Component subcomponents

Subcomponents

Depending on how you've planned and or refactored your code, you may find yourself needing to break off chunks of code into their own components. We refer to these chunks as subcompenents (although in reality they are just components used by another component). In Flutter, the individual views are components. Flutter, also a component, makes use of the individual views as subcomponents.

In fluid.defaults, you specify a subcomponent using a name and an object containing the key-value pair of **type: "component name"** as shown below. In this case the subcomponent's name is "subComponent".

```
fluid.defaults("fluid.flutter", {
    selectors: {
        mainPanel: ".flutter-main-panel",
        settingsPanel: ".flutter-settings-panel"
    },
    styles: {
        hidden: "flutter-hidden"
    },

    subComponent: {
        type: "nameSpace.componentName"
    }
});
```

Initialization is done using fluid.initSubcomponent. This function takes three arguments:

1. the component you are attaching it too, which is represented by "that"
2. the name of the subcomponent as specified in fluid.defaults
3. an array containing the arguments for the subcomponent

```
that.subComponent = fluid.initSubcomponent(that, "subComponent", [arg1, arg2, ...]);
```

Subcomponents for Flutter

Try adding a subcomponent to Flutter. Download the fluid:Sample Code. Rename the file to FlutterInfused.js and place into the "flutter-infused" directory. To start we'll add the subcomponent options to the fluid.defaults function, on line 109. The initial structure of the fluid.defaults is shown below.

```
fluid.defaults("fluid.flutter", {

        selectors: {
            mainPanel: ".flutter-main-panel",
            settingsPanel: ".flutter-settings-panel",

            statusPanel: ".flutter-status-panel",
            friendsList: ".flutter-friends",
            tweetsList: ".flutter-tweets",

            allTabs: ".flutter-panel-tabs li",
            friendsTab: ".flutter-friends-tab",
            settingsTab: ".flutter-settings-tab",

            errorDialog: ".flutter-error-dialog"
        },

        styles: {
            hidden: "flutter-hidden",
            activeTab: "fl-activeTab"
        },

        events: {
            // View-related events.
            afterFriendSelected: null,
            onSaveSettings: null,

            // Model change events.
            onFriendsFetchSuccess: null,
            onFriendsFetchError: null,
            onTweetsFetchSuccess: null,
            onTweetsFetchError: null,
            onStatusSaveSuccess: null,
            onStatusSaveError: null
        }
    });
```

Now add the subcomponents friendsView, tweetsView, settingsVeiw and statusView.

```
fluid.defaults("fluid.flutter", {
        //subcomponent friendsView
        friendsView: {
            type: "fluid.flutter.friendsView"
        },

        //subcomponent tweetsView
        tweetsView: {
            type: "fluid.flutter.tweetsView"
        },

        //subcomponent settingsView
        settingsView: {
            type: "fluid.flutter.settingsView"
        },

        statusView: {
            type: "fluid.flutter.statusView"
        },

        selectors: {
            mainPanel: ".flutter-main-panel",
            settingsPanel: ".flutter-settings-panel",

            statusPanel: ".flutter-status-panel",
            friendsList: ".flutter-friends",
            tweetsList: ".flutter-tweets",

            allTabs: ".flutter-panel-tabs li",
            friendsTab: ".flutter-friends-tab",
            settingsTab: ".flutter-settings-tab",

            errorDialog: ".flutter-error-dialog"
        },

        styles: {
            hidden: "flutter-hidden",
            activeTab: "fl-activeTab"
        },

        events: {
            // View-related events.
            afterFriendSelected: null,
            onSaveSettings: null,

            // Model change events.
            onFriendsFetchSuccess: null,
            onFriendsFetchError: null,
            onTweetsFetchSuccess: null,
            onTweetsFetchError: null,
            onStatusSaveSuccess: null,
            onStatusSaveError: null
        }
    });
```

All of the subcomponents have been added to fluid.defaults and are ready to be initialized. On line 33 we'll add in the initialization code.

```
// Create the Friends View, responsible for showing and selecting the list of  friends.
that.friendsView = fluid.initSubComponent(that, "friendsView",
    [that.locate("friendsList"), that.twitter, that.events, fluid.COMPONENT_OPTIONS]);

// Instantiate the Tweets View, which displays the list of tweets
that.tweetsView = fluid.initSubComponent(that, "tweetsView",
    [that.locate("tweetsList"), that.twitter, that.events, fluid.COMPONENT_OPTIONS]);

// The Settings View controls the panel allowing users to edit their username and password for Twitter.
that.settingsView = fluid.initSubComponent(that, "settingsView",
    [that.locate("settingsPanel"), that.twitter, that.events, fluid.COMPONENT_OPTIONS]);

// The Status View shows the user's icon, name, and allows them to update their status.
that.statusView = fluid.initSubComponent(that, "statusView",
    [that.locate("statusPanel"), that.twitter, that.events, fluid.COMPONENT_OPTIONS]);
```

Their are two things to note in the initialization code above:

1. the use of that.locate. Recall from the #selectors section above that that.locate is used to find DOM elements.
2. fluid.Components
    - This tells fluid.initSubComponent to check if any options were specified for the subcomponent when the parent component was initialized. In this case the Fluid Framework will merge in those options. For example:

```
fluid.flutter(container, {friendsView: { selectors: { friends: ".liClass"}}});
```

When you are done, the finished code will look like this

Custom Options

There may be times when there are options specific to a component you are writing which don't fit into any of the options already mentioned. fluid.defaults allows you to add custom options.

Here is an example of how you would specify custom options to a component called nameSpace.custom. This declaration follows the same pattern that the other options.

```
fluid.defaults("nameSpace.custom", {

    customOptions: {
        meaningfulName: value
    }

});
```

To access this option use:

```
var getOptionValue = that.options.customOptions.meaningfulName;
```