

ITエンジニアという仕事

三輪 智樹

LINEヤフー株式会社

今日のゴール

学科・進路選びのヒントを持ち帰る

OBのリアルな経験を共有する

- 大学でどう過ごすと良いかイメージできる
- 進路選択の判断材料が増える
- ITエンジニアという選択肢を知る

まず教えてください

プログラミング経験ある人？

授業でも趣味でも

まず教えてください

エンジニアに興味ある人？

なんとなくでOK

まず教えてください

行きたい学科がすでに決まっている人？

手を挙げてみてください

今日の話は

情報系に進む人以外にも聴いてほしい

学科・進路選びに役立つヒントをお伝えします

今日の流れ

1. 自己紹介

どうやってエンジニアになったのか

2. ITの仕事

エンジニアにも種類がある

3. 技術を覗いてみよう

LINEの裏側

4. 大学時代の過ごし方

エンジニアになるまで

5. AI時代のエンジニア

これからどうなる？

6. まとめ

今日お話ししたこと

1

自己紹介

どうやってエンジニアになったのか

プロフィール

基本情報

- **出身:** 早稲田実業学校 (2018卒)
- **大学:** 早稲田大学
基幹理工学部 情報理工学科
- **現在:**
LINEヤフー株式会社 エンジニア

やっていること

- 金融の部署でエンジニア
- 全社の生成AI活用Project

高校時代の自分

- 高1: 高等部から入学。陸上部に所属。勉強は普通（テストは平均点くらい）
 - 高2: 暗記が苦手だったため理系一択
 - 高3: 父の仕事の影響で機械系に興味 → 基幹理工学部 学系2を選択
- 友人がプログラミングやっていたが、全く興味がなかった

学部1年での転機

大学1年のC言語授業

- ・何も分からぬ...
- ・友人に課題を助けてもらう
- ・スラスラ解く友人がかっこいい



このままじゃヤバい

- ・長期休みで必死に勉強
- ・少しづつ理解できるように
- ・プログラミングの面白さに気づく

学部2年で情報理工学科へ進学

- 少しうまくいっただけだったが、情報理工学科へ進学を決意
 - 競技プログラミング: アルゴリズムの面白さに気づく
 - アルバイトの誘い: 授業+独学で力をつける
- 入学当初は機械系志望だったが、気づいたら情報系に夢中だった

インターンで実務経験

学部3年～大学院

- ・ インターン・業務委託を開始
- ・ 合計10社で実務経験
- ・ 様々な業界・技術に触れる

気づいたこと

- ・ 授業と実務は全然違う
- ・ 実際に使われるものを作る緊張感
- ・ やってみないとわからない

なぜエンジニアを選んだか

- 1 インターンで実務の面白さを知った
- 2 10社の経験で「これを仕事にしたい」と確信
- 3 学び続ければ成長し続けられる

ポイント: 実際にやってみて「合う」と思えた

1章のまとめ

明確な目標がなくとも、何があるかわからない

- ✓ 高校時代、エンジニアなんて考えてなかった
- ✓ 大学で出会って、ハマって、進路が変わった
- ✓ 特に学部1年は大切な時期

2

OBが語る、ITエンジニアのリアル エンジニアにも種類がある

IT業界の全体像

企画 → 設計 → 開発 → テスト → 運用

エンジニアは「開発」だけじゃない。全部に関わる

グループワーク

「LINE」を作るなら？

どんな役割の人が必要だと思う？

グループワーク

ルール

- ・グループで話し合い（3分）
- ・最低3つの役割を考える
- ・「なぜ必要か」も考えてみよう

思いついたものを自由にしてみよう！

みんなが考えた役割

「何を作るか決める人」「デザインする人」「プログラムを書く人」…

そういう役割、実際に存在します

それぞれの役割に名前がついている。

プロダクトマネージャー (PM)

何を作るか決める人

- ・「既読機能、つける？つけない？」
- ・「スタンプは有料にする？無料にする？」
- ・「次はどの機能を優先して作る？」

 プロダクトマネージャー

デザイナー

どう見せるか決める人

- ・ トーク画面のレイアウトは？
- ・ 吹き出しの色や形は？
- ・ スタンプの大きさは？

 デザイナー

フロントエンド / モバイルエンジニア

画面を作る人

- ・ モバイルエンジニア: スマホアプリの画面を作る
- ・ フロントエンドエンジニア: Webブラウザの画面を作る



モバイルエンジニア

バックエンドエンジニア

裏側の処理を作る人

- 送信ボタンを押したらメッセージが届
<
- 相手が読んだら「既読」がつく
- 過去のトーク履歴を保存する



バックエンドエンジニア

インフラエンジニア

土台を作る人

- LINEは日本だけで何千万人も使っている
- 全員が同時にメッセージを送っても落ちない
- データを安全に保存する



インフラエンジニア

QAエンジニア

品質を守る人

- メッセージが届かない...
- 既読がつかない...
- アプリが突然落ちる...



