

サードパーティウィジェットの アクセシビリティ

各社の事例に学ぶ！ アクセシビリティを向上させる開発プロセス

yuhei

- ・株式会社プレイド
- ・デザインエンジニア
- ・Developer Experienceチーム

 yuheiy.com

固定

全部入り HTML 太郎 @_yuheiy · 5月 8日 ...

アクセシビリティについて一から説明することは容易ではありません。概念的な理解から、取り組むべき理由、ガイドライン、実践方法、その学び方など、一言では言い表せません。そこで今回は、主要なテーマについて概説しつつ、各論の参考資料も併せて紹介する記事を書きました

PLAID Engineer Blog

#アクセシビリティ

アクセシビリティ学習の手引きとしての入門講座

 Yuhei Yasuda
Design Engineer

アクセシビリティ学習の手引きとしての入門講座

PLAID

tech.plaid.co.jp から

1 91 239 2.1万

サードパーティウィジェットとは

- ・ サードパーティ（第三者）は、サービスの提供者および利用者以外の立場を指す
- ・ サードパーティのベンダーから提供される、ウェブページに埋め込むことができるコンテンツのこと

サードパーティウィジェットの例

- ・ YouTubeなどの動画プレイヤー
- ・ Googleマップなどの地図
- ・ SNSの共有ボタン
- ・ 広告のiframe
- ・ クッキー使用の同意を求めるダイアログ
- ・ 問い合わせフォーム
- ・ CAPTCHA

よくあるアクセシビリティ方針:

「サードパーティが提供する
コンテンツは当社の
管理管轄外のため対象外とします。」

**サードパーティウェイジェットの
アクセシビリティの確保は
ベンダーの責務**

よくある問い合わせ:

「KARTEはアクセシビリティ対応
できますか?」

サードパーティウェイジェットを 提供する製品

- ・ KARTE Web / KARTE for App
- ・ KARTE Message
- ・ KARTE Blocks
- ・ RightSupport by KARTE (グループ会社の製品)

サードパーティウィジェットを 提供する製品

- ・ KARTE Web / KARTE for App  接客アクション
- ・ KARTE Message
- ・ KARTE Blocks
- ・ RightSupport by KARTE (グループ会社の製品)

ユーザーの声を 集める

ユーザーに寄り添うためには行動情報からユーザーの思いを想像することも大事ですが、手段はそれだけではありません。直接ユーザーに聞くことでサービスやコンテンツ、事業に対する様々なヒントを得ることが可能です。

