

はじめに

本書は、NTTテクノクロス株式会社の中で技術を愛し、人に教えることが大好きな有志が、「**生成AIを利用したプログラミング体験**」についてまとめた本です。

執筆に至った背景

NTTテクノクロス株式会社では、

- ・社員研修
- ・ジョブシャドウイング(中高生を招き、プログラミング体験やキャリア教育等を行う社内イベント)
- ・インターンシップ

で「**生成AIを利用したプログラミング体験**」を行なっています。

「生成AIを利用したプログラミング体験」とは、『**ChatTX(ChatGPTに相当する生成AI)**』と『**DartPad**』や統合開発環境(**IDE**)を利用して簡単なアプリケーションを作成する体験のことです。私達は、活動を通じてITに関する人材を1人でも多く増やしたい。活動をキッカケに未経験の人でもIT業界に興味を持ち、将来的にエンジニアとして活躍頂きたい思いを持っています。

本書では、**社員研修とジョブシャドウイングで使用するテキストを手を加えずに掲載**しています。※1

これから「**社員研修や勉強会で生成AIを利用したい**」と考えている方々の一助となり、現場で活用頂ければと思っています。

※1 物理本を購入頂いた皆様：製本の関係で図等見づらい箇所がある場合は、お手数をお掛けしますが**技術書典オンラインマーケット**から電子書籍版をダウンロード頂ければと思っております。

お問合せ先

本書に対するお問合せは下記メールアドレス宛にお願いいたします。

- xmobile.support-ml@ntt-tx.co.jp

注意事項

本書は情報の提供のみを目的としています。

免責事項

本書の内容を用いた開発、運用は、必ずご自身の責任と判断によって行ってください。

開発、運用の結果について、著者はいかなる責任も負いません。

なお、この本は2024/09/13時点の内容です。開発環境や言語のバージョンアップ等により内容に差異が生じる可能性があります。その点は了承願います。

商標について

本書に掲載するシステム名や製品名は、関係各社の標章または登録商標です。

本書では、™、®等は省略しています。

著作権について

「Flutter」はGoogle LLCに帰属します。

その他の本書に掲載するソフトウェア、アプリ、プログラム等に関する著作権は、関係各社に帰属します。

本書では、©等は省略しています。

対象読者

本書は、『プログラミング未経験者・初心者』で生成AIにおける以下項目とカテゴリを学びたい方を対象としています。

- **生成AIの学習項目**

①アプリ開発&生成AI活用の流れを体験する

②アプリ開発で生成AIを実際に活用する

③生成AI活用の勘所と注意点をつかむ

項目

- **生成AIのカテゴリ**

コードの生成

コードのデバッグ

本書での対象

コードのレビュー

リファクタリング
&モダナイズ

テストケースの生成

パフォーマンス改善

生成AIカテゴリ

本書を読み進めるに際し、前提となる知識はありません。ぜひそれぞれの現場でご活用いただければと思います。

動作環境

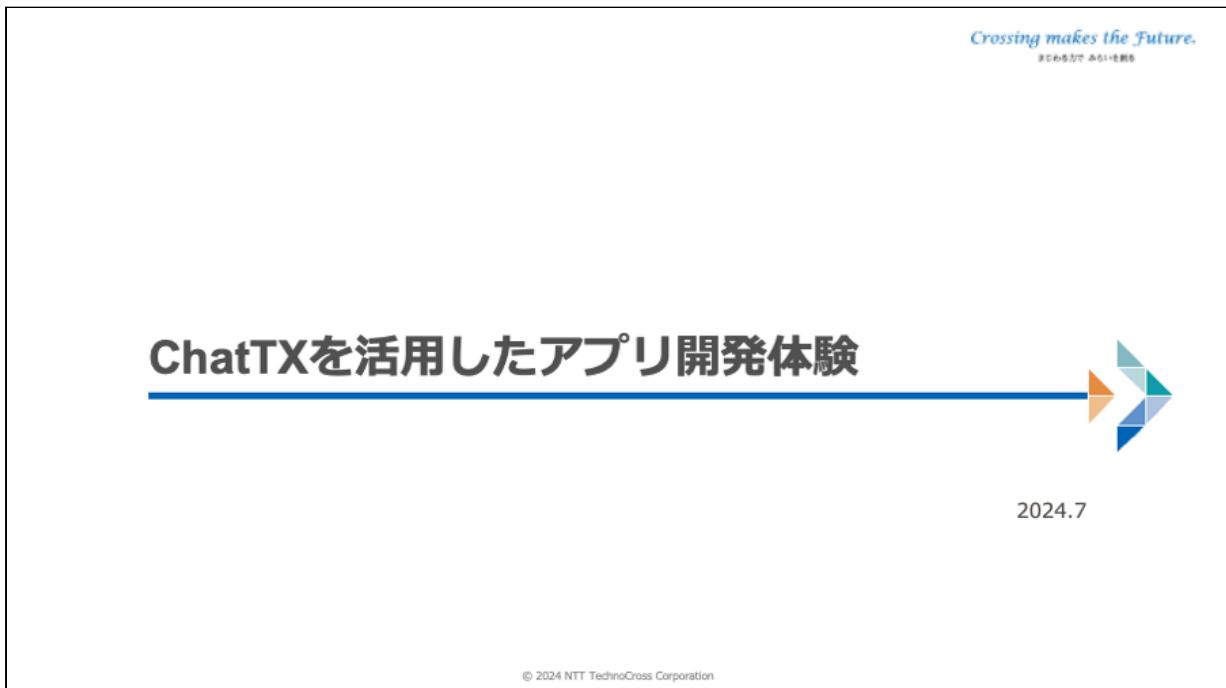
本書で提供するコンテンツは、以下環境で動作確認を行っています。

- Dart SDK 3.5.1 and Flutter SDK 3.24.1

目次

• はじめに	1
• 対象読者	3
• 動作環境	3
• 1.本書の概要	5
◦ 1.1.研修のアジェンダ/目的	7
◦ 1.2.生成AIの概要	9
◦ 1.3.生成AIが効果的なところに利用しよう	15
• 2.Flutterおよび開発環境の概要	19
• 3章.生成AIを活用したアプリ開発実践	22
◦ 3.1.生成AIを活用したアプリ開発の流れ	23
◦ 3.2.みんなで進める課題：計算機アプリ	24
◦ 3.3.追加演習	32
◦ 3.4.振り返り	36
• おわりに	39
• 表紙デザインについて	40
• 著者紹介	41