QAエンジニア

実は、これはほんの一部



データエンジニア

PMが判断するためのデータを集めて分析する



機械学習エンジニア

「おすすめスタンプ」などAI機能を作る



プラットフォームエンジニア

他のエンジニアが使う道具や環境を作る

作るものが変われば、必要な人も変わる

ゲームを作るなら？

- 3Dエンジニア
- サウンドデザイナー
- ゲームプランナー

自動運転を作るなら？

- 画像認識エンジニア
- センサーエンジニア
- シミュレーションエンジニア

ITの仕事は本当にいろいろある

2章のまとめ

- 1 1つのサービスを作るには、たくさんの役割が必要
- 2 今日紹介したのは代表的な例。実際はもっと多い
- 3 自分に合った役割が、きっとどこかにある

3

技術を覗いてみよう

Webページが表示されるまでの裏側

質問です

ブラウザでURLを入力してEnterを押したあと
何が起きてると思う？

考えてみてほしいこと

どこに繋がる？

URLからどうやって
目的地を見つける？

何を取得する？

テキスト？画像？
どんなデータ？

なぜ速い？

世界中のサイトが
すぐ表示される理由

なぜ安全？

パスワードが
盗まれない仕組みは？

グループワーク

Webページが表示されるまでの流れを
想像して図にしてみよう

グループワーク

ルール

- 正確さより想像力優先！
- 矢印で流れを書くだけでOK
- 「心配ポイント」も入れてみて（遅い、盗み見、サーバー落ちなど）

時間: 5分

何グループか発表してもらいます

どんな図になつ
た？

実際の流れ (簡略版)

1
DNS検索

2
接続確立

3
HTML取得

4
CSS/JS/画像

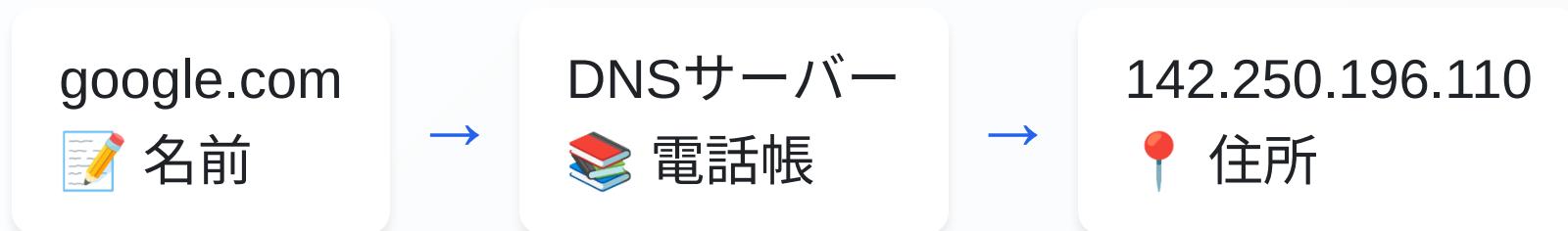
5
画面描画

URL → IPアドレス → サーバー → データ取得 → 表示

すごいポイント①：DNS

インターネットの電話帳

「google.com」を「142.250.196.110」に変換する仕組み



すごいポイント②：速度の工夫

普通に考えると

- ・日本→アメリカのサーバー
- ・毎回遠くまで取りに行く
- ・時間かかりそう...



実際は

- ・CDN（世界中にコピー配置）
- ・キャッシュ（一度取ったら保存）
- ・高速に表示される

すごいポイント③：セキュリティ

HTTPS（暗号化通信）

あなたとWebサイトの間の通信を暗号化。途中で盗み見できない

あなた
 暗号化



インターネット
 読めない



サーバー
 復号化

こんなことも考えてる



表示の最適化

画像を後から読み込んで、文字を先に表示



レスポンシブ対応

スマホでもPCでも見やすく自動調整



負荷分散

大量アクセスでも落ちないように複数サーバーで対応

エンジニアが考えること

- 1 大量のアクセスでも動く設計
- 2 どこからでも速く表示する仕組み
- 3 セキュリティと利便性のバランス
- 4 様々なデバイスへの対応

3章のまとめ

「Enterキー」を押した瞬間から
膨大な処理が動いている

- ✓ 普段見ているWebページの裏側は超複雑
- ✓ でも、それを作り人たちがいる
- ✓ 技術の積み重ねが「当たり前」を支えている

4

大学時代の過ごし方

エンジニアになるまで

大学6年間でやったこと

- **学部1-2年:** C言語 → 情報系進学・競プロ開始
- **学部3-4年:** インターン → 卒研・業務委託開始
- **修士1-2年:** 研究・就活・業務委託継続 → 内定