ユーザーの声をあつめるために大事なことはタイミングと聞き方。KARTEではユーザーの声を集めるための多くのテンプレートをご用意しています。

「よくある質問&回答」を適正なタイミングで表示し問い合わせ削減を



人気商品の投票アンケートとその集計結果を回答後に表示



よく来訪・利用してくれるファンにサイトへの要望をヒーリング



未購入者にアンケートで定性情報を取得しサービス改善に活用



好みのブランドをアンケート収集し属性の拡充を



CVを促進する

事業を成長させるには、訪問者数やコンバージョン数を増やす、購入者の単価を向上させるなど、様々な改善施策が必要です。

「訪問者の母数は一定あるのに売上が伸びない」と悩んでいる方向け、コンバージョンを促進するためのテンプレートをご用意しています。

商品詳細ページで関連商品の動画を埋め込み表示



商品ページで「閲覧人数」を表示し購入のひと押しを



カード落ちユーザー対策、再来訪時にカード商品をリマインド



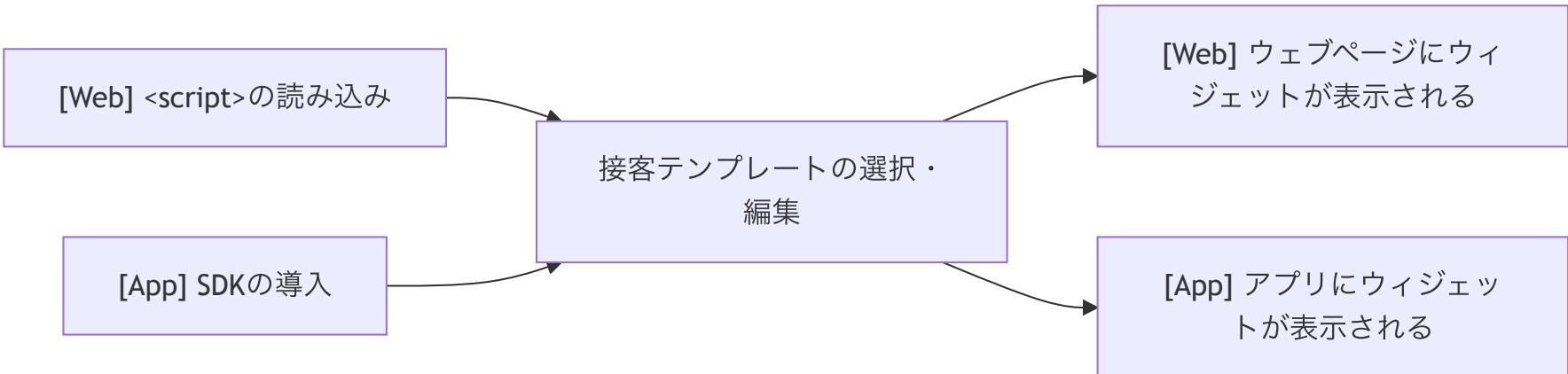
初回及び2回目購入促進のクーポンを配布し継続利用を促す



サイトに長期滞在しているユーザーへ注文ボタンを表示



接客配信の流れ



テンプレート名で検索



カスタムテンプレート 2

新着テンプレート 10

やりたいことから探す

種類から探す ^

ポップアップ 100

チャットアイコン 3

チャットメッセージ送信

チャット後アンケート 1

Talk リッチメッセージ送信

Talk マルチアクション

メール 0

SMS 0

LINE 0

アンケート 16

スクリプト 30

その他 1

アーカイブテンプレート 87

非公開テンプレート Admin 225

【モバイル向け】#6 見出し+詳細テキスト+画像+ボタン2...

アプリ (iOS・Android)
ウェブ (モバイル)

【モバイル向け】#5 見出し+詳細テキスト+画像+ボタン2...

アプリ (iOS・Android)
ウェブ (モバイル)

【モバイル向け】#3 画像+見出し+詳細テキスト+ボタン2...

アプリ (iOS・Android)
ウェブ (モバイル)

【モバイル向け】#2 画像+見出し+詳細テキスト+ボタン2...

アプリ (iOS・Android)
ウェブ (モバイル)

【モバイル向け】#1 画像+見出し+詳細テキスト+ボタン1...

アプリ (iOS・Android)
ウェブ (モバイル)

【モバイル向け】#4 見出し+詳細テキスト+画像+ボタン1...

アプリ (iOS・Android)
ウェブ (モバイル)

【モバイル向け】画像+蒂ボタン

アプリ (iOS・Android)
ウェブ (モバイル)

【モバイル向け】通知 01

アプリ (iOS・Android)
ウェブ (モバイル)

【モバイル向け】通知 02

アプリ (iOS・Android)
ウェブ (モバイル)

Card 07

アプリ (iOS・Android)
ウェブ (PC・モバイル)

Image Notification 07

アプリ (iOS・Android)
ウェブ (PC・モバイル)

Card 06

アプリ (iOS・Android)
ウェブ (PC・モバイル)

Image Notification 06

アプリ (iOS・Android)
ウェブ (PC・モバイル)

Notification 06

アプリ (iOS・Android)
ウェブ (PC・モバイル)

Card 05

アプリ (iOS・Android)
ウェブ (PC・モバイル)

Image Notification 05

アプリ (iOS・Android)
ウェブ (PC・モバイル)

Notification 05

アプリ (iOS・Android)
ウェブ (PC・モバイル)

Sidebar 05

アプリ (iOS・Android)
ウェブ (PC)

