カスタムウィジェット

カスタムウィジェットの作成

Cisco Webex Contact Center Agent Desktopは、ウィジェットの形式で拡張機能をサポートします。ウィジェットは、デスクトップのカスタマイズに必須のコンポーネントです。ウィジェットは、特定のカプセル化された機能を備えたコンポーネントであり、デスクトップ内に配置されるカスタム HTML 要素としてエクスポートされます。より厳密には、これらのカスタム HTML 要素を Web コンポーネントにすることを推奨します。これは、どのウィジェットも最終的に Shadow Root のネストツリー内に配置されるためであり、スタイル設定のカプセル化（または何らかのスタイル設定）を有効にするためにはそれぞれ固有の Shadow Document Object Model（DOM）を有効にする必要があります。

Webex Contact Center Agent Desktopのウィジェットは Web コンポーネントまたは埋め込み iFrame である必要があるとわかると、開発者はテクノロジーを強制されるように感じるかもしれません。しかし、互換性のある任意のフレームワークでこれらのコンポーネントを構築できます。唯一の要件は、カスタムウィジェットを Web コンポーネントとしてエクスポートする必要があるという点です。これは、どのような機能も、Shadow DOM を有効にした状態で、JS バンドル内に定義される単一のカスタム HTML 要素としてラップする必要があるためです。

次に、いくつかの例を示します。

- Angular JS でウィジェットを構築する場合は、ご利用の Angular バージョンが Angular Elements（バージョン 6 で追加）をサポートしていることを確認し、アプリケーションを Angular Element としてエクスポートします。
- （エージェントデスクトップのパーツと同様に）React でアプリケーションを構築する場合は、作成する React アプリケーションをカスタム HTML 要素にラップして、Shadow DOM

を手動で有効にするか、このプロセスを自動化するオープンソースソリューションを利用します。たとえば、Direflow があります。

- Web コンポーネントの利用や Web コンポーネントとしてのエクスポートに完全に対応しているフレームワークは他にもさまざまなものがあります。参考として、広く利用されているフレームワークに対して定期的に自動テストを実行して継続的な互換性を確認している https://custom-elements-everywhere.com を利用できます。

ウィジェットとしてエクスポートする Web アプリケーションを構築するのに、フレームワークを使用する必要はありません。実際に、このマニュアルに記載しているすべての例で、通常の JavaScript を参照しています。

はじめに

Agent Desktop環境の規模が拡大してもウィジェットを配置できるような設定にすることを推奨します。参考として、Cisco DevNet の GitHub コミュニティ（https://github.com/CiscoDevNet/webex-contact-center-widget-starter）で入手可能な定型プロジェクトのいずれかを使用使用できます。

または、制約付きのコンテナにウィジェット（Webコンポーネント）を配置して、以下の項目についてテストできるようなサンドボックス環境を用意することも推奨します。

- さまざまなコンテナサイズ（推奨サイズから全画面まで）での応答動作。
- コンテンツがX軸またはY軸（スクロールバー）の方向にオーバーフローした場合のウィジェットの表示と動作に関する、オーバーフロー動作。
- ウィジェットに渡されるデータに対する反応。リアクティブ型のコンポーネントに関する実際の使用方法の詳細については、リアクティブプロパティまたは属性の変更 （3 ページ）を参照してください。
- Momentum UI Web コンポーネントのライブラリを使用しているかどうかに応じた、ライトモードとダークモードのサポート。

プロパティまたは属性インターフェイスの定義

前提条件：

Web コンポーネントとカスタム HTML 要素の操作について理解しておく必要があります。詳細については、MDN のドキュメンテーションを参照してください。

カスタム要素は、HTMLElement クラスまたはHTMLElement クラスから継承するクラスを拡張できます。

例：

```
class MyCustomElement extends HTMLElement {
    constructor() {
```

```
        // Always call super first in constructor
        super();
        const shadow = this.attachShadow({
            mode: 'open'
        });

        // write element functionality in here

        ...
    }
}
```

プロパティと属性を定義するには、コンストラクタでデフォルト値を割り当てる必要があります。

（注） データ型が文字列またはブール値であるデータを受け取る場合に限り、任意のプロパティを属性として渡すことができます。詳細については、JavaScript.Info を参照してください。

