Events for Component Developers

Overview

The Fluid framework defines a event system which is used by many of its components. For an overview, see Infusion Event System. This page provides specific information for developers who are creating Fluid components.

- Component developers must decide what events their component will fire, based on what it does, and what they think that component users would be interested in being notified of.
- Component developers declare their events in the defaults for the component.
- The component initialization process will instantiate firers for the declared events and attached them to the component's `that` object.
- Component developers must fire the events at the appropriate times.
- Component developers may add listeners to the firers, if desired.

The rest of this page discusses these points in more detail.

Declaring Events

Naming

The general convention for naming events is to use `on` and `after` prefixes for events that happen before and after certain things, such as `onBeginEdit` or `afterTransferComplete`.

On This Page

- Overview
- Declaring Events
    - Naming
    - Event Types
    - Declaring events
- Event firers
- Using an event firer
- Instantiating Event Firers in a Component
- Adding Listeners
- Firing Events

See Also

- Infusion Event System
- Events for Component Users

Still need help?

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

Event Types

Events may optionally be declared as one of two possible 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.

Declaring events

Component developers declare the events their component will fire using the `events` object in the defaults for their component, for example:

```
fluid.defaults("fluid.myComponentName", {
    events: {
        onBeginEdit: null;
        afterTransferComplete: null;
    }
});
```

The `events` object's keys correspond to the event types that this component wishes to support, and the values are either null or the string values "unicast" or "preventable."

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.

Instantiating Event Firers in a Component

A component's creator function must call `fluid.initView()` to initialize the component's view:

```
fluid.myComponent = function (container, options) {
    var that = fluid.initView("fluid.myComponent", container, options);
    ...
};
```

The `fluid.initView()` function automatically instantiates event firers for all of the events declared in the defaults, and attaches them to the returned `that` object.

Adding Listeners

After `fluid.initView()` has instantiated the event firers, the component itself may wish to attach listeners to events using the event firer's `addListener()` function, for example:

```
var bindEvents = function (that) {
    that.events.onBeginEdit.addListener(function() {
        ...
    });

    that.events.afterTransferComplete.addListener(transferCompleteHandlerFunc);
};
```

Firing Events

The component is responsible for firing the events at the right time, using the event firer's `fire()` function, for example:

```
var edit = function () {
    that.events.onBeginEdit.fire(args);
    ... (handle editing) ...
};

var transfer = function () {
    ... (handle transfer) ...
    that.events.afterTransferComplete.fire(args);
};
```

The arguments passed to `fire()` are arbitrary, and component developers are free to determine what information should be passed on to event listeners by the event firer.