JavaScript API Functions

This page lists static functions exposed by the Structure API.

window.almworks.structure.api.subClass(className, superclass, prototype)

Creates a subclass of a specific class. Returns a constructor function that will create the instances of the class.

This function provides light-weight polymorphism for the purposes of extending Structure's classes.

Parameters

`className`	*string*	Class name as string (optional, used for friendly instance names in debugger)
`superclass`	*Object*	Superclass reference
`prototype`	*Object*	Subclass prototype

The returned value – class constructor – takes a single optional `options` parameter.

The prototype may contain a special `init()` initializer method, which is called when an instance is being constructed. Superclass' `init()` method is called before subclass' method. Options that were passed to the constructor are passed through to the initializer.

Example

```
var MyClass = window.almworks.structure.api.subClass('MyClass', BaseClass, {
  init: function(options) {
   ...
  },

  someMethod: function() {
   ...
  }
});

var options = { ... };
var instance = new MyClass(options);
```

window.almworks.structure.api.registerColumnType(type, key)

Registers a new column type. If you're extending Structure by adding a new type of column to the grid, the type must be registered from your additional JavaScript web resource.

Column types are identified by a unique key, which is recorded in the view specification, along with the type-specific parameters and column name.

Parameters

`type`	*Object*	A `ColumnType` instance, implementing a specific column type – see ColumnType Class
`key`	*string*	Column type key (can also be array of strings if the type can handle multiple variations of a column specification)

Example

```
window.almworks.structure.api.registerColumnType(new MyColumnType(), 'com.acme.structure.awesome-column');
```

> We recommend using a unique key that has low chance of conflicting with column types provided by other, independent developers. A good approach is to have Java-like package notation for the keys.

window.almworks.structure.api.registerColumnGroup(options)

Registers a new column type group. Column groups are used in the "Add Column" panel to group column configuration presets, provided by column types.

Parameters

`options.groupKey`	*string*	Group key. The same key should be returned by all `ColumnConfigurator`'s `getGroupKey()` method for all column types which need to appear in this particular group.
`options.title`	*string*	Group title
`options.order`	*number*	Group order is used to sort groups. Order value for groups provided by Structure are 100, 200, 300 and so on.

Example

```
window.almworks.structure.api.registerColumnGroup({
  groupKey: 'com.acme.structure.colgroup', title: 'Acme Columns', order: 50
});
```

window.almworks.structure.api.registerItemDetailsProvider(itemType, ProviderClass)

Registers item details support for the given item type.

Parameters

`itemType`	*string*	Item type for which details are registered
`ProviderClass`	*Object*	Subclass of ItemDetailsProvider that defines details behavior

Example

```
var api = window.almworks.structure.api;
var MilestoneDetails = api.subClass('MilestoneDetails', api.ItemDetailsProvider, {
  ...
});

api.registerItemDetailsProvider('com.acme.my-plugin:type-milestone', MilestoneDetails);
```