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。

2022年4月29日金曜日

動的アクションのJavaScriptコードで使用するthisについて

動的アクションのTrueアクションをJavaScriptでコーディングする際に、this.triggeringElement、this.browserEventなどが使用できます。この使い方を紹介します。

アクションとしてJavaScriptコードの実行を選択したときの、設定のコードのオンライン・ヘルプには、以下の記載があります。

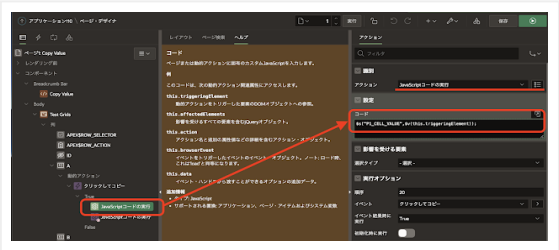
- **this.triggeringElement**
  - 動的アクションをトリガーした要素のDOMオブジェクトへの参照。
- **this.affectedElements**
  - 影響を受けるすべての要素を含むjQueryオブジェクト。
- **this.action**
  - アクション名と追加の属性値などの詳細を含むアクション・オブジェクト。
- **this.browserEvent**
  - イベントをトリガーしたイベントのイベント・オブジェクト。ノート: ロード時、これは'load'と同等になります。
- **this.data**
  - イベント・ハンドラから渡すことができるオプションの追加データ。

一般にOracle APEXでアプリケーションを開発している方は、HTML、CSS、JavaScriptはそれほど経験がない方が多いといわれています。なので、この説明だけではピンとこないようです。

先日作成したアプリケーションを使って、これらの属性を使ってみます。

そもそも、この**this**は何か、から確認します。

TRUEアクションに以下の**コード**が記述されています。(ページ・アイテムへの値の移入を題材にとります。列への移入はコメント・アウトします。)



このように作成した動的アクションは、HTMLのソースとして以下のように出力されます。

```
<script type="text/javascript">
apex.da.initDaEventList = function(){
    apex.da.gEventList = [{
        "triggeringElementType":"COLUMN",
        "triggeringElement":"C11489677485938601",
        "triggeringRegionId":"test_grids",
        "isIGRegion":true,
        "bindType":"bind",
        "bindEventType":"click",
        "anyActionsFireOnInit":false,
        actionList [{
            "eventResult":true,
            "executeOnPageInit":false,
            "stopExecutionOnError":true,
            javascriptFunction:function (){
                $s("P1_CELL_VALUE",$v(this.triggeringElement));
            },
            "action":"NATIVE_JAVASCRIPT_CODE"
        }]
    }];
}
</script>
```

設定の**コード**のJavaScriptの記述が、ファンクションとして**actionList**に登録されています。**this**はこのファンクションの実行コンテキストになります。

<script>から**</script>**の間に記載されている内容は、概ねページに作成された動的アクションの定義を転記しています。動的アクションの定義リストが**apex.da.initDaEventList**になります。このリストを受け取って、動的アクションを実行するコードは**dynamic_actions_core.js**に含まれています。

https://apex.oracle.com/i/libraries/apex/dynamic_actions_core.js

dynamic_actions_core.jsは、すべてのAPEXのインストールに含まれています。（実際に、ページにロードされているのはミニファイされたdesktop_all.min.jsです。）

JavaScriptのコードを実行する部分は、以下のようにコーディングされています。

```
/**
 * doAction function
 * Executes the action (pAction) on certain elements (pSelector)
 * @ignore
 */
da.doAction = function( pContext, pSelector, pAction, pDynamicActionName, pResumeCallback ) {
    var lContext = {
        triggeringElement : pContext.triggeringElement,
        affectedElements  : $( pSelector, apex.gPageContext$ ),
        action            : pAction,
        browserEvent      : pContext.browserEvent,
        data              : pContext.data,
        resumeCallback    : pResumeCallback
    };
```

