

M2 CPFED

5. Knockout /

UI components

John Hughes

Head of Magento - Fisheye Media Ltd.

Overview.

- Introduction
- Knockout bindings
- Knockout observables
- UI component configuration
- Linking UI components

What you think working with KO should be...



Reality...



Introduction

Introduction.

- **What is Knockout (KO)?**
- What are UI components?

Introduction.

What is Knockout (KO)?

- **Another outdated JS framework...?**
- Enables binding of data to DOM elements and observing their values to automatically update them
- Heavily used (and customised) in the Magento 2 frontend

Introduction.

- What is Knockout (KO)?
- **What are UI components?**

Introduction.

What are UI components?

- **A mess...?**
- Javascript modules that define / manage data for for a UI element (or component) on the page
- Make heavy use of Knockout for data binding (updates)
- Allows mapping of KO to templates

Introduction.

What are UI components?

- If you see a spinner on the frontend / admin panel - **it's most likely a UI component!**
- Mainly used to power grids/lists & forms in the admin panel
- Main frontend usage is for the checkout

Knockout bindings

Knockout bindings.

- **What are KO bindings?**
- How do you use KO bindings?
- What are common KO bindings?
- What is KO view model scope binding?

Knockout bindings.

What are KO bindings?

- Bindings allow linking of reactive (changeable) data to DOM elements

Knockout bindings.

- What are KO bindings?
- **How do you use KO bindings?**
- What are common KO bindings?
- What is KO view model scope binding?

<my-template>.html

Using data-bind attribute

```
<span data-bind="text: title"></span>
```

Value



Binding type



<my-template>.html

Using custom attribute
e.g. **text**

Value





Binding type

<my-template>.html

Using a 'virtual element'

Value



```
<!-- ko text: title --><!-- /ko>
```



Binding type

<my-template>.html

```
<span data-bind="text: title"></span>
```

```
<span text="title"></span>
```

```
<!-- ko text: title --><!-- /ko>
```

Knockout bindings.

- What are KO bindings?
- How do you use KO bindings?
- **What are common KO bindings?**
- What is KO view model scope binding?

Knockout bindings.

Type	Overview
text	Set innerText of an element
html	Set HTML content of an element
visible	Hide or display an element
css	Bind css classes to an element
style	Bind styles to an element
attr	Bind data to a given attribute for an element e.g. a URL to href

Knockout bindings.

Type	Overview
foreach	Loop through array of data
if / ifnot	Like visible, but based on condition
click	Click event listener
value	Bind to elements value attribute
i18n	Magento custom translatable text binding
and more...	https://knockoutjs.com/documentation/introduction.html https://devdocs.magento.com/guides/v2.1/ui_comp_guide/concepts/knockout-bindings.html

Knockout bindings.

- What are KO bindings?
- How do you use KO bindings?
- What are common KO bindings?
- **What is KO view model scope binding?**


Knockout bindings.

What is KO view model scope binding?

- Sets the scope for an element and it's children
- i.e. the linked JS component to bind data with

<my-template>.html

```
<div data-bind="scope: 'scope-name'">  
  <span data-bind="text: title"></span>  
</div>
```



Any content within this element is bound to a component of the same name

Knockout observables

Knockout observables.

- **What are KO observables?**
- How do you use KO observables?
- How do observables work with arrays?
- What are computed observables?

Knockout observables.

What are KO observables?

- Used to track and update variable values (and the UI) automatically

Knockout observables.

- What are KO observables?
- **How do you use KO observables?**
- How do observables work with arrays?
- What are computed observables?

Knockout observables.

How do you use KO observables?

- Using the **ko.observable()** method

<my-component>.js

```
// Set variable as observable
```

```
var name = ko.observable('Max Eisenhardt');
```

```
// Update by calling as a function
```

```
name('Magneto');
```

<my-component>.js

```
// Manually watching changes (rarely used)
name.subscribe(function(newValue) {
    alert('Name has changed to ' + newValue);
});
```

Knockout observables.

- What are KO observables?
- How do you use KO observables?
- **How do observables work with arrays?**
- What are computed observables?

Knockout observables.

How do observables work with arrays?

- Using **ko.observableArray([])**

<my-component>.js

```
// Initialise array with values
```

```
var superheroes = ko.observableArray([  
    'Magneto',  
    'Batman',  
    'Superman'  
]);
```

```
// Add more values
```

```
superheroes.push('Wonder Woman');
```

Knockout observables.

- What are KO observables?
- How do you use KO observables?
- How do observables work with arrays?
- **What are computed observables?**

Knockout observables.

What are computed observables?

