は二人でした。これはメンバー側にアイスブレイクの意図が伝わっていない可能性がありますが、意図はちゃんと最初に説明しています。それでも不満に思うメンバーはいるので不満を感じていそうだなと思ったり、フィードバックを受けた際は押し付けないでやめたほうが良いでしょう。

ちなみにそのメンバーに朝会のふりかえりとして無駄な項目はないか?と聞いたら特に無いと言っていたので、アイスブレイクが無駄という不満を感じておきながら無駄な項目はないという矛盾した言動を取っています。このようにメンバーは本音と建前を使い分けるのでリーダーからすると本音を見抜くのは大変ですが、チーム開発を円滑に進める上で重要なポイントなので気を配りましょう。正直こういった気を使わなくても一緒に開発できるメンバーが欲しいのがリーダーとしての本音ですが、残念ながら多くの場合においてメンバーは選ぶことはできません。与えられたカードで勝負するしかないのであります。

1.3.7 みんながみんなアジャイルなどが好きな訳では無い

ひとつ前のプラクティスを押し付けないにも関わりますが、ITエンジニアのみんながみんなアジャイルが好きなわけでもないですし、スクラムやXPなどの開発手法に興味があるわけではありません。たとえアジャイルがどんなに素晴らしい考え方でもそんなことどうでもいい、割り当てられたタスクさえ終わればそれでいいと思っているITエンジニアもいるのです。

私はアジャイルやスクラムの考え方やが大好きなので、メンバーにもそう思って欲しいと考えてしまいますが、それは単なる押しつけにすぎません。開発を進める上で明らかにアジャイルが低い、問題があるとしても、それを理由にいきなり正論で、アジャイル棍棒で殴ってもメンバーが不満に思う可能性が高いです。特に「正しそうで反論しづらい概念」については注意が必要です。正論だから、こちらが正しいからという理由で自分の意見を押し付けていると相手の反論を押さえ込んでしまい、モチベを奪いかねません。正しいことを言うのが正しくないこともあるのです。

そこで、まずは他人を変えるのではなく自分で変えましょう、自分が変わりましょう。開発に問題があれば黙々と手を動かして解決する手段を構築すればよいのです。そして効果が出て感謝さ

れ、信頼貯金が溜まってきたら新しいプラクティスや考え方の提案をしましょう。信頼貯金がない中で提案しても採用されづらいですし、正論でかかっても反感を買うだけで上手く行かない可能性が高いです。それなら自分で手を動かしてまずは実績を作りましょう。

1.4 メンバーからリーダーへのコミュニケーション

1.4.1 可能な限り直接フィードバックをする

リーダーとはいえる人間です。当然たまにはミスや間違いを犯します。前に挙げたリーダーからメンバーへのコミュニケーションの内容についても、同じような不満を覚えた方もいらっしゃるでしょう。

そのときはできるだけリーダーに直接フィードバックしましょう。大体の場合、リーダーはメンバーより年上や経験が豊富な方がつくことが多いので、何か進言するのは憚られる、言いづらいから気が進まないとは思いますが、ここで言いやすいからと第三者を通してデメリットはあれどあまりメリットはありません。例えば次のようなデメリットがあります。

- 伝言ゲームになってしまふ
- アジリティが下がる
- 問題が大きくなってしまう
- チームが自立しなくなる

第三者を通した場合、リーダーの視点とメンバーの視点の違いを考慮しなくてはいけません。もしこの視点の違いを理解していない人を通してしまうと、言ってることが捻じ曲がってしまい、伝言ゲームの様になってしまいます。また言った言わないの水掛け論にもなりかねません。これは直接会話することで簡単に防げる所以メンバーやリーダーに対して直接フィードバックした方がメリットがあります。

また、第三者は公平な立場に立とうとする所以メンバーから言われたことを鵜呑みにせず、リーダーにも事情を聞こうとします。この時点でリーダーに対して会話する時間を調整し会話しと時間がかかることが多いです。直接会話すれば5分で解決するかもしれない話が、1時間、2時間とかかり、かつ、解決にも時間を要してしまい、チームのアジリティが下がります。

第三者を通すことでチーム内だけで完結できていたはずの話が組織、または会社全体の問題になります。その分、関わる人も多くなり多くの人の時間を奪ってしまいます。それが管理職としての仕事といえばそれまでですが、当人が会話して解決すれば無駄な時間を使わなくて済むのも事実です。

最後は一つ前にも関連することですが、チーム内の問題をチームで解決できないとなるとチームが自立しません。常に第三者がいないと成り立たないチームはすでに破綻していると言っていいでしょう。本来チームの問題はチーム内で解決すべきで、それが自立したチームというものだと思います。そのためにも対話が必要であり、フィードバックは直接行った方が良いのです。

フィードバックする際の工夫

フィードバックは問題や不満を感じたその場で行うのが最も効率が良いのですが、その場では言い出せず後々になってしまうことがあります。その際、今更言うのもなと言い出せないなあにしたり、言い出しづらいから第三者を頼ったりします。

なにもフィードバックは直接口頭で伝えるだけではありません。DM やメールなどの文章で伝えてもいいのです。その方が言いたいことが整理されますし、感情的にならず改善して欲しい点も伝えることができます。コミュニケーションがうまく取れないという自覚がある人は文章で伝える努力をしてみましょう。

またフィードバックする際に役立つ知識として NVC(非暴力コミュニケーション) が役立つと思います。NVC は相手や自身の気持ちを大事にして、誤解や偏見を見きわめて本当にやりたいこと、望むことを見出して自らの力で実現する原動力が生まれるコミュニケーションプロセスです。重要なのは自分と相手の気持ちを大事にする点で、フィードバックするにしても不満を言うだけではなく、自分と相手の気持ちを大事にしてフィードバックすることが可能になります。NVC はメンバーだけでなくリーダーにも有効な手法ですが、メンバーが心得ているとより円滑にチーム運営が進むでしょう。

1.4.2 逃げるべきこんなリーダー

チームのためにフィードバックしてほしいと書きましたが、リーダーによっては独善的で改善する気持ちがなく、またはハラスマントをするようなタイプの人もいるでしょう。そういったリーダーに対してまで直接フィードバックするべきとは言いません。なによりまず自分を守るのが大事です。チームはその後です。次のようなリーダーからはチーム外の第三者を頼ってでも抜け出した方がいいでしょう。

- 怒る、キレる、怒鳴る人
- セクハラしてくる人
- 常に正論しか言わない人
- 自分の経験しか信じない人

過去に直接関わったことはないですが、こういった言動をする人を見てきました。そしてそのチームにいる人はメンタルを病んでしまうことが多かったです。チームのために行動するのも大事ですが、あくまで自分あってこそそのチームなのでまずは自分を守ることを優先しましょう。

1.5 メンバー同士のコミュニケーション

1.5.1 相手の地雷に注意する

メンバー同士のコミュニケーションでありがちなのが、知らず知らずのうちに相手の地雷を踏み抜いていたということがあります。自分は問題ない言葉、語句、表現、やり方だったとしても相手

には問題がある可能性があります。そのため、コミュニケーションを取る際には言葉遣いに気をつけた方がいいでしょう。例えば次のようなことは要注意です。

- ・自分が嫌いな技術、苦労した技術をけなさない
- ・ビジネスライクにも法兰クにもなりすぎない
- ・バグやマイナスなコードをあげつらわない

一つ目は自分が嫌いなプログラミング言語、技術や、過去に苦労した技術をけなさないというものです。過去に自分が経験したこととしては、とあるプログラミング言語に対してツライ点があると Slack に投稿したところ、その Slack にはそのプログラミング言語が好きな人が多く、そういう不満は見てる人に嫌な思いさせないようにすべきという投稿がされたことがあります。自分が嫌いでもツライ経験があってもチームメンバーもそうとは限らないので、愚痴や不満は安易に投稿すべきではないでしょう。

昨今はリモートから出社に回帰し始めていますが、それでもテキストコミュニケーションが主流になっていると思います。そこで投稿するテキストがビジネスライクすぎる、つまりいつも文末が句読点で終わっていたり、感情が感じられないリモートに慣れてない相手だと怒ってる、機嫌が悪いと勘違いされることがあります。仕事なのであまり法兰クな表現は使うべきではないですが、!や～などは使っても問題になりづらいでしょう。また、絵文字が使いづらいという方はリアクションで使うのがおすすめです。逆にテキストが法兰クすぎると馴れ馴れしいと感じられることがあるので、硬すぎるのが良くないからといって法兰クになりすぎるのはよくないです。バランスが重要なのでよく他のメンバーを観察して柔軟に変える必要があるでしょう。

最後は発生したバグやマイナスなコードを槍玉に上げてしまうとそれを実装した人がチームメンバーだったりすると一気に空気が悪くなります。自分が改修する際にバグを踏んで手間取ったり、緊急対応で時間を取られたりすると槍玉に上げたくなりますが、実装した当人に悪気なんてないですし、槍玉に挙げたところで問題が解決するわけではないのでデメリットしかありません。愚痴を言うなら機密情報は伏せた上で SNS や匿名ブログ、仕事に関係ない閉じた空間の無関係な相手にすべきでしょう。

1.6 コミュニケーションは最初が肝心

ここまでチームにおけるコミュニケーションをパターンごとに見てきましたが、どのパターンにも適用できる対処法としてチームのキックオフ時に以下の項目を明確にしておくと良いでしょう。

- ・チーム運営方針の合意を見る
 - 会議体
 - 朝会
 - 1on1
 - 定期的なふりかえり実施の合意など

- 新しいプラクティス適用の有無
- コミュニケーションの取り方
- 教育についてなど
- リーダー、メンバーそれぞれが明確にすること
 - 私が実現したいこと
 - やりたいこと/やりたくないこと
 - 我慢できること/我慢出来ないこと
 - やってほしくないこと

これはリーダーが行うべきことではありますが、やってくれない場合はメンバーから働きかけて明確にするのも良いでしょう。リーダー・メンバー間だけでなく、メンバー同士でも認識の齟齬が減るのはチーム開発を行う上で非常に重要です。最初に認識の齟齬をなくすのも重要ですが、継続してその努力を続けていくのも重要です。

1.7 さいごに

コミュニケーションは古城に難しいです。人間相手ですので、ここに特性があり、気を使うこと、考えるべきことが多く、正解もないので苦労します。

それでもチーム開発を行う上では避けては通れない道ですので、なんとか負担をなくし、円滑にチーム開発を進めていきたいです。

それには途中でも触れた NVC がひとつ特効薬になるかもしれません。これはリーダーだけでなく、チーム全体で行うことでさらに効果が出るでしょう。習得するには大変な手法ですが、効果はあると思うので気になる方はぜひ書籍などをご覧ください。

それでは、すべてのチーム開発がうまくいくことを願って、本章の内容がそれに少しでも役立つことを願って締めたいと思います。

第 2 章

実務で通用する設計・プログラミングスキルの学び方

FORTE(フォルテ)@FORTEgp05

IT エンジニアに必要なスキルとして設計とプログラミングは外せないでしょう。もちろん他にも必要なスキル、あった方がいいスキルはありますが、もっとも現場で必要とされる機会が多いのはこのふたつではないでしょうか。

この章では実務で通用するレベルで、設計とプログラミングスキルを学ぶにはどうしたらいいか、考えてみます。

2.1 「実務で通用する」の定義

まず「実務で通用する」とはどういう状態なのか、きちんと確認しておきましょう。ここがずれていると、せっかく学んだのに現場で役立つものかわからなくなったり通用しない知識になってしまいます。まずは全般的に実務で通用するレベルとはなにかを確認し、その後設計とプログラミングでそれぞれ見ていきます。

2.1.1 実務で通用するとは

通用するかどうかの判断方法

自分のスキルが実務で通用するかどうか分からぬと思うのであれば、視座を一段あげて考えてみることをお勧めします。もしあなたがマネージャーやリーダーのような指示をする側だったとして、あなたにその作業(設計やプログラミング)を任せられると思えるでしょうか?

安心して任せられると想像できるならあなたのスキルは実務で通用すると言えるでしょう。逆に不安だな、心配だな、難しいなと思うなら実務で通用するレベルに至っていないかもしれません。しかし安心してください。業務では 100% 安心して任せられるから作業を指示するということの方が稀です。多くの場合、やったことないけどできそだからやらせてみる(やってもらう)というケースの方が圧倒的に多いです。

スキルアップの考え方

そのため逆に言えば、不安や心配な点をカバーできれば実務で通用すると言えます。具体的には「設計書の書き方が不安なので一通り書けたらご確認いただけますか?」とか「実装はできますが、

もっと良い書き方があるかもしれない「一通り書けたらご確認いただけますか?(またはペア・モブプログラミングしていただけますか?)」のように自分が不安に思う点を補足するようにすれば十分実務で通用すると言えるでしょう。

もしそんな不安や心配な点を素直に伝えて大丈夫かなと思われたら安心してください。往々にして現場のリーダーやマネージャーというのはメンバーの教育、スキルアップも業務に入ってます。そのため、メンバーから教育してほしい点を伝えてもらえるのはむしろありがたいことなのです。稀にコイツは使えないヤツだな…のように判断するリーダーやマネージャーがいますが、その場合はこちらから見限るチャンスです。メンバーのスキルアップに貢献できるリーダーやマネージャーの元に移動しましょう。

2.1.2 実務で通用する設計スキル

それでは設計における実務で通用するスキルを考えてみます。

設計に求められるスキルとは

まず設計に求められることが、工程や環境、ステークホルダー、プロダクト、開発ルール、設計対象などによって違ってきます。そのため、これができれば実務で通用すると一言でいうのは難しいです。たとえば設計書を書く必要がない環境で設計書が書けますといつても何も意味がないですし、逆に設計書が必要な環境で設計書を書いたことがないと言ってしまうと相手にこの人に任せ大丈夫かなと不安を与えてしまいます。まずは自分がいる環境を正しく観察、把握して設計の中でも必要なスキルを考えてみましょう。

設計書を書く場合に必要なスキルを考える

例えばもし設計書を書く必要があるのなら過去に書かれた設計書を参考にして書けるかどうか、書けないのなら何が書けないのかを考えてみます。もし書けないのが文章なのであれば実際に設計書を書く練習をしてみるのが一番です。もし技術的な問題であればその技術について学ぶ必要があるでしょう。たとえばDB設計書を書こうと思った際にER図やリレーション関係、正規化などがわからないのであれば設計書に起こせませんので、そういう技術を学ぶ必要があります。このように問題(困難)を分割することで、曖昧で大味だった「設計ができない」という問題(不安や心配)から、より具体的ではっきりしたER図の書き方を知らないのように変換できます。ここまでくれば簡単です、ER図の書き方を調べて練習すれば設計書を書けるようになります。

困難の分割と問題と対策の具体化

設計書の書き方について例を出しましたが、要件定義や技術選定、詳細設計なども同じです。それぞれ何が必要で、自分には何かできない、不安、心配なのかを洗い出し、問題を分割して具体的にしていくことで学ぶ対象をはっきりさせることができます。これは先に述べた通り環境などによって変わるので、繰り返しになりますがまず自分がいる環境で何が必要なのかを明確にしてから明確にしていきましょう。

2.1.3 実務で通用するプログラミングスキル

次にプログラミングにおける実務で通用するスキルについて考えてみます。これは単純です、実装できるかどうか、その環境で求められる品質やルールに適したものにできるかどうかです。

実装できるか

まず実装できるかどうかは単純ですね、そのプログラミング言語で要求されたプログラムを書けるかどうかです。これも何も調べずに書ける必要はありません。細かい文法などは調べながら書いてもいいですが、書こうと思った際に何からしたらいいんだろう?と固まってしまうようであれば実務で通用するとは言えないでしょう。もちろん詳しい人に助けてもらいながら進めて良いのですが、事前に助けてもらうことを明言してから進めるのと、できますと言ってから助けてもらうのでは印象が異なります。もし不安や心配事があるのであれば、できるかどうか確認する(検討する)のでちょっとお時間をくださいのように予防線をひいておくのもいいでしょう。

求められる品質やルールを満たせるか

次に品質や開発ルールに則っているかが求められます。必要なエラー処理が抜けていたり、エラーや例外が発生した際に適切に処理ができるない、DRY原則やYAGNI原則に反しているなど書けているし動きもあるが、リリースできる状態にないコードでは実務に通用するとは言えないでしょう。ここで現場でありがちな例が、品質の悪いプロジェクト内のソースを参考にしてしまい指摘されてしまうケースです。これはスタートアップでとにかく早くリリースすることを目的としていた場合や有識者が少ない(いない)場合、デスマーチでとにかく納期に間に合わせた場合などに起きたのがちなのですが、DRY原則に反していても、多少のエラー処理がなくても動くのならリリースしてしまったことにより技術的負債として残ったソースであり、それを知らない人がこのプロジェクトはこのレベルでいいんだなど真似して実装するとレビューで指摘されてしまうというものです。

事前に求められているレベル(期待値)を確認する

これはまず期待値を調整すると回避できることが多いです。具体的には参考にできる既存ソースはこれでよいですか?や参考にするにあたって注意点はありますか?など参考にする際にお互いの期待値を調整すること、またはこれから実装するソースは品質優先か速度優先かどちらなのか、もし実装上で守るべき開発ルールや原則があれば教えて欲しいなどのように期待値調整するのも有効です。これにより環境に合わせて適切な品質のコードを提出することができるようになります。これなら実務に通用すると言えるのではないでしょうか。

2.2 実務で通用するスキルの学び方

実務で通用するとは具体的にどういうことを考えてみました。では、それを満たすためにはどのように学べばいいのでしょうか。設計とプログラミングに分けて学び方を考えてみます。

2.2.1 設計スキル

定義で確認したとおり、設計においてどのようなスキルが必要なのかはその環境などによって変わってしまいます。そのため、学び方としては一般的な開発知識として設計論を学ぶ方法と、いまいる環境に合わせて学ぶ方法の大きく2通りが考えられると思います。一般的な開発知識としての設計論は「システム開発 設計論」などで検索すれば技術書がヒットすると思いますので、それから学ぶことができるでしょう。書籍以外にも Udemy などにも設計に関する講座があるので購読してみても良いかもしれません。

現場に合わせて学ぶ方法

次にいまいる環境に合わせて学ぶ方法は、そのプロジェクトすでに作成されている設計書やドキュメントを参考に自分が担当している機能の設計書を書いてみたり、そのプロダクトに機能を新たに実装するしたらどんな設計書やドキュメントを書く必要があるか考えて実際に書いてみるなどが良いでしょう。可能であればそれらをチーム内の誰かにレビューしてもらうとより学びが深まります。実際に作業することも大事ですが、フィードバックをもらうことも大事です。人はどうしても視野が狭くなりがちです。自分では問題ないと思っていても客観的にはそうではない、またはもっと良くするための新たな視点が必ず存在するものです。そう言った意味でフィードバックは重要と言えます。もし、チーム内でフィードバックをもらうのが難しいのであれば、外部でもらうことも検討しましょう。その際は機密情報漏洩にならないようにシステムや用語を一般的なものに置き換えたものに対してフィードバックをもらうようにします。例えば Youtube にコメントが流れる機能を付けるとしたらどうすればよいか?などを考え、必要な設計書を書いて誰かにレビューしてもらうなどのイメージです。工夫すればできることは必ずあるはずです。

2.2.2 プログラミングスキル

プログラミングはまず書けるようになることと、一定以上の品質を担保することの2段階があると思います。

まず書けるようになること

まず書けるようになることは if 文や for 文などの一般的な制御構文はもちろん、その言語独自の言語仕様を把握しておく必要があります。これは全ての仕様を把握すべきという意味ではなく、存在を知っている程度でも大丈夫です。必要になるかどうかはわからないですし、必要になった時に調べるのでも大体のケースは間に合います。ごく稀に調べている時間がないというケースもあるかもですが、その際は詳しい方に聞けばいいですし、聞ける人がいないのならもう調べるしかありません。覚えておいて損はないと思いますが、時間をかけて言語仕様を暗記するくらいだったら調べながらでも動くプロダクトを作る経験をした方が時間の使い方としては有意義なのではないかなと思います。

求められるレベルのコードにすること

次に一定以上の品質を担保する必要があります。ここでいう一定以上のとは開発ルールに則っているか、という意味です。極端な話、自分1人でしかメンテナンスしないコードであれば、どんなコードを書いても自分しか見ないので問題ないかもしれません。ですが、多くの場合がチーム開発であり、何かしらの開発ルールがあるものなので、それに則ったコードを書けるようになる必要があります。これは開発ルールごとによって異なるので具体的にこれをとは書きづらいですが、一般的に読みやすいコードを書けるようになっておくのは有効かと思います。名著リーダブルコードなど、読みやすいコードについて書かれた技術書はたくさんありますのでそれらで勉強すると良いでしょう。きっとそこで勉強した概念が開発ルールに取り込まれていると思いますし、独自のものがあればあらためて勉強すれば大丈夫です。

特許について

本業の方で特許を書くようになり、これまで抱いていた特許についてのイメージがだいぶ変わりました。

一例&実体験として挙げますが、案外そういうイメージと実際に乖離があって、必要以上に構えたりしませんか？ という気づきのきっかけになれば幸いです。

3.1 特許のイメージ

特許ってどんなイメージがありますか？

ものすごい発明！ 世紀の大発明！ 特許が取れるなんてすごい！

そういう発明があることも事実です。ですが、世の中の特許は、必ずしもそういうものばかりではありません。

日本の特許は、年間30万件程度出願されており^{*1}、登録になるのは24万件程度^{*2}です。

この数を見て、「案外多いな」と思いませんでしたか？

まず、この数を見て、世紀の大発明ばかりではないと認識できたでしょうか？

3.2 特許には何が必要か？

では、特許を取るには何が必要でしょう？

まず、特許の要件として、新規性と進歩性が挙げられます。

有り体に言うと、新規性は、これまでになかった内容・技術であること。進歩性は、これまで解決できなかった課題を解決できること。

ここまで読んでくれた方にはもうお分かりですよね。特許を取るために必要なことは、「課題をいかに設定するか」です。

その課題を解決することができれば、進歩性は確保できます。新しい手法で課題を解決すると新規性が確保されます。そうです。特許の肝はこれだけなんです。

^{*1} 特許行政年次報告書 2023年度版 <https://www.jpo.go.jp/resources/report/nenji/2023/matome.html> をとりまとめました

^{*2} <https://www.jpo.go.jp/resources/report/nenji/2024/document/index/0101.pdf> 国内外の出願・登録状況と審査・審判の現状

3.3 似たような課題だって特許になる

よくある話で、物体のサイズを変えたい、という課題に対し、ビックライトがありますね。ドライモンの道具で、懐中電灯型のデバイスから出る光を当てるとき大きさを変えることができます。

もちろん、その光線をどう作るかという問題はありますが、まあそこは置いておいて、「拡大・縮小光線」を作る技術があったとして、任意の場所で拡大・縮小をしたいので、懐中電灯型のデバイスとしました。これで特許が取れましたね。

競合他社も、同じような目的を持っていました。ところが、ビックライトの特許があります。自分が拡大・縮小光線を懐中電灯型のデバイスに組み込むと抵触してしまいます。

特許は、強力な排他権ですから、特許に抵触するものを製造・販売することはできません。でも売りたい！

そこで、別の課題を設定しましょう。例えば、霧の中でも使える！（ビックライトは光線なので霧の中では使えませんね！）のために、据え置き型のデバイスとして、片側から入ることで大きさを変えます。

特許回避技術が成立しました。

3.4 まとめ

特許のイメージが少し変わりましたか？

特許を取るからには、新しく、技術の発展に寄与したものです。ですが、余りにハードルが高いものだと思っていませんか？ という話でした。

I型人材、T型人材、π型人材とは

現代の社会や企業では、多様なスキルセットを持つ人材が求められています。特に、定型業務はいわゆるAIやRPAなどにとって代わられてしまう部分もありつつ、その中で人が行うべき業務とのすみわけという観点でも語られることがあります。

人材のカタチとして「I型人材」「T型人材」「π型人材」という3つの概念が注目されています。それぞれの特徴と、それが現代の環境においてどのような意義を持つのかを確認します。

4.1 人材のカタチ

I型人材

I型人材は、専門的なスキルや知識に特化した人材を指します。Iの形が示すように、1つの領域に深く掘り下げており、高度な専門性を有しています。例えば、医師やエンジニア、研究者などがこれに該当します。彼らは特定の分野で大きな成果を出すことができ、その専門性が社会や企業にとって重要な資産となります。

しかし、I型人材は往々にして視野が狭くなりがちです。自分の専門分野以外の知識やスキルが不足していると、異分野との協働や新しいアイデアの創出に困難を感じる場合があります。

もっとも、この例では医師やエンジニア、研究者がI型人事の例として取り上げられましたが、最近これらの人々は次のT型人材、あるいはπ型人材であることも多く、むしろJTCで提携業務しかやってこなかった人がこういったところになるようにも思われますね。

T型人材

T型人材は、I型人材に比べてより広範なスキルセットを持っています。Tの横棒が示すように、複数の分野にまたがる幅広い知識やスキルを持ちながら、1つの分野で深い専門性を有しています。例えば、エンジニアでありながらプロジェクトマネジメントスキルを持つ人材や、マーケティングの専門家でありつつデータ分析のスキルを備えた人材がこれに該当します。

T型人材は、チームや組織内で橋渡し役を担うことができ、異なる専門性を持つ人々の間でのコミュニケーションや問題解決に優れています。そのため、イノベーションを推進する上で重要な役割を果たします。

マネージャになるような人は、この型であってほしいところです。

π型人材

π型人材は、T型人材をさらに発展させた概念で、2つ以上の分野で深い専門性を持つとともに、幅広い知識やスキルを持つ人材を指します。πの形が示すように、2本の縦棒がそれぞれの専門性を、横棒が広範な知識やスキルを表しています。

例えば、AIの専門知識を持つと同時にビジネス戦略にも精通し、さらにデザイン思考や心理学といった異分野のスキルも活用できる人材がπ型人材の例です。彼らは高度に複雑な課題に対して、多面的なアプローチを取ることができます。現代の変化の激しい環境で特に価値があります。

4.2 π型人材になるために

では、どのようにしてπ型人材を目指すことができるのでしょうか。以下にいくつかのポイントを挙げます。

- 2つ以上の専門分野を深める

一つの分野で専門性を磨くことはもちろんですが、それだけでなく、別の分野にも積極的に挑戦することが重要です。例えば、技術系のスキルを持つ人が経営学を学んだり、クリエイティブ分野の専門家がデータ分析を学ぶなど、異なる分野のスキルを身につけることで、π型人材に近づくことができます。

- 異分野間の統合を意識する

単に複数の専門性を持つだけではなく、それらを統合して活用する能力が求められます。これには、クリティカルシンキングやシステム思考といった、全体を俯瞰する力が不可欠です。

- 繼続的な学習習慣を持つ

現代は技術や知識が急速に進化する時代です。自ら学び続ける姿勢を持つことが、π型人材になるための基本条件と言えます。オンラインコースや勉強会、専門書の読書など、多様な学びの機会を活用しましょう。

- 実践を通じてスキルを磨く

新しいスキルや知識を得たら、それを実践で試してみることが重要です。プロジェクトや副業、ボランティア活動など、異なる分野の実務に関わることで、知識を現実の課題解決に結びつける力を養います。

- 他者との協働を大切にする

π型人材は単独で成果を上げるだけでなく、他者と連携して新たな価値を創造することにも優れています。そのため、チームでの活動や異なる専門性を持つ人々との交流を積極的に行いましょう。

- ひとつの専門を深く掘る

また、一つの専門を掘っていくと、自然と周辺知識が増えてきます。T型人材であるとしても、非常に太いT型となるイメージです。穴を掘った跡を見返してみましょう。何の支えもなく井戸のように細く掘ることは難しいですが、労力はかかるものの、露天掘りのように太く掘っていくイメージです。中心部は深く掘りつつ、周辺部も合わせて掘っていく。これがπ型人材への近道かもしれません。

4.3 結論

I型人材、T型人材、π型人材はそれぞれ異なる特性と価値を持っていますが、現代の複雑で多様な社会では、特にπ型人材が大きな可能性を秘めています。

複数の専門性を持ち、異なる分野を結びつける力を磨くことで、個人としての価値を高めるだけでなく、社会や組織に貢献することができるでしょう。まずは自分の得意分野を深めつつ、新しい領域に挑戦する一步を踏み出してみてはいかがでしょうか。とはいえ、あなたの身近にもなんかやたら広い知識や経験を持っている人いませんか？ そういった人の体験を聞いてみるのもいいですね。

あと、この本の著者にもそういう人がいます。この本を読むことで、そういった学びを追体験してみるのも良いかもしれません！

第 II 部

学校での学び

学校で学んだことも多いですよね。今大学生の人、あるいは社会人をやりながら大学(放送大学など)に入っている人もいるでしょう。

もちろんそれだけでなく、学校で何を学ぶのか、学校で学ぶ意義、などをすこし掘り下げ、改めて考えてみませんか?

第 5 章

放送大学に入学した話

ロジウム @_rhodium_

5.1 序文

私事のストレスが原因で食事すると吐いてしまうようになり、会社の福利厚生でカウンセリングルームにかかりました。これが私が心理学(特に臨床心理学)とのファーストコンタクトでした。

臨床心理は、他の多くの医療分野とは異なり、特殊な領域です。

多くの科では細菌やウイルスによる感染症や骨折など、何か不快になる原因を見つけることができます。一方で臨床心理分野では、MRI や CT を撮っても異常はないのに、食べたら吐くという症状が存在します。うまく言葉にはできませんが、漠然とした興味が沸きました。

あと、吐かなくはなったものの、禄でもない状態は続いており、自己メンテナンスできるようになるためにも臨床心理の技法を身に着けたいと思いました。

5.2 放送大学入学

では実際に学んでみようとなった時、一番簡単なことは、本などを紐解くと思います。ただ、(ネット上のブログよりはまともですが、) 本の中身について背景理論はどうか、著者のオリジナルな部分は何かを切り分けて読むというのは初学者にとって不可能です。

また、就労している都合上、忙しいを免罪符として後回しにしがちです。後回しにした結果、二度と開かないことも予想されました。体系立っており、スケジュールを強制されるという面で、大学に通うという選択をすることにしました。

私は会社から給料をもらう身であり、平日には仕事があります。仕事と大学での学びを両立できる方法として、通信制大学という選択肢があります。

Google で検索したところ、放送大学で心理学を学ぶことができるとわかりました。名前を聞いたこと也有ったことから^{*1}精神がすり減っていた当時の私はそれ以外の大学を探すエネルギーはありませんでした。

^{*1} 知っていると言ってもテレビのチャンネルを回していた際に見たことがある程度です。

問題は学費です。以下のコラムのように、ちょっといい本のお値段どころではない費用が掛かります。

悩みました（2日ぐらい）。

結果として、仮に精神状態が悪化した場合、自傷他傷のおそれが考えられたため、セルフメディケーションの一環として、費用の執行を認可しました。

■コラム：学費

私は1年間だけ在学できる専科履修生でした。これを執筆している2024年2月現在では費用は以下の通りです。

入学料 9,000円/年

授業 6,000円/単位^{*2}

入学に当たり、学生の種類や、受講する講義を選択する必要があります。

すでに学士号を持っているため、学士号取得コースである全科履修生ではなく、1年間だけ在籍できる専科履修生を選択しました。

受講する講義は特に制限されないため、導入科目を履修せずに専門科目を受けることも可能でした。受講するコースに悩むという方は、放送大学エキスパートというモデルコースが設定されているため、臨床心理学コースから6科目を選びました。^{*3}

5.3 入学そして受講

多くの場合、授業自体はテキストと放送授業を見るだけです。私の場合は、通勤の間にテキストを読み、週末に放送授業をまとめて見るとしていました。放送授業は45分間の授業が15回で、2単位が出ます。普通の大学だと90分×15回なので、45分は気力的にも楽です。

1.5倍速で見ても毎週30分/科目の視聴時間を確保する必要があります。スクーリングも行いつつの場合、それよりも密度を高くしないといけないので、スケジューリングをちゃんとしましょう。

また、申し込める期間が少ないので、対面式の授業もあります（スクーリング）。こちらは90分×7.5回で1単位です。土日で連続開講される場合、土日ががっぽりなくなるので注意です。期末試験はなく、授業後のテストやレポートで合否が決まります。（レポートを書くのも時間がかかる作業です）

後は中間・期末試験の期間中に、期末試験を受けるだけです。多分6割程度とれていれば、単位が出ます。

^{*2} <https://www.ouj.ac.jp/admission/gakubu/tuition/> (2024.2.12閲覧)

^{*3} https://www.ouj.ac.jp/booklet/2024/21_2024_expert.pdf

5.4 結果

資格としては、次節のように認定心理士の登録までできました。

実際に自己メンテナンスができるようになったかは、同時期にカウンセリングルームに通っていたため、これまでに修得したうちで授業の効果がどれだけあったかを測るのは難しいです。ただ、カウンセリングの中で専門用語で会話できるので、1回のカウンセリングの密度は上がっていると思っています。

個人的には、バウムテスト^{*4}が好きで、自分自身との対話のきっかけに使います。自分が描いた木とそれに関する解釈を事前に行い、カウンセリングでレビューしてもらっています。

5.5 良かったこと

中間・期末テストの時期が明確にあり、先延ばしにすることなく、勉強できました。外部から強制されないと動けないことは締め切り駆動開発などにも通じるものがあります。

過去に体系立てられた分野を俯瞰して見れたと思います。特に心理学分野は胡散臭そうな本もあるので、何が標準で、何が著者の主張なのかを区別するための基準になっていると思います。

放送大学の柔軟な学習システムです。入学時のコースとは異なる授業も受けることができます。さらに単位が要らなければ、ビデオを無料で見ることができます。これをを利用して、自然言語処理など情報系の授業なども受講することもできます。このような垣根なく学べるのはいいことです。

美術館に大学生として入ることができます。3つぐらい大学生として入館すると、学費分は回収できるかなという感じです。

Github Copilot も学生価格で使えるようです。

5.6 悪かったこと

時間がとられるのはクリティカルに効いてきます。たとえ美術館に学生価格で入れるにしても、そもそも美術館に行く時間を確保することができません。独身だからできていることだと思います。

最低でも受講科目数については自分の余力とよく相談してください。

*4 樹木画テストとも。クライアントに木の絵を描いてもらい、その後の質問応答などを含めクライアントの無意識下の情動に迫ろうとする技法。

5.7 まとめ

読んでいない積読が山になる私にとって、放送大学の期末試験という明確なマイルストーンは私の勉強を強力にサポートしてくれました。もし、勉強が続かないと悩まれている方は、自分が変えられない目標地点を設けてみるのも一手と感じました。

第 6 章

放送大学で認定心理士を取った

6.1 認定心理士とは

認定心理士とは

日本心理学会認定心理士とは大学における心理学関係の学科名が学際性を帯びてきて、必ずしも「心理学」という、直接的名称が使われていない場合が多いことから、心理学の専門家として仕事をするために必要な、最小限の標準的基礎学力と技能を修得している、と日本心理学会が認定した人のことです。

<https://psych.or.jp/qualification/> (2024/2/8 viewed)

前項で書いたように、ストレス性の嘔吐をきっかけに心理学に興味を持ち、放送大学で心理学を学ぶことを決意しました。

当初は放送大学エキスパートプラン^{*1}の「臨床心理学基礎プラン」の修得を目指していましたが、1 学期の中間テストごろに、それに加えて認定心理士^{*2}の取得を目指すことにしました。

実際に心理を専門として仕事をする場合は、公認心理士などの資格が必要です。しかし、公認心理士や臨床心理士の取得のためには大学院に進んで実習を行う必要があり、仕事をしながらでは不可能であるため、認定心理士をターゲットとしました。^{*3}

6.2 目標

認定心理士に必要な物は以下の通りです。

- 学士号または修士号(取得見込みを含む)
- 指定された分野ごとに所定の単位を修得している
- 16 歳以降通算 2 年以上日本に滞在した

修士号（工学）の取得と日本国内での長期滞在要件は既に満たしています。残るは、認定心理士

*1 放送大学が独自に設けているコースプラン。社会的に効力はない。

*2 日本心理学会が設けている制度。類似する資格名として、「公認心理士（公認心理士法により定められる名称独占資格）」「臨床心理士（日本臨床心理士会による民間資格）」がある。本当に紛らわしい。

*3 働きながらでも取りやすい資格として、大阪商工会議所が主催する「メンタルヘルスマネジメント検定」もあります。

資格取得に必要な指定単位の獲得です。さらに、放送大学は公認心理士に合わせたカリキュラムにしており、同カリキュラムを参考にする認定心理士を取得するのも容易です。(時々、なぜ公認心理士ではこの2つの科目を合わせカリキュラムなのかと、放送授業では恨まれていたりしますが)

短期に集中して学習を行い、1年半を目途に公認心理士取得を目指します。1年目前期の科目を含めた最低必要な単位取得スケジュールは以下です。取得単位は再履修をしないことを前提に均等割りにします。

なお、受講が必須とされる科目のうちに心理学実験^{*4}があるため、最低4単位分は大学に行く必要があります(スクーリング)。コロナ禍で受験者の密度が下げていたため、倍率は高まっているようです。徐々に緩和され、今は2019年程度まで戻りました。

表 6.1: 履修スケジュール

期間	放送授業	スクーリング	
2022年前期	6科目	--	
2022年後期	7科目	2科目	※うち2科目は放送大学エキスパートの為認定心理士取得には利用せず
2023年前期	6科目	2科目	

6.3 結果と感想

修士号(工学)と分野違いの修士号でも公認心理士として認められるかが不安でしたが、無事に資格を取ることができました。

6科目受講したということは週末に3時間以上を費やして(1.5倍速)動画を見ていたことになります。スクーリングのレポートを書いたりしていたので、かなり無理したスケジュールだったと思います。

スクーリングは移動の手間がありますが、同じ受講生の間でコミュニケーションを取る楽しみや、休み時間に理解した内容を深めあうなどができる、得難い体験でした。心理学は文系の学問のイメージがありますが、実際には統計学が重要な役割を果たします。統計などの数学的手法に苦手意識のある人も多いので、統計についてイメージで伝えられると、クラスのヒーローになれるかもしれません。

レポートは授業当日にできるだけ完成させると、記憶が新しいうちにレポートを書けるので後で思い起こす必要がなくなります。特に、1日目の授業の内容についてレポートを書いておけば、2回目の授業で教員に質問するなど、学びを深めることができます。

通学を主とする教育機関のように学友と教えあったりする体験が無いのは残念です。人

*4 「>---<と<-->で長さが違って見えるよね。人によってその程度は違うかな?」という実験の設計から報告までを行います。

に教えるときは間違えたことを言わないように気を付けますし、分からぬところをちゃんと言語化するという訓練が積めます。

放送大学で情報学を体系的に学ぶ

7.1 はじめに

yuusukesan@yuusukesan1018

はじめまして、yuusukesan1018と申します。エンジニアとして働き始めてから、日々の業務をこなすことに精一杯だった私が、放送大学で情報学を学ぶことについて考えてみます。

7.2 文系エンジニアが放送大学で学び直しを決意した瞬間

エンジニアとして 3 年目を迎えるこれまでの振り返りと今後について考えるようになり、仕事を通じて、以下のことを考えるようになりました。

「システム開発の実装自体は容易になっていく一方で、より深いレベルでの理解がなければ質の高いものは作れない」

開発作業をすること自体は、AI できます。しかし、チューニングは、別途知識が必要があると考えました。そのためには、コードを書く開発経験と共に情報学の基礎を学ぶ必要があると考えました。

7.3 情報学を体系的に学べる方法

情報学を体系的に学べる方法を調べていく中で、以下の 3 つの方法を検討しました。

- 大学
- 書籍
- オンラインサービス

7.3.1 大学

大学は、1 年生からやり直すので、時間が勿体ないと考える人もいることでしょう。編入制度を利用すると、過去の大学の単位を利用できます。この制度を利用することで、専門の科目だけを履修して、卒業できます。

7.3.2 書籍

書籍を利用して、情報学を学べます。必要な分野をピックアップして、該当する書籍を書店やAmazonにて購入して、勉強できます。また、図書館を利用することで、書籍代を抑えることができます。学びにコストをかけたくないという人におすすめの学びの方法といえます。

7.3.3 オンラインサービス

cousera や edX といったオンラインサービスを利用して、情報学を学ぶことができます。cousera では、海外の有名大学のコースを提供しており、一部のコースにおいて、日本語字幕を対応している講座もあります。必要な講座だけを履修できること、また、講座を修了すると、LinkedIn のステータスに追加できます。

7.4 放送大学以外の選択肢

大学には、従来の対面式とオンライン型の2種類があります。社会人の学び直しでは、近年、オンライン大学が注目されています。

日本では、通信制の放送大学や帝京大学理工学部情報科学科通信教育課程などがあります。海外では、UoPeople というアメリカのオンライン大学などがあります。

7.5 放送大学で情報学を学ぶメリット

放送大学で情報学を学ぶメリットは、以下の2つです。

- 理論と実践の往復学習ができる
- 低コストで体系的に学べる

それについて、具体的な紹介します。

7.5.1 理論と実践の往復学習

学びには、実践という経験からの学びと書籍などの体系的な概念や理論から学びという2つがあります。学生時代は、教科書を利用した知識としての学びが中心です。一方で、社会人は、仕事を通じた経験からの学びが中心となります。

理論と実践の往復学習をすることで、理論が実践でどう活かされるのか理解できる。経験で学んだことを理論的に裏付けられるという効果が期待できます。

理論と実践の往復学習は、理解をより深めることができます。

7.5.2 低コストで体系的に学べる

大学のキャンパスに通って、体系的に学ぶ場合、学費というコストがかかります。放送大学では、面接授業とオンライン授業が 1 単位あたり 6,000 円 (2024 年 12 月時点) です。放送授業が、1 科目 2 単位で、12,000 円 (2024 年 12 月時点) です。卒業研究を行う場合は、6 単位が必要となります。例えば、本科生として 3 年次編入し、卒業を目指す場合は、入学料、既修得単位認定審査等手数料、そして、授業料が必要です。卒業までに、約 40.6 万円が必要となります。

表 7.1: 学費

項目	価格
入学料	24,000 円
授業料 (62 単位分)	372,000 円
既修得単位認定審査等手数料	10,000 円
合計	406,000 円

7.6 仕事と放送大学での学びを両立するために気をつけること

仕事をしながら、放送大学で情報学を学ぶにあたり、両立ができるのかと気になる人もいるかもしれません。大学に注力するあまり、仕事に影響が出るかもしれない、そして、仕事に追われて時間が多く、大学の授業を等閑 (なおざり) にしてしまうこともあります。

仕事と放送大学での学びを両立するにあたり、時間管理と Youtube の活用という 2 つの点を意識して、取り組む必要があると考えています。

7.6.1 時間管理

放送大学は、1 つの授業が 45 分となります。ルールを決めて、習慣化するようにしています。

例えば、月曜日と火曜日のどちらかに、「数学」、水曜日と木曜日に「仕事に関する科目」、そして、土日の午前中に、残りの科目を受講するというように決めています。

時間枠を固定して、習慣化するように意識しています。

7.6.2 Youtube の活用

Youtube の活用もおすすめです。Youtube では、質の高い授業が行われているケースもあります。特に、数学は、Youtube でわかりやすい授業を発信している人もたくさんいます。

事前に授業で学ぶテーマを確認して、関連するテーマの Youtube を探します。Youtube を取り入れて、以下のサイクルを意識しています。

- 予習：動画を 2 倍速で視聴して大まかな概要を掴む

- 講義：理解に注力、わからないワードをメモ
- 復習：わからないワードを ChatGPT で確認する、講義の参考文献を確認する

Youtube は、通勤している人ならば、通勤途中に見ることもできます。放送大学の本科生の場合、学割を利用して、月額 780 円にて、Youtube Premium に加入できるので、広告をカットでき、ストレスなく視聴できます。

7.7 放送大学で勉強するにあたっての Q&A

放送大学で情報学を学ぼうとしている人から受けたことのある質問を最後にご紹介します。

7.7.1 単位認定試験の準備はどうしている？

放送大学では、通信指導という名の中間試験があります。この通信指導を受けないと、最後の試験を受けることができません。通信指導には、評価に使われる通信問題と評価には評価されない自習型問題があります。テスト前には、通信問題の復習と自習型問題を解くようにしています。また、過去問も教務情報システムに掲載されているので、該当する科目的過去問を解くようにしています。更に、教科書を読んだり、わからないところは、ChatGPT に聞いて、理解するようにしています。

7.7.2 モチベーション維持方法は？

モチベーションを維持するのは、大変な時もあります。ただ、学問として、情報学を楽しむようにしています。また、学位授与機構を通して、学位を取得してからのキャリアをイメージしてモチベーションを維持しています。もちろん、テストが終わった後は、美味しいものを食べたりして、気分転換をしています。

7.7.3 仕事にどのように活かせているか？

ネットワークの授業を受講してから、インフラ関連業務の解像度が上がりました。これまでのデータベース関連の業務についてより理解したいので、来期データベースの授業を受講しようと考えています。

7.8 さいごに

この原稿の執筆にあたり、私は放送大学での学びに関する過去のメモを見直しました。インターネット上において、多くの方が経験を発信している中で、自分の体験が他者の役に立つか迷いがありました。一方で、執筆を通じて、なぜ情報学を学ぼうと決めたのか、入学前にどのような準備をしたのかを整理できました。この私の経験が、放送大学で情報学を学ぼうとする方々のキャリア形成の参考になれば幸いです。

小中高の学びは重要

春は揚げ物段々キツくなりゆく腹際

これにくすっと笑えるということは、中高の学びが活きているということ。

教養と言ってもいいかもしませんね。

X(旧 Twitter) でみんな大好き山月記。

共通することは、小中高で一応習うこと。それを活用することはなくても、詳しいことは覚えていなくとも、そういうものがあったと知っているか知らないいかには大きな差がありますね。

小中学校から高校での学習内容は、それなりに練られた、必要最低限の内容です。

春はあげぼよ夏はよきよき秋はばえばえ冬はつらたん

なんじゃこれは？ でも生きていくには困らないでしょうが…

第9章

スポーツ嫌いを量産する方法

スポーツ好きですか？

個人的にはとっても苦手意識があるのですが、それは、小中学校の体育のせいだと思ってます。小学校の体育は、鉄棒や跳び箱、その他のカリキュラムがありますけど、ロクにやり方を教えてもらえないのにさあやれ。なんだできないのか、とバカにされ、スクールカーストを固定化するわけです。

同じ事は、小中学校の美術や音楽にも言えますね。歌の歌い方、絵の描き方を教えることなく、さあやってみろ、お前下手だな。これで好きになるわけないじゃない。

でも、算数ならば、計算のやり方を教えてくれるし、いきなり「この計算やってみろ。なんだできないのか」とは言われないでしょう。なんだこんな漢字も読めないのか、とは言われないでしょう。この差は一体？

もちろん、いろいろ教える時間がかかることもわかります。授業時間の制約や、先生の教えられる範囲に制約があることも。

それでも、こんな状態では好きになれるわけないですよね・・・

<https://togetter.com/li/2480781> 市民がスポーツする場をつくるNPOの人が「学校の体育を恨む人がいるなんて知らなかった」と呆然としてて衝撃的だった話…スポーツは好きでも体育は嫌い

これを見かけて…

学びのために教育学の基本を勉強してみた！

こうの@hk_it7

10.1 一番「学び」を研究していそうな分野を勉強すれば学びを得られるのでは？

「学び」を考えたときに真っ先に浮かんだことは、自分はどうやって学んできたのだろうかということです。

自己学習も多かったですが、IT 関係の知識以外の広範な基礎知識のほとんどは学校教育で習ったものです。

そこで多くの知識を与えるために体系化された「教育学」のエッセンスを学ぶことで、自分や他の人の学びにつなげることができるのであればと考えました。

10.2 本章で取り上げる教育学的なアプローチ

とはいっても教育学は膨大で、それだけで大学のカリキュラムになってしまいます。そのため有名どころをつまみ食いして勉強し、それをまとめます。

本章で取り上げる教育学的なアプローチは次の通りです。

- 4 段階教授法
- 5 段階教授法
- 9 教授事象
- プログラム学習の 5 原理
- 問題解決学習

では、順番に紹介していきます。

10.3 4 段階教授法

4 段階教授法は、学習者がなにかを学ぶメカニズムを 4 つの段階で定義したものです。

- 明瞭：学ぶ事象を明瞭にする

- ・連合：明瞭化した事象を既知の概念と比較する
- ・系統：連合された物事を体系化する
- ・方法：系統化した事象をほかのものに応用可能にする

そもそもここで言う「教授」の前提には「管理」と「訓練」という2つの概念があります。

- ・管理：教育の妨げを除外する
- ・訓練：正しい方向に正す

一方で「教授」は教材を通して間接的に働きかけを行います。当然ながら学ぶ主体は学習者と教材側にあるわけです。

教授法と聞くと「教師がどのように教えるか」という概念のように思えますが、4つの段階を見ると学習者視点での知識獲得までの流れを示していることがわかります。

この4段階教授法は『一般教育学』にてヘルバートによって示され、体系化したことにより「教育学」が学問分野として確立したきっかけになります。それまでの「教育」は「教育者」の経験や技能、感覚に基づくもので体系化されていませんでした。

ヘルバートは経験のない人に一定の方針を出すことでベースアップを目指しました（教育的タクトというらしいです）。そのため、教師がどう教えるかではなく、学習者がどう学んでいるかを体系化し、レールを作ったということですね。

この4段階教授法は4段階目に「方法」として「応用可能にする」ところまでを知識獲得の定義にしていることが面白いところかと思います。つまりは暗記だけに終わるものは学習ではないということでしょうか。耳が痛いです。

ここからは新たなプログラミング言語の習得という具体例を使い、4段階教授法を深堀りしようと思います。

10.3.1 明瞭：学ぶ事象を明瞭にする

まずは学ぶ対象を明確化します。ここでは TypeScript を学ぶことを想定します。

もともと別の言語を習得している人が学ぶという前提に立つと、ここでの学びの対象はアルゴリズムやデータ構造ではなく文法がメインになりますね。

JavaScript をすでに習得している場合、さらに具体化して「型定義や型システムについて」が学ぶ対象になります。

それらについて明確な知識にします。

- ・文法は基本的に JavaScript と一緒に
- ・型の記述がとても柔軟
- ・型システムは構造的部分型

10.3.2 連合：明瞭化した事象を既知の概念と比較する

続いて明確になった知識を既知の概念と比較していきます。

TypeScript を学んでいくなかで以下のようなことに気づいていきます。

- 型システムが Java（公称型）とは異なる
- Java と同じように型を扱うには class にして private な変数を持つ
- 型システムが Go（構造的部分型）と少し似ている
- JavaScript に対して新しい文法も入っている
- Sass と CSS の関係のようにトランスパイルする

10.3.3 系統：連合された物事を体系化する

他の言語と比較して TypeScript の特徴が見えてきました。それらを少し抽象化してみます。

- 構造的部分型は型のインターフェイスに注目して型を判別する
- スーパーセットとトランスパイルによって既存言語を大きく拡張できる

などなど、学べば学ぶほどここで列举しきれないほどの特徴が見えてくることでしょう。

10.3.4 方法：系統化した事象をほかのものに応用可能にする

さて、最後に応用可能な状態にしましょう。

例えば次のようなものです。

- 構造的部分型と公称型というものがあり、他の型付け言語もこれらの考え方を持っている
- トランスパイルすることでベースの言語のエコシステムに乗っかることができる

上記の知識が定着すれば、さらに別の言語を学ぶ際に、「Java と Kotlin の関係性は JavaScript と TypeScript に似ている」や「PHP の PHPStan は TypeScript の型システムっぽい」等の気づきが得られます。

これは応用可能な状態になっていると言えそうです。

10.4 5 段階教授法

続いて 5 段階教授法についてです。これはツィラーのものとラインのものがありますので、まずはそれぞれを説明します。

10.5 ツィラーの 5 段階教授法は 4 段階教授法の順当な発展形

ツィラーはヘルバートの 4 段階教授法における「明瞭」を 2 つに分割して以下の 5 段階教授法を示しました。

- 分析
- 総合
- 連合
- 体系
- 方法

ツィラーの5段階教授法も学習者がどう学んでいくかに焦点を当てたものです。

「明瞭」で学ぶ対象がどんなものであるかを調べていきますが、その過程は「分析」と「総合」からなるとしました。

「分析」では対象を細かく見て単純な構造の組み合わせとして捉えるようにしていき、「総合」では「分析」で明らかにした単純な構造を再構築します。

これは具体と抽象を行き来する過程に似ています。「分析」と「総合」を通して学びを深めていくことを意味します。

10.6 ラインの5段階教授法は教師側に対する論理

一方、ラインは教師側から教授法を再定義しました。つまり、ヘルバルトとツィラーが学ぶ側のメンタリティを段階的に示したのに対して、ラインは教える側のステップを示しています。

ラインが示した5つの段階は次の通りです。

- 予備：既知の概念を整理する
- 提示：新しい学習対象を提示する
- 比較：新しい対象と既知の概念を比較する
- 概括：比較結果をもとに体系化する
- 応用：体系化された知識を応用して利用する

この5段階教授法は日本にも輸入され、教師中心の学校教育の基本となりました。一方、現在の日本の学校教育はアクティブラーニングや「主体的・対話的で深い学び」を主軸にしています。そちらは「問題解決学習」で後述します。

ラインの5段階教授法を詳しくみます。ここでは具体例として四角形の内角の和を導出する授業を考えます。

10.6.1 予備：既知の概念を整理する

既知の概念として「三角形の内角の和が 180° であること」があります。

そこから「四角形の内角の和を導き出す」にはどうしたら良いか問い合わせます。

10.6.2 提示：新しい学習対象を提示する

新しい学習対象である四角形の内角の和を考えるために、四角形を例示します。

そして、対角線を 1 本引くことで 2 個の三角形で四角形が成り立っていることを確認します。
最後に 2 つの三角形の内角の和を足し合わせることで、四角形の内角の和が 360° であること示します。

これで新しい学習対象である「四角形の内角の和」が 360° であることを提示できました。

10.6.3 比較：新しい対象と既知の概念を比較する

比較ではより、詳細に新しい学習対象を調べていきます。
台形ではどうなるか、凹みのある四角形でも同じなのかなどです。
そこで改めて「四角形は常に 2 つの三角形に分割できるから、内角の和は $180^\circ + 180^\circ = 360^\circ$ になる。」ことを示します。

10.6.4 概括：比較結果をもとに体系化する

学習結果をまとめます。次のようなことが考えられます。

- 三角形の内角の和が 180° であること。
- 四角形は三角形 2 つに分割できること。
- 四角形の内角の和が 360° であること。

10.6.5 応用：体系化された知識を応用して利用する

最後に応用を考えていきます。例えば「五角形や六角形の内角の和を求める」のようなものです。五角形であれば三角形が 3 つ、六角形であれば三角形が 4 つで構成されることがわかります。結果として頂点の数を n として $(n - 2) * 180$ が多角形の内角の和になることが導出できます。

10.7 9 教授事象

ガニエの 9 教授事象は、効率的に教えるためのシステムです。より具体的にどういうことを行ったら良いか示しています。

次のように整理されています。

- 導入
 - 学習者の注意を喚起する
 - 学習の目標を知らせる
 - 前提条件を思い出させる
- 情報提示
 - 新しい事項を提示する
 - 学習の指針を与える
- 学習活動

- 練習の機会をつくる
- フィードバックを与える
- まとめ
 - 学習の成果を評価する
 - 保持と転移を高める

勉強会やワークショップの設計に利用できるフレームワークかと思います。

10.8 プログラム学習の5原理

行動心理学の観点から、オペラント条件付けを学習に応用するために考えられたものが「プログラム学習の5原理」です。

オペラント条件付けは自発的な行動後の変化によって自発行動の頻度が変化する学習のことです。

表 10.1: オペラント条件付けの例示

行動	結果	行動随伴性	刺激	変化
仕事を丁寧にやる	報酬がもらえる	正の強化	報酬（好子）	行動増加
仕事を雑にやる	報酬がもらえない	負の弱化	報酬（好子）	行動減少
仕事を雑にやる	怒られる	正の弱化	怒られ（嫌子）	行動減少
仕事を丁寧にやる	怒られなくなる	負の強化	怒られ（嫌子）	行動増加

本来は「行動」と「結果」の前に「条件」という要素^{*1}があるのですが、ここでは説明を省きます。次の5つの原理から構成されます。

- スモールステップの原理
- 即時確認の原理
- 積極的反応の原理
- 自己ペースの原理
- 学習者検証の原理

それぞれの原理を見ていきます。

10.8.1 スモールステップの原理

学習のステップを細かく設定するほうが良い、というものです。

学習者が失敗しないように細かく設定します。何か失敗してしまうと、間違えた理解で定着してしまう危険性があります。

*1 三項随伴性と言います。オペラント条件付けの詳しい説明は主題から離れるので詳細は各自調べてください。

10.8.2 即時確認の原理

学習者が問題や質問に答えた際は、すぐに正否を知らせます。即時フィードバックですね。

次の問題や質問より前に学習者の反応が正しかったかどうかフィードバックすることで定着を促します。

正しいことをしたことがすぐにわかったことが学習意欲も向上しますよね。

10.8.3 積極的反応の原理

学習者がどのくらい理解しているかは問題への回答で判断します。

例えば学習者がいくら「完全に理解した！」と言ってもそれは外部に理解度が表出しているものではありません。

理解した内容を確認する問題に答えることで初めて学習の結果を測定します。

自己学習でもブログにアウトプットして理解の証跡にする等、応用できるかもしれません。

10.8.4 自己ペースの原理

学習のペースはみんな違うので、自分のペースで進められるようにカリキュラムを設定したほうが良いというものです。

適切なスピードは人によって違いますし、内容によっても違うと思います。「数学は得意で学習をどんどん進められたけれど、英語は苦手で」みたいなことはありますよね。

10.8.5 学習者検証の原理

学習プログラムの良し悪しは、あくまで当事者の学習結果で判断します。

専門家がどんなに太鼓判を押しても、実際に伝わらないのでは意味がありません。

未学習者が学習プログラムを使ってもらい、必要に応じて改善することで初めて良い学習プログラムになっていきます。

10.9 問題解決学習

問題解決学習は教育学者のジョン・デューイが提唱したもので課題解決型学習（PBL、Project-Based Learning）とも呼びます。知識の暗記という受動的なものではなく、「能動的に問題を発見し解決していく能力が学習の本質である」とするものです。

いわゆるアクティブ・ラーニングの一種です。アクティブ・ラーニングはグループ討論や実体験、他人に教えることなどを指します。これらは「講義を聞く」や「読書」に比べて学習定着率が高いとされています。