Card 04

アプリ (iOS・Android)
ウェブ (PC・モバイル)

Image Notification 04

アプリ (iOS・Android)
ウェブ (PC・モバイル)

Notification 04

アプリ (iOS・Android)
ウェブ (PC・モバイル)

List 04



リスト

Image List Vertical 04



リスト

Movie 04



動画

Bar 04



グラフ

Image List Horizontal 04



リスト

Card 03



カード

Image Notification 03



通知

X

ベースック </> カスタマイズ

保存

▼



https://

LIVING Powered by KARTE

初めの方へ

基本サービスやご利用方法をご紹介します。
お得な特典や役立つ情報が満載です。

確認する

テーブル・デスク シェルフ



テンプレート設定

テンプレート名(27文字以下推奨)

Card 07

表示タイプ

ウィジェット

説明

ボタンのレイアウトは、1つ・横
並び・縦並びに変更できます。画
面構成を切り取る事でさわぎ

 ステート変更イベントを送信する

画像



見出し

初めの方へ

詳細テキスト

基本サービスやご利用方法をご紹
介します。
お得な特典や役立つ情報が満載です。

スタイル

最小化表示

HTML CSS Script 変数 プレビュー

エディタ設定 ES5

エディタ設定 ES5



https://

LIVING Powered by KARTE

コンテンツ *

初めての方へ

基本サービスやご利用方法をご紹介します。
お得な特典や役立つ情報が満載です。

確認する

テーブル・デスク

シェルフ

HTML

CSS

Script

変数

プレビュー

```
1 <div class="wrapper">
2   <section krt-if="state == 1">
3     <div class="card _edged border-4">
4       <div class="card-body">
5
6         <!-- 閉じるボタン -->
7         <button type="button" class="btn-close karte-close">
8           <i class="icon-close" aria-label="閉じる"></i>
9         </button>
10
11        <!-- 画像 -->
12        <figure class="card-image" style="width:128px">
13          
14        </figure>
15
16        <!-- 見出し -->
17        <h1 class="card-heading text-center">
18          #{heading}
19        </h1>
20
21        <!-- 詳細テキスト -->
22        <p class="card-detail">
23          #{detail}
24        </p>
25
26        <!-- ボタン -->
27        <div class="card-button">
28          <button-block></button-block>
29        </div>
30      </div>
31    </div>
32  </div>
33</div>
34
```

考えられる対応策

- ・SDKの改修
- ・既存テンプレートの改修
- ・アクセシブルな新規テンプレートを開発する体制の確立
- ・ユーザーへのガイダンスの提供

SDKの改修

- ・接客アクションはWebViewを介して表示される
- ・WebViewの内側の状態に応じて、外側にも状態を反映させる対応が必要
- ・モーダルダイアログが開いたらその外側もモーダルに

Native

既存テンプレートの改修

- ・ テンプレートは独立したソースコードとして管理されており、一つ一つ書き換えていく対応が必要
 - ・ 一箇所直したら一気に直るような仕組みではない
- ・ 簡単なものから複雑なものまで、数は100以上ある
 - ・ カルーセル、アンケートフォーム、チャット等々

アクセシブルな新規テンプレートを開発する体制の確立

- ・さまざま人の手によって、不定期で新たなテンプレートが作成され続けている
- ・既存テンプレートだけを改善しても、新たなテンプレートが非アクセシブルになる可能性がある
- ・新規テンプレートを最初からアクセシブルにするための社内教育やレビューア体制が必要

ユーザーへのガイダンスの提供

- ・テンプレートの出来にかかわらず、ユーザーの入力内容によって非アクセシブルになる可能性がある
- ・アクセシブルなコンテンツを作成するための支援が必要
 - ・テンプレート作成画面に注釈を入れる
 - ・ドキュメントを作成する

