JSAPI

Rave JavaScript API

Rave Core JS
** GOALS

- rave core should have clear dependency tree; it should have no external dependencies outside of underscorejs.
- rave core should be separate from the portal; it should be lightweight, built to provide rave's core functionality of managing and rendering sets of widgets and no extra overhead related to the portal. *rave core should be extensible. As in implementer you should NEVER need to overlay a file (rave_ajax.js is the only exception) in order to get required functionality. You should be able to build a portal or website around rave core entirely through its api / extending its objects.

File: rave_ajax.js

namespace: rave.ajax

dependencies: jQuery

description:

Wraps jQuery's $.ajax function, isolating rave core's dependency on jQuery to this one file. Any implementer who wishes to use another ajax library will overlay rave_ajax.js, and wrap their ajax library of choice so that it maches the api of $.ajax.

File: rave_api

namespace: rave.api

dependencies: rave.ajax

description:

Defines functions to access rave's rest & rpc apis. This file manages all ajax interaction with the server. It should depend only on the rave.ajax namespace. It should provide callbacks for any interaction that needs them, and should have know knowledge of further results(i.e. if errors should be displayed to a user on a failed call, it is on the calling library to implement - no ui or core interaction should be invoked from rave_api. Besides factoring out dependencies there have been no major changes to the rave.api spec.

File: rave_core.js

namespace: rave

dependencies: rave.api, rave.RegionWidget

description:

Defines rave's core functionality. Manages registration of widgets, widget providers, views (agnostic).

rave.registerProvider(name, provider)

- name String the name of the provider, such as 'open social' or 'wookie'. This should correspond to the string provided by a widget definition's TYPE field (case insensitive)
- provider Object widget provider object, implementing those methods expected by the rave.RegionWidget interface.
- returns Object returns the provider object

rave.getProvider(name)

- name String the name of the provider (case insensitive)
- returns Object widget provider object

rave.registerWidget([regionId], definition)

- *deprecated* [regionId] Number or String the regionId of the widget. This parameter is not used and is deprecated. Currently supported until the rave rendering is updated.
- definition Object the widget definition as provided by rave's RegionWidgetRenderer class
- returns Object rave.RegionWidget instance

rave.getWidget(regionWidgetId)

-regionWidgetId String

-returns rave.RegionWidget instance

rave.getWidgets()

-returns Array of rave.RegionWidget instances for the registered widgets

rave.unregisterWidget(regionWidget)

- regionWidget String the regionWidgetId of the widget to unregister OR Object the rave.RegionWidget instance

rave.registerView(name, view)

- name String key to register the view under (case insensitive)
- view Object an agnostic view object implementing those methods expected by the rave View specification OR Function constructor for a view object implementing those methods expected by the rave View specification

rave.getView(name)

- name String key view was registered under (case insensitive)
- returns Object registered view object OR Function registered view constructor

rave.renderView(name, [args...])

If the registered view is a constructor function, first instantiates a new instances of the view. Then invokes the render function of the view, passing it any remaining arguments.

- name String key view was registered under
- args * any remaining args are passed to the registered view's render function
- returns Object the rendered view instance. view._uid will be defined on the view - this is a unique identifying string that can be used to retrieve and destroy the rendered view.

rave.getRenderedView(_uid)

- _uid String unique identifier string as defined on the view._uid that is returned from rave.renderView
- returns Object rendered view instance

rave.destroyView(_uid, [args…])

invokes the destory() method on the view object, passing it any remaining arguments.

- _uid String unique identifier string as defined on the view._uid that is returned from rave.renderView
- args * any remaining args are passed to the registered view's destroy function

rave.registerOnInitHandler(handler)

registers a function that will be invoked after rave.init() has been called. onInit handlers will be called in the order they were registered

- handler Function function to invoke

rave.init()

function to be called once all rave resources and extensions are loaded. Kicks off initializing of providers, widget objects, and any registered onInit handlers

rave.log([args…])

utility function wrapping console.log that can safely be called without throwing an error in any browser environment.

- args * arguments that will be passed to console.log if it exists

rave.getManagedHub()

Instantiates (if needed) and returns an OpenAjax.hub.ManagedHub instance

- returns Object an OpenAjax.hub.ManagedHub instance

File: rave_widget

namespace: rave.RegionWidget

dependencies: rave.api

description:

Defines rave.RegionWidget as an abstract class that provides a common interface & functionality for any registered rave widget.

rave.RegionWidget(definition) constructor for abstract RegionWidget objects. All attributes from the definition are placed on the instance.

Ex: var widget = new rave.RegionWidget({type: 'opensocial', name: 'myWidget'};

widget.name == 'myWidget' //true

In general you will not call this constructor directly - instead you will use rave.registerWidget() to create new RegionWidgets and use rave.getWidget() / rave.getWidgets() to access the RegionWidget instances.

- definition Object the widget definition as provided by rave's RegionWidgetRenderer class
- returns Object rave.RegionWidget object instance

rave.RegionWidget.extend(mixin)

convenience function to extend or override RegionWidget's prototype, allowing implementers to add custom functionality.