1.本書の概要



研修タイトル

「はじめに」で紹介したとおり、研修では、生成AIの環境として、当社のChatTXを利用して実施しています。

講師紹介

DPBD(技術支援チーム)



神原 健一 神長 貴博 戸部 雄太郎

DSBD(ChatTXチーム)



山本 昂輝

© 2024 NTT TechnoCross Corporation

2

講師紹介

社内の2つの部署で連携し、業務・プライベートで普段から生成AI活用に取り組んでいる4人の知見を広く展開すべく、本研修を企画・実施しました。巻末にそれぞれのプロフィールを掲載していますので、合わせてチェックいただけますと幸いです。

1.1.研修のアジェンダ/目的

研修のアジェンダ

講義編

- ・本研修の目的・対象
- ・ChatTXの概要・使い方基本
- ・アプリ開発における生成AI(ChatTX)利用の勘所・注意点

演習編

- ・研修環境説明・接続確認
- ・利用する開発環境・言語の概要
- ・生成AI(ChatTX)を活用したアプリ開発
- ・振り返り(知見の共有)

© 2024 NTT TechnoCross Corporation

7

研修のアジェンダ

実際の研修では0.5日(半日)で集合形式で取り組んでもらっています。オンラインでも実施可能な内容ですが、初めての研修だったこともあり、雰囲気を全員で分かち合うことを優先するため、対面開催としました。

「講義編」で生成AIなどに関して、基本的な内容を共有した後、「演習編」で生成AIを使いながら、実際にアプリ開発に取り組むという流れにしています。最後に、講師・参加者問わず、知見をみんなで共有すべく、振り返りの時間も設けています。

本研修の目的

①アプリ開発＆生成AI活用の流れを体験する

②アプリ開発で生成AIを実際に活用する

③生成AI活用の勘所と注意点をつかむ

© 2024 NTT TechnoCross Corporation



8

本研修の目的

本研修では、「アプリ開発と生成AI活用を手を動かしながら体験できる」ことに重きを置いています。具体的には、この3点を研修のコンセプトにしています。

1.2.生成AIの概要

生成AIについて

- 生成AIは、**データを基に新しいコンテンツを生成するAI**のこと。
テキスト、画像、音楽、動画などの新しいデータを作ることが可能。
- ChatTXでは生成AIの中でもLLM（大規模言語モデル：Large language Models）を使うことができます。
- LLMは特に自然言語処理（NLP）に特化した生成AIの一種です。
(LLM=生成AI というよりは LLMc生成AI)

「ChatTXを活用したアプリ開発体験」という言葉を
画像で表現して

生成AI

© 2024 NTT TechnoCross Corporation

11

生成AIについて

研修の本題に入る前に、「生成AI」とは何かについて説明しました。生成AI自体の説明に加え、世の中に存在する生成AIサービスを紹介することで、受講者が生成AIを活用するイメージを持てるようにしました。

「生成AI」という言葉が広く知られるようになったきっかけの一つに、ChatGPTの影響があります。そのため、「生成AI=LLM（大規模言語モデル）」と認識している方もいるかもしれません、LLMは言語に特化した生成AIの一種に過ぎず、生成AIは画像、音楽、動画など、多種多様なコンテンツを生成できるAI全般を指します。

生成AIの例

Dall-E3

テキスト to 画像

GPT-4o

テキスト to テキスト

画像 to テキスト

※GPT-4oのモデル自体は音声、画像、テキスト、映像、…とマルチモーダル対応しています

他にも...

- テキスト生成 : Gemini (Google) 、 Claude (Anthropic)
- 音声生成 : FurtureVoice (NTT-TX) 、 Text-to-Speech AI (Google)
- 動画生成 : Sora (OpenAI)
- 画像生成 : Stable Diffusion (Stability AI)

© 2024 NTT TechnoCross Corporation 12

生成AIの例

生成AI活用のユースケース

ChatGPT

Gemini※

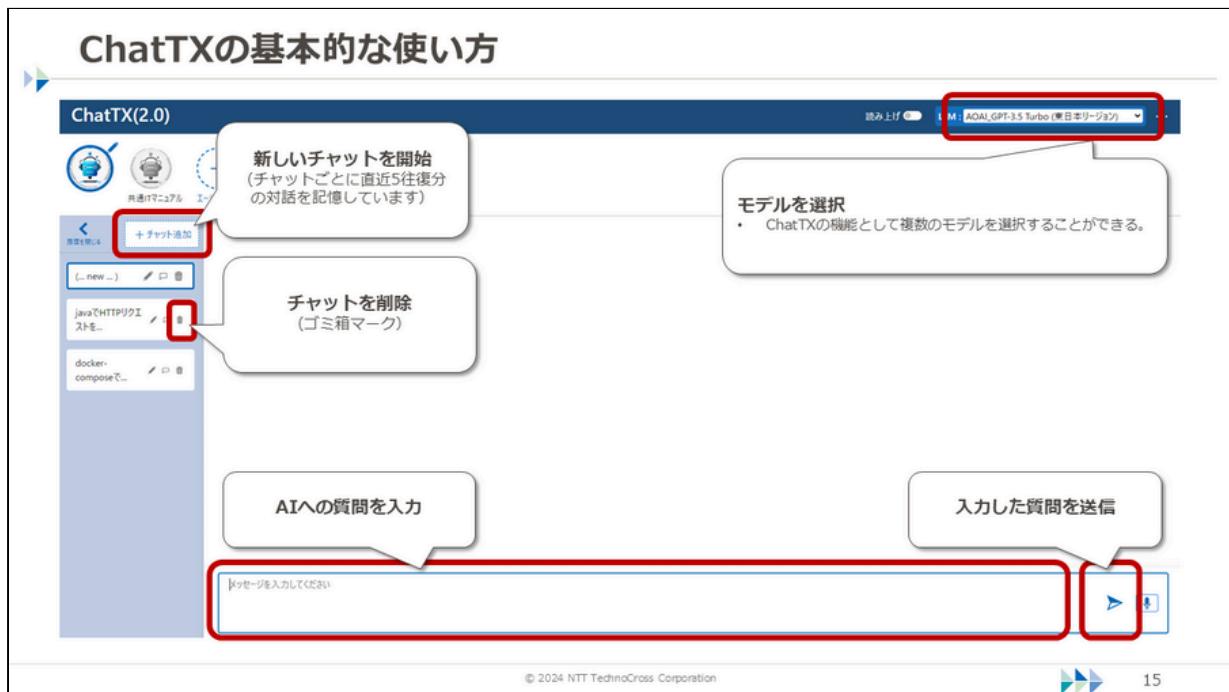
Claude

Flutterでアナログ時計を表示するコードを作って

※検証時の挙動としては、main関数が含まれないコードが生成された。そのため追加指示を実施

© 2024 NTT TechnoCross Corporation 13

生成AI活用のユースケース



ChatTX の基本的な使い方

本研修で使用した生成AI環境「ChatTX」の画面です。これは、生成AIのチャットサービスとして基本的な機能を備えています。ChatTXでは、複数のLLMを利用することができます。

