# FAQ - FRB 语法变化

在 [FRB文档](http://docs.montagestudio.com/montagejs/documentup.com/montagejs/frb/)查看更多关于FRB介绍。

#### 如何监听对象属性值改变?

以前

myObject.addPropertyChangeListener("path", handler)

当属性path值改变之后会调用myObject.handleChange(notification)

现在

aMontageObject.addPathChangeListener("path", handler, opt\_methodName)  
  
// or  
  
Montage.addPathChangeListener.call(  
 myObject, "path", handler, opt\_methodName  
)

aMontageObject对象的原型链上需要包含Montage：Montage.isPrototypeOf(aMontageObject) === true。

当属性path值改变（path属性被重新赋值而不是path值对象内部改变）之后，下面列表中的方法按照顺序调用，参数包括newValue, path, 和 myObject（注意，只会调用一个方法）。

* handler[methodName]
* handler.handlePathChange
* handler

#### 如何监听对象属性值将要发生改变?

以前

myObject.addPropertyChangeListener("path", handler, true)

现在

aMontageObject.addPathChangeListener("path", handler, "handleMethodName", true)  
  
// or  
  
Montage.addPathChangeListener.call(  
 myObject, "path", handler, "handleMethodName", true  
)

#### 如何把一个对象的属性绑定到另外一个对象的属性，两个属性值始终相同?

以前

Object.defineBinding(myObject, "myProperty", {  
 boundObject: anotherObject,  
 boundObjectPropertyPath: "foo.bar"  
});

现在

aMontageObject.defineBinding("myProperty", {  
 "<->": "foo.bar",  
 source: anotherObject  
});  
  
// or  
  
var Bindings = require("montage/core/bindings").Bindings;  
Bindings.defineBinding(myObject, "myProperty", {  
 "<->": "foo.bar",  
 source: anotherObject  
});

#### 如何绑定一个对象的属性到另外一个对象的属性，但是当第一个对象属性值发生改变后第二个对象属性值不改变?

以前

Object.defineBinding(myObject, "myProperty", {  
 boundObject: anotherObject,  
 boundObjectPropertyPath: "foo.bar",  
 oneway: true  
});

现在

aMontageObject.defineBinding("myProperty", {  
 "<-": "foo.bar",  
 source: anotherObject  
});  
  
// or  
  
var Bindings = require("montage/core/bindings").Bindings;  
Bindings.defineBinding(myObject, "myProperty", {  
 "<-": "foo.bar",  
 source: anotherObject  
});

#### 如何监听数组添加或者移除元素?

aMontageObject.addRangeAtPathChangeListener(  
 "array", handler, "handleArrayRangeChange"  
);  
  
// or  
  
Montage.addRangeAtPathChangeListener(  
 myObject, "array", handler, "handleArrayRangeChange"  
);

handler.handleArrayRangeChange方法会传入三个参数： plus, minus, 和 index。

#### 如何当一个私有属性改变之后，触发公开属性改变?

以前

myObject.dispatchPropertyChange(  
 "affectedProperty",  
 "anotherAffectProperty",  
 function () {myObject.\_underlyingProperty = newValue}  
);

现在

myObject.dispatchBeforeOwnPropertyChange(  
 "affectedProperty", myObject.affectedProperty  
);  
  
myObject.dispatchBeforeOwnPropertyChange(  
 "anotherAffectedProperty", myObject.anotherAffectedProperty  
);  
  
myObject.\_underlyingProperty = newValue;  
  
myObject.dispatchOwnPropertyChange(  
 "affectedProperty", myObject.affectedProperty  
);  
  
myObject.dispatchOwnPropertyChange(  
 "anotherAffectedProperty", myObject.anotherAffectedProperty  
);

#### 如何让checkbox禁用有自动取消选择?

使用单向绑定<- checked 值为 checked && enabled， 这样就可以实现当checkbox禁用以后自动取消选择，启用以后又恢复之前状态

#### 如果实现 “全选” 或者 “全部不选” checkbox

绑定是双向的，checkbox状态包括：是否所有的checkbox全部选中，是否全部没有选。 也需要在checkbox点击之后全选或者全部不选。

checkboxes.every{checked} <-> allChecked  
checkboxes.every{!checked} <-> noneChecked

FRB支持绑定到一组数据的 "every" 或者 "some"。

#### 如果Checkbox可以被禁用呢？

checkbox.checked <- checked && enabled  
checkboxes.every{checked || !enabled} <-> allChecked  
checkboxes.every{!checked} <-> noneChecked

|| !enabled 表达式有两个目的。如果checkbox禁用之后，第一个绑定改变它为不选。如果没有|| !enabled表达式，会让allChecked为false,因为不应该让禁用的checkbox参与到allChecked判断中。 有|| !enabled表达式限制，checked || !enabled值就是true,这样allChecked就是正确的值true。

反方向上，当allChecked改变为true，如果没有|| !enabled表达，会让所有禁用的checkboxes改变为选中状态。 现在这样checked || !enabled表达式的值已经是true，checkbox就不会改变为选中状态。