例：

```
class MyCustomElement extends HTMLElement {
    constructor() {
        // Always call super first in constructor
        super();
        const shadow = this.attachShadow({
            mode: 'open'
        });

        this.width = '100px';
        this.height = '100px';
        this.active = true;

        // Data of types Array, Object, Map, Set, etc. cannot be passed as an attribute
        this.options = [];
    }
}
```

リアクティブプロパティまたは属性の変更

エージェントデスクトップ（ウィジェット `connectedCallback()`）から属性とプロパティを使用して最新のデータを取得するには、`HTMLElement` ライフサイクルメソッドを使用します。詳細については、MDN のドキュメンテーションを参照してください。

（注） ライフサイクルメソッドが指定されていない場合、コンポーネントは、最初のレンダリング後に渡される新しいデータに反応しません。

例：

```
class MyCustomElement extends HTMLElement {
    constructor() {
        // Always call super first in constructor
        super();
```

```
        const shadow = this.attachShadow({
            mode: 'open'
        });

        this.width = '100px';
        this.height = '100px';
        this.active = true;

        // Data of types Array, Object, Map, Set, etc. cannot be passed as an attribute
        this.options = [];
    }

    attributeChangedCallback(name, oldValue, newValue) {
            console.log('Custom element attributes changed.', name);
        }

        ...
}
```

データプロバイダー：ウィジェットのプロパティと属性

（注） カスタムウィジェット内のプロパティまたは属性を使用してリアルタイムデータを受信するためには、管理者がレイアウト JSON 構成で適切に割り当てる必要があります。JSON レイアウトの詳細については、『*Cisco Webex Contact Center Setup and Administration Guide*（*Cisco Webex Contact Center* 設定およびアドミニストレーションガイド）』の「プロビジョニング（Provisioning）」の章の「デスクトップレイアウト（Desktop Layout）」セクションを参照してください。

JavaScript SDK サブスクライバを介してデータにアクセスするほかに、プロパティまたは属性を使用してデータを渡すようにリクエストすることもできます。プロパティまたは属性の変更に反応するようにコンポーネントが作成されている場合は、信頼できる唯一の情報源であるエージェントデスクトップから、この方法で更新されたデータを常に取得します。これはデータプロバイダーと呼ばれます。

現状では、キー STORE の下にデータプロバイダーが 1 つあります。以下に、STORE キーを使用して取得可能なすべてのデータと型定義の詳細を示します。

例：STORE キーの詳細

```
STORE.agent.agentId: string;
STORE.agent.agentName: string;
STORE.agent.agentPhoto: string;
STORE.agent.agentProfileID: string;
STORE.agent.allowConsultToQueue: boolean;
STORE.agent.dialPlan: Service.Aqm.Configs.DialPlan;
STORE.agent.dnNumber: string;
STORE.agent.enterpriseId: string;
STORE.agent.enterpriseId: string;
STORE.agent.cadVariables: Service.Aqm.Configs.CadVariables[];
STORE.agent.channels: {
    voiceCount: number,
    chatCount: number,
```

```
    emailCount: number,
    socialCount: number
  };
STORE.agent.idleCodes: Service.Aqm.Configs.Entity[];
STORE.agent.idleStatusTimestamp: Date;
STORE.agent.isAgentAvailableAfterOutdial: boolean;
STORE.agent.isAdhocDialingEnabled: boolean;
STORE.agent.isCampaignManagementEnabled: boolean;
STORE.agent.isEndCallEnabled: boolean;
STORE.agent.isOutboundEnabledForAgent: boolean;
STORE.agent.isOutboundEnabledForTenant: boolean;
STORE.agent.privacyShieldVisible: string;
STORE.agent.profileType: string;
STORE.agent.subStatus: string;
STORE.agent.subStatusChangeTimestamp: Date;
STORE.agent.teamId: string;
STORE.agent.teamName: string;
STORE.agent.wrapUpData: Service.Aqm.Configs.WrapupData;
```

例：アプリケーションレベル

```
STORE.app.logo: string;
STORE.app.title: string;
STORE.app.darkMode: boolean;
```