プロンプト(ChatTXに入力する質問)の個人的コツ①

・要件の具体化

- ・単に「時計アプリを作って」と質問しても思った通りのコードを作成してくれないので、言語や実行環境、具体的な仕様を明確にする
- ・作成されたコードはコピペで実行するのではなく、一通り目を通してどのような処理を実装されたかを理解することが重要

```
以下のプログラムを作成して下さい。  
# 環境  
・言語 : Dart version x.x.x  
・実行環境 : Flutter  
  
# 仕様  
・実行するとアナログ時計が表示される。  
・~~~
```

```
以下のPythonプログラムを作成して下さい。  
・FastAPIを使用  
・起動するとHTTPリクエストを受け付ける  
・"/chat"へのPOSTリクエストを~~~  
：
```

© 2024 NTT TechnoCross Corporation



17

プロンプトのコツ1

生成AIに質問を入力する際、講師陣からいくつかのコツを紹介しています。一つは「質問の具体化」です。例えば、アプリ開発に用いる言語や実行環境、仕様を明確に指示することで、より目的に合ったソースコードを得やすくなります。また、出来上がったソースコードを確認せずコピペするのではなく、どのような意図がありそのソースコードが出来上がったかを理解することで、アプリ全体の理解や品質を向上させることができます。

プロンプト(ChatTXに入力する質問)の個人的コツ②

・ 実際のソースコードやエラー文を含めて質問する

- ・ 言葉だけで思い通りのコードを作ってくれることは難しいので、コードの断片やエラーが起きた際はエラー文も含めて質問してみる
- ・ ソースコードであることをAIに明確に伝える方法として ```` (バッククオート3つ)で囲う方法があります

(この書き方を、Markdown（マークダウン）という軽量マークアップ言語における「コードブロック」と呼びます。

以下のDartプログラムを、○○の仕様になるよう修正して下さい。

```
```dart
import 'package:flutter/material.dart';

void main() {
 runApp(MyApp());
}
```
```

```

ソースコード

Python3.11で以下のエラーが置きました。  
OSはAmazonLinuxです。  
解決方法を教えてください。

```
Traceback (most recent call last):
 File "example.py", line 20, in <module>
 main()
 File "example.py", line 17, in main
 result = 1 / 0
ZeroDivisionError: division by zero
```

エラー文

© 2024 NTT TechnoCross Corporation



18

### プロンプトのコツ2

生成AIでアプリ開発を行う際、一度作ったソースコードを修正したり、エラーが発生した場合に解決を図ることもあるかと思います。その際、ただソースコードやエラーメッセージを入力するのではなく、それがソースコードやエラーメッセージであることを生成AIに明確に伝えることが重要なテクニックです。

## プロンプト(ChatTXに入力する質問)の個人的コツ③

### ・繰り返し、段階的に聞いてみる

- 理想的なプログラムを一発で作ってくれることはまれなので、「ここの処理をこうして」「ここはこういう仕様で作り直して」と繰り返し聞いてみる
- 実際に作られたソースコードの解説をしてもらうのもOK



Dartで○○のアプリを作成して

```
class Person {
 String name;
 int age;

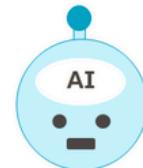
 Person(this.name, this.age);
 void introduce() {
 print('Hello, my name is $name and I am $age years old.');
 }
}
```

□□の関数は結果がintになるように修正して

```
import 'dart:io';
Future<void> main() async {
 // 2秒待ってからデータを出す
 await Future.delayed(Duration(seconds: 2));
 print('Data loaded');
}
```

先程のコードと具体的にどこが変わりましたか？

