Infusion Event System

Overview

The Infusion framework defines an event system which is used by many of its components.

Why did we construct a new event system when there are (at least) two event systems already in common use in a jQuery-enabled page? Browser events and their jQuerified equivalents are too tied to the DOM and its infrastructure. Considered in MVC terms, a jQuery event is more appropriately attached to the View layer of an application, whereas Fluid events are used to propagate pure Javascript function calls, and more appropriately attached to the Model layer. Using browser/jQuery events in this context would lead to inappropriate responsibilities for having to always determine a DOM node as the target /originator for an event, and to have a constrained event signature which must accept a browser event as the first argument.

Events Types

By default, most Fluid component events are multi-cast (that is the event is fired to all registered listeners), but some events may be one of the following two special types:

Type	Description
unicast	The event will fire to only one listener. If the implementer tries to attach multiple listeners to a unicast event, only the last registered listener will receive the event.
preventable	The event represents a "preventable" action. The listeners may each return a boolean value of `false`, representing both • that further listeners should fail to be queried, and • that the operation represented by the event should be cancelled. This is similar to the default semantics on browser events.

On This Page

See Also

- ChangeApplier
- Events for Component Users
- Events for Component Developers

Still need help?

Join the infusion-users mailing list and ask your questions there.

Event firers

The Fluid event system is operated by instances of an "event firer" which are created by a call to `fluid.event.getEventFirer()`:

```
var myFirer = fluid.event.getEventFirer(unicast, preventable);
```

Argument	Type	Description

`unicast` (optional)	`boolean`	If `true`, this event firer is a "unicast" event firer (see Event Types).
`preventable` (optional)	`boolean`	If `true`, this event firer represents a "preventable" action (see Event Types).

Using an event firer

Once an event firer is constructed, it can be called with the following methods:

Method	Arguments	Description
`addListener`	`listener: Function, namespace: String` [fluid: optional]	Registers the supplied listener with this firer. The listener is a function of a particular signature which is determined between the firer and listener of an event. The namespace parameter is an optional String which defines a key representing a particular "function" of the listener. At most one listener may be registered with a firer with a particular key. This is a similar system to that operated by the JQuery namespaced events system. For an event firer which is of type `unicast`, the namespace argument will be ignored and will default to a fixed value. The use of namespaces is highly recommended.
`removeListener`	`listener: String /Function`	Supplies either the same listener object which was previously supplied to `addListener`, or else the String representing its namespace key. The designated listener will be removed from the list of registered listeners for this fierer.
`fire`	(arbitrary)	Fires an event to all the registered listeners. They will each be invoked with the exact argument list which is supplied to `fireEvent` itself. If this is a `preventable` event, `fireEvent` may return `true` indicating that a listener has requested to prevent the effect represented by this event.

Events and options merging

In order to make it as easy as possible for Fluid component authors to define their event types and accept and manage listeners, Fluid event firers have a special status during the Fluid options merging process.

events in options

A component may declare as part of its `options` structure, a top-level structure named `events` whose keys correspond to event types that this component wishes to support, and whose values are either `null` or the string values `"unicast"` or `"preventable"` corresponding to the accepted arguments for `getEventFirer`. As part of the normal construction process of `fluid.initView()`, the top-level `that` object for the component will automatically have constructed a corresponding event firer object for each one of these events.

For example, the `events` section of the default options for the Reorderer component indicates that the Reorderer supports 6 events of the listed types, of which one, `onBeginMove` represents a "preventable" event - a listener may countermand the beginMove effect by returning `true` when the event is received. If the top-level `that` constructed by `fluid.initView()` is stored in a variable named `thatReorderer`, these events and firers will be accessible at its top level by making a call for example of the form `thatReorderer.onSelect.fire(item)`.

```
events: {
    onShowKeyboardDropWarning: null,
    onSelect: null,
    onBeginMove: "preventable",
    onMove: null,
    afterMove: null,
    onHover: null
}
```

listeners in options

Both as part of defaults, and also as supplied instantion options, a fluid component can accept a structure named `listeners`, whose keys are taken from the set listed in the `events` structure, and whose values are either single listeners or arrays of listeners. For example, the following structure as part of an options object:

```
listeners: {
    onMove: function(item, requestedPosition) {
        fluid.moduleLayout.updateLayout(item, requestedPosition.element,
                    requestedPosition.position, layout);
    }
}
```

indicates that the supplied listener should be registered for the event `onMove`.

Keys for listeners in `options` may also be qualified with namespaces following a period character `.` - this follows the jQuery convention for namespaced events.

For example,

```
listeners: {
    "onShowKeyboardDropWarning.setPosition": defaultOnShowKeyboardDropWarning
}
```

represents that the function stored in `defaultOnShowKeyboardDropWarning` should be attached as a listener to the event `onShowKeyboardDropWarning` under the namespace `setPosition`. The use of namespaces for event listeners is highly recommended.