Bindings and Observables

John Papa

@john_papa

http://johnpapa.net

Outline

- Without Data Binding
- Observables
- Computed Observables
- ObservableArray
- Subscribing to Changes

Without Data Binding

- **Manual push**
  - from source object to target elements
- **Manual pull**
  - From target elements to source object
- **jQuery can assist**
  - Simplify code, but code still required
- **What about “Data Binding” Notifications?**
  - When do you push, when do you pull?
  - Not true data binding

Manual Push via jQuery

```
<span>Guitar model:</span><input id="guitarModel" />
<span>Sales price:</span><input id="guitarSalesPrice" />
```

```
var product = {
  id: 1001, model: "Taylor 314ce", salePrice: 1199.95
};
```

Source object

```
$("#guitarModel").val(product.model);
$("#guitarSalesPrice").val(product.salePrice);
```

Push from Source to Target

DEMO

Without Data Binding

Outline

- Without Data Binding
- Observables
- Computed Observables
- ObservableArray
- Subscribing to Changes

Knockout's Observables

- Wrap properties in observable function
  - ko.observable();
- 2 Way Binding
  - Both sides are updated with changes

Target Element Binding

Source Object Property

Taylor 110

Taylor 914ce

2 Way Binding

```
<span>Guitar model:</span>
<input data-bind="value: product.model"/>
<span>Sales price:</span>
<span data-bind="text: product.salePrice"></span>
```

Declarative Binding

```
product: {
    id: ko.observable(1001),
    model: ko.observable("Taylor 314ce"),
    salePrice: ko.observable(1199.95)
}
```

