Knockout.jsの運用

前提

- エンタープライズWebアプリを作成することを前提として考える
- サーバーサイドレンダリングは、必要最低限の画面情報のみとする。
- サーバーサイド、クライアントサイドのデータはjson形式とする。
- マスタデータによる動的な画面情報の構築（selectタグなど）が必要な場合も、jsonデータを利用することにする。
- データの取得・検索・保存などは、全てAjaxを利用することにする。

双方向バインディングを必ず使う

Knockout.jsでは、片方向バインディングと双方向バインディングの何れかを選択することができる。

結論から言うと、双方向バインディング（observable）を必ず利用することにする。

何故、双方向バインディングか？

双方向では、ViewModelを更新するとUIが更新され、UIを更新するとViewModelが更新される。まとめると、双方向にデータが同期されるため、Knockout.jsを利用するメリットを最大限に享受できる。

Mappingプラグインを必ず使う

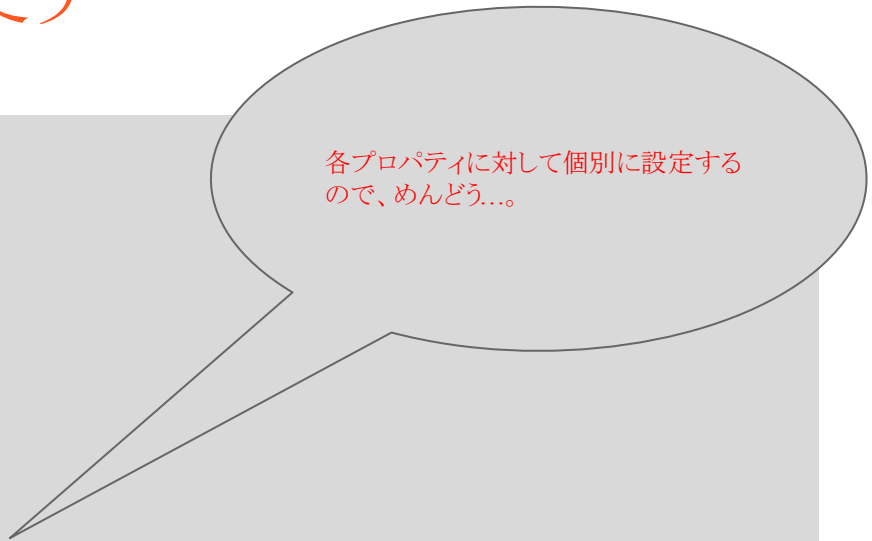
ViewModelの各プロパティにobservableを適用するのは非常に面倒。よって、Mappingプラグインを必ず使うこと。

Mappingプラグインを必ず使う

・通常

```
// 以下は、Ajaxにてjsonとして取得するという想定
var myViewModel = {
    personName: 'ボブ',
    personAge: 123
};


var myViewModelObs = {
    personName: ko.observable(myViewModel.personName),
    personAge: ko.observable(myViewModel.personAge)
};
```

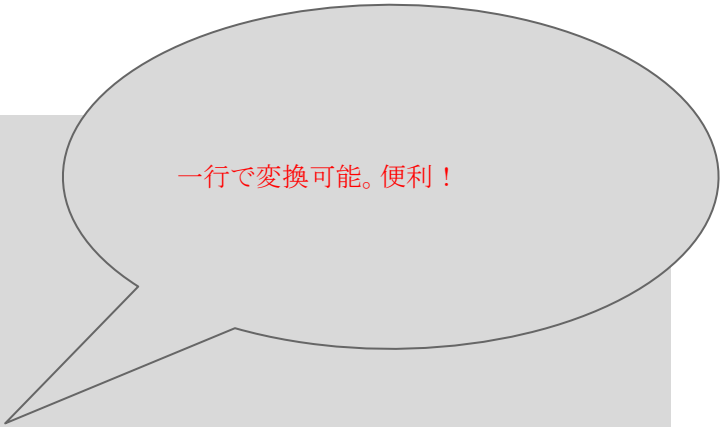


Mappingプラグインを必ず使う

・Mappingプラグイン

```
// 以下は、Ajaxにてjsonとして取得するという想定
var myViewModel = {
    personName: 'ボブ',
    personAge: 123
};

// 以下のユーティリティ関数にjsonデータを入れ込むことで、observableなデータオブジェクトに変換することができる
var myViewModelObs = ko.mapping.fromJS(myViewModel);
```



Knockout.jsにおけるイベントは使用禁止

DOMに対してのイベントハンドリングは、jQueryの on / off を使用する。jQueryを使用することで、特定のグループ（一つまたは一つ以上）に対するイベント関数を登録できるが、Knockout.jsでは、単体のDOMに対して個別にイベント関数のバインドを適用しないといけないため、不便に感じることもある。

よって、イベントハンドリングは、jQueryで一元化する。

Knockout.jsのViewModelのライフサイクル

1. AjaxでデータロードしてViewModelに変換（マッピングプラグイン）

2. ↓↓ ここから生成

3. ユーザーが画面上で色々とデータを入力する
4. ユーザーの入力に伴い、UIに与えた影響がViewModelに同期される
5. ユーザーの入力に伴って発生したイベントにより、javascriptコードがバックグラウンドで作用する中で、ViewModelなどを編集することで、UIを更新する
6. サーバーに HTTP送信する際に、Formデータをserializeすることで、入力されたデータを抽出する
   $("#form").serialize()

7. ↑↑ 後続処理でページ遷移した時点で、ViewModelが破棄される