

T3 Stack を触ってみた

個人開発をする際の技術選定では何を重視しますか？


- 学習コストが低いこと
- 開発速度が速いこと
- トレンドに乗ること
- ドキュメントが豊富であること

自分の場合

- 学習コストが低いこと
- **開発速度が速いこと**
 - 短期間で作らないとモチベ下がる 😓
- トレンドに乗ること
- ドキュメントが豊富であること

T3 Stack

T3 Stack とは

The “*T3 Stack*” is a web development stack made by Theo  focused on simplicity, modularity, and full-stack typesafety.

<https://create.t3.gg/en/introduction>

T3 Stack とは

- simplicity(シンプルさ)
 - 開発プロセスを簡潔に保つこと
 - Next.js と TypeScript をコアとし、必要に応じて他の技術を追加できる
- modularity(モジュール性)
 - モジュールとしてそれぞれの技術が独立している
- full-stack typesafety(フルスタックの型安全)
 - フロントエンドからバックエンドまで、全ての層で型安全性を確保することを重視している
 - 開発中のエラーを減少させて、コードの信頼性と保守性の向上ができる

T3 Stack って具体的に何？

公式では以下の構成

- **Next.js** (フロントエンドフレームワーク)
- **TypeScript** (型安全な言語)
- **tRPC** (型安全な API 通信)
- **Prisma, Drizzle** (型安全な ORM)
- **Tailwind CSS**: ユーティリティファーストの CSS フレームワーク
- **NextAuth.js**: (認証ライブラリ)

tRPC について

- RPC とは、「Remote Procedure Call」の略です。あるコンピューター（サーバー）上の関数を、別のコンピューター（クライアント）から呼び出す方法

```
getAll: publicProcedure.query(async ({ ctx }) => {  
  const posts = await ctx.db.post.findMany({  
    orderBy: { createdAt: "desc" },  
    include: {  
      createdBy: true,  
      likes: true,  
      _count: {  
        select: { likes: true },  
      },  
    },  
  });  
  
  return posts;  
}),
```


tRPC について

何を解決するのか

- 型の不一致
 - REST API の場合、バックエンドで定義した型と、フロントで送る型が違う！
みたいなことがある。
 - tRPC の場合
 - フロントエンド、バックエンドで同じ型定義を使用することができる