学業・研究

授業で学んだこと

✓ C言語

✓ アルゴリズムとデータ構造

✓ ソフトウェア工学

研究で得たもの

✓ 一つのテーマを深掘りする力

✓ 英語で論文を読む・書く力

✓ プrezen・発表の機会

C言語の授業

学部1年の必修科目

課題提出メイン・教場試験なし

当時の状況

- ・授業だけで理解するのは難しかった
- ・分かる人に頼るしかなかった

アドバイス

- ・できる人と仲良くなつておく
- ・少しでも予習しておくと楽
- ・AIに聞くのもあり（概念理解が大事）

C言語クイズ①：フォルダ構造

Q. 次のコマンドで移動先はどこ？

現在地: /home/user/documents

コマンド: cd ..

A

/home/user/documents/..

B

/home/user

C

/home

クイズ①：回答

正解：B) /home/user

現在地: /home/user/documents

↓ cd ..

移動先: /home/user

クイズ①：解説

「..」とは？

- ・親ディレクトリを表す
- ・1つ上の階層に戻る
- ・「cd ..」で上に移動

フォルダ構造

```
/home
  └── user           ← ここに移動
      └── documents  ← 現在地
```

覚えておこう：「.」は現在地、「..」は1つ上

C言語クイズ②：if文【何が表示される？】

```
int x = 5;
if (x > 10)    { printf("A"); }
else if (x > 3) { printf("B"); }
else            { printf("C"); }
```

A

A が表示

B

B が表示

C

C が表示

クイズ②：回答

正解：B

```
int x = 5;
if (x > 10) {           // 5 > 10 ? → ✗
    printf("A");
} else if (x > 3) { // 5 > 3 ? → ✓ ここ !
    printf("B");
} else {
    printf("C");
}
```

クイズ②：解説

1 上から順に評価: 最初に真になった条件だけ実行

2 **else if**: 前の条件が偽のときだけ評価される

3 **else**: どの条件にも当てはまらないとき

ポイント：条件分岐は「上から順番に」チェックされる

C言語クイズ③：配列

Q. 配列の3番目の要素にアクセスするには？

```
int arr[5] = {10, 20, 30, 40, 50};
```

A

arr[3] → 結果: 40

B

arr[2] → 結果: 30

C

arr[1] → 結果: 20

クイズ③：回答

正解：B) arr[2] → 30

```
int arr[5] = {10, 20, 30, 40, 50};  
// [0] [1] [2] [3] [4]  
//  
//  
//  
// 1番目 3番目
```

クイズ③：解説

日常の数え方	プログラミング
1番目、2番目、3番目...	arr[0]、arr[1]、arr[2]...

超重要！ プログラミングでは「0から数える」
これを忘れると「Off-by-oneエラー」というバグの原因に...

C言語クイズ④：整数の割り算

Q. 次のコードでzの値は？

```
int x = 7;  
int y = 2;  
int z = x / y;
```

A

3.5

B

3

C

4

クイズ④：回答

正解：B) 3

```
int x = 7;  
int y = 2;  
int z = x / y; // 7 / 2 = 3 (3.5ではない！)
```

クイズ④：解説

C言語 (型を宣言する)

```
int x = 7;      // 整数型
int y = 2;      // 整数型
int z = x / y; // → 3
```

int同士 → 結果もint (切り捨て)

Python (型を宣言しない)

```
x = 7
y = 2
z = x / y # → 3.5
```

自動で小数になる

型とは？ データの種類のこと。int=整数、double=小数など 宣言が必要
C言語はPythonと違い「この変数は整数です」と

C言語クイズ⑤：ループの回数

Q. このループは何回実行される？

```
int count = 0;
for (int i = 0; i <= 5; i++) {
    count++;
}
printf("%d", count);
```

A

B

C

クイズ⑤：回答

正解：B) 6回

```
for (int i = 0; i <= 5; i++) {  
    count++; // i = 0, 1, 2, 3, 4, 5 の6回実行  
}
```

ループ	i の値	条件 <code>i <= 5</code>
1回目	0	✓ 真
2回目	1	✓ 真

クイズ⑤：解説

条件	ループ回数	i の値
<code>i < 5</code>	5回	0, 1, 2, 3, 4
<code>i <= 5</code>	6回	0, 1, 2, 3, 4, 5

Off-by-one エラー : 1つずれるバグ

「<」と「 \leq 」の違いで回数が変わる！よくあるミス

覚え方 : 0から始めて「 $< N$ 」なら N 回、「 $\leq N$ 」なら $N+1$ 回

C言語のまとめ

一度理解するとスラスラできる

- 1 最初の壁が一番高い: 知っている人と知らない人で差が出やすい
- 2 英語に似ている: 文法を覚えれば読み書きできるようになる
- 3 一度身につけば一生モノ: プログラミングの土台になる