例：シングルサインオン（SSO）

```
STORE.auth.accessToken: string;
```

例：アプリケーション通知

```
STORE.generalNotifications.isNotificationsEnabled: boolean;
STORE.generalNotifications.isSilentNotificationsEnabled: boolean;
STORE.generalNotifications.countActivated: number;
STORE.generalNotifications.countDeactivated: number;
STORE.generalNotifications.countPending: number;
STORE.generalNotifications.countAdded: number;
```

例：JSON レイアウト

```
"widgets": {
        "my-component": {
          "comp": "my-custom-component",
          "properties": {
            "agentDnNumber": "$STORE.agent.dnNumber",
            "subStatus": "$STORE.agent.subStatus",
            "teamId": "$STORE.agent.teamId",
          }
        }
      }
```

Momentum UI Web コンポーネントライブラリ

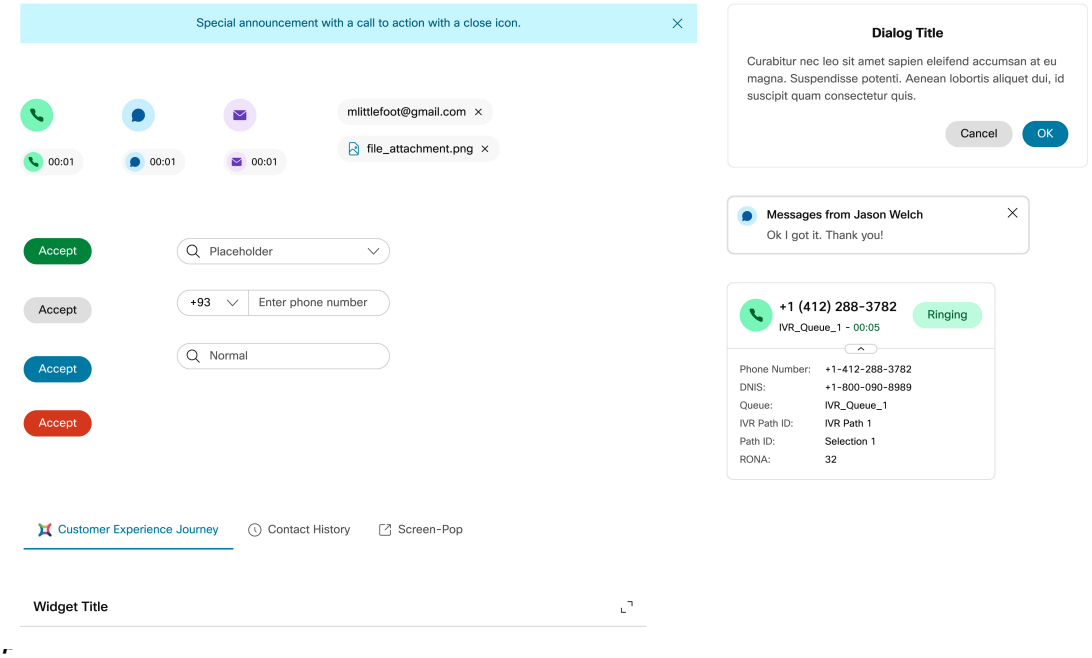
Webex Contact Center エージェントデスクトップのビジュアル言語を使用して、より短時間でカスタムウィジェットのインターフェイスを作成できるように、シスコの UI コンポーネントライブラリを確認することを推奨します。Momentum UI Web コンポーネントライブラリの詳細については、GitHub と MIT ライセンスを参照してください。問題が発生した場合は、GitHub ページで報告してください。

現在のコレクションには次のものが含まれます。

- アクティビティボタン
- アラート
- アラートバナー
- アバター
- バッジ
- トピックパス(パンくずリスト)
- ボタン
- チェックボックス
- 編集可能なフィールド
- フローティングモーダル
- ヘルプテキスト
- アイコン
- 入力
- ラベル
- リンク
- リスト
- 項目リスト
- 読み込み中インジケータ
- ミーティングアラート
- メニューのオーバーレイ
- モーダル
- 電話番号入力
- 進行状況バー（Progress Bar）
- 無線
- スピンボックス
- テーブル
- タブ
- タスク項目
- テーマ
- 切り替えスイッチ
- ヒント