```
void main() async {
 print('Data loaded');
 await Future.delayed(Duration(seconds: 2));
 print('Data loaded');
}
```



□□関数の内部処理で…

© 2024 NTT TechnoCross Corporation



19

### プロンプトのコツ3

また、一度の質問で完璧なソースコードを作成することを目指すのではなく、生成AIと繰り返しやり取りをしながらブラッシュアップすることで、より良いアプリに仕上げていくことも重要です。これは人間同士のやり取りに近い部分もあるでしょう。

## 1.3.生成AIが効果的なところに利用しよう

### ChatTXが効果的なところに利用しよう

ChatTXを効果的に使う

ならば

得意なところに適用する

© 2024 NTT TechnoCross Corporation

20

カバー

この節では生成AIを効果的に活用するための考え方についてお伝えします。

「生成AIを効果的に使うならば、得意なところに使う。」当たり前のことではありますが、これが実態だと著者は考えます。

以下、このようなことを説明しています。

- ・「得意なところ」について考える前に、「苦手なこと」を先に考える
- ・生成AIの得意なこと「変換」
- ・さらにうまく活用するための注意点

## ChatTX(生成AI全般)が苦手なことは何か

- ・元となる情報が少ないものに対する生成
- ・新しいものに対する生成

→このようなことを避けねば精度が高くなる可能性がある。

© 2024 NTT TechnoCross Corporation



21

生成AIが苦手なこと

## ChatTXが扱うのが得意なものは？

- ・元となる情報が多いもの
- ・古くからのもので枯れている、普遍的である。

© 2024 NTT TechnoCross Corporation



22

生成AIが得意なこと

## ChatTXの得意なことを一言で：変換

- 一定の確定した普遍的な情報をもとに何か作業をする
- 既存のものをもとに、近い何かに作りえる
- 既存の情報を組み合わせる

© 2024 NTT TechnoCross Corporation



### 変換 1

#### 変換の例：情報の収集

- 英語の技術ドキュメントを日本語に要約する
  - プロンプト（後略）

下記は、バックグラウンドからのアクティビティ起動制限についての記載です。内容を要約してください。

Restrictions on starting activities from the background

bookmark\_border

Android 10 (API level 29) and higher place restrictions on when apps can start activities when the app runs in the background and keep the user more in control of what's shown on their screen.

- 回答（後略）

バックグラウンドからのアクティビティ起動制限の要約

概要

Android 10 (APIレベル29) 以降では、アプリがバックグラウンドでアクティビティを起動する際に制限が設けられ、ユーザーが画面に表示される内容をよりコントロールできるようになります。

通知を使用する代替案

バックグラウンドでアクティビティを直接起動する代わりに、アプリは緊急情報をユーザーに提供するために通知を送ります。通知を使用することで、ユーザーは現在のコンテキストを維持しつつ、必要な情報に迅速に対応できます。

アクティビティを起動できる条件

Android 10以降でアプリがアクティビティを起動できる条件は以下の通りです：

© 2024 NTT TechnoCross Corporation



### 変換 2

## 変換の例：情報の整理

- まとめた情報を似ている話題ごとにまとめる
    - プロンプト（例えばある技術の変更内容を整理したい例）

この変更内容について、近い内容でカテゴライズすることはできますか？粒度については一旦お任せしますが、大分類・中分類くらいの2段階でまとめられると良いです。

• 回答例（後略）

もちろん、変更内容を大分類と中分類でカテゴライズしてみます。

新機能とAPIの変更まとめ

新機能

- ・テキスト関連
    - `parseAsHtml` メソッドを追加し、HTMLタグ付き文字列を `AnnotatedString` に変換可能に。
    - `BasicTextfield2` が手書き機能をサポート。
  - ・スクリーンショット関連
    - Composeスクロールコンテナでの長いスクリーンショットの実践的サポートを実装。
  - ・グラフィックス関連
    - 新しい `GraphicsLayer` APIを導入し、ディスプレイリストに描面コマンドを記録する機能を提供。
  - APIの変更
    - ・セマンティクス関連
      - `SemanticsProperties.Editable` を `IsEditable` にリネームし、`SemanticsPropertyReceiver.setEditable` を削除。
      - `GetScrollUIViewportセメント`セマンティクスアクションを追加。
      - 新しいセマンティクスAPI `unset!` を追加。
      - `maxTextLength` セマンティクスプロパティを追加。
  - ・テキスト関連
    - `Textfield` の `text` 属性を `String` から `Text` に変更。
    - `Textfield` の `placeholder` 属性を `Text` に変更。
    - `Textfield` の `label` 属性を `Text` に変更。
    - `Textfield` の `error` 属性を `Text` に変更。

© 2024 NTT TechnoCross Corporation



25

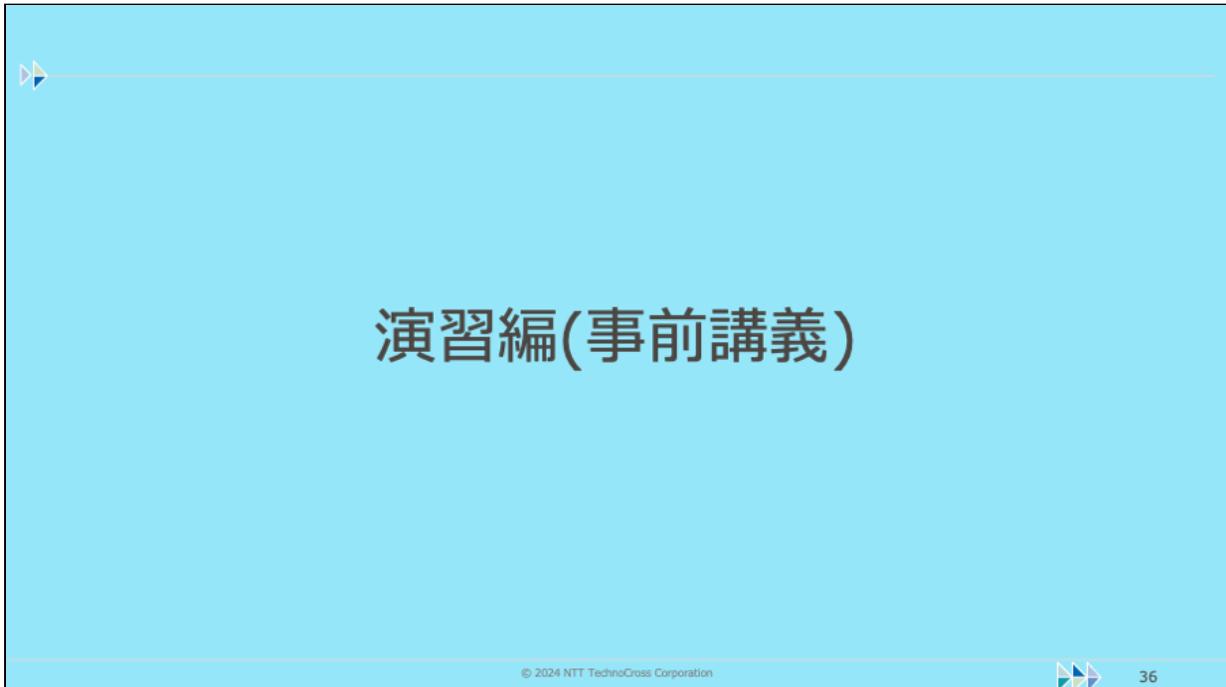
變換 3

#### さらにうまく活用するための注意点

- ドキュメントの要約は完璧ではない
    - 時々スキップされる
    - スキップされていても、一からやるよりは早いという割り切り
    - 例えば以下のようなプロンプトで対応する。
      - 「〇〇の部分がスキップされていませんか？他にスキップされている部分がないか再度チェックしてください。」
  - ソースコードの内容は完璧ではない
    - 誤りが入ることもある。
    - ソースコードはすぐに確認する、確認できる環境を準備する。
    - 細かい単位で取り込み、細かい単位で動作確認することができると理想。
      - 本日の演習は確認しやすい環境です！

© 2024 NTT TechnoCross Corporation

## 2.Flutter および開発環境の概要



カバー

アプリ開発の題材として、Flutterを採用しました。理由は幾つかあります。その1つが、後述するクラウド上の実行環境を用いることでプログラミングした結果をその場で実行し、動作確認を比較的、容易に行えるためです。演習に入る前に、Flutter未経験の受講者がいることも想定して、簡単な講義を実施しています。

Flutterについては、弊社の別の技術書「ゼロから始めるFlutterアプリ開発入門 2023.11版」の方で詳しく取り上げています。合わせて読んでいただけすると幸いです。

## Flutter(フラッター)について

- ・共通のソースで複数のプラットフォーム向けアプリを同時に開発できるフレームワーク(クロスプラットフォーム開発)
- ・Googleが開発している
- ・公式サイト：<https://docs.flutter.dev/>

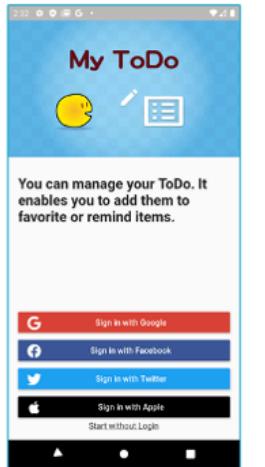
© 2024 NTT TechnoCross Corporation



37

## Flutter概要

### Flutterを使ったアプリの例



MyToDo(Todoアプリ)



Sekaiphone Pro(スピーチ翻訳)



© 2024 NTT TechnoCross Corporation



38

## Flutter概要

## Dart(ダート)について

- Flutterでのアプリ開発に利用するプログラミング言語
- Java/JavaScriptに似た特徴を持つ言語
- 公式サイト：<https://dart.dev/>

