



Alloy UI with Liferay – Part 1

By Mohib Mansuri



About Author

Mohib Mansuri

Liferay | Magento | Drupal

- Over 3+ years of hands on experience in Liferay Portal and theme development, and Open Source technologies like Drupal, Magento, and Joomla.
- Successfully executed 7+ trainings on Liferay technology.
- Creating and using his own initiative
- Very good expertise on creating a responsive theme



Table of Contents

I.	Installation of Alloy UI	3
II.	Sample Web App for Auto Complete	5
III.	Alloy UI with Liferay	7
	Theme	7
	Layouts	8
	Portlet	9
IV.	Sample Alloy UI Portlet in Liferay	10
V.	Components of Alloy UI	11
	Forms related components	11
	Image related components	17
	Audio-Video related components.....	20
	Webpage and Layout related components	21



Installation of Alloy UI

AlloyUI is the UI framework for Liferay Portal, and covers 3 main areas of the front-end stack: HTML, CSS, JavaScript.

Download library of Alloy UI using below link

<http://cdn.alloyui.com/downloads/alloy-1.7.0.zip>

Or

<https://github.com/liferay/alloy-ui/archive/master.zip>

After download, unzip alloy-1.7.0.zip or alloy-ui-master.zip into your web application source code. And add Alloy UI javascript and css files into your HTML page.

Javascript file

```
<script src=" alloy-1.7.0/build/au/au.js" type="text/javascript"></script>
```

This javascript that allows you to load all alloyUI component's javascript files into HTML page.



CSS file

```
<link rel="stylesheet" href=" alloy-1.7.0/build/au i - ski n- cl ass i c/ css/ au i - ski n- cl ass i c- al l - mi n. css" type="text/ css" />
```

This CSS file that allows you to load all alloyUI component's CSS files into HTML page.

If you don't want to download than first load the seed file into your HTML source code.

Copy and paste below script in your HTML page.

```
<scri pt src="http: //cdn. al loyui . com/1. 7. 0/au i /au i - mi n. j s"></scri pt>
```

This is a small JavaScript that allows you to load any AlloyUI component like javascripts, css and images on your page.

Let's create an AUI instance and do some simple stuff, like auto complete textbox with appropriate values.

Then let's run auto complete textbox!



Sample Web App for Auto Complete

Initialize AlloyUI and load a module, e.g., Auto Complete.

Write below code in your HTML page

```
<div id="myAutoComplete"></div>

<script type="text/javascript" charset="utf-8">

AUI().use('aui-autocomplete', function(A) {

    var states = [

        ['AL', 'Alabama', 'The Heart of Dixie'],

        ['CA', 'California', 'The Golden State'],

        ['DE', 'Delaware', 'The First State'],

        ['DC', 'District of Columbia', 'The Nation's Capital'],

    ];

    window.AC = new A.AutoComplete(

        {

            dataSource: states,

            schema: {

                resultFields: ['key', 'name', 'description']

            },

            matchKey: 'name',

            delimChar: ',',

            contentBox: '#myAutoComplete'

        }

    );

    AC.render();

});

</script>
```

Output of “Auto Complete” component.

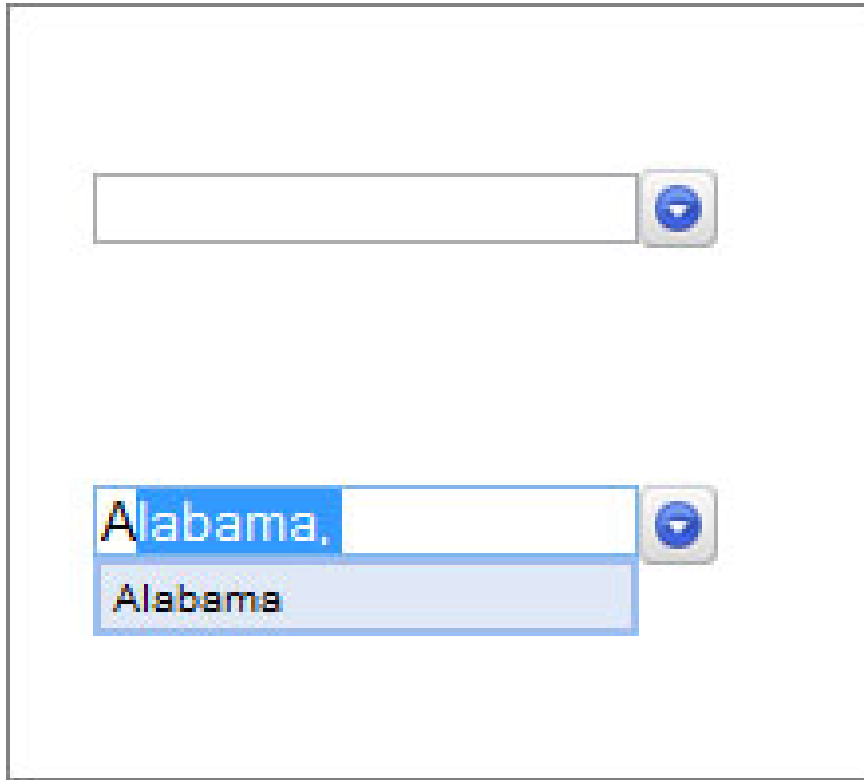


Figure 1 : Output of “Auto Complete” component.

This component is used to provide suggestions while users type into the field.



Alloy UI with Liferay

AlloyUI is a UI framework that helps you build dynamic and flexible web applications in liferay. It is a JavaScript library, a CSS framework, a set of HTML technique and a taglib library that controls patterns from the best libraries to allow developers across multi-skilled teams to deliver rich and dynamic applications like portlets, themes and layouts.

Alloy UI is an In-Built library of liferay, All AlloyUI component like javascripts, css and images are pre-loaded in liferay portal.