Momentum UI Web コンポーネントライブラリは、今後も追加と改善が行われます。ぜひご意見をお寄せください。Web コンポーネントライブラリの詳細については、「Components（コンポーネント）」を参照してください。

図 *1* : サンプルのコンポーネント : *Agent*



ライトモードまたはダークモードのサポート

Webex Contact Center Agent Desktopでは、デフォルトではライトモードとダークモードをサポートしています。これは、Momentum UI Web コンポーネント <md-theme> によって実現されます。<md-theme> をコンポーネントにラップすると、事前定義されたコア CSS のカラー変数とカスタム CSS プロパティのコレクションに値が割り当てられます。これらのプロパティは、テーマのスイッチでライトからダークに切り替わります。CSS カスタムプロパティを利用すると、これらのプロパティは Shadow DOM を貫通し、Shadow DOM にネストされた多数のレイヤー内で使用できるという利点があります。

（注） ウィジェットの CSS が次の形式の場合、コア CSS のカラー変数を使用できます。

```
.container {
  background-color: var(--md-primary-bg-color, #fff);
}
```

例 : ライトモード

```
--md-default-focus-outline-color: #{$md-blue-60};

  --md-primary-bg-color: #{$md-white};
  --md-secondary-bg-color: #{$md-gray-05};
  --md-secondary-white-bg-color: #{$md-white};
  --md-tertiary-bg-color: #{$md-gray-10};
  --md-tertiary-white-bg-color: #{$md-white};
  --md-quaternary-bg-color: #{$md-gray-20};

  --md-primary-success-bg-color: #{$md-green-10};
```

```
  --md-primary-success-text-color: #{$md-green-70};

  --md-primary-text-color: #{$md-gray-100};
  --md-secondary-text-color: #{$md-gray-70};
  --md-disabled-text-color: #{$md-gray-40};
  --md-highlight-text-color: #{$md-theme-20};
  --md-hyperlink-text-color: #{$md-theme-70};
  --md-hyperlink-hover-text-color: #{$md-theme-90};
  --md-hyperlink-focus-text-color: #{$md-theme-70};

  --md-primary-seperator-color: #{$md-gray-30};
  --md-secondary-seperator-color: #{$md-gray-40};

  --md-presence-active-bg-color: #{$md-green-50};
  --md-presence-do-not-disturb-bg-color: #{$md-red-60};
  --md-presence-away-bg-color: #{$md-gray-30};
  --md-presence-busy-bg-color: #{$md-yellow-40};
```

例：ダークモード

```
--md-default-focus-outline-color: #{$md-blue-40};

  --md-primary-bg-color: #{$md-gray-100};
  --md-secondary-bg-color: #{$md-gray-95};
  --md-secondary-white-bg-color: #{$md-gray-95};
  --md-tertiary-bg-color: #{$md-gray-90};
  --md-tertiary-white-bg-color: #{$md-gray-90};
  --md-quaternary-bg-color: #{$md-gray-80};

  --md-primary-success-bg-color: #{$md-mint-70};
  --md-primary-success-text-color: #{$md-mint-20};

  --md-primary-text-color: #{$md-gray-05};
  --md-secondary-text-color: #{$md-gray-40};
  --md-disabled-text-color: #{$md-gray-70};
  --md-highlight-text-color: #{$md-theme-80};
  --md-hyperlink-text-color: #{$md-theme-40};
  --md-hyperlink-hover-text-color: #{$md-theme-20};
  --md-hyperlink-focus-text-color: #{$md-theme-40};

  --md-primary-seperator-color: #{$md-gray-70};
  --md-secondary-seperator-color: #{$md-gray-60};


  --md-presence-active-bg-color: #{$md-green-50};
  --md-presence-do-not-disturb-bg-color: #{$md-red-60};
  --md-presence-away-bg-color: #{$md-gray-30};
  --md-presence-busy-bg-color: #{$md-yellow-40};

  --md-auto-wrapup-bg-color: #{$md-blue-90};
```