

### KnockoutJS

#### 1. Inline Component

##### A. Define

```
ko.components.register(component-name, {
  viewModel: fn(params),
  template: html string
});
```

##### B. Declare

##### i. Custom Element

```
<component-name params=""></component-name>
```

##### ii. Component Binding

```
<div data-bind="component: 'component-name'"></div>
<div data-bind="component: observable">
</div>
<div data-bind="component: {name: '', params: {}}"></div>
```

#### 2. External Dependencies Component

##### A. Define

```
ko.components.register(widget-name, {
  viewModel: { require: moduleid },
  template: { require: 'text!url' }
});
```

### KnockoutJS Defining ViewModel

#### 1. Constructor Function

```
function GenericViewModel(params) {
  // 'params' is an object whose key/value pairs
  // are the parameters
  // passed from the component binding or
  // custom element.
  this.someProperty = params.something;
}
ko.components.register('component-name', {
  viewModel: GenericViewModel,
  template: ...
});
```

#### 2. Shared Instance

```
var sharedViewModelInstance = { ... };
ko.components.register('component-name', {
  viewModel: sharedViewModelInstance,
  template: ...
});
```

#### 3. createViewModelFactory

```
ko.components.register('component-name', {
  viewModel: {
    createViewModel: function(params,
      componentInfo) {
      // - 'params' is an object whose key/value pairs
      // are the parameters
```

### KnockoutJS Defining ViewModel (cont)

```
// passed from the component binding or
// custom element
// - 'componentInfo.element' is the element the
// component is being
// injected into. When createViewModel is
// called, the template has
// already been injected into this element, but
// isn't yet bound.
// - 'componentInfo.templateNodes' is an array
// containing any DOM
// nodes that have been supplied to the
// component. See below.
// Return the desired view model instance, e.g.:
return new MyViewModel(params);
}
},
template: ...
});
DOM manipulation should be done in
custom bindings
4. AMD ViewModel
ko.components.register('component-name', {
  viewModel: { require: 'some/module/name' },
  template: ...
});
```