日日是Oracle APEX

Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2021年11月26日金曜日

Oracle APEX 21.2新機能(22) - JavaScript APIの追加

Oracle APEX 21.2で追加されたJavaScript APIについて紹介します。リリース・ノートの記載はこちらです。

追加されたネームスペースは以下です。**apex.env**、**apex.items**、**apex.regions**はJavaScript APIのドキュメントのNamespacesの一覧ではなく、apexの**Properties**に記載があります。**apex.pwa**については、PWAの説明で触れています。

- apex.env
- apex.items
- apex.regions
- apex.date
- apex.pwa

JavaScript APIのリファレンスに追加されたインターフェースは以下です。

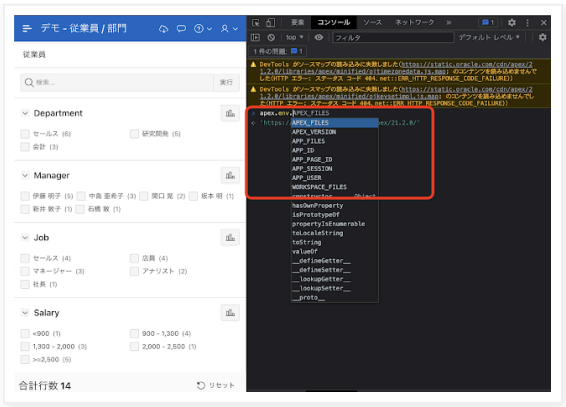
- facetsRegion
- mapRegion
- numberFieldItem

ただ、インターフェース（およびそれを実装しているウィジェット）についてはそもそもJavaScript APIリファレンスに記載されていないものが多数あります。リージョンのインターフェースについては、それぞれのリージョンのカスタマイズ（例えば対話グリッドやファセット）方法を紹介する際に取り上げたいと思います。

以下より、ネームスペースとして追加されたAPIについて確認してみます。

apex.env

ブラウザの**JavaScriptコンソール**を開き、**apex.env**のネームスペースに含まれるプロパティを確認します。



APP_USER(サインインしているユーザー名)、**APP_FILES**(静的アプリケーション・ファイルへのパス)といった値を参照することができます。コード・エディタでの補完の対象になります。

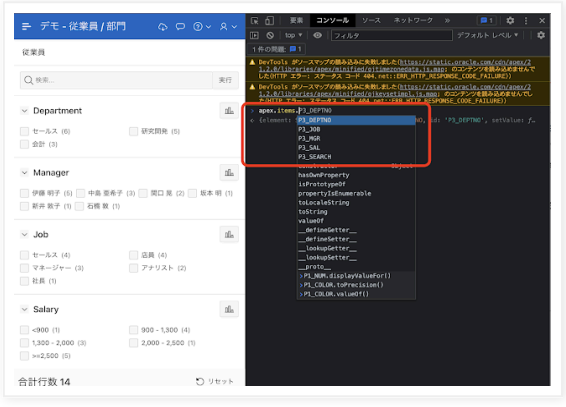
アプリケーション・ビルダーのマニュアルに記載がありますが、システムが提供している置換文字列の名称が更新されています。apex.envにてサポートされているのは、21.2から有効になった置換文字列の名称のみです。

- IMAGE_PREFIX -> APEX_FILES
- WORKSPACE_IMAGES -> WORKSPACE_FILES
- APP_IMAGES -> APP_FILES
- THEME_IMAGES -> THEME_FILES
- THEME_DB_IMAGES -> THEME_DB_FILES

静的ファイルは画像に限らないので、より適切になるようファイルを指す名称に変更されています。

apex.items

現在開かれているページ上のページ・アイテムが含まれます。



例えばページ・アイテム**P3_DEPTNO**は、以下のようにオブジェクトとして取得できます。

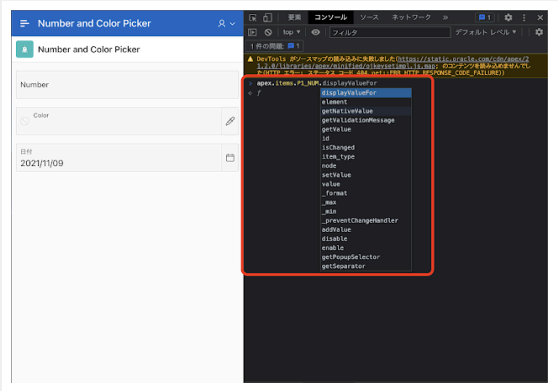
```
let myItem = apex.items.P3_DEPTNO;
```

これは以下の記法と同等です。

```
let myItem = apex.item("P3_DEPTNO");
```

取得したオブジェクトは、どちらも同じです。

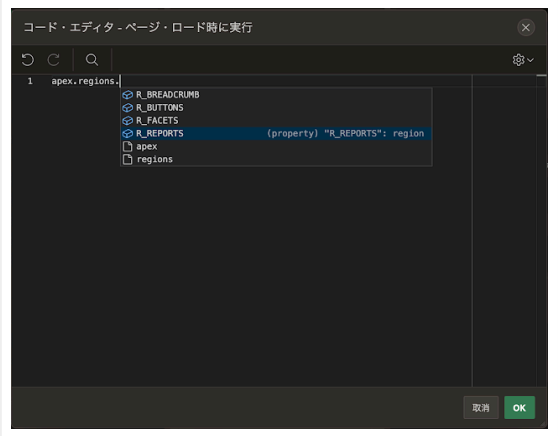
今回のバージョンより、JavaScript APIのリファレンスにnumberFieldItemのインターフェースについて記載されました。このインターフェースは**アイテム・タイプ**が**数値フィールド**のページ・アイテムに実装されています。実際には、数値フィールド以外でもアイテム・タイプごとに異なるインターフェースが実装されています。概ねAPEXのメディアのimages/libraries/apex/以下にwidget.XXXX.jsとして実装が記載されていますが、今回の拡張によりページ・アイテムを選択して、そのページ・アイテム（のオブジェクト）に実装されているプロパティやメソッドを補完することができるようになりました。