Here we display how use AlloyUI with Theme, Layout and Portlet

Theme

In liferay theme, when we use to define any JavaScript globally, than we define it in main.js file of theme. For e.g.:

```
AUI().use(  
    'aui-viewport'  
);
```

It is used to provide a cross-browser technique of adapting web design to display size.



Viewport allows you to modify your CSS for the four most frequently used page widths: 960px, 720px, 480px, and 320px for desktops, tablets, smart phones and mobile phones.

When you write above code in main.js file and deploy theme, it adds a few classes to the HTML element depending on the width of the window.

```
<html class="aui-view-gt320 aui-view-gt480 aui-view-gt720 aui-view-gt960 aui-view-960">
```

Based on these classes, you may create selectors which match some devices only.

Layouts

In Liferay, Layout is used to set position and create drag zone for portlet in page. Alloy UI allow us to define width for particular column in percentage (%) value. For e.g.:

```
<div class="aui-w20 portlet-column" id="column-1">  
    $processor.processColumn("column-1", "portlet-column-content")  
</div>
```

Here “aui-w20” is used to set 20% width of column. It will automatically create CSS for each column.



Portlet

In liferay portlet, Alloy UI is used to create professional user interface. It contains custom form tags. Like text box, lable, radio button etc.

Alloy UI provides a graphic user interface method for constructing forms, form validations, pagination, tool tip, tree view etc.

In Alloy UI, Liferay has customized form elements into Java Server Pages Standard Tag Library (JSTL) tags.

The form elements start with aui, like in the following piece of code.

```
<aui:form action="<%= editArticle ActionURL %>" enctype="multipart /form-  
data"method="post" name= "fml "></aui:form>
```

These aui tags are defined in liferay-aui.tld.

They are interpreted by classes in the com.liferay.taglib.aui package.

For using AlloyUI in liferay portlet, we need to define Taglib for aui in jsp page

```
<%@ taglib uri="http://alloy.liferay.com/tld/aui" prefix="aui"%>
```

Paste this line in your jsp. Now you can use aui tags in your jsps.

Here the button tag and button is shown with the syntax of if it.

```
<aui:button name="re moveArticleLocaleButton" onClick='<%= renderResponse.getNamespace() +  
"re moveArticleLocale();" %>' type="button" value="re move" />
```

Sample Alloy UI Portlet in Liferay

Here we create a simple liferay portlet, which contains AlloyUI form.

Step1: Create simple jsp portlet in liferay

Step2: Edit view.jsp of portlet and write below code in jsp page

```
<%@ taglib uri="http://alloy.liferay.com/tld/au" prefix="au"%>

<au:form action="" method="post">

    <au:input name="c_name" type="text" value="" label="Company Name" />

    <au:input name="c_phone" type="text" value="" label="Phone Number" />

    <au:button value="submit" type="submit" name="Submit" />

</au:form>
```

Step3: Deploy portlet and check output.



Figure 2 : Output of simple AlloyUI form portlet

Components of Alloy UI

Alloy UI provides us different components to create awesome website. All components are defined in various categories like Form, Images, Audio-Video, Webpage and layout, Diagrams etc.

Forms related components

Form Builder

Form Builder provides a graphic user interface method for constructing forms with Textbox, Textarea, Checkbox, Button, Select, Radio Buttons, File Upload and Fieldset.

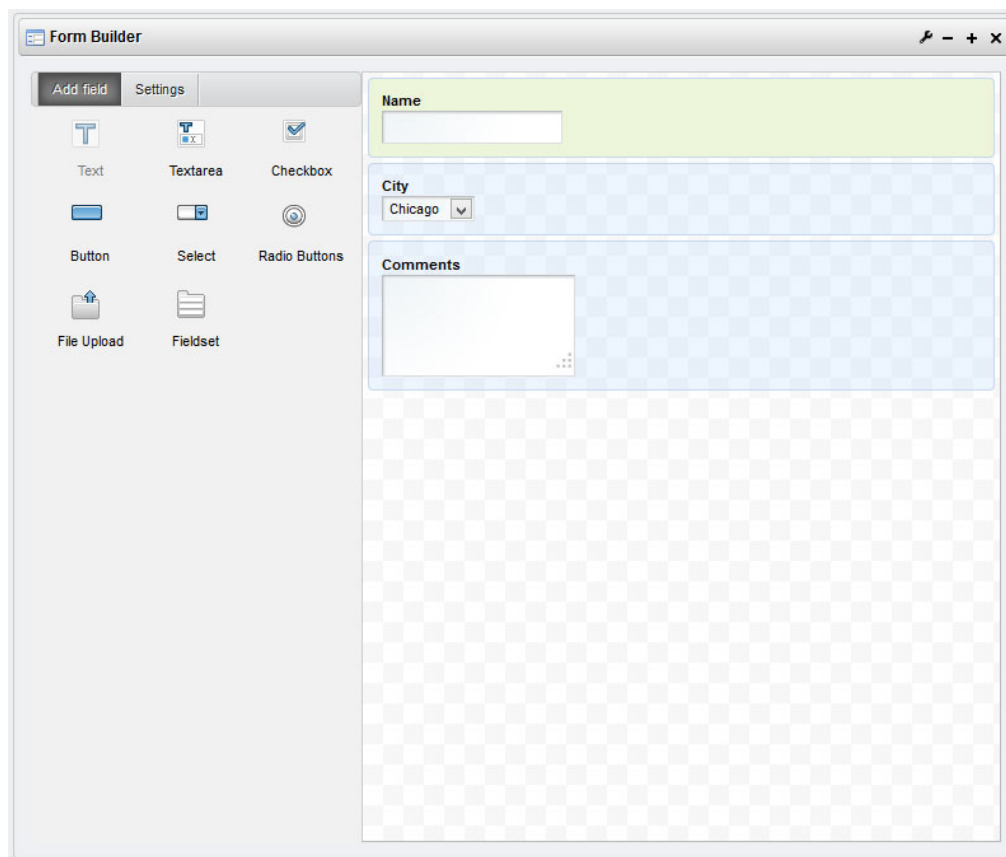


Figure 3 : Example of Form Builder

Form Validator

Form Validator allows us to create validation for form entries.