Source object

```
ko.applyBindings(data);
```

Bind Source to Target, & Vice Versa

DEMO

Observables

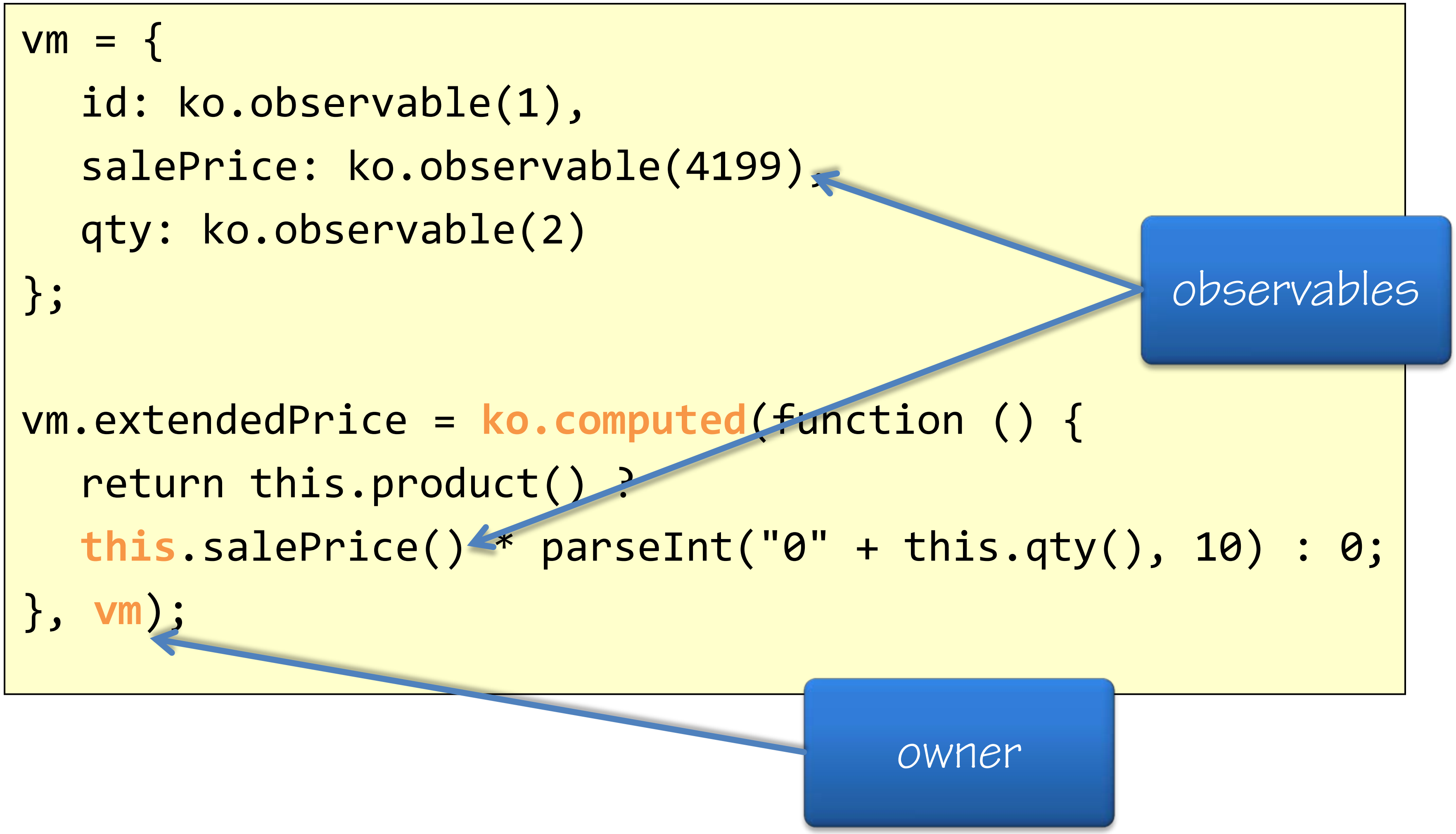
Outline

- Without Data Binding
- Observables
- Computed Observables
- ObservableArray
- Subscribing to Changes

Computed Members

- **Define a function to evaluate a value, and use it for binding**
  - Ex: Last Name, Photo Url, Currency Totals
- **When its observables change, it also notifies that changes occurred**
- **Manage “this”**
  - Pass in an owner for “this”, if needed
- **computed is formerly known as ~~dependentObservable~~**

Defining a Computed Member



```
vm = {
   id: ko.observable(1),
   salePrice: ko.observable(4199),
   qty: ko.observable(2)
};

vm.extendedPrice = ko.computed(function () {
   return this.product() ?
   this.salePrice() * parseInt("0" + this.qty(), 10) : 0;
}, vm);
```

DEMO

Computed

pluralsight
hardcore developer training

Defining a Computed Converter

- Computed members can define read and write behavior
- Great for custom converters

```
vm.extendedPrice = ko.computed({
   read: function () {
       // return an expression with observables
   },
   write: function (value) {
       // parse values and store in an observable
    },
   owner: //put your viewmodel here
});
```

read (required)

write

owner

DEMO

Computed Converters

Outline

- Without Data Binding
- Observables
- Computed Observables
- ObservableArray
- Subscribing to Changes

ObservableArray

- **Tracks which object are in the array, not their state**

- **Notify when items are**
  - Added
  - Removed

- **No notification when properties of item in collection change**
  - use ko.observable for those properties

Working with observableArray

```
var myViewModel = {
   salesPerson: ko.observable("John"),
   empNum: ko.observable(39811),
   products: ko.observableArray([
       { model: "Taylor 314CE", price: 1749, id=321 },
       { model: "Martin D40", price: 1899, id=479 }
   ])
};
```

Pre-populating

Operating on observableArray

```
<span data-bind="text: products().length"></span>
```

DEMO

ObservableArray

Observable Array Functions

- destroy
- destroyAll
- indexOf
- pop
- push
- remove
- removeAll
- reverse
- shift
- slice
- sort
- splice
- unshift

DEMO

ObservableArray Functions

Outline

- Without Data Binding
- Observables
- Computed Observables
- ObservableArray
- Subscribing to Changes

Subscribing to Changes

- Register to be notified when changes occur
- Similar to writing code in a property setter in .NET
- Useful when you need to take action when a property changes

```
// Whenever the selectedMake changes, reset the selectedModel

viewmodel.selectedMake.subscribe(function () {
      viewmodel.selectedModel(undefined);
}, viewmodel);
```

DEMO

Subscribing to Changes

Summary

- Without Data Binding
- Observables
- Computed Observables
- ObservableArray
- Subscribing to Changes