Bindings

A binding creates a link between two properties such that when one changes, the other one is updated to the new value automatically. Bindings can connect properties on the same object, or across two different objects. Unlike most other frameworks that include some sort of binding implementation, bindings in Ember.js can be used with any object, not just between views and models.

The easiest way to create a two-way binding is to use a computed alias, that specifies the path to another object.

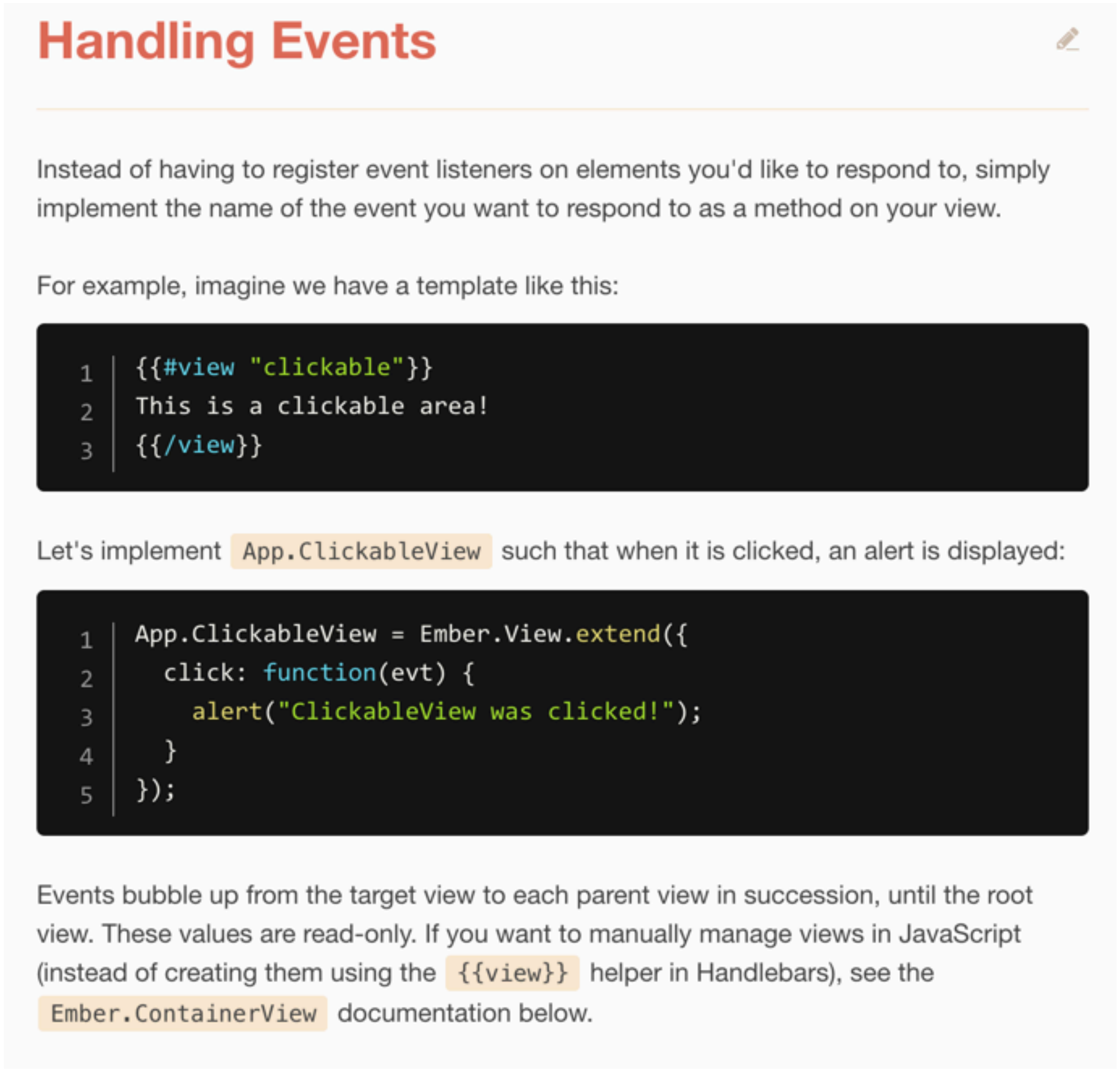
```
wife = Ember.Object.create({
  householdIncome: 80000
});

Husband = Ember.Object.extend({
  householdIncome: Ember.computed.alias('wife.householdIncome')
});

husband = Husband.create({
  wife: wife
});

husband.get('householdIncome'); // 80000

// Someone gets raise.
husband.set('householdIncome', 90000);
wife.get('householdIncome'); // 90000
```



Handling Events

Instead of having to register event listeners on elements you'd like to respond to, simply implement the name of the event you want to respond to as a method on your view.

For example, imagine we have a template like this:

```handlebars
{{#view "clickable"}}
This is a clickable area!
{{/view}}
```

Let's implement `App.ClickableView` such that when it is clicked, an alert is displayed:

```javascript
App.ClickableView = Ember.View.extend({
  click: function(evt) {
    alert("ClickableView was clicked!");
  }
});
```

Events bubble up from the target view to each parent view in succession, until the root view. These values are read-only. If you want to manually manage views in JavaScript (instead of creating them using the `{{view}}` helper in Handlebars), see the `Ember.ContainerView` documentation below.

Bindings

A binding creates a link between two properties such that when one changes, the other one is updated to the new value automatically. Bindings can connect properties on the same object, or across two different objects. Unlike most other frameworks that include some sort of binding implementation, bindings in Ember.js can be used with any object, not just between views and models.

The easiest way to create a two-way binding is to use a computed alias, that specifies the path to another object.

```
wife = Ember.Object.create({
  householdIncome: 80000
});

Husband = Ember.Object.extend({
  householdIncome: Ember.computed.alias('wife.householdIncome')
});

husband = Husband.create({
  wife: wife
});

husband.get('householdIncome'); // 80000

// Someone gets raise.
husband.set('householdIncome', 90000);
wife.get('householdIncome'); // 90000
```