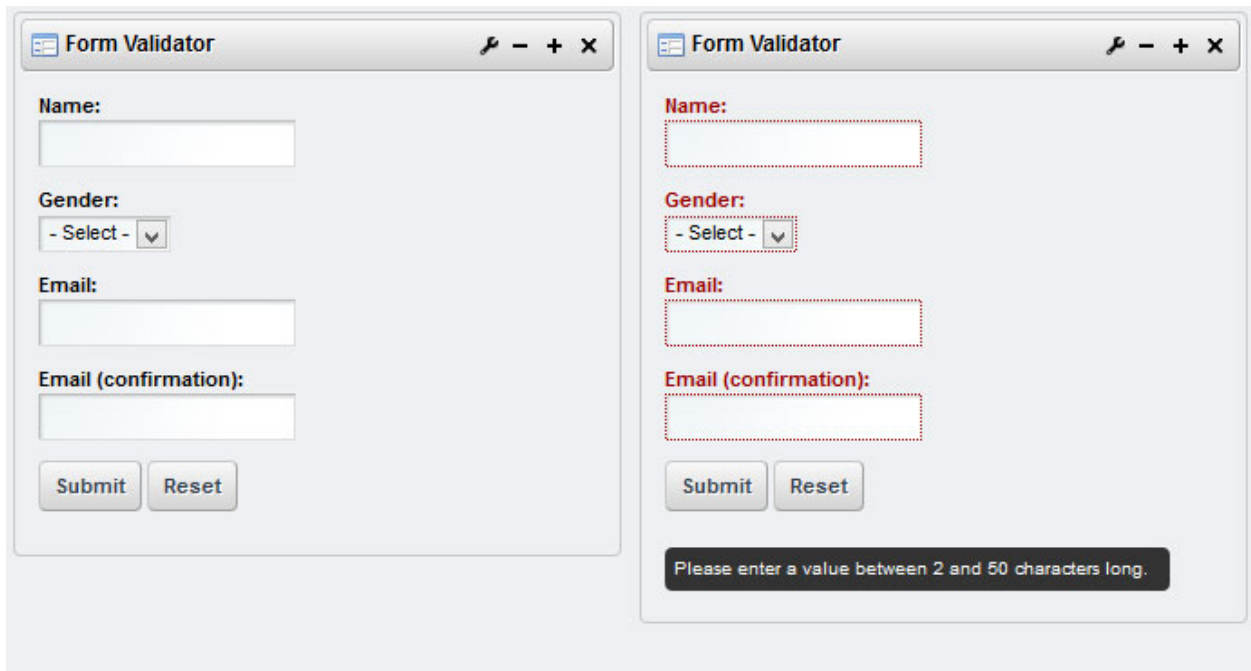


Figure 4 : Example of Form validator

Form validator component provides javascript to validate all form elements in portlet and display each element with error message with label in red color text.

Auto Complete

Auto Complete component is used to provide suggestions while users type into the field.

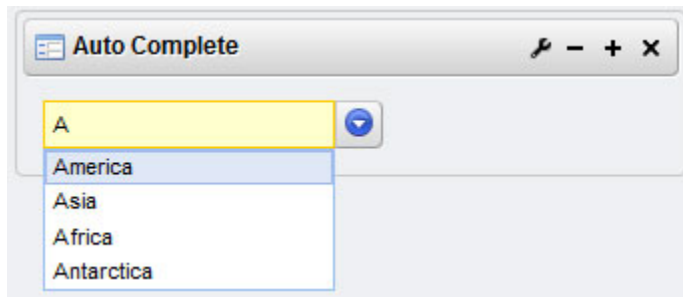


Figure 5 : Example of “Auto Complete” component.

Button Item

Button Item provides an iconic buttons for constructing graphical interface for website.

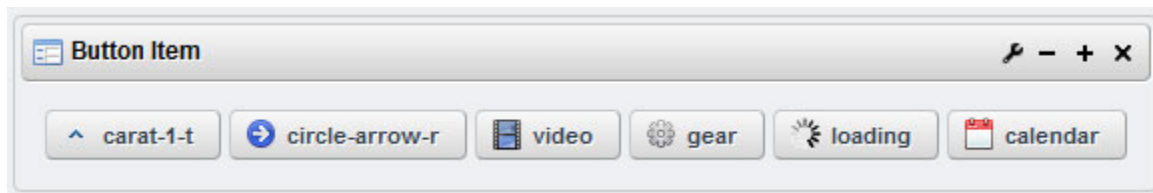


Figure 6 : Example for Button Item with various iconic buttons.

Char Counter

Char Counter allows us to count characters from textbox and limits the amount of text in a field. With new instance of Char Counter using these two elements and setting the maximum length to any number.

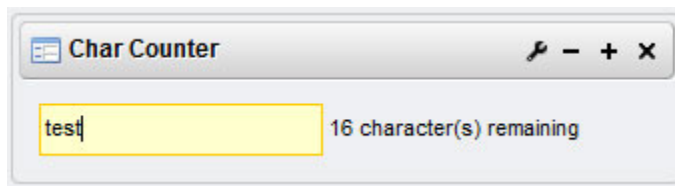


Figure 7 : Example of Char Counter with single textbox.

Date Picker

Create a dynamic datepicker that allows users to select the date with a calendar.