ところで、エンジニア文化としては OST（オープン・スペース・テクノロジー）がこの問題解決学習に当たります。

OST とは参加者が話したいテーマを決定し、オープンな話し合いを実施する形式で、答えのない課題に対して参加者の主体的な発案と対話によって進行していきます。

日々課題解決に奔走する私達技術者は常に問題解決学習を行っていると言っても過言ではないのかもしれません。

10.10 まとめ：これをもってどうしていくか

ここまで教育学的な学びのアプローチを列挙して紹介しました。

エンジニアが独学で勉強する場合を例にとり、ここまで内容を活かした学習プランを立ててみます。

まず学ぶ課題の選択は問題解決学習をベースにするものになります。「何か作りたいけれど、何が最適なんだろう？」とか「この言語、全くわからないけれど、どういう思想なんだろう」とかですね。

課題をプログラム学習の5原理に従い分解します。新しい言語を学ぶケースでは、基本文法と特徴的なデータ構造、標準関数など対象を区切っていき、スマールステップを形成します。

そして学ぶ側はツィラーの5段階教授法に従いステップを踏んで学んでいきます。学習対象を分析しまとめます。このへんは技術書を読んだりネットで調べたりです。

学んだことについては積極的反応の原理の通り、アウトプットすることで理解状況を確認します。メンターのような人がいるとなお良さそうですね。

既存の知識と比較、体系化し知識の定着を行います。そして、体系化した知識をベースに次の課題にステップアップしていきます。

このように進めていくと、自分の達成感にも繋がり、外部へのアウトプットによって学習が進捗しているかの検証もできます。

自己学習を例に取りましたが、新卒者や中途入社の方への研修担当者はかなり応用できる内容が多いと思います。

教材の作り方や教えるという意味では LT の構成などもラインの5段階教授法を参考にするなどできそうですね。

ぜひ、教育学の基礎知識を自己学習や研修に活かしてください。

大学と大学院の学びの違い

大学院に行くと、まだ勉強するの？とか、勉強好きね、と言われることがあります。

でも、大学院での学びは、それまでの学びとはちょっと毛色が違います。

注) 筆者は理工系出身ですので、まあ局所的な話だと思ってください。

11.1 大学で学ぶことは基本的には専門性の向上である

大学で学ぶことは何でしょう？

大学はレジャーランドであるといわれたのは遠い昔、最近はカリキュラムも厳しく、試験も厳しく、ともすると留年したり。講義も教授の雑談なども少なく、みっちりですね。

それでも、大学のそれぞれの講義で学ぶことは、基本的には誰かの解き明かしたこと、構築した理論体系をより詳しく、より難しいところまで学ぶ、となることでしょう。

そういう意味では、高校までの授業と本質的に大きく変わるものではありません。選択肢が増える、自分で学びたい内容を設計できるという意味での自由度は増えますけど。

11.2 大学院で学ぶことは「問題解決の手順」である

一方で、大学院で学ぶことは、問題解決のプロセスとその方法です。

新しい理論を導くとまではいかなくとも、なんらか大きな問題に対して実験や理論構築をし、その過程を論じて、修論にまとめる。その過程を一通り行うものです。

課題や問題設定は指導教員（教授など）から与えられたものかもしれません、なんらかの得られた結果を他の人に伝えるという点で、これまでの授業とは決定的に異なります。人に伝えるという点では、例えばグループワーク的なカリキュラムがあることもありますが、かける時間と、考察の深さ、また参考文献の調査の中で関係する過去の取り組みについて論じるなどの点も含めると、他とは相當に異なった経験となるでしょう。

11.3 それは当然仕事でも役に立つ

仕事、とくに研究・開発などの仕事は、課題を解決するための方法を検討し、実験・考証を行い、それを形にする、人に伝えるという点で共通点があります。もちろん大学院卒でないとできないというわけではありませんが、一度その経験があるのとのないのではだいぶ趣が異なりますね。

ということで、大学院で学ぶことは、単なる知識の深堀ではないんだよ、と考えます。

ま、博士行ったわけではないので、このくらいで。

第 12 章

学ぶ手法（資格・スクール・自習）のトレードオフについて

@ditflame

学ぶ手法には色々とありますが、社会人になってからの「学び」方でいうと、大きくは次の3つに集約されるのではないか？ と筆者は考えています。

それは

- 自習する
- スクールを利用する（他者の助けを借りる）
- 資格試験を使う

の3つです。

この章では、それぞれのメリット・デメリットについて考えてみましょう。

12.1 自習する

まずは自習です。一番簡単で手軽ですが、デメリットもあります。

12.1.1 自習のメリット

まずは、自習のメリットについて挙げてみましょう。

次のようなものが考えられます。

- 自分のペースで進められる
- 資格試験やスクールがなくとも（メンターが探せなくとも）学べる
- 学習費用は安い

12.1.2 自習のデメリット

続いて、自習のデメリットについて挙げてみましょう。

こちらも、次のようなものが考えられます。

- 自分のペースなのですぐ進まなくなったりする
- 満遍なく学べているかは自習だけだと保障しにくい

- ・時間がかかる（学びの最短コースを走れない事がままある）

12.1.3 自習について感じるアレやコレ

すべての学びの根っこともいえる「自習」なのですが、合うケースと合わないケースが明確に分かれるのが特徴的だな と思っています。

合うケースの代表格でいうと、時間が継続的に取りにくい状況下ではないかと思います。個人的に思い当たるケースでパッと思いつくのは

- ・仕事が多忙すぎて細切れ時間しか取れない
- ・育児中

などでしょうか。

逆に、合わないケースの代表格は

- ・（仕事などで）一定のタイミングまでに想定するレベルまでキャッチアップやスキル向上を終わらせる必要がある
- ・自分が得意でないジャンルのスキルや知見を仕込みたいケース

などがありそうです。

筆者としても、他分野の学びを始める時などでは特に顕著な気がしていますが、他分野の有識者に聞きながら学ぶほうが圧倒的に効率よく学びを進める事ができる事があります。

そういう学び方の代表格、「スクール」の場合はどうでしょうか？

12.2 スクール

スクールのメリット、デメリットについても検討してみましょう。

12.2.1 スクールのメリット

スクールのメリットは、ほぼほぼ自習で学習を進めた際のデメリットを潰せそうです。具体的には次のようなポイントが挙げられます。

- ・スクールにペースメーカーを任せられる
- ・お金が結構かかるので、それを理由に頑張れる
- ・プロに教えてもらえるので、俯瞰的な視点がスピーディに得られ、立ち上がりも早くなる。
詰まった時にも爆速で解決できる
- ・そんなこんなで、時間をお金で買えるケースが多い

12.2.2 スクールのデメリット

逆に、スクールのデメリットは次のようなものが挙げられます。

- スクールのカリキュラムに合わせて時間を見る必要があり、まとまった時間をコンスタントに取れないし利用しにくい
- スクールが開講されていないものについてはスクールでは学べない
- 教えてくれる先生が自分と合うかどうかは別問題
- 学習費用は結構かかる

12.2.3 スクールについて感じるアレやコレ

学習において、一番コストがかかる部類の学び方なので、メリットも多いのですが、最たるもののはこの一点でしょう。

時間をお金で買える

圧倒的メリットです。つよいつよすぎる。

なお、筆者の場合はクソむずい簿記（日商簿記1級）のスクールに通っていたことがあるのですが、当時は結構業務量が変動するタイプの仕事（受諾開発のSE）だったので、途中で超ハードな案件に突っ込まれ、途中から通学できなくなってしまったことがあります（＾＾；）

学校（大学）などでは半期毎などのタイミング休学などの措置もありますが、スクールはそういうの無い（そんなことやってたら経営が成り立たない）のでまあ致し方ないよなあ……と。

※ただ、そこで学んだ事は無駄にはなっていなくて、会計的な観点でのモノの見方が明らかに自分で増えたので、そういう意味では良かったです。

スクールについては、通信式コースなどではDVDなどを受け取れるものもあるので、仕事でそういうリスクがある人はそういうコースにするのも手です（すこし高くなりますが。そして通学の縛りが無くなると進捗が出にくくなるというのもあるあるで……）。

12.3 資格

さて、スクールを使わずに資格試験を受験し、その過程で学ぶ場合はどうでしょうか？

12.3.1 資格のメリット

実は、資格試験を使った学びはメリット・デメリットとともに自習とスクールの間ぐらになります。

メリットについて挙げてみましょう。

- 試験日があるので〆切効果が強い

- ・資格試験には絶対にシラバスがあるので、特定分野について横断的に過不足なく学べるようになっており、それが客観的に保障される
- ・受かると客観的な知識・スキルが証明できる（ものによっては名称独占や業務独占資格も！）
- ・そこまでコストがかからないケースも多い

12.3.2 資格のデメリット

逆にデメリットもあります（まあ何にでもメリット・デメリットというのはあるものでして……）。

- ・そもそも申込みを忘れるか受けられないというトラップがある *1
- ・試験日までに走り切る必要がある
- ・資格試験が存在しないものについては試験では学べない
- ・業務ではすぐに使わない、不要なことまで学ぶ必要が出てくるケースもままある
- ・受験は結構体力を使う
- ・会場試験の場合は「起床試験」、「会場到達試験」、「持ち物試験」もある *2
- ・自習よりはコストがかかる *3
- ・学習時間もそれなりにかかる

■コラム：未公表となっている翌年の試験日程を予測する

なお、大抵の定期的に行われる試験は、試験申し込み期限は3ヶ月ぐらい前になり、また毎年同様のスケジュールで運用されるケースが圧倒的に多いので、気になる試験がある場合は、昨年の同時期のスケジュールを確認すれば、おおよそのスケジュールが予測できます。

また、前述のとおり資格とスクールを組み合わせれば、より効率的に学びを進める事もできるでしょう。

なお、一般的には試験制度やそれを取り巻くエコシステムは、試験を受ける為に使うものですが、敢えて試験は受験せず、そのエコシステムだけをうまく使う方法もあります。

具体的には次のようなものです。

- ・労務関連の情報をまとめて知るために社労士試験の教科書を買って読む

*1 これを「申込試験」と言ったりします。同様の試験の受験までに乗り越えないといけないハードルとして「入金試験」「写真撮影試験&貼付試験」などがあります。

*2 朝起きれないと終わる。そもそも会場に到達する必要もあるし、受験票など必要物も忘れてはいけない 前日に必要な準備を終えておくのだ……！

*3 対策書籍買ったり、受験費もあるので自習よりは高くなります（とはいってもスクールに通うよりはよほど安いわけですが）

- kintone の使い方をまとめて学ぶために「kintone 認定アソシエイト試験対策テキスト」^{*4} を買って、本に沿って全機能を触る^{*5}

※なお、これらはどちらも筆者が観測した例です。

12.4 おわりに

仕事に関連するような学びの場合は、効率よく学びきってモノにしないといけないケースもままあるので、スクールや資格など、うまく使えるものは使って効率的に学んでいけると良いんじゃないか という気がします。

^{*4} <https://shop.cybozu.co.jp/products/8091452145888>

^{*5} まあ、これは kintone に限らないのですが、大抵のソフトウェア製品で資格試験があるものは同様のアプローチで最初の学習を加速する事ができるんじゃないかな？ という気がします

第 III 部

学びの体験談

学ぶ方法、得た学びは十人十色。

他の人は何を学び、どうやって学び、どう感じたのか。そういった学びの体験談が集まります。
参考にしたいところも、参考にしてそれを避けるべき話もあるかもしれません。

他の人の経験を見ることができるのは、やはりたくさんの人の体験が集まる合同誌ならではの部分ですね。むしろこの章が一番厚く、熱くなった章かもしれません。

IT エンジニアが初めての副業で得られた 学び

FORTE(フォルテ)@FORTEgp05

執筆者の FORTE(フォルテ) と申します。世の中は学びに満ち溢れていますね。本章では業界歴 17 年のもうベテランと呼ばれてしまう IT エンジニアが初めて副業先を探して実際に契約し業務を遂行、請求を行うまでにやったこと、分かったこと、次にやること、つまり YWT なふりかえりによる学びです。

2024 年 11 月現在、ご縁があった 1 社とお仕事をさせていただいております。今のところ年明けも継続できそうですので、実際の業務を始めてからの話もお伝えできたらと思います。

13.1 やったこと (Y)

まずは単純にやったことを列挙し、それぞれ詳細と注意点をお伝えしていきます。

- マイナンバーカードの申請
- 本業の会社に副業許可申請を出す
- 副業の始め方について学ぶ
- freee で各種手続き
- 開票届け
 - 青色申告の手続き
 - 事業用クレジットカードの作成
 - 事業用銀行口座の作成
- 適格請求書発行事業者登録
- 副業先を探すためにサービス登録
- カジュアル面談
- 選考用の書類の準備
- 採用面接
- 契約書の確認
- 実際の業務
- 納品
- 請求

13.1.1 マイナンバーカードの申請

これは必須ではないですが、せっかくの機会なので作成しました。数年前に役所から来ていた申請書から Web 申請を行い、近くの区民センターで受け取りました。

数年前の申請書ということもあり引っ越していて住所も変わっていたのですが、ちゃんと申請できました。受け取りは場所にもよりますが平日日中である事が予想されます。調整しましょう。

結果としてマイナンバーカードは作っておいて良かったというくらい使ったので、こだわりが無ければ作ることをオススメします。

13.1.2 本業の会社に副業許可申請を出す

これは申請不要な会社ならやらないで大丈夫です。ですが、副業を禁止している、あるいは条件がある会社もあるので必ず就業規則を確認して、必要なら社内の詳しそうな人に相談しましょう。

筆者は上司が副業をしており副業をオススメもされたこと也有ったので、その上司(役員)に聞きました。

申請書のフォーマットはそれぞれ違うと思うのですが、だいたい「何を」「どれくらい」「いつからいつまで」「理由」などを書くことになると思います。あとは(本業の)業務に影響ないことみたいな注意や制約事項が書いてあると思います。理由については悩みますが、私は素直に自己研鑽と所得向上のためと書いて受理されました。一例として参考にしてください。

13.1.3 副業の始め方について学ぶ

これは知人のITエンジニアに聞くのと、ネット検索を行いました。知人にはこの合同誌の寄稿している方々のSlackで質問し、ネット検索ではいくつか参考になる次のような記事を見つけました。

- <https://zenn.dev/kgsi/articles/55ec60e67c8177>
- <https://zenn.dev/nyancat/articles/20231115-side-hustle>
- <https://zenn.dev/naoist/articles/9810c9442b042d>
- <https://zenn.dev/taiyop/articles/212058258230f6>

これらを参考にこのあとにやるべき事を決めて取りかかりました。

13.1.4 freee で各種手続き

freeeでは副業や個人事業主を始めるのに便利なサービスが揃っています。役所への届出も含めオンラインで完結するので非常に楽です。

開票届け

詳細な手順は freee のサイトの通りです。私は屋号は作成しませんでした。どちらでも良かったのですが、考えるのが面倒だったのが主な理由です。それ以外はすんなり終わると思います。

青色申告の手続き

確定申告の際により多くの優遇措置を受けるため、青色申告の手続きを行います。これも freee の画面通りに進めるだけです。特に悩んだこともなかったので注意点もないと思います。

まだ実際に確定申告していないので、その辺りはまた別の機会に。

事業用クレジットカードの作成

青色申告、ひいてはそのための帳簿をつけるのにクレジットカードや銀行口座は生活用のものと副業用のものと分かれていた方が都合がいいので作成しました。

筆者は審査も通りやすく日数がかからないネットバンクのカードではなく、生活で使用しているメインバンクにカードにしました。これは口座間の送金手数料を無くしたかったのと、契約先を増やしたくなかったからです。その引き換えにカードが手元に届くまで 3 週間ほどかかりました。ですが、すぐに経費で買いたいものは何もなかったので特に影響はありませんでした。もし経費で PC を買いたいと思っているのなら、その辺りも考慮する必要がありそうです。

筆者のメインバンクも freee と提携していたようで、freee の画面から銀行の画面に飛びそこで手続きをしました。そのため freee の事業用カードによる申し込み扱いになっており、届いたカードにも freee と記載されているものでした。

13.1.5 事業用銀行口座の作成

カードで触れた通り freee でおすすめされているネットバンクのカードではなく、提携しているメインバンクのカードにしたため必然的に口座もメインバンクに新しく作りました。

口座は freee の画面からではなく、スマホから普段使用している銀行のアプリから口座を作成しすぐ作成できました。銀行によってスピード感は変わると思いますが、口座に関してはすぐに口座番号など別の申請に必要な情報を得られたので良かったです。

なお、freee で作成しなかったので freee による手続きに含めませんでしたが、カードの申し込みに口座情報が必要です。なので、先に口座がないとカードの申し込みができない場合があるので順番には注意しましょう。

13.1.6 適格請求書発行事業者の登録

いわゆるインボイスというやつです。これも必須ではないですし、デメリットもあると聞いたような気もしますが、副業先を探すサービスの一つで入力を求められたので登録してみました。

筆者はマイナンバーカードがあったので、e-Tax ホームページから案内に従って入力しました。特に悩んだり困った箇所はなかったですが、お役所特有の難しい言い回しには苦労しました。期間

を入力する際に条件によって期間が変わるので、その条件が良く分からなかった項目がありました。無事に登録できたので良かったですが、日本語は難しいです。

今のところ企業からインボイスについて聞かれたことはないので必要なのか分かりませんが、登録が完了するのに半月以上かかったので事前にやっておいて良かったとは思います。

13.1.7 副業先を探すためにサービス登録

副業先を探すために次のサービスに登録してみました。

- <https://shuumatu-worker.jp>
- <https://offers.jp>
- <https://www.lancers.jp>
- <https://www.wantedly.com/>
- <https://www.linkedin.com/>

この他にすでに登録済みであった Findy、YOUTRUST、LAPRAS も活用しています。

この中で最も反応が多いのは Offers です。企業からいいねが来ますが、正社員採用が前提であることが多く副業の有無を確認するとお祈りされるパターンが多いです。ただ、中には副業でお声がけいただいた企業さんもあり、カジュアル面談や採用面接まで進んだものもあります。

他のサービスはほとんど反応がありません。ただ LinkedIn だけはメッセージがよく来ますが、だいたいがエージェントからの営業メッセージであるので期待したものは来ないです。

13.1.8 企業に直接カジュアル面談を申し込む

副業を始める直前に技術系カンファレンスに参加し、その際のスポンサーブースで副業を始めようと思っているという話をしたところポジションがあるか分からないのでカジュアル面談がてら面談で聞いて欲しいと言われました。そのため、その企業さんのサイトから面談の申し込みをして面談予定です。

このように間にサービスを挟むことなく、直応募で探してみるのもいいでしょう。ただ多くの企業は正社員採用のためにカジュアル面談をやっていることが多いと思います。そのため可能なら応募時のやり取りや日程調整などの際に副業のポジションはあるか確認しておくと良いと思います。ポジションがなくても先方が良ければ話を聞いてみることをオススメします。自社以外のビジネス、開発文化、プロセスなどに触れるのは非常に有意義だと思います。

13.1.9 カジュアル面談

副業のカジュアル面談と言っても正社員採用のカジュアル面談と一緒にです。よく聞かれる質問としてなぜ転職を?がなぜ副業を?に変わるだけでしょう。採用面接ではないのでリラックスして事業や開発チームの説明を聞き疑問点を質問したり、なぜその開発プロセスを選んだのかを聞いてみたり、聞かれて答えられなかった事をメモしておき次は答えられるように準備しておきましょう。

筆者の実績としてカジュアル面談まで行けたのは 3 社、1 社は採用面接へ、1 社はカジュアル面談の冒頭でお互いの条件が不一致であることが判明したため終了、1 社はカジュアル面談後に持ち帰っての採用検討となりました。

面談冒頭で条件が不一致であると判明したのは先方から週 5 日 8h 日中稼働、週に 2 日出社の副業で大丈夫ですか?と条件確認をされたのに対し、あくまで本業あっての副業でありその条件では本業に影響が出るため難しいとお伝えしでは条件不一致なのでという事で終わりました。事前にメッセージでやりとりしていたので、そこで確認してくれればと思いつつ事情があったのだろうと思います。

持ち帰っての検討は採用プロセスとして面接がなく、送った書類や Github やポートフォリオなどのアウトプットを見てチーム内で検討とのことでした。結果としてはこの会社さんに採用いただきお仕事させて頂いております。

13.1.10 選考用の書類の準備

採用面接まで行きますと、履歴書と職務経歴書があったほうが良いでしょう。転職経験者なら過去に使用したものメンテすればいいと思いますが、初めて作る場合、特に職務経歴書は悩むと思います。

そこで転職ドラフトのように職務経歴書を無料でレビューしてくれるサービスで添削してもらう事をおすすめします。筆者も転職ドラフトで添削してもらったことがあります、それ以降の書類の通過率は非常に高く 9 割は通過している印象です。その分、面接で落ちることが多いのでこれは対策が必要と思っています。転職ドラフトは定期的にレビューしてくれるので、経験に更新があったら添削してもらうのも良いでしょう。

13.1.11 採用面接

まだ 1 次面接を 1 回だけですが、正社員採用の採用面接とあまり変わりませんでした。その企業さんに必要な技術 (ハード・ソフト両方) に関して質問されてそれに答えるというお馴染みのパターンです。

直近でやった面接で思った事は、過去に経験があるからといってその場ですぐに答えられるほど記憶に残っていないという事です。筆者はデータベース設計の経験があり、今でも設計はできるかと思いますが、現在の本業はインフラなので最近は全く使っていません。なのでトレンドの話題や、設計パターンなどについて問われてもパッと回答が出なかったです。

これは設計に限らず、フロントエンドエンジニアを採用したい企業さんは当然フロントエンドについて質問してきますが、前述の通り筆者はいまインフラをやっているので TypeScript のお作法や React の hook について説明とか言われてもパッと出てこないのです。オンライン面接だったので裏で調べながら回答しましたが、バリバリにフロントエンドやっている方と比較したらマイチだらうなと思います。

そのため、企業さんがどの分野の IT エンジニアを求めているのか、使用技術スタックから何に

ついて質問されそうか、この辺りは事前に調査、解答を作つておくと良いでしょう。当たり前に思えますが、筆者はやってなかったので、今後はやります。また全ての質問を事前に予想するのは不可能なので、想定外の質問内容は次回以降に別の企業では回答できるようにしておくのも必要です。いわゆる、ふりかえりです。

最後に採用面接に受からなかつたからといって落ち込む必要はないと思います。もちろん求められたレベルに達していないということもあるでしょうけども、単純に現時点での方向性の違い、比較してより魅力的な候補者がいた、タイミングが悪く席が埋まってしまったなどお祈りされる理由は様々です。現時点での方向性の違いというのはフロントエンドエンジニアを採用したかったけど例えばReactに関して求められるレベルではなかつたとか、その技術の即戦力が欲しいので今は未経験の人ではないとか、現時点で求めている方向性とアンマッチだったためお祈りされるパターンです。これは裏を返せば、タイミングが合えば欲しいと思われることでもあるのでそういう意味でも落ち込む必要はありません。反省点は改善すべきですが、気にするべきことではない事柄まで気にする必要はありません。それより次に行きましょう。

13.1.12 契約書類の確認

面談や面接に通れば採用連絡が来まして、次は契約に関する手続きを進めていくことになります。ここでは契約書について何を確認したかを紹介していきます。なお、これだけ確認すれば良いのではなく全体をきちんと確認した上で、特に確認すべき項目や観点を記載しています。

まず基本項目として次の項目を確認します。

- 契約形態
- 業務内容
- 期間
- 成果物
- 報酬(単価)
- 支払い期日と方法

ひとつずつ見ていきましょう。

契約形態

なによりもまず契約形態を確認します。この形態によってこのあと確認すべき項目も、要不要が変わる項目があります。一般的に業務委託での副業で考えられる契約形態は次のとおりだと思います。

- 準委任契約
 - 成果完成型
 - 履行割合型
- 請負契約

この中で気をつけたいのは請負契約と言われた場合です。これはいわゆる SIer が受けるような契約形態であり、いわゆる瑕疵担保責任（最近は契約不適合責任と言います）が適用されます。つまり納品したものに何か契約と不適合な点（問題）があれば 1 年以内は無償で対応する必要がある責任を負うものです。企業としてならアフターサービスや保守運用業務を引き受けことで対応可能かもしれません、個人の副業としては対応するには荷が重すぎます。契約終了後、11 ヶ月ほど経ってから開発環境もないシステムのバグについて責任を持って無償で修正しろと言われても無理難題です。そのため、契約形態は準委任契約一択と言えるでしょう。

準委任契約は 2 種類ありますが、これを大きく分けると成果物を持って報酬をもらう成果完成型と、作業した実績（時間）を持って報酬をもらう履行割合型となります。ただ、履行割合型でも作業した時間を報告するために報告書を実質的な成果物として求められる場合があります。これは実施的に成果完成型と言えますが、フルリモートでの作業が当たり前になった現在では作業時間を見ることが難しいため、お互いに合意を得るために許容できる範囲と言えるでしょう。実際に私も履行割合型で契約していますが、月末に作業報告書を提出しており、それを元に請求書を送付しています。

もし成果報酬型と言われたら成果物については明確に確認しておきましょう。

業務内容

ここでは自分が認識しておる業務内容と契約によって定められている（契約先から求められている）業務内容に不一致がないか確認しましょう。

特に注意したいのが、曖昧な文言や表現の場合です。例えば「〇〇システム開発業務とその支援業務」くらいなら大丈夫ですが、「〇〇システム開発関連業務」、「〇〇システム開発業務、およびその他業務」などの場合は具体的な業務内容を確認した方がいいでしょう。関連業務という表記では極論「関連している」と言われてしまえばどんな業務でも指示できてしまいます。もちろん断ることも自由ですが、話が違う！となるとお互いいい気分はしないですしトラブルの元なので事前にどういった業務かを確認しておき、極力その確認した業務に乗っ取った記載になるように必要があれば修正を依頼します。その他という記載は曖昧極まりなく業務内容が決まっていないから書いているか、悪意を持っている可能性すらあるので絶対に具体的な業務内容にしてもらいましょう。

期間

次に期間は意図が合っていれば特に確認すべき点はないのですが、期間が 1 ヶ月だと継続するのに毎月契約をむすぶ必要があるためお互いにコストが増えます。かといって半年や 1 年だと何かあった時に契約を切りづらくなります。私は 3 ヶ月契約で 2 ヶ月終わるくらいに継続するかどうか判断してもらっています。

契約書に独立した条項として契約の解除について書かれている場合もあるので、事前の確認と併せて内容をチェックしておきましょう。

成果物

成果完成型の場合は確実に、履行割合型の場合で契約書に記載がなくても事前に確認して合意をしておくと良いでしょう。もし成果物がソースコード一式などの場合は電子納品で良いのか、CD/DVDなどの物理メディアでの納品なのか、納品方法も確認しておくべきです。もしフルリモートのつもりで契約したのに納品が物理メディアでは出社する必要や、郵送などの対応が必要になってしまいます。今時はあまりないかとは思いますが、確認しておいて損はないでしょう。

報酬(単価)

これは人によって変わってくると思いますが、消費税の扱いについては確認しておきましょう。内税なのか外税なのかによって単価が変わってきます。実際の収入が10%減るのは大きいので、単価に対して消費税はどう扱われるか確認しておきましょう。

支払い期日と方法

これは月末締の翌月末払いのような、いつまでを計算期間としその計算期間の報酬がいつ支払されるのかを確認します。受け取る側としては早い方が嬉しいですが、相手にも経理業務などがありますので翌月末払いなどが多いのではないかでしょうか。万が一、契約書に明示されていない場合はきちんと確認して記載してもらいましょう。

今時は現金取っ払いということはないかと思いますが、銀行振込なのか、振込なら手数料はどちら持ちなのか確認しておきましょう。また、銀行振込でも銀行を指定されるケースがありますので、そこも確認と交渉が必要かもしれません。

13.1.13 実際の業務

契約が済めば実際に業務が始まります。企業によっては開発用の端末を指定される場合があるので、出社して取りに行ったり郵送してもらって受け取ったりして業務を開始します。

業務で気をつけるべきことは案件や現場によって変わりますので具体的なポイントより、汎用的な姿勢についてお伝えしたいと思います。伝聞ではありますが、あるミスにより契約を2週間で打ち切られてしまったケースがあるそうです。このミスはタスクを全力でこなさず、温度感が分からぬからと参考にした品質の低いコードのレベルでいいやと提出してしまったため、能力不足と判断されてしまったのが原因でした。いつも何事も全力で取り組む必要はありませんが、期待値の前例がない初回の提出物は念には念を入れておいて損はないでしょう。もちろん予め期待値を調整しておくのも重要です。これは急ぎのタスクなのか、じっくり取り組んでいいタスクなのか、そういった期待値を調整しておけばこのような事例は起こらないでしょう。

何事もそうですが、期待値の調整は非常に重要なと思います。

13.1.14 納品

契約書の成果物のとおりに納品物を用意し、指示された方法で納品します。契約書のとおりでないと受け付けてもらえず、報酬も貰えないのできちんと確認しておきましょう。最初はこまめに納品物の内容について確認してもらい、月末になってから慌てないようにすると良いでしょう。

13.1.15 請求

無事に納品できたら請求を行います。相手が個人でも企業でもそうですが、お金を払うにはそれ相応の根拠となる文章が必要になります。人生で請求書を書いた経験などないかもしれません、freee の有料サービスを使えばテンプレートが使えます。私が契約した 2024 年 10 月の時点では年間 23,958 円でした。これは請求書だけでなく、確定申告などもできるサービスなので文字通り必要経費として考えた方がいいでしょう。特に IT エンジニアの副業であれば月 3 万円は稼げると思いますし、年間で考えれば十分に価値があると思います。

もちろん自分で調べることで出費を抑えることは可能ですが、もうあなたは副業をしているのです。その無料にするための時間を副業の業務時間に置き換えたらいくらの単価になるか、計算してみてから判断しても遅くはないでしょう。

13.2 わかったこと (W)

やったことを通して分かったことを紹介します。

13.2.1 IT エンジニアの副業需要は多くない (2024/07 現在)

実際に副業先を探して分かったことですが、副業のみで募集している会社さんは少ないということです。多くは正社員を求めており、副業でも副業転職を前提としたものが多く、純粋な副業を求めている会社さんは非常に少ないと感じました。

これは知人から指摘された気づいたのですが、自社や自チームに副業で正社員と働き方が違うメンバーがいるとマネジメントコストが上がります。そこまでして副業で契約するなら最初から正社員で採用する、副業でも正社員と同じ働き方を求めるなど限られた時間だけ稼動する副業よりフルタイムで一緒に働く人材を求めるのは理にかなっています。筆者が過去に経験した自社サービスの会社はフルリモート・フルフレックスでありコアタイムもなかったため、稼動時間は社員の自由裁量でした。また居住地も自由であり入社後に沖縄に引っ越したり、海外に住んでいる社員もいました。それでも回っていたので、そういう文化がない企業にとってマネジメントコストが高いということに気づけませんでした。

13.2.2 転職に比べ焦りがない分、必死さも出ない

転職する際は大体がのっぴきならない理由や事情があってのことだと思います。筆者はやりたい事ができたり、やりたくないことをやらされる未来しかなさそうだったり、待遇に我慢ができないって転職してきました。どんな理由にせよそこには転職したい欲求があったため、人生の中でも優先度が高く転職活動を行ってきました。またお祈りされると原因を分析し手を変え品を変え採用を得るために努力してきました。しかし、副業先を探すとなると焦りもなければ必死さも出ません。

もし副業をしないと生活していくとなればもっと必死になりますが、そうであれば手段を選ばなくなります。極端な話、システム開発業務に限る必要はなく、お金がもらえる仕事であればなんでも良くなってしまいます。そうではなく生きていくために必須ではない副業を探すのは焦りもなく必死さも出ないため、どうしてもなあなあになってしまいがちです。筆者の場合、サービスに登録して企業からオファーがあったものは受けていますが、案件を探して自分から応募したのは1社のみです。またサービスの登録情報も質より数を優先したため、経歴が直近の一社だけしか登録していないサービスもあります。これはできたらいいなで行動しており、他にもっと優先度が高いものがあるとそちらを優先してしまうため受動的になっていると思います。つまり焦りもなければ必死さもないのです。

13.2.3 意外と情報が少ない

ITエンジニア 副業などでググると多数の情報がヒットしますが、個人のITエンジニアが副業先を探している情報はなかなか目に入りません。ひょっとしたら探し方や探す場所などが悪いのかかもしれません、それでも企業による不企業に関するPR情報ばかりで、あって副業を始める際にやった事くらいの情報です。いま副業先を探していてこういう事ができます、みたいな記事は全く出てきませんでした。

これも知人から言われた事ですが、いま副業をしているITエンジニアの方はリファラル、つまり知人からの紹介がきっかけな人が多いのかもしれません。そうであれば自分から探しに行かなくても相手から声をかけてくれるため、副業先を探すという事自体をやっていないのかもしれません。そうであればネット上に副業の始め方の情報はあっても探し方の情報がないのも納得です。リファラルでは再現性がなさすぎて情報になり得ないからです。筆者はITエンジニアは人の縁こそ大事だと思った事が何度もありましたが、今回もまた実感しました。

13.3 次にやること(T)

ではわかったことから、次にやることを考えてみます。

13.3.1 優先順位を決める

次に行動する内容を考える前に、自分の中でどれくらいの優先順位で行動するのかを決めます。最近はやりたい事が多く、また期限があるものを優先してしまいそれを言い訳に期限がないものは手がつかない状態になってしまっています。そんな中で行動内容を決めてても、やらないだけです。そのためどれくらいやるかを決めてそれを目指して行動して行きます。

筆者の場合、週 10 時間は副業のために当てようと思っていたため、まずはこれを目安にやってみます。平日 1 日 1 時間として 5 時間、土日休日に 5 時間としてやっていきます。いまは本業が忙しいため平日に 1 時間は正直厳しい日もあるのですが、実際に副業で業務を始めたらやらなくてはならないため今からやってみて感覚を掴んでいきます。

13.3.2 機会を増やすため能動的に行動する

あまり時間を割いてない分、最低限の行動だけだったのでもっと能動的に次のような行動をいろいろやってみます。

オファーをもらえる様に公開情報を増やす

副業先を探せるサービスに登録している情報を充実させます。シンプルですが、自分が仕事を依頼する立場になってみると情報が少ない人よりも多く登録されている人の方が判断材料が増えます。また業界的 17 年と書いているのに経歴が直近の 1 社や 2~3 年ほどしかないと怪しく見えたり、何か事情があるのかなと痛くもない腹を探られてしまいます。公開可能な情報は可能な限り公開し、また矛盾や疑問が湧かないようにした方が良さそうです。

副業を募集している企業を探して応募する

企業の採用ページに行くとカジュアル面談を応募できる場合があります。必ずしも副業を募集しているわけではないですが、話してみたら応募できるかもしれません。宝くじは買わなければ当たらないのと同じで、まずこちらの希望を伝えてみないことには始まりません。まずは門を叩いてみましょう。

なお、ホームページからカジュアル面談を申し込む際は日程調整など何かしらテキストでのやり取りが発生すると思います。その際にキチンと副業先を探している旨を伝えましょう。もし副業転職を考えているのなら半年や 1 年後に副業転職を考えていることも伝えると良いですし、逆に転職まで考えていないのならそれも伝えましょう。企業さんにとっても貴重なお時間をいただきるので、事前に調整可能な期待値は調整しておくべきです。その誠実さが評価される場合もあるでしょう。

露出を増やす

ここでいう露出は大きく 2 つあり、オンラインとオフラインです。

オンラインは SNS やブログ、やっていれば Podcast や動画などです。これらで副業先を探している事を発信し、IT エンジニアを探している方に触れる機会を増やします。副業先を探せるサービスは基本的に人事の方が見ていることが多いでしょう。それ以外の発信は直接 IT エンジニアに

届きます。その分、理解度や共感などが高くなり副業先を見つけられる可能性が上がると思われます。ただし、単に副業先を探してます！だけだと情報不足にも程があるので、キチンと自分ができることや条件と一緒に投稿しましょう。筆者はブログに情報をまとめた後、それと一緒に投稿する予定です。

オンラインは勉強会やカンファレンスに参加し、その参加者やスポンサー企業に自己紹介する際に可能なら副業先を探してますと伝えてみます。もちろんそれ目当てでやるとルールやマナー違反になるので、あくまで可能なら、です。個人的な所感としてはスポンサー企業の方は自社の宣伝に来ており、技術カンファレンスであればITエンジニアの採用に繋げるための採用なので話をしやすいと思います。また勉強会と違って土日に開催されるケースも多いので参加しやすいのもメリットです。個人相手は名刺を渡すくらいにして、聞かれない限りはアピールしない方がいいでしょう。自分が相手の立場でも勉強会で話した相手が開口一番仕事ないですか?と聞いてきたらあまり印象は良くないです。逆にこの人と一緒に仕事をしたいと思わせたら自然と流れが来るので、勉強会の目的に沿った会話をしましょう。なお、名刺にはSNSやブログのURLを貼っておき、後で見てもらった際に副業先を探してると気付いてもらえることも重要です。

13.4 そもそも、なぜ副業を始めようと思ったのか？

最後になぜ筆者が副業しようと思ったか書いておきます。

会社への申請書にも所得向上と書きましたが、これが1番の理由であり、上げた所得をNISAに入れたいと思ったからです。現状NISAは毎年満額投資すれば5年で枠を使い切れます。NISA自体は副業してまで焦ってやるものではないですが、投資による複利を考えると投資するのは早ければ早いほどお得といえます。一方で現在アラフォーの筆者は年々体力は低下していくばかりであり、5年後に副業するより1番若い今やった方が多くのバリューを出せるでしょう。また5年後に投資するよりいま投資した方が安く買え福利の効果も望めるとなれば、いまできることを最大限にやって最大の効果を得たいと思っています。

もうひとつは様々な開発現場を経験することで自分にプラスにしたいということもあります。これは学びという意味での経験もそうですし、もしあ互いに合意が得られれば副業転職も見えてきます。特に一生で一社しか所属しない人と、副業で様々な現場の経験がある人では学びのチャンスに差が出ると思います。そういう意味でも自分にプラスになると思うので、副業を通して様々なことを経験したいと思っています。

13.5 さいごに

実際に副業を始めて2ヶ月弱業務した学びをまとめてみました。筆者は経歴の割には技術力が低い方だと自分では思っています。これはある分野に特化した技術力がないという意味であり、スペシャリストよりジェネラリストであるという意味もあります。またそうでありたいとも思っています。そのため、副業のようなスポット的に入つて欲しい採用では不利な方に入るでしょう。

そんな不利な私でも実際にお仕事をいただき実際に業務までできたのは、何よりも私が行動した

からに他なりません。もちろんお互い求めている技術と提供可能な技術や働き方がマッチしたなどほかにも理由はあるのですが、それらは行動しなければ分からぬものでした。行動すれば必ず副業できるわけではありませんが、行動しなければ副業につながることは絶対にないということは言い切れると思います。特に今回はカンファレンスきっかけでの縁でしたので、本当にやってよかった、文字通り行動してよかったと思っています。

まだ実際に報酬は振り込まれていないのと、個人的に最も学びとなるであろう（そして多くの人が気にするであろう）確定申告については体験しておらず書けていないので、この続きをまたどこか別の機会にご報告させていただければと思います。

第 14 章

お出かけから得られた学び(仮)

FORTE(フォルテ)@FORTEgp05

執筆者の FORTE(フォルテ) と申します。もはや学びしかないこの世の中、みなさんお出かけしますか?引きこもってる?自分はインドア派?分かります。筆者もインドア派です。そんな筆者もさまざまな縁で外出する機会を得て、実際にお出かけすると自分が思っていたより気づきや学びがあったので共有したいと思います。

14.1 どうやってお出かけする機会を作るか

なかには暇だから目的もないけど出かけよう、適当に大きめの駅で降りてウインドウショッピングしてカフェにでも寄って帰るか、みたいな理由でお出かけできる人、逆に家にいられないという人もいるそうです。その逆に理由がなければ家で引きこもっていたい、ベッドでゴロゴロしながらスマホを見ていたい、食事は Uber か、なんなら食べなくてもいいみたいな人もいるわけです。そんな人がどうやったらお出かけする機会を作ることができるでしょうか?考えてみましょう。

14.1.1 好きなことや興味あること

自分が好きなものがあるのなら、それに関するイベントや催し物があるか探してみましょう。探すといっても公式サイトや作者のサイトや SNS を見るだけで十分です。そういったイベントは宣伝したいものなので News のトップなど 1 番目立つところに書いてあります。自分が好きなこと興味あることを発信しておくのも効果的です。併せてお誘い歓迎!とも発信しておきましょう。自分 1 人では集められる情報量や費やせる時間に限界がありますが、集合知を使えば 1 人の限界を突破できます。これによりあの作品のイベントやるんだ!行きたい!となる可能性が高まります。また自分が好きなものならば実際にお出かけできる可能性も上がるでしょう。

好きに新しいも古いもない

自分が好きなものは古い作品が多くてもう供給がないんだよな…と思ってる方、そんなことありません。たしかに昔の作品の供給は今リアルタイムでやってる作品の供給と比べれば少ないことが多いです。だからこそ、その少ない供給を逃してはいけないです。その作品の 20 周年や 30 周年はイベントや展示会などの供給が生まれる大チャンスです。実際にここ数年は筆者が子供の頃(今から 30 年ほど前)の作品の供給があります。単独イベントやフェスに参加など合同の場合もありますが、貴重なチャンスです。思い出はいつもきれいなのです。ぜひ少ないチャンスを流さないよ

うにしましょう。

14.1.2 unlearn とリフレーミング

それぞれ意味は違いますが、言いたいことは同じです。その思い込みを無くしましょうということです。自分は引きこもり、インドア派、地方住みだから遠いイベントは関係ない、どうせ行つたって大して楽しくない、面倒くさい、時間の無駄など自分の中でお出かけしない理由を作つてないでしょうか?今こそ unlearn やリフレーミングでそれらの思い込みを撲滅する時です。お出かけは必ず楽しいと言いたいのではありません。人がいっぱいいたり、電車が止まつたり、人に絡まれたりして嫌な思いをしたり出かけなきゃよかった…家から出なければこんなことにならなかつたと思うこともあるでしょう。同様に出かけて良かった! 楽しかった! と思えるのも出かけたからなのです。何事もすべてが自分の思い通りに行くことは少ないです。それでも、そこに自分が好きなことがあるのです。変わらない部屋の中の日常より、変化するスリルのある毎日を過ごしませんか?

言動と老化

これはお出かけに限らないですが、傾向としてまた実感として高齢な方ほど新しいことに忌避感を覚えている気がしませんか?逆に高齢な方がライブに行つたり、esports をやってしたりすると若いと思つたりしないでしょうか?概してそういったアクティブな方は外見も思考も若々しく柔軟であることが多い気がします。マザー・テレサではありませんが、人は言動から老化していくとも言えます。逆に言動が若い人は若々しくいられるということです。17歳を公言している某声優さんが実の娘の学生服が似合つてしまふのは一つの例と言えるでしょう。unlearn もリフレーミングも無意識な意思を意識的に変えていくことは難しいです。そこで他人に指摘してもらうのも手です。周囲に新しい体験を得ることをためらっているようなら指摘してくれとお願いしておきましょう。自分が気づいていない傾向に気づけるかもしれません。

14.1.3 真似る

学ぶは真似るからとも言いますが、お出かけ先に他人の真似をするというのもありだと思います。自分が興味あること好きなことに限定するとそれだけで対象が狭まってしまいます。そこで誰かが行って良かった場所、イベントに足を向けてみるのも一興です。そこから広がる世界に新しい自分の好きがあるかもしれません。筆者が見聞きした中ではスポーツ観戦があります。野球好きの人が野球に興味ない人に野球観戦を勧めていました。勧められた側はいやートイレがなあとか、暑いんでしきう?部屋でテレビでいいじゃんなどと言っていましたが(やらない理由を探している状態)、いざ行ってみるとそれらの懸念は払拭され存分に楽しめたようです。トイレは定期的に改修され新しくきれいで、ドームなら空調があるため快適、球場独自のフードや限定ビールなど楽しみ、何より一球一打に盛り上がる雰囲気が最高だったそうです。このように自分がそうでなくとも他人が好きなことを真似てみるのも一興です。特にお笑いライブ、舞台、展示会、美術館、映画な

どは楽しめるかどうか行くかどうかの判断が難しいので、自分で判断せずに真似てみるのも良いでしょう。

14.1.4 複利で広がる世界

複利とは投資用語ではありますが、ここでの意味としては一度お出かけするとその一回の労力で次もそのまた次のお出かけにつながるということです。例えば映画を見に行ったとしましょう。目的の映画を見る前には必ず予告編が流れます。いわゆるCMではありますが、意外と出会いがあるものです。自分が好きなジャンル、知らなかつた続編、好きな俳優や声優、監督の作品などきっかけは人によりますが、また映画館に来たいと思うことができます。これは予告編がない配信では体験できないことであり、また映画館という環境だからこそ体験できるものです。もし配信で予告編があったとしてもそれは単なるCMでしかなく意識することはないでしょう。あの巨大なスクリーンと部屋では体験できない音響だからこそ予告編でも得られる体験があると思います。他のお出かけ先も同様です。カフェや居酒屋であれば次はあのメニューにしよう、落ち着くから気分を変えたくなったらまた来ようなどお出かけには部屋で体験できないものがあります。そしてそれはあなたにしか得られない感情と体験なのです。静かなカフェでも人によっては落ち着かない騒がしい方がいいという人もいるでしょうし、その逆もあり得ます。投資しなければ得るものはなく、宝くじは買わなければ当たりません。利を得るには一度行くしかないのです。ねだるな、勝ち取れ、されば与えられん。の精神で行きましょう。

14.1.5 セレンディピティ

これは思いもよらなかつた偶然がもたらす幸運という意味の言葉ですが、まさに部屋にいては得ることが少ないものでしょう。あなたは気づいてしまったのです。そう、やすらぎよりも素晴らしいものがあることに。セレンディピティはその性質上、行動すればするほど得られる可能性が高くなります。また行動する思考の人は些細なことでも感じやすくなります。電車がすぐ来たり、トイレが空いていたり、雨に降られずに移動できたというだけで幸運を感じるようになります。その姿勢でいるとより多くの幸福が感じられるようになります。これも部屋にいては感じづらいでしょう。毎日ベッドでゴロゴロしている人が、今日はゴロゴロできて幸せだと思うことは少ないとします。むしろ今日もダラダラ過ごしてしまった…と自己嫌悪することの方が多いのではないでしょうか。これではセレンディピティには出会えませんし気付けません。果報は寝て待てとも言いますが、人事を尽くして天命を待つという言葉もあります。機会を増やすこと、そういうた行いそのものはいいことなのではないでしょうか。

14.1.6 締め切り駆動する

もしいつでもできることと今しかできないことの2つを比較するなら、今しかできないことを優先するのではないでしょうか。お出かけも同じでベッドでゴロゴロするのはいつでもできるが、お出かけ先はいつ無くなるかわからないし、コロナ禍のような好きに外出ができなくなるような事態

が発生する可能性があります。特に目的地がお店や自然のものである場合は、優先的に行った方が良いと言えるでしょう。何かを維持するにはコストがかかります。そのコストが払えなくなった時にその場所は無くなってしまうので、行きたいと少しでも思ったのなら仮でもいいのですぐ予定を立てましょう。筆者もまた行きたいと思っていたお店が閉店してしまい、またいつかではなく、すぐ行けば良かったと後悔したことがあります。

14.2 お出かけする

お出かけ先が見つかったら、意を決して実際にやってみることです。これがかなり重要です。行けるけど後でレポート記事や配信で見れるから現地はいいや…ではダメなのです。ここが分水嶺です。あなたが今までお出かけができない、難しいと思ってしまっているのはここです。行けるのに行かない、という判断があなたからお出かけを奪っているのです。人間は一度でも楽をするとズルズルと楽をし続けてしまいます。水は低きに流れるのと同じです。まずは一度行ってみましょう。

14.2.1 面倒なお出かけ前の準備対策

筆者は男性なので男性の場合はになってしまいますが、汎用的な部分はあると思います。いざ出かけようと思っても、出かけるまでにアレもやってコレもやってといいくつもステップがあったらモチベも下がってしまいます。それを防ぐにはなるべく家を出るまでにやることを少なくしておくことが重要です。スマホと鍵をつかんで出れるならハードルは非常に低いと言えます。ですが現実にはここまで準備ができる事はないので、一つずつステップを刻んでいきます。なお、これはどこまでなら許容できるかという主観の話なので唯一無二の正解はありません。自分がこれならできるなという範囲で出かける準備をしておくといいでしよう。まずは着替えです。多くの人が部屋にいる時は部屋着でいると思います。着替えるのがダルくて出かけられないなら着替えておけばいいわけです。同様に身だしなみや、持ち出す荷物などを自分が許容できる範囲で準備しておけばいつでも外出できます。その準備ができないから外出できないという人は準備が要らないように工夫するのもひとつの手です。部屋着は外出に耐えられるものにして可能な限りスマホなどに集約してしまえば準備不要でスマホさえ持てば外出可能になります。問題の解決方法は何も正面からあたるだけではないので、自分に合った方法を考えるのも手です。

14.3 具体的なお出かけ先紹介

抜粋ではありますが、筆者が実際にお出かけして良かったお出かけ先を紹介します。

14.3.1 カフェ

定番ですが、王道と呼ばれるには理由があります。実際に作家や作曲家などはあえて事実やスタジオではなく、お気に入りのカフェで作業をすることもあるそうです。お気に入りを探すには実際に行くしかないですが、きっかけとなる情報はネットにたくさんあるので好みの条件で当たりをつ

けて実際に行ってみることをおすすめします。お気に入りのカフェを探すこと自体をお出かけの理由にしても良いかもしれません。

14.3.2 居酒屋

個人的におすすめのお出かけです。お酒が飲めなくても居酒屋料理を目当てに行くのも悪くありません。かくいう孤独のグルメの五郎ちゃんが実践しています。鳥の唐揚げなどはどこでも食べられますが、ごま油付きのレバ刺しや珍味などは居酒屋の方が置いてある期待値が高いです。また焼き鳥屋さんおすすめです。締め用にご飯物を用意しているお店も多いので美味しい焼き鳥とご飯で夕食、なんてことも可能です。カフェと同じく非日常感を体験できるのがポイントかなと思います。

14.3.3 ショッピングモール

何か買い物したい目的がなくてもとりあえず行ってみるのにおすすめです。本屋さんや靴屋、洋服、輸入物を扱っているお店、乾物屋、インテリア小物などいくつか覗くだけで何か発見があるかもしれません。また季節に関わらず快適でトイレもあり休憩スペースもありお腹が空けばご飯も食べられます。併設されていれば映画館もあるし、中には駄菓子屋などもあるので特定の目的はないが出かけたい時にぴったりです。

14.3.4 高級旅館に泊まる体験

旅行先を選ぶ時はどこに行くか?を考えがちですが、泊まる体験そのもを目的にするものあります。ホテルに泊まるだけでも非日常感があって良いのですが、高級旅館に泊まる体験というのは一度体験しておくのをおすすめします。高級と言ってもピンキリなので値段とサービスで決めれば良いのですが、観光地や温泉ではなく泊まる行為自体を目的にして考えるとまた違った楽しさと体験があります。

14.3.5 音楽ライブ・コンサート

ライブに行かない人、行ったことがない人はわからないと思いますが、同じ音楽を聴くという行為でも現地に行くのはまったく違います。こればかりは言葉で説明しても伝わるものではないで、ぜひ自分が好きなアーティストさんのライブがないか検索してみてください。すでに引退や解散てしまっている方もいますが、現役で今もライブをやっている方がいます。自分が子供の頃に聞いて大好きだったあの曲、今でも聞いてるあの曲をライブで聴くという体験は非常に素晴らしい体験です。ぜひとも一度は足を運んでみてください。

14.3.6 デパ地下

ちょっと大きい駅の駅ビルにあるデパ地下に行って、普段は買わないようなお惣菜やお菓子を買って帰るのも楽しいものです。コンビニや近所のスーパーでは買えないような料理や美味しさが体験できます。特に珍しいお茶やお菓子などが置いてある場合があるので是非とも見てほしいです。また何よりもあのデパ地下の雰囲気が良いと思います。普段の生活というよりは特別なお出かけ感がある感じがたまりません。デパートによっては軽くお酒が飲める場所もあるので、そんな体験も良いでしょう。

14.4 お出かけから得られた学び

一度お出かけをして良い体験をするとまた次もどこかへ行こうとなります。そしてまた良い体験をしてさらに出かける気になると正のループを回せます。逆に出かけることができずずっと引きこもっていると負のループになってしまい、出かけることが難しくなってしまいます。ここで言いたいのは出かけるべきだ、ということではなく行動の選択肢が多い方がいいので出かけてもいいし出かけなくてもいい、どちらも選べる状態にしておいた方が幸せなのではないか、ということです。人は選べない環境にいるとストレスを感じると言います。逆に主体性を持って選んでいるとストレスが少ないそうです。自らが選んで部屋でゴロゴロするのと、本当は他にやりたい事があるのにゴロゴロしかできないので違うというのはわかっていていただける人が多いと思います。何も毎月毎週毎日のように出かけなくてもいいのです。ただ出かけたいなと思った時に出かけられる方がよりQOLが上がるかなと思います。

14.4.1 お出かけ先ストックが増える

これは鶏が先か卵が先かという話になってしまいますが、お出かけ回数が増えることでお出かけ先となる候補、つまりストックが増えます。これはその時の体調や気分によってお出かけできる可能性を増やしてくれます。もしベッドでゴロゴロしかできないほどでも、そういえばあのお店はゆっくりできたなど居心地が良くて行きたいなど、散歩なら頭空っぽになっていいかもとか選択肢を増やすことができます。この体調や気分で選ぶお出かけ先は自らが体験したお出かけ先でしか選択肢になりません。他人が疲れた時に行ける場所と自分が行ける場所は違いますし、そういう時に何を求めるかも違います。そのため実際にやっておいて体感しておくことが重要になります。

プランBに使える

行こうと思っていたお店が急遽お休みだった場合に代わりにあのお店にしようとか、そういえば最近は行ってなかったあのお店にするなどプランBができます。特にカフェや居酒屋、食事などあったく同じお店はなくても同系統だったり別ベクトルのお店を選ぶことができます。これがストックがないと帰る以外の手段がなくなってしまい、せっかく外に出たのに無駄足だったで終わってしまう危険性が高まります。もちろん新規開拓として新しいお店に飛び込んでみることもできますが、それが気軽にできていればお出かけに関して悩むことは少ないでしょう。なかなかお出かけ

できない人が意を決してお店まで行ったら休みだったことを考えると、次善の策としてプランB、次のお店を考えることができるのはお出かけの際の強力な味方となるでしょう。

14.4.2 無料のものがある

お出かけといえばお金がかかるイメージでいました。しかし中には無料で体験できるものもあります。当然有料のものと比べるものでもないですし、無料だから行くものではありません。興味があつたり面白などがある前提の話ではありますが、人によっては支払ったお金で得られる価値が釣り合っていないと不満に感じる人もいるのも事実です。そういう人にとっては自分が面白いかどうかわからない未知のジャンルに投資するのはリスクが高いといえます。しかし無料ならその点を気にせずに体験できて、自分に合うかどうか確認できます。もし費用が気になるなら無料のものを探してみるのも一つの手です。

14.5 選択肢という手札を増やそう

休日や暇な時間に家でゴロゴロする以外の選択肢がないより、いくつもある中から選べるのはその時の体調や気分に合わせた時間を過ごすことができると言えます。これはそうすべきという話ではなく、もし魅力を感じたらやってみたらいいかもしない程度の話ではあります。しかし、それを知っているのと知らないのでは大きな差になると言えるかもしれません。それこそが学びと言えるのではないでしょうか。

夏休み・春休みに高度情報試験を取った話

大学生をしていると、夏休みや春休みは授業がなくなるので多くの時間が取れるようになります。せっかくまとまった勉強時間が取れたので、その他の領域の知識について体系だった勉強をしたいと思いました。一つの目安として「資格取得」を設け、2ヶ月という短期集中で高度情報試験を受験したこと共有します。

15.1 バックグラウンド

IT 業界外で働く私にとって、サブスキルの IT を学ぶことはあくまでも趣味です。以下の資格は主業務で取り扱うものではなく、もちろん昇格要件ではありません。

ただ、IT に詳しいポジションにいるため時々頼られます。頼られたのに応えられないというのも悲しいので、自己啓発として以下の 2 つの資格を受けました。

15.1.1 データベーススペシャリスト(DB)

仕事で職場でノーコードデータベースを触るのですが、実際に理論だって学習をしたことが無く、オレオレ実装になっていました。

思いつく範囲で最良の構造にしているつもりではありますが、自分の知らない構造があるかもしれないという思いがありました。

これまで数多の DB エンジニアがこの問題にチャレンジし理論として体系化してきたはずと信じ、データベースの勉強をしたいという思いがありました。

とはいっても、申し込みは放送大学学期中であり、すぐに対策を始められません。また、Twitter(現:X)などで無為に時間を過ごしてしまい、しばらくは勉強を開始しませんでした。

2022 年の夏コミ、東待機列に並んでいる最中におすすめの参考書^{*1}を教えてもらい、勉強を始めました。

コミケに並んでいる時点でのスキルは以下のようないい感じです。良くこれで受けようと思ったなと言われそうですが、そもそも問題集をここまで見ていないのでダメダメということすら知りません。

DB を取得したい理由が、会社のデータベースを効率よく使いたいという理由のため、以下のよ

^{*1} 翔泳社さんの情報処理教科書シリーズ。この参考書は試験に合格することに強めにフォーカスしているので、好き嫌いが分かれるかもしれません。

表 15.1: DB の領域と修得状況

分野	修得状況
運用経験	PHP から MySQL を触ったことがある。
SQL	SELECT、COUNT と CREATE は知ってる。JOIN、DISTINCT 分からん。
概念データモデル	ER 図を見て「どういうこと?」というレベル。
関係スキーマ	初めて聞いた。

うに方針を立てました。

- SQL は業務では触らないので、試験に合格できる最低限にする。午後でぶち当たった時は合格をあきらめよう。
- 概念データモデル、関係スキーマはしっかりやる。特にビジネスロジックは使えそうなのでしっかり理解すること。
- 午前は記憶力で何とかする。