```
import 'package:flutter/material.dart';

void main() {
 runApp(
 Center(
 child: Text(
 'こんにちは、Flutter',
 textDirection: TextDirection.ltr,
),
),
);
}
```



© 2024 NTT TechnoCross Corporation



41

## Dart概要

### DartPad

Webブラウザ上でコードを書く→その場で動作確認可能

公式サイト：<https://dartpad.dev/>

The screenshot shows the DartPad interface. On the left, there's a modal for creating a new pad, with 'Dart' selected. The main area has tabs for 'Dart' and 'Flutter'. A code editor window titled 'Counter app in Flutter' contains the following Dart code:

```
import 'package:flutter/material';
void main() => runApp(MyApp());
class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
 return MaterialApp(
 title: 'Flutter Demo',
 debugShowCheckedModeBanner: false,
 theme: ThemeData(
 primarySwatch: Colors.blue,
),
 home: MyHomePage(title: 'Flutter Demo Home Page'),
);
 }
}
```

To the right of the code editor is a preview window titled 'Flutter Demo Home Page' showing a simple counter application with a button labeled '+'. Below the preview, a note says 'You have pushed the button this many times: 0'. At the bottom of the page, there are links for 'Privacy notice', 'Send feedback', 'no issues', and 'Based on Flutter 1.20.0 Dart SDK 2.9.0'.

© 2024 NTT TechnoCross Corporation



43

## DartPad

## 3章.生成AIを活用したアプリ開発実践



### 演習編

© 2024 NTT TechnoCross Corporation



46

#### 演習カバー

ここからは実際にアプリ開発を実践する演習に入ります。 前の章で説明したDartPadと、ご自分が利用可能なテキスト生成AIを手元にご用意ください。

## 3.1.生成AIを活用したアプリ開発の流れ

### 生成AIを活用したアプリ開発の流れ

#### ・目的

- ・生成AIとやり取りしながら簡単なアプリを作り上げる体験をする。
- ・生成結果に「揺らぎ」があることを体験する。
- ・ソースコードの生成と動作確認のサイクルを繰り返しながらアプリをブラッシュアップする体験をする。

#### ・進め方

- ・1. アプリのソースコードを生成するためのプロンプトを検討する。
- ・2. ソースコードを生成する。
- ・3. ソースコードを使って動作確認する。
- ・4. 改善点・課題をリストアップし、改善のためのプロンプトを検討する。
- ・5. 2から4のサイクルを繰り返す。

© 2024 NTT TechnoCross Corporation

47

### 生成AIを活用したアプリ開発の流れ

研修ではこの流れに沿ってアプリ開発を実施しました。特に、ソースコードの生成から改善の検討、実行のサイクルを繰り返すことで、生成AIを利用したアプリ開発の特徴を体感することができます。

以降の説明ではChatTXを利用してますが、ご自身の利用可能な環境に合わせて読み替えてください。

## 3.2.みんなで進める課題:計算機アプリ

### 演習テーマ1: みんなで進める課題（この後細かく説明します）

- 計算機アプリを作成する

- この画像のようなアプリのソースコードを生成してみましょう。

※プロンプト例は後述（一緒にやります）

- 条件

- Flutterのソースコードとします

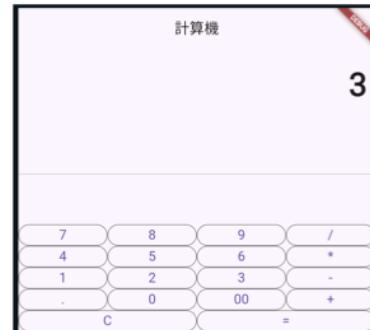
- 基本課題

- 見た目の改善

- 計算結果を目立たせる など

- 追加課題

- 関数電卓のように、平方根など四則演算以外の計算ができる など



© 2024 NTT TechnoCross Corporation



48

みんなで進める課題

### みんなで進める課題：計算機アプリ

- アプリを作るプロンプト

- 以下のようなプロンプトをChatTXに与えます。

Flutterで計算機のアプリを作ろうと思います。  
サンプルコードを生成してください。

ユーティ 2024/05/25 19:24:19  
Flutterで計算機のアプリを作ろうと思います。  
サンプルコードを生成してください。

AI[OpenAI API] 2024/05/25 19:24:19  
もちろんです！Flutterでシンプルな計算機アプリを作成するためのサンプルコードを提供します。このアプリは基本的な四則演算（足し算、引き算、掛け算、割り算）をサポートします。  
以下がそのサンプルコードです：