どこから始めるか

接客テンプレートのガイドライン作成

- ・既存テンプレートの改善および新規作成の方針作り
- ・構成要素をパターン化した実装例集を作る
 - ・すべてのテンプレートに目を通して調査

1. ポップアップ

ほとんどすべての接客シナリオに実装する必要がある。ポップアップの扱いについて説明します。

接客シナリオの内容にかかわらず、組み込むべきソースコードは同じです。したがって、基本的にコピー＆ペーストだけでは出来ます。

HTML

次のHTMLをベースにしてテンプレートを実装します。

```
<karte-widget-dataset>
  krt="r#state > #"
  key="karteWidgetPosition"
  value="position"
</karte-widget-dataset>
```

```
<style krt="r#state > #"
  id="root"
  --karte-widget-position: {{ position }};
  --karte-widget-width: {{ size.width }};
  --karte-widget-height: {{ size.height }};
  {{ customStyles.top }} -- "karte-widget-top": {{customStyles.top}};
  {{ customStyles.right }} -- "karte-widget-right": {{customStyles.right}};
  {{ customStyles.bottom }} -- "karte-widget-bottom": {{customStyles.bottom}};
  {{ customStyles.left }} -- "karte-widget-left": {{customStyles.left}};
}</style>
```

```
<karte-widget-dataset>
  krt="r#state > #"
  key="karteWidgetOutside"
  value="outside"
</karte-widget-dataset>
```

```
<karte-widget-escape-handler>
  krt="r#state > # && !modal"
  </karte-widget-escape-handler>
```

```
<section krt="r#state === 1" role="dialog" aria-modal="!modal,toString()">
  ...
</section>
```

```
<section krt="r#state === 2" role="dialog" aria-modal="!modal,toString()">
  ...
</section>
```

ステートの数だけ `<section>` 要素を作成して、それに対応するコンテンツを挿入します。

```
<section krt="r#state === 1" role="dialog" aria-modal="!modal,toString()">
  ...
</section>
```

```
<section krt="r#state === 2" role="dialog" aria-modal="!modal,toString()">
  ...
</section>
```

ステートの数が1つだけでも、同時にコンテナ要素を作成します。

```
<section krt="r#state === 1" role="dialog" aria-modal="!modal,toString()">
  ...
</section>
```

モードダイアログの中で、最初にフォーカスする要素には `data-karte-widget-autofocus` 属性を付与します。ボタンとして閉じるボタンに適用してください。

```
<section krt="r#state === 1" role="dialog" aria-modal="!modal,toString()">
  <button type="button" class="karte-close" data-karte-widget-autofocus>
    ...
  </button>
</section>
```

JavaScript

ポップアップの内容にかかわらず、次のソースコードをそのまま利用するだけで対応が完了するようになっています。

```
class DocumentDatasetElement extends HTMLElement {
  static tagname = "karte-widget-document-dataset";
  static observerAttributes = ["key", "value"];

  attributeChangedCallback(name, oldvalue, newValue) {
    if (this.hasAttribute("key") && this.getAttribute("value")) {
      document.addEventListener(`set${name}State`, () => this.setAttribute("key", newValue));
    }
  }

  disconnectedCallback() {
    delete document.documentElement.dataset[this.getAttribute("key")];
  }
}

if (!customElements.get("documentdatasetelement")) {
  customElements.define("documentdatasetelement", DocumentDatasetElement);
}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
class EscapedContentElement extends HTMLElement {
  static tagname = "karte-widget-escaped-content";
  connectedCallback() {
    document.addEventListener("keyup", this.handleEscape);
  }
}
```

2. コントラスト要件

文字色と背景色など、2つの色の輝度の比のことを指すコントラスト比と呼びます。すべてのコンテンツは、規定のコントラスト比を満たすように実装される必要があります。

コントラスト比を計算するには、次のようなツールを用いる方法があります。

- [WebAim Contrast Checker](#) (Webアプリケーション)
- [Contrast](#) (macOS用)