- Functions that can contain multiple child observables that also update when any one of the children change

<my-component>.js

```
var name = ko.observable('Max Eisenhardt');  
var alterEgo = ko.observable('Magneto');  
  
var superhero = ko.computed(function() {  
    return name() + ' is ' + alterEgo();  
});
```

Ui component configuration



PROFESSIONAL
SOLUTIONS PARTNER

:fisheye

UI component configuration.

- **How do you specify UI component config?**
- What are common config options?
- How do you specify a components template?
- Whats is default.tracks?

UI component configuration.

How do you specify UI component config?

- In the JS component itself (using **defaults**)
- JSON (normally directly in a **.phtml** template file)
- Layout XML (Christmas trees)
- Layout providers (backend)

<my-component>.js

```
define([
  'jquery',
  'uiComponent',
  'Ko'
], function ($, Component, ko) {
  'use strict';

  return Component.extend({
    defaults: {
      key: 'value'
    }
  });
});
```

<my-component>.js

```
define([
    'jquery',
    'uiComponent',
    'Ko'
], function ($, Component, ko) {
    'use strict';

    return Component.extend({
        defaults: {
            key: 'value'
        }
    });
});
```

<some-file>.phtml

```
<div id="someElement"></div>
```

```
<script type="text/x-magento-init">
```

```
{
```

```
    "#someElement": {
```

```
        "Magento_Ui/js/core/app": {
```

```
            "components": {
```

```
                "myComponentName": {
```

```
                    "component": "Vendor_Module/js/path/to/file",
```

```
                    "config": {
```

```
                        "key": "value"
```

```
                    }
```

```
            }
```

```
        }
```

```
    }
```

```
}
```

```
}
```

```
</script>
```

<some-file>.phtml

```
<div id="someElement"></div>
<script type="text/x-magento-init">
  {
    "#someElement": {
      "Magento_Ui/js/core/app": {
        "components": {
          "myComponentName": {
            "component": "Vendor_Module/js/path/to/file",
            "config": {
              "key": "value"
            }
          }
        }
      }
    }
  }
</script>
```


<some-handle>.xml

```
<block class="..." name="..." template="...">
  <arguments>
    <argument name="jsLayout" xsi:type="array">
      <item name="components" xsi:type="array">
        <item name="myComponent" xsi:type="array">
          <item name="component" xsi:type="string">
            Vendor_Module/js/path/to/file
          </item>
          <item name="config" xsi:type="array">
            <item name="key" xsi:type="string">
              value
            </item>
          </item>
        </item>
      </item>
    </argument>
  </arguments>
</block>
```

<some-handle>.xml

```
<block class="..." name="..." template="...">
  <arguments>
    <argument name="jsLayout" xsi:type="array">
      <item name="components" xsi:type="array">
        <item name="myComponent" xsi:type="array">
          <item name="component" xsi:type="string">
            Vendor_Module/js/path/to/file
          </item>
          <item name="config" xsi:type="array">
            <item name="key" xsi:type="string">
              value
            </item>
          </item>
        </item>
      </item>
    </argument>
  </arguments>
</block>
```

<some-file>.phtml

```
<div id="someElement"></div>
<script type="text/x-magento-init">
  {
    "#someElement": {
      "Magento_Ui/js/core/app": <?= $block->getJsLayout() ?>
    }
  }
</script>
```

<some-file>.phtml

```
<div id="someElement"></div>
<script type="text/x-magento-init">
  {
    "#someElement": {
      "Magento_Ui/js/core/app": <?= $block->getJsLayout() ?>
    }
  }
</script>
```

UI component configuration.

- How do you specify UI component config?
- **What are common config options?**
- How do you specify a components template?
- Whats is default.tracks?

UI component configuration.

Config value	Usage
template	Template file to render the component
elementTpl	Template file for a reusable element (e.g. input / select)
title	Component title (normally rendered in template)
deps	Components this component depends on

UI component configuration.

- How do you specify UI component config?
- What are common config options?
- **How do you specify a components template?**
- Whats is default.tracks?

<my-component>.js

```
define([
    'jquery',
    'uiComponent',
    'Ko'
], function ($, Component, ko) {
    'use strict';

    return Component.extend({
        defaults: {
            template: 'Vendor_Module/path/to/file'
        }
    });
});
```


<some-file>.phtml

```
<div id="someElement"></div>
<script type="text/x-magento-init">
  {
    "#someElement": {
      "Magento_Ui/js/core/app": {
        "components": {
          "myComponentName": {
            "component": "Vendor_Module/js/path/to/file",
            "config": {
              "template": "Vendor_Module/path/to/file"
            }
          }
        }
      }
    }
  }
</script>
```

<some-handle>.xml

```
<block class="..." name="..." template="...">
  <arguments>
    <argument name="jsLayout" xsi:type="array">
      <item name="components" xsi:type="array">
        <item name="myComponent" xsi:type="array">
          <item name="component" xsi:type="string">
            Vendor_Module/js/path/to/file
          </item>
          <item name="config" xsi:type="array">
            <item name="template" xsi:type="string">
              Vendor_Module/path/to/file
            </item>
          </item>
        </item>
      </item>
    </item>
  </arguments>
</block>
```

UI component configuration.