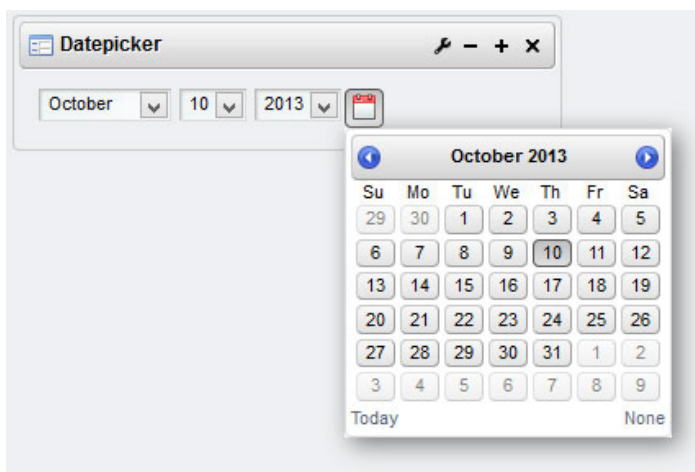


Figure 8 : Example of Datepicker with calendar view.

Textbox List

Textbox list allows us to displays user`s choices to them as a graphic list. The input field should be automatically updated with the first result as the user types, automatically selecting the section of the text the user has not typed yet. You can define a more definite schema and add the `matchKey` choice to choose the key or numeric index on the schema to match the result against.

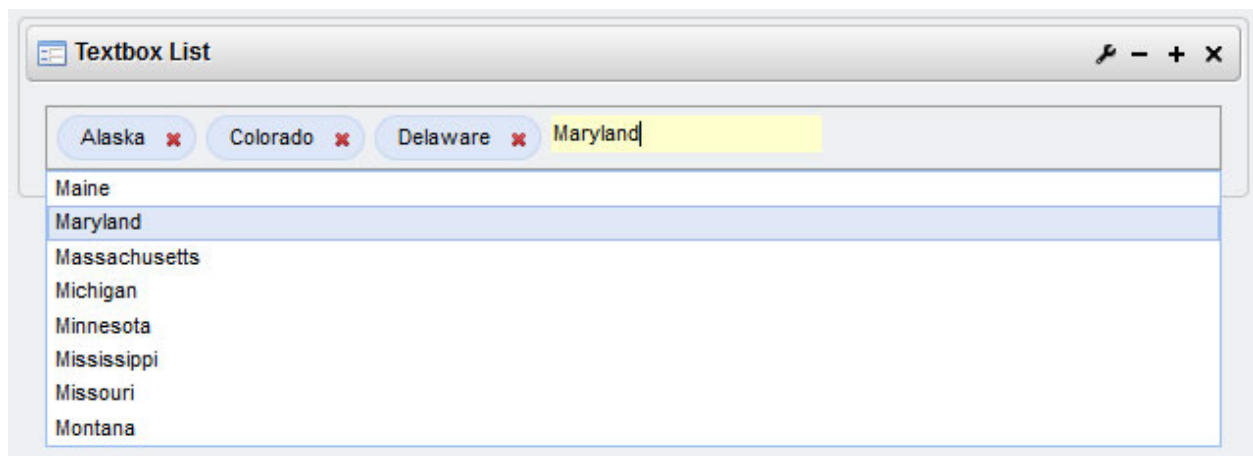


Figure 9 : Example of Textbox List with dummy data.

Tooltip

It gives users related information or content, such as help text, images or video. You can set the content of a tooltip with the title attribute of a HTML element.

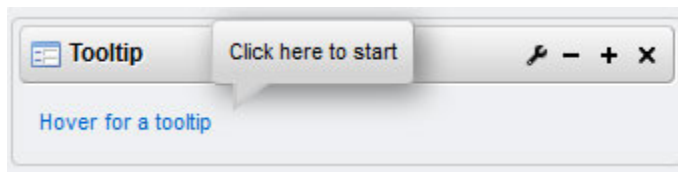


Figure 10 : Example of Tooltip.

Ace Editor

Ace Editor allows us to use embeddable code editor that matches the features of native editors. It is a simple HTML element Editor. The editor can also be set to load with code already written.

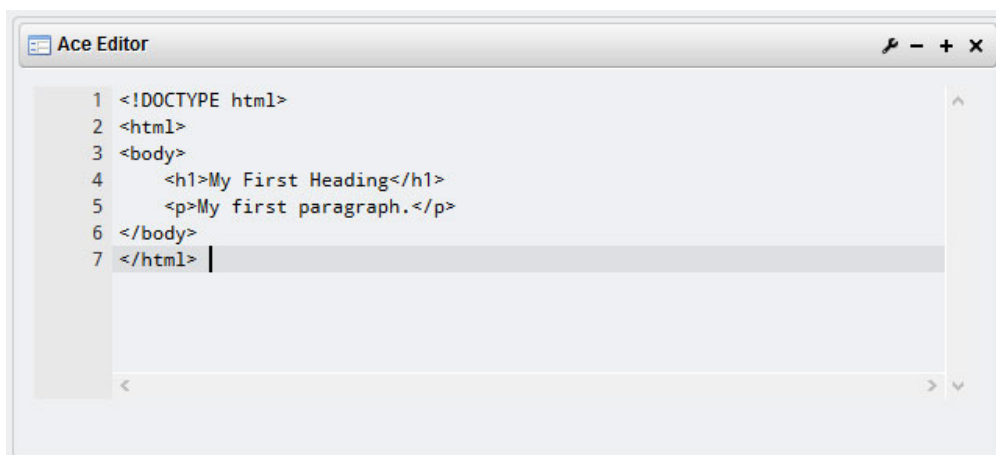


Figure 11 : Example of Ace Editor

Image related components

Carousel

Carousel provides an interactive way of cycling images through elements. It is simple image scroll with navigation.