```
import 'package:flutter/material.dart';

void main() {
 runApp(CalculatorApp());
}

class CalculatorApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
 return MaterialApp(
 title: '計算機',
 theme: ThemeData(
 primarySwatch: Colors.blue,
),
 home: CalculatorHome(),
);
 }
}
```

最初に生成されるコードは  
あくまで雛形

© 2024 NTT TechnoCross Corporation



49

みんなで進める課題:プロンプト例 1

「Flutterで計算機のアプリを作りたい。サンプルコードを生成して」と言うプロンプトを実行した例です。

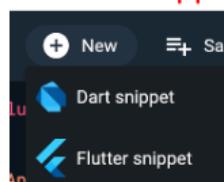
## みんなで進める課題：計算機アプリ

- 動作確認の準備（初めてでもリセットしておきましょう。）

- [DartPadを開いてください](#)
- + Newをクリック

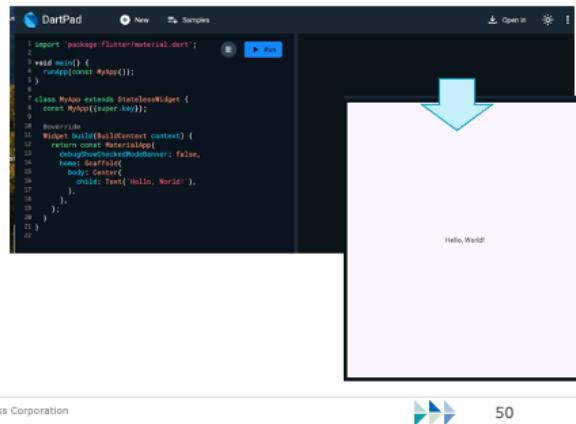
```
1 import 'package:flutter/material.dart';
2
3 void main() {
```

- FlutterSnippetを選択



- このような画面になるはず

- Runを押すとアプリ(Hello,world)の画面が出てきます。



50

### みんなで進める課題:環境の準備

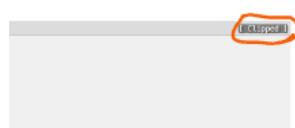
DartPadについては、以前のキャッシュが残っている可能性がありますので、一旦新しいアプリのプロジェクトとします。

## みんなで進める課題：計算機アプリ

- 動作確認

- ソースコードをクリップボードにコピーします。

算、引き算、掛け算、割り算）をサポートします。



- Runを押したら以下の通りになりましたか？



- DartPadにカ��드と貼り付けましょう

```
1 import 'package:flutter/material.dart';
2
3 void main() {
4 runApp(CalculatorApp());
5 }
6
7 class CalculatorApp extends StatelessWidget {
8 @override
9 Widget build(BuildContext context) {
10 return MaterialApp(
11 title: 'Flutter Demo',
12 theme: ThemeData(
13 colorScheme: ColorScheme.fromSwatch().copyWith(
14 primary: Colors.indigo,
15 secondary: Colors.amber,
16 error: Colors.pink,
17 background: Colors.brown,
18 surface: Colors.brown,
19 onPrimary: Colors.white,
20 onSecondary: Colors.black,
21 onError: Colors.white,
22 onBackground: Colors.white,
23 onSurface: Colors.white,
24),
25),
26);
27 }
```

- 実際に動かしてみましょう。

© 2024 NTT TechnoCross Corporation

51

### みんなで進める課題:動作確認

## 結果の揺らぎを抑止するには

- さて、実際にDartPadで動かしてみてください。講師の環境と同じ画面、動きになったでしょうか？
  - 大体近いけど同じ結果になっていない人もいると思います。
- 揺らぎを抑止する方法 = **より細かく具体的なプロンプトとする。**
- 今回の場合の例：

Flutterで計算機のアプリを作ろうと思います。

サンプルコードを生成してください。

ただし、以下の条件を前提とします。

•  
•  
•

- 余裕があれば、新しくチャットを作り直して、同じプロンプトでソースコードを生成、動作確認してみてください。

© 2024 NTT TechnoCross Corporation



52

みんなで進める課題：ゆらぎの抑止

## うまく動かない場合

- もしもうまく動かなかったら以下のアプローチを取ります。

※今回のテーマの計算機アプリなら初っ端で動かないケースは少ないとは思いますが、可能性としてゼロではありません。

- エラーメッセージをChatTXに貼り付けてその対処方法を生成させる。
- エラーメッセージを確認して、怪しい箇所を直接修正する。
- エラーメッセージに関連してインターネットで検索する。
- チャットを再度作り直して、再度実施する。

© 2024 NTT TechnoCross Corporation



53

みんなで進める課題：うまく動かない場合

## ★コラム：エラー発生時の対応

モダンなプログラミング言語では既存のAPIが非推奨になり、最新の環境で動かなくなるという経験をしたことがある人は少なからずいるのではないでしょうか。

この研修の中でも、人によっては生成されたソースコードがコンパイルエラーになったり、ひどい時には講師の環境でもその状況に陥ることがあります。（リハーサルの時にはうまくいったのに、どうして本番で！？）

これもゆらぎの一つです。先ほどのスライドでは、やり直してしまうと言うのも一つのオプションに入れていきましたが、慌てずに生成AIに聞いてみると言うのが実用的な方法だと言えます。

（※ただし、生成AIが持っている知識がいつまでのものなのか、と言う点も注意が必要です。）

たとえば、今回の例の場合、以下のようなエラーがれます。

```
71 Widget _buildButton(String buttonText) {
72 return Expanded(
73 child: OutlineButton(
74 padding: EdgeInsets.all(24.0),
75 child: Text(
76 buttonText,
77 style: TextStyle(fontSize: 20.0),
78),
79 onPressed: () => _buttonPressed(buttonText),
80),
81);
82 }
```

1 issue

⚠ The method 'OutlineButton' isn't defined for the type '\_CalculatorHomePageState'.

line 73, col 14

Try correcting the name to the name of an existing method, or defining a method named 'OutlineButton'.

エラー発生

このような時に、以下のようなプロンプトを入れてみます。

以下のようなFlutterのソースコードを実行したら、エラーが出ました。

```
```
import 'package:flutter/material.dart';

void main() {
runApp(CalculatorApp());
}

class CalculatorApp extends StatelessWidget {
// 以下、生成されたソースコードが続く
```

```

エラーメッセージ：

The method 'OutlineButton' isn't defined for the type '\_CalculatorHomePageState'.

このソースコードの修正方法を提案してください。

回答は以下の通り。

OutlineButtonはFlutterの最新バージョンでは非推奨となり、代わりにOutlinedButtonが使用されます。以下のように修正することで、エラーを解消できます。