平日はまとまった時間が取れないため、通勤の行き帰りに過去問道場を解く習慣づけを行い、午前の対策を行いました。

過去問道場は大体 80% ぐらいの正答率になったと思います。(最後のあたりは問題自体を記憶し始めた) 土日は午後 2 なら 2 問/日、午後 1 なら 3 問/日で週 5 問程度を目安に解きました。

15.1.2 情報処理安全確保支援士 (SC)

同人誌でアンケートやりたいなどとなった時に、個人情報を守るのは必須技術だと思います。できないなら「アンケートをやらない」などを検討・決定しなければいけません。

また、仕事では「メールの From は偽称可能か」などのフィッシング、パスワードの奪取などのリスクにさらされ、IT に強いと思われているためかよく質問を受けます。

正しく理解し答えることができるようになりたいと感じることが多々あります。DB と同様に、学ぶことを強制するように何かしらの資格を取ろうと思い、それっぽい名前をしているこの資格を受けてみることにしました。

PHP などを触っていた関係から、SQL インジェクション、XSS、DoS などの攻撃は知っていました。ただ、それ以外はかかわったことが無いので、学ばないといけない状況です。DB に対してセキュリティはかかわる領域が広く、何から手を付けるかすら決まりませんでした。

DB は例年どの分野が出るか傾向があったのですが、SC はどこが出るのかわからなかったので、テキスト (DB のが良かったので、今回も翔泳社さんの対策本) を総なめして、過去問を解きまくりました。

午前 1 が無かったのでその分楽でした。どうしても攻撃の実感が持てないと覚えにくかったので、攻撃の事例とそれに対する防衛法をネットで調べたりしながら進めた気がします(うろ覚え)。

15.2 結果

両試験に共通しますが、受験日までにどの問題を解くなどのスケジュールを立てて挑んでいました。それが自分がサボらないようにする抑止力になっていたと思います。

また、長期休みではあるものの2か月程度のうちに一気に仕上げるため、中だるみしにくかったのも良かったと感じます。鉄は熱いうちに打ったほうがきれいに良く伸びます。

また、勉強するモチベーションとして、何に役立つかわからないと他の誘惑に負けやすいです。私の一つの解は、無心になることです。

通勤するときは過去問道場を繰り返し行うこと、土日になったら午前に1つ午後に1つ解くこと、というのを徹底してこれに疑問を覚えないようにします。家に帰ったら手洗いうがいをするのと同レベルに過去問を解くという生活サイクルを作るとよいと思います。(この点で通勤という定例作業があるのは良いかもしれません。)

残念ながら、私の会社では一緒に受けようという機運などが無かったため、完全な独学での受験となりました。

前項の認定心理士課程ではスクーリングで同志に会うことができる楽しみがありましたが、高度情報試験ではまるで機械のように問題を解くだけでした。

どちらがいいかは人にもよるかもしれません、「遠くに行きたいのならば仲間と行きなさい、早く行きたいのならば一人で行きなさい」という言葉のように、短期集中で行うならば一人の方が良いかと思いました。

試験はゴールではなく、そのあとにいかに使うかが重要です。会社のDBは完全なRDBではなく、まだアンケートを取るようなこともしていました。

しかし、SCを取ったことをきっかけにそういったニュースを見ることに興味を持つようになりました。どちらかというと、DB取っているのだから知っているべきではなど、技術を追いかけるという業を背負いこんだ気がします。

それもまた、強制力を持った学びの入り口かもしれません。

第 16 章

緑内障になってから変わった心境とインプット方法

16.1 はじめに

Camo(カモ)@camomile_cafe

本章を執筆した Camo(カモ) です。初めての方が多いと思いますので簡単に自己紹介をさせてください。

自分は韓国から日本に留学生で来てからかれこれ 12 年くらい経つ者です。

大学では情報学部とは全然違う学問を専攻して社会人になってからエンジニアになったケースの人です。

社会人になってからエンジニアとしての仕事を始めたこともあり、PC と向かって独学で勉強したり、勉強会に顔を出して知見を得たりしてキャリアを積んできました。

そんな自分が、ここ最近眼科で検査を受けた際に緑内障の診断を受けました。

そういう目の病気はまだ先の話かなと思っていたところ、自分にもそういうことがおきたことで、いろいろと心境や行動に変化が起きました。

今回は、緑内障になったことによる心境の変化や普段の生活において見方が変わったこととそこから学んだことについて紹介したいと思います。

16.1.1 緑内障の発覚

自分は眼科でコンタクトレンズ着用の検査を 3 回ほど受けましたが、その都度「眼圧が高い」と診断され、コンタクトレンズをつけることができませんでした。

今度こそ着用できるだらうと思い、今年の 5 月に眼圧が正常になったと眼科で診断を受け、これからはメガネをかけずに出かけられると嬉しく思っていましたが、眼科で別の検査も一緒に受けた際に先生から緑内障であると診断されました。

普段モニターに張り付いて目を酷使するような仕事をしているため、眼精疲労を感じることが多く、いつかは目に関する病気を経験するだらうと思っていましたが、30 代で緑内障と診断されるのは予想外で突然のことでした。

緑内障について調べてみると、どうやら 40 歳以上の 20 人に 1 人が緑内障だと診断されているそうです。^{*1}

^{*1} <https://www.jaaikosei.or.jp/magazine/1221/>

眼科の先生によると、この「20人に1人」というのは、実際に眼科で診断された人の割合であり、眼科を受診していない人も含めると、実際の罹患率は**20人に3~4人**、あるいはそれ以上である可能性もあるとのことです。

健康診断では見つかりにくい病気であるため、実際の罹患率は高いと考えられます。

また、緑内障は日本で**中途失明の原因の第1位**とされている目の病気です。^{*2}

緑内障は、目と脳をつなぐ視神経がダメージを受けることで、徐々に視野が狭くなり、最悪の場合失明に至ることがあります。

初期段階では自覚症状がほとんどないため、気づかないうちに進行してしまうことが多いです。

視野が欠けてくるという特徴的な症状が現れたときには、すでに進行していることが多いのです。

緑内障の原因は多岐にわたりますが、自分の場合は眼圧が高いために視神経が圧迫され、徐々に視野の見えている部分が狭まるタイプでした。

緑内障の治療法としては、眼圧を下げるための点眼薬や内服薬、レーザー治療、手術などがあります。

ただ、治療とはいえ、眼圧の上昇によって一度障害されてしまった視神経の修復は困難とされます。^{*3}

つまり、緑内障は完治する病気ではなく生涯にわたり視機能を保つようにするために生涯通院することになります。

自分の場合、たまたま早期に発見できたことと眼圧を下げるための点眼薬を投与することで進行の具合を抑えられる段階でして、これから定期的な眼科検診を受けながら過ごすことになりました。

振り返ってみると、もし検査を受ける機会がなければ、緑内障とは知らずに「目が疲れているな」と感じるだけで、徐々に視力が失われていったかもしれないです。

そこで、これを読んでいる読者の皆さんには、特に20代や30代の方で、**最近眼精疲労を感じる、目の異常を感じるなど**、何らかの異常を感じたら、ぜひ眼科に行って緑内障検査を受けることを強くおすすめします。

16.1.2 目の病気があることを受け入れる

一生使う目が病気になるという経験は初めてで、放心したりやけになったりするかと思いましたが、案外自分が緑内障だと診断されたことについて受け入れるのが早かったです。

心理学には、喪失体験がもたらす「喪失の五段階」というプロセスがありますが、自分はそのプロセスが早く収束したように感じました。

喪失の五段階とは、エリザベス・キューブラー＝ロスが1969年に提唱した喪失体験のモデルで死や重大な病気を経験した人々の感情のプロセスを理解するためのものです。現在では本人の病気

^{*2} <https://www.nhk.jp/p/kyonokenko/ts/83KL2X1J32/episode/te/6PV36ZZ8NN/>

^{*3} <https://www.daiyukai.or.jp/department/ganka-ryokunaisho#h30>

や失業による役割の喪失なども含めた様々な種類の喪失体験にも応用されています。 *4

具体的には、それぞれの段階で以下のような経験をします。

- 否認

- 「こんなことが起きるなんてありえない！」と現実を否定する段階。

- 怒り

- 「なんで私がこんな目に遭わなければならないのか！」と怒りを感じる段階。

- 取引

- 「何でもしますのでどうか助けてください！」と取引しようとする段階。

- 抑うつ

- どうしようもない現実に対して深い悲しみや抑うつを感じる段階。

- 受容

- 現実を受け入れ、前向きに生きていこうとする段階。

自分の場合、この喪失の五段階のうち、四段階までが驚くほど早く過ぎ去りました。

現実の否定から抑うつ感を覚えるところまで、体感として1日もかからなかったのです。

普段健康的な生活を心がけるため、早朝から歩いたり、こまめに目を休ませたりする習慣があるので、最初は何かの間違いだと思っていました。

これまでの努力が無駄だったのかとイライラしたり、何をすればよかったのかと考えたりしました。

毎日健康管理をしても病気になるときはなるんだと思い、抑うつを感じながらも自分の現状について理解しようと努め、これからることをどう考えるべきか悩んだりもしました。

この時に思ったのは、**体の調子や病気は自分の理解を超えた何かによって訪れる**ということです。

エンジニア的な表現で例えるならば、人の身体はあらかじめプログラミングされていて、遺伝や生活習慣で多少の変化は加えられるものの、大きな出来事が起きるシナリオは変えられないという感覚を覚えました。

今回もそのような感覚でいられたことで、喪失の五段階のうち、四段階目まで早く進むことができたのではないかと思いました。

16.1.3 次は何をするのかについて考える

喪失の五段階のうち、四段階までは自分で消化したものの、一番難しかったのは五段階目の受容でした。

仕事やプライベートにかかわらず、生活の中で目に依存してやってきたことがいかに多いのかをしみじみと感じ、それらができなくなるのが恐ろしいと思いました。

キャリア、物理的な移動、新しい知識のインプットなど、目から受け取る情報の多さと、それが

*4 https://www.jamh.gr.jp/kokoro/147_kan.html

享受できなくなることを考えると、目が見えなくてできないことが増えることについていかに自分が納得できるかが一つの大きな問い合わせとなりました。

ただ、まったく答えが見つからない問題かというとそうでもないと思いました。

というのも、緑内障はある日突然目が見えなくなるわけではなく、目の見えない部分が徐々に広がる病気です。

現代医学で治すことはできないものの、緑内障の症状を遅らせることはできます。

つまり、最悪のパターンとして失明によってまったく見えない状況を考えるべきですが、それに至るまでの過程で見えている部分が多い・少ないときにやれること・できることについてフォーカスを当てて考えると、少なくとも次にやることは決められるのではないかと思いました。

例えば、「キャリア・仕事をどうするのか」に関して、エンジニアとしてやってきた経験を活かし続ける場合、緑内障の後期・末期や失明までは見える部分が半分以上あるので今の仕事を続けることができると言えました。

コーディング、コードレビュー、仕様書の確認と作成、コミュニケーションツールの利用など、目に依存する割合が多い業務に関しても、緑内障の中期でも両目で視野を補っているため、モニターから文字を見ることは可能です。

また、緑内障の進行度にもよりますが、障害者雇用としてIT業務のサポートに就くことも手段の一つです。

開発がメインではなくなったり、データ入力や急ぎではない仕事を担当したりするなど、これまでの業務とは異なるかもしれません、IT業界の仕事には関わり続けることができる可能性があります。

ただ、後期・末期になると日常生活に支障をきたすほど視野の見えない部分が広がるので、その時はモニターを見なくてもいい仕事に就くのが良いかもしれませんと思います。

例えば、業種・職種ともエンジニアとは異なりますが、あはき師という選択肢もあります。

「あはき」とは、あん摩マッサージ指圧、はり、きゅうの略称で、日本では国家資格免許を有する者がなることができます。^{*5}

免許取得には専門的な養成施設で一定期間のカリキュラムを修了し、国家試験に合格するという条件がありますが、免許を取得すると自宅開業や治療院・接骨院への就職が可能ですので、就業の手段として考えられます。

ほかには、物理的な移動、新しい知識のインプットにおいて制約がかかった状態で生活することになりますが、考え方を変えるとこれからの行動において制約がかかるほうに優先順位を上げて行動することも可能だと思いました。

「いつかやる」と決めていたことは多分そのときになればやると思いますが、自分の目で見てやることに関してはタイムリミットがかかった状態で、しかもそれがいつ訪れるか分からない状態で行動しないといけなくなります。

そのため、いつか行こうと思っていたところに行けるように週末の計画を立てるとか、後で読もうと思っていた本を今のうちに読むみたいな考え方で身軽にやる姿勢は持てるようになったのでは

^{*5} <http://nichimou.org/employment/ahaki-work/>

ないかと感じます。

また、それらの活動をすることになにか意味を見出そうとする考え方よりも深く考えるようになりました。

「いつかやる」と思っていたことを気軽にやることに加えて、同じ体験でも目が見えているときと徐々に見えなくなるときで体験の質が変わるだろうと考えました。

人の五感はどこか一箇所の機能が衰えると別の感覚のほうが鋭くなると言われますが、目の見え方によって自分の感じ方も変わるだろうと思ったためです。

目が見えているからこそ視覚情報による美しさの基準や色・形の好みが優先され、感じ方がある程度固定化されることがありましたが、これからは同じ体験でも視覚情報に依存しない感じ方をすることで新しい感覚が得られるのではないかという期待もあります。

そのような感覚を得るためにも、目が見えているうちに「自分は目で見たときにこのような感じ方をするんだな」という体験から感じ方の意味を導くことを大事にすることで、徐々に視覚情報に依存しない感じ方をしたとき、自分が大事にすることやいいと思っていることの基準がさらにはっきりしていくと思っています。

喪失の五段階の最後の段階で今までできていたことができなくなるという体験をこれからすることになる前提で「これから仕事観」や「心境の変化とともに感じ方の変化」について書いてみました。

どちらも今までの価値観から別の価値観へシフトすることを前提に、自分はどう思ったのか、また次のプランとしてどんなことを考えたのかについて整理してみた結果、心身ともに変化が訪れても自分なりに大事にしたいことを導き出すための考え方を柔軟に変えられると感じました。

緑内障の発覚は心境の変化としてはあまりポジティブではない経験でしたが、それでも生きていくために次は何をするのかを考える良い機会だったと改めて感じました。

16.2 少し変わってきた生活習慣

16.2.1 耳を使ったインプット

普段のインプットでは、目を使ったやり方がやはり多いのではないかと思います。

実際、自分も本、電子書籍、ブログ、YouTubeなどから新しい情報を得ることが多いです。

しかし、目の負担を減らすために、目を使わずにインプットできる方法があるといいなと思っていました。

そこで、現在使用しているスマホやPCのユーザー補助機能（アクセシビリティ機能）を使えば、目を使わずに情報をインプットする方法があるのではないかと調べてみました。

ユーザー補助機能の中には読み上げ機能があり、目の不自由な方や視力の弱い方に対して補助的なUIを表示したり、音声で案内するフィードバックを提供するものがあります。

スマホの機種によって有効化する機能名は異なり、iOSの場合は読み上げコンテンツ、Androidの場合はTalkBackを有効にすることで使用できます。

iPhoneの場合の設定方法



図 16.1: iOS では読み上げコンテンツを有効化する

Android の場合の設定方法

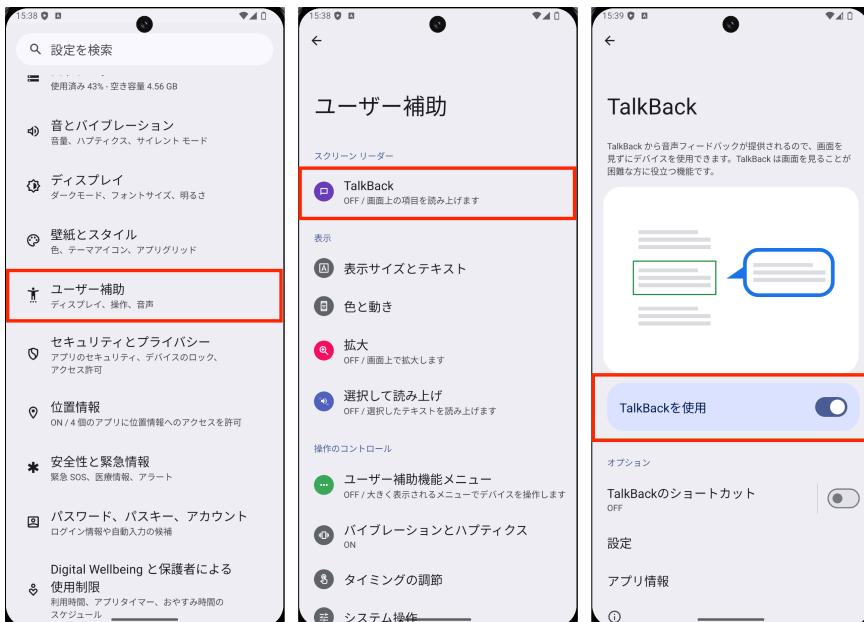


図 16.2: Android では TalkBack を有効化する

自分が普段使っているデバイスは Android で TalkBack を使っていまして、iPhone の読み上げコンテンツ機能に関する詳細は割愛します。

下記で説明するコンテンツの読み上げは Kindle での使い方を例にしていますので、詳しく知りたい方は Kindle の読み上げ機能を使用してコンテンツを聞く^{*6}を参考にしていただければ幸いです。

Android の場合、TalkBack をオンにすると、どの UI 要素にフォーカスが当たっているのかを振動や色付きの枠で知らせてくれます。一度オンにすると、タップで UI の選択、ダブルタップでその UI 要素に対して一般的なタップ操作を行うことができます。



図 16.3: TalkBack を有効化すると、現在選択している UI 要素対してハイライト表示をする

また、TalkBack を設定からその都度起動するのではなく、ユーザー補助機能ボタンのショート

*6 <https://www.amazon.co.jp/gp/help/customer/display.html?nodeId=G201829850>

カットとして登録しておけば、読み上げ機能を使いたいアプリを起動してから即座に使用することもできます。

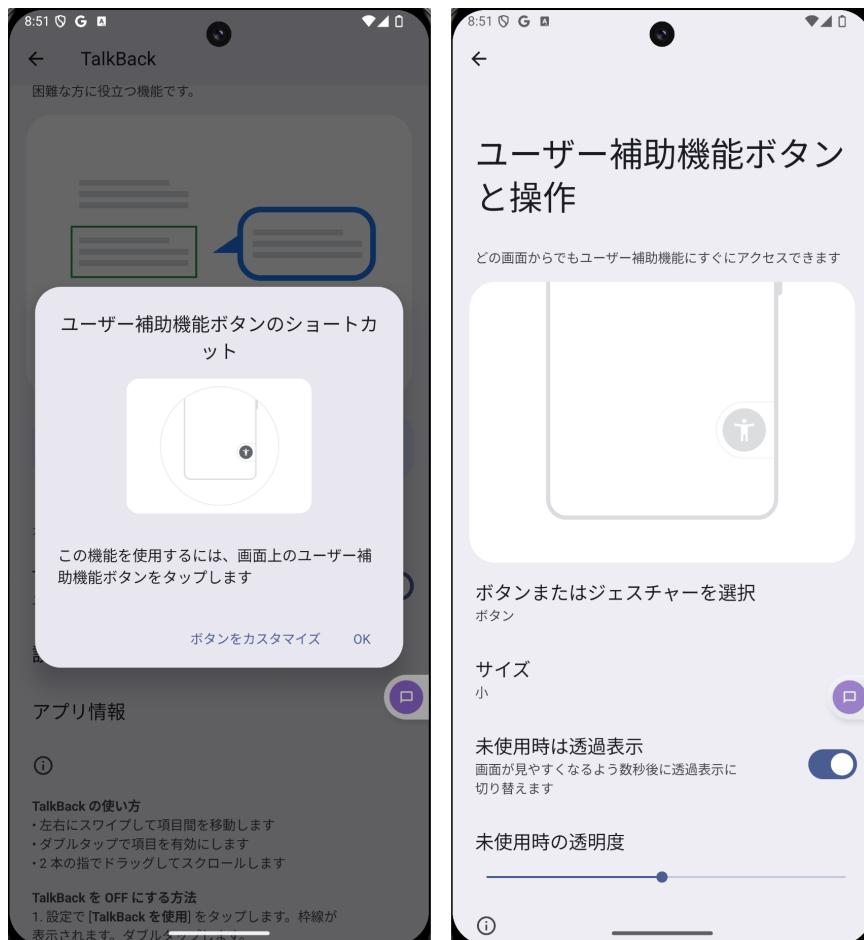


図 16.4: ユーザー補助機能を有効化すると、ツールチップのようなショートカットが表示される

この状態で文字を読み上げてほしいアプリを起動し、ショートカットで TalkBack をオンになると、文字として認識できるコンテンツであれば、読み上げてくれます。

自分がよく使う Kindle の場合、「Text-to-Speech（テキスト読み上げ機能）」が有効になっている電子書籍であれば、読み上げ機能を使用することができます。

登録情報

ASIN : B082H8GWRD

出版社 : インプレス NextPublishing (2024/04/02)

発売日 : 2024/04/02

言語 : 日本語

ファイルサイズ : 16465 KB

Text-to-Speech (テキスト読み上げ機能) : 有効

X-Ray : 有効にされていません

Word Wise : 有効にされていません

付箋メモ : Kindle Scribeで

本の長さ : 253ページ

Amazon 売れ筋ランキング: - 9位 Kindleストア (Kindleストアの売れ筋ランキングを見る)
- 3位 工学 (Kindleストア)

カスタマーレビュー: 5 ★★★★★ ▾ 8個の評価

図 16.5: Text-to-Speech に対応する Kindle の電子書籍であれば、読み上げ機能が使える

使い方のイメージとしては、Kindle を起動してから TalkBack を起動します。

すると、以下のように電子書籍が選択された状態になりますので、ダブルタップすると電子書籍が開かれます。(例として挙げている書籍は Forte さんの『入門！ 実践！ サーバーサイド Kotlin』^{*7}です。)

電子書籍が開かれた状態でダブルタップすると、TalkBack 機能でテキストを読み上げてくれます。

読み上げのスピードや声のトーンは TalkBack の設定から調整できます。

自分は中間よりも速めのスピードに設定することで、1 ページを 10 秒以内で読み上げられるようにしています。

^{*7} <https://www.amazon.co.jp/dp/B082H8GWRD>



図 16.6: Kindle で書籍を選択して読み上げる際のイメージ

読み上げ機能を知ってからは、積読していた本を少しずつ消化できるようになり、最近は通勤時によく活用しています。

目を使わずに新しい情報をインプットできる点も大きなメリットですが、以下の 3 つのメリットも感じています。

- 読み上げを聞くことで、漢字の読み方を調べる回数が減少する
- 目が滑って、読んでいる途中で「どこまで読んだっけ?」という現象が起こらない
- 本を持たずに済むので、手の疲れが軽減される

そのため、電子書籍 1 冊分を読破するまでの時間が以前より遥かに短くなりました。

以前は通勤で 1 日 1 時間半くらい読む時間があったとして、1 冊を読了するまでに通勤 4 回分が必要でしたが、読み上げ機能を使って聞き流すようになってからは、通勤 1.5 回分で 1 冊を読了できるようになりました。

ただし、読み上げ機能は絵やイラストの説明を含まないため、完全に耳だけでインプットできるわけではありませんが、それでもテキストのインプット効率は体感で2倍以上あると感じます。

通勤中や仕事の合間に目を使わずにできるインプット方法として、スマホの読み上げ機能を活用すると、感じていた疲れは軽減されつつ、テキストのインプット効率も向上します。

通勤やPCに向かっていないときに、ぜひ読み上げ機能を活用して「前から読みたいと思っていたあの本を聞き流す感覚でインプットする」という体験をしてみると、よりインプットを続けやすい生活ができるのではないかと思います。

16.2.2 目の負担が少ない画面にする

緑内障になってから目を使わずにインプットする方法を実践していますが、100%それに頼ることはできず、目で見て確認したり情報を得たりすることの頻度がまだまだ多いです。

耳を使ったインプットでテキスト情報を聞き流していますが、それでは絵やイラストでの説明が分からぬまままで、ある程度は視覚情報を目にしないといけない場面もあります。

そこで、目を使わざるを得ない状況であれば、できるだけ目の負担が減らせる方法でインプットしたほうが良いと思いました。

主に自分が使っている方法は2つあります。1つ目はPCやスマホで読み物を見る際、できるだけ画面をグレースケールにして見ることです。

グレースケールにする主なメリットは、色による刺激が軽減される点です。

カラー画面は色による刺激が強く、画面の輝度を下げても長時間見続けることで目の疲れが生じることがしばしばありました。

しかし、画面が白黒になると色のコントラストが低くなるため、目の負担はカラーでいろんな色を見るときよりも減ったと感じます。

文章を読むときやチャットを見るとには、グレースケールで見るのがちょうど良いと思いました。

2つ目はMacのNightShiftを使う方法です。

フロントエンドの仕事をしていると、どうしてもカラーで見る必要があるときがあります。

その際は、時間制限を設けて一定時間まではカラーで見て、それ以上は色は見えるものの、目に優しくするという使い方をしています。

目の疲れ具合でいうと、グレースケールにしたほうが効果はあると感じますが、NightShiftを使うことでブルーライトを制限する効果も感じられており、まったく使わないよりはマシだと感じました。

仕事をしているとモニターやスマホの画面を見る時間が長くなりがちで、どうしても目に負担がかかります。

目の負担を減らすためにいろいろと工夫をしている中で、長時間使ってすぐにできる方法で効果が感じられたのは上記の2つであり、今も仕事・プライベートにかかわらず使用しています。

自分のように画面と向き合う時間が長い方は、ぜひグレースケールとNightShiftを試してみてください。

16.2.3 いつかやろうと思ったことを今やる

目がだんだん見えなくなっていくと分かってから、目の使い方を大事にする以外にできることはだんだん少なくなっていくだろうと思いました。

特に目を使った体験については制限がかかるだろうと感じ、今のうちに良い景色を見に行ったり、誰かと話すに行ったりしたほうが後悔しないだろうと思いました。

自分の場合、緑内障だと診断されてから以下のようなことをしました。

- 前から行きたいと思っていた場所に行ってみる
 - 「THE RYOKAN TOKYO: 湯河原」に行き、「自分と向き合う専用ノートと一緒に泊まるパック」を利用する
 - 「おふろ cafe utatane」に行って、何もしない一日を楽しむ
- 会って話したいと思っていた人と話す
 - コロナ前に勉強会で会っていた方とオンラインで会い、話し合う
 - 韓国人のエンジニアの先輩や後輩と会って話し合う
 - 高円寺の飲み仲間と開拓していない店で飲み会をする

遠くへ行って得られる体験や、目が見えているからこそ楽しめることを経験したのはもちろんですが、それ以外にも前からやってみたいと思っていたことや、どうするのが良いのか分からなかつた抽象的なことが、次にやってみたいことや今後の方針として繋がるというありました。

- THE RYOKAN TOKYO: 湯河原
 - 大学時代に教育実習生として接していた学校の生徒さんと偶然会う
 - その生徒さんから「365 日日本一周絶景の旅」という本を知り、購入して次の旅行先として検討する
- おふろ cafe utatane
 - 「カメラはじめます！」という本に出会う
 - 読み進める中で、自分がやりたいのは写真をきれいに撮るよりも、撮った写真に補正をかけてきれいにしたいということが分かる
- コロナ前に勉強会で会っていた方とオンラインで会い、話し合う
 - SNS 上で交流があって近況を知ることはできたが、オンラインでの活動については新鮮な話題を聞く
 - 話の中で執筆する機会ができそうだったので、参加表明をする
- 韓国人のエンジニアの先輩や後輩と会って話し合う
 - 前から帰国してエンジニアの仕事をすることも考えたが、韓国のエンジニア事情について話を聞くことができた
 - 聞き終えてから、自分は日本語を使って現地の人と交流しながら仕事がしたい感覚が強いことが分かり、帰国して仕事をするのは当面考えないことにした
- 高円寺の飲み仲間と開拓していない店で飲み会をする
 - コロナ前はいつも集まって飲んでいたメンバーと久々に飲みに行き、何年も住んでいる

のに一度も入ったことがない店に行く

- 高円寺といえば酒のイメージがあったが、飲食店として美味しいところや全然知らない音楽の話を聞くことができる店が多く、まだまだ行ってみたい店が多いことが分かった

人との出会いには偶然な出来事もありましたが、ちょっとしたきっかけで行動を起こすことで、想定していたことに出会うことができました。

ただ、このような出来事には、以前とは違って何かしらの行動をするたびにその意味を考えたり、もう少し詳しく話を聞こうとする行動を意識的に行ってはいたからこそ起きたことだと思います。

緑内障が発覚し、はっきりと見える時間が限られてきたため、その間にできることと今に集中することに時間を有効に使おうと意識するようになりました。そのおかげで、出来事に意味を見出したり、前から思っていたことに具体性を持たせたりするようになったと感じます。

振り返ってみると、今よりも時間的な余裕があり、目の健康を気にしなかったときにはやりたいと思っていたことを後回しにして他のことを優先してしまい、あまり活力のない生活をしていました。

皮肉なことに、目の健康があまり良くない今、時間的な余裕もあまり感じられない中で、やりたいと思ったことをできるだけ行動に移し、活力を感じる生活ができるようになりました。

きっかけはあまり良いとは言えない出来事から始まりましたが、これからも行動に移して次のやりたいことや抽象的に思っていたことに具体性を持たせることにつなげられる活力のある生活ができるといいなと思います。

16.3 さいごに

海外で一人暮らしをしている中で突然目の病気が発覚し、最初はどうしたものかと思いました。

しかし、体を酷使していた自覚はあったので、思っていたよりも早かったものの、いよいよ来たのかと受け入れるようになりました。

今回、自分が伝えたいことをざっとまとめると以下のようになります。

- 健康診断だけでは目の病気にかかっているかどうかが分からないので、眼科に行くべき
- 突然の病気が訪れても「自分ではコントロールできないことが起きた」と考え、「これからどうして生きたいのか」について考えると、時間とともに次にやるべきことややりたいことが見つかる
- 耳を使ったインプットを実践すると、目で読むよりも早いと実感した
- 目に優しい画面設定をして、目を大事にしよう
- 新しい体験をしに行くと、次にやりたいことが見つかり、前からやってみたいと思っていたことがより具体的になる体験につながる

目の健康が悪化したことをきっかけに、目の使い方を見直したり、見えなくなる未来に向けてこれから何をするのかについて自分なりにまとめることができました。皆さんも目を大事にしつつ、目を使わないインプットを実践してみてください。

まわりまわって繋がった話

おやかた@oyakata2438

はじめに、この画像を見てください。



図 17.1: とりさんの 2018 年の Tweet

とりさん@flytorisan の 2018 年末のポストです。

このポストは、2018 年に発行した「見積もり本」の感想をいただいたものです。お買い上げありがとうございます。

さて、このあと、実はとりさんには、見積もり本の商業版への寄稿をお願いしました。見積もり本の商業版では、建築業界の見積もりについて、工数単価がきちんと決まっているとか、そういう話を寄稿いただきました。業界が違っても見積もりは重要なことなど寄稿いただき、もともと厚かった本ですが、さらに厚くなりました。もともとの見積もり本では、IT 業界の話がほとんどだったので、業界が異なったときにどうなるのか、そして建築業界のように後戻りが難しい業界でどうやって精度を高めるのかとても参考になりました。

そして、時は流れで 2025 年、ある商業誌^{*1}を発行することとなりました。その中にソフトウェア開発の歴史についての記載があります。その章は、ソフトウェア開発の生産性と品質を高めるために、建築業界の開発手法を参考にして、ウォーターフォール型開発およびその派生手法が誕生しました(1970 年とか 1980 年の頃)という記載があります。過去と全く同じものを作ることがない、手戻りが基本的になく前工程が過不足なく完成している前提であるという、ウォーターフォール型開発の原型ですね。そうですよね。ビルを建てるのに、図面ができないのに基礎を打ち始めないですよね。

久しぶりにとりさんのポストを見かけて、そこに繋がるか！ という衝撃を感じた次第です。

綺麗に伏線を回収しましたね！ という話なのですから、そういう感じで、予想だにしなかった伏線を数年越しで回収することもあるんですね。

過去に知ってへー、と思ったことが、数年後に別のシチュエーションで突然思い出される、知識との一部となっていることに気づく、といったこともあります。

それは必ずしも自分の興味の範疇外の内容かもしれません。それでもそういった知識の横のつながりがあると、何らかの問題解決につながることもありますし、その間をつなぐ糸がほかの知識を連れてきてくれることもあります。

この本、あるいは他の本が、そういった学びの元となったらうれしいですね！

^{*1} 親方 Project 編、みんなのアジャイル、技術評論社 (2025).2022 年夏に発行した「ぼくのアジャイル 100 本ノック」を底本に、再構成した、アジャイルの意味と価値をそれぞれの著者の体験に結び付ける形でアジャイルを読み解く本です。

何を学べばいいのかわからなかった自分の生存戦略

yumechi

社会人歴も長くなってきて、ようやく一人前のエンジニアになってきたような気もする yumechi です。私はソフトウェアエンジニアとして主にバックエンドやインフラ周りを担当するエンジニアとして働いています。

今回は私が社会人としてこれまでエンジニアを続けてきた中で、自分がどう考え学び、今日を生きるエンジニアとしての生存戦略について書いていきたいと思います。

18.1 自分のバックグラウンド

私は大学・大学院で情報について学び、基本情報の取得や授業・研究を通してプログラミングを学びました。競技プログラミングなども少しやっていましたが、プログラミングがそこまで得意であるとはあまり思いませんでした。

そんな中で授業や研究、競技プログラミングのコミュニティなどを見ていると、自分自身が突出して優れているわけではないことを感じていました。自分はプログラミングに限らず、何かを学ぶこと自体が好きなタイプだったのでどうしても専門的に何かを学ぶということがあまり得意でなかったように感じます。むしろ勉強する中で、何か似たものを見つけたり、何かを組み合わせたりする発見を楽しんでいたように思います。

そんな学生生活を過ごしていた私も、大学での専門性を生かしソフトウェアエンジニアとして働き始めました。最初は小さい SES の会社で働いていましたが、そこでソフトウェアエンジニアといっても様々な領域・働き方があることを知りました。自分自身は漠然とソフトウェアエンジニアとして、プログラムを書いて働くことを考えていましたが、同年代や先輩エンジニアのスキルを目の当たりにして、自分がどうすればいいのかわからなくなりました。また関東、特に東京という場所も、自分が生まれ育った地域とは違い、人々の考え方や働き方も異なり苦労した覚えがあります。

18.2 なりたいものがわからなかったので、とりあえず学ぶ

社会人の 1 年目が終わりに近づくころ、東京でのいろいろな経験を通し、少しづつですがモダンな技術とは何かを考え始め、技術的なトレンドに興味がわきました。自分の技術的な知識の浅さや狭さを感じていたこともあり、活躍しているエンジニア（先輩・同年代）を見て行動を真似してみ

ることにしました。その人たちとはカンファレンスや勉強会などに参加していたので、私も参加してみました。

当時、私は Python に興味が強く、PyCon JP^{*1} に参加して見たことを覚えています。自分が全く知らない分野の話などや機械学習の話を聞き、ソフトウェア技術の多様さを知り感動しました。またコミュニティベースで開催されている勉強会では、当時業務で触っている分野以外の勉強会にも参加してみて、自分の知識を広がりました。参加した勉強会についてはブログを書いたり、実際に小さく動くコードを書いてみたりして、自分の理解を深めるようにしました。

今思い返すと、社会人の早い時期に闇雲にいろんな技術を学んでおいて本当によかったと思っています。カンファレンス・勉強会の内容から技術的なトレンドを追ったり、技術との関わり方・選び方を学ぶことができました。30代になりメンバーからリーダーになり、自分自身のプライベートの変化や考え方の変化などもあり、今から多くの技術を追うのは難しいかもしれません。ですが、この時期に広く学んでおいたことが、いろんな分野のエンジニアと協力して働くことに役立つたり、自分のキャリアを考える上での選択肢を増やすことができたと思います。

勉強会などに参加した次のアクションとして趣味で勉強していた Python を使って実際に仕事してみたいと思い、転職もしました。自社で SaaS サービスを作っている会社に入ったのですが、学んできた Web の知識や Python の知識が足らず、入社直後は多くの方に助けていただきました。そこで Web を中心に学び直し、技術的な理解を深めるように努力しました。ISUCON^{*2}への参加も通じて、インフラに対しての理解も深めることができました。最終的には Python を使ったバックエンド開発だけでなく、インフラを含めてプロダクト全体を考え動いていけるようになりました。その後、何度か転職していますが、ソフトウェアエンジニアとしての礎はこの時期に Web 技術を学びなおしたことにより築かれたと感じています。

少し話は変わりますが、このころになると何度か登壇する機会もありました。発表するためには、自分の理解を深めることも必要になります。発表のために下調べをして、自分の理解を深めることができるので、自分の理解を深めるためにも登壇することは大変有意義だと感じています。登壇以外にもブログを書いたり、この本のように執筆することも同じような効果があり、アウトプットが大事なことと考えています。

18.3 広く学んできたものがようやく活きてきた

最近はリーダーとして、技術的な選択やアーキテクチャの設計や、プロジェクトの一部進行管理やメンバーの育成、チームビルディングを行っています。コードも書いていますが、ここまでやると時間の捻出が難しい状況です。そのため自分自身ではなかなか新しい技術を追いかけることが難しくなってきています。

しかし、これまで広く学んできたことが技術の早いキャッチアップや勘所をつかむことに役立っていると感じています。今年に入ってからインフラの選定にも関わることが多くなりました。

*1 日本で開催される Python のカンファレンス <https://pycon.jp/>

*2 Web サービスを高速化するチューニングバトルコンテスト <https://isucon.net/>

AWS は業務での利用もそうですが、趣味で触ってみたり資格試験を受けたりしたことで知識がありました。その知識を生かして、インフラの選定や設計に関わることができ、プロダクトの成長に貢献できることができていると感じています。技術的な質問についても勘所を抑え、AI に聞いたりドキュメントや OSS のコードをピンポイントに読むことで、早急に判断・解決することができるようになったと感じています。

またキャリアの中でスクラムマスターを専任する機会がありましたが、そのタイミングでスクラムを含めたアジャイルやマネジメントについて深く学びました。その結果、自分の思い込みや経験だけで進めるのではなく、本や勉強会などで学び直し、再現性のある方法論の典型パターンを学ぶことの大切さを知りました。この経験がリーダーとして、メンバーの育成やプロジェクトの進行管理にも活かせているように思います。

そして今の自分の仕事はこれまで自分が広く学んできた知識を生かした総決算である、と感じています。現在の私の仕事は期間やコストが限られている中、チームにとって最善の選択をしていかないといけない状況です。広く学んできた知識を使って、期間・コスト・技術のバランスを考え、チームと共に最善の選択をしていくことで、プロダクトの成長に貢献しています。いつか役に立つかなどうかもわからない知識もありましたが、いざというときに役立ったこともあります。いつどこでその機会が来るかはわからないので、広く学ぶことの大切さを感じています。もう少し時間ができたら、新しい技術もそうですが、今後マネジメントやビジネスについても考えてコミットしていく機会が増えてくると思うので、そちらについても学びを深めたいと考えています。

18.4 これから何を学んでいくものがわからない人へ

以上の経験から、私は広く学ぶことの大切さを感じています。必要とされたときに必要なものを学ぶことも大事ですが、広く学んでおくことで、いざというときの対応や新しい状況へのチャレンジにつなげることができます。私の生存戦略として広く学んでおくことで、様々な可能性を模索し、自分のキャリアを広げてきたことが今の自分につながっていると感じています。

ただ私のように広く学ぶことが得意な人もいれば、何を学べばいいのかわからない人もいるかもしれません。その場合は興味を持った技術や分野に積極的に触れてみることが大切だと考えています。最初は何が自分に合っているのかわからないかもしれません、実際に手を動かしてみることで、自分の興味や得意な分野が見えてくることがあります。

次に、コミュニティや勉強会に参加することをおすすめします。自分一人で学ぶよりも、他のエンジニアと交流・議論することで新しい視点や知識を得ることができます。ほかの人と比較していく中で、自分の弱点や強みが見えてくることもあり、自分の学びの方向性を見つけることができるかもしれません。

最後に、自分の学びをアウトプットできるとベターです。私のように業務の中で実際に使ってみることや、応用できるとさらに理解が深まります。ただすぐに実行できない場合はブログや SNS で感想を書くところからスタートするのもよいのではないかと思います。文章にしてみることで自分の理解を確かめることに繋がりますし、ほかの人の学びの助けやヒント、あるいはフィードバックのネタになるかもしれません。理解を深めより高度に応用していくことで、自分の学びをより深

めて様々なところで活かし貢献できると思います。

私自身もまだまだ理想とするようなエンジニア像からは遠いと感じていますが、これからも広く学び、新しい技術や分野に挑戦していきたいと思っています。そのためにも技術トレンドや勉強会にアンテナを張り、学んだことを生かして周りにとってより良い影響を与え貢献し続けていきたいです。

少しづつでも毎日続けてみる

yumechi

ここ最近は公私ともに自分の時間が取りにくくなってきた yumechi です。業務ではコードを書くエンジニアとしての仕事以外にも、チームをリードする立場としての仕事も増えてきています。またプライベートでは、東京以外の地域のコミュニティにも興味があり、そちらにも参加しているため時間が取りにくくなっています。

しかしソフトウェアエンジニアとしての仕事を続けていくためには、日々新しい技術や知識を取り入れていくことが大切だと考えています。時間がない中でも自分が実践していることを今回は紹介します。

19.1 移動時間を活用する

私は関東近辺に住んでいることもあり、電車での移動が多いです。ただしいわゆるカンファレンスや勉強会が開催されがちな場所からは少し離れているので、30 分～1 時間程度の移動時間がかかります。通勤に関しても同様です。

私はこの時間によく Kindle 書籍^{*1} で本を読んでいます。Kindle には多くの技術書、ビジネス書などがあり、自分の興味に合わせて読むことができます。私が移動中に心がけていることは次の点です。

- なるべく行きの電車で読む（帰りは疲れていることが多いので頭に入らないため）
- タブレットなどで読む（スマホだと画面が小さく目が疲れやすいため）
- 本を読むときは単語、キーワードレベルでメモを取る（Discord とかにメモを取るチャンネルを作っている）
 - 最近は ChatGPT^{*2} に質問をしておいて要約とメモを兼ねてもらっています
- 移動中はビジネス書や雑学を中心に読む（技術書は集中力が必要でかつコードレイアウトが読みづらく理解しづらいケースがあるため）

特にタブレットで読むのはおすすめです。ディスプレイサイズが大きく目が疲れにくいこと、またディスプレイサイズが大きいために一度に表示される文字数が多く、スクロールが少なくて済むためです。私は Lenovo 社の Lenovo Tab M8^{*3} を使っていますが、価格も 2 万円以内のため手頃

*1 Amazon 社が展開している電子書籍サービス <https://www.amazon.co.jp/kindle-dbs/storefront>

*2 OpenAI 社が提供している自然言語処理モデル及びチャットサービス <https://openai.com/gpt/>

*3 Lenovo 社が提供しているタブレット端末 <https://www.lenovo.com/jp/p/tablets/android->

でおすすめです。

ただし電車が混んでいるときもあります。その場合は Audible^{*4} で適当な本を流しています。Audible は月額プランに入ることで多くの本が聴き放題になるため、本を読むのが苦手な人にもおすすめです。

ちなみに私は帰りの電車では SNS やソーシャルゲームをやっています。大体帰るタイミングでは疲れていることが多いので、無理に学ぼうとするよりもリラックスして過ごすことが大切だと考えています。

19.2 注文の待ち時間を利用する

私はリモートワークで働いていますが、お昼休みは散歩も兼ねて外で食事をすることが多いです。その際に注文をしてから料理が出てくるまでの待ち時間もあるので、その間には英単語アプリなどをやっています。

Duolingo^{*5} や モノグサ^{*6}などのアプリを使って、英単語を学んでいます。個人的に英単語は見聞きする回数を増やしていくことで、自然と覚えられるようになると感じているため、このようなアプリを利用しています。意外と中学生レベルの単語を忘れていることが多いので、復習も兼ねています（英語学習に対して当時時間をかけていれば…という後悔もあります）。

またなぜ英単語アプリなのか？ というと、お昼ご飯を食べるときにはスマホと鍵くらいしか持ち歩かないで、荷物が少ない状態でもできるからです。また英単語アプリは短時間でできることが多いため、待ち時間にちょうどよいと感じています。他にもこの待ち時間で Qiita^{*7} や Zenn^{*8} を見る時間にもしております、トレンドから面白そうな記事を流し読みしています。

19.3 ブログを書く

最近はあまりかけていませんが、なるべくイベントがあったときにはブログを書くようにしています。

ちなみにブログを書くためのハードルを下げる方法に最近気づいたことがあります。以前までは学んだことを最大化しようとして長い文章を書こうとしていました。そうするといつまでもブログを書くことができず、結局出せないままになってしまふことが多かったです。最近は自分が覚えられるのはせいぜい 1 トークで 3 つくらいだと気づきました。覚えている 3 つのことをキーワードにしてブログを書くようにしています。この方法にすることでブログを書くためのハードルを下げ、続けやすくなりました（とはいえる時間の都合で書けていないことが多いのですが…）。

tablets/tablets-m-series/lenovo-tab-m8-gen-4-8-inch-mtk/len103l0013

*4 Amazon 社が提供しているオーディオブックサービス <https://www.audible.co.jp/>

*5 DuoLingo 社が提供している言語学習アプリ。私は英語のコースをやっています。 <https://www.duolingo.com/>

*6 Monoxer 社が提供している学習用アプリ。学生でなくても一部の英単語帳が無料なので活用しています。
<https://monoxer.com/>

*7 Qiita 株式会社が運営している技術系の情報共有サービス <https://qiita.com/>

*8 クラスマソッド社が運営している技術系の情報共有サービス <https://zenn.dev/>

そしてキーワードに絞ることでブログを書く時間も大幅に短縮できるため、続けやすくなると感じています。具体的にはオンライン開催の場合、だいたい帰りの電車で書ききれてしまうくらいの時間で書けるようになりました。書くのが大変という方は小さく自分が学んだことをアウトプットとして書いてみることをおすすめします。

また自分の学んだことを自分の言葉で文章にすることで、自分の理解度を深まり記憶にも定着しやすくなります。私の中で意識しているポイントではその段階でわからなかったことでも、わからなかったことを記録しておくことだと考えています。特に良い発表というのは学ぶべきポイントが多いので、その時点ではわからないことも多いです。しかし後から見返してみてわかるという経験が何度もありました。ですので、わからないことの記録も大事だと考えています。

19.4 まとめ

移動時間や待ち時間を活用して、少しづつでも毎日続けてみることが大切だと感じています。自分の時間が取りにくくなっている中で、自分の仕事やプライベートも大事にしつつ学び続けることが大切だと考えています。スキマ時間も学びに当てていて、少し窮屈な部分があるとは思いますが、それでも続けることが大切だと感じています。

学び続けるために大切なこととしては、行動の目的を持つことが大切だと感じています。私が英単語を勉強しているのは、海外旅行に挑戦したいという気持ちがあり、そのための準備として進めています。時間があるときはなるべくライティングやスピーキングも練習していますが、いつもその時間が取れるとは限りません。少しでも時間があるときに学び続けることで、少しでも理解できるよう努力しています。

他には習慣化していくことが大事だと考えています。私も環境の変化によってまさかこんなに時間が取れなくなるとはと思っており、いつまでもダラダラとしている時間がないことを実感しています。育児や介護などで時間を取ることができない人も多いと思いますが、少しでも時間を取れるように習慣化していくことが大事です。習慣化さえてしまえば、その時間を取ることが当たり前になるため、続けやすくなります。私も以前まではダラダラと SNS を見て、自分に関係がないことに一喜一憂などしていましたが、今はそういった時間を使わないようにし、自分と向き合って学びに当てるようにしています。

最後になりましたが、勉強や学びを続ける腰が重い皆さんに良い動画があります。そちらを紹介して終わりとさせていただきます。その動画とは YouTube の「桜井政博のゲーム作るには」チャンネルの、「とにかくやれ!! 【仕事の姿勢】」*9 です。私は朝起きてから活動するまでに時間がかかることがあります。毎朝この動画を見て勇気づけられてから活動しています。とにかく今日からやっていきましょう！

*9 桜井政博のゲーム作るには -とにかくやれ!! 【仕事の姿勢】
https://www.youtube.com/watch?v=JV3KOJ_Z4Vs

第 20 章

はんだ付けを学びました

@ponyoxa

20.1 はじめに

この夏、わたしは初めてはんだ付けを学びました。もしかしたら中学の技術の授業でやっていたかもしれません、まったく記憶ないのでノーカンにします。

はんだ付けを学ぼうと思ったきっかけは、自作キーボードと呼ばれる自分で組み立てる必要のあるキーボードを使いたくなったからです。肩の痛みに悩まされていたため、分割キーボードにしたかったのですが、市販の分割キーボードはあまり選択肢がなく、ある程度必要に迫られて学ぶことになりました。

とはいっても、はんだは必ずしも自分でしないといけないというわけではなく、料金を支払って依頼することもできます。ただし、料金は安くはないですし、今後も別のキーボードを使ってみることになったり、動かなくなったりしたときのメンテのことを考えると、自分でできるに越したことはないという判断になりました。

20.2 どうやって学んだか

20.2.1 社内で聞いてみる

まずは社内ではんだ付けの経験がある方に聞いてみて、必要そうな道具のイメージをしたり、初心者がいきなり始めるにあたっての難易度がどれくらいなのかをリサーチしました。

初心者でも少し練習すれば動くものは作れそうということがわかり、「何かわからないことがあつたら聞かせてください」とお願いしつつ、どうやって練習しようか、練習方法について調べてみました。その結果、遊舎工房さんではんだ付けの初心者向けにワークショップが開催されるということを知り、そちらに参加することにしました。

20.2.2 ワークショップに参加してみる

ワークショップは小さめのマクロパッドを実際に作りながら学ぶもので、キーソケットやダイオード、LED、各種スイッチをつけるという、キーボードを組み立てる際に必要なことがまるっと学べるものでした。

そこで諸々の注意事項も学びました。不要なのに、コンスルーの基板側もはんだ付けてしまっ

たのは失敗でしたが、ここで失敗したこと、自分で実践する際には変に失敗することなく完成できたと思います。

作っている間はスタッフさんがついていてくださるので、わからないことや疑問に思ったことはすぐに聞ける環境でした。自己流のやり方ではなく基本を教えていただけたことで、ひとりで組み立てる際にも迷いがなく作業できたため、最初に少し費用をかけてワークショップに行ったのは良い選択だったと思います。

20.3 実践！！

ワークショップで基本的な技術を身につけたらついに実践です。2万円以上する製品を買ったので絶対失敗したくありませんでしたが、なんとか一発で完成させることができました。

ワークショップのときよりも多くのはんだ付けをしないといけなかったため、根気はいりましたが、そのほかの部分で迷うことはほとんどありませんでした。はんだがとれてもまたはんだ付けすればいいや、くらいの軽い気持ちで行えました。

20.4 終わりに

はんだ付けを学んだ今回の教訓として得られたことは、「基礎基本は独学しないほうが安心」ということでした。「安心」というのは自分が実際に取り組む際に感じたことです。

自作キーボードはビルドガイドはあるものの、はんだ付けの細かい指示があるわけではありません。「これでいいのだろうか」と悩む事柄は少ないほうが良いです。みなさんもはんだ付けを身につけて自作キーボードしましょう。楽しいですよ！

第 21 章

キーボード布教の過程で学んだこと

@ponyoxa

21.1 はじめに

わたしはキーボードが好きです。何が特に好きかというと、スイッチを押している感覚が好きです。最近は自分ではんだ付けをして組み立てる自作キーボードにも手を出し始めました。そんなわたしが大好きなキーボードの魅力を周りの人にも知ってほしい！ そういう思いが少なからずあります。

この夏、「キーボードって何が違うの？」「適当に 2,3 千円のものを選ぼうと思っていた」、「ノート PC についているもので十分」と最初は言っていた同僚ふたりを素敵なキーボードの沼にひきずりこんだ経験から、「キーボード布教の過程から学んだこと」をお話します。

21.2 学んだこと

学んだことは主に 3 つありますので順にお話します。なお、キーボード布教は実際に秋葉原を歩いてお店をまわりながら行いました。計 2 回の訪問をし、1 回目でひとりがメカニカルキーボードを「これがいい！」と言ってくれて購入し、2 回目はもうひとりの同僚から「もう 1 回見に行きたいからついてきてほしい」と言われて再度秋葉原に訪問し、結果として通販でペアボーンなキーボードを購入するに至りました。

21.2.1 押しつけをしない

当然ながら自分の好みの押しつけをしないということは重要です。キーボードの押し心地はもちろん、見た目も大きな要素であり、どの要素をどのバランスで重視するかというのは各々の個性があります。

布教の際には相手の好みに共感しつつ、相手が何を重視しているのか、好みのタイプ感はどのようなものかといったことをつぶさに観察して分析することが肝要です。秋葉原でお店をまわっているあいだは、自分の経験から相手が好きそうなキーボードを探し出してひたすらおすすめする、レコメンドエンジンと化していました。何度かキーボードを触ってもらって反応を見ていくと、傾向がわかってきます。

おすすめをする際ももちろん、押しつけは厳禁です。レコメンドの精度がいまいちだったら修正するのみです。

21.2.2 実際に触ってもらうのがこうかばつぐん

今回の布教で良かったのは、実際に店舗が多くある秋葉原に出かけられたことだと思います。とりあえずたくさん触れる家電量販店に行ってたくさんキーボードに触ってもらうということができました。家電量販店はたくさんのキーボードが並んでいるため、デザイン面でも比較しやすく、「キーボードはたくさん種類がある」ということを実感してもらいやすいです。

キーボードにたくさん触ってもらった結果、押し心地に対しての解像度が上がりだんだんと好みがわかってくることでおすすめもしやすくなりました。百聞は一見にしかず、ならぬ百見は一触にしかず、でした。

21.2.3 体験に対してラベル付けする

また、できるだけ知識を押し込まないようにも注意していました。キーボード初心者はスイッチとかキークリップとか配列とかよくわかっていないことだらけです。そんなところに、軸がどうとか配列はどれが好みだと話すのは危険な行為です。興味深く聞いてもらえるなら万々歳ですが、キーボードを触る前に疲れてしまうかもしれません。キーボードを触る、スイッチを押すという行為を楽しんでもらえるように体力を残しておくのが肝心です。

キーボードに触りなれてくると、「あ、これが好きかも」という瞬間がやってきます。そうしたときに、「この押し心地は、青軸っていう種類なんですよ」といった具合に体験にラベル付けするイメージで知識を補うと、「わたしのが好きなのは青軸なのか！！！」という発見と感動につながります。ここまでくるともうにこにこです（わたしが）。

あくまでも背景に徹しつつ、発見に対してちょっとだけ言語化して、認識可能にするのがおすすめです。ここで、例えば青軸が認識できると、触り心地が異なるものに対して「青軸ではないこれはなんだろう」という認識ができるようになります。ここまで来たらもうキーボードの布教は達成したようなものです！

21.3 おわりに

こんな感じで同僚ふたりをキーボード沼にお誘いした結果、社内で「こいつはキーボードにこだわりがあるんだな」と認識されるようになりました。今は Keyball61 を仕事で使用していますが、興味を持ってくださる方もいて、今では社内にわたしも含めて 3 人の Keyball 利用者がいます（3 つともわたしがビルトしました）。

キーボードの話をするのは大好きです。が、布教の際には相手が主役。気持ち控えめに。

わたしの経験が、みなさんがハマっている沼に誰かを誘い込むヒントになれば嬉しいです。

第 22 章

技術同人誌の執筆を通して得られた学び

みずりゅ@MzRyuKa

2024 年 11 月に開催された技術書典 17 で、「継続に関する本」を頒布しました。

本記事では、この本の執筆を通して得られた「学び」について考察しましてみます。

22.1 学びについてまとめる前に

そもそも「学び」とはなんでしょうか。

Google の「AI による概要」によれば以下の通りとなります。

学びとは、好奇心や興味を持ち、自発的に探求する行為です。知識や技能を習得するための勉強とは異なり、新しい知識を生み出す「発見と創造」が本質です。学びの例としては、次のようなものがあります。

- 教わる通りに本を読んだり、物事を考えたりする
- 技芸を覚えたり、習得したりする
- 経験することによって知る
- まねをする

この例にしたがって、学びの内容を考察してみます。

22.2 継続に関する本の執筆に関する学び

「継続に関する本」は、自分が行なっているイラスト投稿活動を通して、継続に関する考え方をまとめた本です。

元々は、仕事や家庭の事情でなかなかできなかつた「アウトプット」に対して、何かしら短時間で続けられるものがないかと試行錯誤で生み出した行為でした。その過程の中で、これ自体を本の題材にしていけば良いのではないか（継続する材料と本の執筆ネタの両得になる）、という発想になりました。

執筆した本の冒頭にも書いてあるのですが、自分は本当に継続することが苦手でした。買っただけ満足してしまって、使わずに段ボールに積んである本や電子機器はたくさんあります。

そんな自分が書いた継続本にどれだけの価値があるのか。継続中といえどもまだ"1 年"には達していないのに、どれだけの説得力があるのか。そんなことを考えてしまい、数日は筆を進めることができませんでした。

しかし、ある日ふと考えました。いま、このタイミングでもイラスト投稿活動が続けられていること自体に、価値はあるのではないか、と。

デキる人がものすごくデキるようになった知見は素晴らしいものです。その手法は知りたいし、できれば真似もしたい。しかし、それは本当に自分にできるのだろうか、という不安がつきまといます。

「できた人」というのは、ある意味で上澄みの人達です。那人達と同じような手法は、自分にとってはレベルが高すぎて実施できない、もしくは続けられないかもしれません。

ですが、「考え方」ならどうでしょうか？

実施レベルが違うとはいえ、同じ「考え方」を得て行った「行動」です。10とはいかなくとも1や2は得られるかもしれません。

1や2を得て何が変わるのが、という考え方もあるとは思います。ですが、「何かは変わる」のです。だって、0が1や2に変わったんですから。それは、はじめの一歩かもしれないけど、確実に成長なのです。小さな一歩でも、その変化を前向きに受け入れ、できなかったことができるようになっていきましょう。