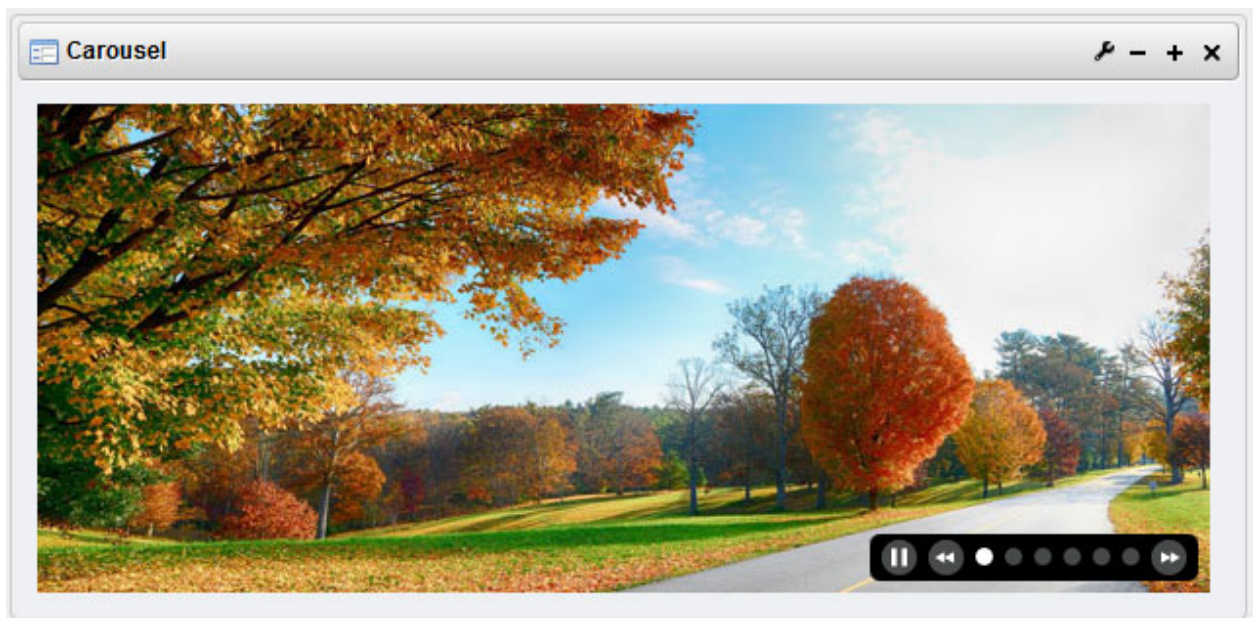


Figure 12 : Example of Carousel

Image Viewer

Image Viewer allows users to view and navigate throughout a collection of images. Parameters can add into Image Viewer to change element such as image captions, image positioning, and when images within the viewer are loaded. It is also use to declaring what elements are to be used to caption, navigation arrows, close button, and even the loader, which image being displayed. When user will click on image thumbnail, it will open large image in popup.

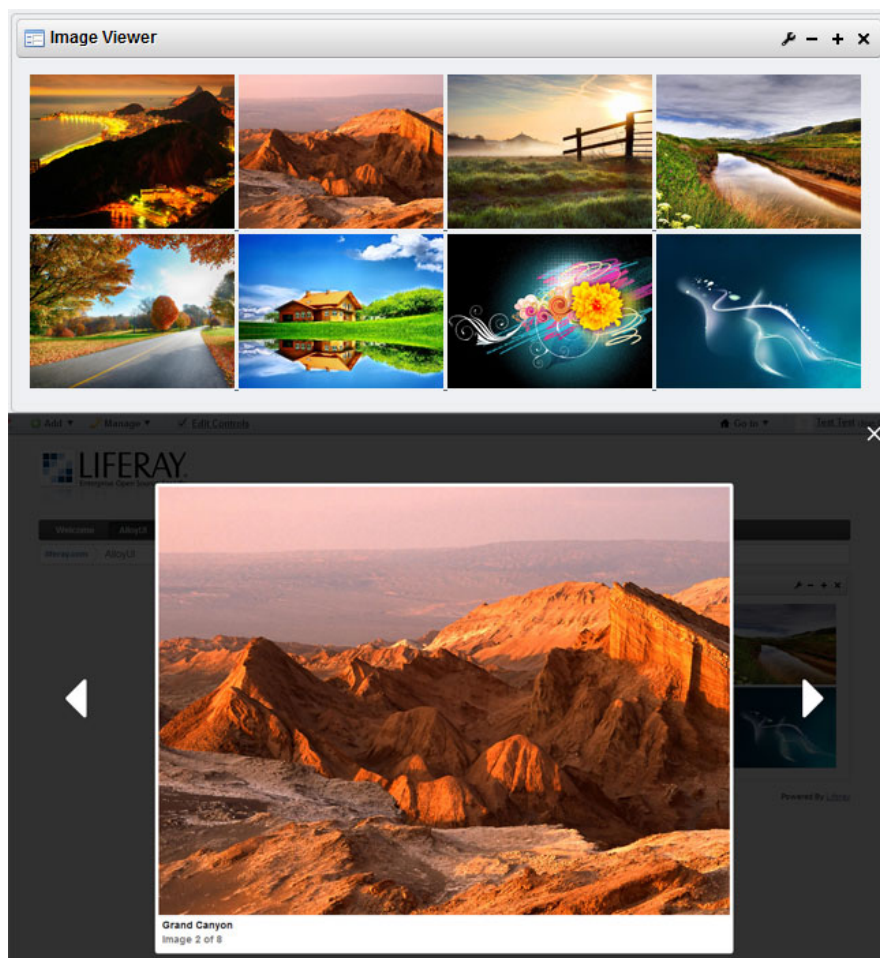


Figure 13 : Example of Image Viewer

Image Cropper

Image Cropper provides a draggable or resizable cropper widget to crop image sizes. You can prevent Image Cropper from moving, you can turn of Cropper resizing and you can also maintain a continuous aspect ratio for Image Cropper.