文字と背景のコントラスト

文字色と背景色は4.5以上とのコントラスト比を確保します。

#FFFFFF : #009688

X 3.67:1



文字はテキストだけにかぎらず、画像化された文字にも同様の基準が適用されます。

非テキストのコントラスト

ボタンやアイコンなど、コンテンツの意味を理解するのに重要な要素とその周辺の色には、3:1以上のコントラスト比を確保します。

3.32:1

#8D8DBD : #FFFFFF



背景色や枠線などの視覚的な境界が存在しないボタンなどについては、ボタンの中のアイコンや背景色にのみ基準が適用さ

5. リンク

リンクの実装方法について説明します。

リンクの目的を理解できるようにする

リンク本体のコンテンツやその前の文脈から、リンク先やリンクの目的が理解できるようにします。

この良い例には、「レベルAAA相当」と「レベルAA相当」のものがあります。可能な限り「レベルAAA相当」の例を採用することを推奨します。「レベルAA相当」の例でも許容されます。

悪い例 「詳しいはこちら」など不正確なリンクテキストを使用している

```
<a href="https://example.com/~詳細を見る">~詳細を見る</a>
```

```
<a href="https://example.com/~確認する" style="color: inherit; text-decoration: none;">~確認する</a>
```

```
<a href="https://example.com/~登録する" style="color: inherit; text-decoration: none;">~登録する</a>
```

```
<a href="https://example.com/~CHECK" style="color: inherit; text-decoration: none;">CHECK</a>
```

悪い例: 代替テキストを設定していない

```
<a href="https://example.com/~" alt="リンク" style="color: inherit; text-decoration: none;">
```

```

```

```
<a href="https://example.com/~" style="color: inherit; text-decoration: none;">
```

```
<span class="aria-hidden" style="color: inherit; text-decoration: none;">~確認する
```

良い例 (レベルAAA相当): a 要素のテキストコンテンツでリンク先を示す

ボタンにテキストラベルが存在せず、アイコンや画像のみがボタンを識別する唯一の要素である場合、それらに代替テキストを設けます。

```
<a href="#" style="color: inherit; text-decoration: none;">
```

```
 すぐ登録するアイテムを購入!
```

```
</a>
```

```
<img alt="Cart icon" style="vertical-align: middle;"/> カート
```

```
 <img alt="Checkout icon" style="vertical-align: middle;"/> コード・ブルソン
```

3. テキスト

テキストの実装方法について説明します。

テキストのサイズ変更ができるようにする

ブラウザの設定によって、ユーザーはウェブページのテキストサイズを変更することができます。CSSは適切に実装されていないと、そのユーザー設定に適応することができなかったり、ウェブページのレイアウトが崩れたりすることがあります。そうした問題を避けるために注意すべき点について説明します。

相対的な文字サイズにする

`font-size: 1em;` プロパティで相対的な値を指定します。

※悪い例: px 単位を使用している

```
.heading {
```

```
  font-size: 24px;
```

```
}
```

```
.text {
```

```
  font-size: 14px;
```

```
}
```

※良い例: em 単位を使用している

```
.heading {
```

```
  font-size: 1.5em;
```

```
}
```

```
.text {
```

```
  font-size: 0.875em;
```

```
}
```

ボックアップを埋め込むウェブページにおいて、HTML要素の `font-size` プロパティが変更されている場合、相対的な値を使用するに意図しない大きさになってしまふことがあります。必要に応じて、ウェブページ側の `font-size` プロパティを調整してください。

```
html {
```

```
  font-size: 62.5%;
```

```
}
```

```
.karte-widget {
```

```
  font-size: 1.6rem;
```

```
}
```

テンプレートの実装において、`em` 単位を使用すると、上記のような調節ができなくなってしまいます。原則として `em` 単

4. アイコン

アイコンの実装方法について説明します。

機能的な意味を含める場合

開けるボタンなどのアイコンとして使用する場合は、`span` 要素に `role="img"` を指定したうえで代替テキストを設定します。このとき、`aria-label` 属性を使用していません。

お探しの商品は見つかりました?