```
```dart
import 'package:flutter/material.dart';

void main() {
  runApp(CalculatorApp());
}

//中略
Widget _buildButton(String buttonText) {
  return Expanded(
    child: OutlinedButton(
      style: OutlinedButton.styleFrom(
        padding: EdgeInsets.all(24.0),
      ),
      child: Text(
        buttonText,
        style: TextStyle(fontSize: 20.0),
      ),
      onPressed: () => _buttonPressed(buttonText),
    ),
  );
}
//以下略
```

```

この修正により、OutlineButtonをOutlinedButtonに置き換え、スタイルの設定もOutlinedButton.styleFromを使用して行っています。これでエラーが解消され、アプリが正常に動作するはずです。

このように、エラー理由と対処後のコードが生成されます。なお、該当の変更は[こちらで示されており](#)、確かにReplaceが求められているわけです。

## みんなで進める課題：計算機アプリ

- ・基本課題：以下の課題を解決しましょう

- ・入力する数値の桁数に上限を設ける。

- ・計算機の結果を目立たせる。

© 2024 NTT TechnoCross Corporation



54

### 基本課題1

実際に動かしてみると色々と課題があると気づくのではないかでしょうか。 そのような課題を修正してみましょう。

#### 基本課題：計算機の結果を目立たせる。

- ・以下のようなプロンプトをChatTXに与えます。

計算結果だけを目立つようにしたいです。  
ソースコードを生成してください。

- ・実行結果のソースコードをDartPadに貼り付けてください

- ・動作を確認してください。

© 2024 NTT TechnoCross Corporation



55

### 基本課題1: プロンプト案1

## 基本課題：入力する数値の桁数に上限を設ける。

- 以下のようなプロンプトをChatTXに与えます。

入力する数字の桁数を20桁に制限したいです。  
ソースコードを生成してください。

- 実行結果のソースコードをDartPadに貼り付けてください
- 動作を確認してください。

© 2024 NTT TechnoCross Corporation



56

### 基本課題1: プロンプト案2

## みんなで進める課題：計算機アプリ

- 追加課題

- 実際にアプリに触れてみてください。
- 触れたことで感じた課題や、やってみたいことをメモしましょう
- メモした内容を解決するプロンプトを考えてみましょう
- 生成されたソースコードの動作確認をしましょう。

- 追加課題およびプロンプトの例

- 四則演算以外にも対応する

関数電卓のように、現在の数値の平方根を取れるようにしたいです。  
ソースコードを生成してください。

© 2024 NTT TechnoCross Corporation



57

### 基本課題1: 追加課題

他にも課題を探し、改善してみてください。

## みんなで進める課題：計算機アプリ

### ・追加課題およびプロンプトの例

#### ・私が考えたこと

- ・「触ってみると、演算子を押したっけ？とか、今どの演算子で演算をしようとしているんだっけ、とか考えてしまうな。ちょっと改善してみたい」

#### ・クリックした演算子がわかるように表示する

計算に利用している演算子がわかるように画面上に表示したいです。  
ソースコードを生成してください。

### 基本課題1:追加課題とプロンプト例

### 3.3.追加演習

ここからは追加のテーマにトライしましょう。

#### ここからの進め方

- ここからは以下のテーマから選んでみてください
  - 演習テーマ1: 計算機をさらにリッチにする
  - 演習テーマ2: アナログ時計アプリを作成する
  - 演習テーマ3: TODOリスト風アプリを作成する
  - 演習テーマ4: その他自由

© 2024 NTT TechnoCross Corporation

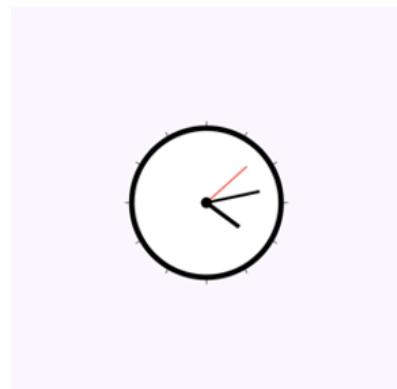


59

追加課題

#### 演習テーマ2【選択課題】：アナログ時計アプリを作成する

- 基本課題
  - この画像のようなアプリのソースコードを生成してみましょう。  
※プロンプト例は次スライド
- 条件
  - Flutterのソースコードとします
- 追加課題
  - 初級: 見た目の改善をしてみましょう
    - 時計の背景色の設定、針の色の変更
  - 中級: 実用的になるよう改善してみましょう
    - 大きさの変更、文字盤の追加(1-12)、
  - 上級: 思いつくままに改造してみましょう
    - デジタル時刻も盤面上に表示する



© 2024 NTT TechnoCross Corporation



60

追加課題2:概要

## 演習テーマ2: アナログ時計アプリを作成する プロンプト例

- 以下のようなプロンプトをChatTXに与えます。

Flutterでアナログ時計のアプリを作ろうと思います。  
サンプルコードを生成してください。

- 追加課題およびプロンプトの例

- 文字盤を追加する

時計に文字盤を表示してください。

© 2024 NTT TechnoCross Corporation



61

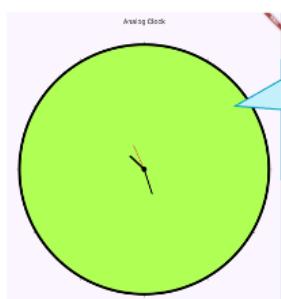
## 追加課題2: プロンプト例1

## 演習テーマ2: アナログ時計アプリを作成する プロンプト改善例

時計を緑色にしてください。  
→時計の背景色を薄い緑色にしてください

できるだけ具体的な指示を意識する

時計を大きくしてください。  
→画面の大きさに対して、時計が小さ過ぎます。画面いっぱいに時計を表示してください。



それでも  
うまくいかないことも

段階的に指示を

時計の枠に対して、針が短過ぎます。  
修正してください。

© 2024 NTT TechnoCross Corporation



62

## 追加課題2: プロンプト例2

## 演習テーマ3【選択課題】：TODOリスト風アプリを作成する

### ・基本課題

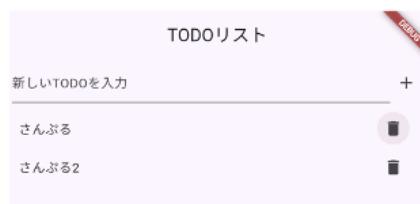
- ・この画像のようなアプリのソースコードを生成してみましょう。

※プロンプト例は次スライド



### ・条件

- ・1画面のアプリとします
- ・Flutterのソースコードとします



### ・追加課題

- ・日時を表示する
- ・グリッド状に表示する
- ・その他見た目を改善する など

© 2024 NTT TechnoCross Corporation



63

追加課題3:概要