But liferay does not provide library for Image Cropper, We need to add bellow script in our project.

```
<script src="http://cdn.alloyui.com/1.7.0/aui/aui-min.js"></script>
```

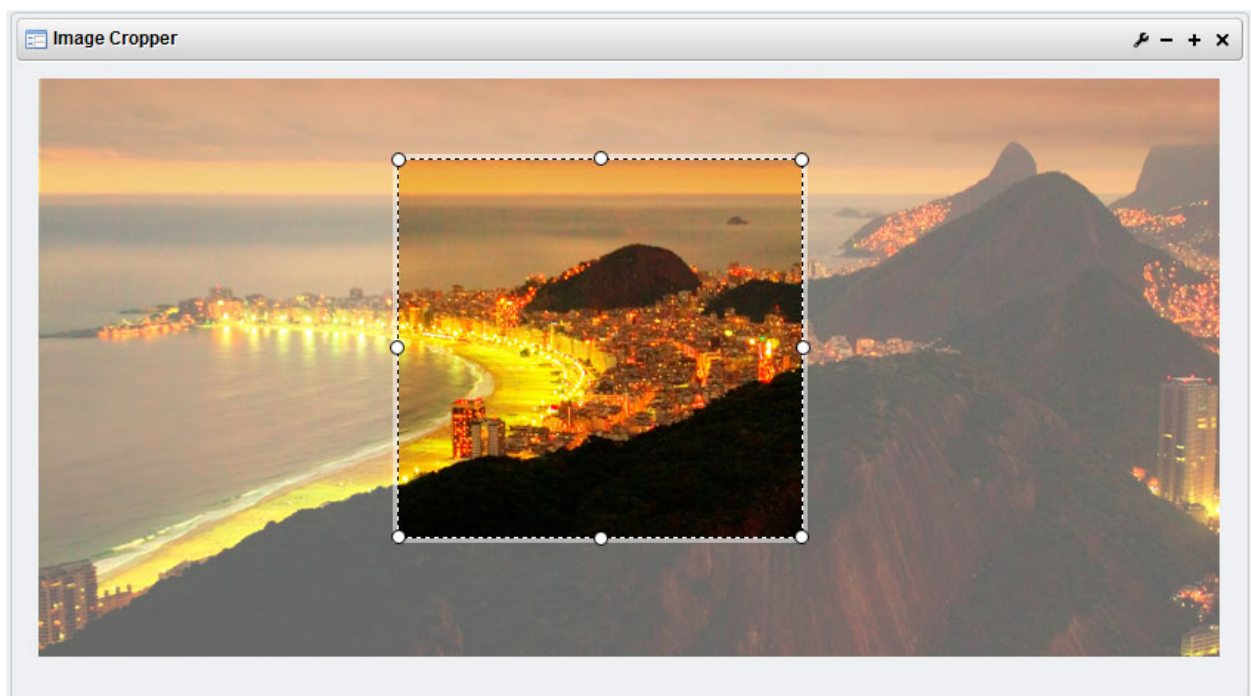


Figure 14 : Example of Image Cropper

(Note: Image Cropper is not totally feasible with liferay UI)

Audio-Video related components

Audio

Audio Component allows us to create an interactive audio player in HTML5 with contingency for old browsers.

(Note: Audio Component is not totally feasible with liferay)

Video

Video Component allows us to create an interactive HTML5 video player with contingency Flash video player. You can specify the Video's width and height by passing values. You can also load a poster image in the Video player before the user begins playing the video.

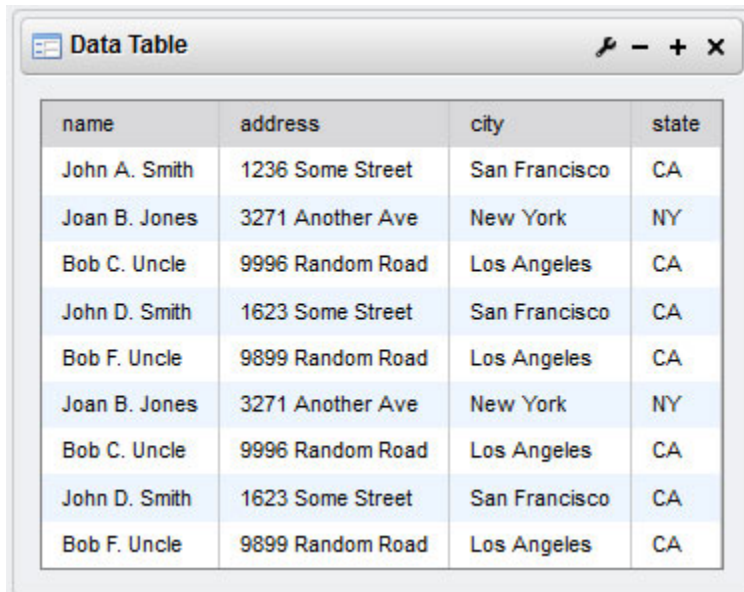


Figure 15 : Example of Video component

Webpage and Layout related components

Data Table

Data Table provides the user with a technique of organizing, displaying, and editing tables of information. Table cells can also be edited when permitted in the columns object.



name	address	city	state
John A. Smith	1236 Some Street	San Francisco	CA
Joan B. Jones	3271 Another Ave	New York	NY
Bob C. Uncle	9996 Random Road	Los Angeles	CA
John D. Smith	1623 Some Street	San Francisco	CA
Bob F. Uncle	9899 Random Road	Los Angeles	CA
Joan B. Jones	3271 Another Ave	New York	NY
Bob C. Uncle	9996 Random Road	Los Angeles	CA
John D. Smith	1623 Some Street	San Francisco	CA
Bob F. Uncle	9899 Random Road	Los Angeles	CA

Figure 16 : Example of Data Table

Diagram Builder

Diagram Builder allows us to drag-drop diagram elements, build new tasks, draw connectors from node to node. We should also populate our new Diagram fields with a Diagram field passing it a name, type, and x y location.