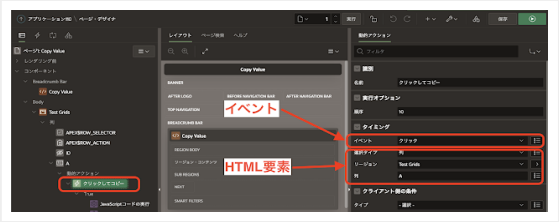
```
        // Call the javascript function if one is defined and pass the lContext object as this
        if ( pAction.javascriptFunction ) {

            // Log details of dynamic action fired out to the console (only outputs when running in debug mode)
            apex.debug.log( "Dynamic Action Fired: " + pDynamicActionName + " (" + pAction.action + ")", lContext );
            return pAction.javascriptFunction.call( lContext );
        }
    }; // doAction
```

変数**lContext**が、実行コンテキストになっていることが分かります。

では、lContextに含まれるtriggeringElementやbrowserEventは何か？ということになります。

動的アクションは以下のように定義されています。

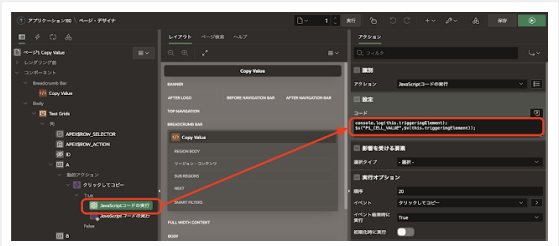


指定した**HTML要素（リージョンTest Gridsの列A）**が、**クリック**されたときにアクションが実行されます。

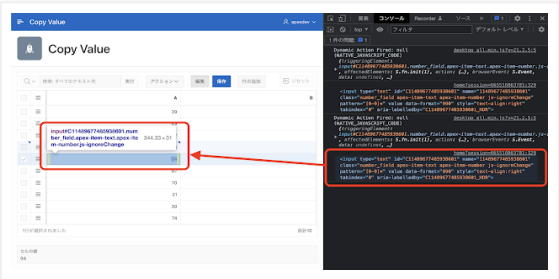
よって、**this.triggeringElement**にはクリックされた**列Aのセル**への参照が含まれます。
this.browserEventにはクリックであるMouseEventが含まれます。MouseEventはEventから派生したインターフェースで、イベントの種類によってインターフェースは変わります。

TRUEアクションのコードにconsole.log(this.triggeringElement);を追加して、内容をJavaScriptコンソールに出力してみます。

```
console.log(this.triggeringElement);
$s("P1_CELL_VALUE",$v(this.triggeringElement));
```



対話グリッドの列Aをクリックすると、JavaScriptコンソールに**クリックしたセルのHTML要素（inputタグ）**が印刷されます。



this.triggeringElementとしてHTML要素、this.browserEventとしてMouseEventが渡されていることがわかったので、これを使ったコードを追加してみます。

Shiftキーを押してクリックしたときだけ、値をコピーするようにします。

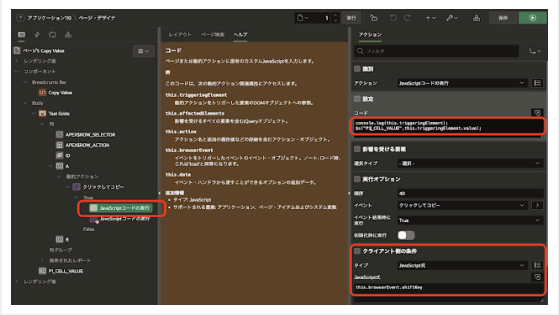
this.triggeringElementはinputのHTML要素ですから、APEXが提供しているファンクション$vの代わりに**.value**でも値を取ることができます。

コードは以下に変更します。

$s("P1_CELL_VALUE",this.triggeringElement.value);