- How do you specify UI component config?
- What are common config options?
- How do you specify a components template?
- **Whats is default.tracks?**

UI component configuration.

Whats is default.tracks?

- A shortcut to setting KO observables on a component
- i.e. setting which variables you want to track changes for
- Variables can be updated like any other JS variable

<my-component>.js

```
return Component.extend({
  defaults: {
    name: ko.observable('Max Eisenhardt'),
    alterEgo: 'Magneto',
    tracks: {
      alterEgo: true
    }
  },
  updateHero: function () {
    // Must be updated using ()
    this.name('Peter Parker');
    // Can be updated like a standard variable
    this.alterEgo = 'Spider-Man';
  }
});
```

<my-component>.js

```
return Component.extend({
  defaults: {
    name: ko.observable('Max Eisenhardt'),
    alterEgo: 'Magneto',
    tracks: {
      alterEgo: true
    }
  },
  updateHero: function () {
    // Must be updated using ()
    this.name('Peter Parker');
    // Can be updated like a standard variable
    this.alterEgo = 'Spider-Man';
  }
});
```

Linking UI components

Linking UI components.

- **How can UI components be linked?**
- What are the link types?
- Explain template strings usage

Linking UI components.

How can UI components be linked?

- Using **exports**, **imports**, **links** and **listens** properties
- Can be set in **defaults** within JS component or under **config** in layout XML

Linking UI components.

- How can UI components be linked?
- **What are the link types?**
- Explain template strings usage

Linking UI components.

Type	Overview
exports	Copy a local property (e.g. variable) to external component
imports	Track changes for a property of an external component
links	Essentially dual purpose (exports and imports)
listens	Used to track the changes of a component's property

superheroes.js

```
return Component.extend({
  defaults: {
    superheroes: [
      'Batman',
      'Superman'
    ],
    provider: 'supervillains',
    supervillains: '',
    tracks: {
      superheroes: true,
      supervillains: true
    },
    imports: {
      supervillains: '${$.provider}:supervillains'
    }
  }
});
```

supervillains.js

```
return Component.extend({
  defaults: {
    supervillains: [
      'Joker',
      'Lex Luther'
    ],
    tracks: {
      supervillains: true,
    }
  }
});
```

superheroes.html

```
<h3>Heroes</h3>
```

```
<ul>
```

```
  <!-- ko foreach: superheroes →
```

```
    <li data-bind="text: $data"></li>
```

```
  <!--/ko→
```

```
</ul>
```

```
<h3>Villains</h3>
```

```
<p>Imported using `default.tracks`</p>
```

```
<ul>
```

```
  <!-- ko foreach: supervillains →
```

```
    <li data-bind="text: $data"></li>
```

```
  <!--/ko→
```

```
</ul>
```

Linking UI components.

- How can UI components be linked?
- What are the link types?
- **Explain template strings usage**

Linking UI components.

How can UI components be linked?

- `${}` evaluates expression within the curly braces and converts to a string
- Would normally be expressed within backticks
- Inside `${}`, `$` equals **this**, so `${$.provider}` would be **`${this.provider}`**

<some-file>.js

```
var string = 'some string';
```

```
var sentence = `Including {string} within another string`;
```

superheroes.js

```
return Component.extend({
  defaults: {
    superheroes: [
      'Batman',
      'Superman'
    ],
    provider: 'supervillains',
    supervillains: '',
    tracks: {
      superheroes: true,
      supervillains: true
    },
    imports: {
      supervillains: '${$.provider}:supervillains'
    }
  }
});
```

Wrap up / resources

Code challenge 1.

- Create a simple UI component (in a template) and render some text to an element using KO binding
- Override the text using config
- Make the element observable and update after an interval
- Move the component JSON to layout XML (including config)

Code challenge 2.

- Render the element in a dedicated .html template file
- Add an observable array and loop through the elements
- Use ko.computed to manage multiple observables and update a child observable after an interval
- Use the 'click' binding to update one of your created observables

Code challenge 3.

- Create a secondary component and test out linking:
 - Import
 - Export
 - Link
 - Listen

Credit / Resources / Thanks.

- **Official Magento Study Guide**
- **Swift Otter Study Guide**
- **Magento DevDocs**