そしてそれが、次の一步を踏み出すための糧になっていきます。

前置きが長くなってしましましたが、答えのようなものが見えてきました。

前述の「学びの例」で言えば、「経験することによって知る」「まねをする」「物事を考えたりする」が本記事を通して伝えたかった「学び」に該当しそうです。

そして、この学びに気づけたこと自体が、次の学びを見つけるための一歩になるかもしれません。

22.3 まとめ

このページの記述自体も、実は結構苦労して無理やり捻り出して書いていました。

なぜそんなことをしたのか。それは、執筆活動をすることで「インプット」「アウトプット」「ふりかえり」のすべてが体験できるからです。

また、執筆には締め切りがあります。時間が限られているからこそ、短時間で試行錯誤して、その時点で何かしらのアウトプットを生み出す。さらに、過去に実施したことを思い出すことで、「その時にどんな気持ちだったのか」「どんなことを考えていたのか」もふりかえることができます。

これらを繰り返し、気づきを得て、思考や行動を変えていく、自分自身を更新していく。

それこそが、「学び」そのものなのかもしれませんね。

■コラム：コードレビューのループ本の執筆に関する学び

技術書典17では、「継続に関する本」以外にもう一冊、「コードレビューのループ本」を発布しました。

こちらの本は、自社のものすごくデキるエンジニアがSlack内で発言した内容、を用いて作

成したものです。

こちらの本についての「学び」は、「デキる人の発言は、何を切り抜いても名言になるんだな」というものでした。物の見方が上手／言語化が上手い、ということなのかもしれません。

デキる人たちの考え方を整理して世に出し、少しでも多くの人に届けていく。そうすることで、"普通の人"が"デキる人"にちょっとだけでも近づけようになる。そんな想いを込めて執筆した一冊でした。

ふりかえってみると、本記事で書いた内容と同じようなことをしていたのだと気が付きました。本当に、「学び」とは奥が深いものですね。

第Ⅳ部

コミュニティでの学び

コミュニティに所属することで得られる学びがあります。

コミュニティは、バックグラウンドの共通点のある人たちが集まる場所。最近ではオンラインのコミュニティも増えましたが、共通点を持ちつつ異なる点も多いので、様々な学びがあります。

学校や仕事とは異なる切り口で得られる学び、あるいは自主的な活動がもたらしてくれる何か。そういう体験およびコミュニティを支える技術的な側面、あるいは自己の成長などを見てみませんか？

技術書同人誌博覧会の次回案内デザイン 制作で画像生成 AI を活用する

23.1 はじめに

現在私は「技術書同人誌博覧会（以下、技書博）」のコアスタッフとしてデザイン協力に携わっています。最近では、個人的な取り組みの中で、デザインワークが必要な場面が増えた事もあり、「画像生成 AI を上手に活用する事でアイデアを効率的かつ素早く形にする事はできないか？」と思いつきました。

その様な中、丁度良きタイミングで技書博 11（2025 年 1 月 25 日開催）の次回案内を制作をする運びになったので、以前から気になっていた画像生成 AI 「Adobe Firefly^{*1}」をデザインの中で活用する事にしました。

本稿では、技書博 11 の次回案内デザインの過程において、どの様な形で画像生成 AI を活用したかを簡単ではありますがご紹介できればと考えております。

23.2 完成イメージと素材選定

デザインに関する打ち合わせの中で「横浜のみなどみらいの夜景と開いた書籍の中からジャンルを表すもの、または関連キーワードが溢れ出しているイメージのものにしたい！」というアイデアが出ました。背景デザインを考える際に「1. 夜景」と「2. 開いた書籍」という 2 つの全く異なる属性のものを上手く組み合せた形にする必要があったため、前段となる素材選択が重要になると考えました。

私がいつも利用している写真素材サービス等はありますが、お目当てのものが見つからなかった事や、思い切って表現の幅を広げてみたい意図もあったため、背景デザインの元となる画像素材については Adobe Firefly で生成したものを利用しました。場合によっては、生成した画像素材を更にデザインツールを等を利用して適宜加工等を必要はありますが、発表当初よりも精度やクオリティは格段に上がっているので有効活用可能な余地は十分にあると感じています。

^{*1} <https://www.adobe.com/jp/products/firefly.html>

23.3 Adobe Firefly を活用してお目当ての画像素材を得る

Adobe Firefly の使い方そのものは結構シンプルで、設定パネルで詳細情報を設定して、プロンプトにキーワードを入力して実行すると、最大4枚の画像が生成されます。

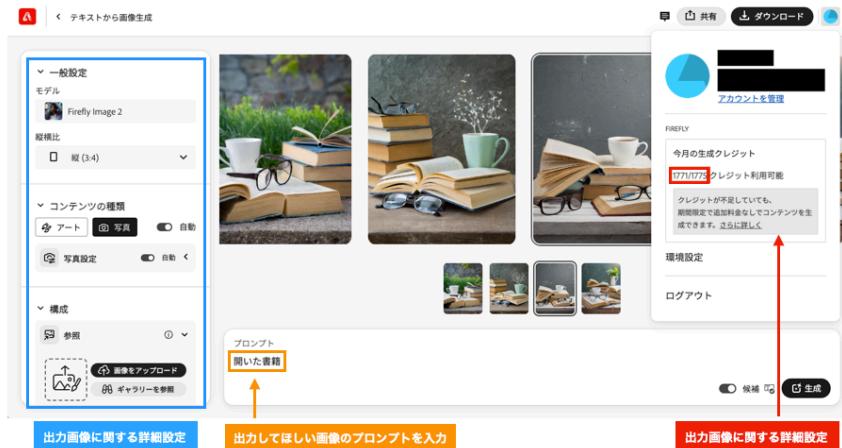


図 23.1: Adobe Firefly の操作画面

これを何度も繰り返しながら、今回のコンセプトに合いそうな画像素材を探すことになります。

画像生成時の詳細情報については、

- 利用する生成モデル・縦横比・アートか写真かの設定- 構成やスタイルをユーザーがアップロードした画像を参考にする- 画像生成時に効果を付与して加工した状態にする- カラーとトーン・ライト・カメラアングルを指定する

事が可能ですので、用途やイメージに合わせて柔軟に条件を設定できると思います。また、実行プロンプトと出力された画像素材のイメージを確認しやすいインターフェイスである点も、個人的にお気に入りポイントだと感じています。

※ Beta ラベルのない機能の場合は、生成された出力を商用プロジェクトで使用可能です。

※ 料金やクレジットに関する詳細は、生成 AI を含むプランを比較^{*2}の Web ページに詳細が掲載されています（ちなみに、私は「Adobe Creative Cloud」の年間サブスクリプションを契約しています）。

今回のデザインで利用した素材候補と実行したプロンプトはこちらになります。

23.3.1 書籍が開いた状態を表現した画像素材

書籍が開いた状態を表現して、なおかつ書籍から様々な光が溢れ出る素材を選択しました。

^{*2} <https://www.adobe.com/jp/products/firefly/plans.html>

- 実行プロンプト: ‘書籍に囲まれている’ / 生成実行回数: ‘5~6 回程度’



図 23.2: 写真素材候補（書籍に囲まれている）

23.3.2 横浜の夜景を表現した画像素材

当初は「横浜の夜景」で生成していましたが、ぱっと見て横浜が少しイメージしづらかったため「豪華客船」というキーワードを追加する事で、より連想できると思ったので、海上の豪華客船と夜景が組み合わせた素材を選択しました。

- 実行プロンプト: ‘横浜の豪華客船と夜景’ / 生成実行回数: ‘3~4 回程度’



図 23.3: 写真素材候補（横浜の豪華客船と夜景）

今回は AI で生成した画像素材ではありますが、なるべくリアルの写真に近くなる様な設定を生成時であります。実際は画像素材をそのまま利用するのではなく、デザインツール等を利用して加工する想定でしたが、この時点で実際の写真にかなり近いものを生成できた点は驚きました。

例えば、予め自分が想像している写真で「こんな感じのイメージが合いそうと感じたものを準備しておく」様にすると、画像生成用のプロンプトを考える際や、生成した画像素材を選定する際には役に立つと思います。（過去の制作物についても、#9 一般参加案内&技書博 mini 案内の背景では AI 生成画像を活用しています！）



図 23.4: これまで作成に携わった次回案内・参加募集のクリエイティブ

23.4 実際にデザインを作成するプロセスに関する解説

ここからはデザインを組み立てていくプロセスを簡単に解説していこうと思います。背景素材は Adobe Photoshop を利用して加工し、フライヤー制作は Adobe Illustrator を利用しています。

23.4.1 フライヤー背景素材で Photoshop で作成する過程

こちらは、前述した Adobe Firefly で生成した 2 つの画像を重ねて合成する事で実現しています。今回は幸運にも、拡大縮小・レベル補正やトーンカーブによる色味・アルファ値の調整などの基本的な調整のみで済ませる事ができました（素材制作の工数は選択した素材やデザインによって変化しますが、今回は想像以上にこの工程がすんなり進みました）。

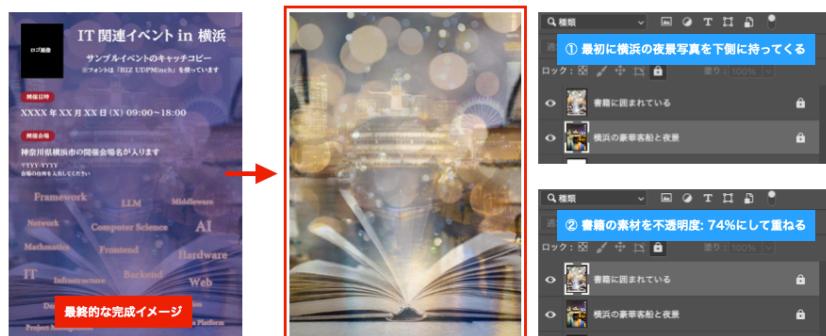


図 23.5: 背景素材を Photoshop で合成する

23.4.2 書籍が開いた状態を表現した画像素材

次に、制作した背景素材を利用して A4 サイズを想定したフライヤーに落とし込んでいきます。前述した背景素材をそのまま利用すると全体的にくすんでしまった感じになってしまったため、紫色と青色の中間くらいの色を重ねる事によって告知情報が目立ちやすくなる様にしました。また、技術系同人誌即売会であることを想定し、関連するキーワードが開いた書籍から湧き上がって来る様なデザインに仕上げました。

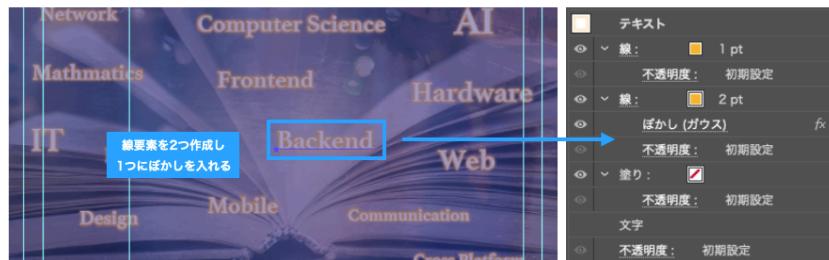


図 23.6: 書籍からキーワードが湧き上がる部分の表現

※ 関連するキーワードの表現については、文字に対して枠線を作り、その枠線に対してぼかしを加える事で表現しています（制作当初はネオン光彩の様なデザインを考えていましたが、大きな明朝体でないと映えない事や加工手順が少し面倒に感じたので、今回は利用しませんでした。※参考:光る文字を作る^{*3}）。

23.5 まとめ

今回は、想像以上に画像生成 AI の力を借りる事ができた実感を持っています。デザイン進行管理をしていた方も「いい表現が思いつかない」と悩んでいたので、背景イメージに合う表現を素早く見出す事ができた点が解決に繋がったのではないかと思います。

Adobe Firefly を利用した AI 生成画像の完成度も高く、プロンプト操作や各種設定をする操作体験も良かったので、今後も積極的に活用ができればと考えています。

画像生成 AI を上手くデザイン関連作業に活用する事で、これまで大変であった手順を簡略化・省力化する工夫に繋がり、それが更に速さを出す事にも繋がるのではないかと感じた次第です。

^{*3} <https://www.mdn.co.jp/reference/Illustrator/170>

第 24 章

己の学びのために地域技術コミュニティ 立ち上げた話

こうの@hk_it7

24.1 俺は別に東京に行きたいわけではない！！

学びを得る機会としてさまざまな勉強会があります。プログラミング言語系だと、JJUG CCC や Pycon JP、PHP カンファレンスなどです。ほかにも開発手法にフィーチャーしたイベントやクラウド技術、製品を中心としたイベントもありますね。

現在、その多くが東京で行われています。コロナ禍に多くの勉強会が中止、オンライン開催になりましたが、2024 年になり、多数が再度オフラインでの開催に戻しています。

そのような経緯のもと、オンラインでの開催を経験した結果、2 つのことを考えるようになりました。

1 つは、エンジニア同士で会話できずに、感想戦が難しいという点です。もう 1 つは東京まで行くの少し…… という点です^{*1}。

カンファレンスの参加目的は東京に行くことではありません。もちろん地方開催のカンファレンスの醍醐味として、観光や現地での食事であることも認めます。しかし、主目的はやはり技術的な知識であったり交流であったりだと考えています。

毎回場所を変えて実施しているカンファレンスとして RubyKaigi などがありますが、家の近くで行われる可能性はかなり低いでしょう。

そうなると、開催されることを待つよりも自分でコミュニティを作ってしまうのが早いですね！

24.2 立ち上げたコミュニティはどんなものか？

千葉県東葛地区（松戸市、野田市、柏市、流山市、我孫子市、鎌ヶ谷市あたり）を拠点とするエンジニアコミュニティです。

目的は情報交換や学習の相互支援、単純な人付き合い、つまりエンジニア同士の交流とっています。

エンジニアコミュニティとしては、知見の共有を中心とした学習を主目的とすることが多いと思

^{*1} あくまで筆者個人の感想であることに留意が必要です。

いますが、そのようにはしませんでした。

地方コミュニティという特性上、強いエンジニアが集まるという形式は想像しづらく、間口を広げたいと考えたからです。

また実施するコンテンツも、カンファレンスではないので、ライトニングトークや OST が中心となります。そうすると高度な内容を学ぶことは時間制約上も難しいです。

24.3 実践しているコミュニティ運営指針の紹介

大きな方針として、「可能な限り主催者は場の提供に留める」ということにしています。

個人でなんとなく始めたものなので、主催者（＝私）が「やらされている感」なく続けることを目指しています。

具体的には次のような方針です。

- ・企業スポンサーはつけない
- ・ノベルティや軽食を出さない
- ・運営側でテーマを絞りすぎない
- ・振り返り等はしない
- ・運営（主催者）は2人以上を維持する

誰かが責任を感じることが少ないように運営を進めています。

24.3.1 オフラインイベントにあたって会場と金銭面を公共施設利用で解決する

会場は公共施設を利用してことで金銭面のハードルを下げています。

現在主に使っている柏市の施設では、20人程度入る会議室を3時間借りても800円程度です。

プロジェクトを追加でレンタルしても、合計で1500円払えばお釣りがきます。

また、ノベルティや軽食を出さないので、「おもてなし」に費用はかかりません。そして、「用意したもののが余ってしまった……」みたいなことも起きません。

人が集まらなくて800円の出費であれば、そのまま1人で会議室をコワーキングスペースとして使って、もくもく作業しても納得できるかと思っています。

このようにオフラインイベントでハードルになりがちな会場と金銭面については公共施設を使って解決しています。

会場に公共施設を使う上のハードル

公共施設利用にはたいていの場合、市内在住や在勤の条件が付与されています。

満たさない場合、借りられないところや料金が倍になるところもあるので注意が必要です。

例えば柏市の施設を利用するためには次のいづれかが必要です。

- ・個人（市内在住または在勤）で利用登録を行う
- ・5人以上のメンバーを集め（半数以上が市内在住または在勤）、団体利用の登録を行う

個人利用の場合、予約が個人名義になりますので、参加者にどこの部屋かの案内をする際に本名でのやり取りが必要です。しかし、それさえ許容できればハードルは低いです。

団体利用の場合は市内のメンバーを集める必要があります。

私の場合はTwitter（現:X）で見つけた市内のエンジニアや、所属会社で同じ市に住んでいる人がいたので協力を依頼しました。

コミュニティがない地域であれば、声をかけると案外希望者はいるように思います。

また、近隣の技術コミュニティに参加することも良い案だと思います。CoderDojoなどは全国に拠点があり、学習やコミュニティに対して前向きな人が多いです。

地域コミュニティを立ち上げる際は少し調べてみると良いかもしれません。

24.4 地域コミュニティでやっていること

前述の通りコミュニティの主目的を交流としているため、会話を通して学習が促進されるように設計しています。

大まかな流れとしては自己紹介、アイスブレイク、トークセッション、締めという流れです。

アイスブレイクとして、特定の答えのない議題について話していくOST（オープン・スペース・テクノロジー）などを行っています。

最初の議題は地元のオススメスポット紹介など誰でも話せる非技術的な内容にして話しやすい空気を作るようにしています。

次にトークセッションです。基本的にLT会を実施しています。

しかし、一般的なLT会とは異なり、3枠に対して60分時間を取りています。

話すのは1枠10分までとしているので1枠あたり10分くらい時間が余るわけですが、感想戦タイムにしています。イメージとしてはここでもLT内容についてOSTをしているような感じです。

LTを聞いてもっと知りたいことや自分の考えていることを自由に発言して、発表者と参加者の相互で学びを深められればと考えています^{*2}。

また、地域コミュニティは日常の交流があってこそだと思っているので、Discordサーバーを作成し、そこでも互いの作業進捗や地域グルメ等について話し合っています。

大きなカンファレンスとの住み分けを意識して、交流という大きな目的のもとでコミュニティ内でうまく学習のサイクルを作れるようにコミュニティ設計をしています。

24.5 コミュニティの今後

私利私欲で作ったコミュニティですが、あくまで「地域のエンジニアコミュニティ」であり「私のコミュニティ」にすべきではないと考えています。

そのため、私がいなくとも勝手にコミュニティがまわり、少しずつスケールしていかなければ良いなと考えています。

^{*2} 初回に何も考えず時間を切って、後付けの理由だけど今のところ主旨にあってるなんて言えない。

コミュニティの理想は井戸端会議です。なんとなく集まってなんとなく情報が交換される、そこに強いオーナーシップはなく、ゆるい所属意識となくならないといいなって気持ちで駆動していく。そんなコミュニティになっていけばと思っています。

最後になりますが、2つ伝えたいことがあります。

1つはコミュニティ作るのはそんなに難しいことではないということです。これはここまで書いている内容を見ればなんとなく伝わるかなと思います。

そしてもう1つは、近所にコミュニティがあれば、ぜひ参加してほしいということです。自分と異なる考えを持っている人との交流は知識の幅や視野を広げる一助となります。

というわけで、これを読んでいて千葉県の東葛地区近辺にお住まいの人は、東葛.devで検索して参加してください！！（笑）

第 25 章

横断的な知識をまとめる本を作った。そこでの学び

おやかた@oyakata2438

2024 年末の冬コミ合わせで、この「ワンストップ学び」を書いていますが、それと並行して個人プロジェクトとして、「ワンストップ物理解析」という本を作っています。

SEM とか、FTIR とか、XRD とか。SEM は電子顕微鏡、XRD は X 線回折。物質の表面構造をナノレベルで観察したり、結晶構造（結晶構造を解析して、結晶の同定を行ったり）といった、さまざまな技術があります。それぞれの技術には、それぞれのエキスパートがいて、それらを活用してそれぞれの目的の分析を行っています。また、それぞれの教科書、論文がたくさん出ていますし、使う上でのノウハウもたくさんあるでしょう。

でも、案外、「こういう測定をしたいときに何を使えばいいのか？」といった問題に対して、サクッと答えてくれる本ってなかったように思います。そして、それは特に分野外の人にはハードルが高いもの。

だから、「やりたいこと」から始めてツールを見つけるような入門書があったら面白いのでは？と考えたのが本書のきっかけです。

例えば、1um 程度の凹凸がある表面の凹凸形状を観察したいときは何を使えばいいのか、それが 1 mm だったら？ 逆に 1nm、1 原子層だったら？ そういう観点で書いてみるのも面白いかも。

そして、それぞれの技術・装置について、ごく簡単な説明（具体的な分量としては 1 ページ）があって、原理と得意不得意がなんとなくわかる。みたいな本です。

さて、そんな本を書き始めたのですが、ChatGPT に大変お世話になりました。

いろいろな装置名を入れて、ChatGPT に、「SEM（走査型電子顕微鏡）について、600 字程度で説明をしてください。教科書を 1 冊つけてください」と聞くことで、概要、原理、特徴、用途などについてのかなり自然な文章を得ることができます。それを読むだけでも、特に用途や活用・注目されている分野についての学びがあります。関連技術との関係などもわかりますね。

1 ページにまとめるとなると、細かい話、数式や図などは落ちますが、それでも概要を知る程度には役に立ちます。X 線を打って、その反射（回折）の様子から、結晶の並びを見るんだな、くらいの概要をつかむ話です。

この本を書いての学びは、やはり、調べながら書くことで、「非常に広い分野（の表面）を割と網羅的に知ることができた」という学びがありました。実際に使っている、かかわっている技術ば

かりではないので、名前は聞いたことがあるレベルだったとして、それがどういうものなのかよくわからな買ったところに対し、研究機関の紹介ページや、論文、その他の資料で概要を知ることができました。

こういうきっかけに網羅的に界限の情報を仕入れられました！

第 26 章

思考の限界と言語化

ごまなつ@akrolayer

26.1 私たちの思考

私たちは様々なことを考えています。日々のスケジュール、食事は何にするか、何して遊ぼうかなどです。この時、私たちはあるものを使わないと思考できません。それは、言葉です。私たちは思考を言葉を用いて行っています。つまり、自分の知っている言葉の範囲内でしか思考を行うことができません。言い換えると、自分の語彙の範囲内でしか思考できないのです。知っている語彙を増やすことは、思考の範囲を広げることにつながります。自分が見えている世界は、自分の言語の範囲内でしかないのです。

26.2 エモいの是非

近年、エモいという言葉が使われることが増えてきました。これは感情が動かされるような状態を示すとされていますが、持つ意味が広すぎます。主に感動の意味で使われていますが、懐かしいのか、儚いのか、美しいのか、切ないのかどれかわかりません。エモいという言葉が出てきても使うべきではないとは言いませんが、少なくとも、区別はできるようになっておかないと思考の時に間違った方向に行く可能性があります。

26.3 怒りの感情を区別できない人たち

すぐキレる若者が話題になった時期がありました。原因はいろいろ言われていますが、私の考える原因是、すべての感情が思考の結果ムカつくになってしまふからです。どういうことかというと、勝負に負けた時悔しいと思いますが、悔しいという言葉を知りません。思考の結果、ムカつくになります。自分の思い通りになりません。歯がゆいです。歯がゆいという言葉を知りません。思考の結果、ムカつくになります。このように、感情の語彙を知らないことによって異なった感情を想起してしまい、その感情になってしまうのです。

26.4 いい言語化とは

最近は言語化が注目されています。ある主張を目にしたときに、自分が違和感を持っていたことだったのでよい言語化だ、ということがあります。これで、自分の考えはこの主張と同じことだ、

と考えて考えることをやめてしまつていいのでしょうか。本当に、自分が考えていることすべてがその言語化で表せているのでしょうか。

私が思う言語化の問題点は、書き手の主張と読み手の主張の重なった部分しか言語化で伝わっていないことです。いい言語化を見つけたときに、自分の主張の中で書き手の主張と重なっていない部分はそぎ落とされてしまいます。つまり、書き手の主張と読み手の主張の共通している部分だけが伝わっています。この時、それぞれの主張の共通していない部分は消えてしまうことになります。そのため、消えてしまった部分を見つめることを忘れず考え続けてほしいと思います。

26.5 スポーツの熟達と言語化

スポーツでは、コーチから技術を教えてもらうことが多いです。例として野球のバッティング技術をあげます。

プロ入り前は好成績を残していたのに、プロになった瞬間にバッティングフォームが変わり、不振になる選手がいます。これはなぜでしょうか。

原因の一つとして、自分で考えることを辞めてしまったことがあると考えられます。今までコーチからの指導の通りにやっていれば好成績が出た、これからもそうしようと思っていたのに、プロになってからコーチが変わり、自分の体に合っていない指導が入って不振になったと考えられます。プロ入り前はたまたま自分の体格や思考に合っている指導を受けられていたが、プロになってから自分に合っていない指導を受けてしまったのです。

ですが、これは誰も悪くありません。選手からすると合理的な判断ですし、コーチからすると、自分が現役時代に好成績を出すためにやったことを教えているわけです。これを解決するには、指導する側は選手に伝えたいことを伝えようと努力すべきですし、選手はコーチが伝えたいことを見つけて自分に合った指導として身に着けられるように努力すべきです。

かなりあいまいなことを言っていると思いますが、当然です。今は感覚の話をしているからです。指導とは、指導者が感覚を伝えて、指導を受ける側がその感覚を自分のものとして身に着けることだからです。

ある野球選手は指導の時に「ボールがガーッって来たら、それを打ち返すだけや」と指導したらしいですが、指導を受ける側はボールをよく見ろってことなのか、タイミングを見極めろと言っているのかなど色々考えられると思います。自分が普段考えていることがこの時に出てくるので、指導者が伝えようとしている感覚以上のことを身に着けられるチャンスでもあります。そのため、多くを学ぼうとする姿勢や多くの語彙、自分の体の動かし方の理解が必要となるでしょう。

26.6 権威がある人からの発言をそのまま受けることは非

世の中には権威がある人からの発言をそのまま信じる人がいます。認知バイアスと言われたりしますが、世の中の玉石混合の情報から正しい情報を得るためにある程度正しそうな情報だけ受け入れてほかの情報をシャットアウトするのは合理的な判断だといえます。しかし、それでよいのでしょうか。

権威がある人の発言も間違っていることがあります。人間なので間違えるのは当然のことです。また、その人が見えている世界が自分の世界や正しい世界と違ったのかもしれません。そのため、自分が見ている世界の感覚に変換したり、正しい世界と比べる必要があります。ですが、正しい世界なんて誰にも見えません。正しい世界を見つけるために、自分の世界を広げたり、見直すことも必要なではないでしょうか。

26.7 感想

価値観の違いやノリの違いといったものは突き詰めるとそれぞれの思考によるものです。思考は言葉で行われているため、言葉の使い方や知識によって見えている世界が変わります。見えている世界を誰かと完全に同一にすることはできませんが、それでも誰かとの世界の共通点から学べることもあるのではないでしょうか。

LT の実況で学びを最大化する

おやかた@oyakata2438

LT 会に参加したら、ぜひ実況をやってみましょう。楽しいだけじゃなく、勉強会・LT 会での学びを最大化できます。おっと、LT 会のハッシュタグは忘れないように。

ここでいう実況とは、話の要点をまとめて、Twitter(X) に流すこと。自分のためにもなりますし、イベントの盛り上がりの一助にもなります。

27.1 学びの最大化

実況すると、自分にとってうれしい点があります。

27.1.1 LT 会のメモになる

自分のタイムラインに話の要点がメモされます。紙に書くのもよいですが、ぜひタイムラインに流しましょう。

紙に書いたメモはどっか行っちゃうこともあります。TL ならまあ基本的には残りますね。この時点ですでに、単に聞くだけよりもメモが残るという意味で大きなメリットが得られます。

27.1.2 インプット・アウトプットの練習になる

インプットしながらアウトプットすることは、定着に非常に効果的とされています。猛烈な勢いで流れてくる内容を処理・整理して、アウトプットする内容を考え、整理することは非常に高負荷な頭脳労働ですが、その分定着した内容はこれから役に立つでしょう。

また、この技術は、会議での議事録の整理といった、仕事でのスキルの一つとしても重宝されます。本番の会議で議事録取りをミスったりするとヤバいこともありますが、何の義務でもない LT 会であればその練習に持ってこいです。

上手くいかなくてもそれはそれで、うまくいかなかった体験となります。次の登壇者、あるいは次の勉強会はきっともうまくいくはず。あとで自分のタイムラインを見て、どの点がイマイチだったのか見直してみるのもいいですね。

27.1.3 あとからブログとかにまとめやすい

自分のタイムラインをあとからざっと眺めて、いくつかの Tweet を貼ったり、補間したり、感想をつけたり、スライドへのリンクを貼ったりするだけで立派なイベント参加レポブログが作れま
すね。

一回自分でかみ碎いた内容なので、スムーズに整理してブログ化できます。

イチから思い出してブログを書くとなるとかなり大変な作業ですが、下準備、マテリアルはほぼそろっていますね。

27.2 その他のメリット

27.2.1 登壇者・コミュニティへの入り口

実況することで、登壇者からフォローされたり、同じイベントに参加した人からフォローされたり、フォローしたりしやすくなります。

勉強会の参加者と重なるコミュニティがある場合がありますので、そこへの入り口として大きな役割を果たすでしょう。

27.2.2 実況してくれると登壇者がうれしい

登壇者は、ほぼ確実にエゴサします。

参加者が流してくれた登壇内容まとめは、いいね！ を付けて回ってます。

登壇者にとって、あの懇親会での会話、質問も重要ですが、TL の反応もきわめて重要なフィードバックです。伝えたいことが伝わったのか、あるあるとして共感してくれたのか、誤解を生む表現・内容ではなかったか、あの小ネタは滑ったりしてなかったか、など、あとから見返すフィードバックとして重要です。

好意的な Tweet はうれしいですし、少々ネガティブだったとしても、次への改善になります。一番悲しいのは、何も手ごたえがない時・・・

27.2.3 実況するとイベント的にうれしい

イベント的にもうれしいことがあります。イベント主催者にとって、参加者のフィードバックは何より大切なものです。

ハッシュタグ付きで Tweet がたくさん流れると、イベントの盛り上がりの大きな助けになります。

実況があると Togetter などの便利ツールでまとめを作りやすいですし、割とみんな TL 見ながら LT を聞いています。その結果、あとから資料を確認する、配信動画をチェックするなどの動線の元になったりと、さまざまな活用がなされます。トレンドに乗ったりするのは例外的な大きなイベントだとしても、実況があることで、ここまで述べてきたようなさまざまな効能が得られます。

27.3 まとめ

LT 実況で波及する効果を思いつくままにあげてみました。いずれも自身の学びを最大化しつつ、波及する効果があります。

LT 実況は今すぐにでもできる LT 会の楽しみ方。来週の勉強会で試してみませんか？

第 28 章

イラスト制作・デザインにおける個人作品と依頼品の違い

トカゲ@Hiro_Sauria

28.1 違いの早見表

表 28.1: 違いの早見表

項目	依頼品	個人作品
作品完成の決定者	依頼主	制作者
重要事項	・依頼への理解 ・対応スピード	情熱
手段（使用ソフト）	CLIP STUDIO・Illustrator	Procreate・Illustrator
制作時意識すること	・コンスタントな進捗共有 ・ペース配分	作りたいものの具体化

28.2 違いの大前提

最大の違いは、完成の決定者が違うところ。

依頼品は「依頼主（他者）」が完成イメージを持ち、個人作品は「自分」が持つ。

依頼品の場合、「完成」の判断は依頼主（他者）が持つため、自分の完成の判断と異なることがある。むしろ完全に一致することはあり得ない。ただし、その差を埋めることはできる。

当たり前のことだが、依頼品作業時は常に意識するようにしている。そして、それを具体的なポイントとして次の 2 点に集約し、行動するようにしている。

依頼品は特に作業ペースのコントロールを心掛ける

依頼品は時間をかけなければかけるほど相手の時間も食ってしまうため、作業ペースは非常に重要。

特に、デザインの大枠が決定するまではコンスタントに発信することで、すり合わせしやすくなる。

大枠決定までにいかにすり合わせができているかが成否を決めると考えている。

作品への満足度を上げるため、作品方向性を依頼主の軸に合わせる

どんな作品であれ、最初はイメージが不透明であったり、途中で方向性が変わったりする場合が多い。個人作品は自己の中で完結するが、依頼品はそうはいかない。

そこで、依頼者の好みや、依頼の意図を把握することで、制作時に重要となる作品の軸を、依頼者の持つイメージに近づける。

28.3 制作工程の違い

28.3.1 依頼品の場合

依頼品・・・本の表紙を想定

[スケジュール例 ※納期 1/20、製作期間約 1.5 カ月と仮定]

- 11/30 依頼受領
- ~12/7 ラフ案 2 点提示
- ~12/21 イメージすり合わせと最終決定
- 1/5 完成案提示
- ~1/10 調整依頼受付
- 1/15 納品

工程

工程を少し詳細に書きつつ、進捗の数値と併記してみる。

1. 依頼を受ける（進捗 1 %）
2. 詳しい依頼内容、納期、予算のヒアリング（進捗 2 %）
3. スケジュール感・報酬額を決定し依頼受領（進捗 3 %）
4. 依頼内容の精査しラフのアイディアを作る（進捗 5 %）
5. ラフを提出（進捗 10 %）
6. 1~2 個に絞ったラフで最終イメージがわかるものを提出（進捗 25 %）
7. 完成イメージ決定・色味共有（進捗 30~40 %）
8. 制作の実作業。進捗管理のため、1~1.5 週の間隔で進捗共有（進捗 50~80 %）
9. 最終案提出（進捗 90 %）
10. 最終調整して納品（進捗 100 %）

作例とイメージのすり合わせの過程を、実際のラフで示す。

なお、筆者の場合、最終納期が 2 ヶ月以上先なら 2~3 個のラフを提出している。納期によって調整し、2 カ月以上先なら 2-3 件、1 カ月強であれば 2 個、1 カ月弱であれば 1 個を目安としている。



図 28.1: LT 秘伝の書の表紙（決定ラフ）として最初に提出。全体における進捗 25 %



図 28.2: 表紙最終イメージ。進捗としては 30~40 %。

完成版は、この本の隣に並んでいたはずなので、思い出してほしい。

決定ラフから最終イメージ決定までは微修正が多いので案外時間がかかる。キャラクターものの

場合、顔の向きやポーズ、配置や小物などの調整が必要となるので、使いやすいツールを使うことも重要である。

この時点で依頼者と密に会話しておくことで手戻りの可能性を減らすことができる。また、会話の中では、依頼者の意図をあいまいにせず、あるいは言語化の手助けをすることも必要と考えている。「xx な感じ」の「xx」の部分をいかに言語化・具体化するかがここでの要点である。

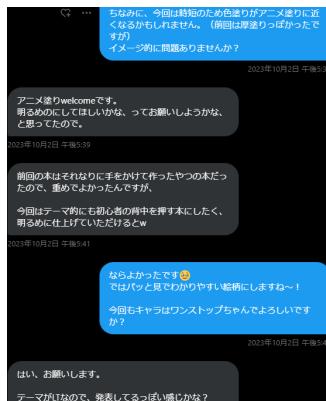


図 28.3: 相談例: 厚塗りかアニメ塗りかの相談・イメージ共有

また、最終調整・納品とあわせて、相手先への事務作業（請求書提出等）も忘れずに行う必要がある。また、2024年11月以降、いわゆる『フリーランス新法』に従う必要がある点にも留意。

28.3.2 個人作品の場合

依頼ものに対し、個人作品の場合の例を示す。

個人作品のイラスト本を想定

スケジュール例 [締切 1/20、製作期間 3ヶ月 (イラスト 6枚程度) と仮定]

- 10/20 なんとなく描きたいイラストが思い浮かぶ
- ~11/15 コンセプトを具体化、初出時期と期限を決定
- ~12/31 イラスト制作
- 1/10 本文の構成、表紙・裏表紙作成
- 1/20 脱稿

工程

1. コンセプト出し
2. 具体的コンセプトの決定
3. イメージアートを一つ作成し色味やデザインの具体化

4. 必要であればキャラクターや小物等のデザイン
5. イラストを3~5点ほど思いつくままにラフ作成、または、先陣を切る一枚目を作成
6. 気に入ったものから仕上げる
7. 納期に合わせてページ数と収録構成や表紙を考える
8. 完成

1.2. の中に、具体的になるまで資料や他者の作品をあさる期間がある。

なお、個人制作における1、2の段階は、先の依頼品の工程の4までに相当し、ラフのアイディアを作るまでのところに相当する。

また、この時点で、どの時期に初出の本にするか、どんな宣伝をするかもなんとなく考える。

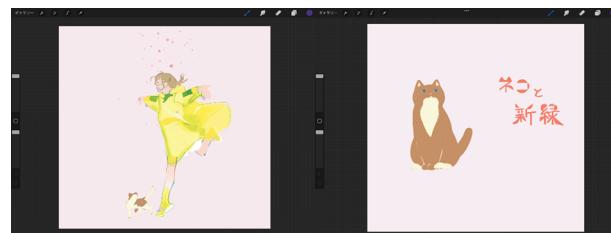


図 28.4: 個人作成時のラフ例

5~6の段階で、複数ラフ出しを行うか、一枚ずつ仕上げるかはコンセプトによって変わる。

ただし、個人的に、工程(5)を追加のコンセプトや、方向性変更の最終局面と定めている。そうしないと、コンセプトや方向性がぶれてしまったり、イラスト集として完成しなかったりというリスクがあるからである。



図 28.5: 表紙例 (猫と新緑)

28.4 制作手段の違い

依頼品と個人作品で主に使用するソフトも変わる。

次の表に依頼品と個人作品で主に使用するソフトを示す。

表 28.2: 主に使用するソフト

	主に使用する	サブソフト
依頼品	ClipStudio Illustrator	Procreate Photoshop
個人作品	Procreate Illustrator	ClipStudio Photoshop/SAI

ソフトに対する考え方:

- 依頼作品で気にするのは、「あとからのリカバリが可能か、細かな調整が可能か」である。そのため変更があっても後戻りしやすいクリスタをイラスト描写ソフトとして使い、文字入れと最終デザインをイラレで行っている。
- 個人製作は紙に絵の具を使って描いているようなシンプルなソフトを使い、イメージを筆に乗せやすくしている。
- 個人製作は完成イメージが全て自分に寄与するため、今この瞬間のイメージを反映できるのが依頼作品と違うところ。大幅な変更がすぐに具現化できるプロクリはその強みを持っていると考える。
- 私の制作におけるデザインはイラレが一強だ。文字入れから画像の編集まで、広範囲を一つのソフトでカバーでき、3Dにも対応可能なのが大きな要因だ。他のソフトでも文字を入れられるが、細かい調整が必要な「デザイン」という分野においてとにかく強いと感じる。(もちろんこれは、最終形態が平面である作品においての話である。Webページ、動画、立体においてはその限りではない)

28.5 制作をスムーズにするための工夫

依頼品「丁寧よりもスピード」

- 依頼作品の工程(7)の「完成イメージ決定・色味共有」までは、相手に分かるレベルであれば完成度を気にせずイメージを共有している。(図 20.1、20.2 参照)
- 具体的スケジュールを依頼主に共有し、進捗管理を曖昧にしない(例: この日までには次の進捗を公開、と事前に共有)
- 上記スケジュール共有は、双方が振り返れるよう、口頭ではなく文章で記録しておく。

個人作品「“作りたい”の具体化と統一」

- 個人製作こそコンセプト決定は最重要課題。
- 筆者の場合、イラスト本制作が主であるため、収録されるイラストの統一感は作品の質向上にかかせない。イラストの統一感・デザインの納得感を出すために、最初のコンセプト作り

が大きく影響してくる。

- コンセプトはより具体的に。特に初期コンセプトは、パステルで柔らかい雰囲気、とか、動きのある構図、のようなふわっとしたイメージではなく、「固有名詞」を使う。

[具体例] ※個人制作の同人誌「猫と新緑」より

- 桜の若葉
- 木漏れ日
- 黄色のレンコート
- 雨上がりの春日(ピンクにけぶる空気感)
- 柔らかな日差し
- 水たまり
- 跳躍

28.6 まとめ

ここまで筆者の所感を語ってきた。場合分けはしたが、伝わりづらいところ、基本的過ぎて参考にならない部分もあっただろう。それでも、この章が皆様の制作の一助となることを願っている。

「ワンストップ学び」表紙作成秘話

この項は、前章の「イラスト制作・デザインにおける個人作品と依頼品の違い」にて紹介した依頼品制作フローに従って、表紙が出来上がるまでを解説していく。先に該当記事に目を通していただけるとイメージしやすいかもしれない。

今表紙に使用した主なソフト：CLIP STUDIO, Illustrator

29.1 依頼内容

「ワンストップ学び」の本の表紙イラストとデザイン。

本の内容が『習得したいことはあるのに、思い通りに行ってない人に向けて、著者の皆さんノウハウを届けてみませんか？』^{*1}という、広く“学び”をコンセプトにしている。

また、下記をイラストのキーアイテムとしてご提案いただいた。

- 机に座って勉強しているワンストップちゃん
- 学びでイメージする何か
- ミニキャラ、その他アイテム

29.2 ラフ提出

納期も 2 ヶ月以上あり、かつ私が暇だったため、ラフを 4 種類（うち 2 種類に別パターンあり）を提出。

^{*1} 【新企画】ワンストップ学び. 2023/9/16. (<https://note.com/oyakata2438/n/n241fe6992846>) より引用
(2024/11/28 参照)

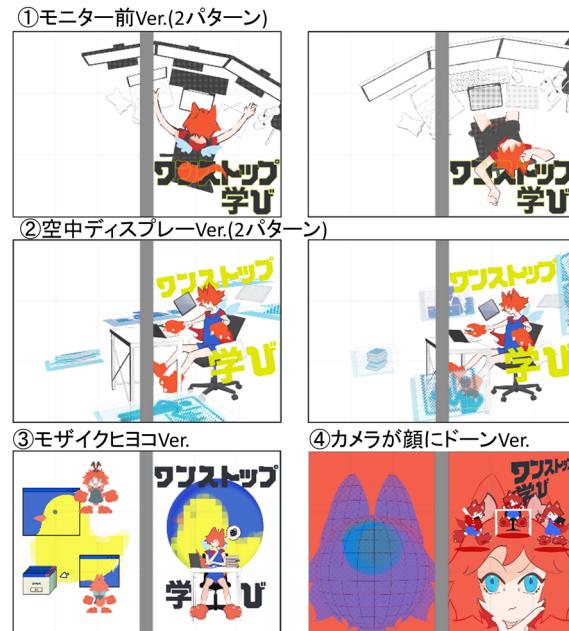


図 29.1: ワンストップ学びの初期ラフ 4 パターン (6 種)

秘話なので正直にいうと、顔アップ版が気に入っていた。ワンストップちゃんの顔がかわいくて(笑)。

本がオムニバス形式で多様なジャンルを包括することもあり、説明的にならないキャッチャーな表紙にしたいが、机に座っているだけでは動きが出にくいため、文字を斜めにしたり、小物で斜線を作ったり、思いつくままに書き出した4種類だ。

ラフ1・2はキャッチャーに加えシャープな印象が出るように、3・4では表紙で多くを語らず、本文が気になるようにと考えながらデザインした。

29.3 修正・ブラッシュアップ

ラフをもとに、依頼主の方で選んでもらった。今回は複数人で書く合同誌なので著者が何人かいたため、アンケート形式で選んだそうだ。その結果、2・3が候補として残り、さらにいくつかのアイディアが出たとのことなのでそれを反映する。

主なコメントは以下の通り。

- 2、3がともに同じ程度に好評だったのでこの二つをブラッシュアップする。
- 2も3も、もう少しワンストップちゃんを大きく（椅子とか机とかは枠からはみ出させる）
- 2は、画面は垂直な方がいい。
- 3のミニキャラはGood。ヒヨコの解像度が上がるモチーフもGood。
- ミニキャラは動きが欲しい。

これらをもとに修正したパターンが図のイメージ。



図 29.2: ラフ 2 稿

結果として、ラフ 2（空中ディスプレー Ver2）が採用され、仕上げと進んでいくことになる。タイトル・クレジットの位置がまとまっていた方がわかりやすいし、キャラクターも目立ちやすくなると考え、タイトルを下に配置した。

また、裏表紙も作成。タイトルロゴとミニキャラがまさに裏表紙ですよ～って感じだ。裏表紙は一発 OK をもらい嬉しかった思い出。

29.4 キャラクターデザイン

今回のワンストップちゃんは以下の思想・要素でデザインした。当社比、幼めな風貌である。

- シンプルで小物を邪魔しない
- A ラインオーバーオールの丈短めパンツ

今回のワンストップちゃんは専用グローブ・シューズありで手足がボリューミーなことも、少し幼い印象のキャラデザに一役買っている。

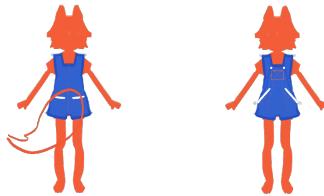


図 29.3: ワンストップちゃんキャラデザ

そもそも、キャラクターデザインは依頼されているわけではなく、自分の趣味嗜好で行っている。私の趣味が色濃く反映されているので、注目してもらえると嬉しい。

29.5 CLIP STUDIO の 3D 機能を使用した画面構成

今回の表紙では、お絵かきソフト「CLIP STUDIO」(以下、クリスタ) の 3D 機能を多用した。必要な 3D データは、「CLIP STUDIO ASSETS」というクリスタと連携した素材提供サイトから入手している

3D モデルは、配置したものを 360° 回転・リサイズできることから、キャラクターや物の位置関係把握と、画角調整による構図のプラッシュアップができることが最大の利点だ。

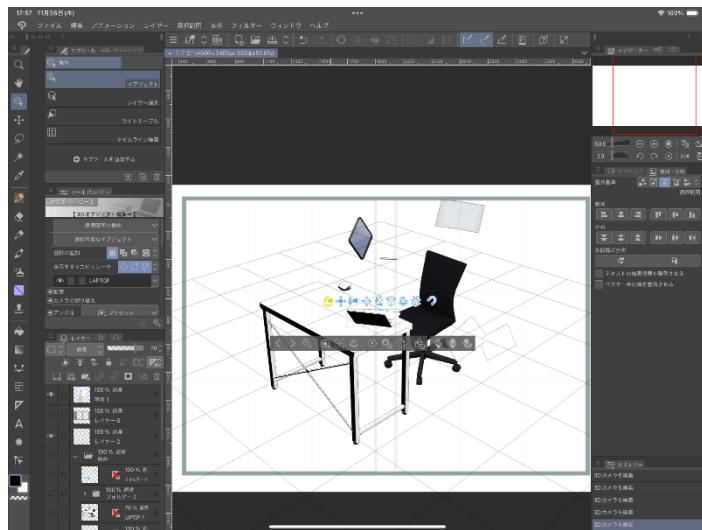


図 29.4: クリスタの 3D モデルを併用したデザイン

また今回は、電子機器にたいしての知識不足が否めず、フリーハンドで破綻無く描くことは難しいと考え、3D モデルを使用した。

3D モデルを利用するデメリットは、イラストへの個性の消失、ダイナミックな表現の小規模化など少なからずあるが、ここは制作者の腕の見せ所だと解釈している。

図のようにキャラクターと小物のみ配置した状態にすると、今回の表紙の立役者は 3D モデルであることがよくわかる。

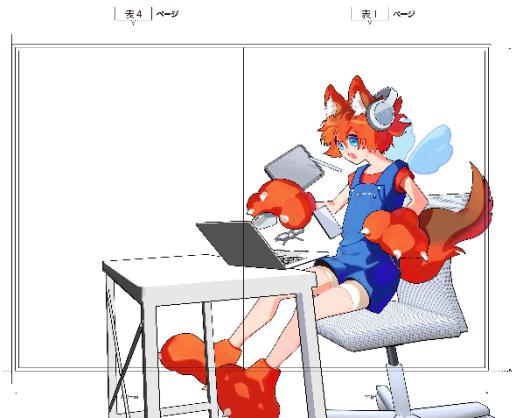


図 29.5: キャラクターと小物のみ配置した例

29.6 完成

こうして出来上がったのが今回の表紙！

最後に裏表紙のミニキャラが一体増えたり、タイトルロゴが少しリッチになったり、背表紙ができたりした。

第 30 章

執筆を促進するために Slack 通知をつくってみた

@mottox2

本書「ワンストップ学び」では「学び」という抽象的なテーマなため、執筆状況が芳しくない状況が続いていました。コミュニケーションツールとして利用している Slack でも雑談は進む一方で、執筆はあまり活発でない時期がありました。

自分が執筆する以外に、サークル内の執筆状況を改善したいと考えたときに次のような課題感がありました。

- 執筆状況は別のツールで管理しており、そもそも執筆状況が順調かどうかかもわからない
- 執筆が活発な雰囲気が感じられず、新規著者の参加ハードルが高いように感じられる

また、「自分の学びネタがほしい」ということがあり、これを解決するために独自の連携を作りました。最終的には、GitHub（原稿を管理しているサービス）にはみんなの原稿が集まっているので、1 日に 1 回プログラムで原稿の数を数えて、原稿が増えていた場合、Slack に次のような通知を飛ばすようにしました。



図 30.1: 最終的に出来上がった通知

この試行が成功かどうかは、実際にできたこの本が厚いと思うのであれば成功ですし、薄かったら失敗だと思います。しかし、失敗したとしても次回の執筆に活かせると思うのでそれもまた学びになるでしょう。

これは自分の学びでもありますが、誰かの学びになるかもしれない作るときに意識したポイントについて考えてみました。

30.1 状況を把握するためのアプローチ

弊サークルでは執筆は Git（主にソフトウェア業界でコードを管理するために使われているツール）を利用しておらず、GitHub（Git を使ったプロジェクト管理を行うサービス）というサービスに原稿をおいて管理しています。単純に通知を行うだけであれば、GitHub が標準で持っている Slack 連携を使うという手もあったのですが、Git の通知をそのまま投稿しても、執筆に最適化されたフォーマットではないので、単にうるさいだけの通知になるように思えます。また、普段から git に慣れた人が見るならともかく、いろんな業界の人が参加している親方 Project ではもっと親切な通知のほうがいいと考えました。

30.2 ワイワイ感を出すためのアプローチ

ここ最近、本を作るワイワイ感が足りないし、メンバーも多くなってきて発言する人が偏ってきた感覚があります。執筆が活発であれば、まだ執筆していないメンバーも書き始める動機になるかもしれません？ と考えました。最初に作った通知は次のような通知で、伝えたい情報は伝えていますがサークル活動のワイワイ感は全くないものでした。



図 30.2: 最初に作った通知

そこで、章の数だけ絵文字が増えて積み上げている様子がわかるフォーマットに変更しました。かなり主観的ではありますが、こちらの方が盛り上がっている感じがでていると思います。



図 30.3: 最終的に出来上がった通知

また、味気なかった標準のアイコンと名前から我々のサークルでおなじみのキャラクターとフレーズに変更したことで親しみやすさを出してみました。

30.3 得られた学び

通知機能を導入したことでの、原稿の進捗に関するメンバー間のコミュニケーションが活発になりました。「前日より増えてる！」といったコメントが増え、締め切り間際という事情もありましたが、いつもの本より進捗への関心が高くなった印象があります。また、通知に関するフィードバックを募集した際には、「いい圧を感じる」「通知の一つになりたい」といったポジティブな意見がありました。このことから、通知は有効な手段だったと感じています。ただ、通知される期間が短かったため目新しさがあったからかもしれません、長期的に運用された際にマンネリ化するかもしれません。

今回の取り組みでは、GitHub Actions を活用した通知機能の実装を行いました。バグはあったのですが、締め切り間近だったのでバグがありながらも通知を優先させて動かしました。実は ChatGPT に生成してもらったコードをほぼそのまま使っており、細かい検証を怠らないことの重要性を痛感しました。これもまた学びです。

今後はこの取り組みを次の本でもやっていこうと思うので、バグの修正やさらなる参加のしやすさを目指した改善を行っていきます。

場がオープン／クローズドであることにより、得られる学びの性質は変わる

@ditflame

学べる場というのも世の中には色々とありますが、今回はそういった「場」、すなわちコミュニティにおける、オープンコミュニティとクローズドコミュニティにおいて、得られる学びの差について考えてみましょう。

31.1 オープンコミュニティについて

まずは、参加が自由に行える（情報が公開されており、誰でも参加できる）コミュニティについて、メリット・デメリットを考えてみます。

よくあるのは分野を横断した勉強会や、LT 会、オープンカンファレンスなどです。

31.1.1 オープンコミュニティのメリット

オープンコミュニティのメリットには次のようなものが挙げられます。

- だれでも自由にいつでも・どんな状態でも参加できる *1
- 結果、人の新陳代謝が早い
- コミュニティによっては話す内容がバラバラなので新たな知らない分野の話を聞ける事がある

31.1.2 オープンコミュニティのデメリット

逆に、デメリットは次のようなものが挙げられます。

- 話を聞いてみたはいいが、専門外の話だとよくわからん事がある
- 自分が学びたい分野の専門家はその会には少ないかもしれません
- 守秘義務のあるタイプの話 *2 はそうそう出てこない
- そのコミュニティに興味のある人しか来ない

*1 初学者などその分野に入ってみました・入りたいけど気になってます 的な人でも参加できる

*2 抽象化した上でならかろうじて……

31.2 クローズドコミュニティについて

逆に学びの場としては、クローズドなコミュニティが存在する事もあります。

これは何らかの理由でオープンとなっていないコミュニティで、具体的なケースとしては

- ・特定の資格を所持している人を対象とする、有資格者コミュニティ
- ・大学のOB会
- ・大学の研究室のOB会
- ・大学のサークル^{*3}のOB会
- ・町内会・マンションの管理組合など地域的な繋がり^{*4}

などがあります。

31.2.1 クローズドコミュニティのメリット

クローズドコミュニティはオープンではない事が問題となるケースもありますが、それを補って余りあるメリットがある事もあります。

- ・有資格者コミュニティの場合、その分野のエキスパートが揃うので尖った情報が得やすく、壁打ちもしやすい
- ・古参の人とかがいて、その人しか知り得ない知見^{*5}を教えてくれるケースがままある
- ・資格によっては守秘義務必須だったりするので、逆にその会の中では（皆が有資格者なので）守秘のある話でも情報が出てくるケースがままある

31.2.2 クローズドコミュニティのデメリット

逆に、次のようなデメリットもあります。

- ・クローズドになっている理由を開けられる（ドアノックできる）もの^{*6}が自分にないと会にすら入れない（ケースによっては知る機会すら無い）
- ・新陳代謝が遅い 起きにくく

なお会社に所属する というのもクローズドコミュニティに近いもの・特性があるかもしれません（それこそ、情報が得られるかどうか？ みたいな観点においては）。

^{*3} 技術系のサークルというのが大学によっては存在し、OB同士が繋がって意見交換しているケースがままあります。

^{*4} 本筋とはすこしづれるかもしれません、筆者としては「特定の地域などのレイヤーに引っかかる人を対象とするため、それ以外の分野を横断した形で繋がりが発生する」点で無視できないような気がしています。

^{*5} 例えば、歴史的な経緯とか、表に出てこなかった裏話的な歴史、失敗談など。特に失敗には学びが詰まっています。

^{*6} これの代表格は、有資格者コミュニティにおける特定の資格でしょう。

31.3 おわりに

エキスパートとして学びを進めていく中で、コミュニティに参加したほうが良い というのは恐らく理解を得やすいと思うのですが、クローズドコミュニティに参加する事でより学びを先鋭化する事ができ、 オープンコミュニティで幅広く学んでみる事で、より広範な視野を得る事ができそうです。

そういう点なども勘案しながら学びをすすめて行けると、筆者としてはより学びが深くなりそうな気がしています。

どうでしょうか？

第 32 章

興味関心を持つために情報や人の近くにいる

32.1 はじめに

うーたん (uutan1108)

うーたん (@uutan1108) です。株式会社ゆめみでサーバーサイドエンジニアをしています。私は、現在新卒 2 年目のエンジニアで、経験も浅いですが、様々なことをしてきたつもりなので、今までの経験からの学びを書きます。

これまで、他の技術同人誌に寄稿、コンピューター雑誌に寄稿、技術カンファレンスでの登壇、技術コミュニティの主催なども行ってきました。

これらの経験を通じて、常に情報や人の近くにいることが興味関心をかき立てる鍵ではないかと考えるようになりました。

この章では、情報や人の近くにいたことでどのように興味関心が生まれ、それが何に繋がったのかを書きます。

32.2 興味関心を持つことが学びのスタート地点

まず、私は「興味関心を持つことが学びのスタート地点」だと思っています。この本では「学び」についてたくさん書かれていますが、その前段階にある「情報を得る」や「興味を持つ」ために私がしてきたことについて書きます。

皆さんも、何かの情報を見聞きして興味を持ち、その情報をきっかけに学びを深めていった経験があるのではないでしょうか。

32.2.1 情報を得るために

情報は、人から媒体を通じて伝わります。そのため、まずは媒体を探すところから始まります。例えば、X (旧 Twitter) などの SNS、YouTube などの動画、あるいは書籍、有名な技術系ブログなどです。また、情報が集まる場として、勉強会やカンファレンスも有益です。

これらの媒体や場に積極的に触れることが「学び」のファーストステップだと考えています。

また、特定の媒体だけではなく、様々な媒体から情報を得ることで、興味の幅を広げるきっかけになることがあります。

32.2.2 人の近くにいる

情報は人から生まれます。そのため、情報を発信している人の近くにいることで、興味関心をかき立てられることが多いです。また、媒体を介さないで直接触れることで、見えなかった情報や背景が理解できる場合もあります。

どのように近づけばいいのかわからない場合は、Xなどでフォローしている人がいそうな場所に足を運び、その人が発信している情報について感想を伝えるところから始めると良いと思います。

32.3 負の状態から脱却するのは新しい興味を見つけた瞬間

「あの人凄そう。自分も何かしたいけど、何をしたいのかわからない。」「何かしたいけど、何もしたいことがない。でも、何かしたくて情報を追ったり、人と話したりしている。」

私は、このような負の状態によく陥ります。陥ると表現しますが、悪いことだとは思っていません。このような感情を持っても数ヶ月後には消えていることが多いような気がします。

何かしたいけど何をしたいのか分からぬという時期や瞬間はあると思います。その状態を抜け出す時も人や情報だと思っています。

何気なく目に飛び込んできた情報がきっかけで、やる気が湧いてきたり、他者からの何気ない言葉がきっかけで新しいことに挑戦する勇気をもらえることがあります。

32.4 具体的に私が人や情報で学びに繋げられた事例

今まで抽象的な話を書いてきたので、最後に私の経験について書きます。

32.4.1 場に参加して人と話せたからコンピューター雑誌に寄稿できた

技術同人誌を書いてみたいけど書く勇気がないと思っていました。負の感情を持ち続けていた状態でした。

しかし、技術同人誌即売会に参加して、コンピューター雑誌に寄稿できることを知り、技術同人誌を執筆されている方々に相談して勇気をもらえたのがきっかけで寄稿しようと思えました。

