Dialogs Factory

AlertifyJS not only provides a replacement for default browser dialogs, it makes it super easy to create your own!

Looking for a commercial license ? Keep your source code proprietary and Buy a Commercial License Today!

Default usage

To create a new dialog, you use the alertify.dialog API. which takes a dialog name followed by a factory function. This factory function should return an object with the different options to create the dialog.

```
/**
 * Dialogs factory
 *
 * @name      {string}   Dialog name.
 * @Factory   {Function} Dialog factory function.
 * @transient {Boolean}  Indicates whether to create a singleton or transient dialog.
 * @base      {String}   The name of an existing dialog to inherit from.
 *
 * alertify.dialog(name, Factory, transient, base)
 *
 */

if(!alertify.myAlert){
  //define a new dialog
  alertify.dialog('myAlert',function factory(){
    return{
      main:function(message){
        this.message = message;
      },
      setup:function(){
          return {
            buttons:[{text: "cool!", key:27/*Esc*/}],
            focus: { element:0 }
          };
      },
      prepare:function(){
        this.setContent(this.message);
      }
  }});
}
//launch it.
```

Passing true to the transient (none singleton) parameter tells AlertifyJS to create a new dialog instance each time the dialog is invoked via alertify.{dialogName}. The last parameter 'base' enables the creation of new dialogs based on existing ones.

```
// Extend existing 'alert' dialog
if(!alertify.errorAlert){
  //define a new errorAlert base on alert
  alertify.dialog('errorAlert',function factory(){
    return{
            build:function(){
                var errorHeader = '<span class="fa fa-times-circle fa-2x" '
                +    'style="vertical-align:middle;color:#e10000;">'
                + '</span> Application Error';
                this.setHeader(errorHeader);
            }
        };
    },true,'alert');
}
//launch it.
// since this was transient, we can launch another instance at the same time.
alertify
    .errorAlert("This is a none singleton modal error alert! <br/><br/><br/>" +
        "To show another error alert: " +
      "<a href='javascript:alertify.errorAlert(\"Another error\");'> Click here </a>");
```

Run

Factory Function

The factory function should return an object with the different options to tell **AlertifyJS** how the dialog should behave when created.

```
function factory(){
    return {
        // The dialog startup function
        // This will be called each time the dialog is invoked
        // For example: alertify.myDialog( data );
        main:function(params){
                // manipulate parameters and set options
                this.setting('myProp', data);
        },
        // The dialog setup function
        // This should return the dialog setup object ( buttons, focus and options overrides ).
        setup:function(){
            return {
                /* buttons collection */
                buttons:[
```

```
            /*bind a keyboard key to the button */
            key: 27,

            /* indicate if closing the dialog should trigger this button action */
            invokeOnClose: true,

            /* custom button class name  */
            className: alertify.defaults.theme.ok,

            /* custom button attributes  */
            attrs:{attribute:'value'},

            /* Defines the button scope, either primary (default) or auxiliary */
            scope:'auxiliary',

            /* The will conatin the button DOMElement once buttons are created */
            element:undefined
        }
    ],

    /* default focus */
    focus:{
        /* the element to receive default focus, has differnt meaning based on value type:
            number:      action button index.
            string:      querySelector to select from dialog body contents.
            function:    when invoked, should return the focus element.
            DOMElement: the focus element.
            object:      an object that implements .focus() and .select() functions.
        */
        element: 0,

        /* indicates if the element should be selected on focus or not*/
        select: false

    },
    /* dialog options, these override the defaults */
    options: {
        title: ...,
        modal: ...,
        basic:...,
        frameless:...,
        pinned: ...,
        movable: ...,
        moveBounded:...,
        resizable: ...,
        autoReset: ...,
        closable: ...,
        closableByDimmer: ...,
        maximizable: ...,
        startMaximized: ...,
        pinnable: ...,
        transition: ...,
        padding:...,
        overflow:...,
        onshow:...,
        onclose:...,
        onfocus:...,
        onmove:...,
        onmoved:...,
        onresize:...,
        onresized:...,
        onmaximize:...,
        onmaximized:...,
        onrestore:...,
```

```
// This will be called once the dialog DOM has been created, just before its added to the document.
// Its invoked only once.
build:function(){

   // Do custom DOM manipulation here, accessible via this.elements

   // this.elements.root          ==> Root div
   // this.elements.dimmer         ==> Modal dimmer div
   // this.elements.modal          ==> Modal div (dialog wrapper)
   // this.elements.dialog         ==> Dialog div
   // this.elements.reset          ==> Array containing the tab reset anchor links
   // this.elements.reset[0]       ==> First reset element (button).
   // this.elements.reset[1]       ==> Second reset element (button).
   // this.elements.header         ==> Dialog header div
   // this.elements.body           ==> Dialog body div
   // this.elements.content        ==> Dialog body content div
   // this.elements.footer         ==> Dialog footer div
   // this.elements.resizeHandle   ==> Dialog resize handle div

   // Dialog commands (Pin/Maximize/Close)
   // this.elements.commands           ==> Object containing  dialog command buttons references
   // this.elements.commands.container ==> Root commands div
   // this.elements.commands.pin       ==> Pin command button
   // this.elements.commands.maximize  ==> Maximize command button
   // this.elements.commands.close     ==> Close command button

   // Dialog action buttons (Ok, cancel ... etc)
   // this.elements.buttons              ==>  Object containing  dialog action buttons references
   // this.elements.buttons.primary        ==>  Primary buttons div
   // this.elements.buttons.auxiliary      ==>  Auxiliary buttons div

   // Each created button will be saved with the button definition inside buttons collection
   // this.__internal.buttons[x].element

},
// This will be called each time the dialog is shown
prepare:function(){
    // Do stuff that should be done every time the dialog is shown.
},
// This will be called each time an action button is clicked.
callback:function(closeEvent){
   //The closeEvent has the following properties
   //
   // index: The index of the button triggering the event.
   // button: The button definition object.
   // cancel: When set true, prevent the dialog from closing.
}
// To make use of AlertifyJS settings API, group your custom settings into a settings object.
settings:{
       myProp:'value'
},
// AlertifyJS will invoke this each time a settings value gets updated.
settingUpdated:function(key, oldValue, newValue){
       // Use this to respond to specific setting updates.
       switch(key){
          case 'myProp':
             // Do something when 'myProp' changes
             break;
      }
},
// listen to internal dialog events.
hooks:{
   // triggered when the dialog is shown, this is seperate from user defined onshow
   onshow: function(){
   },
```

```
            // IMPORTANT: This will not be triggered for dialog custom settings updates ( use settingUpdated instead).
            onupdate: function(){
            }
        }
    }
}
```

Tutorials

Report an issue | View on GitHub | Follow Me on GitHub and Twitter

Created by Mohammad Younes | Licensed under GPLv3
