SF.UI

***** This class will not be supported in 2015.***

*All functionalities of sf.ui class are more improved, integrating with CAPH. Therefore sf.ui class is not supported since 2015 Smart TV. To use functions of sf.ui class, refer to **here**.*

This class defines the UI core functions.

> Add the following line for sf.ui class into a html file your own :
>
> <script type="text/javascript" src="$MANAGER_WIDGET/Common/af/2.0.0/loader.js"></script>

> You can declare sf.ui class like this :
>
> ex) var ui = sf.ui;

Contents

Methods

addSelector

Description	
This function adds the widget selector to jQuery selectors.	
Parameters	■name - String - Selector of the same name passed to the sf.ui.bridge function.
Return	■Void
Emulator Support	N
SDK Constraint	None
Example	
sf.ui.addSelector("mySuperWidget");	

bridge

Description
This function binds the widget class with jQuery plugin function.

Parameters	■name - String - jQuery plugin function name to bind the widget class ■widget - Function - The Widget class constructor
Return	■Void
Emulator Support	N
SDK Constraint	None
Example	
sf.ui.bridge("myWidget", MyWidget); // bridge jQuery plugin function to widget	

widgetFactory

Description	
This function is the Widget factory function.	
Parameters	■base - Object - The Base widget constructor - [default : sf.ui.Widget] ■extensions - Function - Literal object with fields and functions to be added to the new widget (fields and functions will extend or override the base widget prototype).
Return	■Function - The new widget class
Emulator Support	N
SDK Constraint	None
Example	

[Example 1]

```
//Define new widget with sf.ui.Widget as its base.

MyWidget = sf.ui.widgetFactory({
   foo: "foofoo", // custom field
   templates: {
      item: "<p>${name}</p>"
   },
   _create: function () {
      this.view.item = $.tmpl(this.templates.item, {
         name: this.foo
      });
      this.element.append(this.view.item);
   },
   _destroy: function () {
      this.view.item.remove();
   },
   widget: function () {
      return this.view.item;
   }
});
// make a bridge
sf.ui.bridge("myWidget", MyWidget);
// define selector
sf.ui.addSelector("myWidget");
```

[Example 2]

Define new widget with MyWidget as its base.

```
MySuperWidget = sf.ui.widgetFactory(MyWidget, {
   options: {
      counter: 0
   },
   templates: {
      item: "<p>${name} was initialized</p>",
      counter: "<span></span>"
   },
   _create: function () {
      this._super();
      this.view.counter = $(this.templates.counter);
      this.view.item.append(this.view.counter);
   },
   _init: function () {
      this.options.counter += 1;
      this.view.counter.text(" " + this.options.counter + " times");
   },
   _destroy: function () {
      this.view.counter.remove();
      this._super();
   },
   _setOption: function (name, value) {
      if (name !== "counter") {
         this._super(name, value);
      }
   }
});
// make a bridge
sf.ui.bridge("mySuperWidget", MySuperWidget);
// define selector
sf.ui.addSelector("mySuperWidget"); |
```