単に「寄稿できる」という情報だけだったら、寄稿せずに諦めていたかもしれません。場に参加して人と話して興味関心を駆り立てられたのが要因で踏み出せたと思っています。

32.4.2 場に参加して技術カンファレンスなどで登壇したいと思えた

私と同期のエンジニアが登壇していて、自分もしたいと思ったのがきっかけでした。場所に参加せずに、どのような人が話しているのかに関心を持たなければ、自分も登壇してみたいと思うことはなかったと思います。

そして、技術カンファレンスについて知ったのも情報を追っていたからであり、情報を追っていないければ開催されていることすら知らなかったと思います。

32.4.3 場に参加して勉強会運営に誘われて、自分でもエンジニアコミュニティーの主催をした

技術カンファレンスに参加し、そこで会った方に誘っていただいたのがきっかけでした。それがきっかけで、勉強会を開く側を経験することができました。それ以前から勉強会には参加や登壇していて、雰囲気や必要そうなことは知っていたので誘っていただけたのかなと思います。

その後、自分でも勉強会を主催することにしました。主催するときも、今までに会った方や X で相互フォローしていた方々に来ていただき、嬉しかったことを覚えています。

32.5 まとめ

情報を得ることや人とのつながりを持つことは、興味や学びを生み出す大きなきっかけになります。特に、実際に場に足を運んで人と話すことは、情報だけでは得られない具体的な刺激や、踏み出すための勇気を与えてくれます。また、他者の経験に触れることで、新しい視点を得たり、自分では気づかなかっただ可能性に気づくこともあります。

「何かしたいけれど具体的にわからない」という負の状態に陥ることははあると思います。この状態自体が悪いわけではありませんが、そこから抜け出すためには、自ら情報を追いかけたり、人と接点を持つことが有効です。何気ない会話やちょっとした情報が、新しい興味を見つけたり、新しい挑戦を始めるきっかけになるかもしれません。

また、情報を得る際には、一つの媒体や一つの場に偏らず、複数から得ること重要だと思います。そして、複数から得ることで興味の幅を広げるのにも役立つと思います。

そして、情報を蓄え、人とのつながりを持つ習慣を作ることで、自然と新しい機会や挑戦が生まれやすくなると思います。

私自身の経験からも、情報や人との関わりが新しい学びや成長の原動力になると感じています。技術同人誌の執筆やカンファレンスでの登壇、コミュニティの主催なども、情報を得て人とつながった結果、実現できたことです。このような実体験を通じて、情報と人の近くにいることの大切さを実感しました。

これからも情報を蓄え、人と交流する習慣を続け、興味を深め、新しい挑戦をしていきたいと思っています。

第 V 部

学びの Tips

学びを深める、効率的に行うための様々な Tips を集めてみました。

効率化のための方法、学びを広く深くする方法、あるいは今自分が体験・実践していることなど。

他の人はどうしているんだろう？

そういったところをぜひチェックしてみてください。

メタ認知のすゝめ

FORTE(フォルテ)@FORTEgp05

執筆者のFORTE(フォルテ)と申します。最近、知人と会話していてとある気づきと学びを得たので、自分の中の整理も兼ねてアウトプットしてみます。メタ認知ってなに? という人はもちろん、もしあなたが次のように思っていたらぜひ読んでみてほしいです。

- 自分はメタ認知できているから大丈夫
- 他人の意見はよく聞いて受け入れる方だ
- 既にもういくつも試して自分に合うやり方を分かっている
 - だから新しく試したり変える必要はない
- 考え方を変えて損するのは嫌、だから変えたくない

33.1 メタ認知とは

メタ認知というのは自分の認知を認知しているか、ということになります。では認知とは何かというと、ある事柄に対して自分がどう理解しているかになります。

例えば泥棒と聞いて犯罪や悪いことthoughtしたり、駅で困っている人を助けたと聞くと良いことだthoughtしたり、パクチーと聞いて大好きthoughtりあまり好きではないthoughtたり、そういったある事柄に対して自分がどう思うかどう理解しているかを認知といいます。特に感覚的なものであったりとか感情的なものに限定している訳ではなく、どう理解しているか、どう考えているか、判断しているかなどが認知というものの重要な点となります。

メタ認知に戻ると、自分の認知を自分が認知できているかという事になります。イメージとしては主観視点ではなく三人称視点、自分の言動を自分の視点からではなく天から見ている感じです。正しくメタ認知ができないと自分は仕事ができると思っていても実際はそうではないとか、自分はできないダメな人間だ…と思っていても実はそうではなかったりと自分の認識と客観的な事実がズレたりします。このズレに気づいて欲しいと思っています。

もう少し具体的な例でメタ認知について知って、認知してみましょう。

33.1.1 アンコンシャスバイアス

メタ認知できるいい例としてアンコンシャスバイアスがあります。これは日本語で「無意識に思い込み」と訳され、自分が無意識のうちに当たり前だと思っている事柄が実は偏見だったりする、偏見になってしまっていたときに使われる言葉です。ご存知ない方も多いかもしれません、日本

政府からも人権の観点から広報活動などがされています。実際に政府広報のページを紹介します。

<https://www.gov-online.go.jp/tokusyu/unconsciousbias/>

このサイトからアンコンシャスバイアスの例を引用してみましょう。

- 女性に理系の進路（学校・職業）は向いていない
- 仕事より育児を優先する男性は仕事へのやる気が低い
- 性別による向き、不向きがある

これらが当たり前だと思った方は、アンコンシャスバイアス（無意識の思い込み）があるという事になります。そして、それが認知できていないのであればメタ認知できていないという事になります。もう少し例を見てみましょう。

- レディースデーで 500 円割引
- 生理休暇
- 女性専用車両

これらをみて普通のことであり何も思わない、感じないのであれば、アンコンシャスバイアスかもしれません。レディースデーは企業の利潤追求のために価格の感じ方の違いに着目して設定されたと言われることが多いですが、男性だから女性だからという括りで価格の感じ方が全く同じということはないでしょう。価格という点で言えば自由にできるお金の差や、価格がつけられた商品やサービスにどれだけの価値を見出しているかの方が重要なのではないでしょうか。もちろん昔はそんな細かい点では判断できなかったので分かりやすく見た目でわかりやすい男女にしたのかもしれません（見た目は見た目でどうかとは思いますが）

生理休暇も同様です。身体の構造の問題で一定の周期で体調が悪くなるためというのは理解できますが、男性だって体質などが原因で一定の周期で体調不良になる人もいるでしょうし、性別問わずに体調不良になることはあるでしょう。そうであれば生理休暇ではなく病気休暇の方がいいのではないか？ という話もあります。そして体調は人それぞれ違うので、単に病気休暇にするとよく体調不良になる人ばかり病気休暇が消費され健康な人ほど休暇が残るという不平等さがあると思う人もいます。理想論で言えば個人にアジャストした働き方ができればいいのですが現実的にはいろいろと課題があるとは思います。生理休暇という言葉だけでここまで認知を書きましたが、あなたはここまで意識や理解、考えがありましたか？

女性専用車両は今までと同じ流れなので詳細には書かないですが、この例は男性差別や女尊男卑と呼ばれる考え方です。ここで書きたいのは性差別の実態についてではなく読んでいるあなたのアンコンシャスバイアスについて、ひいてはメタ認知はどうでしたか？ という事です。何も感じなかったことが問題なのではなく、真剣に考えたり行動していることが良いのではありません。自分が普段どう考えているか思っているか理解しているか、認知できていましたか？ という事です。自分は当たり前だと思い込んで何も感じていなかったな…とか、何かモヤモヤを感じていたり問題だと思っていたな…と認知できたこと、それがメタ認知なのです。

33.2 メタ認知できていない人

メタ認知について説明したところでこれができないとどういう問題があるのか、客観的にどういう問題があるように見えるのか、という点を紹介していきたいと思います。メタ認知できていない人は私にはこう見えています。

- 生き急いでいる
 - ただし何に焦っているか分からぬしその焦りが必要なのかも分からぬし焦っている自覚もなさそう
- 自分の思考が凝り固まっている事に気づいていない
 - 自分は○○だからなあと問題点のように言うが改善しようとする素振りはない
- 物事には何事も多面性がある(可能性がある)という事が分かっていない
 - ある一面だけ見て判断てしまっている、そしてそれに気づいていない
- 自分の可能性を信じられない
 - 狹いまたは誤った認知で勝手に天井や線引きしチャレンジや改善をしない

いろいろと書きましたが、簡潔にすると身近な知人では心配になりますし、相手がどうでもいい人ならストレスが溜まります。とはいっても他人は変えられません。馬を水辺に連れていくことはできても、馬に水を飲ませることはできないのです。

そうなるとどうなるか、親しさの度合いに関わらず自然とその人とは疎遠になっていきます。どうでもいい人なら別に疎遠になってしまってもいいのですが、せっかく身近と言える知人なのであれば末長く居心地のよい関係性でいたいものです。これは自戒も込めていますが、お互いにストレスを感じないようにしたい、何かあればフィードバックしあって良くしていく関係性でありたいという願いが込められた物語りもあります。

ではどうすればメタ認知しこれらの客観的に見た問題点を自力で克服できるか、考えてみます。

33.2.1 常に自分の言動のふりかえり(メタ認知)をする

当たり前のことと当たり前に書いているだけになってしまっているのですが、えてして基本にして王道とはこういうものなのかなと思います。やり方としては簡単で、ことあるごとに今の自分の言動はどうだったかをふりかえります。例えば朝起きて家族に挨拶したときをふりかえり、自分は眠そうだったな、機嫌が悪そうだった、おはようしか言わなかつたけどもう一言二言あってもよかったです。これら全てを問題視する必要はないですし、すべてどうでもいいと切り捨てるものでもありません。大事なのはまずありのままに自分の言動を認知することです。

自分の言動を認知したら、その認知に対して自分がどう思うか認知します。先ほどの例で言えば機嫌が悪そうだったのは自分では良くないと思うな、とか、眠そうだったのは寝るのが遅かったり睡眠の質が良くなかったんじゃないか、など自分の言動に対する認知を意識してみます。そこから学びがあれば今後の行動に活かしていくチャンスです。メタ認知ができるメリットを享受してい

るといえるでしょう。

さて、この一連の流れを自分の言動すべてに対して行うべきかというとそうでもありません。例えばトイレに行こうと思って席を立ち、右足から踏み出した今の言動についてメタ認知を…なんてやっていたら1日が終わってしまいます。なのである程度の期間で区切ることをお勧めします。一番わかりやすいのは夜寝る時に1日をふりかえりすることですが、時間が経ってしまうともう細かくは思い出せなかつたり、そもそもメタ認知せずに忘れて寝てしまうことだって起こり得ます。そのため、時間を決めるとしたらできれば30分から1時間くらい、あるいは誰かとの会話が終った直後や、外出して帰ったあと、身支度をしたとき、なんでもいいのですが、ふと外から内へ意識が向いた時にいまの言動についてふりかえりをできると上手くメタ認知する癖がつけられるでしょう。もし明示的に時間を区切った方がやりやすければポモドーロのような時間を区切るテクニックが役に立つかかもしれません。

33.2.2 ジョハリの窓

メタ認知を知る上で、ぜひもう一つ知っておいてほしい概念があります。それがジョハリの窓です。これは対人関係における気づきのモデルとして発表されたものですが、自分自身に対しても使えますし、メタ認知には重要な概念があります。

まずジョハリの窓とは自分にたいして自分が知っている/知らない、他人が知っている/知らないを組み合わせた次の4つの概念です。

- 自分は知っていて、他人は知らないもの
 - 秘密の窓
- 自分は知っていて、他人も知っているもの
 - 解放の窓
- 自分は知らない、他人も知らないもの
 - 未知の窓
- 自分は知らない、他人も知っているもの
 - 盲点の窓

これはメタ認知をした際に非常に示唆に富む概念だと思います。自分が知っている二つの窓に関しては既に知っているので自分としてはそういうものだなで済む話ではありますが、それを他人から見たらどうか?というのは面白い観点だと思います。例えば自分が知っていることは自分の認知としては当たり前なので、他人も知っていて当たり前と思い込んでいる場合があります。実際には他人が知っているかどうかは分からないので、その認知は秘密の窓になっていたりするわけです。これは自分をPRしたい時に非常に役立つ観点だと思います。例えば会社で評価される際や、営業をする際など他人に働きかけるときは自分が知っていて当たり前と思っていても他人が知らなければPRポイントになったりします。メタ認知して考えてみましょう。

次に自分が知らないものです。これはメタ認知の上で非常に重要なポイントとなります。特に盲点の窓は文字通りメタ認知できていないため盲点となっており、なおかつ他人は知っているという

状態になっています。これはあの人は自分が気づいていないけどこんないい点があるというようなポジティブな内容であればまだいいですが、あの人はこういう悪い癖があるというようなネガティブな内容であれば是非ともメタ認知したいものです。ポジティブな内容でも認知できていれば他人から評価されている点であることからもPRなどに使えるため、認知できると得られるメリットは大きいでしょう。

最後に自分も知らない、他人も知らない未知の窓についてメタ認知とどう関係があるかというと、自分には未知の窓があるということをメタ認知(自覚)することが大事です。つまり、あなたはどこまでいっても未知のあなたがいるということです。そしてそれは、あなたにはどこまでいっても新たな可能性があるということです。ガラスの天井なんて言い方をされますが、環境や周囲の圧力はともかくあなた自身に限界などありません。常に未知の窓があることを忘れず、自分の言動をふりかえることで未知の自分を認知することができるでしょう。そしてそれこそが学びであり、成長であると思っています。

フィードバックを貰う

これはジョハリの窓でいうところの盲点の窓を認知するためのものです。これは盲点なのだから他人からフィードバックを貰おうということではありません。自分にとって盲点だからこそ、どうすればメタ認知できめようになるかを考えてほしいのです。これは完璧にメタ認知をできるようになろう、そうでなければダメという完璧なメタ認知を目指すものではありません。フィードバックをきっかけに自分が知らない自分がいることを認知し、フィードバックを含めた様々な手法でそれを認知する方法を考えてほしいのです。そこから得られる学び、気づきが必ずやあなたの役に立つでしょう。もちろん完璧には無理です。ですが、完璧主義になって最初から無理だと諦めずにそれでも、と理想に向かって足掻いていくその行為、嘗めこそ大事なものなのではないかと、私は思います。

33.2.3 損得は多面的に考える

物事には必ずといっていいほど多面性がありますが、こと損得、とくにお金が絡む話になると正常にメタ認知できなくなってしまう方が多いと思います。例えば100円安いから遠いスーパーまで買い物にいくという話を聞いて、100円でも安く買うのはよいと思う人もいれば、100円程度で余計に時間をかけるのはもったいないという人もいます。これはそれぞれの意見なのでどちらが正解というのではないのですが、その人にとってどういう得があるか、多面的に考えるメタ認知は重要だと思います。例えば100円程度で時間をかけるのはもったいないというのはその通りかもしれないですが、時短やタイパに囚われすぎていなか?というメタ認知をするのは重要です。時間をかけてスーパーまで移動することはデメリットだけではありません。移動手段が徒歩や自転車などあれば運動になりますし、車やバイクなどでも運転が好きな人にとってはメンタル面で気分転換ができます。特に現代人は運動する機会が減っていますので、100円得しつつ運動にもなると言わると、えてして悪いことだけではないように思えてきます。このように物事には多面性があり、時間がもったいないという一面性に囚われるているかも?ということをメタ認知できると取りうる

選択肢や相手の理解が進みます。

もうひとつ例として、家賃を挙げてみます。お金がもったいないと思う方の中にはなるべく安い家賃の築古の物件を選ぶ傾向があります。これはこれで悪いことではないのですが、築浅で家賃が高いことによるメリットについて認知できていない、知ろうともしていないことがあります。著者はいま築2年弱ほどの1kとしては比較的家賃が高い物件に住んでいますが、安い賃貸物件より明らかに良いと感じられる点があります。いくつか挙げてみましょう。

- 室内の設備が新しく性能がよい、便利、高級感を感じる
 - 床が傷つきづらい（傷つかない？）
 - キッチンシンクにシャワー・ヘッドがついている
 - 室内の断熱性が高い（エアコンの効きがよい）
 - センサーによる玄関のオートライト
 - 広めの室内と多めの収納
 - 風呂トイレ別
 - 独立洗面台
 - 浴室乾燥・暖房あり
 - 汚れづらい
 - Wi-Fi備え付き
- 大きめ、かつ、余裕がある宅配ボックス
- 24時間ゴミ出し可能な敷地内ゴミ捨て場
- 住民の民度の高いため騒音など他の住民による悩み事が少ない
- 虫が湧かない（湧きづらい）
- 新しいのでメンテナンス頻度が少なくて済む
- 管理会社が大手で入居者の審査や設備メンテナンスの評判がよい
- 駅から徒歩10分程度
- 東京23区内
- 山手線まで電車で15分未満

これらの中で築浅じゃない自分の部屋でもその設備やメリットがあるよ！という人はいるでしょうけども、これらがすべて揃っているということはあまりないと思います。これはだから築浅が高いということではなく、家賃が高いということだけでこれらのメリットを知らず、または体験もせずに「高いから」というだけで自分の人生から遠ざけてしまい機会を損失するのはもったいないと言えます。もちろん絶対に知らなければならないことではありませんし、必須なものではないのです。ただ、老後には必要以上のお金は不要という話があります。これは老後にいくらお金があってもだいたいのことには身体がついてきませんし、それが体験できるときになるべく早く体験した方がよいという話に基づいています。お金は使える時に使ってこそ意味があるものだと主張した本でベストセラーになった作家からの引用です。

安いことがあるひとつの一面性でしかなく、絶対的なものではありません。そこには安い代わりになくなったり、あるいは得られなかったものがあると思います。あなたはそれをメタ認知で

きていますか？

33.2.4 自分の背水の陣を決めてチャレンジする

自分の自分に対する思い込みをメタ認知できると、その思い込み（フレーミング）をなくしてさらにチャレンジしたり行動を変えたりできます。それにはまず自分が思い込んでいるということをメタ認知する必要があります。その上でその思い込みは本当に自分にとってどうしても譲れないものなのかなと考えてみます。具体的に例を出してみましょう。

もしもあなたが賃貸物件の家賃は安い方がいい、なぜならば一度でも高い家賃の部屋に住んでその快適さを経験してしまうと生活水準を下げて低い家賃の部屋に住めなくなるから、と思い込んでいるとします。これは一見するとメタ認知できているように見えます。自分の考えに自覚的であり、なぜ高い家賃ではなく安い家賃の家に住みたいか説明できているからです。しかし、メタ認知できていない点もあると思います。それは自分の可能性を思い込みによって潰してしまっているという点です。ここで可能性というのは、快適さに慣れても自分は生活水準を下げることができるかもしれないという可能性です。この可能性を活かし金銭的余裕があるうちは生活水準を上げる、余裕がなくなれば生活水準を下げるということができれば人生全体の幸福度はより上がるとも言えます。ではどうすればこの可能性を潰さずに生かすことができるでしょうか。それが自分にとっての背水の陣を決めてチャレンジすることです。

この背水の陣とは中国の故事成語であり、戦で背後を川や湖などにして退路がない陣を形成しあえて不利となることで味方の士気を上げて戦いに勝ったことが由来となった故事からきてています。これを応用すると自分にとって絶対に譲れない点（背水）を決めることで、それ以外はチャレンジできる、してみることが可能になります。著者の場合は前述した通り高い家賃の部屋に住んでいますが、家賃の高い賃貸に住むことは自分の中でどうしても譲れない点ではありません。必要なら生活水準を下げるることは厭いません。しかしそれ以外では、どうしても譲れない点があります。それは自分が一番好きな月額制ネットゲームができなくなることです。これだけは譲れませんが、逆に言えばそれ以外の全ては譲れます。そのため、生活水準が下げられないかもと不安に思ったりはしません。譲れないものがあるからこそ、それ以外を全て犠牲にしてでもチャレンジすることができます。そして、譲れないもの以外の全てを失ってもそこからまた這い上がることができると言じていますし、信じられます。これは自分にとって失いたくものが残っているからこそ、そう思えますし自分を信じられるのです。

このようにできないかもしれないからやらないより、どうしても譲れない点だけを決めてそれだけは守りつつ積極的にチャレンジしていく生き方の方がより多くのものが手に入るのではないでしょうか。もちろん、それがよい、そうすべきだと言っているのではないのです。そう、それを知っているかどうか、メタ認知した上で自分はどうしたいかだと思っています。ただ、自分がそういった生き方ができているかどうかメタ認知できなければ、どうしたいか?と自問することもできません。機会を増やすという点からもメタ認知することはおすすめできると言えます。

33.3 さいごに

今回は知人との会話から気づきと学びを得たのでアウトプットしてみましたが、その知人を含めて誰かに変わってほしい、変わるべきだ、とは思っていません。思っていないというより、執着はしていません。もしこの章が変わるきっかけになったら、こんなに嬉しいことはないですし、そういうきっかけになれたらと執筆しています。ですが、変わらないからといって私のストレスになったりはしません。

では、なぜこの章を書いたのかというと結果ではなく書くという行為、伝われば良いなと筆を取ったその営為そのものが重要だと思っているからです。これもメタ認知ですね。

さて、学びというテーマだと、何でもかんでも学びであるとしてしまえば原稿になってしまふのが未恐ろしいです。これからたくさんの学びと出逢いますし、またこういった形で形に残していくことはあるかと思います。またその時にお会いしましょう。

やる気に頼らず学ぶ、自分プログラミングという考え方

@ponyoxa

34.1 はじめに

みなさん、やる気はお持ちですか？ いつでもやる気があってたくさんのインプットあるいはアウトプットをコンスタントにすることができる、業務も生活も問題なく回っている、そういう状態なら何よりです！ そういう方にはこの章は役に立たないと思います。むしろ、どうやってやる気を維持し、一定したインプット、アウトプットを生み出しているのか、毎日安定した生活を営んでいるのかといった知見を共有していただきたいです。

学びにもいろいろあると思いますが、どこかに出かける、ドキュメントを読む、頭の中を整理する時間をとる、などなど学ぶためには学びに向き合うという行動がたびたび必要かと思います。

この章では、「学び」には必要不可欠な「行動」について、やる気駆動ではない方法についてご紹介したいと思います。やりたいこと、学びたいものがあるにも関わらず、「やる気がないから何もできないよ」、「やり始めてもやる気なんて出ないよ」という方に向けて、「そんな不確かなものに頼るのはやめよう」と声をかけたいと思います。

やる気が出たのも束の間、次の日には「あれ、あのやる気どこいった？」となり、なかなか行動につながらないあなた向けです。

34.2 そもそも「やる気」とは？

やる気はどういうときに出るでしょうか。外からの刺激を受けて「自分もがんばろう！」となったり、目標設定をして「達成するぞ！」となったり。その時点ではやる気はマックスになっていることでしょう。しかし、翌日、あるいは1週間後にそのやる気がそのまま持続していたことはありますか？ だいたいはやる気は減退しており、場合によってはやる気のかけらも見えなくなっていることもあるかもしれません。それは筆者も同じです。勉強会に行って、「わたしもがんばるぞ！」となりますますがだいたい次の日にはすっと消えています。

34.3 自分プログラミングの考え方

明日あるかもわからないやる気に頼って行動しようとするのは不安定だと思いませんか。やる気をコントロールすれば良いというわけですが、不安定なものを安定させるにはエネルギーが必要です。ではそのエネルギーはどこから持つてくればいいんでしょうか。

何にしてもどこからかエネルギーを持ってきて、不安定状態を安定状態にする必要があります。そのため、やる気に頼ったり、それをコントロールしようとすると、やりたいことを成し遂げるために必ずエネルギーが必要になり、ハードモードになります。やる気があるときはいいのですが、やる気はたいてい持続しないため、やる気に頼ったやり方をするとやる気がなくなったとたん行動を始めるのが難しくなってしまいます。

ここでは筆者の経験から、やる気に頼らない行動の構築方法を紹介します。

34.3.1 if-then

一つ目は、「if-then」で行動するということで、「もし～だったら、～する」と決めてしまうことです。自分をプログラミングしているという感覚に近いと筆者は考えています。

筆者が実践している例としては、「もし出勤したら、Feedlyを開く」、「もしわからない言葉に出会ったら、メモをする」、「もしイベントに参加したら、ポメラでメモをとる」、「もしライブラリについて疑問が出たら、最初に公式ドキュメントを開く」などがあります。

このように自分が好みと思う行動パターンを自分にプログラミングします。「もし～だったら」の部分はできるだけ日常的にすでにしていることに設定すると、「～する」の部分に取り組みやすいです。

この方法の運用は失敗することももちろんあります。筆者もいろいろと失敗してきました。例えば、集中力を少しでも持続させようと「お手洗いに立ったらスクワットをする」ということを考えましたが、「スクワットをする」ということがだんだん面倒になってきてハードルが高くなり、お手洗いに行くのを我慢してしまうようになったという失敗がありました。これは「スクワット」の部分に対しての負荷調整が必要だったと思います。

34.3.2 自分のやる気を奪ってしまう行動をしない

環境作りという面でも、マイナスになってしまふような行動をしないというのは重要なことだと思います。「この行動をした後にはやりたいことができなくなるな」ということを把握することから始めましょう。

筆者の場合には、「すぐに座らない」ということをかなり意識しています。座ってしまうと立ち上がるのがおっくうになり、次の行動に移るのがかなり難しくなってしまうからです。筆者は自宅が和室で地面に近い生活をしていることもあって、平日帰宅後に座ってしまうとその後立ち上がって何かをすることは困難になってしまいます。

このような場合には、「座らない」ことが次の行動への助けになります。次の行動を妨げる行動を把握し、その行動をしないということです。具体的な動きとしては、座る前に飲み物を用意す

る、必要な本やノートなどをテーブルに用意するといったことをし、学びに向かってから立ち上がりなくてもいいように環境作りをしています。

具体的にどのような行動がやる気を奪ってしまうのかというのは、人によって様々でしょう。以下は一例です。あてはまっているものはないでしょうか。

- 動画を見始める
- すぐにベッドに横になる
- パソコンに電源を入れた後、まずネットサーフィンをする

自分を観察し、行動に移せているときと移せていないときで何が違うのかということを明らかにしましょう。やる気を奪ってしまう行動だけでなく、やる気にさせてくれる行動も発見できたらうれしいですね。

34.3.3 好ましくない行動をどうやってやめるのか

ある行動をするとやる気を奪ってしまうからといってその行動をすぐさまやめるのは難しいことだと思います。行動は癖になっていることも多く、また、従来通りの行動から変化をするのは脳にとっても負荷が高いためです。

最初のうちは、自分の観察をしながら、10回に1回でもうまくいったら万々歳くらいの感覚でいたほうが続けやすいです。気長にやっていきましょう。

好ましくない行動をしている自分を観察しながら、「～の状態のときには～しやすい」というようなパターンを見つけられると良いと思います。例えば、「仕事が溜まっているときは、夜ご飯にジャンクフードを食べてしまいがち」、「イベント行った日はそれだけで満足してしまいがち」といった自己分析を通して、少しずつ改善していくのが地道かつ確実だと思います。

筆者の経験では、極端なこと、無茶なことをしてしまうと、心が折れやすかったり自己嫌悪にながってしまうことが多かったので、少しずつ地道にやっていくのが結果的に多くのことを積み重ねられるかなと考えてみます。

34.4 終わりに

大前提として、行動をするのは非常に困難なことです。それがやりたいことであっても、スムーズにやるのは難しいこともあります。

筆者は熱しやすく冷めやすいタイプなので、熱されたときのやる気を維持したいと思ってこれまでの人生やってきましたが、やる気にあれこれアプローチする方法が自分には向いてないんだろうと思うに至りました。そのため、やる気があるときはやる気と協力こそすれ、やる気が不在なときでも気にせずやるという態度を身につけようとする考え方へ移行しました。

この考え方で良かったことは、毎日何かしら進むことができているだろうという実感が持てることがあります。とりあえず、何もしていないわけではなく、やることはできているからです。やる気が出ないときでも、資格試験日は近づいてきますし、なんとかやっていかないとやらないとい

けない毎日のことは溜まっていき、毎日のことをやることで消耗してしまい学ぶ時間を持てないと
いう悪循環に陥ってしまいます。

筆者はこれまであまり深く考えず、「朝起きたら、20分資格の勉強をしよう」、「電車で座れたら、
ポメラを開こう」という、やる気が介在しないシンプルな考え方をすることで、学びにつなげてく
ることができました。

寄稿することができたのも、「自分をプログラミングする」ということをイメージした考え方の
おかげです。シンプルな「if-then」から皆さんもぜひ試してみてください。

学びの時間を最大限増やす方法

ゆーき@lizhi05853

本章に目を留めていただきありがとうございます。

ここでは、「勉強したいけど時間がない」「勉強はしているけど、まだまだ時間が足りない」という方に、少しでも勉強時間を増やせるような方法をご提案します。

対象は日常生活で勉強に充てづらい下記 3 つの時間帯です。

- お風呂時間
- 睡眠時間
- 勤務時間

読む順番はありませんので、気になった節からご覧ください。

35.1 お風呂時間に学ぶ！

お風呂時間は素っ裸だし、机にも向かえないし、せいぜい入浴中に本を読むのが精一杯、と思つていませんか。あるガジェットを使えば、両手が塞がってようが、目を開けられなかろうが、勉強ができます。

それは、『防水イヤホン』です。

防水イヤホンを使って勉強系動画や語学音声を聞けばよいのです。これならお風呂に浸かってても、シャンプーしても、ドライヤーしても勉強できます。

以降、下記内容を解説します。

- 防水イヤホンの選び方
- 各サービス紹介

35.1.1 防水イヤホンの選び方

お風呂での勉強に使う防水イヤホンの選ぶポイントは下記です。

- 防水性能
- イヤホンの形状

防水性能（基本は自己責任！）

防水性能は「IP コード」呼ばれる表示形式で表記されています。例えば、私が普段お風呂で使っているイヤホン 2つについて調べると、「IPX7」と「IP67」となっていました。

じゃあ、「IPX7」か「IP67」の防水イヤホンを選べば問題なしなんだな？ と思うと、そうではありません。私が持っている防水イヤホンと同じ性能のものを使用しても、壊れてしまう可能性はあります。それはなぜでしょうか？

まず、IP コードは 2つの性能を表したコードです。「IP」に続く 1つ目の数字が防塵性能、2つ目の文字が防水性能です。もし数字ではなく「X」になっている場合は省略を表しています。防塵性能は 0 から 6 まで 7 段階あり、防水性能は 0 から 8 までの 9 段階があります。

しかし、注意が必要なのは、単純に最大レベルの「IPX8」の製品を選べばいいんだな！ とはならないところです。参考リンク^{*1}などをご覧いただくと、「IPX5」や「IPX6」は噴流水でのテストをクリアしていますが、「IPX7」や「IPX8」は水没させるテストをしています。仮に「IPX7」と書いてあったとしても、シャワー時に使ったら壊れてしまうかもしれないのです。（ちなみに「IPX5/IPX7」と表記される場合は、噴流水のテストも水没テストのクリアしています）

じゃあ、IPX6 を選べば問題なしなんだな？ と思うと、そうでもありません。

IPX6 のテストは直接噴流を 3 分間当てるテストです。髪の長い女性だと、3 分でシャンプー／リンスが終わらないかもしれません。そもそもテスト自体が常温の水道水で行われるため、シャンプー／リンスや皮脂が溶け込んだ温水という条件は何も保証されていないということです。また、ドライヤーをしているときも同様です。

事実として、私は「IPX7」の防水イヤホンを 2 年間、少なくとも週に 2 日は使い続けていますが壊れてはいません。しかし、お風呂時間での防水イヤホンの利用は基本的に自己責任、壊れてしまったらしょうがないという気持ちで実践してみてください。

イヤホンの形状（オススメは完全ワイヤレスのカナル型）

もう一つのポイントはイヤホンの形状です。お風呂時間に使うには、完全ワイヤレスのカナル型を選びましょう。

まず、ワイヤレスイヤホンには「完全」とつくものとつかないものがあります。その違いは、左右のイヤホンの間を繋ぐケーブルの有無です。左右を繋ぐケーブルがあるのが「ワイヤレスイヤホン」で、ケーブルがないものが「完全ワイヤレスイヤホン」です。

なぜ完全ワイヤレスがいいかというと、頭や体を洗うときに邪魔になるからです。

次に、なぜカナル型がよいかと言うと、シャワー中はシャワーの音がうるさく、オープンイヤー型だとイヤホンからの音がかき消されてしまうからです。

35.1.2 各サービス紹介

前節でイヤホンが選べたので、次に動画・音声サービスを選びましょう。

^{*1} kikakurui.com 内 - 電気機械器具の外郭による保護等級（IP コード）: <https://kikakurui.com/c0/C0920-2003-01.html>

YouTube

YouTube 動画は玉石混交ですが、探せば勉強に使える動画も見つかります。オススメはカンファレンスや勉強会の動画。2時間もあってマトモに見たら寝てしまいそうな動画もありますが、お風呂に浸かりながら2倍で聞けばあっという間です。お風呂でながら聞きして、あとで気になった部分をじっくり聞くのがいいです。

Udemy

勉強動画の定番サービスです。課金が必須ですが、IT系講義動画が豊富にあり、広告がないのでオススメです。（私は福利厚生でビジネス版を使っているので助かっています。ありがたい…）ハンズオンや演習問題が多いタイプの講義はながら聞きに向かないので、解説中心の動画を選びましょう。

Audible

自己啓発やマネジメント系の本を読みたい方にオススメです。音声サービスなので、資料を見ずに耳だけで完結するのがよいです。キクタンがいろんな言語で入っているので、語学系の勉強をしている人にもよさそうです。

35.2 睡眠時間に学ぶ！

残念ながら、枕の下にテキストを置いておいたら全部頭に入ってるなんてことはありません。しかし、睡眠時間を勉強の味方につけることは可能です。

それは、『寝る直前に暗記して、起きてすぐにおさらいする』です。

睡眠時、脳内で記憶の整理が行われている、ときいたことはありませんか。私もテレビかなにかで知っただけなので、その科学的根拠の説明はできないのですが、体感として睡眠前後に暗記するのが一番効率がよいと感じています。学生時代は、このテクニックでテストをいくつも乗り切りました。

以降、英単語を500語覚えることを例にとって下記の項で詳しく解説していきます。

- 寝る前の暗記方法
- 起床後のおさらい方法
- 私の暗記テクニック

35.2.1 寝る前の暗記方法

まずは暗記用の教材を入手しましょう。TOEICなどのメジャーな試験を受けようと考えているなら、専用の書籍があるはずです。今どきはアプリを探してみるのもよいでしょう。

暗記に向く教材は、英単語と対応する日本語訳を別々に確認できる仕様になっているものです。例えば、表ページに英単語が並んでいて裏ページに対応する日本語訳が並んでいる書籍であったり、英単語と日本語訳の表示を切り替えられる機能のついたアプリなどです。

ちょうどよい暗記用教材が見つからないときは、自分で作ってしまう手もあります。この節の最後の項に私が学生時代にやっていた方法を記載していますので、参考になさってください。

教材を用意できたらそれをひたすら頭に詰め込んでいきます。その時に私が気をつけているポイントをご紹介します。

雪だるま式で覚える

500単語覚えようとしているとき、一気に500単語は相手にしません。20単語くらいを1セットとして、下記のように詰め込んでいきます。（下記プロセスは声に出したり紙に書いたりするとより暗記率が高まります）

1. 1セット分の英単語と対応する日本語訳を確認する。
2. 1セット分の英単語を隠した状態で、日本語訳から英単語を思い出す
3. 8割ほど正答できるようになったら、次の2セット目で1と2をおこなう
4. 2セット目も8割正答できるようになったら、1セット目と2セット目をまとめて1と2をおこなう
5. 1セット目+2セット目を8割ほど正答できるようになったら、3セット目で1と2をおこなう
6. 3セット目も8割正答できるようになったら、1から3セット目をまとめて1と2をおこなう
7. 以降、少しづつ暗記対象を増やしながら続ける

このように雪だるま式に詰め込んでいくと、暗記効率がよいです。

8割覚えたらOKとする

2つ目のポイントとして、すべてを完璧に覚えようとしないことです。まっさらからから8割覚える労力と、8割から10割覚える労力を比べると後者のほうが大きいと感じています。完璧を目指してずっと同じセットを繰り返し続けるのも気が滅入ってしまいます。8割覚えたらOKとしましょう。

慣れたら順番を入れ替える

前述のプロセスを繰り返していると、前後のつながりで単語を覚えてしまうこともあります。わざと逆順やランダムでやってみましょう。正答率が落ちる可能性が高いです。バラバラの順番でも正答できるように覚え直しましょう。日本語訳→英単語の一方通行ではなく英単語→日本語訳をしてみるのもよいです。

35.2.2 起床後のおさらい方法

寝て起きたら、記憶が定着したか確認しましょう。

時間があれば夜と同じプロセスをおこないたいのですが、たいてい時間に追われているので、1

回通して口に出したり目視したりするのに留めます。

それだけでいいの？と思われるかもしれません、これをするだけで記憶の定着率が高まると感じています。

35.2.3 私の暗記テクニック

この節の最後に、私が学生時代にやっていた暗記教材の自作法をご紹介します。ちょうどよいアプリや書籍が見つからなかったときに試してみてください。

【準備手順】

1. ノートもしくはルーズリーフを縦半分に折る
2. 折り目の左側に 1 行 1 単語ずつ書く（英単語など）
3. 折り目の右側に、左側の同じ行の単語と対応する日本語を書く

【暗記手順】

1. 準備した暗記シートを半分に折って、左側か右側だけが自分に見えるようにする
2. 隠しているほうを思い出しながら、紙に書いたり口に出したりする
3. 1 ページ分終わったら、折り目を開いて答え合わせをする

35.3 勤務時間に学ぶ！

まず、大前提として勤務時間は仕事に全力投球しましょう。仕事に全力投球して定時にあがつて、勉強は勉強で時間を確保するのが一番健全です。

とはいって、1 日の 1/2～1/3 を占める勤務時間をもっと有意義に使いたいと思うのも人情です。そのためのテクニックをご提案します。

35.3.1 仕事内容と勉強内容をすり合わせる

仕事内容と勉強内容を事前にすり合わせておくことが重要です。今の仕事がイヤで転職するために勉強しているという方だと難しいのですが、今の職場でのステップアップを目指して勉強をされている方は職場環境にフォーカスを絞って勉強対象をチューニングしてみましょう。

以下のようなメリットがあります。

- 周囲の理解を得やすい
- 仕事を実践の場にできる
- チームを巻き込める

周囲の理解を得やすい

【手法】

1. 会社が奨励している資格の取得を目指す
2. 仕事で使う技術要素の研修に参加させてもらう

手法1の場合、社内で公言しやすく、制度によっては合格したら受験料の補助を受けられたり、奨励金がもらえたりすることもあります。公言しておくことが重要で、これをしておくとお昼休みに勉強するのを邪魔されにくくなったり、落ちた時に恥ずかしいので自分にプレッシャーをかけたりする効果が期待できます。

手法2は、自分が勉強したいことと業務内容を一致させておくと有効になります。会社によっては毎年度1回は外部研修に参加させられる、というところもあると思います。その時に自分の興味が業務内容と合っていると、積極的に学ぶ姿勢で研修を有意義に過ごすことができます。研修機関の研修は個人で受けると万単位のお金かかるので、会社に支払ってもらうチャンスを逃さないようにしましょう。

仕事を実践の場にできる

【手法】

1. 事前に調べた技術を実践で使ってみる
2. 勉強したいことを、仕事に組み込む

手法1は、例えばLinuxを現場で使っている方なら、事前に家で学習したLinuxコマンドを現場で試してみる、ということができます。私の場合は、勉強したLinuxコマンドを仕事で使う手順書に組み込んでみたら、いつの間にかそれがチームの手順書の標準になっていたことがあって嬉しかったです。

手法2は、例えば趣味で英語の勉強をしている人なら、仕事でエラー調査をするときに積極的に英語ページを自力で読むようにしたり、わからない単語を調べたりしましょう。傍目には仕事をしているだけに見えますし、少し単語を調べるくらいで咎めされることも考えづらいです。テキストとは違う、実践の英語に日々ふれる時間が自然と増えます。

チームを巻き込む

【手法】

1. チームで勉強会を開催する
2. わからないことをメンバーにまとめさせて報告してもらう

手法1は、チームで使う技術のナレッジ共有や技術力向上などと理由をつけて、勉強会を主催してしまうことです。公式な活動として認められれば、資料作成や勉強会自体の時間を勤務時間に入れる 것도できるかもしれません。

手法2は、例えば開発に新しい技術を組み込みたいと考えた時、自分自身で一から調べるのではなく、メンバーに調査タスクとして振ってしまうことです。これは私自身ではなく私の上司がとっていた手法です。オンプレ環境で動いていたソフトウェアをAWSに移植しようというときにこう

いったタスクを振っていました。管理職になれば忙しく、仕事内容も技術から離れてしましますので、ある程度チームの活動に口を出せるような立場の方は検討してみてください。

35.4 さいごに

以上が、私からの勉強時間を増やす方法のご提案でした。少しでもどなたかの参考になっていれば幸いです。

第 36 章

受け身でも持続的に学べる環境作り

ターク@tark_ann

本章を執筆したタークです。社会人から iOS 開発を学び始め 8 年目になりますが、モバイルは毎年のアップデート紹介がイベントとなるくらい大きな変化が生まれ続けていて、日々学習することが特に重要な分野の 1 つだと言えると思います。

プライベートで個人開発する体力はないけれど、なんなく目に付いた技術ブログを読む程度の行動はできる私が考えた、持続的に学ぶ環境についてご紹介します。

36.1 受け身で気が向いたら学ぶことを良しとする割り切り

皆さん、何か勉強しようと思ったとき、「よしやるぞ！」とやる気が出てからチャレンジしていませんか？ 逆に、やる気が出ないときは何もできずだらだらしてしまい、後から後悔するという経験はありませんか？

脳科学的には、やる気というのは行動して初めて出てくるものだという話が広まっています^{*1}。そもそも、脳を使う作業はエネルギー消費が大きく、外から刺激を与えなければ省エネモードのまま変わらないという仕組みになっています。また、やる気が出ないことで学習が進まないことへの罪悪感など、負の側面もあります。したがって、必ずしもやる気がいいこととは限りません。

そこで、自分にとってやる気に頼らず学べる環境とは何かを模索し始めました。最終的にたどり着いたのは、受け身で情報を得られる仕組みを作り、外から興味の刺激を与えられるような環境を作ることでした。

36.2 欲しい情報を整理し、どこで手に入るか調べる

まず、自分が学習したいことを考えてみましょう。業務に役立つスキルを身に着けたいのか、業界の最新情報をキャッチアップしたいのか、キャリアアップのために知識の幅を広げたいのか、色々な方向性がありますが、何が行動しやすい興味になっているのかを考えてみましょう。また、複数同時に学ぶのか、1 つに絞って専念するのかも考えましょう。同時に複数のことを学べるタイプもいれば、苦手なタイプもありますし、その時の状況によって深さや広さも違ってきます。次に、有名な情報源を調べてみましょう。エンジニアであれば、Qiita、Zenn、connpass などで勉強会や学びの情報源を探します。いくつか興味がありそうな記事を開いてみて、その中のリンクを見るよう

^{*1} <https://r25.jp/article/540681193689662300>

になると自然と次の情報源が見つかり、少しづつ情報の幅を広げていくことができます。

36.3 受け身で情報を得られるように仕組み化する

ぱっと思いつく方法として、RSS フィードを使ったりメールで通知を飛ばす方法もありますが、私は採用しませんでした。RSS は普段使ってないため導入の手間があり、メンテナンスをさぼって見なくなってしまうことが容易に想像できてしまったためです。メールは試してみましたが、通知も徐々に煩わしくなって作業中の集中力が阻害されてしまい、結局 OFF にして見なくなってしまいました。

私は趣味の情報収集で使っている Twitter(現 X、筆者の宗教的な理由で旧名を使います) を毎日見る習慣があるので、これを学習用に用意するのが一番行動のハードルが低いと思い採用しました。

36.3.1 有益な情報を集める

情報収集の媒体でよくアウトプットしている人や興味が似ていそうな人の Twitter アカウントをフォロー、もしくはリストに追加します。アウトプットしている人は Twitter アカウントを持っていることが多いので、媒体の通知よりアカウントからの発信を見るほうが鮮度の良い情報を手に入れられます。

私は学習用の Twitter アカウントを作成してフォローしていますが、趣味用アカウントを既に持っていてアカウント切り替えが面倒な方は、Twitter のリスト機能を使うと良いでしょう。ただし、見るときにリストを選ぶひと手間が加わるので、可能ならフォローしていつでも見られる環境にすることをお勧めします。余力があれば、フォローした人がフォローしている人を見ると同じ分野の人たちが見つかるので、追加でフォローするのも良いと思います。

36.3.2 見る習慣を作る

趣味の情報は気の向いたときに見るという、まさしくやる気駆動の行動になっていたので、学習用 Twitter を見る仕組みは別に考えたいと思いました。

例えば、何かをした後に学習用 Twitter を見るようになりますなど、ルーティンワーク化することで少しづつ脳のスイッチが入りやすくなるのでおすすめです。私は、通勤時間など業務前に見るようにルーチン化することから始めましたが、仕事後の休憩時に変えたり、両方のタイミングに増やしたりして無理なくできる範囲を模索していました。Twitter の場合、夜のほうが投稿数が増え情報も多くなるので、こうした媒体情報の性質も考慮しながらルーティンワークを作ると良いでしょう。

注意として、フォローした人がどんなアウトプットをするかに依存するため、必ずしも学習系の情報だけではなく、趣味を含めた雑多なアウトプットが入ってくることも覚悟してください。時間が経つにつれフォローした人が別分野のことばかりアウトプットするようになることもあるので、必要に応じてフォローを整理することも大切です。

36.4 仕組み導入後の効果

Twitterの仕組みを導入後、技術的な流行りだけでなく仕事ですぐ取り入れられるような情報が得られるようになります。仕事への取り組み方が少しづつ変わりました。

例えば、自分がいけなかった勉強会の実況感想が流れるようになります。登壇資料を探して読むといった行動の変化が生まれるようになりました。そのほかにも、ちょっとしたIDEのショートカットなど自身の業務で取り入れられそうな開発Tips的な情報も流れてきて、次の日に試すといった行動が自然にできるようになっていました。その結果、インプットだけでなくアウトプットも少しづつするようになります。できることが増えただけでなく同僚と話して盛り上がるなどコミュニケーションにも影響がありました。

他の恩恵として、アウトプットしている人と少しづつコミュニケーションを取るようになった結果、自分がこうしたアウトプットをする側になれたというのもいい刺激を貰えた効果と言えると思います。

36.5 まとめ

自分の行動のハードルが低い環境を見つけ、そこに学びとなる情報が流れるようにすることで、無理なく自分のペースでスキルアップできる環境ができました。

仕事が忙しいなど精神的な余裕がないときはTwitterを見ないという選択が取れることも、ストレスなく学べることにつながっていると思います。

ChatGPT を使った文章の書き方の一例

おやかた@oyakata2438

37.1 ChatGPT は文書作成の強力なツールである

ChatGPT は、ユーザーが指定したタイトルやキーワードに基づいて、自然でそれらしい文章を自動生成してくれるツールです。いわゆる生成 AI は文章生成のみならず、画像、音楽、音声、動画などなど、様々な分野で劇的な進歩を遂げています。この文章も、骨組みには ChatGPT(4o:無料版)を使って、全体を修正して使ってています。技術的な内容や特定のテーマについて概要を説明したい場合、非常に簡単に情報を引き出すことができます。文章の構造や言い回しも自然で、初めて見るとかなりの精度で文章を仕上げてくれると感じるでしょう。特に、文章を書く際に時間と短縮したい場合やアイデア出しのサポートを求める際には、とても役に立ちます。

ChatGPT を効果的に使うためには、明確な指示やキーワードを与えることがコツです。あいまいな指示では曖昧な結果が返ってくるため、具体的に何を伝えたいのか、どのような構成にしたいのかをしっかりと考えてから入力することが大切です。また、複数の指示を組み合わせたり、テーマを細かく指定したりすることで、より精度の高い文章を生成することができます。

では、万能なのかというと、いくつか問題があります。個人的には、ChatGPT(をはじめとする生成 AI) の問題は二つと考えています。

1. 嘘が混じることがある
2. 数を数えたり計算したり並び替えたりは苦手
3. AI 臭い文章になる

37.1.1 ChatGPT の嘘対策

ChatGPT の回答には嘘が含まれることがある、という話を聞いたことがある人も多いでしょう。実際、とてもそれっぽく、嘘や別の内容が混じることがあります。それっぽいけど関係ない内容が入ったり、実在しない書籍を参考文献として挙げたり。

ですので、ちゃんとそのワードで検索して、大学や研究所、あるいは装置メーカーの記述にあたって裏どりすることが必須です。同人誌だとはいえ、ファクトチェックは必須ですね。

また、実在しない書籍を挙げることも少なくありません。少なくともその書籍名で Amazon で検索して存在するか、発行年や著者が間違っていないか、版数が間違っていないかは確認が必須です。

37.1.2 数を数えたり並び替えたりは苦手

また、純粋に数を数えたりすることは苦手です。単純な足し算ができない、というのが話題になったこともありますね。

xxについて、候補を50個挙げて、といって、50個出てくるとは限りません。また、XXのリストをアルファベット順に並べて、といった指示に対して、ちゃんと2文字目以降がアルファベット順か、抜け落ちたり加わったりしているものはないかという確認は必要です。

37.2 AI臭い文章になる

しかも、ChatGPTを使った文章作成には、いくつか注意が必要です。まず、ChatGPTは学習に用いた非常に多くの(過去の)情報をもとに、関連性の高いキーワードを抽出して文章を生成しますが、不適切な情報や誤った内容が混じる可能性があるという点です。例えば、技術的な専門用語を使ったり、複雑な理論を説明したりする場合、AIが生成した内容が正確ではない場合があります。そのため、生成された文章を鵜呑みにせず、自分の知識と照らし合わせてチェックすることが大切です。ということは、それが自分の知識や専門分野の範囲を超えた内容の場合、そのチェックができないという問題が生じます。

また、ChatGPTが文章を生成する際は、あくまで提示された情報やキーワードに基づいているため、個々のトピック間の関連性が薄くなることもあります。そのため、最終的に文章全体を確認し、論理的な一貫性や構造が整っているかをチェックする必要があります。特に長文になると、話題が散漫になったり、結論にたどり着くまでの展開が冗長になったりするがあるため、注意が必要です。ChatGPTの生成結果は非常に便利な「骨組み」として活用しつつ、最終的な仕上げは自分の手で行うことが重要です。

さらに、ChatGPTは内容の最新性や信頼性に限界があることも覚えておくべきです。AIの知識はその学習時点でのデータに基づいており、最新の情報やトレンドを正確に反映しているとは限りません。そのため、特に技術やビジネスの分野では、最新のデータや知見を反映するために自分で情報を追加・修正することが必要です。例えば、ソフトウェアのバージョンや法改正の情報など、タイムリーな内容を含む場合は、常に確認して必要に応じて更新・修正が求められます。ここも、ファクトチェックという文脈につながりますね。

最後に、ChatGPTが生成する文章は、文法的には正しくとも、時折「AI臭さ」を感じことがあります。AI臭さとは何かを言語化することはなかなか難しいのですが、同じようなフレーズの繰り返しや、文章全体が単調だったり、やたらと説明的であったり・・・何となく不自然な文章の「匂い」です。このような「AI臭さ」を消すためには、生成された文章を自分で手直しする作業が必要です。具体的には、言い回しを変えたり、冗長な表現を削除したり、文体を自分のスタイルに合わせたりすることで、自然な文章に仕上げることができます。内容のチェックと並行して、表現修正でAI臭さを消せれば生成AIを文章作成に便利に使えるでしょう。AIの力と人間の力を組み合わせて、よりよい文章を短時間で仕上げができるといいですね。

ChatGPT の生成した生の文章

修正の参考に、ChatGPT が output した生の文章を 1 段落分掲載します。本章の最終段ですね。この修正が Best であるというつもりはないですが、編集によってもとの文章にある AI 臭さがある程度消せていると嬉しいです。

次に、ChatGPT が生成する文章は、文法的には正しい場合が多いですが、時折「AI 臭さ」と呼ばれる機械的な文章が感じられることがあります。これには、同じようなフレーズの繰り返しや、文章全体がやや型にはまっている感じが含まれます。このような「AI 臭さ」を消すためには、生成された文章を自分で手直しする作業が必要です。具体的には、言い回しを変えたり、冗長な表現を削除したり、文体を自分のスタイルに合わせたりすることで、自然な文章に仕上げることができます。特に、読者が違和感を感じないように、言葉遣いや段落の流れに気を配ることがポイントです。

第 38 章

学びと言語化

@erukiti

「学ぶ」とは何だろうか？「まねぶ」が語源だの、知識を得る行為といった辞書に書いてある事はさておき、学ぶとは本質的には自己変容だ。少し大げさな言い方に聞こえるかもしれないが、我々は物事を知らない状態から、学習を通じて、理解する状態へと変化していく。

「学び」とは何だろうか？学びは、学んで得た知識や経験、そしてその獲得過程全体を指す。学びには二つの形態がある。一つは経験や直感として個人の内部に留まる「暗黙知」だ。自転車の乗り方のように、言葉で説明するのが難しい知識だ。もう一つは、言葉や図表で表現された「形式知」だ。教科書やマニュアルのように、誰でも共有できる形に整理された知識だ。学びは暗黙知として始まり、言語化を経て形式知となる。それによって自分の理解が深まり、他者とも共有できるようになる。これこそが、皆さんが目にする一般的な学びの形だ。

現代は大量の情報が流通していて、その中に多くの言語化された学びがある。ブログや本、同人誌は、それらの典型的な形だ。学びを読者に伝えるための物だ。今この本を手に取っている読者は、まさに言語化された学びを読んでいて、ここから何かを学んで自己変容を遂げようとしている。あるいはもう既に自己変容した後だろうか。

この奥深い学びの本質に迫るために、まず避けて通れないのが「言語化」という要素だ。なぜなら、言語化は学びを深め、定着させ、共有可能にする核心的なプロセスだからだ。

これから数章にかけて、筆者なりの学びについての考察をお伝えしていきたい。

それでは、この学びにおいて最も重要な言語化について、具体的に掘り下げていこう。

38.1 言語化とは何か

言語化について、まずはシンプルな座学のケースをもとに学びのプロセスを考えよう。

我々は何かを学ぶときに、まず情報をインプットする。これは読書かもしれないし、ウェブ検索で見つけたページや、講義の受講かもしれない。他者が言語化したそれを読むなり、聞くなりする。

次に、その情報を自己の中でかみ砕いて整理する過程で、より意識的な言語化が行われる。「なるほど、これはこういう意味を持つのか」「これには共通点があるな」といった具合に、インプットしたものからの気づきを言葉にする。そのとき、既存の知識と結びつけることで、より深い理解が生まれる。

最後に、理解した内容を言語化してアウトプットすることで、学びはより確かなものとなる。これは口頭で説明することもあれば、文章として書き記すこともある。人に伝えるためには分かるように説明しなければならないし、分かりづらい説明をすれば質問が飛んでくることもある。テスト

勉強で、他人に教えると理解が深まる現象を考えれば分かりやすいだろう。

では、このプロセスを踏まえて、言語化の本質について考えてみよう。

言語化とは、単なる言葉への置き換えではない。我々は日常的に言語を用いて思考しているが、それと意識的な言語化には本質的な違いがある。その違いは主に三つの点に現れる。

第一に、構造化の度合いが挙げられる。我々の日常的な思考では、言語だけでなく、イメージや感覚、直感的な理解など、様々な要素が未整理のまま混在している。「なんとなくそう思う」「どことなく良い感じがする」といった曖昧な認識が大半を占め、それらの論理的な関係性も曖昧なままとなっている。

第二に、意識的な分析の有無だ。日常的な言語使用が無意識的・自動的なのに対し、言語化は意図的な分析のプロセスを必要とする。例えば「なんとなく良い」と感じている商品があったとする。これを言語化する際には、その感覚の背景にある要素を一つ一つ検討し、「価格が3000円と手頃で、デザインが主張しそうで落ち着いており、必要な機能が過不足なく揃っている」といった具体的な要因に分解していく。

第三に、検証可能性の違いがある。漠然とした印象は主観的で検証が難しいが、言語化された内容は他者による確認や批判が可能だ。「コストパフォーマンスが優れている」という結論も、具体的な根拠があってこそ意味を持つ。現代ならこの検証のプロセスをChatGPTに任せることもできる。

このように言語化とは、漠然とした思考を、意識的に整理・構造化し、他者と共有可能な形に表現するプロセスだ。

38.2 言語化の重要性：プログラミングの例から

プログラミングの世界には、半年後の自分は別人だと思えという格言がある。半年後の自分は文脈を忘れた状態で、自分が書いたソースコードを、まるで他人が書いたもののように読むことが多いからだ。これは言語化の重要性を端的に示す例だ。

プログラミングとはソースコードを書いて、それをプログラムとして動かすものだ。ソースコードはコンピュータ専用の特殊な言語であって、人間にとてわかりやすい言葉ではない。ソースコードは、難解な指示書のようなものだ。

指示書からは、書いた当時の経験、思想をもつて書かれたものでは失われやすい。なぜこういう形の指示にしたのか？なぜこういう形の指示にしなかったのか？選択肢が複数あるとき、どうしてこれを選んだのか？当時はおそらく根拠があったはずだ。経験、知識、意思決定の基準をもつたのだ。これらは単なる指示書からは読み取りづらいものだ。だからこそ人間が読み書きしやすい言葉で、失われやすい情報を書き記しておくべきだ。

コメントの例

```
// このアルゴリズムを選んだ理由：  
// * データサイズが小さいため、メモリ効率より可読性を重視  
// * 将来の機能拡張を考慮し、モジュール化を優先  
// * チームでの保守性を考慮し、標準的な実装方法を採用
```

もちろんこれは、プログラミングに限った話ではなく、学びにおいても同様だ。我々は学んだ内容を十分に理解したつもりでも、時間が経つと詳細を忘れてしまう。その時の文脈や、重要だと考えた背景も、徐々に失われていく。

しかし、学びを言語化し文書として残しておけば、それらの情報は保持できる。未来の自分がその文書を読み返すことで、当時の学びを再現できる。検索可能な形であれば、必要な時に必要な情報を素早く取り出すこともできる。