アルゴリズムとデータ構造

学部2年の科目

数学好きなら絶対ハマる

どんな授業？

- 効率的な問題の解き方を学ぶ
- 理論と実装の両方がある
- 頭の中のイメージを現実に落とし込む

扱う内容

- ソート（並び替え）
- 探索（二分探索など）
- グラフ・木構造

問題：二分探索

1～100の中から数字を当てるゲーム

ルール

- 1～100の中から1つの数字を選ぶ
- 「もっと大きい/小さい」のヒントあり

目標

- 最小の回数で当てたい！

Q. 最悪でも何回で必ず当てられる？

二分探索：回答

答え：最悪7回で必ず当たる！

戦略：毎回「真ん中」を聞く → 候補が半分になる

1回目で 50個 → 2回目で 25個 → ... → 7回目で 1個 に！

二分探索：解説（具体例）

1 「50」 → 「もっと大きい」 → 範囲: 51～100 (50個)

2 「75」 → 「もっと小さい」 → 範囲: 51～74 (24個)

3 「62」 → 「もっと大きい」 → 範囲: 63～74 (12個)

… 繰り返すと → 最悪7回で1個に絞れる！

なぜ7回？ $2^7 = 128 > 100$ なので、7回あれば100個から必ず見つかる

アルゴリズムのまとめ

おすすめ：AtCoder

競技プログラミングで腕を磨こう

1 頭の体操になる：論理的思考力が鍛えられる

2 イメージ→コードの力：考えたことを実装に落とし込む

3 コードを読む力：他人の解法から学べる

4 授業の先取り：基礎ができた状態で臨める

就活・進路選択

1

自己分析

何が好き？何が得意？

2

企業研究

どんな会社がある？

3

インターン

実際に体験してみる

4

選考・内定

面接・技術テスト

エンジニア就活の特徴

他の職種より差別化が明確



コーディング試験

コードが書ける・書けないがはっきりわかる



成果物が証拠になる

自分の作ったもの・携わった仕事が面接で語れる

大学入学までにできること

- 1 プログラミングに触れてみる: Python、C言語など何でもOK
- 2 PCに慣れておく: タイピング、ショートカットキーなど
- 3 興味のある分野を探す: Web、AI、ゲーム...色々ある
- 4 英語に触れておく: ドキュメントは英語が多い

4章のまとめ

大学は「やりたいこと」を見つける場所

授業だけじゃなく、自分から動くことが大事

- ✓ 学業と課外活動のバランスが重要
- ✓ インターンは早めに経験しておくと有利
- ✓ 入学前から少しづつ準備できることがある

5

AI時代のエンジニア

これからどうなる？（2026年1月時点）

今日話すこと

いま何が起きている？

エンジニアの価値はどう変わった？

だから、なぜプログラミングを学ぶべき？

まず現実：AIはコードを書ける



AIは「それっぽいコード」を高速に作れる



画面の部品 / 定番の処理 / テスト など



現場でも普通に使われ始めている



だから「勉強いらない？」と思うのは自然

例：フォーム、ログイン周り、データの整形...

でも重要：AIは"責任"を取れない

AIが出したコードは…

正しい？

仕様に合ってる？

安全？

危ない穴はない？

続けられる？

あとから直しやすい？

価値が下がった能力

(以前ほど重要ではない)



定型作業としての実装

CRUD、テンプレUI、ボイラープレート



ググってコピペでつなぐ



構文・APIを暗記する



まとめ：「作業量」や「実装スピード」だけでは差が出にくい

変わらず必要な能力

(AIがいても消えない)



問題設定：何を作るべき？



設計：どう分ける？どう続ける？



デバッグ：なぜ壊れた？



レビュー：正しい？安全？



今まで以上に必要な能力



AIを使いこなす力

指示 → 検証 → 修正



品質保証

テスト、監視、再現性



セキュリティ意識



システム全体を見る視点

締めの一言

「AIがあるから勉強しなくていい」ではなく

「AIがあるからこそ、分かる人が強い」

6

まとめ

今日お話ししたこと

振り返り

1 **自分の話:** エンジニアへの道は一つじゃない

2 **ITの仕事:** チームで協力してサービスを作る

3 **技術の深さ:** 「当たり前」の裏に膨大な技術

4 **大学時代:** 今から準備できることがある

5 **AI時代:** 道具として使いこなす側に

最後に伝えたいこと

「今わからなくても大丈夫。
やってみたら意外とできる。
興味を持ったら、まず触ってみて。」

Q&A

なんでも聞いてください！

ありがとうございました！

質問があればいつでも連絡ください

@tamofplease | mmiwatomoki@gmail.com