But liferay does not provide library for Diagram Builder, We need to add bellow script in our project.

```
<script src="http://cdn.alloyui.com/1.7.0/au/au-min.js"></script>
```

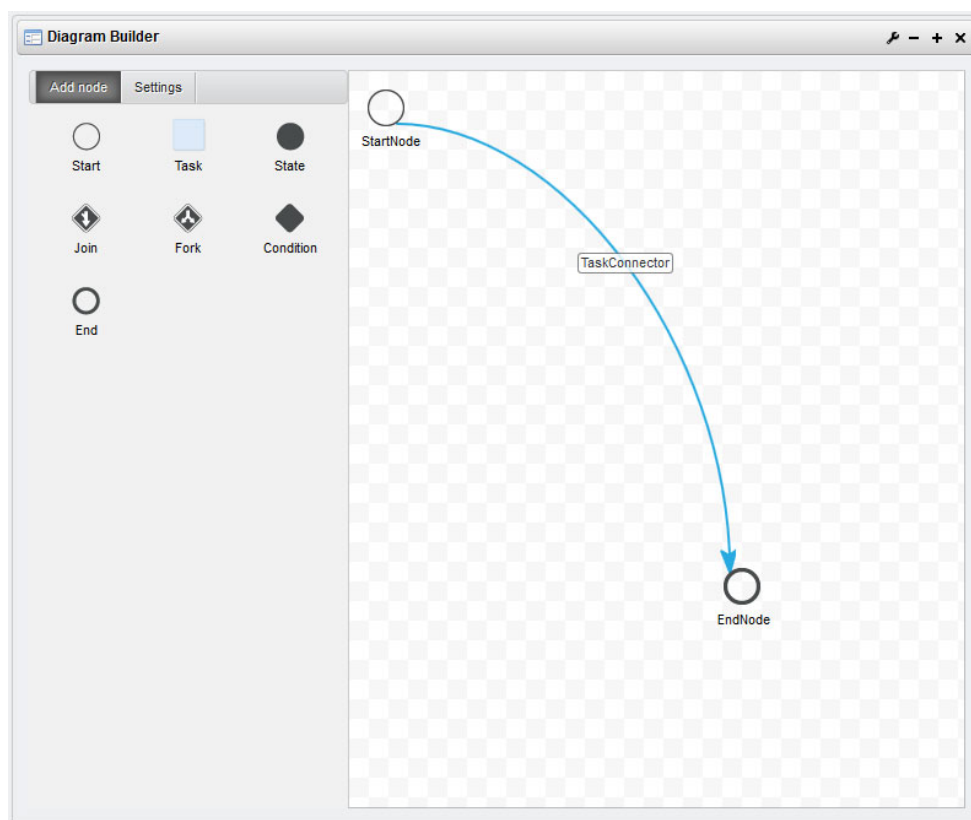


Figure 17 : Example of Diagram Builder

(Note: Diagram Builder is not totally feasible with liferay UI)

Dialog

Dialog allows us to displays content in a dialog window that can be dragged or accessible as a model. You can align the component on the center of your screen. You can even change the width and height of the dialog box. You can also turn off the facility to drag or resize your dialog. Another thing is that you can also add buttons to your dialog component.

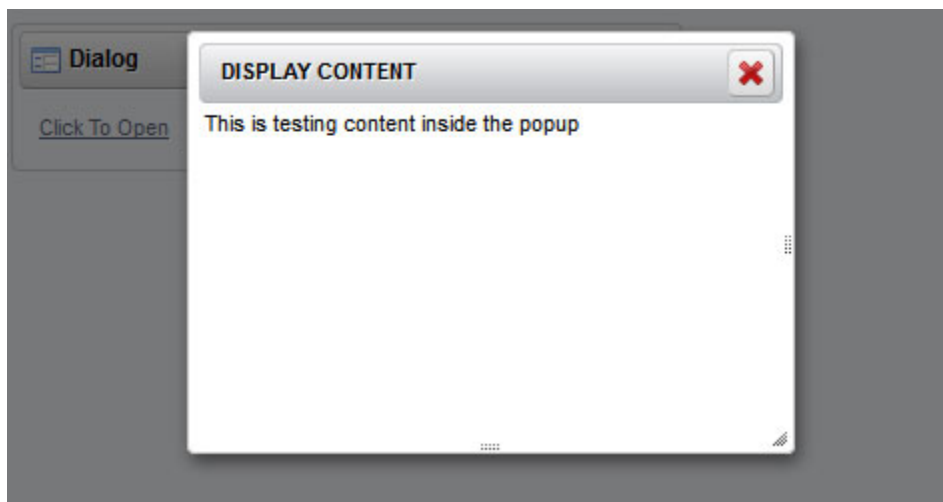


Figure 18 : Example of Dialog Component

I/O

I/O allows you to make asynchronous HTTP (Ajax) requests. You can specify the type of the request for e.g., json, xml, text, javascript.