実際に何か作ってみる

以下コマンドを実行

```
npm create t3-app@latest
```

```
> create-t3-app
```

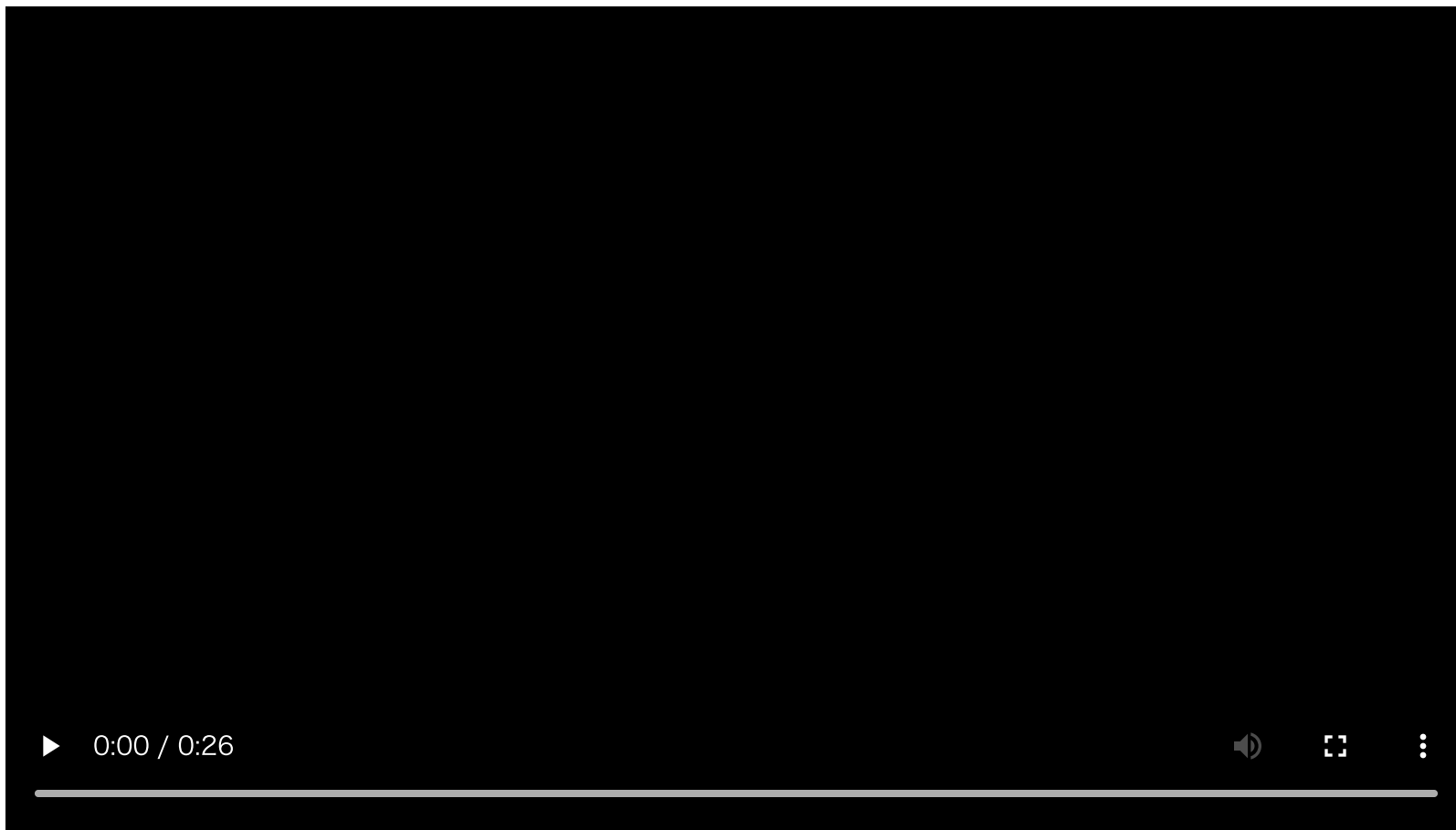
```
CREATE T3 APP
```

```
|
| ◇ What will your project be called?
|   t3-todo
|
| ◇ Will you be using TypeScript or JavaScript?
|   TypeScript
|
| ◇ Will you be using Tailwind CSS for styling?
|   Yes
|
| ◇ Would you like to use tRPC?
|   Yes
|
| ◇ What authentication provider would you like to use?
|   NextAuth.js
|
| ◇ What database ORM would you like to use?
|   Prisma
|
| ◇ Would you like to use Next.js App Router?
|   Yes
|
| ◇ What database provider would you like to use?
|   SQLite (LibSQL)
|
| ◇ Should we initialize a Git repository and stage the changes?
|   Yes
|
| ◇ Should we run 'npm install' for you?
|   Yes
|
| ◇ What import alias would you like to use?
|   ~/
```

投稿アプリを作ってみる

- メッセージの投稿ができる
- いいねができる
- ユーザー認証ができる

できたもの



かかった時間

t3 Stack を触ってみた

1:10:57

実際にやってみて感じたことなど。

メリット

- 環境構築がめちゃくちゃ速い
 - DB の接続や、認証関連の設定がわかりやすい
- 型の補完が抜群
 - Next.js, **tRPC**, Prisma など、すべて TypeScript ベースなので、型の恩恵を最大限に受けられる。

実際やってみて感じたことなど。

デメリットや許容する必要がある点

- 自分が触っていない技術は**キャッチアップに時間がかかる**(どの技術もそうですが...)
 - Next.js (フロントエンドフレームワーク)
 - TypeScript (型安全な言語)
 - **tRPC** (型安全な API 通信)
 - Prisma, Drizzle **** (型安全な ORM)
 - Tailwind CSS: ユーティリティファーストの CSS フレームワーク
 - NextAuth.js: (認証ライブラリ)
- **外部向けの API や マイクロサービス構成** には適さない。

T3 スタックを触ってみた感想

✅ メリット

- 環境構築が爆速 → DB 接続や認証設定がスムーズ
- 型の補完が強力 → フロントもバックも TypeScript で統一
- 開発速度が上がる → 短期間でアプリが作れる

❌ デメリット & 許容すべき点

- 新しい技術をキャッチアップするコスト (tRPC, Prisma, NextAuth など)
- 外部向け API やマイクロサービスには不向き (tRPC の特性上)
- ◆ 個人開発にはかなり使いやすいが、適用範囲は考慮する必要あり！

試してみる価値アリ！ 🚀