- mixin Object key / value pairs to extend the RegionWidget prototype

rave.RegionWidget.defaultView

property that dictates the default view widgets will be rendered into

- String view name, initializes to 'default'

rave.RegionWidget.defaultWidth

property that dictates the default width of a rendered widget iframe

- Int width in pixels, defaults to 320

rave.RegionWidget.defaultHeight

property that dictates the default height of a rendered widget iframe

- Int height in pixels, defaults to 200

rave.RegionWidget.prototype.render(el, opts)

Renders a region widget instance into a dom element. Invokes the widget providers renderWidget method.

- el DomElement the DOM element into which the widget's iframe will be injected and rendered OR

String if a string is provided as the element, rave will look for a registered view by that string. It will render that view and inject the widget into the dom element returned by the view's getWidgetSite() method

- opts Object options object that is passed to the registered widget provider's renderWidget method
- returns Object this. Returns the regionWidget instance for chaining.

rave.RegionWidget.renderError(el, error)

This method should be invoked by widget providers if there is an error in rendering the widget.

- el HTMLElement the DOM element the widget was to be inserted into
- error String the text if the error message

rave.RegionWidget.prototype.close(el, opts)

Closes the region widget and removes the region widget from the page (persisted back to rave). If the widget was rendered within a view, destroys that view. Invokes the widget providers closeWidget function.

- opts * options object that is passed to the widget provider's closeWidget function

rave.RegionWidget.prototype.show()

Changes the widget's collapsed state and persists to the rave api. Does not take any ui actions - it is on the implementer to bind ui interactions and use this method to persist show / hide state.

rave.RegionWidget.prototype.hide()

Changes the widget's collapsed state and persists to the rave api. Does not take any ui actions - it is on the implementer to bind ui interactions and use this method to persist show / hide state.

rave.RegionWidget.prototype.moveToPage(toPageId, [cb])

Changes the widget's pageId and and persists to the rave api. Does not take any ui actions - it is on the implementer to bind ui interactions and use this method to persist widget location state.

- toPageId String or Int id of page that the widget is being moved to
- cb Function callback function to be invoked after persist is completed

rave.RegionWidget.prototype.moveToRegion(fromRegionId, toRegionId, toIndex)

Changes the widget's region and index and and persists to the rave api. Does not take any ui actions - it is on the implementer to bind ui interactions and use this method to persist widget location state.

-fromRegionId String or Int id of regionId the widget is being moved from

-toRegionId String or Int id of regionId the widget is being moved to

-toIndex String or Int id of index within the region the widget is being moved to

rave.RegionWidget.prototype.savePreferences(updatedPrefs)

Overwrites the widget's userPrefs object and persists to the rave api. Does not take any ui actions - it is on the implementer to bind ui interactions and use this method to persist user prefs state.

File: rave_opensocial, rave_wookie

namespace: n/a

dependencies: rave, opensocial & wookie implementations, respectively

description:

These files provide implementations of the abstract rave.RegionWidget interface for open social and wookie widgets. They do not attach anything to the rave namspace, but call rave.registerProvider directly.

Specifications -

Rave Provider

The object handed to rave.registerProvider must conform to the following specification or there will be errors

usage: rave.registerProvider('providerName', provider);

requirements:

provider.init()

Should run any setup needed to implement the provider. This method is invoked by rave.init() and will run before any rave.RegionWidget instances are instantiated.

provider.initWidget(widget)

Should do any work to preload caches or prepare for widget. This method does NOT render the widget, but does any work that can happen early.

-widgetDefinition Object rave.RegionWidget instance.

provider.renderWidget(widget, el, [opts])

Provider-specific implementation needed to render a widget into a dom element

- widget Object rave.RegionWidget instance.
- el DomElement
- opts Object bag of options that are passed from rave.renderWidget()

provider.closeWidget(widget, [opts])

Provider-specific implementation needed to close a widget

- widget Object rave.RegionWidget instance.
- opts Object bag of options that are passed from rave.closeWidget()

provider.setDefaultGadgetSize(width, height)

Provider specific implementation needed to set default size of widget

- width Int width in pixels
- height Int height in pixels

provider.setDefaultView(view)

Provider specific implementation for setting the default view a widget should be rendered into

- view String view name

Rave View

The argument handed to rave.registerView. Rave is completely agnostic of any dependencies, libraries, mv* frameworks, etc. It simply expects the view to be either…

an object that implements the following methods, or…

a constructor function that instatiates objects which implement the following methods

usage: rave.registerView('myView', {render: function(){...}, getWidgetSite: function(){...}, destroy: function(){...});

requirements:

view.render([args…])

Renders the view into the ui. Can take any number & type of arguments, which are simply passed from rave.renderView

-returns Object the render function MUST return itself (i.e. return this😉

view.getWidgetSite()

OPTIONAL. This method is required only if it will ever be used for rendering a widget. Meaning either a call from widget.render('viewName'), or if it will be exposed to opensocial open views spec.

- returns DomElement a raw DomElement (not a query object) which the widget iframe will be inserted into

view.destroy([args…])

Performs any teardown necessary to unrender the view. Can take any number & type of arguments, which are simply passed from rave.destoryView().