

# デザインシステムとは何を解決する仕組みなのか

takeda.k

# はじめに

最近デザインシステムやデザインエンジニアという言葉をよく目にするようになったけど。。。

そもそもデザインシステムとは何なのか？

どんな問題を解決するものなのか？

# デザインシステムとは

- ・ UIを構築する上で決められたデザインのルールや規則

例: 使用するfont, space, colorなどの値

- ・ 再利用可能なコンポーネントや定数など

↑これらの集合体を指してデザインシステムと呼ぶ

(ここは現場によって多少解釈が異なりそうなので大体こういうものって分かっているだけでいい)

# デザインシステムの目的

一貫性のあるデザインを実現し、プロダクトの体験を向上させる

判断基準を統一し、意思決定のスピードと質を高める

再利用可能な規則やコンポーネントを整備し、開発効率を向上させる

**もう少し具体的に考えてみる**

# 一貫性のあるデザインを実現し、プロダクトの体験を向上させる

→ ここでいう「一貫性」とはどのように決まるものか

→ ユーザーの認知負荷を軽くする（迷わずに利用させる）ために検証されたパターンをもとに決められるもの

→ ユーザーに対する検証※によって決められた一貫性の実現でプロダクトの体験の向上を目指す

※ A/Bテストやユーザビリティテストなどを想定しています

# 判断基準を統一し、意思決定のスピードと質を高める

- プロダクトに関わる人が増える・初期から関わっていたメンバーが抜けるなどのケースで判断基準（font, space, colorなど）がない場合、意思決定のコストが上がる
- メンバーがそれぞれ個人の解釈で意思決定をした結果、一貫性のあるデザインの維持が難しくなる
- 判断基準があることで開発における意思決定のコストを下げられる



# 再利用可能な規則やコンポーネントを整備し、開発効率を向上させる

- 判断基準があるだけでは開発者同士が同じUIを作ってしまう（車輪の再発明が起きる）可能性がある
- 同じUIでも作る人が違えば差が出てしまい、一貫性を損ねる
- 開発においては共通部分のコンポーネント化、デザイナーとエンジニアのコミュニケーションにおいては「Primary Button」といった共通言語などの整備をしておくことで一貫性を保ちつつ開発効率の向上を目指す

**実際にデザインシステムをやってみる**

# **Style Dictionaryで デザイントークンを実装する**

# デザイントークンとは何か

デザイントークンは、スペーシング、カラー、タイポグラフィ、オブジェクトスタイル、アニメーションなど、デザインシステムを構築し維持するために必要なすべての値をデータとして表現したものです。カラーはRGB値、不透明度は数値、アニメーションはベジェ座標など、デザインで定義されたあらゆるものを表現することができます。ハードコーディングされた値の代わりに使用することで、すべての製品体験において柔軟性と統一性を確保します。

Adobe Spectrumから引用(<https://spectrum.adobe.com/page/design-tokens/#What-are-design-tokens?>)

スタイルを直書きするのではなく変数や定数にして使い回すものという認識でok

デザインシステムの目的のうち「判断基準の統一」と「再利用可能な規則」を達成できる

# Style Dictionaryとは何か

Amazonが開発しているライブラリ

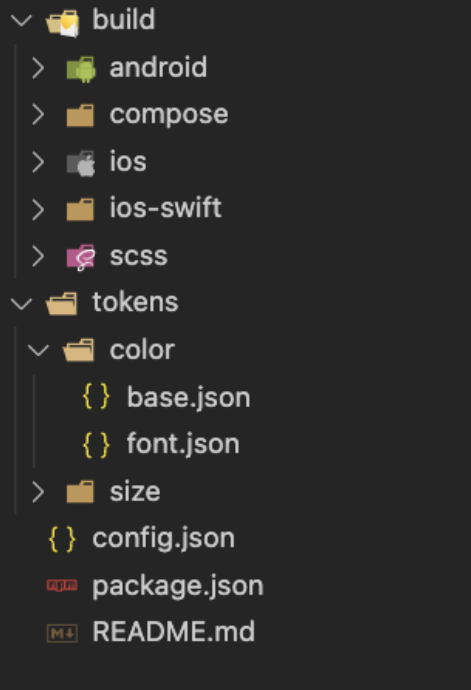
JSONで定義したスタイルから（webに限らず）様々なプラットフォームに対応したデザイントークンを出力してくれるライブラリ

# 初期化コマンド

```
$ yarn init -y  
$ npx style-dictionary init basic
```

※ 公式のbasicテンプレートを使用

# 初期化後のディレクトリ構造



```

└─ build
   ├── android
   ├── compose
   ├── ios
   ├── ios-swift
   └── scss
└─ tokens
   ├── color
   │   ├── {} base.json
   │   └── {} font.json
   ├── size
   │   └── {} config.json
   ├── package.json
   └── README.md

```

build → デザイントークンの出力先

tokens → スタイルが定義されたJSON群

config.json → style-dictionaryの設定ファイル（commonJS形式でも書ける）

**tokensの中身を覗いてみる👁👁**



colorディレクトリの中では主に色に関するスタイルを定義している

```
tokens > color > {} base.json > ...
1  {
2    "color": {
3      "base": {
4        "gray": {
5          "light": { "value": "#CCCCCC" },
6          "medium": { "value": "#999999" },
7          "dark": { "value": "#111111" }
8        },
9        "red": { "value": "#FF0000" },
10       "green": { "value": "#00FF00" }
11      }
12    }
13  }
```

```
tokens > color > {} font.json > ...
1  {
2    "color": {
3      "font": {
4        "base": { "value": "{color.base.red.value}" },
5        "secondary": { "value": "{color.base.green.value}" },
6        "tertiary": { "value": "{color.base.gray.light.value}" }
7      }
8    }
9  }
```

base.jsonでは汎用的な値を定義していることから**Primitive Token**、

font.jsonでは意味的に定義していることから**Semantic Token**と呼ばれる。










Semantic Tokenでは値をハードコーディングしないようにするため、必ずPrimitive Tokenから参照する。

ビルドしてみる🔧

# ビルドコマンド

```
$ npx style-dictionary build
```

## 出力例（SCSS）

```
build > scss >  _variables.scss > ...  
1  
2 // Do not edit directly  
3 // Generated on Sun, 31 Jul 2022 17:45:32 GMT  
4  
5 $color-base-gray-light:  #cccccc;  
6 $color-base-gray-medium:  #999999;  
7 $color-base-gray-dark:  #111111;  
8 $color-base-red:  #ff0000;  
9 $color-base-green:  #00ff00;  
10 $color-font-base:  #ff0000;  
11 $color-font-secondary:  #00ff00;  
12 $color-font-tertiary:  #cccccc;
```

※ CSS、JSもサポートしている

# まとめ

- ・デザインシステムとはデザインのルールや規則、再利用可能なコンポーネントの集合体のこと。
- ・デザインシステムによって、  
判断基準を統一し、意思決定のスピードと質を高め  
一貫性のあるデザインを実現し  
開発効率とプロダクトの体験を向上させることができる。
- ・デザイントークンはデザインの規則を決められるのと同時に再利用可能にできる仕組み

# もっと知りたい人へのおすすめ

- ・ [WEB + DB PRESS VOL.129 「小さく始めるデザインシステム」](#)
- ・ [デザインシステムの目的を考える](#)
- ・ [グッドパッチエンジニアが選ぶ、推しデザインシステム10選](#)