そのため、JavaScriptのコーディングの負担がかなり軽減されます。また、JavaScript APIリファレンスに説明のないインターフェースでも、どのようなプロパティやメソッドが実装されているか確認できます。

apex.regions

現在開かれているページ上のリージョンが含まれます。補完の対象となるのはリージョンの静的IDです。こちらはブラウザの開発ツールからはうまく補完されませんでした。ページ・デザイナからJavaScriptのコード・エディタを開いて確認しています。



例えばリージョン**R_FACETS**(静的IDをR_FACETSとして設定)は、以下のようにオブジェクトとして取得できます。

```
let myRegion = apex.regions.P_FACETS;
```

これは以下の記法と同等です。

```
let myRegion = apex.regions("R_FACETS");
```

取得したオブジェクトは、どちらも同じです。

apex.itemsと同様にリージョンのタイプに応じてプロパティやメソッドが補完されるため、コーディングの負担がかなり軽減されます。

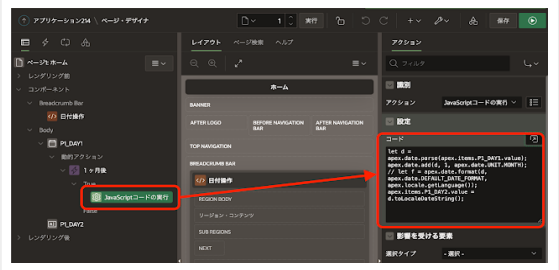
apex.date

日付を操作するファンクションを提供します。JavaScriptで日付を扱うためにDay.jsなどのライブラリを導入する必要が無くなります。

選択した日付から、apex.date.addを呼び出して１ヶ月後の日付を求めるアプリケーションを作ってみます。

左のページ・アイテムP1_DAY1が変更されたときに動的アクションを呼び出し、以下のスクリプトを実行します。

```
let d = apex.date.parse(apex.items.P1_DAY1.value);
apex.date.add(d, 1, apex.date.UNIT.MONTH);
// apex.items.P1_DAY2.value = apex.date.format(d);
apex.items.P1_DAY2.value = d.toLocaleDateString();
```



このようなケースでは以下のようにPL/SQLを記述し、データベースのADD_MONTHSファンクションを使っていたケースもあったかと思います。

```
begin
    :P1_DAY2 := add_months(:P1_DAY1, 1);
end;
```

上記の実装ではサーバーとの通信が発生するため、画面の反応は今ひとつでした。これからは日付の操作の多くをブラウザ側で実行できるため、画面の反応も改善できるでしょう。

apex.date.formatはオラクルの日付書式フォーマットによって、JavaScriptのDateを文字列にするファンクションです。ただし、現行では書式フォーマット"DD"は日本語で適切に変換できない不具合があります。

DDの変換を以下のファイルから確認してみます。
https://static.oracle.com/cdn/apex/21.2.0/libraries/apex/date.js

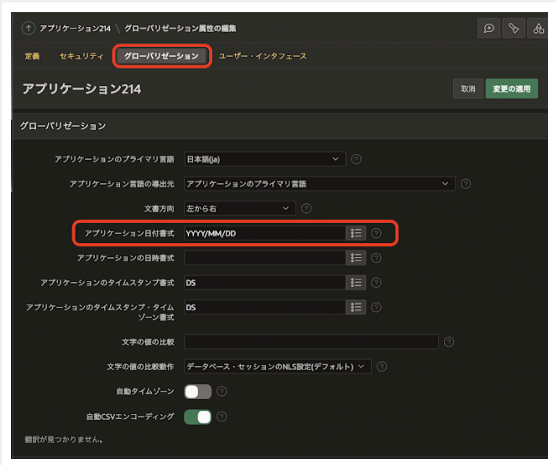
```
DD: ( d ) => {
    return ( "0" + d.toLocaleString( "default", { day: "numeric" } ) ).slice( -2 );
},
```

d.toLocaleStringは日本語だとnumeric - 数値だけではなく"日"も後ろにつけてしまいます。getDateを使うと回避できます。

```
DD: ( d ) => {
    return ( "0" + d.getDate() ).slice( -2 );
},
```

すでに開発チームには修正を依頼しています。

日付関連ではapex.locale.getDateFormatファンクションが新設されています。呼び出すとアプリケーション定義の**アプリケーション日付書式**が返されます。



JavaScriptによる開発全般に使用できるAPIの追加については以上になります。追加、変更されたAPIは他にもありますが、それは別の機会に紹介できればと思います。

完

Yuji N. 時刻: 11:02

自己紹介

Yuji N.

日本オラクル株式会社に勤務していて、Oracle APEXのGroundbreaker Advocateを拝命しました。
こちらの記事につきましては、免責事項の参照をお願いいたします。

詳細プロフィールを表示