Component Lifecycle

Every Fluid component has a standard lifecycle, various points of which are signalled by firing of standard framework events. Every component which has a grade of fluid.eventedComponent is able to receive and react to these events. The events, in the expected order of firing for a component, are as follows:

Event	Arguments	Lifecycle Point
onCreate	**`that`** {Object} - the component being constructed	Fired when component construction is complete - that is, all options have been merged for the component and all subcomponents (which were not marked with createOnEvent) have constructed.
onAttach	**`that`** {Object} - the component being attached **`name`** {String} - the member name of the component in the parent to which it is being attached **`parent`** {Object} - the parent component to which the component is being attached	Fired when the component is attached to the overall component tree of which it is a part.
onClear	**`that`** {Object} - the component being cleared **`name`** {String} - the member name of the component in the parent from which it is being cleared **`parent`** {Object} - the parent component from which the component is being detached	Fired when the component is detached from the component tree, as a preliminary to destroying it completely
onDestroy	**`that`** {Object} - the component being destroyed	Fired when the component is about to be destroyed. This will be a preliminary to beginning the destruction process for all its subcomponents.
afterDestroy	**`that`** {Object} - the component which has been destroyed	Fired after the component and its children have been completely destroyed. **NOTE** - at this point you may only safely access plain data members of the component such as `id` and `typeName`. ***Do not attempt to invoke any methods, fire any events, or resolve any IoC references from listeners to this event.***

Note that since JavaScript is a garbage-collected language, the component object reference and many of its members will hang around in memory during and after the destruction process, although it will as noted above be detached from its parent (via a call to `delete`) and similarly all subcomponent references will be recursively detached from their parents. The component author may schedule various actions to clean up any external resources (perhaps a jQuery widget, or a network connection) during the destruction process by adding listeners to the `onDestroy` event.

Every Fluid component is supplied with a standard method named `destroy` which is available after `onCreate` has fired. destroy takes no arguments and will initiate the destruction process for the component - that is, `onClear` followed by `onDestroy and afterDestroy.`

Note that this simplified lifecycle in Infusion 1.5 replaces the much more complex system which was available in Infusion 1.4 and before known as the initFunction system. Increased power in the IoC system to naturally schedule construction across the entire component tree makes the initFunction system unnecessary. This system is still available in Infusion 1.5 but will be withdrawn in a following release.