これにて商品の購入が可能となります。

あなたにあった人気商品を販売者からお届けします。

確認する

```
<button type="button" class="karte-close">
```

```
  <span class="icon-close" role="img" aria-label="閉じる"></span>
```

```
</button>
```

同様のテキストが提供されている場合、代替テキストの設定は不要です。「装飾として使用する場合」の指示に従ってください。

装飾として使用する場合

テキストのラベルが存在する要素に対して、附加的な装飾として使用する場合は、`span` 要素に `aria-hidden` 属性を指定します。

購入、コーヒーアイテム

コーヒー好きなアイテムをました。

コーヒー豆を買います。

コーヒー豆を買います。

ギフトセットを探す

ギフトセットを探す

確認する

[coffee-beans](#)

商品一覧

7. フォーム

アンケートなどのフォームの実装方法について説明します。

フォームの作成

フォームを作成するときは、全体を `form` 要素で包みます。作成した `form` 要素の `submit` イベントをハンドリングしてデータを保存します。

```
<form @submit.prevent="submitData">
```

```
  <h3>商品を購入してください!</h3>
```

```
  <p>サービス料込みの、アフターハードルを負担いたします。</p>
```

```
  <label>名前</label> <input type="text" name="name" />
```

```
  <label>メールアドレス</label> <input type="text" name="email" />
```

```
  <label>支払方法</label> <input type="radio" name="stripeMethod" checked="checked" /> お支払い方法を選択する場合
```

```
  <input type="radio" name="stripeMethod" /> お支払い方法を選択しない場合
```

```
  <input type="radio" name="stripeMethod" /> お支払い方法を選択しない場合
```

```
  <input type="radio" name="stripeMethod" /> お支払い方法を選択しない場合
```

```
  <button type="submit">準備完了!</button>
```

```
const scopeid = crypto.randomUUID().substring(0, 8);
```

```
window.setInterval(`id: ${id} => ${id}+scopeid();`, 1000);
```

関連する WCAG の達成基準

• 1.3.1情報及び関係性 (レベル A): 何らかの形で提示される情報、構造、及び関係性は、プログラムによる解釈が可能である。又はテキストで提示されています。

• 2.1.1.1 ページ (レベル A): コンテンツのすべての機能は、個々のキーストロークに特定のタイミングを要するところなく、キーボードインターフェースを通じて操作可能である。ただし、その根本的な機能が利用者の動作による終点だけではなく、操作の開始と終了の両方を必要とする場合がある。

• 2.1.2.1 ページ (レベル A): ページ内に表示されるすべてのデータ、リンク、イメージ、スクリプト、スタイルシート、スコープ、名前空間、属性、リスナー、スクロールが生成するコンボネートなしで、各名前 (name) 及び役割 (role) は、プログラムによる解釈が可能である。又は、構造、プロパティ、利用者が設定可能な値はプログラムによる設定が可能である。そして、技術実装も含むユーザージェントが、これらの項目に対する変更通知を発行できる。

代替テキストを設定する

ボタンにテキストラベルが存在せず、アイコンや画像のみがボタンを識別する唯一の要素である場合、それらに代替テキストを設けます。

テキストラベルのみの例:

```
<button type="button" class="karte-close">
```

```
  <span class="icon-close" aria-hidden="true"></span>
```

```
</button>
```

画面のみの例:

フォームを作成する際、並列な操作などを達成する方法があります。これはアクセシビリティの観点から好ましくないといわれています。この操作方法は、入力用の各ボタンの位置を固定せざるを得ません。

代わりに、送信ボタンは常に最初に配置してください。必ず最初に配置してください。この操作方法は、入り口用の各ボタンの位置を固定せざるを得ません。

片手間ではやり切れない

協力会社に依頼する

- ・アクセシビリティを専門とした外部の会社に依頼する
- ・プロジェクトの方針や進め方から相談中



デザインエンジニア募集中

- ・ プレイドでは、ユーザーインターフェースとフロントエンド技術が好きなエンジニアを募集しています
- ・ ユーザブルでアクセシブルなプロダクトを開発するために、あなたの力が必要です



Design Engineerの紹介