クライアントの条件の**タイプ**に**JavaScript式**を選択し、**JavaScript式**に以下を記述します。シフト・キーを押した状態でMouseEventが発生したときに**true**になります。

this.browserEvent.shiftKey

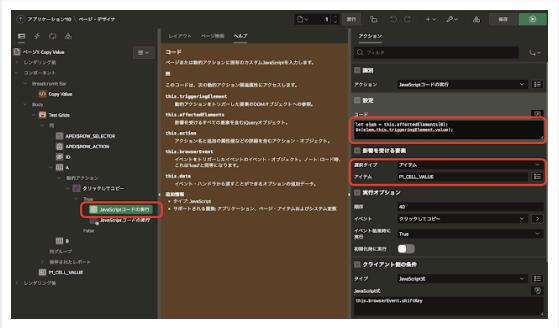


this.action、**this.affectedElements**、**this.data**は、主にOracle APEXに依存した情報になります。

this.actionからは、呼び出されているTRUEアクションの定義を参照できます。プラグインを作成するようなケースを除き、参照することはないと思います。

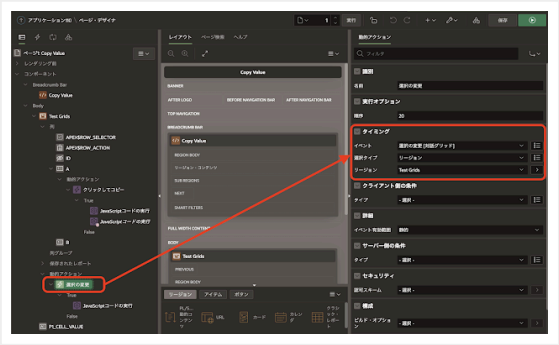
this.affectedElementsは、**アクション**の**影響を受ける要素**として設定された**HTML要素**が（配列として）渡されます。今回のTRUEアクションは、**影響を受ける要素**に**P1_CELL_VALUE**を設定して、次のように書き直すことができます。

```
let elem = this.affectedElements[0];
$s(elem,this.triggeringElement.value);
```

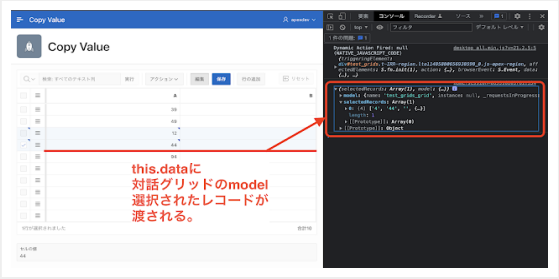


this.dataとして渡される値は動的アクションによって異なります。

例えば、**対話グリッド**で**選択を変更**したときに発生するイベントで、動的アクションを作成します。

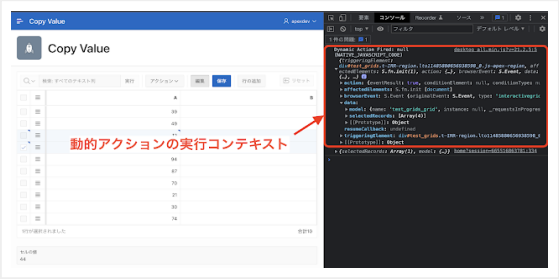


TRUEアクションとしてconsole.log(this.data);を実行し、内容を確認します。



this.dataには、対話グリッドのmodelと選択されたレコードが含まれていることが確認できます。

最後に、アプリケーション開発中であればthisの内容を確認するためにconsole.logを埋め込む必要はありません。開発者ツール・バーが開いている状態で、JavaScriptコンソールを開いていれば、動的アクションが実行される都度、thisの内容がコンソールに出力されます。



以上で、動的アクションのJavaScriptコードで使用するthisの紹介は終了です。

Oracle APEXのアプリケーション作成の参考になれば幸いです。

完

Yuji N. 時刻: 13:29

共有

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