## 演習テーマ3: TODOリスト風アプリを作成する プロンプト例

### ・以下のようなプロンプトをChatTXに与えます。

FlutterでTODOリストのアプリを作ろうと思います。  
サンプルコードを生成してください。  
以下を前提条件とします。  
・1画面で完結するアプリとする。

### ・追加課題およびプロンプトの例

- ・TODOに日時を追加で表示する  
TODOを追加した日時を表示するようにしてください。

© 2024 NTT TechnoCross Corporation



64

追加課題3:プロンプト例

## 演習テーマ4【オプション課題】：その他自由

- ・自由にアプリを考えてみてください
- ・他のテーマと同様、1)はじめに大雑把なものを考える→2)徐々に肉付けしていく という流れが良いです。
- ・もしいいものができたらこの後の時間で発表してください！

© 2024 NTT TechnoCross Corporation



65

### 自由課題

最後はもはやテーマというテーマではありませんが、このような研修を行なう場合、ぜひ普段の業務とは離れたテーマでのアプリ開発にトライしてみてください。

当社での研修の中では、「テト○ス」風のパズルゲームやアクションゲームを作り上げた人や、見た目の可愛さにとことんこだわった時計作りなど、(少なくとも著者には思いつかないような)柔軟な発想が見られました。

ぜひ、お互いの「作品」を見せ合う時間をとっていただけと良いと思います。

## 3.4.振り返り



カバー

冒頭で紹介したとおり、研修の最後に、講師と受講者全員で振り返りを行いました。研修では、アプリ開発の経験を十分に有している人からプログラミング経験が全くない人まで参加しています(当初の狙い通りです)。研修の最後に実施している振り返りでは、事前に受講者が有しているスキルのレベルに関わらず、受講者それぞれで様々な気づきを得ていることを確認できました。

## 関連リンク

- Flutter公式サイト(社外) <https://docs.flutter.dev/>
- DartPad公式サイト(社外) <https://dartpad.dev/>

© 2024 NTT TechnoCross Corporation



67

## 関連リンク

- Flutter公式サイト(<https://docs.flutter.dev>)
- DartPad公式サイト(<https://dartpad.dev>)

## 振り返り

- ・本研修での気づきを教えてください。[全員/お一人30秒目安]
- ・講師からも気づきや補足などをお願いします。[全員]

振り返り

# おわりに

「生成AIではじめるプログラミング体験ガイド」を最後までお読みいただき、本当にありがとうございます。

生成AIを利用したジョブシャドウイング、インターンシップは2023年度から、社員研修は今年度より取り組みを開始しました。

活動過程で

- 生成AIの勘所をよく理解できた
- 生成AIを利用してプログラミングをはじめとした作業を効率化できた
- 生成AIを利用することで難しい・できないと思っていた作業ができるようになった

といった嬉しい感想を数多く耳にしました。

執筆メンバー一同、まだまだ改善の余地があることは承知しています。しかし、本書を様々な場所で活用頂くことで「ITに関わる人が増える」、「IT業界に興味を持ち、将来的にエンジニアとして活躍する人が増える」なら十分価値があると考え、技術書典17でのリリースを決断しました。

リリース直後ですが、生成AIの発展と合わせて今後も内容のブラッシュアップを行う考えです。引き続き注目頂けると嬉しいです。また、お気づきの点が御座いましたら『お問合せ先』まで一報を頂けると非常に助かります。

最後に、NTTテクノクロス株式会社では、本書以外にも技術書典で多くの書籍をリリースしています。ダウンロードよろしくお願いします。

---

本書は[技術書典17\(2024/11 オンライン・オフライン同時開催\)](#)に向けて執筆しました。 · · ·

是非、当社の今後にご期待いただければと思います。

2024年11月 NTTテクノクロス株式会社 執筆メンバー一同

## 表紙デザインについて

★表紙デザインを検討した際の背景など。

# 著者紹介

## 神長 貴博(X:@onyankopon\_0311)

会社でモバイル向けアプリケーションの開発・技術支援、学生向けプログラミング教育、講演活動を行っています。趣味はランニング、フェス・ライブ参戦、食べ歩き。本書では「はじめに」、「対象読者」、「おわりに」、スケジュール等全体のディレクションを担当しました。

- Qiita(<https://qiita.com/subutakahiro>)
- エンジニアが本気で学生向けプログラミング教育をやってみた([https://www.ntt-tx.co.jp/column/cross\\_field.html](https://www.ntt-tx.co.jp/column/cross_field.html))

## 神原 健一(X:@korodroid)

FlutterやWear OS、生成AIなど新しい技術大好きエンジニア。会社では技術支援・講師、技術/英語コミュニティ運営、社外では講演や執筆などエバンジェリスト活動を実施しています。マイブームは、英語とランニングとアプリ開発。本書では「本書の概要」、「研修のアジェンダ/目的」、「Flutterおよび開発環境の概要」「振り返り」を執筆し、PM/統括を担当しました。

- Qiita(<https://qiita.com/korodroid>)
- 神原健一の技術で広がる世界([https://www.ntt-tx.co.jp/column/kambara\\_blog.html](https://www.ntt-tx.co.jp/column/kambara_blog.html))

## 戸部 雄太郎(X:@etctaro)

Android、Kotlin、自動テストなどの技術をフォローしたり、社内の勉強会を開催したり、社内のちょっとした話題から技術的なノウハウまで様々な情報を社内展開しています。

趣味はボクシング観戦とゲーム。モンハンはハンマー使い。あと家に生成AIのローカル環境を構築したりしました。本書では「生成AIを活用した実践演習」の執筆を担当しました。

- Qiita(<https://qiita.com/etctaro>)

## 山本 昂輝

会社では主に業務で活用できる生成AI環境の開発を行っています。趣味はお笑いを見ることと散歩(山手線を1周徒歩で歩くなど)。技術書の執筆は今回が初めてですが、過去に趣味で夢日記を自費出版したことがあります。本書では「生成AIの概要」にて、生成AIの説明やプロンプトのコツを担当しました。

- Qiita([https://qiita.com/iron\\_yamamoto](https://qiita.com/iron_yamamoto))

★表紙デザイン：足達 俊雅

# 生成AIではじめるプログラミング体験ガイド

2024年11月02日 初版発行

- 著者 NTTテクノクロス株式会社
- 編集 NTTテクノクロス株式会社
- 発行 NTTテクノクロス株式会社

(c) 2024 NTT TechnoCross Corporation