それでは次に、言語化を実際にやってみよう。

38.3 言語化をやってみよう

言語化の練習方法には様々なものがあるが、最も基本的なアプローチは、筆者が「思考の垂れ流し」と呼ぶ方法である。これは、アイデアの整理や推敲を気にせず、思いついたことを文字として書き記していく作業だ。

キーボードで入力するにせよ、鉛筆やペンでノートに書くにせよ、重要なのは思考を明示的に外部化することだ。文字として外に追い出することで、思考の整理が促進される。他人に見せないことを前提とすれば、より自由な表現が可能となる。

合同誌で、学び本だすことになったけど、そもそも何を書けばいいのかな。学びについては何かしらあるはずだから僕に書けることはあるはずだけど、読者は何を読みたいのだろうか。新しい技術の習得は好きだし、そういうことを書けばいいのだろうか。ReactとかLLMのことを書いてしまうとそれを知らない読者を置いてけぼりにするよな。であれば、汎用的な話を書いた方がいい気がする。汎用的なぶん他の人も同じ事を書いちゃう？まあそれはあとで考えればいいか。まずは雑に検索からはじめてみよう

このような思考の垂れ流しは、まだ整理されていない生の思考の記録だ。次のステップとして、これを構造化された形に整理していく。箇条書きは、思考を構造化する上で特に効果的なツールである。

- 何を書くか?
 - 読者が読みたいもの
 - 新しい技術の習得
 - * 詳細に踏み込むと読者を置いてけぼりにしてしまう
 - * 汎用的なことを書く
 - 他の人と被ることは気にしない

この例からわかるように、最初の散漫な思考が、箇条書きによって明確な構造を持った形に整理されている。慣れていない人は、思考の垂れ流しと、それを箇条書きにまとめる作業は分けて行う

ことをお勧めする。

さらに、図や表の活用も効果的な手法である。文章は本質的に一次元の構造だが、図表であれば二次元的な思考が可能となり、要素間の関係性をより直感的に把握できる。

パソコン上でのマインドマップや付箋ツールの利用、あるいは実際の付箋を使用した手書きの方法など、自分に合った形式を見つけることが重要である。

■コラム：筆者は言語化が苦手だった

筆者は、雑誌に記事を書いたり、同人誌を書いたり、商業化したり、ブログがそれなりの頻度でバズることがある、言ってしまえば文章を書くセミプロのようなことをしている。仕事においても言語化能力をフル活用している。

ところが昔の筆者は言語化がとても苦手だった。言葉にできないもどかしさを抱えていた。人に伝わらない。人に言いたいはずのことを言えない。やり込められる。納得がいかない。

そこで始めたのがブログだった。しょうもないことをつらつら書き連ねるブログだ。場所を変え、アカウントを変え、文章を書いてきた。

本格的に文章を書くようになったのは2017年に技術同人誌を出してからだ。この頃には文章を読み書きするのにも慣れて、ブログがバズるようになった。

元々は、思考の垂れ流しという手法も、言語化の練習の一つだった。それは今でも活用している。本文中にも書いてたよう第二の思考領域として活用することで、頭をすっきりできるからだ。

そうやって意識して鍛えてきた結果、言語化は様々な場面で使えるスキルに成長した。

38.4 言語化の困難さと限界

言語化の利点を述べてきたが、このプロセスには本質的な困難さが伴う。特に、複雑な感情や直感的な理解の言語化には大きな課題が存在する。

たとえば、音楽の感動を言語化する場合を考えてみよう。「この曲は心に響く」という感覚を、具体的な言葉で説明しようとしても、その本質を完全に捉えることは難しい。「印象的なメロディーライン」や「心地よいリズム」といった表現を用いても、その感動の全体を言い表すことはできない。

専門的な内容を初学者に説明する際の言語化も、高度なスキルを要する。例えば、プログラミングの非同期処理について、プログラミングの知識がない人に説明することは極めて困難である。料理の手順のような直線的な処理とは異なる実行の流れを、どのように言葉で表現すればよいのだろうか。

さらに、すべての学びが言語化できるわけではない。自転車の乗り方のような身体的なスキルは、言葉だけでは十分に伝えることができない。「バランスを取る」という指示だけでは、実際の

感覚を掴むことは困難である。料理人が「火加減は経験で覚えるしかない」と言うのも、この類の学びの特質を示している。

職人の技術も同様である。熟練した大工が「この木は柱に適している」と判断する際、その判断基準には木目の細かさ、色味、重さ、手触りなど、多様な要素が複雑に絡み合っている。その判断は確かな精度を持つが、そのプロセス全体を言語化することは容易ではない。しかし、このような困難さがあるからこそ、言語化への挑戦は重要な意味を持つ。言語化の限界を認識しつつ、可能な範囲で言語化を試みることで、理解は深まり、他者との共有も可能となる。例えば、身体的な学びであっても、野球選手のフォームを物理学的に分析したり、料理の火加減を温度計で数値化したりすることで、部分的な言語化は可能となる。完璧な言語化は難しくとも、その過程での気づきが学びをより確かなものとするのである。

38.5 学びのコミュニティにおける言語化の役割

言語化の意義は、個人の学びを超えて、コミュニティ全体の知識発展にまで及ぶ。技術書典やミケのような同人誌即売会、テックブログ、プログラミングのQAサイトなどでは、個人の学びが言語化され、共有されることで、コミュニティ全体の知識が豊かになっていく。本書もまた、そのような知識共有の一環として位置づけられる。

個人の言語化された学びは、他者の新たな学びのきっかけとなり、その連鎖は知識の螺旋的な発展をもたらす。このプロセスは、特にオープンソースソフトウェアの開発コミュニティにおいて顕著に観察される。

38.6 言語化のさらなる練習方法：ブログを始めよう

思考の垂れ流しと箇条書きによって、自分の考えを整理する方法は説明した。次はそれを誰かに伝えてみよう。ブログは、言語化の練習にとても向いている。

- 匿名での開始が可能
- 読者からのフィードバックが期待できる
- 実践的な言語化の機会となる

もしあなたがブログを書き慣れていないなら、是非ブログを書き始めるべきだ。まずは、ブログサービスでアカウントを適当に作ってみよう。練習なのだから、失敗してもいいように、自分の本名やよく使う名前を避けて、新規の名前で初めてみるといい。

ブログを書き慣れていないなら「しづかなインターネット^{*1}」は日々思ったことを言語化するのに最適な場所だ。バズりづらいが、名前のとおり、しづかで荒れにくい。

note^{*2}はバズりやすく活発なので、慣れてきたら力試しができる。ソフトウェア技術者なら

^{*1} <https://suzu.me/>

^{*2} <https://note.com/>

Qiita^{*3}や Zenn^{*4}も良いだろう。

学びという題材からは少しだけズレるかもしれないが、小説のような形であれば、小説家になろう^{*5}・カクヨム^{*6}がいいだろう。学びを小説というかたちで伝えられるかもしれない。人類は物語を介して様々な学びを伝えてきた歴史を持っている。

- 日々の言葉を練習するなら、しづかなインターネットが一番おすすめできる
- note は活発だし、書きやすいので、力試しに向いている
- ソフトウェア技術者なら、Qiita か Zenn が良い。ソフトウェア技術を伝えるなら note よりも望ましい
- 小説なら、小説家になろう・カクヨムがいいだろう

アカウントを作ったらブログを書くことそのものに慣れよう。自分の思いや得たものを何でもいいので書いて、伝える練習から始めればいい。書けば伝わるとは限らないし、読んだ人が誤った解釈をしてしまうかもしれないが、匿名で始めることで恥ずかしさや失敗に怯えなくて済む。

慣れてきたら、学びをブログに書いてみよう。たとえばソフトウェア技術者なら、自分が使っている技術について、しょうもないことでもいいから学びをブログに書けばいい。手順を書いてみて、成功したこと・失敗したこと、考えたこと・感じたこと、難しかったこと・簡単だったことを書いてみよう。そこに、何かしらの学びがあれば、それを書いてみよう。

これを何回も繰り返せばいい。自分が書きたいものはこれではないと思ったら、新しいアカウントを作り直せばいい。

- まずは、しづかなインターネットで、日々の言葉を書く練習をする
- note や Qiita や Zenn で力試しをする
- 書きたいものがこれではないと思ったら、新しいアカウントを作り直す

こういった目的にはSNSもありだ。Twitterは元々マイクロブログと呼ばれていた。最初は140文字に限定されたブログだったのだ。

あなたが何かしらのエンタメが好きなら、そのエンタメの感想や考察をつぶやいてみるといい。このときもブログと同じように新規アカウントを作ることをおすすめする。メインの人格でやってしまうと、取り返しがつかないミスをしてしまうこともあります。それでも安心して言語化の練習ができるだろう^{*7}。

うまくいくことがわかれば、改めて自分のアイデンティティとなる名前で活動すればいい。

^{*3} <https://qiita.com/>

^{*4} <https://zenn.dev/>

^{*5} <https://syosetu.com/>

^{*6} <https://kakuyomu.jp/>

^{*7}もちろん匿名性を過信して、誹謗中傷など犯罪行為を行えばその限りではない。

38.7 まとめ

学びと言語化は不可分の関係にある。思考を言語化することで理解は深まり、その学びは持続可能なものとなり、さらには他者との共有も可能となる。すべての学びを完全に言語化することは困難だが、だからこそ、その試みには意義がある。

デジタル技術の発展は、言語化のための手段を多様化させている。これらを効果的に活用し、定期的な言語化を習慣とすることで、より深い学びが可能となる。

そして最も重要なのは、言語化された学びの共有が、コミュニティ全体の知識発展に寄与するという点である。我々は学ぶだけでなく、その学びを言語化し、共有していくことで、より豊かな知的環境を築いていけるのである。

学びのプロセスを詳細に分解する

@erukiti

前章では、学びと切り離せない言語化について説明してきた。言語化によって、漠然とした理解を明確なものにし、他者と共有可能な形にできることを見てきたが、言語化は学びのプロセス全体から見れば、その一部に過ぎない。

多くの人は「目についた本を読む」「目についた動画を見る」「実践してみる」といった行動を思いつくままに行っている。これは間違いではないが、効率的な方法とは言えない。本章は、こうした学びのプロセスを詳細に分解し、各段階で何が起きているのかを明らかにする。

39.1 学びのサイクルを理解する

学びは一瞬で完結するものではなく、複数の段階が循環的に繋がったプロセスだ。このサイクルには以下のような要素がある。

- 現状と目標の明確化
 - 今の自分の立ち位置を理解する
 - 具体的で現実的な到達点を設定する
- 動機づけの確立
 - 外的動機（試験、資格など）を把握する
 - 内的動機（知的好奇心、競争心など）を理解する
 - 両者のバランスを取る
- 実践方法の選択
 - 直接的アプローチ（手を動かす）を計画する
 - 間接的アプローチ（本を読む、観察する）を組み合わせる
 - 進捗の確認方法を決める
- フィードバックの活用
 - 即時フィードバックを得る
 - 長期的フィードバックを収集する
 - 改善点を次のサイクルに活かす

このプロセスを理解することで、自分の学び方の課題が見えてくる。そしてその課題に対して適切なアプローチを選択できるようになる。

39.1.1 現状と目標の明確化から始める

学びの第一歩は、現状と目標を明確にすることだ。これは単なる目標設定以上の意味を持つ。現在の自分がどこにいて、どこに行きたいのかを具体的に言語化する過程だ。漠然とした「できるようになりたい」という願望を、具体的な到達点へと変換していく作業だ。

39.1.2 動機づけの本質を理解する

学びのサイクルを継続的に回すためには、強い動機づけが不可欠だ。これは単なる「やる気」以上の、自分自身の本質的な動機を理解することを意味する。動機づけは学びの質と持続性を大きく左右する要素であり、その理解と活用は効果的な学びの鍵となる。

人間の行動を動かす動機には、大きく分けて二つの種類がある。外的動機と内的動機だ。外的動機は、学校での授業や資格試験のように、外部からもたらされるものだ。明確な目標と期限を持つため、学びの強い推進力となる。一方、内的動機は個人の内側から湧き出るものだ。知的好奇心、競争心、創作意欲など、その形は人によって様々だ。

これらの動機は、学びの質に大きな影響を与える。外的動機は明確な目標を持つため、短期的には効果的だ。試験が終わったり、目標を達成したりすると、その原動力は失われやすい。対して内的動機は、自発的な興味や関心に基づくため、より持続的な学びを可能にする。理想的には、両者をうまく組み合わせることで、より効果的な学びが実現できる。

学びにおいて、受動的な姿勢と能動的な姿勢では、その効果に大きな違いが生じる。受動的に学ぶ人は、能動的に学ぶ人に比べて効率が落ちることが多い。これは「本気になれない」「自分事にできない」という状態につながりやすいためだ。

■コラム：勉強嫌いから学びの探求者へ

筆者の子どもの頃は、親や教師に強制される「勉強」がとても嫌いだった。他人から「勉めるを強いられる」のがとても嫌いでそれは今も変わっていない。しかし、知的好奇心に基づく自発的な学びは好きだ。

筆者にとっては知的好奇心が強力な内的動機になっている。それはソフトウェア技術にしても、ゲームにしても、雑学にてもいえる。Wikipedia めぐりをしてたら1時間以上経ってたということもざらだ。気になることがあれば調べないと気が済まない。知的好奇心は、学びの効率どころか自分のキャリア形成に影響を与えたほどだ。

そういう意味では、能動的な姿勢でしか筆者は本領を發揮できないタイプなのかもしれない。

重要なのは、自分にとっての本質的な動機を見つけることだ。それは他人と同じである必要はない。自分なりの「学ぶ理由」を持つことで、学びはより深く、より持続的なものとなる。

39.1.3 効果的な実践方法を選択する

実践は学びの中核を成す要素だ。ただ漫然と取り組むのではなく、効果的な方法を選択する必要がある。実践には直接的なアプローチと間接的なアプローチがある。

直接的なアプローチとは、実際に手を動かして学ぶことだ。プログラミングであればコードを書き、デバッグし、動作を確認する。ゲームであれば実際に対戦を重ね、新しい技を試す。これは学びの核となる活動だ。

一方、間接的なアプローチも重要な役割を果たす。技術書を読む、他人のコードを分析する、上手い人のプレイを観察する。これらの活動は、直接的な実践を支え、より深い理解へと導く。

39.1.4 フィードバックを活用する

フィードバックは、学びを確実なものにする重要な要素だ。これは単なる結果の確認ではない。学びの過程全体を見直し、改善につなげるプロセスとして捉える必要がある。

フィードバックには、即時フィードバックと長期的フィードバックがある。プログラミングを例にとると、即時フィードバックは実行時エラーやコンパイルエラーだ。コードを書いて実行すると、すぐに結果が分かる。一方、コードレビューや性能測定は長期的フィードバックとなる。時間はかかるが、より本質的な改善につながることが多い。

この過程で重要なのは、フィードバックを単なる評価として受け取るのではなく、次の学びのステップを見出すきっかけとして活用することだ。コードレビューで指摘された問題点は、新たな学びのテーマとなる。対戦ゲームで負けた原因を分析することは、次の練習の方向性を示唆する。

39.2 メタ学び：学び方を学ぶ

学びの過程で最も重要なのは、個別の知識やスキルの習得以上に、「学び方を学ぶ」ことだ。これをメタ学びと呼ぶ。メタ学びは、自分の学びのプロセスを客観的に観察し、改善していく営みだ。

メタ学びの核心は、効率の最適化にある。これは単なる時間管理以上の意味を持つ。新しい技術を学ぶとき、どの程度の深さまで理解する必要があるのか、どのような順序で学ぶのが効果的かを判断する能力が求められる。すべてを完璧に理解しようとするのは非効率的かもしれない。逆に、表面的な理解だけでは不十分な場合もある。

39.3 例をもとに学びのプロセスを理解する

学びのサイクルの理論的な説明に続いて、実際の学びの場面での適用を見ていく。具体的な例を通じて、学びのプロセスを掘り下げる。

39.3.1 Python を学ぶプロセス

プログラミング言語の学びは、学びのサイクルを理解する良い例となる。「Pythonができるようになりたい」という漠然とした目標は、具体的な段階に分解する必要がある。

まず現状を正確に把握する。プログラミング未経験なのか、他の言語の経験があるのか。次に、目標を具体化する。「基本文法が分かって簡単なプログラムが書ける」「データの流れを理解してライブラリを使って基本的なデータ処理ができる」といった形だ。

実践においては、ただ文法書を読むだけでなく、具体的な課題に取り組むことが効果的だ。簡単なゲームを作る、データ分析を試みる、など。そしてエラーとの戦いも重要な学びの機会となる。エラーメッセージを理解し、解決策を見つけ出す過程で、より深い理解が得られる。

39.3.2 ゲーム上達のプロセス

対戦ゲームの上達は、学びのプロセスを別の角度から示す好例だ。FPS ゲームを例にとると、上達に必要な要素は多岐にわたる。エイム力、マップ知識、状況判断力、チームワーク。これらは一度に習得するのは困難だ。

エイムの練習は専用の練習モードで、マップ知識はオンラインでの探索で、状況判断力は上手いプレイヤーの配信を見て学ぶ。このように要素を分解し、それぞれに適した練習方法を選択することで、効率的な上達が可能となる。

フィードバックも重要な要素だ。リプレイを見直し、失敗の原因を分析する。チームメイトからのアドバイスを受ける。これらの情報を次の練習に活かすことで、着実な進歩が得られる。

39.4 学びの共有と発信：知識の循環

学びは個人の中で完結するものではない。それを共有し、発信することで、さらなる深まりを見せる。これは前章で説明した言語化の発展形といえる。

技術書やブログの執筆、登壇などの活動は、自分の理解を他者と共有可能な形に整理する機会となる。この過程で、自分の理解の不十分な部分が明らかになり、新たな学びが生まれる。

特に重要なのは、「分かる」と「伝わる」の違いを実感できることだ。自分が理解していることと、他者に伝わることには大きなギャップがある。このギャップを埋めようとする過程で、より深い理解が得られる。

技術書の執筆を例にとると、読者の視点に立って説明を組み立てる必要がある。何を説明し、何を説明しないか。どのような順序で説明するか。これらの判断には、自身の知識を体系的に整理する作業が伴う。

プレゼンテーションでは、その場の空気を読み、聴衆の反応を見ながら、リアルタイムで説明を調整する必要がある。これは、知識を柔軟に再構成する能力を養う。

このように、学びの共有は単なる情報発信以上の意味を持つ。それは個人の理解を深め、コミュニティの知識を豊かにし、さらなる学びを生み出すサイクルを形成する。

39.5 まとめ

学びのサイクルは、目標設定から実践、フィードバック、知識の統合まで、複数の段階から成る継続的なプロセスだ。このサイクルは固定的なものではなく、内容や個人の特性に応じて柔軟に調整していく必要がある。

個々の知識やスキルの習得はもちろん大切だが、「学び方を学ぶ」こと、つまりメタ学びの視点が不可欠だ。これは単なる効率化のテクニックではない。自分の学び方を理解し、改善していく営みそのものが、生涯学びの基盤となる。

個人の成長だけでなく、社会全体の知的発展にも寄与する。そこに本質的な意味での学びの実現があるのだ。

第 40 章

学び hack

@erukiti

学びの hack とは、学びのプロセスを理解し、それを自分にとって最適化する技術だ。前章で見てきた学びのサイクルをより効率的に、より持続的に回すための実践的なアプローチを探る。目の前の課題を分解し、段階的に理解を深め、経験を重ねながら成長していく、そのための具体的な方法を見ていく。

40.1 学びの複雑さを解きほぐす

何かを学ぼうとするとき、最初の一歩として複雑さを解きほぐすことが重要だ。全く何も分かっていない状態では、その学びの難しさや分量、得られるものすら把握できない。プログラミング未経験の人が Python を覚えたい、という状況がこれに当たる。概略さえつかめば、どれくらいのスケール感なのか、どの程度の時間や労力が必要なのかが見えてくる。

現代では、概略を掴むためのツールが豊富に存在する。入門書の目次を眺めたり、ChatGPT や Claude に概要を説明してもらったりすることができる。Google の NotebookLM のような特化型 AI を使えば、PDF などの資料を基に質問ができる、より正確な情報を得ることもできる。

概略を掴んだ後は、その全体像を適切なサイズに分割していく。Python の学習なら「文法」「開発環境」「ライブラリ」といった要素に分解できる。さらに文法であれば「変数」「制御構文」「関数」というように細分化していく。音楽の学習でも同様の手法が有効だ。「楽器の扱い方」「基本的な演奏技術」「楽典」「リズム感」といった要素に分解し、それぞれを段階的に学んでいく。

■コラム：細分化は色々応用が利く

物事を細分化して捉えるというやり方は、社会人として仕事をやっていく上で必須となる。

たとえば目標設定がある。会社全体の目標があって、部門ごとの目標に分解され、それがさらにチームごとの目標になって、個人の目標になるという会社が多い。目標に関しても半年とか 3 ヶ月とか単位で区切られ、その目標は、さらに細かいものを積み上げていく。

日々の仕事もこういった大小の積み重ねだ。ソフトウェア技術者ならある機能を実装する。その機能の実装には 5 日かかる。その場合、その機能の実装はさらに細かい作業に分けられるはずだ。1 日単位の作業。そして 1 日単位というのですら大きすぎる。その 1 日かかるはずの作業は、さらに細かい単位にまで落とし込める。ソフトウェア技術者の考え方には、分割統治法というものがある。複雑な問題は、小さな問題の組み合わせに変えてみましょうというも

のだ。

大きなもの、多いものには、認知負荷が存在する。それ全体を脳で一瞬で把握できるだろうか？出来ないからこそ時間がかかるのではないか？ならば脳が認知しやすい単位まで小さくすればいい。難しいものに取り組んでいる人も多くの場合は、こういった小さな問題に分割することに慣れているからそれができるのだ。

多方面同時作戦、物量攻撃、といったものから解放されると、難易度は下がるものだ。

この分割のプロセスそのものが、すでに学びの第一歩となっている。分割することで、各要素の相互関係や全体における位置づけが見えてくる。これは次のステップである「あたりをつける」作業の基礎となる。

40.2 あたりをつけて進める

未知の領域に挑戦するとき、最初は完全な正解を求める必要はない。むしろ、あえて大雑把な推測、つまり「あたり」をつけることが重要だ。あたりは、最初は完全に的外れな場合もある。しかし、それは問題ではない。

あたりは一種の基準点となり、そこからの学びによって理解がどう変化したのかが明確になる。たとえば新しい技術について「これはデータベースに似た仕組みだ」と考えたとする。学びが進むにつれ、その推測が正しかったのか、あるいはどう違っていたのかが明確になる。何も考えずに始めるよりも、あたりとの差分を認識する方が学びの効果は高い。

この過程を繰り返すことで、次第に精度の高いあたりがつけられるようになる。そうなると、新しい未知の領域に出会ったときも、完全な未知ではなく、ある程度の見当をつけられるようになる。

認知科学では、人間が新しい概念を理解する際に、既知の概念との類似性を見出すことが効果的だと指摘されている。これを「アナロジー（類推）」と呼び、学習において重要な役割を果たす。

料理の学習では、新しいレシピに挑戦する際に既知の調理法との類似点を探ることが有効だ。「この調理法は炒め物に似ている」「この味付けは以前作った料理と似ている」という具合に、既存の知識を足がかりにして理解を深めていける。

プログラミングにおける「変数」という概念も、初学者にとって理解が難しい場合がある。しかし、これを「名前付きの箱」というアナロジーで説明すると、多くの人がすんなりと理解できる。この箱には様々な値を入れることができ、中身を取り出したり更新したりできるというイメージだ。

ただし、アナロジーはあくまでも入り口であり、本質的な理解に到達するための足がかりとして捉えるべきだ。最終的には、その技術や概念が持つ本来の特性や仕組みを理解する必要がある。

■コラム：アナロジーの罠

個人的にはアナロジーは諸刃の剣になりやすくあまり好きではない。オブジェクト指向にまつわるアナロジーなんかは特に混乱が多いと感じる。というか、オブジェクト指向を始めて聞いたときのアナロジーでその後混乱がかなり続いた。

この経験から、アナロジーは理解の入り口として使いつつ、その限界も同時に示すことの重要性を学んだ。

40.3 経験と本質の両立

学びにおいて、経験と本質的な理解は両輪となる。経験だけでは表面的な理解に留まりがちで、逆に理論だけでは実践力が身につかない。この二つをバランスよく組み合わせることで、より深い理解と確かな実践力を獲得できる。

経験から学ぶことと、本質を理解することは、必ずしも対立するものではない。むしろ、両者は補完し合う関係にある。経験を積むことで本質への理解が深まり、本質を理解することで経験がより意味のあるものになる。

料理の腕を上げるには、レシピ通りに作る経験と、調理の科学的な理解の両方が必要だ。火加減の感覚は実践でしか掴めないが、なぜその温度が重要なのかを理解することで、より確実な調理が可能になる。

アルゴリズムの学習でも同様だ。まず実装して動かしてみて具体的なイメージを掴み、その後で計算量などの理論的な理解を深めていく、というアプローチが効果的だ。実践と理論を往復することで、より深い理解に到達できる。

プログラミング言語の学習では、最初は「とりあえず動くコード」を書くことから始めて、徐々にその言語の特徴や思想、背景にある考え方を学んでいく。この過程で、なぜその言語がそのような設計になっているのかを理解することができる。

40.4 経験から学びを深める

経験からの学びは、体系的な知識の獲得とは異なる独自の価値を持つ。「賢者は歴史から学び、愚者は経験から学ぶ」という言葉があるように、時として経験による学びは非効率だと考えられがちだ。確かに、人類の積み重ねてきた知見を軽視するべきではない。しかし、すべてを机上の学習だけで身につけることはできない。特に実践的なスキルは、実際に手を動かすことで初めて身についていく。

経験から効果的に学ぶためには、計画的なアプローチが必要だ。その核心は試行回数の最大化にある。多くの試行を可能にするには、一回あたりの試行にかかるコストとリスクを適切に管理する必要がある。

コストには様々な種類がある。時間的コスト、金銭的コスト、心理的コストなどだ。料理の練習であれば、少量から作り始めて材料費を抑えられる。楽器の練習なら、防音室を借りるのではなく消音機器を使うことで、場所や時間の制約を減らせる。プログラミングの学習では、AIチャットボットを活用することで素早くフィードバックを得られ、時間的コストを大きく削減できる。

リスクに関しては、完全になくすることは難しい。しかし、適切な環境を用意することで、リスクを軽減できる。料理なら練習用の包丁を使う、楽器なら初心者向けの機材を選ぶ、プログラミングならソースコード管理システムを使って変更履歴を残すなど、失敗しても影響を最小限に抑える工夫が可能だ。コンテナ型の仮想化技術を使えば、システム全体を安全に試行錯誤できる環境も作れる。

40.5 モチベーションを持続させる

難易度を下げるることは、成功体験を得やすくすることにもつながる。小さな目標を達成していくことで、「できた」という実感が得られる。その積み重ねが、より大きな課題に取り組むモチベーションになっていく。

プログラミング学習では、まずは「Hello, World!」を表示するところから始める。次に簡単な計算、そして条件分岐といったように、段階的に難しさを上げていく。この過程で得られる小さな成功体験が、次の学習への原動力となる。

学びの原動力として、好奇心ほど強力なものはない。「これを変えたらどうなるだろう」「なぜこうなるのだろう」という素朴な疑問から、大きな発見が生まれることも多い。

プログラミングの学習では「写経」という方法がある。写経とは、本に書いてあるソースコードを、実際にタイピングして動かす学習方法のことだ。写経の効用には賛否両論あるが、そのコードを元に積極的に好奇心を発露するなら有効な手段となる。

40.6 実践的な学びの例

ウェブフロントエンドは、好奇心に基づいて学びやすい分野だ。有名なウェブサイトの構造を分析し、真似る練習から始めることができる。ウェブページを構築する基本技術であるHTML/CSSをどう書けば、有名なウェブサイトと同じように見えるようになるか、試行錯誤を重ねる。手元で練習する分には、単なるコピーから始めて構わない。そこから何を削ったら崩れるか、何を書き換えればどう変わるかを、思いつく限り試すことで理解が深まる。

40.7 失敗からの学び

難易度を下げるということは、必ずしも失敗しないようにすることではない。むしろ、失敗したときの影響を小さくすることだ。小さな単位で取り組むことで、失敗してもその原因を特定しやすくなり、修正も容易になる。

失敗はむしろ学びのチャンスだ。失敗を恐れすぎると、チャレンジすることすらできなくなってしまう。適切な難易度設定により、失敗を恐れない環境を作ることが、持続的な学びには必要だ。失敗からの学びを確実にするためには、以下の点に注意を払う必要がある。

- 失敗の原因を具体的に特定する
- 次回どうすれば防げるかを考える
- 失敗の経験を記録として残す

40.8 コミュニティでの学び

一人で学ぶことには限界がある。学習コミュニティに参加することで、モチベーションの維持や知識の共有が容易になる。

オンラインコミュニティは、時間や場所の制約なく参加できる利点がある。Slack や Discord のような場所が活発なコミュニティであれば、参加することで学びが加速する。質問をしたり、他者の質問に答えたりすることで、自分の理解も深まっていく。

プログラミングの世界では、2人で1つのコードを書くペアプログラミングや、チームで1つのコードを書くモブプログラミングといった共同作業も、学びを加速させる効果的な方法だ。経験者と一緒にコードを書くことで、普段は気付かない視点や効率的な手法を学ぶことができる。

40.9 振り返りの重要性

モチベーションを保ちやすい振り返り手法として、Fun! Done! Learn!がある。これは以下のようないくつかの観点で日々の経験を振り返る手法だ。

- Fun!はその日楽しかったこと、喜びを感じたこと
- Done!は実際に達成できたこと、小さな成功体験
- Learn!は新しく学んだこと、気づいたこと

このフレームワークは、ポジティブな感情を大切にしながら、達成と学びを整理できる点が特徴だ。

40.10 まとめ

学びの hack とは、学びのプロセスを理解し最適化することだ。本章ではその具体的な方法として、複雑な課題を細分化し、あたりをつけながら進み、経験と本質的な理解を両立させていくアプローチを見てきた。失敗を恐れず、むしろそこからの学びを重視する姿勢も欠かせない。適切な難易度設定により、失敗のコストを下げ、むしろそれを次への学びのチャンスとして活用していく。そして何より、好奇心を原動力とした持続可能な学びの環境を作ることが重要だ。それぞれが自分なりにこれらの技術をカスタマイズし、自分に合った形で活用していくことで、より効果的な学びが実現できる。

自己変容を受け入れる

@erukiti

第 38 章「学びと言語化」にて、学ぶとは自己変容であると書いたが、ここまで自己変容については触れてこなかった。この章では、学びにおいて意外と重要な自己変容について書いてみよう。

41.1 学びにおける自己変容

学びによる自己変容とは、単なる知識の蓄積以上のものだ。それは、物事の見方や考え方、さらには自分自身に対する理解までもが変化していく過程を指す。時として、それは人生の大きな転換点ともなりうる。

プログラミングを例に考えてみよう。最初は「プログラミングは難しい」と感じていた人が、学びを重ねることで「プログラミングは工夫次第で実現できる」という考え方へ変わっていく。これは単にプログラミングの技術を習得しただけでなく、問題に対するアプローチの仕方自体が変容したことを意味する。

自己変容は、主に四つの側面で起こる。

第一には認知的枠組みの更新だ。新しい情報や視点に触れることで、これまで持っていた思考の枠組みが更新される。「エラーは失敗の証」という考え方から、「エラーは次のステップへのヒント」という見方への転換は、その典型例だ。

第二には価値観や態度の変容だ。「完璧なコードを書かなければ」という強迫的な考え方から、「まず動くものを作り、それを改善していく」という現実的な姿勢への転換などが、これにあたる。これは単なる技術的な進歩ではなく、ものづくりに対する根本的な態度の変化を示している。

第三には自己認識の深化だ。学びを通じて「自分は何を知らないのか」「なぜこのように考えるのか」といった問い合わせを行うことで、自分自身への理解が深まっていく。たとえば、「自分は細部にこだわりすぎる傾向がある」という気づきは、より効率的な学習方法の選択につながる。

第四には日々の行動様式の変化だ。問題に直面したとき、以前なら諦めていたかもしれないが、「きっと解決方法がある」と信じて調査を続けられるようになる。エラーメッセージを見ても慌てなくなり、むしろ「次に何をすべきか」を考える手がかりとして捉えられるようになる。

41.2 自己変容のプロセスと段階

自己変容は一朝一夕には起こらない。それは段階的なプロセスとして進行する。最初の段階は「気づき」だ。自分の現状や限界に気づき、変化の必要性を認識する。新しい技術に触れたとき、

「このままでは時代に取り残される」という危機感を覚えるのは、この段階の典型だ。

次に「準備」の段階がある。新しい知識や技術を学び、変化に向けて準備を整える。この段階では、具体的な目標を設定し、必要なリソースを集め。新しいプログラミング言語を学ぶと決めたら、まずは基本的な文法や開発環境の整備から始める。

そして「実践」の段階で実際の変化が起こる。学んだことを実際に試し、失敗と成功を繰り返しながら、新しい自分を形作っていく。最初は不安定で、以前の方法に戻りたくなることもある。しかし、一つ一つの経験が、新しい習慣や考え方を形成していく。

最後に「定着」の段階で新しい自分が形成される。変化が自然なものとなり、新しい考え方や行動が無意識のうちに出てくるようになる。たとえば、問題解決のアプローチが、以前とは全く異なる形で自然に行えるようになる。

このプロセスをより効果的に進めるには、内省的な態度が欠かせない。自分の思考や行動を客観的に観察し、「なぜそう考えるのか」「なぜそう行動するのか」を問い合わせ続ける姿勢が必要だ。ソフトウェア開発で言えば、コードレビューは単なる品質管理以上の意味を持つ。自分のコードの特徴や癖を知り、より良い方法を学ぶ機会となる。

批判的思考も重要な要素だ。新しい情報や手法に出会ったとき、それを鵜呑みにせず、自分なりに検証し、既存の知識と照らし合わせて評価する。これにより、より深い理解と確かな変化が可能になる。たとえば、新しいフレームワークを導入する際、その利点だけでなく、自分たちのプロジェクトにとって本当に適切かを慎重に検討する姿勢が必要だ。

41.3 自己変容の個人差を理解する

自己変容に対する適性は、人によって大きく異なる。変化を自然に受け入れられる人もいれば、強い抵抗を感じる人もいる。これは単なる性格の違いではなく、それぞれの経験や価値観、思考様式が影響している。

変化を受け入れやすい人には、いくつかの共通した特徴がある。新しいアイデアや異なる価値観に対して、「面白そうだ」「試してみたい」という好奇心を自然に感じる。失敗を恐れすぎず、むしろ学びの機会として捉えられる。そして、常に自分を改善していきたいという強い意欲を持っている。

一方、変化を受け入れづらい人は、既存の自分への愛着が強かったり、失敗への不安が大きかったりする。「今のやり方で十分だ」「変化によって何かを失うのではないか」という思いが、新しい一步を踏み出すことを躊躇させる。

しかし、これは克服できない壁ではない。変化への抵抗を感じる人でも、適切なアプローチを取ることで、徐々に自己変容を受け入れられるようになる。最も効果的なのは、小さなステップから始めることだ。大きな変革を一度に目指すのではなく、わずかな思考や行動の変化から試してみる。

たとえば、新しい技術を学ぶ際、いきなり本番のプロジェクトに導入するのではなく、まずは小さな個人プロジェクトで試してみる。あるいは、チーム内で限定的な範囲から始めてみる。こうした段階的なアプローチにより、変化への不安を最小限に抑えながら、新しい経験を積むことがで

きる。

41.4 メタ認知と自己変容

自己変容をより効果的に進めるには、メタ認知、つまり自分の思考や学習プロセスを客観的に観察し、理解する能力が重要となる。これは単に「何を学んでいるか」だけでなく、「どのように学んでいるか」「なぜそのように学ぼうとしているのか」を理解することを意味する。

具体例を見てみよう。ソフトウェア開発の場合、新しい技術を学ぶとき、多くの人は「とりあえずチュートリアルをやる」というアプローチを取る。しかし、メタ認知を働かせることで、このアプローチ自体を見直すことができる。なぜチュートリアルを選んだのか、自分の学び方の特徴は何か、より効果的な方法はないか、といった具合に。

実践的な振り返りの方法として、開発者の多くが活用している「ふりかえり」の手法が参考になる。一日の終わりに、その日の学びを三つの観点から振り返る。「今日理解できたこと」「まだ理解できていないこと」「次に試したいこと」。これを継続することで、自分の学びのパターンや、効果的な方法が見えてくる。

技術書を読むときも同様だ。単に内容を追うだけでなく、「なぜここで理解に詰まったのか」「どの説明が特に分かりやすかったか」「自分なりの理解の仕方は何か」を意識的に観察する。この観察を通じて、自分に合った学び方が見えてくる。

41.5 自己変容を阻む要因

変化を望みながらも、なかなか一步を踏み出せないことがある。その背景には様々な要因が存在する。現状への安住は、その代表的なものだ。慣れ親しんだ環境や方法から抜け出すことへの不安は、誰しもが感じるものだ。

具体的な例を見てみよう。プログラマーがある言語やフレームワークに長年携わっていると、新しい技術への移行に大きな抵抗を感じる。その理由は、蓄積してきた知識や経験が一時的に通用しなくなる不安、新しい環境での失敗への恐れ、そして学び直しの労力への躊躇だ。

この抵抗を和らげるためには、段階的なアプローチが効果的だ。新しい技術を完全に採用する前に、小規模なプロジェクトや個人的な実験から始める。既存の技術と新しい技術を並行して使うことで、急激な変化によるストレスを軽減できる。

自己診断のポイントとして、以下のような点に注目するといい。新しい技術や方法に触れたとき、即座に「それは無理だ」と判断していないか。失敗を過度に恐れて、挑戦を避けていないか。過去の投資を過度に気にして、新しい選択肢を検討する機会を逃していないか。これらの点を定期的にチェックすることで、自己変容を妨げる要因に気づきやすくなる。

41.6 大きな自己変容を受け入れる

自己変容には様々な規模がある。日々の小さな気づきによる変化もあれば、キャリアを大きく転換するような変化もある。特に大きな自己変容は、勇気が必要だ。しかし、それを受け入れる準備ができていれば、新たな可能性が開かれる。

技術者のキャリアを例にとると、ある技術スタックから全く異なる技術スタックへの移行は、大きな自己変容を必要とする。バックエンドエンジニアからフロントエンドエンジニアへの転向、あるいは特定の言語やフレームワークから全く異なるものへの移行は、その典型例だ。これは単なる技術の習得以上に、開発に対する考え方そのものの変革を求められる。

また、技術者からマネージャーへの転身も重要な自己変容の例だ。これまでとは全く異なる役割や責任を担うことになる。コードを書くことから人を導くことへ、技術的な問題解決から組織的な課題への対応へと、求められるスキルセットが大きく変わる。

このような大きな変化に向き合うとき、重要なのは段階的なアプローチだ。いきなり全てを変えようとするのではなく、まずは小さな実験から始める。新しい技術なら、サイドプロジェクトでの試用から。マネジメントなら、チームでの小さなりーダーシップの機会から。そして、その経験から学びながら、徐々に変化の規模を広げていく。

■コラム：筆者が体験した二度の大きな自己変容

筆者は実に大きな自己変容を二度行っている。

一度目は、JavaScript と React への転向だった。当時の筆者はバックエンド開発をしていたのだが、2015 年に登場した ECMAScript 2015 という、今あるモダンな JavaScript の形が定まったことで、あらゆる技術が JavaScript に収束していく確信を得たことと、React に出会い同様に React に収束していく確信を得たことで、フロントエンド、JavaScript/React に一点集中するという大きな賭けに出た。

この転向から得た最大の学びは、技術の変遷を見極める目だ。新しい技術が登場したとき、それが一時的なブームなのか、本質的な変化なのかを見分ける力が磨かれた。

2024 年末である現代においては Python, Java, JavaScript が世界で最も使われている三大プログラミング言語である。JavaScript やその発展形である TypeScript はフロントエンドのみならず、バックエンドでもメジャーな言語になった。フロントエンド技術としては React が文句なくスタンダードである。つまり筆者は掛けに大勝ちできた。

次の大きな自己変容は、一回目の転向で得たフロントエンドエンジニアとしてのキャリアを置いて、LLM（大規模言語モデル）の社会実装に全力を注ぐ決断だった。2022 年末に ChatGPT と出会い、2023 年 3 月頃の土日に寝食を忘れて 48 時間耐久 LLM 研究をやってみたら、物の見事に LLM の魅力にとりつかれた。その後すぐ会社を辞める連絡をして、当時 LLM で知り合った人と一緒にやっていく決断をして、そのあとその人が新しく立ち上げた会社に入社することが決まった。いまは、TypeScript でフロントエンドもバックエンドも触っている、というかここ一年はバックエンドと LLM しか触っていない。

この二度目の転向では、直感を信じる勇気の重要性を学んだ。ChatGPTとの出会いは、まさに「これが未来を変える」という直感的な確信をもたらした。また、既存のキャリアを捨てるような大きな決断でも、それまでの経験は必ず新しいフィールドで生きることも実感として理解できた。

この大いなる掛けが、もう一度大勝ちをするかはこれから歴史が証明することになるが、筆者は当然一回目よりも大きな勝ちになると確信している。

これらの大きな変化を可能にしたのは、強い知的好奇心だ。筆者にとって、興味の対象を徹底的に探求することは自然なことだった。このような好奇心は、自己変容を受け入れる強力な原動力となる。

41.7 自己変容を促進する環境づくり

自己変容を効果的に進めるには、適切な環境づくりが重要だ。まず、変化を受け入れやすい心理的な環境が必要だ。「このままではうまくいかない」という危機感と、「変われば良くなる」という希望のバランスを保つことが重要だ。また、理想と現実のバランスも重要だ。高すぎる理想は挫折を招き、低すぎる理想は変化の動機を失わせる。

物理的な環境も重要だ。新しい学びや実践の機会を得やすい環境、失敗しても大きなダメージを受けない環境を整えることで、より積極的に変化に取り組める。たとえば、新しい技術を学ぶなら、実験用の開発環境を用意する。新しい役割に挑戦するなら、まずは小規模なプロジェクトで試してみる。

また、変化のプロセスを支えてくれる仲間や、メンターの存在も重要な要素となる。同じような変化を経験した人からのアドバイスは、特に価値がある。彼らの経験は、自分の変化の道筋を照らす灯台となる。特に技術の世界では、先人の経験から学ぶことで、同じ失敗を避け、より効率的な成長が可能となる。

さらに、変化を受け入れやすい思考様式を育むことも大切だ。新しいアイデアや異なる視点に対してオープンな態度を持つ。失敗を恐れすぎず、それを学びの機会として捉える。そして、自分の考えや行動を客観的に観察し、必要な調整を加えていく。この姿勢は、継続的な成長の基盤となる。

41.8 自己変容の多様性を理解する

自己変容のプロセスは、人によって大きく異なる。知的好奇心が原動力となる人もいれば、必要に迫られての変化もある。また、競争心や達成欲求が変化を促す場合もある。重要なのは、自分に合った形で自己変容を受け入れることだ。他人の変容のペースや方法と比較する必要はない。

たとえば、新しい技術への適応を考えてみよう。早い段階から積極的に取り入れる人もいれば、十分に成熟してから採用する人もいる。どちらが正しいということはない。むしろ、その人の状況

や性向に合った選択が最適なのだ。技術選択において重要なのは、その技術が自分やチームにとつて本当に必要かどうかを見極めることだ。

また、変化の動機も人それぞれだ。技術的な興味から始まることもあるが、キャリアの必要性から始まることもある。時には、周りの影響や、偶然の出会いがきっかけとなることもある。これらの違いは、変化のプロセスや結果に影響を与えるが、どれも等しく価値のある動機となりうる。

41.9 自己変容がうまくいかないとき

すべての自己変容が成功するわけではない。時には目標とは異なる方向に進むこともある。しかし、一見失敗したように見える変化でも、長期的には思わぬ形で活きることがある。技術学習を例にとると、ある技術の学習は直接的には実務で活用できなくても、その過程で得た考え方や知見が、別の場面で価値を持つことがある。

重要なのは、その経験から何を学び取れるかだ。たとえ目標とした変化が達成できなくても、そのプロセスで得られた気づきや、培われたスキル、広がった視野は、必ず何らかの形で役立つ。それは即座に現れることもあるが、何年も経って初めて価値を發揮することもある。

また、「うまくいかない」と感じるとき、それは本当に失敗なのか、それとも単に時期尚早だったのか、あるいは別の形での成功だったのかを、改めて考えてみる価値がある。変化の価値は、必ずしも当初の目標達成だけにあるわけではないのだ。

特に技術の世界では、一見無駄に見える回り道が、後になって重要な意味を持つことがある。たとえば、実務では使わない言語やパラダイムを学ぶことで、プログラミング全般への理解が深まることがある。また、一つの技術を深く理解することで、他の技術への応用力が高まることもある。

■コラム：筆者の「失敗」から得た学び

筆者がバックエンドをやっていた頃、色々とうまくいかないことがあった。

Haskell や Scala や Go 言語に手を出してみたり、DDD を学んだりしてきた。これらは当時の実務としてはうまく活用できなかった。

しかし、この「失敗」は別の形で実を結んだ。

JavaScript は元々関数型言語の要素が色濃い言語である。最近は特に関数型言語としての特性が好まれる傾向が強く、たとえば React なんかも関数型言語としてのエッセンスを学んでいると理解が捲る。そのため、Haskell や Scala の経験が役に立った。

同様に DDD や Go なんかもその後の設計方針に色々と反映されている。一見無駄に思えた学びが、思わぬ形で活きてくることは少なくない。

最近では、技術的負債への向き合い方も変わってきた。以前は完璧な設計を目指していたが、今は「本当に守るべき部分」を見極めることの方が重要だと考えている。技術の進歩は速く、今の負債は明日の当たり前になることもある。むしろ、変化に対する柔軟性を保つことが大切なのだ。

また、理想と現実のバランスも変わった。理想を追い求めるることは大切だが、地に足のつい

た現実的なアプローチも同様に重要だ。どこにも完璧な環境や理想の楽園はない。それを認識した上で、今できる最善を尽くすことの方が生産的だ。

不安との付き合い方も学んだ。コントロールできないことを過度に心配しても仕方ない。むしろ、そのときできるベストを尽くし、それで十分だと認めることが建設的だ。

41.10 まとめ

学びにおける自己変容は、避けて通れないプロセスである。それは時として不安や抵抗を伴うが、これらの感情は変化の自然な一部として受け入れることが重要だ。変化は段階的に進み、時には後戻りしながらも、着実に前に進んでいく。

自己変容は、必ずしも計画通りに進むとは限らない。時には失敗し、時には予想外の方向に進むこともある。しかし、その過程で得られる学びは、必ず何らかの形で価値を持つ。重要なのは、変化を恐れず、かといって無理もせず、自分のペースで着実に前進していくことだ。そして、その過程で得られる気づきや学びを、次の変化への糧としていくことである。

このような自己変容の過程は、まさに学びそのものであり、それは終わりのない旅だ。その旅を楽しみ、自分なりの歩み方を見つけていくことこそが、真の学びの姿なのかもしれない。自己変容を受け入れることは、より豊かな学びの世界への扉を開くことなのだ。

第 42 章

生成 AI を活用する

@erukiti

これまでの章では「学び」自体を徹底的に深掘りしてきた。この章では学びへの生成 AI 活用について語りたい^{*1}。

ただし、この章に関しては注意事項がある。生成 AI にまつわる情報は極めて頻繁にアップデートが入るため、あなたがこの章を読んでいるタイミングでも有効かはわからない。なるべくそういうないように心がけてはいるが、内容が一瞬にして陳腐化している恐れがある。

42.1 生成 AI とは

生成 AI とは「生成する AI」である。馬鹿馬鹿しい定義だが、生成 AI は研究・開発における専門用語でもなんでもなくて、単なるマーケティング用語だ。生成 AI はその程度の情報量しかない薄っぺらい言葉にすぎないが、使い勝手の良い言葉でもある。

筆者は AI エンジニアである。主に大規模言語モデル（以後、Large Language Model の略で LLM と書く）を取り扱っている。LLM は文字列を入れたら、それに応じた最もそれらしい文字列を返してくれる魔法の技術くらいに考えればいい。

第 38 章「学びと言語化」で述べた通り、学びには「言語化」という重要な要素がある。実は LLM は、私たちの言語化のプロセスを強力に支援してくれる存在だ。漠然とした理解を明確な言葉に変換する手助けをしてくれるのだ。

ChatGPT が LLM 活用の代表的な事例だ。ソフトウェア技術者なら GitHub Copilot を使っているだろうか。あれも LLM を活用したものである。

LLM は主にアプリケーション・サービスとして提供されている。プロフェッショナル向けに API という形でも提供されているため、学びを深めたい読者は API を触ってみるのもいいだろう^{*2}。

以下では、生成 AI としてマーケティングされているアプリケーション・サービスの中で、学びに活用できそうなものを紹介していこう。なお、筆者は LLM が専門なため、画像生成や、音声、動画、音楽などといったサービスに関しては特に触れない。

*1 これを書こうと思ったきっかけは「(AI エンジニアとして) 何かおすすめツールはありますか?」と聞かれたことだ。

*2 さらにプロ向けの存在として、ローカルマシンでも動くローカル LLM というものもある。HuggingFace という機械学習に特化した GitHub のようなサイトで、LLM の中身（学習されたデータ）がオープンで配布されている。これとオープンソースの LLM ソフトを組み合わせると、自宅やクラウド上で好きなだけ使える LLM になる。興味がある人は「ローカル LLM」「ollama」「LM Studio」「llama.cpp」などを調べるといいだろう。

42.2 対話型 AI

OpenAI 社の ChatGPT (チャット GPT) は対話型 AI と呼ぶべきサービスである。同様のものには Anthropic 社 (アンスロピック) の Claude (クロード) や Google 社の Gemini (ジェミニ) がある。

対話型 AI は、対話、チャットによって、LLM に返答してもらうアプリケーションだ。利用者は何かしらの指示を出す。「学びについて教えて」「学びのプロセスについて詳細に掘り下げてみて」のような指示で、こういう指示のことをプロンプトという。

LLM や対話型 AI の一番の特徴は、汎用性にある^{*3}。

プロンプトの与え方一つで、返答内容が千差万別に変わるというのが、対話型 AI の強みであり、弱みでもある。強みはプロンプトの与え方さえマスターすれば汎用的に色々なことをこなせるということだ。弱みは、プロンプトの与え方を間違えると、とてもイライラさせられることだ。このプロンプトそのものがエンジニアリングの対象であるため、プロンプトエンジニアリングというものもある。

もう一つ重要なこととして LLM という性質上の向き不向きもある。LLM はたとえば OpenAI 社がアクセスを許可されている範囲の世界をある時点で切り取ったものなので、知識は最新のものではないし、学習の偏りもある。LLM は言葉を処理する能力が極めて高いが、知識は不完全なのだ。

学習素材が十分あるものは、つまりウェブ上に公開されている情報が多いジャンルのもの、たとえば TypeScript や Python の情報であれば知識の正確さはある程度担保されている。これがマイナーなプログラム言語のことになると途端に正確さが失われる。つまり対話型 AI の知識を無条件で信用してはならない。

42.3 プロンプト

本書は学びの本なので、あまり踏み込んだ内容にすべきではないが、プロンプトは、LLM や対話型 AI を使う上で欠かせないため、軽く説明をしよう。

プロンプトにおける一番重要な考え方は絞り込みである。検索エンジン活用と同じようなものだ^{*4}。汎用技術なのだから文脈を限定しないと、利用者の望む答えを返さない可能性は高い。

「学びについて教えて」と指示を出すと概略をつかむための一般論が帰ってくる。あなたは何を知りたいのだろうか？ 学びの定義か、学びのプロセスか、学びのコツか。検索エンジンを活用のと同じように絞り込むと、少しずつあなたの知りたいことに近づくはずだ。第 40 章「学び hack」で見てきたように、学びには明確な目的意識が重要だ。同様に、AI との対話においても、自分が何を知りたいのかを明確にすることで、より効果的な学びが可能になる。

^{*3} 旧来の AI と呼ばれる技術はどうしてもやれることができが限定了だったが、LLM はあらゆることをこなせるポテンシャルがある。まだまだ人間の知的活動のすべてを奪えるものではないが、人類史上でも類を見ないスピードで進化し続けているため、数年以内に人間と同等の汎用性能といわれる AGI に到達する確率は高いとされている。

^{*4} プロンプトエンジニアリングとしてのコツは、最初は雑なプロンプトを投げて、どこまで指示を追加したらより適切な返答をしてくれるかを絞り込んでいく、という検索エンジンの活用テクニックに似たところがある。

検索エンジンと違うのは、より詳細で自然な指示を追加出来る点だ。「学びのコツについて、中学生の情報の授業で役立つように、教えて」というような指示は、旧来の検索エンジンではできない。

ここまでを理解できれば、プロンプトに対する知識としては十分だ。インターネットには胡乱なプロンプトテクニックと称したものが回ってるがそれらのほとんどは意味がない。ある時点では有効だったかもしれないが、最先端のLLMにとっては不要なものとなっていることが多い。

LLMも対話型AIも、なるべくプロンプトエンジニアリングが不要な方向に進化しているのだ。多くのユースケースではこう答えた方がいいようにアップデートし続けている。汎用技術であろうとする点と些かの矛盾を抱えているため、ときどきとんでもない会話をすることもある点には気をつけなければならない。

42.4 ChatGPT, Claude, Gemini

ChatGPTは、2024年12月6日時点ではgpt-4o/gpt-4o-mini/o1/o1-mini/o1-proなどのモデル(LLMのこと)を使うことができる。これらのモデルはそれぞれ性能や応答速度が違う。

- gpt-4oは標準的な最先端モデルである。賢いが応答が少し遅い
- gpt-4o-miniは小さな最先端モデルである。gpt-4oよりは明らかに馬鹿だが、応答は早い
- o1は実験的な最先端モデルである。gpt-4oと比べて遙かに思慮深いが計算コストはかかる。原稿執筆時点では世界で2番目に賢い
- o1-proも実験的な最先端モデルである。原稿執筆時点では世界で最も賢い
- o1-miniも実験的な最先端モデルである。gpt-4oとo1の中間のような存在である

くどいくらいに最先端モデルと書いているが、ChatGPTやClaudeやGeminiをウェブ上で普通にアプリとして使う分には、どのモデルを選んだとしても、最先端モデル^{*5}なのだ。

無料でChatGPTを使っていると、モデル選択ができずにgpt-4o-miniに該当するモデルしか使えない。要するに無料ユーザーはとても馬鹿なモデルしか使えないのだ。

有料プランには二つの選択肢がある。ChatGPT PlusとChatGPT Proだ。ChatGPT Plusではgpt-4o・gpt-4o-mini・o1・o1-miniが利用できる。この中でo1は強い利用制限がかかっていて、12/6現時点では一週間のリクエスト数は50に限定されている。つまりお試しでo1が使えるという状況だ。月額\$20(日本円だと3000円くらい)だ。

ChatGPT Proは全部のモデルをほぼ無制限に使える。他にも色々と追加機能が登場している^{*6}。月額\$200(日本円だと3万円くらい)だ。

ライバルのClaudeも同様の構造を持っている。Claude 3.5-sonnet(ソネット)は極めて賢い最

^{*5} 最先端モデルとは、最低でも1兆円以上のハードウェアと、原子力発電所を専用に建造したくなるほどの莫大な電力消費と、世界のトップの中のトップの技術者を数百人以上費やして、開発されている、文字通りの最先端モデルである。最先端モデルを開発できる企業は2024年時点ではアメリカにしかない。二番手グループであれば、カナダ、フランス、中国、イギリスなどにも存在している。日本もせめて二番手グループには入ってほしいものだ。

^{*6} このような書き方をしているのは、12/6以後12日間に渡って、OpenAI社が新機能を相次いで発表するというスタイルを取っているため原稿の締め切りまでに全容を明らかにできないからだ。

先端モデルで、特に日本語の扱いでは間違いなくトップクラスの性能を誇る。応答速度は gpt-4o と同程度だ。3.5-haiku（ハイク）は sonnet よりは劣るもの、gpt-4o にも引けを取らない性能を持ち、応答速度はより速い。3-haiku は小さな最先端モデルで、賢さも応答速度も gpt-4o-mini と同程度だ。この三段階の性能差は、第 39 章「学びのプロセスを詳細に分解する」で述べた学びの段階的なプロセスにも似ている。より高度なモデルを使うことで、より深い思考と理解が可能になるのだ。

Claude の無料版も 3-haiku に該当するモデルが使われている。有料版は 2024 年 12 月 6 日時点では、Professional Plan がある。\$20 で、3.5-sonnet をほぼ使い放題というのだ。

これらのモデルの違いは、単なる性能の差以上の意味を持っている。無料ユーザーは馬鹿なモデルしか使えないため、学びに活用しようと圧倒的に力不足だ。絶対に後悔するので無料モデルなら使わぬ方がマシだと思った方がいい。ChatGPT を使えないという人の大半は無料モデルしか使っていない上、プロンプトの基本を知らないので、活用ができないのだ。

有料プランである ChatGPT Plus や Claude Professional plan は、両方とも毎月 \$ 20、つまり日本円だと 3000 円くらいかかるが、その分とても賢い。もちろん、ChatGPT Pro の \$ 200 を支払う覚悟があればさらに賢いモデルにアクセスできる。

無料版は高校生か大学入ったけど頭の悪い子くらいの性能だが、有料版はトップレベルの大学院生くらいの知的レベルを持っている。特に Claude の場合はその傾向が顕著だ。ChatGPT で gpt-4o を使う限りそこまで頭は良くないが、大学生から大学院生くらいの知的レベルと言える。o1 や o1-pro に至ってはその遙か上だ。筆者はまだ o1 と o1-pro を評価できていないが、それよりも劣る o1-preview という先行公開版だけでも、他のモデルとは異質の賢さ、思慮深さを感じ取っていた。

筆者は生成 AI の専門家であるため ChatGPT Plus と Claude Professional Plan ともに契約しているが、普通の人はどうちらかを契約するといいだろう。どちらも一長一短だ。軍拡競争としか表現できないような競争を続けている両社^{*7}なので、性能も実装する機能も、極めて頻繁にアップデートが入ってしまうため、長期的に見てどちらが良いと言えないのだ。

ここまで述べた評価は 2024 年 12 月初め時点のものに過ぎないことを留意してほしい。2025 年半ばくらいには細かい性能は変わっているだろう。ただし無料版はあきれるほど馬鹿なので有料版を使うべきという原則は絶対に何があっても変わらない^{*8}。そしておそらく Claude の方が日本語性能は高いという傾向は 2025 年も続くはずだ。

