


# Lightning Web Componets について

 Slides are here

スライド:<https://powerninja.github.io/SSCLWC/ja/index.html>

リポジトリ:<https://github.com/powerninja/SSCLWC>

# はじめに

- どういう方に向けて話すかを記載しても良い
  - TODO:画像をもう少し増やしても良いかも
  - TODO:勉強会環境にdemoを用意する
  - Summer '19 に登場し、約4年立ちました。
- 
- フロントの開発では、Aura や visualforce よりも選択される機会が増えてきていると思うので、少しずつ始めてみましょう! 😊👍

# トピックス

1. 画面開発の歴史
2. Lightning Web Components と Aura の違いは？
3. なぜ Lightning Web Components を選択するのか
4. LWCの始め方 知らない気がする
5. 使用した案件の紹介
6. おまけ

# 画面開発の歴史

- Visualforce
  - Summer '08 くらい？
- Lightning Aura Components
  - Auraと記載される
  -
- Lightning Web Components
  - LWCと記載される
  - Summer '19

# Lightning Web Components と Aura の違いは？

- 共通点
  - salesforce上での見た目はほぼ同じ([LDS](#)を標準で使用)
  - classic未対応
  - JavaScriptを用いた開発
  - 外部 JavaScript ライブラリの[使用可能](#)(静的リソースで読み込ませる,npmは不可)
  - [Tailwind CSS](#)のようにクラス名でスタイルを与えることができる
    - そのため、CSSファイルはLDSでよければ不要

# Lightning Web Components と Aura の違いは？

- 相違点
  - Auraは開発者コンソールで作成可能だが、LWCはvsCodeが必要(chromeの拡張機能で開発は可能)
  - LWCはユニットテスト [Jest](#)に対応している
  - LWCで対応していない機能がまだある、その場合はAuraを作成する必要あり
  - ただ、Auraは開発がアーカイブ化されている(サポートはしている)

[Aura開発リポジトリ](#)

[LWC開発リポジトリ](#)

# Lightning Web Components と Aura の違いは？

- Visualforceとの比較
  - 共通点
    - あまりない
  - 相違点
    - LDSを使用する場合でも、cssで指定が必要
    - コントローラーがLWCはJavaScript(ブラウザ動作),VFはApex(サーバ動作)  
そのため、LWCのパフォーマンスが良い

# なぜ Lightning Web Components を選択するのか

## 新しいから



# なぜ Lightning Web Components を選択するのか

- (Auraと比べると)開発コミュニティが活発なため、新機能などに期待できる
- (Auraと比べると)標準的なJavaScriptを使用することができるため、JavaScriptの開発経験がある方は開発しやすい
  - また、(Auraに比べると)直感的なため学習コストが低い
- (Auraと比べると)ファイルの数が少なく、初期段階の理解が早い(個人差あり)
- (Auraと比べると)パフォーマンスが良い

# なぜ Lightning Web Components を選択するのか

- LWCで作成されるファイル数

プロジェクト名(任意で設定可能)

| -HTML

| -JavaScript

| -xml

| -css(任意)

# なぜ Lightning Web Components を選択するのか

- Auraで作成されるファイル数
  - 全部が必要なわけではないが。。。

プロジェクト名(任意で設定可能)

- | -auradoc
- | -cmp(HTML)
- | -cmp-meta.xml
- | -css
- | -design
- | -svg
- | -Controller.js
- | -Helper.js
- | -Render.js

## なぜ Lightning Web Components を選択するのか

```
<template>
  {hello}
</template>
```

```
import { LightningElement } from 'lwc';

export default class Test extends LightningElement {
  hello = 'Hello,World!'
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
  <apiVersion>56.0</apiVersion>
  <isExposed>false</isExposed>
</LightningComponentBundle>
```

# LWCの始め方

1. salesforce CLIのインストール
2. vsCodeのインストール
3. vsCode内でSalesforce Extension Packをインストールする
4. ctrl + shift + P を押下し、新規プロジェクトを作成
5. ctrl + shift + P を押下し、新規LWCを

# 使用した案件の紹介

- ルックアップ検索条件の絞りが画面フローでは対応できずLWCにて作成
- 飲食店の地図情報を詳細ページに表示するためLWCを使用
- experience cloud上で、webページ構築

# おまけ

- こちらのスライドはgithub actionを使用して、vscodeでmarkdownを記載するだけでスライドを作成できるように。

または

- LWCのOSS版紹介

# Markdown 例

```
---  
marp: true  
paginate: true  
---  
  
<!-- _paginate: false -->  
  
# About [Marp CLI Action](https://github.com/KoharaKazuya/marp-cli-action) <!-- fit -->  
  
---  
  
## [Marp CLI Action](https://github.com/KoharaKazuya/marp-cli-action) is...
```

[このスライドのソース](#) を参照してください。



# "Marp CLI Action" は何を提供するの？

GitHub Action として [Marp CLI](#) を実行します。

リポジトリ内の Markdown ファイルから HTML、PDF、PowerPoint、画像を生成の自動化を素早く簡単にセットアップできます。

# 使い方

1. ワークフローに以下を追加します

```
- uses: KoharaKazuya/marp-cli-action@v1
```

または

1. [このテンプレート](#) を使用し、リポジトリを作成します
2. Marp の Markdown を書きます
3. コミットし、プッシュします

*Marp CLI Action は Markdown からファイルを生成するのみです。  
アップロード、リリース、公開などをする場合は他のアクションを*

# オプション

`action.yml` の `inputs` セクションを参照してください。

Marp CLI をカスタマイズするには `marp.config.json`、`marp.config.cjs`、`.marprc` [といった Marp CLI の設定ファイル](#) を使用してください。

設定ファイルはリポジトリルートに置くか、`config-file` input を指定します。

日本語ユーザーへ; Marp CLI オプションとして `lang: ja-JP` を指定してください。そうした場合、Marp CLI Action は CJK フォント問題を修正します。

楽しくスライドを書こう！ 🙌