Subcomponents

Overview

In Fluid Infusion, any function may act as a **subcomponent** creator function. To say that a particular thing is a subcomponent does not say something about the thing, but rather about the context in which the creator function is invoked. A subcomponent, in particular, need not be a Component or "View" - it may simply be any old that.

Some subcomponents, however, are components - for example, the Undo subcomponent **is** a component. It calls the `fluid.initView` initialisation function on creation, since it is associated with a particular piece of DOM real estate in which it maintains a small (decorative) UI. On the other hand, a LayoutHandler, part of the configuration for a Reorderer, whilst it is instantiated as a subcomponent, is itself not a component. Since it is not specifically associated with a particular area of the UI, it enjoys a specialised instantiation contract with its parent component.

Note that nothing about a function which is invoked as a subcomponent constructor is intended to particular tie it to a particular parent component, or indeed to the Infusion framework as a whole - any function which could be invoked as a subcomponent constructor via `fluid.initSubcomponents` could just as easily be invoked as a free function.

A subcomponent encapsulates functionality that is intendend to be independent of the component that is using it (the parent component). By defining subcomponents separately and allowing parent components to identify them declaratively, subcomponents can be reused or interchanged.

The identification and configuration of subcomponents is carried out through the parent component's `options`. As part of the creation of the parent component, any necessary subcomponents are created according to the configuration information provided in the `options`.

On This Page

See Also

- Undo API
- Table of Contents API

Still need help?

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

Creation

The creator function of any component is responsible for creating subcomponents, both those required by the component and those requested by the implementor. Creation is accomplished using the `fluid.initSubcomponents()` functions:

```
fluid.initSubcomponent(that, className, args);
```

```
fluid.initSubcomponents(that, className, args);
```

Parameters

that

The first parameter is the parent component to which the subcomponent should be attached. Inside a component creator function, this will be the `that` object returned by `fluid.initView()`.

The `that` object is expected to contain the configuration information for the subcomponent in its `options` structure. They may have been specified in the defaults for the component, or passed in as options to the creator function, or both. See fluid:Configuration for more information about the `options` used to configure subcomponents.

className

The `className` parameter defines the category of subcomponent to instantiate, and maps to the name of the configuration options present in the `that.options` object. (See fluid:Configuration for more information). Users should consult the documentation for the individual subcomponent to determine its type. Implementors creating their own subcomponents must select a unique category for them.

args

The `args` parameter is an array of JavaScript objects that will be passed to the creator function of the subcomponent. The arguments required are defined by the subcomponent, and are not bound by the `container, options` convention used by regular components.

Example

In the example below, a parent component, `newComponent`, creates a subcomponent, `subComp1`:

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

    ...
    // subComp1 is of class "subComponent1" and its creator function
    // requires the container and events as parameters
    that.subComp1 = fluid.initSubcomponents(that,
                                            "subComponent1",
                                            [that.container, that.events]);

    ...
};

fluid.defaults("fluid.newComponent", {
    subComponent1: {
        type: "fluid.someSubComponent",
        options: {
            selectors: {
                part1: ".flc-part1"
            }
        }
    },
    ...
});
```

Configuration

Subcomponents can be configured in the defaults for the parent component. The key in the defaults is the classname of the subcomponent - this is what is passed as the second parameter to `initSubcomponent()`. The value is an object containing a `type` and `options`. The `type` is the fully namespaced subcomponent creator function name. The `options` are the subcomponent creator function's supported options.

Types of subcomponents

Several subcomponents are currently in use by Fluid components:

Subcomponent	Used in
Textfield Slider	UI Options
PagerBar	Pager
PageList	Pager
PreviousNext	Pager
Summary	Pager
PageSize	Pager
RangeAnnotator	Pager
BodyRenderer	Pager
ListLayoutHandler	List Reorderer
GridLayoutHandler	Grid Reorderer API
ModuleLayoutHandler	Layout Reorderer
Undo	Inline Edit
Table of Contents	UI Enhancer
Cookie Store	UI Enhancer
Temp Store	UI Enhancer