ここまで Google の Gemini を取り上げてこなかったのには理由がある。Gemini には、LLM・

^{*7} たとえばビッグテック各社（Google, Microsoft, Meta, Amazon）は技術開発に毎年ものすごい予算をかけ、ハードウェア・インフラに数兆円規模の投資をしている。ビッグテックの資金が流れ込んでいる OpenAI や Anthropic はインフラへの投資金額こそ低いが、札束を燃やして最先端モデルの開発と運用をしている。これらの競争が続くと、近未来 SF で指摘されてきた一企業が国家を凌駕するのも時間の問題だ。人間を超える知性を持つ AI を一企業が完全に所有するのだ。当然それを持たない存在すべてが対抗できないことを意味する。これは世界で始まった本当の経済戦争だ。

^{*8} gpt-4o-mini や 3-haiku は、比較的軽量で馬鹿で速いモデルだが、それでもこれらを維持するのには考えたくもないほど莫大な費用がかかるのだ。いわゆるビッグテックの金を毎年燃やし続けているから維持できているに過ぎない。これを無料で使えること自体がおかしいレベルである。無料版は性能を抑えて、有料版に誘導したいというのが経済的な合理性である。フリーミアムモデルという、コンピュータの歴史において何度も見かけたやつだ。

対話型AIにおける重大な欠陥である、ハルシネーションが生じやすいという問題がつきまとっていた。ハルシネーションは本当は存在しないものをそれっぽく答えるものだ。もちろんChatGPTやClaudeにも同じ問題はあるが、生じる可能性はより低い。

もちろんGeminiも、ChatGPTやClaudeと争うトップレベルの最先端モデルであり、最近は改善しているが、そういう過去があったので手放しでおすすめしづらい。なお、プロンプトエンジニアリングをちゃんとしたうえで専門家としてAPIで触るならGeminiシリーズも本当に素晴らしい。2025年の間には一般向けにも手放しでおすすめできるようになっていると筆者は期待している。

この章では固有名詞が多く登場し混乱する読者もいるかもしれない、それについて追加の解説を書いておく。

- ChatGPT/Claude/Geminiは対話型AIの名前である
- 対話型AIはLLMを利用したサービスである
- GPT/Claude/GeminiはLLMの名前である
- LLMは個別にモデル名を持つ。モデルごとに性能が異なる
- gpt-4o/gpt-4o-mini/o1/o1-mini/o1-pro/o1-previewはOpenAIが開発しているモデル名である
- claude-3.5-sonnet/clause-3.5-haiku/clause-3-haikuはAnthropic社が開発しているモデル名である
- gemini-1.5-pro/gemini-1.5-flashはGoogle社が開発しているモデル名である
- ChatGPT/Claudeは、無料プランと有料プランがあり、無料プランは信じられないほど馬鹿なので有料プランじゃないと真価が發揮されない

42.5 対話型AIに文章を書かせてみる

実際にClaude Professionalで問答をしてみよう。まず簡単に「学びについて教えて」と投げてみた。なお、この目的においてはChatGPTよりもClaudeの方が圧倒的に適している。Claude 3 Sonnetが登場して以来、日本語の文章を書かせるというタスクにおいてはChatGPTは圧倒的にClaude Sonnetに負けていて、その傾向は3.5 Sonnetが登場した上でさらに進んでいる。日本語の文章を読み書きする点に特化するなら筆者としてはClaudeをおすすめしたい。

Claude Sonnetの返答

学びとは人生において非常に重要なプロセスですね。効果的な学習には、以下のような要素が含まれます：

1. 主体性と目的意識を持つこと - 何のために学ぶのか、どんな成長を目指すのかを明確にすると、モチベーションが保ちやすくなります。
2. 実践と振り返りのサイクル - 知識を得るだけでなく、実践して、その結果を振り返ることで、より深い理解が得られます。
3. 多角的なアプローチ - 一つの事柄でも、異なる視点から学ぶことで、より立体的な理解が可能になります。
4. 繰続的な取り組み - 短期的な詰め込みよりも、日々の少しづつの積み重ねの方が、長期的な成長につながります。
5. 周囲との学び合い - 他者との対話や意見交換を通じて、自分一人では気づかなかつた視点を得られます。

具体的にどのような分野や目的での学びに興味をお持ちですか？ そうすれば、より具体的なアドバイスができるかもしれません。

対話型 AI からの返答は、まず学びの基本的な要素について、主体性と目的意識、実践と振り返りのサイクル、多角的なアプローチ、継続的な取り組み、周囲との学び合いといった観点から説明してくれる。これは第 39 章で述べた学びのサイクルとも重なる部分が多い。しかし、これはまだ一般論の域を出ない。より深い理解を得るには、具体的な文脈での対話が必要だ。

たとえば、学びについて本を書きたいと思ったらどうするか？ そこで「学びについて本の原稿を書こうと思う。どういう構成が考えられるか？」という指示を投げてみると、AI は序章から終章まで、体系的な構成案を提示してくれる。これには学びの意義、メカニズム、形態、実践方法、個人に合った学習スタイル、デジタル時代の学び方といった要素が含まれる。

実際に Claude Professional で問答をしてみよう。まず簡単に「学びについて教えて」と投げてみた。

なお、この目的においては ChatGPT よりも Claude の方が圧倒的に適している。Claude 3 Sonnet が登場して以来、日本語の文章を書かせるというタスクにおいては ChatGPT は圧倒的に Claude Sonnet に負けていて、その傾向は 3.5 Sonnet が登場した事でさらに進んでいる。日本語の文章を読み書きする点に特化するなら筆者としては Claude をおすすめしたい。

このような返答となる。優秀な回答ではあるが、一般論を語っているにすぎない。なお、この程度の会話であれば無料版 ChatGPT でも似たような返答をしてくる。

たとえば、学びについて本を書きたいと思ったらどうするか？ 今度は「学びについて本の原稿を書こうと思う。どういう構成が考えられるか？」という指示を投げてみた。

実際の AI との対話例を見てみよう。

Claude Sonnet の返答

学びとは人生において非常に重要なプロセスですね。効果的な学習には、以下の要素が含まれます：

1. 主体性と目的意識を持つこと - 何のために学ぶのか、どんな成長を目指すのかを明確にすると、モチベーションが保ちやすくなります。
2. 実践と振り返りのサイクル - 知識を得るだけでなく、実践して、その結果を振り返ることで、より深い理解が得られます。
3. 多角的なアプローチ - 一つの事柄でも、異なる視点から学ぶことで、より立体的な理解が可能になります。
4. 継続的な取り組み - 短期的な詰め込みよりも、日々の少しづつの積み重ねの方が、長期的な成長につながります。
5. 周囲との学び合い - 他者との対話や意見交換を通じて、自分一人では気づかなかった視点を得られます。

具体的にどのような分野や目的での学びに興味をお持ちですか？ そうすれば、より具体的なアドバイスができるかもしれません。

このような返答は優秀ではあるが、一般論を語っているにすぎない。より具体的な文脈での対話を試みてみよう。学びについての本の構成案を聞いてみた結果がこちらだ。

Claude Sonnet の返答

学びに関する本の構成について、以下のような案を提案させていただきます：

序章：なぜ今、学びが重要なのか

- 変化の激しい現代社会における学びの意義
- 生涯学習時代の到来と求められるスキル

第1章：効果的な学びのメカニズム

- 脳科学から見た学習プロセス
- 記憶と理解の関係性
- モチベーションと学習効果の関連

第2章：多様な学びの形態

- フォーマル学習とインフォーマル学習
- オンライン学習と対面学習の特徴と活用法
- ソーシャルラーニングの可能性

第c3章：実践的な学習方法論

- アクティブラーニングの実践
- 振り返りと内省の重要性
- 失敗からの学び方

第4章：個人に合った学習スタイルの確立

- 学習タイプの診断と活用
- 時間管理と学習計画の立て方
- 持続可能な学習習慣の形成

第5章：デジタル時代の学び方

- テクノロジーを活用した効率的な学習
- 情報リテラシーとクリティカルシンキング
- オンラインリソースの選び方と活用法

終章：これから学びのビジョン

- 未来の学びの展望
- 個人と組織における学びの統合
- 持続可能な学習社会の実現に向けて

この構成についてどのように感じられますか？特に重点を置きたい部分や、追加したい視点などございましたらお聞かせください。

この構成案は興味深いものだが、より具体的な内容を見てみよう。試しに序章を書かせてみることにした。「試しに序章を書いてみて。2000文字程度で」という指示を出すと、以下のような文章が生成された。

Claude Sonnet の返答

序章：なぜ今、学びが重要なのか

私たちは今、かつてない速さで変化する時代を生きています。技術革新は日々加速し、人工知能やロボティクスの進化は、私たちの仕事や生活のあり方を根本から変えつつあります。昨日までの常識が今日には通用しない。そんな状況が、もはや珍しくありません。

このような激動の時代において、最も重要な能力の一つが「学び続ける力」です。しかし、ここで注意しなければならないのは、従来型の「知識の蓄積」としての学びだけでは、もはや不十分だということです。必要なのは、新しい状況に適応し、創造的な解決策を見出すための、より柔軟で実践的な学びの姿勢です。

例えば、コロナ禍は私たちに何を教えてくれたでしょうか。多くの人々が、オンラインでのコミュニケーションやデジタルツールの活用を、否応なく学ばざるを得ませんでした。この経験は、学びの本質が「必要に迫られた時にいかに素早く適応

できるか」にあることを、まさに実証したと言えるでしょう。

現代社会における学びの重要性は、主に以下の三つの観点から考えることができます。

第一に、職業人としての視点です。世界経済フォーラムの報告によれば、今後 10 年間で、現在の仕事の多くが大きく変容するか、まったく新しい職種に置き換わると予測されています。このような環境下で生き残るためにには、常に新しいスキルを習得し、自己をアップデートし続ける必要があります。

第二に、個人の幸福という視点です。学びは単なるスキル獲得以上の価値があります。新しいことを理解し、習得する過程で得られる充実感や達成感は、人生を豊かにする重要な要素となります。また、学びを通じて広がる人間関係や視野は、私たちの人生に新たな可能性をもたらしてくれます。

第三に、社会の持続可能性という視点です。気候変動やエネルギー問題、格差の拡大など、現代社会は複雑な課題に直面しています。これらの課題に対処するためには、社会全体としての学びと進化が不可欠です。一人ひとりが学び続け、成長し続けることが、社会の持続可能な発展につながるのです。

しかし、ここで一つの重要な問い合わせ浮かび上がります。「どのように学べばよいのか？」という問いです。情報があふれる現代において、効果的な学びの方法を見出すことは、決して容易ではありません。また、日々の忙しさの中で、学びの時間を確保することも大きな課題となっています。

本書は、このような現代における学びの課題に対して、具体的な指針を提供することを目指しています。効果的な学習のメカニズムを理解し、自分に合った学習方法を見つけ、持続可能な学びの習慣を確立する。そのためのヒントを、最新の研究成果や実践例を交えながら紹介していきます。

学びは、決して義務や負担であってはなりません。それは、私たちの人生をより豊かで充実したものにするための、かけがえのない営みです。この本を通じて、読者の皆様が、自分なりの「学びの道」を見出すためのヒントを得ていただければ幸いです。

さあ、これから時代に求められる、新しい学びの形を探っていきましょう。

かなりそれっぽい文章だ。

さて、筆者の文体や癖のようなものを踏まえた文章を生成してみよう。原稿を二つ添付して「今添付した原稿を元に、文体や癖を真似て、改めて序章を書いてみて。構成や主張内容も、原稿を参考にして」と指示してみた。

Claude Sonnet の返答

添付された原稿の文体を参考に、序章を書き直してみます。

序章：知識の蓄積から自己変容へ - 学びの本質を探る

「学び」とは何だろうか？多くの人は「知識を得ること」「スキルを習得すること」と考えるかもしれない。しかし、本質的な学びとは、自己変容のプロセスそのものだ。知らなかつた状態から知る状態へ、できなかつたことができるようになる状態へ、理解していかなかったことを理解できる状態へ。そうした変化の連続が、真の学びなのだ。

私たちは今、かつてないほど大量の情報に囲まれている。インターネットを開けば、膨大な知識の海が広がっている。技術書、ブログ記事、動画教材、そして AI による支援まで、学びのためのリソースは豊富だ。しかし、これらの情報は必ずしも学びに直結しない。むしろ、情報があふれているからこそ、効果的な学び方を理解することが重要になってきている。

学びには様々な形がある。学校教育のような体系的な学び、仕事を通じた実践的な学び、趣味から広がる自發的な学び。これらは一見異なるように見えて、実は共通する本質的なプロセスを持っている。それは「言語化」と「サイクル」だ。

言語化とは、漠然とした理解や感覚を、明確な言葉として表現することだ。私たちは普段、様々なことを「なんとなく」理解している。しかし、それを具体的な言葉として表現できない限り、その理解は不完全なものに留まる。言語化することで、理解は深まり、他者と共有可能な知識となる。

学びのサイクルは、現状の把握から始まり、目標の設定、実践、フィードバック、そして新たな目標の設定へと続く。この

サイクルは一回で完結するものではない。むしろ、繰り返し回すことで、学びは螺旋状に深まっていく。

本書では、この「言語化」と「サイクル」を軸に、効果的な学びの方法を探っていく。プログラミングや技術的な学びの例も取り上げるが、それらは普遍的な学びの原理を理解するための具体例として扱う。学びのメカニズムを理解し、実践することで、あらゆる分野での成長につながるはずだ。

与えられた知識を受動的に受け取るのではなく、自ら考え、言語化し、実践する。そんな能動的な学びのあり方を、これから一緒に探っていきたい。今、あなたの手元にあるこの本も、著者たちの学びを言語化し、共有したものだ。この本を通じて、読者の皆さんのが自分なりの学びの形を見つけるきっかけになれば幸いである。

さあ、学びの本質を探る旅に出発しよう。

細かいところは違うものの、かなり筆者っぽくなった。

Claude Sonnet の場合は、このように日本語を扱うのが達者なので、原稿を添付して、色々注文を付ければ、それなりにそれっぽい文章を生成してくれる。さてあなたは筆者が書いた原稿のうち、どれくらいが生成AIによるものかわかるだろうか？ 特に答えは書かないが、実は使っていたのだ。気づく場所はあったんだろうか？ 気づく場所がなければ、それが Claude Sonnet の恐るべき力の証明だ。

もっとも、学びのアウトプットとしては、これに頼ると本来の目的を果たせなくなるため注意しなければならない。第39章で見てきたように、学びには実践と振り返りのサイクルが重要だ。AIに文章を書かせるというよりは、構成案を出させるとか、そういう使い方が望ましいだろう。

42.6 活用事例：文章を読ませる

言葉を扱うのが得意なのがLLMなので、書かせるだけではなく読ませることにも適している。たとえば原稿についてであれば「添付は原稿である。この原稿は十分だろうか？ もっと膨らませる余地はないだろうか？ 重複はあるだろうか？ 読みやすくすることはできるか？」などと指示すれば、原稿のレビューをしてくれる。

これまで述べたとおり正確性は過信してはいけないが、技術ブログのレビューをさせるのもある程度有用だ。あなたの抜け漏れや、間違いの指摘や、文章の発展性について聞くと、すらすら答えてくれるはずだ。

他にも定番の使い方がある。生成AIに興味があるなら論文を読ませればいい。今時はarxivに様々な論文^{*9}が大量に公開されている。少なくとも生成AIの最先端で外部公開されている論文ならまず大半がarxivにある。

大量にある論文をそのまま読むのはしんどい。分量も多いし、英語で書かれているので、日本語に翻訳したり要約してもらったり、質疑応答をさせるのだ。そうして本当に読みたい論文があれば、原文を読むなりすればいい。

筆者が先日注目していた論文にHtmlRAG^{*10}というものがある。その論文を添付して「解説お

*9 正確には査読前論文。pre printと呼ばれるもの

*10 <https://arxiv.org/html/2411.02959v1>

願い」と投げてみた。

Claude Sonnet の返答

この論文は「Htm1RAG」という新しい手法についての研究を説明しています。以下が主なポイントです：

1. 研究の背景と課題：

- 従来の RAG(Retrieval-Augmented Generation) システムは、HTML をプレーンテキストに変換して使用
- この過程で、HTML の構造的・意味的な情報が失われる問題がある
- 例えば、表形式のデータや見出し構造などが失われる

2. 提案手法 (Htm1RAG)：

- HTML をそのままの形式で知識源として使用
- 主な課題は、HTML が長すぎることと余分なコンテンツ (JavaScript、CSS など) を含むこと
- これに対して以下の 3 つのステップで対処：

a. HTML クリーニング：

- CSS や JavaScript の削除
- 冗長な HTML 属性の削除
- 単一の入れ子タグの統合

b. ブロックツリーの構築：

- DOM ツリーをより粗い粒度のブロックツリーに変換
- 効率的な処理のため

c. 2 段階の HTML 割り込み：

- 埋め込みモデルによる粗い割り込み
- 生成モデルによる細かい割り込み

3. 実験結果：

- 6 つの QA データセットで評価
- 従来の手法と比較して優れた性能を示す
- HTML の構造情報を保持することの有効性を実証

4. 主な利点：

- HTML 構造による意味的情報の保持
- LLMs の事前学習で HTML に触れているため、追加学習不要
- 様々な文書フォーマット (PDF、Word など) から HTML への変換が容易

この研究は、RAG システムにおいて HTML を直接利用することの有効性を示し、新しい研究の方向性を開いた点で意義があります。

雑な指示の弱点は、論文をどのように解説してくれるかはランダムになってしまうことがある。解釈内容はまず変わらないが、本当に求めているタイプの解説が帰ってくるとは限らないし、投げたびに解説の仕方や、その範囲が変わってしまう。

つまり、聞きたい内容はこちらが明示的に指示しなければならない。落合陽一先生の論文要旨のテンプレートを使って試してみよう。201x 年代に他の人が公開しているスライドが slideshare にある^{*11}ので、これを参考にしてみよう。

落合先生メソッドを使ったプロンプト

添付した論文を以下の項目に沿って説明して。なるべく箇条書きではなく、読みやすい文章で。

^{*11} <https://www.slideshare.net/ssuser344829/ss-75062659>

- * 研究の主な目的と問題意識
- * 使用された研究手法
- * 主要な発見や結論
- * 研究の意義や限界
- * 今後の研究への示唆

個人的に箇条書きで出力されるのはあまり好きじゃないので、「なるべく箇条書きではなく、読みやすい文章で。」と付けています。

Claude Sonnet の返答

この論文を要点に沿って説明させていただきます。

「どんなもの？」

この研究は、Retrieval-Augmented Generation (RAG) システムにおいて、HTML をそのまま知識形式として利用することを提案しています。従来の RAG システムは、HTML をプレーンテキストに変換して利用していましたが、その過程で構造的・意味的な情報が失われていました。特に表形式のデータや見出し構造などが崩れてしまう問題がありました。そこで、HTML をそのまま活用しつつ、効率的に処理するための「HtmlRAG」というフレームワークを開発しました。

「どうやって有効だと検証した？」

6つの質問応答データセット (ASQA、Hotpot-QA、NQ、Trivia-QA、MuSiQue、ELI5) を用いて検証を行いました。これらのデータセットは、曖昧な質問、マルチホップ推論が必要な質問、実際のユーザーからの質問など、様々なタイプの質問を含んでいます。実験では、従来のプレーンテキストベースの手法と比較して、HtmlRAG が一貫して優れた性能を示しました。

「先行研究と比べてどこがすごい？」

最大の革新点は、HTML の構造情報を保持したまま効率的に処理できる点です。従来の RAG システムでは、HTML → プレーンテキスト変換による情報損失が避けられませんでしたが、本研究では HTML Cleaning、HTML Pruning という 2 段階の処理により、構造情報を保持しながら効率的な処理を実現しました。特に、block-tree 構造を用いた枝刈り手法は、HTML の階層構造を考慮した効率的な処理を可能にしています。

「議論はある？」

主な議論点として、HTML の処理にかかる計算コストが挙げられています。HTML は生のままでは非常に長く、処理が困難ですが、研究チームは効率的な圧縮・枝刈り手法を提案することでこの問題に対処しています。また、将来的に LLM の性能が向上するにつれて、HTML の構造的な理解がさらに改善される可能性についても議論されています。

「技術や手法の肝は？」

技術的な核心は、以下の 3 つのコンポーネントにあります：

1. HTML Cleaning: 不要な CSS、JavaScript 等を除去する前処理
2. Block-tree 構造: HTML の階層構造を効率的に表現する独自のデータ構造
3. 2 段階の Pruning: エンベディングモデルと生成モデルを組み合わせた効率的な枝刈り手法

「次に読むべき論文は？」

明示的な推奨はありませんが、論文中の引用から以下の領域の論文を読むことが有用と考えられます：

- RAG システムの基礎研究（例：LangChain、LlamaIndex に関する論文）
- 構造化データの理解に関する研究（特に HTML、表形式データの処理に関するもの）
- LLM の長文コンテキスト処理に関する最新研究

Claude Sonnet や gpt-4o や o1 のような十分に賢いモデルであれば、これくらいのシンプルなプロンプトでも、読みやすいバランスの要約がでてくる。おそらく落合先生のメソッドなんかも Claude や GPT の学習データにある程度は含まれているのだろう。

もちろん、ここからさらに凝った指示を与えることは可能だ。たとえば、「定量的・具体的なデータを含めること」「論文に書かれた事実のみを使うこと」などだ。論文要約のプロンプトは探せば

色々出てくるので、試行錯誤してみるといいかもしれない。

とは言え、この手の要約だけだと、本当に軽めの要約にしかならないため、「読むべきか？」のフィルターとして使い、本当に読みたい論文なら、ここからさらに色々質問を投げるといいだろう。

- 埋め込みによる刈り込みの詳細を教えて。クエリの埋め込みとの類似度を見ている？ でもそれだと、本来重要な文章を消してしまいそう
- クエリの埋め込みの方で工夫してることはない？ そのまま埋め込み表現に変換してる？ 「HtmlRAGについて教えて」みたいなクエリをそのまま？
- 粗い粒度のブロックツリーは、サイズだけを気にしている？
- ブロックツリーはどういうデータ表現？
- 埋め込みは HTML のタグとかも含まれる？ それとも文章だけ？ クエリには HTML のタグが含まれないからその分、違いが出ると思うんだけど
- 埋め込みの作成はともかく、LLM を何回も走らせるコストが問題になると思うけど、そこについての教えて
- 埋め込みでの刈り込みで 8Ktokens まで減ってるんなら、いっそそのままで良くない？

最初に HtmlRAG を知ったときはこのような質問を投げていた。結果に関しては紙面を取り過ぎるので割愛しよう。

筆者はしがない専門学校卒なため論文には疎いのだが、対話型 AI を活用すれば、自分に合った説明をいくらでも引き出せる。質問が思いつかない場合は、「あまり理解できなくて質問が思いつかないんだけど、いい質問例はある？ 10 個上げてみて」と聞いてみるのも有効だ。

Claude Sonnet が挙げた質問例

- * 従来の RAG システムが HTML をプレーンテキストに変換する際に、具体的にどのような情報が失われているのですか？ 実例を挙げて説明してください。
- * HTML Cleaning の工程で、なぜ CSS と JavaScript は除去しても問題ないと判断したのでしょうか？
- * block-tree の粒度 (granularity) は性能にどのような影響を与えますか？ 最適な粒度はどのように決定されましたか？
- * 2 段階の Pruning プロセスで、なぜエンベディングモデルと生成モデルの両方が必要なのでしょうか？ どちらか一方では不十分なのですか？
- * 実験で使用された 6 つのデータセットはそれぞれどのような特徴を持っており、なぜこれらが選ばれたのですか？
- * HTML 形式を保持することで、特にどのようなタイプの質問に対する回答が改善されましたか？
- * 計算コストの観点から見て、従来のプレーンテキストベースの手法と比較してどの程度のオーバーヘッドがありますか？
- * LLM の文脈窓 (context window) の大きさは、このシステムの性能にどのように影響しますか？
- * HTML の構造情報を保持することは、マルチホップ推論を必要とする質問に対してどのように役立っているのですか？
- * この手法は、HTML 以外の構造化文書（例：PDF や Word 文書）に対してどの程度適用可能でしょうか？

これらを提案された。いっそ、論文の説明プロンプトに入れてもいいくらいだ。

質問 10 個を上げさせるのを追加したバージョンのプロンプト

添付した論文を以下の項目に沿って説明して。なるべく箇条書きではなく、読みやすい文章で。

- * 研究の主な目的と問題意識

- * 使用された研究手法
- * 主要な発見や結論
- * 研究の意義や限界
- * 今後の研究への示唆
- * この論文に対して、質問例を10個

Claude Sonnet の返答を抜粋

質問例：

- * なぜHTMLからプレーンテキストへの変換で情報が失われる問題が重要なのでしょうか？
- * HtmlRAGの2段階剪定プロセスの具体的なメリットは何ですか？
- * 提案手法は従来のRAGシステムと比べて計算コストはどの程度増加しますか？
- * HTMLクリーニングプロセスで、どのように重要な情報と不要な情報を区別していますか？
- * ブロックツリーの粒度はシステムのパフォーマンスにどのような影響を与えますか？
- * 提案手法は非構造化または不適切に構造化されたHTMLに対してどの程度ロバストですか？
- * 実験で使用された6つのQAデータセットはどのような基準で選択されましたか？
- * HTMLの構造情報保持が特に効果的だったタスクやケースはありましたか？
- * 将来的なLLMの進化により、この手法の有効性はどのように変化すると予測されますか？
- * この手法は多言語のコンテンツに対してどの程度効果的ですか？

先ほどのときは違うため、絞り込みが出来ておらず様々な質問を思いつくままに書いてくれているのだが、それでもとっかかりとしては役立つだろう。自身の知識レベルなどを使ってさらに絞り込んでみると、より適切な質問例が帰ってくるはずだ。

他にも「この論文を理解するために勉強すべきことはなに？」のような質問をすると渉るだろう。

42.7 活用事例：ドキュメントを元に音声を作る

論文などのドキュメントを食わせて問答する目的にはGoogle社のNotebookLMもおすすめである。これは無料でも問題なく使える。ClaudeやChatGPTにない機能として、Podcast風の音声データを作てもらう機能がある。

対話型AIはどうしても文字データがメインになってしまい、たとえば移動中に使うのには向いてないなどの問題がある。音声であれば移動中に聞くということもできる。

なおChatGPTなどでは音声通話で対話型AIを使える機能が実装されているため、スマホで音声のやりとりをすることも可能だが、どうしても一定の遅延が生じやすいため、本当に実用に足るにはもう少しだけかかりそうだ。

42.8 活用事例：問題を作ってもらう

対話型AIは学習課題の生成にも活用できる。「TypeScriptのconditional typeについて理解を深めたい。何か問題を出してほしい」と投げてみた。

Claude Sonnetの返答

TypeScript の conditional types について、実践的な問題を通して学んでいきましょう。まずは基本的な問題から始めて、徐々に難しくしていきます。

1 問目です:

```
```typescript
// 以下の型を完成させてください:
// 型 NonNullable<T> を実装してください。
// この型は、型引数 T から null と undefined を除外した型を返します。

type MyNonNullable<T> = // ここを実装してください
```

この問題を解いてみましょうか？ ヒントが必要な場合は言ってください。

この問題を解いた後、より複雑な問題に進んでいきたいと思います。まずはこの問題にチャレンジしてみてください。

「ヒントをください」と言えばヒントも提供してくれる。

Claude Sonnet の返答

TypeScript の conditional types では、`extends` キーワードを使って型の条件分岐を行うことができます。

ヒントをいくつか出します：

1. 基本的な構文は：

```
```typescript
T extends 条件 ? True の場合の型 : False の場合の型
```
```

2. `null` と `undefined` を除外するには、`T` が `null` または `undefined` に代入可能かどうかをチェックします

3. 型の判定で「または」を表現したい場合は ``type1 | type2`` という形で書けます

これらのヒントを元に、型パラメータ `T` が `null` や `undefined` に該当する場合と、そうでない場合で分岐して考えてみましょう。

まだ難しければ、もう少し具体的なヒントを出すことができます。どうでしょうか？

このやり方の良い点は、自分にあった問題を出してもらえることだ。実は TypeScript の型については問題集が既にある<sup>\*12</sup>のだが、対話型 AI なら、もっと簡単な問題を出してもらえる。

「TypeScript の配列の型について問題をだしてください」と投げると、より簡単な問題を出してくれる。

Claude Sonnet の返答

TypeScript の配列の型に関する問題を作成します。基本から応用まで段階的に学べる問題を出題しましょう。

```
```typescript
// 問題 1:
```

^{*12} <https://github.com/type-challenges/type-challenges>

```
// 以下の配列に適切な型アノテーションをつけてください。
let fruits = ['apple', 'banana', 'orange'];

// 問題2:
// number型とstring型の両方を含む配列の型を定義してください。
let mixed = [1, 'hello', 2, 'world'];

// 問題3:
// 以下のような形のオブジェクトの配列の型を定義してください。
let users = [
  { id: 1, name: 'Taro', age: 25 },
  { id: 2, name: 'Hanako', age: 30 }
];

// 問題4:
// 配列の中に別の配列（ネストされた配列）を含む場合の型を定義してください。
let matrix = [
  [1, 2],
  [3, 4],
  [5, 6]
];

// 問題5:
// readonly修飾子を使用して、変更不可能な数値配列の型を定義してください。
```

```

解答例を見たい問題の番号を教えてください。また、追加で別の問題も用意できますので、お申し付けください。

もっと基礎の問題を出してほしいなら「TypeScriptの型に関する基礎の問題をだして」と指示すればいいだろう。より基礎的な内容の問題をだしてくれる。

この活用方法で注意すべき点は、AIが生成する問題や解答にハルシネーションと呼ばれる嘘の情報が含まれる可能性があることだ。TypeScriptのような情報が豊富な技術であれば比較的安全だが、それでも検証は必要だ。TypeScript Playground<sup>\*13</sup>で実際に試してみるなど、必ず確認作業を行うべきだ。

### ■コラム: Claude Desktop

これはあまりにもタイムリーすぎるため、あまり詳細には踏み込まないが、ClaudeのPC/Mac向けのアプリであるClaude Desktopは、11月末にMCPという機能に対応した。

MCPはModel Context Protocolの略で、簡単に言うとそのプロトコルに従ったソフトウェアに、任意の機能を追加できるというものだ。たとえばデータベースにアクセスする機能、ファイルシステムで直接ファイルを読み書きする機能、ソースコードを実行する安全な環境、記憶を保存する機能など、ありとあらゆるもののが考えられる。

これまでの対話型AIはそれを提供している会社が頑張って機能を開発してリリースしてきた。当然のことだが開発には時間がかかるし、自分がほしい機能が実装されるとは限らないが、MCPに対応した対話型AIアプリであれば、MCPに準拠した機能拡張の恩恵を受けら

\*13 <https://www.typescriptlang.org/play>

れる。

MCP は Anthropic 社が主導して生みだしたが Claude 専用のものではない。執筆時点では登場して二週間も経っていないが、MCP の開発は極めて活発だ。既に色々な人が MCP による機能拡張を作っている。この動きは止まることがない。

筆者は AI エージェントと呼ばれるものを作っている。LLM を活用しつつ様々な I/O を行って対話型 AI 以上のことを行っている。その観点で見ると MCP はとても都合がいい。MCP に対応した AI エージェントは、コミュニティが開発する様々な機能をそのまま AI エージェントに組み込めるのだ。

こういった事情があるため、OpenAI や Google も対応せざるを得ないはずだ。

読者が、この機能の恩恵にあずかれるのがいつ頃になるかは分からないが、機能拡張を簡単にインストールできるバイナリなんかも配布されるようになるだろうからそういうのを使えるようになるはずだ。もちろんこのバイナリの安全性はちゃんと確かめるべきだが、安全にまつわる情報も、公式・非公式問わずやりとりされるようになるだろう。

## 第 43 章

---

# 自分をメンテナンスするために月に1回有給を取ろう

現代の働く社会では、忙しい日々の中で自分自身を見失ってしまうことがよくあります。特に、家族を持っている方にとって、土日は子供や配偶者との時間に費やされることが多く、自分のための時間がほとんど取れません。平日は当然、仕事に追われ、家事や育児との両立で疲労が蓄積します。そこで提案したいのが、月に1回「自分のための有給休暇」を取ることです。この時間は、心身をリフレッシュさせるために、自分自身のためだけに使います。

### 43.1 土日が家族対応でつぶれる現実

家族持ちの多くの人にとって、土日というのは、平日に溜まった（残った）家事をこなす時間であり、子供の世話や習い事の送迎、家族とのイベントなどでほぼ埋まります。もちろん、家族との時間は大切で、一緒に過ごす瞬間はかけがえのないものであることに異論はありません。しかし、その一方で、週末が終わる頃には「自分のための時間が全くなかった」と感じることもしばしばです。

家族を持つということで多くのタスクが生じ、それに応じて自分自身の優先順位が後回しにされがちです。結果として、知らず知らずのうちに心身の疲れが蓄積し、ストレスが溜まってしまうのです。土日が家族のための時間であるなら、では自分自身をメンテナンスする時間はどこにあるのでしょうか？

### 43.2 平日は仕事で忙しい

平日は朝から晩まで仕事に追われ、家に帰れば家庭の仕事が待っています。家事や育児を協力しながらも、働く親としての負担は決して軽くはありません。リモートワークが普及したことで、通勤の負荷が減った面はあるにせよ、それでもギリギリの時間まで仕事をしまい、自分の時間が増えた実感がないのでは？ さらに、仕事とプライベートの境界が曖昧になり、パチッと切り替えが効かない今まで仕事のストレスが何となく抜けきらないまま毎日を過ごしてしまうことも増えました。

それに加えて、仕事が終わった後に趣味やリフレッシュの時間を取りたいとしても、疲れ切っていて何もできないこともあります。「自分の時間」を意識して作ろうとしても、結局は家族や仕事のための時間になってしまい、自分のためだけの自由な時間が少なくなっているのが現状です。

### 43.3 自分のために月に1回、有給を取ろう

こうした状況を改善するために、月に1回「自分のためだけの有給休暇」を取ることをおすすめします。この休暇の目的は、自分をリフレッシュさせ、心身のバランスを取り戻すことです。誰のためでもなく、完全に自分のためだけに使う時間を確保することで、心身の健康を保ち、仕事にも家庭にも前向きに向き合えるようになります。

この日をどう使うかは、完全に自由です。例えば、以下のような過ごし方が考えられます。

- 映画を見に行く

平日の日中は映画館も混雑しておらず、リラックスして映画を楽しむことができます。気になる映画をじっくりと観賞し、非日常の物語に浸ることで、リフレッシュされます。子供と一緒に見に行くアニメ映画や話題の映画もいいですが、自分しか観ないような映画もちょっといいですね。

- 美味しいものを食べに行く

普段は家族や仕事のスケジュールに合わせて食事をすることが多いですが、たまには自分だけのために贅沢なランチを楽しむのもいいでしょう。好きなレストランでの食事や、新しいお店を開拓することで、自分へのご褒美を感じることができます。

- 家でぼーっと過ごす

外出しなくとも、自宅で何もせずにぼーっと過ごすのも立派な過ごし方です。好きな音楽をかけてゆっくりと過ごしたり、読書や昼寝をしたり、日常の忙しさから解放される時間は、心を落ち着けてくれます。

- 趣味に没頭する

忙しい日常ではなかなか時間が取れない趣味に没頭するのもおすすめです。たとえば、プラモデル作りや料理、ゲームやハンドメイドなど、自分が本当に好きなことに集中することも、今までもないステキな時間の使い方ですね。

### 43.4 月1回の休暇がもたらす効果

月に1回の自分だけの時間を持つことで、普段のストレスを解消し、自分を大切にする感覚が戻ってきます。忙しい日々の中で忘れてしまいがちな「自分のための時間」を確保することは、長い目で見れば心身の健康維持に繋がり、家族や仕事へのエネルギーも充実します。月1回のリフレッシュを習慣化すれば、日常の小さなイライラや疲れを溜め込まずに済むようになります。

また、こうした自分のための休暇を取ることは、家庭や仕事でのパフォーマンスにも良い影響を与えます。リフレッシュした状態で家族と向き合い、仕事にも集中できるようになるため、結果的に周りの人たちにもプラスの影響を与えることができるのです。

## 43.5 最後に

自分のために月に1回、有給休暇を取るというのは、非常にシンプルですが効果的な自己メンテナンス法です。忙しい日々の中で、ぜひ「自分を大切にする日」をスケジュールに組み込み、心身のリフレッシュを図りましょう。家族や仕事に全力を尽くすためにも、まずは自分自身を労わることが大切です。

まあなんだかんだで月1回とはいはず、四半期に1回くらいになっちゃうのですが・・・それはそれで、その分貴重な休みとなる面もあるでしょう。どうせ有給なんてたっぷり余ってますよね？来月からやってみませんか？

# HP と MP を別々に回復させよう

日々の生活において、私たちは体力と精神力の両方を消費しつつ生きてています。ドラクエをはじめとしたゲームでも、HP と MP が定義されていて、HP (HP: Hit Point) がなくなると行動不能になりますが、MP (Magic Points) がなくなると魔法を唱えられなくなりますね。

本節ではそれになぞらえ、HP と MP を定義します。HP (Health Point) は肉体的なエネルギーを表し、運動や仕事、家事などの体を動かす活動によって消耗されます。肉体的な疲労で減ると考えてください。一方の MP (Mental Point) は精神的なエネルギーを示し、仕事や学業のストレス、人間関係の悩み、心や頭を使うことによって減少します。

健康でバランスの取れた生活を送るためには、HP と MP を維持することが不可欠です。しかし、これら二つのエネルギーは異なる特性を持ち、それぞれに最適な回復方法が必要です。ついメンタル面の回復に目が行きがちですが、HP の回復も大事です！ HP も MP も現実世界では、ステータス画面には出てきません。

## 44.1 精神力 (MP) の回復と推し活

MP の回復には、自分が心から楽しめることにつぎ込むのがベストですね。好きなアーティストのライブに行ったり、聖地巡礼をしたりと、自分の「好きなこと」に没頭することで、心がリフレッシュされ、ストレスが解消されます。「推し活」って言い換えてもいいですね。ライブ後の満ち足りた精神状態を思い出してください。MP がモリモリ回復することを感じられますね。

しかし、MP の回復には最適なのですが、HP は削られてしまいます。ライブやイベントでは多くの人が集まり、長時間の立ちっぱなしや移動、声を出して応援するなど、意外に体力を消耗します。イベントが終わる頃には、気持ちは満たされていても、体はクタクタになっているという経験は、多くの人がしたことがあるのではないでしょうか。つまり、MP の回復はできても、HP は逆に消耗してしまうことがあります。MP が回復すれば、多少の HP の損耗は覆い隠せてしまます。メンタルが高揚して、多少の疲れも吹っ飛ぶというやつですね。しかしそれが常態化してしまうと、マズいことになります。

## 44.2 体力 (HP) の回復には何が必要か？

HP の回復は、当たり前ですが、十分な休息や睡眠、適度な運動、そして栄養バランスのとれた食事が鍵となります。これにより、身体がリフレッシュされ、翌日またはそれ以降の活動に向けたエネルギーが蓄えられます。たとえば、寝不足が続くと、どれだけリラックスした時間を過ごしても、体力が回復しないため、疲れが抜けないと感じることがよくあります。

体力をしっかり回復させるためには、十分な睡眠時間を確保することが最重要です。1日7~8時間の睡眠を取りましょう。寝貯めは意味がないという話もありますが、睡眠時間は日常的に不足気味な人が多いことでしょうから、たまに多めに寝ることで睡眠の借金を多少なりとも返済できるかもしれません。

また、適度な運動も体力回復に役立ちます。運動をすることで血流が良くなり、疲労物質の排出が促進され、体全体が活性化されます。過度な運動は逆効果ですが、ストレッチや軽い散歩などは、体を休めつつエネルギーの充電を助けています。

食事も体力回復において重要な要素です。栄養バランスの取れた食事は、疲れた体を修復し、必要なエネルギーを供給します。特にビタミンやミネラル、タンパク質が豊富な食事を意識的に摂ることで、体力回復を促進するでしょう。

### 44.3 何もない日も大切

HPとMPを回復するためには、それぞれに合ったアプローチが必要です。しかし、現代の忙しい生活の中で、意識的にどちらも回復することは難しいかもしれません。特に、精神力を回復させようと「推し活」や趣味に時間を割いても、体力が追いついていないと、疲れを貯めることになってしまいます。MPが充実していると予想以上に動けてしまうこともあります、確実に疲れは貯まります。

一方で、HPの回復を優先するあまり、空き時間を休養に充てるような生活、仕事仕事仕事休養のみでは、MPが枯渇してしまいます。

何もない日とは、特定の予定や義務に追われることなく、体を休め、心を解放する時間を意図的に設けることです。例えば、何も考えずにゆっくりと過ごす日、家でリラックスしながら映画を観たり、お風呂にゆっくり浸かる日などが挙げられます。このように、特に外出や積極的な活動を避け、体と心をリセットすることで、HPとMPの両方をバランス良く回復させることができます。

また、デジタルデトックスも有効です。日々スマートフォンやパソコンに長時間触れていることで、私たちの脳は絶えず刺激を受け、知らず知らずのうちに疲労が蓄積しています。SNSやメールを気にせず、自然の中で過ごす時間や、ただ静かに過ごす時間を設けることで、脳がリフレッシュされ、精神的な疲れを取ることができます。SNSを見ていることがMP消費につながることもありますね。

### 44.4 ちゃんと寝る

HPとMPの両方を意識的に回復させる必要があります。体力だけでなく、精神力の回復も重要ですし、その逆もまた然りです。片方だけに焦点を当てすぎると、最終的にはバランスを崩してしまい、低い方に引っ張られてしまうという事態に陥ることがあります。したがって、推し活や趣味に没頭してMPを充電する日もあれば、何もないで体を休める日を設けてHP回復に専念する比を作ることも必要でしょう。人間にもHPとMPがあることを認識し、別々かつバランス良く回復させることを意識することで、日々を平穏かつ元気に過ごすことができるでしょう。

薬草で HP は回復しますが、MP は回復しません。薬草を使いながら徹夜行軍するのではなく、ちゃんと宿屋で睡眠をとることが必要ですね。

## 第 45 章

---

# 幅を広げる学習のススメ

こうの@hk\_it7

## 45.1 深く学ぶか広く学ぶか

学習時のテーマ選びはどうしているでしょうか？

- 使っている技術の深堀り
- 最新技術を試す
- 作りたいものを作ってみる（そして周辺技術を拾う）

他にも、人それぞれのテーマ選びで興味の持ったものを学んでいくことが多いかと思います。

一方で能動的なテーマ選びのみを続けていると、未知の技術の開拓や知識の幅を広げる機会が少なくなってしまいます。

この章では、知識の幅を広げるために実践していることを理由とともに解説していきます。

## 45.2 なぜ広く学ぶのか

まず、なぜ広く学ぶのかについて考えていきます。

エンジニアリングは専門的な知識が必要とされていますが、対象は多岐にわたります。

プログラマであれば、言語やネットワーク関係、インフラ関係などが大きなジャンルとしてありますね。さらに言語では C 言語、Java、JavaScript など無数に領域が存在します。例えば、その中の 1 つである Java を極めたからといってエンジニアとして優れているとは言い難いと思います。なぜなら、多くのエンジニアは Java を使うことが目的ではなく、ソフトウェアを開発するための 1 要素に過ぎないからです。

また、多くのエンジニアは 0 から発明するのではなく、既存技術の組み合わせで開発を進めています。

開発を行っているとエラーであったり調査であったりでどうしても周辺技術の知識が必要になります。一方で未知であっても人間には過去の経験や知識から推測するという能力があります（もちろん推測だけで進めてはいけませんが）。

未知に遭遇したときに ChatGPT 等の生成 AI 技術の台頭により、最新の技術以外はある程度、AI に聞けば良い時代に変わりつつあります。

「既存技術の組み合わせ」と「AI との協業」を前提にすれば、今後のエンジニア像として「広く浅い知識を身につけて、場面（コンテキスト）に合わせた対応（技術選定/課題解決）ができるこ

と」を求められていくと私は考えています。

どういうことかと言うと、「既存技術の組み合わせ」であれば、「AIとの協業」のもと、ある程度の回答が得られます。しかし、これには2つの課題があります。

1つ目は、ちゃんと聞き出せるかという点です。2つ目はコンテキストをAIに伝えることの限界です。

それぞれ見ていきましょう。

### 45.2.1 AIにうまく聞き出せるか

既存技術を自ら再実装してしまうことを「車輪の再発明」と呼びます。意図的に実施する場合は良いですが、業務的な開発としては無駄なコストと考えられるため、避けたほうが良いとされています。

なぜ、車輪の再発明をしてしまうのでしょうか？これは既存技術を知らなかったこと、あるいは聞き出せなかっことによる悲劇だと考えます。

既存技術を知らなかったことについては人間の限界があるため、仕方がないです。しかし聞き出せなかっただという部分は改善の余地があります。

聞き出せなかっただ理由は「聞くという発想に至れなかっただ」や「うまく質問できなかっただ」が考えられます。筆者はこの2つの原因を解決するのが、広い知識であると考えています。

少しでも似たような話を聞いたことがあれば、それを手がかりにAIにたずねることで、過去に開発された「車輪」を導き出せます。

### 45.2.2 コンテキストをAIに伝えることの限界

「AIにうまく聞き出せるか」とも重複しますが、AIに聞く際にはコンテキストが重要になります。課題に対して周辺技術としてどのようなものがあるかであったり、もっと大きく見ればビジネス的な制約の有無なんかもコンテキストに当たります。

開発には、さまざまな制約があることが多いです。その制約を常にすべてプロンプトに設定することは現実的ではなく、AIに伝える制約事項の取捨選択が必要になります。

その際に、この質問であればこの情報が必要かな、というのは人間が判断する必要があります。必要な情報もAIに聞いて進めていくという手法もありますので、まずはこの方法を試してみるは良いと思います。

また、多少知っているものであれば、選択肢を人間が提示して、AIにメリット・デメリットを列举して検討材料にするという使い方もできます。

このようにAIと協業する上でも、広い知識があるかないかで大きな差が生まれます。

さて、知識の幅を広げるためにはどうしたら良いでしょうか？

## 45.3 知識の幅を広げるきっかけづくり

幅を広げるためにはきっかけづくりが重要です。きっかけづくりを能動的に行い、得た情報の選択（テーマ選び）を受動的に行えば、今まで知らなかったことに対しての学習の機会を得ることができます。

きっかけづくりの例には次のようなものが挙げられます。

- 本屋に行って話題の本、新刊を買う
- 技術系雑誌を定期購読する
- 技術イベント（LT会、カンファレンス）に参加する

少し個別に深堀りしていきます。

### 45.3.1 本屋に行って話題の本、新刊を買う

ここでの本屋はネットの本屋（Amazon等）ではなく、リアルな本屋を指しています。出版不況と言われて久しく、段々とリアルな本屋が減りつつありますが、新しいジャンルを開拓するには持って来いの場所です。

本屋に並んでいるのに、全く聞いたことのない技術というのは幅を広げるチャンスに満ちています。なぜなら、出版されるには、ある程度の期間と需要が必要であるためです。つまり、本屋で並んでいる時点で、世間に認知され、ある程度利用が進んでいる技術であると言えます。

にもかかわらず、全く聞いたことがないとなると、その技術ジャンルについてアンテナが張れない可能性があります。もちろん全てに精通する必要はないので、アンテナが張れていなければ問題であるというわけではありません。しかし、詳しくない部分を知っておくことは、開発業務を行う上で重要であると考えます。

### 45.3.2 技術系雑誌を定期購読する

書籍としては技術系雑誌も有効だと考えています。定期購読をすることで受動的なテーマ選びを可能にします。雑誌は毎回異なる特集が組まれ、広範な話題に触っています。紙面の関係上、手を動かして試すのにちょうどよい量になることが多いことも魅力です。

類似するものとして、企業等が出している合同誌タイプの技術同人誌も同じような効果があります。この本もそうですね。

いろんな人が書いているからこそ、広い知識や考え方を得るきっかけになります。

### 45.3.3 技術イベント（LT会、カンファレンス）に参加する

特にオンラインイベントだと良いと考えています。複数トラック構成でも、そうでなくとも、興味の薄いトピックの時間というのはけっこう発生します。その興味の薄いトピックこそが、知識の幅を広げるきっかけになります。

オンラインでの視聴では、興味の薄いトピックが並んだときに、つい他のことをやってしまいがちです。「後日アーカイブで見れば良い」も、実現しないことがけっこうあるかと思います。

オフラインイベントであれば、とりあえず聞いておこうという判断をする可能性が高くなり、結果として、新しいことを知ることができます。

## **45.4 知識を広げたあとは？**

ここまで紹介した内容は「ある程度、受動的に知識を広げる手段」でした。広げるだけ広げても実践で役立つものになるとは限りません。

そのため、気になったものについては、深堀りをすることをオススメします。知識の深堀りをする方法は……他の誰かがきっと書いていてくれると思います！

本章を読んで、知識を広げることで、さまざまな場面でベターな選択ができるようになることを願っています。

## 第 46 章

# 学べる環境に身をおいてしまえば学ばざるを得ず、強制的に学べる

### 46.1 はじめに

この章は要約すると、「身銭を切ると思った以上に学べる」みたいな話です。

筆者は、社会人になって最初に入った会社が上場企業だった為、従業員持株会制度がありました。

なお、従業員持株会制度については以下の説明が参考になります（ChatGPT 調べ）。

従業員持株会制度は、企業がその従業員に対して、自社株を購入する機会を提供する制度です。この制度の主な目的は、従業員が自社の経営に関与し、その成果に直接的な関心を持つことを促すことにあります。以下はその特徴です：

従業員の関与促進：従業員が株を持つことで、会社の業績に対する関心が高まります。これは、従業員が自社の成功に直接的な影響を受けるためです。

利益の分配：従業員持株会を通じて、従業員は配当や株価上昇の恩恵を受けることができます。

長期的な雇用関係の促進：従業員が自社の株を持つことで、会社への忠誠心が高まり、離職率の低下に繋がることがあります。

税制上のメリット：一部の国では、従業員持株会を通じて株を購入することに税制上のメリットがあります。

実施方法：通常、従業員持株会は特別な条件で株を従業員に提供することが多いです。例えば、割引価格で株を購入できる、または特定の期間勤務した従業員だけが購入資格を得るなどです。

この制度は、従業員と会社の双方にメリットをもたらす可能性がありますが、株価の変動リスクや従業員の投資ポートフォリオの多様性の欠如などのリスクも伴います。

筆者が新人で入った会社が良かったのか悪かったのかについては、メリット・デメリットがそれぞれ色々とあったので、手放しで「全部良かった！」という感じでもないのですが、メリットだったと感じる中でトップレベルに大きかったのは「この制度のおかげで株や金融証券についての知識がある程度身についた」点だと思っています。

### 46.2 持株会の何がそんなに良かったのか

持株会の最大のメリット（であり、人によっては最大のデメリット）は、「株を買う事になる」点です。

特に、上場企業の株を持株会で購入する場合は、購入した株が市場で常に売買されている関係で常に株価が上下動を繰り返します。

一般的に、自分ごとになっていない事象についてはあまり真剣に意識を注ぎ続けられないものですが、自分のお金をそこに入れた瞬間、自社の株式の動きは「自分ごと」になり、ずっとそれを気にし続けるようになるケースが多いのではないかと思っており、私もご多分に漏れずそうでした。<sup>\*1</sup>

当然、自社の株価や日経平均、TOPIXなどの動きを日常的に追い続けると、最初はわからなかった経済や世の中の株式・債権の仕組みなどを体感として学んでいく事ができます（株価の上下にともなうテンションの上げ下げを喰らいながらですが）。

そしていつしか、金融リテラシーが身についていくわけです。

## 46.3 「これから上がる株」みたいな本一杯あるけどさあ

そんな本は世に一杯あります。株で爆益だとか、億り人だとか……

ただ、筆者としてはこういった個別株を個人で持つ事の最大のメリットについては、そういった書籍で語られがちな「貯金よりも利率が良い」や「儲かる」といった煽りには違和感を感じていて、たぶんそうではないのだろうな と思っています。

というのも、具体的には株式を軽くでも買ってみると、身銭を切って継続的に経済動向を監視せざるを得ない状況に体を置けます（置かないといけなくなります）。その結果、「社会人としての経済リテラシーがガンガン乗る」所が最大の妙味なんじゃないかな、そしてこれは「学び」なのだ。と感じています。

当然、「学び」が進む仮定で一般的な市場参加者が気づけなかった着眼点に気づけるかもしれませんし、気づいて株式を購入すれば、ある程度美味しいリターンを得られるかもしれません。ただしそれは「学び」の結果の産物であって、学べたからこそ成果が出て、結果として投資リターンも出ている という捉え方が正しいんじゃないかなー と個人的には思っています。

投資はあくまで自己責任にはなりますが、余剰資金<sup>\*2</sup>で色々試し、学んでみると、より豊かな「学び」が待っているのではないでしょうか。

<sup>\*1</sup> 実はこういった経済に連動する金融商品は、細かく見ずにはたらかすのが一番賢いという話もあるのだけど……

<sup>\*2</sup> 余剰資金というのは、『失っても痛くないと思える』金額の範疇に入る資金の事です

## 第 47 章

# 資格を学びに組み込むと段階的に学びやすい

@ditflame

社会人になってからの「学び」方で、よく使われがちななものに「資格」(資格試験の受験)があります。

筆者は資格試験の大きなメリットの一つとして、「適切に設計された順で、段階的に学びを進める事ができる」点があるように思います。

この章では、様々な資格試験を通して、「簡単な論点(大事な基本論点)から応用論点に向けて、順に学ぶ事ができる」事を確認してみましょう。

なお、この章では主に次の資格試験を取り上げます。

- 簿記
- 販売士
- 電気通信の工事担任者・電気通信主任技術者

### 47.1 簿記

簿記を主題とした資格試験は、国内において一般的なもので 4 種あります。

- 日商簿記（日本商工会議所 簿記検定試験）
- 全経簿記（全国経理教育協会 簿記能力検定）
- 全商簿記（全国商業高等学校協会 簿記実務検定試験）
- 日ビ簿記（日本ビジネス技能検定協会 簿記能力検定試験）

今回は、この 4 種の中で一番メジャーであるとされている日商簿記検定<sup>\*1</sup>について紐解いてみましょう。

日商簿記検定は現在、5 種類の試験が行われています。それぞれの試験内容について、上記の試験公式ホームページから引用し、それぞれの試験内容について確認しましょう。

---

<sup>\*1</sup> <https://www.kentei.ne.jp/bookkeeping>

### 47.1.1 簿記初級

「簿記の基本用語や複式簿記の仕組みを理解し、業務に利活用することができる。」(HP から引用)  
試験科目は商業簿記のみです。

### 47.1.2 原価計算初級

「原価計算の基本用語や原価と利益の関係を分析・理解し、業務に利活用することができる。」  
(HP から引用)

試験科目は工業簿記（原価計算）のみです。

### 47.1.3 日商簿記 3 級

「業種・職種にかかわらずビジネスパーソンが身に付けておくべき「必須の基本知識」として、多くの企業から評価される資格。基本的な商業簿記を修得し、小規模企業における企業活動や会計実務を踏まえ、経理関連書類の適切な処理を行うために求められるレベル。」(HP から引用)

試験科目は商業簿記のみです。

### 47.1.4 日商簿記 2 級

「経営管理に役立つ知識として、企業から最も求められる資格の一つ。高度な商業簿記・工業簿記（原価計算を含む）を修得し、財務諸表の数字から経営内容を把握できるなど、企業活動や会計実務を踏まえ適切な処理や分析を行うために求められるレベル。」(HP から引用)

試験科目は、商業簿記、工業簿記（原価計算を含む）の 2 科目です。

### 47.1.5 日商簿記 1 級

「極めて高度な商業簿記・会計学・工業簿記・原価計算を修得し、会計基準や会社法、財務諸表等規則などの企業会計に関する法規を踏まえて、経営管理や経営分析を行うために求められるレベル。合格すると税理士試験の受験資格が得られる。公認会計士、税理士などの国家資格への登竜門。」(HP から引用)

試験科目は、商業簿記、会計学、工業簿記、原価計算の 4 科目です。

### 47.1.6 日商簿記の各級の内容をかいづまんでいうと

各級の内容をかいづまんで、端的に説明すると

初級試験はエントリー向け（初学者向け）の内容です。

日商簿記 3 級では、個人商店で取り扱うような規模の簿記（拠点は 1 つのみ、）が大賞になります。

日商簿記2級でやっと会社らしい簿記（顕著な例を2つ挙げると、1つは複数拠点を踏まえた本支店会計などを取り扱うようになり、またもう1つは工業製品を作る際の簿記である工業簿記）を取り扱うようになります。

また、日商簿記1級では簿記の論点を横断的に取り扱うため、主に上場企業の会計担当者が取得するようなイメージとなります。

このように、日商簿記では試験の級が上がっていく毎に範囲が広範になり、深さが深くなっていますがわかります。

他の試験制度ではどうでしょうか？

## 47.2 リテールマーケティング（販売士）

次は日本商工会議所が同様に主催するリテールマーケティング（販売士）<sup>\*2</sup>について、同様に見てみましょう。

リテールマーケティング（販売士）は現在、3種類の試験が行われています。それぞれの試験内容について、上記の試験公式ホームページから引用し、それぞれの試験内容について確認しましょう。

なお、試験についてはすべての級で

- 小売業の類型
- マーチャンダイジング
- ストアオペレーション
- マーケティング
- 販売・経営管理

の5科目です。

### 47.2.1 リテールマーケティング（販売士）3級

「マーケティングの基本的な考え方や流通・小売業で必要な基礎知識・技能を理解している。接客や売場づくりなど、販売担当として必要な知識・技術を身につけた人材を目指す。流通・小売業に限らず、BtoCの観点から社員教育に取り入れている卸売業や製造業もある。」(HPから引用)

### 47.2.2 リテールマーケティング（販売士）2級

「マーケティング、マーチャンダイジングをはじめとする流通・小売業における高度な専門知識を身につけている。販売促進の企画・実行をリードし、店舗・売場を包括的にマネジメントする人材を目指す。幹部・管理職への昇進条件として活用しているところもある。」(HPから引用)

---

<sup>\*2</sup> <https://www.kentei.ne.jp/retailsales>

#### **47.2.3 リテールマーケティング（販売士）1級**

「経営に関する極めて高度な知識を身につけ、商品計画からマーケティング、経営計画の立案や財務予測等の経営管理について適切な判断ができる。マーケティングの責任者やコンサルタントとして戦略的に企業経営に関わる人材を目指す。」(HP から引用)

#### **47.2.4 リテールマーケティング（販売士）の各級の内容をかいつまんでいうと**

3級は店舗の販売担当、2級は販売組織の企画・実行や店舗・売り場のマネージャー、1級は企業経営ができる人材像となっている事がわかります。

試験内容の概要からはわかりにくいくらいですが、リテールマーケティング（販売士）検定でも、試験の級が上がっていく毎に範囲が広範になり、深さが深くなっています。

### **47.3 技術的な国家資格についても確認してみよう**

折角なので、技術的な国家資格についても同様となっているか確認してみましょう。

なお、情報処理技術者試験については別章でより具体的に説明する為、この章では電気通信の国家資格である電気通信の工事担任者、および電気通信主任技術者試験について掘り下げてみます。

電気通信の工事担任者、電気通信主任技術者のそれぞれの資格は総務省が行っている、電気通信事業者やそのネットワーク網における工事・通信設備の管理のために必要となる資格体系です。

#### **47.3.1 電気通信の工事担任者**

一般的の通信網（利用者による端末設備や、自営電気通信設備）の接続において、その回線工事を行うための資格です。

なお、なぜこういった資格試験があるか（必要とされているか）は、日本データ通信協会 電気通信国家試験センターの HP \*3 に詳しいのでそこから引用します。

利用者による端末設備又は自営電気通信設備の接続は、電気通信に関する知識を要し、その良否は電気通信回線設備を通じて他に与える影響が大きいことから、電気通信回線設備の損傷及び他の利用者への迷惑を事前に防止（接続の技術基準を確保）するとともに、人体の保護を確実ならしめるため、総務大臣がその資格を認定した工事担任者にこれに係る工事を行わせ、又は実地に監督させることを義務づけています。

この資格試験は5種類に分かれています。

- 第二級アナログ通信
- 第一級アナログ通信

\*3 <https://www.dekyo.or.jp/shiken/charge/about>

- 第二級デジタル通信
- 第一級デジタル通信
- 総合通信

大まかに説明すると、第二級はどちらも基本的な通信回線の接続工事（工事できる規模に制限がある）、第一級はその取扱範囲のすべての工事を行えます。

また、アナログの場合はアナログ伝送路（主に電話線）を対象とし、デジタルの場合はデジタル伝送路（主に LAN ケーブルや光ファイバー）を対象とします。総合通信は第一級アナログ通信と第一級デジタル通信を兼ねたものです。

試験範囲もそれらの工事範囲に対応した、端末設備の接続のための技術及び理論と、電気通信技術の基礎<sup>\*4</sup>、端末設備の接続に関する法規が出題されます（当然二級の方がやさしい<sup>\*5</sup>）。

### 47.3.2 電気通信主任技術者

先ほど紹介した工事担当者に加え、電気通信事業者がその設備を維持するための監督者としての資格も存在し、これが電気通信主任技術者となります。

こちらも日本データ通信協会 電気通信国家試験センターの HP <sup>\*6</sup> に詳しいのでそこから引用します。

電気通信事業者は、その事業用電気通信設備を、総務省令で定める技術基準に適合するよう、自主的に維持するために、電気通信主任技術者を選任し、電気通信設備の工事、維持及び運用の監督にあたらなければなりません。

電気通信主任技術者の選任は、原則として、事業用電気通信設備を直接に管理する事業場ごととなります。

但し、多数の事業場が地理的にも組織的にも近接している場合は、電気通信主任技術者が一定の範囲内の他の事業場の設備もあわせて監督できることになっております。

電気通信主任技術者資格者証の種類は、ネットワークを構成する設備に着目して区分されております。

試験は2種類あります。

- 伝送交換主任技術者
- 線路主任技術者

試験範囲としてはそれぞれ伝送交換設備と線路設備になり、その他共通となる試験として 電気通信システム、法規（電気通信事業法、有線電気通信法を中心とした行政法規）が出題 <sup>\*7</sup> され

<sup>\*4</sup> 基礎はマジで基礎なので、オームの法則とかキルヒ霍フの法則、論理回路なんかが出ます。

<sup>\*5</sup> 二級は CBT も開始されました。受けやすいのも良い。

<sup>\*6</sup> <https://www.dekyo.or.jp/shiken/chief/about>

<sup>\*7</sup> なお、筆者としてもこの試験が多岐選択式試験としては恐らくいちばんハードな部類なのではないかという気がしているのですが、20 択試験とかあります。どういうことだ。

ます。

また電気通信主任技術者試験は、電気通信の工事担任者のデジタル一級ないし総合通信資格を保持していれば、前述の「電気通信システム」の試験が永久に免除<sup>\*8</sup>されます。

## 47.4 おわりに

資格試験がグレード毎に「簡単な論点（大事な基本論点）から応用論点に向けて、順に学ぶ事ができる」事の実例を3つの試験を通してお伝えしてみましたがどうだったでしょうか？

段階的にステップアップする事で、着実に学んでいけるのは良いことだと筆者も考えています。

ぜひ、資格試験をあなたの学びに組み込んでみてください！

---

<sup>\*8</sup> まあ、前述の資格を取得できていれば恐らく受けても通るのだが……

## 第 48 章

# 情報処理技術者試験を俯瞰する

@ditflame

IT に関する国家資格・試験といえば、情報処理技術者試験が筆頭に上がるのではないかと思いますが、この試験も「段階的にステップアップ」して学んでいくような試験体系になっています。また、この試験体系では各試験において ITSS (IT スキル標準) <sup>\*1</sup> というレベルが示されており、次のようなスキルレベルが想定されています。

レベル 7：プロフェッショナルとしてスキルの専門分野が確立し、社内外において、テクノロジやメソドロジ、ビジネスを創造し、リードするレベル。市場全体から見ても、先進的なサービスの開拓や市場化をリードした経験と実績を有しており、世界で通用するプレーヤとして認められます。

レベル 6：プロフェッショナルとしてスキルの専門分野が確立し、社内外において、テクノロジやメソドロジ、ビジネスを創造し、リードするレベル。社内だけでなく市場においても、プロフェッショナルとして経験と実績を有しており、国内のハイエンドプレーヤとして認められます。

レベル 5：プロフェッショナルとしてスキルの専門分野が確立し、社内においてテクノロジやメソドロジ、ビジネスを創造し、リードするレベル。社内において、プロフェッショナルとして自他共に経験と実績を有しており、企業内のハイエンドプレーヤとして認められます。

レベル 4：プロフェッショナルとしてスキルの専門分野が確立し、自らのスキルを活用することによって、独力で業務上の課題の発見と解決をリードするレベル。社内において、プロフェッショナルとして求められる経験の知識化とその応用（後進育成）に貢献しており、ハイレベルのプレーヤとして認められます。スキル開発においても自らのスキルの研鑽を継続することが求められます。

レベル 3：要求された作業を全て独力で遂行します。スキルの専門分野確立を目指し、プロフェッショナルとなるために必要な応用的知識・技能を有します。スキル開発においても自らのスキルの研鑽を継続することが求められます。

レベル 2：上位者の指導の下に、要求された作業を担当します。プロフェッショナルとなるために必要な基本的知識・技能を有する。スキル開発においては、自らのキャリアパス実現に向けて積極的なスキルの研鑽が求められます。

レベル 1：情報技術に携わる者に最低限必要な基礎知識を有します。スキル開発においては、自らのキャリアパス実現に向けて積極的なスキルの研鑽が求められます。

この記事では ITSS スキルレベル 3 までの受験プランとその理由、および情報処理技術者試験が段階的に学びを深くするにつれ、インプット重視からアウトプット重視の試験へと内容が変わっていく という点をお伝えします。

<sup>\*1</sup> <https://www.ipa.go.jp/jinzai/skill-standard/plus-it-ui/itss2.html> なお、情報処理技術者試験で対象像とされているスキルレベルは 4 までです。

## 48.1 ITSS の各レベルと情報処理技術者試験の対応

ITSS の各レベルと、情報処理技術者試験の対応は次のようになっています。

表 48.1: ITSS レベルと情報処理技術者試験の対応

| ITSS レベル | 試験名 (略称)                                               |
|----------|--------------------------------------------------------|
| レベル 1    | IT パスポート試験 (IP)                                        |
| レベル 2    | 基本情報技術者試験 (FE)、情報セキュリティマネジメント試験 (SG)                   |
| レベル 3    | 応用情報技術者試験 (AP)                                         |
| レベル 4    | 高度情報技術者試験 (ST,SA,PM,NW,DB,ES,SM,AU)、情報処理安全確保支援士試験 (SC) |

試験の全体像も次に示します (IPA の HP からの引用です)。



図 48.1: 全体像

## 48.2 ITSS レベル 1 の試験

ITSS レベル 1 に該当する試験は 1 つのみで、IT パスポート試験 (IP) です。

### 48.2.1 IT パスポート試験 (IP) について

この試験は CBT で行われており、いつでも受ける事ができます。また試験は 1 つだけで、試験時間は 120 分、出題形式は多肢選択式 (四肢択一)、出題数は 100 問 (小問形式) です。

多岐選択を中心とした試験であるため、おもにインプットがちゃんとできていれば合格できます。

## 48.3 ITSS レベル 2 の試験

ITSS レベル 2 に該当する試験は 2 つあり、基本情報技術者試験 (FE)、情報セキュリティマネジメント試験 (SG) です。

### 48.3.1 基本情報技術者試験 (FE) について

この試験は CBT で行われており、いつでも受ける事ができます。

また試験は科目 A と科目 B に分かれています。

科目 A は、試験時間 90 分、出題形式は多肢選択式（四肢択一）、出題数は 60 問（小問形式）です。

科目 B は、試験時間 100 分、出題形式は多肢選択式、出題数は 20 問（分野別の出題数は、アルゴリズムとプログラミング分野 16 問、情報セキュリティ分野 4 問）です。

科目 A については多岐選択を中心とした試験であるため、おもにインプットがちゃんとできていれば合格ラインに到達できますが、科目 B については擬似言語でのプログラム（アルゴリズム）や、セキュリティに関する設問を正しく読み解いて解答する必要があり、少しアウトプット能力が必要となります。

### 48.3.2 情報セキュリティマネジメント試験 (SG) について

この試験は CBT で行われており、いつでも受ける事ができます。

また試験は科目 A と科目 B に分かれています。

科目 A・科目 B は併せて試験時間 120 分です。

出題形式は、科目 A が多肢選択式（四肢択一）、科目 B が多肢選択式です。また、出題数は、併せて 60 問（科目 A の出題数は 48 問、科目 B の出題数は 12 問）です。

科目 A については多岐選択を中心とした試験であるため、おもにインプットがちゃんとできていれば合格ラインに到達できますが、科目 B についてはセキュリティに関する設問を正しく読み解いて解答する必要があり、少しアウトプット能力が必要となります（基本情報よりは随分マシなはず）。

また、情報セキュリティマネジメント試験は分野的に全く問題が出ないケースがあります。具体的には次の図を併せて確認してみてください（図は試験のシラバスから引用）。

注記1 ○は出題範囲であることを、◎は出題範囲のうちの重点分野であることを表す

注記2 2, 3, 4は技術レベルを表し、4が最も高度で、上位は下位を包含する。  
 注記3 “中公新文庫”に収載。ただし、算術問題には技術レベルを示す

<sup>注 1)</sup> “中分類 11：セキュリティ”の知識項目には技術面・管理面の両方が含まれるが、高度試験の各試験区分では、各上位像に基づいて開連地の強い知識項目を一括りにして出題する。

験区分では、各人材像にとって関連性の強い知識項目をレベル4として出題する。

図 48.2: 試験区分別出題分野一覧表

#### 48.4 ITSS レベル 3 の試験

ITSS レベル 3 に該当する試験は 1 つのみで、応用情報技術者試験 (AP) です。

#### 48.4.1 応用情報技術者試験(AP)について

応用情報からは、CBTは今のところ(2024年12月現在)ないので、春期と秋期の試験<sup>\*2</sup>で受験する事になります。

試験は午前試験と午後試験に別れます。<sup>\*3</sup>

午前の試験時間は 9:30～12:00(150 分)、出題形式は多肢選択式(四肢択一)、出題数は 80 間です。

午後の試験時間は 13:00～15:30(150 分)、出題形式は記述式、11 問出題されるのに対し、5 問選択して解答<sup>\*4</sup> します。

午前試験については多岐選択を中心とした試験であるため、おもにインプットがちゃんとできていれば合格ラインに到達できます。ですが、午後試験については記述が必須となるため、アウトプット力も併せて必要となります(設問によりますが解答量の多いケースだと40文字ぐらいまで)

\*2 例年だと4月は第3週の日曜、10月は第2週の日曜です。

\*<sup>3</sup> 結構長丁場。慣れてないと試験を走り切るだけでも正直しんどい……

\*4 情報セキュリティの設問は必須、他の分野で 10 間出題され 4 間選択します。

書く事がある)。

### ■コラム：Iパス、基本情報と情報セキュリティマネジメント、応用情報ってどの順でどう受けければ良いの？

筆者もよく聞かれる事があるのでですが、丁寧に試験を拾っていくのであれば

- IT パスポート試験 (IP)
- 情報セキュリティマネジメント試験 (SG)
- 基本情報技術者試験 (FE)
- 応用情報技術者試験 (AP)

の順で拾う事をおすすめします。これは次のような理由の為です。

1) IT パスポート試験で CBT 試験に慣れつつ、情報処理技術者試験の用語に慣れる

2) まず重要な点として、基本情報や応用情報（高度もだが）はセキュリティ設問の配点が強めの試験となっており、セキュリティ設問得意にしておくに越したことがない。また基本情報の科目 B のセキュリティ設問と同様の問題セットで情報セキュリティマネジメント試験の問題セットを作っているのではないか？と推察される（同じ ITSS レベルであるところがミソ）

3) IT パスポート試験 (IP) と情報セキュリティマネジメント試験 (SG) が取れたら、基本情報技術者試験 (FE) はいつでも受けられるのでこれを受けつつ、応用情報技術者試験 (AP) はアルゴリズム系の設問が (FE に比べ) 弱いので、取りやすい方から取る。するとおそらく後に基本情報技術者試験 (FE) を受験する事になる。

（※なお FE と AP は 2023 年の試験改訂からかなり味付けの違う試験になってしまったので、就活や転職目的で取るならば、両方取るほうが良い）

なお、この 4 つを取れば、高度情報の午前 1 は余裕で取れる様になっているはずです（恐らくですが、これらも問題セットが同じなので……）。

## 48.5 ITSS レベル 4 の試験

ITSS レベル 4 に該当する試験は 9 つあり、一般的に「高度情報技術者試験」と総称されていますが、大まかな概要のみ紹介します。

基本的に、高度情報の試験は 4 つの試験に別れており、「午前 1」「午前 2」「午後 1」「午後 2」という 4 つの試験を 1 日で受験する事になります。

※なお情報処理安全確保支援士試験のみ、午後試験が 1 本化されています。

各試験で、午前1、午前2では多岐選択試験、午後1では短文記述 という形で試験方式が共通となっており午後2については、次の6つの試験は論文試験、

- ITストラテジスト試験(ST)
- システムアーキテクト試験(SA)
- プロジェクトマネージャ試験(PM)
- エンベデッドシステムスペシャリスト試験(ES)
- ITサービスマネージャ試験(SM)
- システム監査技術者試験(AU)

次の2つの試験は1つの大問を2時間かけて解く大掛かりな記述式の試験となります。

- ネットワークスペシャリスト試験(NW)
- データベーススペシャリスト試験(DB)

※情報処理安全確保支援士試験(SC)もこちら側に似たタイプの試験になります。

このため、午前の試験でインプット力を試されると同時に、午後の試験でアウトプット力を試される試験制度となっており、アウトプット力が無いと受からない試験になっています。

## 48.6 おわりに

情報処理技術者試験では、試験のITSSレベルが上がる毎に、合格するためには求められるアウトプットの比重が高まるようになっていることがわかりました。

これは実際に仕事を進めて習熟が高まるにつれ、仕事で求められるものがの比重がアウトプットに寄っていくのと同様ではないかと筆者は考えているのですが、どうだったでしょうか？

## 第 49 章

---

# 聴いて学ぼう

@ditflame

社会人になってから、学生の頃と大きく「学び」の面で変わる点に、「割ける時間」が激変するというものがあります。

この章では、「割ける時間」が激変する事への対抗策の一つとして、「聴いて学ぶ」点、またその具体的な方法や、自分自身で「聴いて学ぶ」音声をどう作るか といった点を深堀りします。

### 49.1 聽いて学ぶ際のメリット・デメリット

座学と違い、聴いて学ぶ際には次のようなメリット・デメリットがあります。

#### 49.1.1 メリット

- 耳が空いていればいつでも学べる

端的に書くとこの 1 点です。耳があいているタイミングならいつでも OK。

それは例えれば移動中かもしれないし、家事をしている時や、作業中に BGM 代わりに といったケースもアリかもしれません。

#### 49.1.2 デメリット

- 座学に比べて学習の効率が別段いいわけではない
- 聴いて学ぶための手段が要る

当然、机に座って集中して学ぶほうが効率は良さそうです。また、「聴いて学ぶ」特性上、それに対応した手段を用意する必要があります。

### 49.2 聴いて学ぶための手段

聴いて学ぶためには、なんらかの形で音声データになった学習教材が必要です。

具体的には次のような物が考えられます。

- podcast のような音声で提供されている「学び」
- 自分で録音する

- テキストから音声に変換する

### 49.2.1 音声で提供されている「学び」

これは、さまざまな方が配信されている podcast や、たとえば資格のスクールなんかで提供されることがある音声講義などが該当するでしょう。

すでに提供されている音声であれば、手軽に始められるのが最大の利点です。

### 49.2.2 自自分で録音する

自分で録音する場合は、ヘッドセットを使って録音するのがおすすめです。



図 49.1: ChatGPT 画伯に描いてもらった

なお、

- 少しゆっくり目に話す
- マイクと口の距離を一定にする
- 声の音量を一定にする

と聞きやすくなります。

### 49.2.3 テキストから音声に変換する

テキストデータがあれば、音声に変換する事で、音声学習用の音声データを大量に作り出す事ができます。

これには、棒読みちゃん<sup>\*1</sup> や suno<sup>\*2</sup> を使うのが楽でしょう。なお、suno を使うと曲になります！

具体的にこのアプローチが良く効くのは、なんらかの具体的なルール集(例:ISOの条文とか……)や、明文化されたベストプラクティスがあってそれを覚える必要がある場合で、実際に筆者が試験を受けてきた中で、良く効いたなど感じたのは情報セキュリティスペシャリスト試験<sup>\*3</sup>や、kintone認定カイゼンマネジメントエキスパート試験<sup>\*4</sup>です。

### ■コラム：自動変換で読み間違いが起きる件

日本語を読み上げツールや、AI作曲ツールで音にする時、そもそも内部的な処理としては漢字をよみがなにして、「どう読んでいるか」を生成して音にするのですが、漢字って様々な読み方があるので、そのまま世にあるテキストデータの文章を音楽にすると、高頻度で読み間違えた音声や曲が生成されるケースがままあります。

こういった読み間違いが発生する際は、そういった漢字をひらいて(ひらがなにして)、読み間違う箇所をあらかじめクリアして歌詞として食わせるか、棒読みちゃんなどの専門的な音声読み上げソフトの場合は、漢字に対するよみがなをあらかじめ辞書として指定できるので、そういった機能で誤読を回避すると、より学びやすくなります。

## 49.3 おわりに

正確さよりも概念の理解など、大枠での基礎的な能力を求められるタイプの試験では、音声による学習はハマるとめっぽう強い実感があります。

ぜひ、自らの「学び」に取り入れてみてください！

\*1 <https://chi.usamimi.info/Program/Application/BouyomiChan/>

\*2 <https://suno.com/>

\*3 現在の情報処理安全確保支援士試験( <https://www.ipa.go.jp/shiken/kubun/sc.html> )の前身。試験内容はほぼ変わらない気がする

\*4 <https://cybozu.co.jp/kintone-certification/level/kaizenmanagement-expert/>

# 資格試験を受け続けて気付いた事

@ditflame

筆者は 2010 年頃から年 2~3 本のペースで資格試験を受けているのですが、その中で気づいた事を書きます。

## 50.1 資格試験を申し込んでから受験対策を考えても結構間に合う

結構間に合います。

また、余暇時間を日頃から自己研鑽に割ける人は世の中にはそんなにいない気がします。

そんなあなたに『〆切駆動』。 申し込んでしまえば〆切ができます。

なお、資格のスクールの資料などを取り寄せてみるとよくわかるのですが、資格によっては 1 コースは 3 ヶ月ぐらいだったりします。<sup>\*1</sup>

## 50.2 いっぱい〆切ができると結構学べる

射的などでもそうですが、的がないと弾は当たらないもので、〆切やゴールラインがないと どうしても漫然としてしまうものです。

資格試験の場合は申し込んだらゴールラインが決まり、〆切も決まってしまうので、ゴールに向かって着実に学びを進める事ができます。

これを反復的 (iterative) に繰り返すと…… インプットがはかどりそうな気がしませんか？

なお、数を受ける事で試験に慣れる効果も見逃せません。試験回答時のミスが目に見えて減ります。

## 50.3 色々な資格試験を受けると自分の適性が見えやすいかも

様々なスキルスタックが世には溢れています (これは IT に限らず)。 ですが、転職して試すのはハイリスク！

その点、試験であれば極めてローリスク。せいぜいかかって受験費用と書籍代ぐらいなもの

<sup>\*1</sup> これは「資格の本」みたいなものでも良い。そういう本には大抵、学習期間が掲載されている筈だが、3~4 ヶ月を超える資格は早々は無い事がわかります。

です。

色々な試験を受ける中で、取れ高が良い分野は向いているし、取れ高が悪い分野はあまり向いていない 可能性が高いのではないかと筆者は考えています。どうでしょうか？ <sup>\*2</sup>

### 50.4 色々な資格試験を受けると、1+1 は 2 ではなくなる

世には様々な資格試験がありますが、いろいろな試験を受けて合格した場合、1+1 は 2 にならないことがあります。

1 とか 2 って、何のことでしょう？

そう。この 1 とか 2 とか言っているのは「学習の総量」です。

というのも、世の試験は基礎的な分野は共通している事が多く、1 つ目を受験して合格するには 1 の学習量 (100 % の学習量) が必要となりますが、次の 1 を拾う時には、範囲が重複しているものについては 0.8(80 % の学習量) だったり 0.5(50 % の学習量) になる事がままあります。

重複受験によるシナジーが高い試験の見つけ方としては、

- 試験母体が違うが同じ事を問う試験 (例:日商簿記と全経簿記 など)
- 試験母体が違うが近接した内容を取り扱う試験 (例:電気通信主任技術者 (伝送交換) と ネットワークスペシャリスト試験 など)
- 試験母体が同じで近接した内容を取り扱う試験 (例:情報処理安全確保支援士試験を受験後のデータベーススペシャリスト試験 など)

といった形で探す事ができます (シナジーが強すぎると、試験によっては特定資格の所持で科目合格になるケースもあります)。

### 50.5 資格を色々拾って嬉しかった事

筆者の場合はこんな感じでした。

#### 50.5.1 職場に関わらず嬉しいポイント

- 就職、転職が本当に困らない。

ちなみに就職の時はバブル後の就職氷河期のワースト就職率の時でしたが 5 月で就活は終わってましたし、資格を握って転職した場合は普通なら給与・待遇が上がります。

- 余裕ができる。会社をやめてもどうにかなるという安心感がすごく楽。

---

<sup>\*2</sup> なお、実際に筆者はこの理由で大学の学部学科の選択を電子回路 (高周波回路) から、IT に振り替えました。

## **50.6 最後に、イテレーティブに受験し続ける際の最大のコツをお教えします**

35歳超えてから大阪で転職活動した際、ESでお断りされたケースは業務とスキルのミスマッチで合わないと判断されたケースのみでした。兎にも角にも書類が通る。

- いろいろと横断的に知識を拾えたので、ITを俯瞰的に見れる。その結果、ほかの人が持っていない知識で課題を解決したりできる。

高い建物を建てる過程で周辺知識もどうしてもつくので……

### **50.5.2 Sler の時嬉しかった**

- 昇給査定に（多少）加味される。一時金ももらえる。

資格をバリバリ仕込めるのであれば、会社の規模が小さいほど資格に伴う手当は厚い傾向があります。単純に資格を持ってる人のほうが、案件に突っ込みやすいからだと思うのですけどね。

なお、就活などで会社の規模感や大小が判断しにくい場合、資格手当の小さい会社のほうが規模は大きいという可能性はあります。ただしエンジニアを大事にしていない可能性もあります。悩ましい…

- 技術的に難易度の高い、単純作業ではない仕事を振ってもらえる。

これは、実際にあって、その時やった仕事がパッケージソフトの公式開発事例になったりしました。

- 社内報に載ってドヤれる。

(・'-' ) どやあ

### **50.5.3 社内SEで嬉しかった**

- 聞かれた大抵の事は客観的に技術の視点からジャッジしつつ、伴った説明もできるので現場の人に喜ばれる。
- メンターがない状態で、新しい事を始めようとする場合はどうしても知識のフックが多いほうが有利。

## **50.6 最後に、イテレーティブに受験し続ける際の最大のコツをお教えします**

試験前の金曜日は年休を取って休みましょう！  
(結局のところは体力勝負になったりするので……)

## 50.7 おわりに

まずは申し込んで、〆切駆動で走ってみると意外に走れます！（資格もコンスタントに拾えるかも？）

ぜひチャレンジしてみてください！

# 良いアウトプットのためのインプットと ノート整理

酒井文也 (Fumiya Sakai) Twitter & github: @fumiayasac

## 51.1 はじめに

技術的な知識を効果的に学ぶためには、単に情報を受け取るだけでは十分ではありません。真の理解は、得た情報を自分なりに噛み砕き、視覚的に整理することから生まれます。私のエンジニアリング学習において最も大切にしているのは、調査した技術トピックを手書きでノートにまとめるというアプローチです。

このノート作成は、ただの書き写しではありません。サンプルコードの実装における重要なポイントを、自分の視点で抽出し、言葉にすることで、技術的概念の本質的な理解に近づきます。手で書くという行為は、デジタル的な記録とは異なる、より深い思考のプロセスを引き出してくれます。指先で情報を紡ぐことで、抽象的な概念を具体的な知識に変換できるのです。

例えば、

- コードの背後にある設計思想
- アルゴリズムの論理的な構造
- 実装上で工夫しているポイント

等、書籍やサンプル実装コード、動画教材等から得られた情報を自分なりの言葉や図解でノートに記録することで、技術の表面的な理解から、より深い洞察へと成長できると思います。

情報を受け取るだけでなく、自分なりの学びの地図を描くこと。それが、私の技術学習における最も大切な方法だと考えています。

## 51.2 進める際の個人的ポイント

エンジニアとして、技術的な知識を学ぶ際に大切にしているのは、ただ情報を集めるだけではなく、その本質を理解し、将来活用できる形で整理することです。

私の学習スタイルは、様々なリソースから得られる情報の断片を、パズルのピースのようにつなぎ合わせていくアプローチです。書籍、サンプルコード、オンライン講座、動画教材など、多様な情報源から学びを得ながら、それぞれの知識がどのようにつながっているのかを、できる限り丁寧に紐解いていきます。

新しい専門用語や技術的概念に出会うたびに、徹底的に調べて理解を深めます。関連する事例や周辺知識も積極的に探求し、自分の技術的な知識の幅を広げていきます。複雑な概念は、できるだけ簡潔な図解や視覚的な表現に置き換えることで、より直感的に理解しやすくなります。

手書きのノートは、私にとってはこの学習プロセスの中心的なツールです。単なるメモ帳ではなく、後で見返したときに、すぐに記憶が呼び起こせるような、自分だけの知識マップを作り上げる様なイメージでしょうか。ここで大切なのは、このノートを通じて得た知識が、将来の技術的な課題や問題解決に、柔軟かつ実践的に活用できる形で整理されていることです。

情報を取り入れ、整理し、そして最終的に自分のものとして活用できるようにする。このサイクルが、私にとって技術者としての成長を支える原動力となります。

私にとってノート作成は、技術的な知識を生きた知恵へと変換するための方法かもしれません。

### 51.3 実践したメリット

技術を学ぶ上で大切なのは、ただ情報を集めるだけではなく、その本質を理解することです。私は、手書きのノートを通じて、技術的な知識を深く、そして自分なりに理解する方法を見つけてきました。

サンプルコードや技術資料から重要なポイントを丁寧に抽出し、整理していくこのアプローチは、表面的な学習を超えて、技術の本質に迫ろうとするものです。理論的な理解と、実際の実装イメージを同時に育むことで、単なる暗記とは異なる、深い学びを実現しています。

一見すると時間のかかる学習方法に思えるかもしれません。しかし、じっくりと知識を自分のものにしていくこのプロセスには、大きな利点があります。技術的な概念を自分なりに咀嚼し、視覚化することで、似たような課題に直面したときにも、落ち着いて対応できるようになります。

手書きでノートをまとめる行為は、ただの記録以上の意味を持つと考えています。得られた知識は、仕事や個人的な開発において、柔軟に活用できる「引き出し」となります。特に、実装イメージが似ている技術的な課題に取り組む際には、以前の学習が大きな助けとなり、問題解決のスピードを上げることができます。

デジタル技術が急速に進化する現代において、情報の洪水の中で本当の理解を得ることは簡単ではありません。このような学習方法は、ただ知識を詰め込むのではなく、技術の本質を理解し、実践的な応用力を育む、とても効果的なアプローチだと考えています。

エンジニアとして成長し続けるためには、自分に合った学習方法を常に模索し、改善していくことが大切です。手書きのノートを通じた学習は、そんな個人的な学びの探求の一つの形と言えるでしょう。

入力時に改行をすると、入力エリートが伸縮する TextView を見てみる

例として OSS :

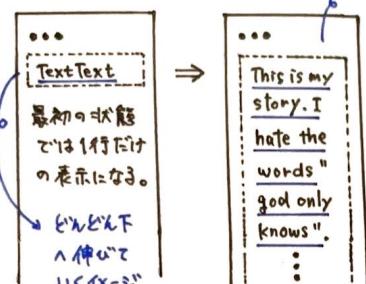
### Resizing TextView の動きを見てみる

\* Androidの様なテキスト入力フォーム  
を実現する形をつくる事ができる。

↓  
struct PreviewWrapper : View {

④ State var sampleText = "" 入力する内容と View で \$sampleText で Binding する

\* 自動でサイズを調整する形



この様に下へ入力エリートがピンドンと伸びていく  
④ Binding で ④ State と  
結合する形にする  
ResizingTextView(text: \$sampleText, ① isScrollable: true, ② canHaveNewLine(characters: true) ③  
次の行を、文字数がそのラインを超過したら表示する

その他にも、最大行数を制限することや、Font サイズや色も設定  
行数は3行まで、多くの文字を入力すると、Scroll 可能になります

TextView のベースにある状態

(UIKit で作っている)

class CustomTextView : UITextView { ... }  
↳ サイズ調整に関する property を定義する

④ MainActor struct

TextView : UIViewRepresentable  
で SwiftUI で使える様にする

ポイントは、Binding<String> を渡していける  
(④ Binding private var text: String)

\* その他にも、initializer で 設定できるのは  
適意を満たすための 設定を渡していく。

\* func makeUIView(context: Context) -> CustomTextView

context.coordinator → Coordinator の中に  
UIkit で protocol

→ CustomTextView をインスタンス化して、view.delegate = context.coordinator ..... ① での実装を定義  
updateUIView(view, context: context) ..... ②

\* func updateUIView(\_ view: CustomTextView, context: Context)

→ この部分で 実行していけるのは、様々な 設定値の 反映を行っていき

\* Coordinator 関連の処理

func makeCoordinator() -> Coordinator {  
 Coordinator(self)  
}

実際のコードを見ながら、ミニコール  
な形で実現できますか？

final class Coordinator: NSObject, UITextViewDelegate {

\* updateUIView 处理内で、

• var parent: UITextView (UIViewRepresentable の要素)

View 要素に 肉する Inset 値

• var selectedRange: NSRange? (選択部分の情報)

を更新するとして、全体 Layout

func textViewDidChange(\_ textView: UITextView) {

を更新している。

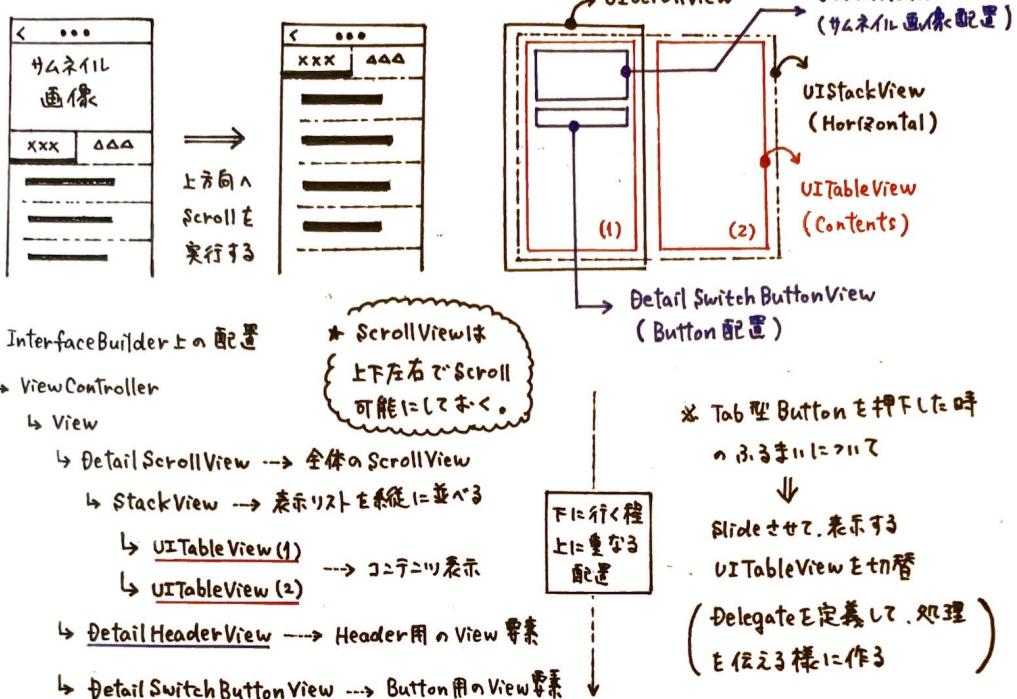
(1) テキスト反映 (2) canHaveNewLineCharacters の反映

(3) 選択範囲 (selectedRange) の反映

}

iOSではUITableView様な表現をどう作るか?

(1番 Simpleに作る with Storyboard)



### AutoLayoutの制約には注意する

要素同士の制約のつけ方 + ScrollViewに合わせた変化を合わせる

#### ① DetailHeaderViewとParallax表現にする場合

→ ScrollViewの offset 値が変化した時に発火

UIScrollViewDelegateの `scrollViewDidScroll(_ scrollView: UIScrollView)` を活用

↳ Y軸方向の offset 値を調整して AutoLayout 制約へ加算する

#### ② Headerとして表示している要素が隠れても、タブ型 Button用 Viewは表示させたい!

↳ DetailHeaderViewの Top に付与した制約を調整する

→ IBOutletでつりあわせて、offset 値の変化に合わせて調整

#### ③ コンテンツ用の UITableViewに対しても、Y軸方向の offset 値を調整する

- case1: 画像の Parallax 効果付きの View が見えている

→ 配置した各 UITableView の Y 軸方向の offset 値も一緒に更新

- case2: 画像の Parallax 効果付きの View が隠れている

→ 現在重なってない UITableView の top 位置を調整する

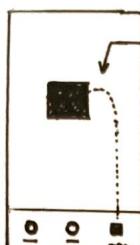
## < Netflix の「あの動き」を clone(クローン) する方法 >

特定の場所に吸い込まれる様な Animation

例、Netflix の視聴対象を選ぶ画面



誰が視聴する  
のかをボタンで  
押す  
細かな所が  
難しそう...



TabBar表示  
(すなはちXに)  
に切替わる時  
Animation件手

Animation  
動き作りX-シ  
飛らかせ放物線の  
様な軌跡を描く  
Tab要素の中へ吸  
込まれていく様な形

- ① Path に沿って Animation  
をする形で実現する
- or
- ② GeometryReader の値  
を使って Customize する

### 実際の Sample を元にuttとく

\* 余談 → Geometry Readerだけではキツイ点

1. そのままの Animation では、動きが直線的になる
2. View の階層によっては、うまく重なる形の実現が難しい

どちらも選択するかは、僕は matchedGeometryEffect ?

結構 View 構造次  
事かも...??

\* GeometryReader を利用した座標計算

#### Pros:

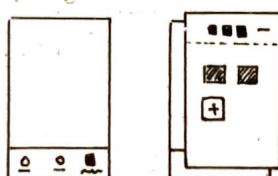
コードの分量が少なくて済む余地がある?

1. GeometryReader o GeometryProxy  
から得られた値を使うのがポイント
2. modifier を利用して、位置情報を  
適用する

$\left\{ \begin{array}{l} \cdot position \\ \cdot scaleEffect \\ \cdot opacity \end{array} \right.$   
 この値を調節  
するのがポイント

#### Cons:

飛らかせた Animation の様には  
なはけ、こうダメだ...



元々は TabBar  
用の画面か  
ベースにある

おおもとの  
Content View 内  
で Stack で重ねる

1. まずはおもとの  
画面があって →  
重ねている
2. 状態管理をする  
クラスの内容を →  
更新する
3. Animation に関する  
処理が実行され、  
TabBar が表示

\* この Sample 実装では、  
Application 全体の状態  
管理をする部分では、  
"③ Observation" で ViewModel  
の様な形で状態管理している。

↓

- 例。  
animateToCenter → 対象が中心まで移動したか?  
animateToMain → X への画面が表示されたか?  
progress → Animation 実行割合

★ Application 全体の  
状態管理クラス  
のインスタンスは  
③ Bindable  
var で

通用!

#### MainView()

\* 条件に応じた処理を  
諸々書き足していく

#### ProfileView()

animateToCenter:  
\$bindableData.animateToCenter,

animateToMain:

\$bindableData.animateToMain,

progress:

\$bindableData.progress

\* ③ Bindable var  
利用することで、  
変化に連動する

\* Animation 处理や、  
切替の状態管理  
を管理するラグが必要

## 51.4 まとめ

技術の世界は日々変化し続けています。その中で、眞の学びとは単に情報を集めることではなく、得た知識を自分のものとし、実践に活かせる知恵に変えていくプロセスそのものです。私が大切にしている手書きノートによる学習は、まさにこの技術者としての成長の本質を体現しているのかもしれません。

デジタル時代には、膨大な情報があふれていますが、本当の理解は情報の量ではなく、その質によって決まります。サンプルコード、技術書、オンライン教材から得られる断片的な知識を、自分の手と頭で丁寧につなぎ合わせていくこと。それが、表面的な知識の Collection から、技術の本質的な理解への道だと考えています。

手書きのノートは、単なるメモ帳ではありません。それは思考を可視化するキャンバスであり、抽象的な概念を具体的なイメージに変換する、最も個人的で創造的な学びの方法です。理論と実践をつなぐ、私だけの知識の地図とも言えるでしょう。

このアプローチの眞の価値は、知識を暗記することではなく、学んだことを実際に活かせるようになります。似たような技術的な課題に直面したとき、過去の学びがすぐに思い出せ、落ち着いて対応できる柔軟性が身につきます。蓄積された知識は、問題解決の力を大きく後押しする、目に見えない財産となります。

エンジニアとして成長するということは、常に自分に合った学び方を探し、改善し続けることです。手書きのノートによる学習は、単なる技術の記録方法ではなく、自分の知的好奇心を大切にする姿勢そのものなのかもしれません。

最終的に、私たちエンジニアに求められるのは、技術の表面をなぞるのではなく、その本質を理解し、創造的に活用する力です。情報を受け取り、自分なりに咀嚼し、血肉化する。このサイクルが、技術者としての成長を支える原動力となるのです。

# 認知のベンチマークをしよう

私たちは日々、さまざまな物事を認知し、判断しながら生きています。しかし、その認知は個々人によって異なることが多く、場合によっては大きくずれることもあります。例えば、同じ景色を見ていても「美しい」と感じる人もいれば、「平凡だ」と感じる人もいるでしょう。たまの 2000 円のランチを高いと思う人も、安いと思う人もいるでしょうし、たまの贅沢として価値を感じる人も、そういうところに価値を見出さない人もいるでしょう。このような認知の違いは、私たちが置かれた環境や経験、価値観に深く影響されています。

## 52.1 他者の認知と比較する

他人との認知の違いに気づくための有効な手段の一つが、自分の認知を開示することです。自分が何を見て、どう感じ、どのように解釈しているのかを他者に伝えることで、他人の認知との比較、すなわちベンチマークを取ることができます。このプロセスは、自分自身の考えを再確認し、他人との違いや共通点を理解するための貴重な機会となります。

しかし、認知はあくまでその人自身のものであり、他人と一致させる必要はありません。むしろ、多様な認知があること自体が社会の豊かさを生み出します。例えば、同じ問題に対して異なる視点を持つことで、新しいアイデアや解決策が生まれることがあります。そのため、自分の認知を他人のそれに無理に合わせようとする必要はありません。

### 52.1.1 他人の認知との差を認知する

とはいっても、自己と他人の認知が異なることを認識することは非常に重要です。他人が何を考えているのか、どのように感じているのかを知ることで、対人関係における摩擦や誤解を減らすことができます。たとえば、プロジェクトチームでのコミュニケーションにおいて、各メンバーが異なる優先事項を持っていることを理解することで、より効果的な調整が可能になります。また、他人の認知に触れることで、自分自身の認知に偏りや盲点があることに気づくこともあります。

ここで重要なのは、自己と他人の認知を比較する際の姿勢です。他人の認知を単に「正しい」または「間違っている」と評価するのではなく、「なぜそのように認知するのか」という背景や理由に目を向けることが大切です。たとえば、ある人がリスクを強く認識する一方で、別の人があそれを軽視する場合、その違いは性格や過去の経験、現在の状況に起因しているかもしれません。こうした背景を知ることで、単なる認知の違いが深い洞察や共感を生むことがあります。異なる認知をする人がいるということを認知しているだけでも十分です。

### 52.1.2 自分の認知を開示する

自分の認知を開示することで、他の人（隣にいる A さん）の認知とどれくらいのギャップがあるのか、といった価値でベンチマークを取ることができますね。

さらに、認知のベンチマークは個人だけでなく、組織や社会全体にも応用できます。企業が顧客や市場の認知を理解することで、より適切なサービスや製品を提供できるようになるのはその一例です。また、社会全体が異なる文化や価値観を尊重し合うことで、共存可能な環境が構築されます。

## 52.2 まとめ

認知の違いは、しばしば誤解や対立を生む原因にもなりますが、それを「違い」として尊重し、理解しようとする姿勢を持つことで、むしろ成長や発展の糧となります。良いものは取り入れることで、自分の発展にもなります。納得できない場合は、納得しないまでも、「他の人はそう考えるのだと記録する」という対処も可能です。

それらは、「比較する」ことで差があることを認知し、その差を活かすことができるようになります。

## 効率よく学ぶには資格を取る

何か学びたいことが見つかったとき、それを手っ取り早く学ぶ方法は何でしょう？

それは、関連する資格試験を受けることです。

たいてい資格試験があります。もちろん、医師などのように、医学部の正規課程を修めたうえで医師国家試験に受かる必要がある厳格な資格もありますが、直接資格試験を受験すれば取得できる資格もあります。（むしろこちらの方が多い）

国家資格、民間の認定資格など様々な資格があります。それぞれの資格の詳細には触れませんが、「その業務を行えるだけの能力がある」ことの認定となるものが資格です。当該業務に関する（基本的な）知識を必要最低限持っていることがその資格を取るうえでの（少なくとも建前上の）条件となるわけです。

ということは、その試験に出てくる問題は、その「必要最低限」を網羅するものであるということです。

例えば、電気について最低限知りたいと思ったとき、電気工事士や電験（電気主任技術者）の勉強をするという感じです。

電気工事などの「実務」は資格を持っていないとやってはいけません。ですが、別に電気工事士の資格を持っていないと電子工作をやってはいけないというわけではありません。しかし、電気にに関する基本的な知識を手っ取り早く、しかも網羅的に知るためにには、とってもいいですよね。

そして、試験の出題範囲は、それなりに考え抜かれて選定された内容であり、その試験問題も基本的かつ良問である期待があります。

業界を広く薄く確実に学ぶには、手っ取り早く関連する試験を受けることを想定し、試験勉強をするのがよいでしょう。ターゲットが限定されている分、むやみやたらに広く勉強しなければならないと絶望することもなく、数日とか、数週間といった勉強時間で関連分野をさらっと知ることができるでしょう。それは絶対に無駄にはならず、たとえ試験を受けなくても、あるいは試験に落ちても、有用でしょう。漫然と勉強するよりよほど的確に、モチベーションを維持して勉強できます。

ただ、受験とセットにした方がいいことも明言しておきましょう。人のモチベーションの維持はなかなか大変です。締切がないと勉強しませんし、身銭を切ると勉強が捲るという話もあります。「受験日」はとってもよいマイルストーンです。それに向けて、過去問を何回か解いてみる、参考書を読む、などの勉強方法がありますが、その過程で必要な知識を抜け漏れなく入力することができます。

受験料はたいてい数千円や数万円です。学びの最短コースの一つとして検討してみてください。そして、たいていの資格には、上位資格があります。とっかかりが終わったら、さらに深く知るために上位資格を狙ってみるのもいいですね。

## 第 54 章

---

# 「読みたい」と「書ける」は違う

おやかた@ oyakata2438

本書のテーマは、学びでした。

予想以上の難産だった本書ですが、締切直前にして（あるいは公称締切を何度もブッちして）ようやく様々な章が集まってきたました。

本章を書いているのは締切の数日前です。最終的にどこまで伸びるのかとても楽しみですが、一方で、テーマ選定に関する学びがあったので、この段階で整理しておきましょう

当初この企画が立ち上がったのは 2023 年 9 月ごろでした。「学び」っていろんな知見があって、他の人たちの知見を読みたい。併せて自分が持ってる内容を書けばパワーのある本が作れるよね。ということで始まりました。

その目論見は大当たりで、本書には様々な知見が集まっています。編集しながら皆さんの原稿を楽しく読みました。

一方で、できあがるまでの過程は、難産に次ぐ難産でした。編集長（=私）が忙しいことを言い訳に、「原稿も集まっていないし、今回は落とします」と何回宣言したことか。もちろん締切直前に一気に増えるのはいつものことですが。

原稿が集まらなかった理由はいくつかあるでしょうが、やはりテーマが広すぎた点でしょう。この本全体としてみれば、仕事上の学び、学びとは、といった一定の流れに沿ったオムニバス形式の本となっているのですが、ではこの構成前の状態で「何が書けるか」と見まわしたとき、あまりにも漠然としすぎていて手が止まるでしょう。

著者の一人である FORTE さんが新しい著者を連れてきたとき、会話してみるとそれだって学びですよね、って即オチ 2 コマになるのはいつものパターンですが、それでもそういった形で、メンタリングをしたから見つけられたと言えるでしょう。一人で考えていては、こういった漠然としたテーマについて書くことはなかなか難しいというのが、本企画最大の学びです。

もう少し具体的にテーマ設定をしていれば、あるいはもっとスムーズに本が出来上がっていたかもしれませんね。

# 2 ミリの学びを捉える

なべくら@nabekurage

## 55.1 はじめに

もしあなたが「自分は学べていない」と感じていたら、この章が役に立つかもしれません。この章で書かれている内容は具体的な学習方法というより、単なる私の実体験に近いです。ですが、だからこそ、そんなあなたに読んで欲しいと思う内容だと思います。

## 55.2 『学び』とは何を指すのか

突然ですが、皆さんにとって『学び』とは何を指していますか？辞書を引けば、「学問をする。勉強する。」や、「教えを受けて身に付ける」といった言葉が並びます。他の章でも、各著者の方が考える『学び』について定義されているかも知れません。そんな中、私は『学び』とは、物事を行う前後で生じる自分自身の差分であると考えています。

私自身の話をします。今年は個人的な活動が少なく、「学べていない」という焦りがありました。仕事で求められるものが変化し、自分がそれについていけないと感じる場面が増え、心に余裕がなくなってしまっていました。もともと私は好奇心旺盛な方で、興味が湧いたらすぐ本を読んだりイベントに足を運んだりしていました。人との交流も大好きで、そこから得られる発見は刺激的で、心を躍らせるものでした。ところが、今年の後半頃から自分が変わってしまったように感じました。感情にモヤがかかり、何かを見て、感動したりエネルギーが湧き上がることが減っていました。それどころか、仕事以外での刺激を避けるようになり、本も読まない、イベントにも参加しない。結果的に「学べていない」という漠然とした思いに囚われていました。

そんなとき、ある一冊の本——『独学の地図』——で、「学びとは自分に生じた差分である」という考え方出会いました。これはその本を支える大切な要素のひとつで、私の固定観念を覆してくれました。

## 55.3 『差分』の解像度を上げる

具体的に『差分』とは何でしょうか。例えば、本を読んだ場合。読んだ内容から新しい知識を得た。さらに、本の中で紹介されていた別の本を読みたくなり、図書館で予約をした。これらは本を読む前後で自分に生じた『差分』と言えます。他にも、勉強会に参加して初めて聞くツールについて知った。気になって調べてみると最近 X や Zenn に関連記事が増えていて、どうやら最近流行っ

ているということがわかった。それなら週末に少し触っておこうと思い、チュートリアルのページをブックマークに保存した。その一連の流れも、自分の中に『差分』が生じていて、学びと呼べるわけです。

さらに小さな『差分』にも目を向けてみます。駅前のスーパーで卵が意外と安いと気づき、「次からはここで買おう」と考える。それも『差分』ですし、ラジオ体操をしたら体の強張りが軽減されていることに気づき、「明日も続けてみよう」と思う、これも『差分』です。『独学の地図』では、こうした日常にひそむわずかな差分を『2ミリの差分』と表現しています。ほんのわずかな変化でも学びとして捉えられるという点で、私もこのイメージは非常にしっくりきました。ほんの2ミリほどの変化でも、以前の自分と違う状態になっているのなら、それは学びと言えるのです。

### 55.4 終わりに

今までの私は、学びとは体系的な知識や新しい技術を身につけるような『大きな成果』だと思い込んでいました。

しかし、学び=差分だと考えると、本を読んだりワークショップに参加しなくとも、日々仕事や生活をしているだけで様々な要素から学びは生じていると考えることができます。言語化しづらい小さな発見や気づきも立派な差分であり、学びと言えるのです。

こうした小さな学びが積み重なれば、ある程度の量になった段階でブログ記事1本分の内容になり、さらに増えていけば本1冊分のコンテンツへと成長していくのです。

もしあなただが「自分は学べていない」と感じたり、他者と比べて焦燥感を抱いたりしているなら、今日の終わりに昨日との自分の差分を振り返ってみてください。たとえ出てきた差分がとても小さなものだったとしても、そこには間違いなく学びがあります。そして、その小さな積み重ねが、いつか今自分が「すごい」と感じている人々に近づく道筋になるのかもしれません。

# 学びと言ってもいろいろありすぎる～

おやかた@oyakata2438

ワンストップ学び。

学びは、方法も体験もいろいろあるよね、いっちょまとめてみるか、と思い立って始めた本書ですが、実は私自身のネタ出しに相当苦労しました。

そして、実はこの章を書いている（公称の）締切日現在、本編に掲載すべきネタはまだ上がっていません。<sup>\*1</sup>

その理由をここで考察しつつ、アタマの整理をして本編に繋げつつ、この章を書くことで、自身的ネタ出しをしつつ、原稿として 1 章追加して進捗を出そうという目論みです。

- 言い出しちゃが 1 本も出してないのはカッコ悪い
- 章の水増し
- ネタが出ないときのじたばたする方法として誰かの役に立つかな？
- あわよくばネタが降ってくるかも。

など、いろいろなメリットを期待することです。

さて、この本の反省の一つは、「テーマが広すぎた」という点でしょう。

<https://note.com/oyakata2438/n/n241fe6992846>

企画書にもありますが、さまざまな場面で学びがありますので、その知見・経験を集めてみよう、と始まった本企画ですが、私自身が予想上に苦しみました。そう。範囲が広すぎるのです。

なんか、広すぎてピンとこない。小ネタな気がして。などなど。いつも他の執筆者に言ってることなんですね…それでもニーズはあるよ。と。

ところが、いざ自分で書こうとすると、やっぱり難しいですね。頭の中にもやっとあって、しかも

では、どうしましょうか。じたばたあがくしかありません。

まずは、少し具体化してみましょうか。

- 仕事での学び
- コミュニティ
- 他の人から
- 学校とか

<sup>\*1</sup> その結果、2024 年 8 月の夏コミ合わせにする予定が、24 年 11 月の技術書典 17 に合わせて発行することになりました。6 月～7 月にかけて忙しくなってしまったということもあります…

学びを得るシチュエーションを考えてみましょうか。

仕事での学び、これは、業務を進めていくと、比較的自然となんらかの学びは生じますね。自分の業務上のスキルはもちろん範囲に入りますし、たとえ日々同じルーチンだったとしても、多少効率化といった形での改善はあるでしょう。それだって学びです。

コミュニティからは、勉強会という形で Give & Take(あるいは Takeだけかもしれません、それは追って… )で学びがあるでしょう。

他の人から。これはコミュニティでもいいですし、友人知人、その他、ひとつてに学ぶこともありますね。それらもこの本のスコープです。

当然学校とかといったところもありますね。高校・大学で学んだこと、大学に入りなおす、放送大学に籍を置く、あるいは資格試験で学ぶといったところもスコープに入れます。

ほら、少し具体化しましたね。

では、どれか一つ選んで、さらに深堀(あるいは具体化)をしてみましょう。その成果は、この本のどこかに入っているはずです・・

# あとがき

ここにあとがきを書く。

たくさん書く。

2023 年 8 月

編集長 おやかた@親方 Project 拝

# 著者紹介

## 編集長



親方 @oyakata2438 <https://twitter.com/oyakata2438>

サークル名：親方 Project

ワンストップ本シリーズ企画・編集（一部執筆）しています。コミケと技術書典に出没。ついに技術書同人誌博覧会（技書博）のコアスタッフとして運営側に参加。技術書イベントが増えて嬉しいけれど、イベント多すぎて外出チケットと徳が不足気味。徳を貯めるべく、家事をこなしつつ、ラボに遊びに行ったり、飲み会や懇親会で著者を新規開拓したり。著者募集はいつでもやってますので、ぜひご参加ください。

## 著者



ふーれむ @ditflame <https://twitter.com/ditflame>

Qiita でちょこちょこ技術ネタ書いたりしてます。 <https://qiita.com/ditflame>

大阪在住。SIer から社内 SE を経由して SaaS のドキュメント職人に着地しました。最近はもっぱら Maker 系イベントのイベントレポートを書いていたり、技術同人誌の校正をしています。この本も読みやすくなっていると良いのですが。



FORTE(フォルテ) <https://twitter.com/FORTEgp05>

Web アプリケーションのバックエンドエンジニアですが、いろいろやってます。Twitter、ブログ、Podcast 配信、数多くの趣味と楽しく活動中。



ゆーき @lizhi05853 <https://x.com/lizhi05853>

都内で Java エンジニアをしています。雪香（せっか）という個人サークルで『見に行こう！ コンピュータ史料展示マップ～関東編～』という本も出しました。



ターカー @tark\_ann [https://twitter.com/tark\\_ann](https://twitter.com/tark_ann)

北の大地で iOS エンジニアをしています。カンファレンスのカメラマンスタッフをやったりコミュニティ活動もやってたりします。



Camo @camomile\_cafe [https://x.com/camomile\\_cafe](https://x.com/camomile_cafe)

フリーランスでフロントエンドエンジニアをしている外国人です。毎日完全燃焼したい気持ちで過ごしています。朝型人間で 21 時あたりには寝る生活をしています。



こうの @hk\_it7 [https://twitter.com/hk\\_it7](https://twitter.com/hk_it7)

サークル名：世迷言ラボ

仕事は toC サービスを Ruby on Rails で作っている。エンジニアリングマネージャーらしい。東葛.dev というコミュニティやっています。



ぽによ @ponyoxa <https://twitter.com/ponyoxa>

主にバックエンドエンジニアをやってます。Joshi.ts という TypeScript の女性コミュニティを最近立ち上げました。



yumechi @\_\_yumechi [https://twitter.com/\\_\\_yumechi](https://twitter.com/__yumechi)

@\_\_yumechi@misskey.systems

サークル名：焼肉のために本気出します

のんびり Python のソフトウェアエンジニアやってます。たまに焼肉のために本気を出して同人誌を書くこともあります。このアイコンは技術書典のサークル画像として設定しているものです。本当はゲームばかりしていたいです。



もっと @mottox2 <https://twitter.com/mottox2>

最近は執筆よりもサポートの仕組み作りが多くなってきたソフトウェアデザイナーです。



みずりゅ @MzRyuKa <https://twitter.com/mzryuka>

のんびりごろごろ、ネコ、うさまる、ときめモ、技術の話は大好きです。サークル URAMASU で技術同人誌を頒布しています。最近のお気に入り言語は Elixir と Gleam です。



えるきち @erukiti <https://twitter.com/erukiti>  
サークル名：東京ラビットハウス <https://rabit-house.tokyo/>

2023年春にフロントエンドエンジニアからAIエンジニアに転向しました。好きなものとそうでないもので濃淡が激しすぎると評判（？）です。自分のサークルではここ数年、本を出せてない。

## 表紙イラスト・デザイン



トカゲ @Hiro\_Sauria [https://twitter.com/Hiro\\_Sauria](https://twitter.com/Hiro_Sauria)  
サークル名：冬眠ホタル

トカゲです。

表紙デザインだけでなく、本文にもお誘いいただき恐縮です！

最近はワンストップちゃんが親戚の子供のような気持ちで描けるようになってきました。本文掲載の表紙作成秘話で表紙のこだわりについて語ったので、よかつたらご一読ください。

# 本のタイトル

---

2023年12月31日 初版第1刷 発行

編集 おやかた

発行所 親方 Project

印刷所 K-9 <https://www.k-k9.jp/>