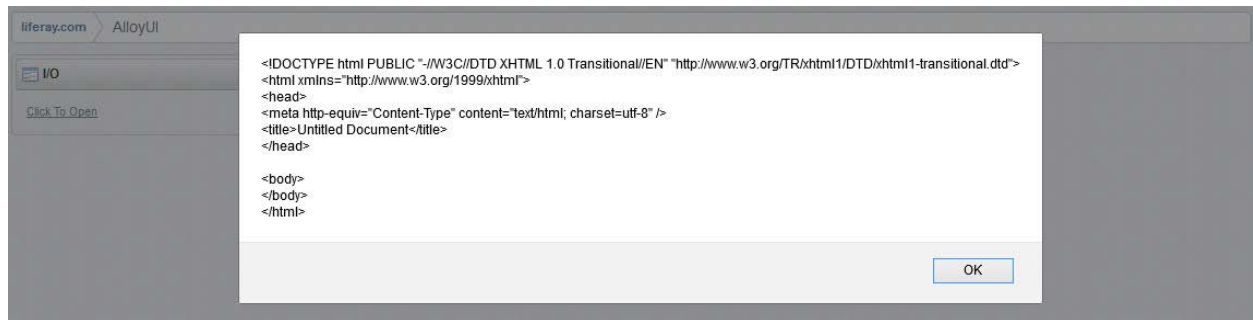


Figure 19 : Example of I/O

Nested List

Nested List gives the user the facility to interact with a list element. You can set a placeholder that will indicate as you drag your element where it will be dropped.

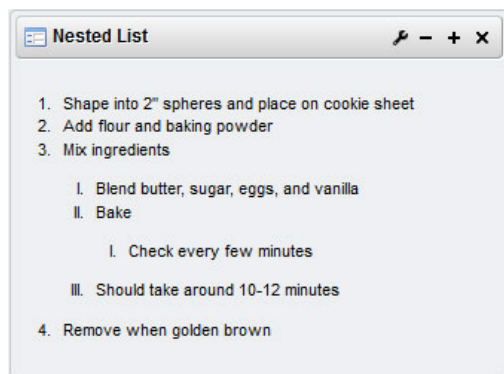


Figure 20 : Example of Nested List

Node

Node allows for communication with the DOM. It provides a set of techniques that assists in DOM manipulation. Simple DOM communication is possible without the node component, with `A.one`, `A.all`, and such commands being supplied in the YUI base code.

However, more complex actions which would be quite difficult with that code alone are made much simpler with the node component.

Node allows you to interact with the DOM in all sorts of ways that would usually be quite complex. You can get the margin or padding on a certain element, change an element's ID, center an element in a predecessor container, get or set an feature of an element, or many other useful things.

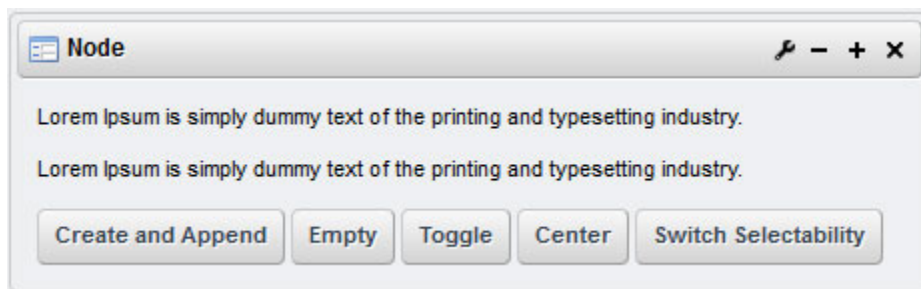


Figure 21 : Example of Node component

Paginator

Paginator provides a set of controls to navigate through paged data. There are many configuration options that can be passed into the Paginator. You can also change what is displayed in the different links of the Paginator.

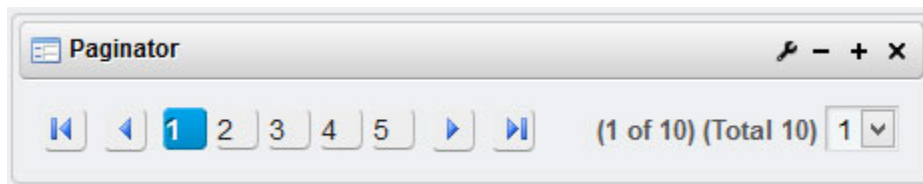


Figure 22 : Example of pagination

Progress Bar

Progress Bar allows users to view loading progress in real time. You can see what Progress Bar looks like when it loads larger content. You can set an Asynchronous Queue which will step through 100 iterations and set Progress Bar value on each iteration. This can give you an idea of what loading will look like in real world usage.

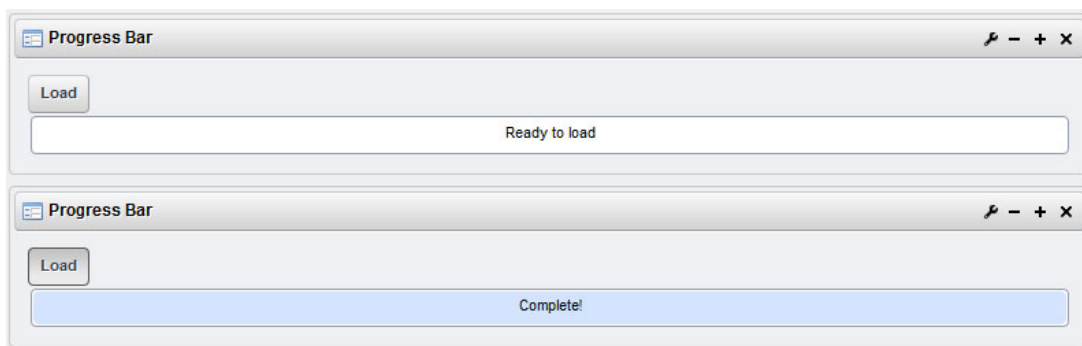


Figure 23 : Example of Progress Bar

Portal Layout

Portal Layout enables a layout with draggable or droppable functionality. You can set another HTML container to receive dropped Portal Layout object. Portal Layout can now drag and drop content nodes onto one another and inside the separate DIV. Liferay Layout is also based upon Portal Layout.

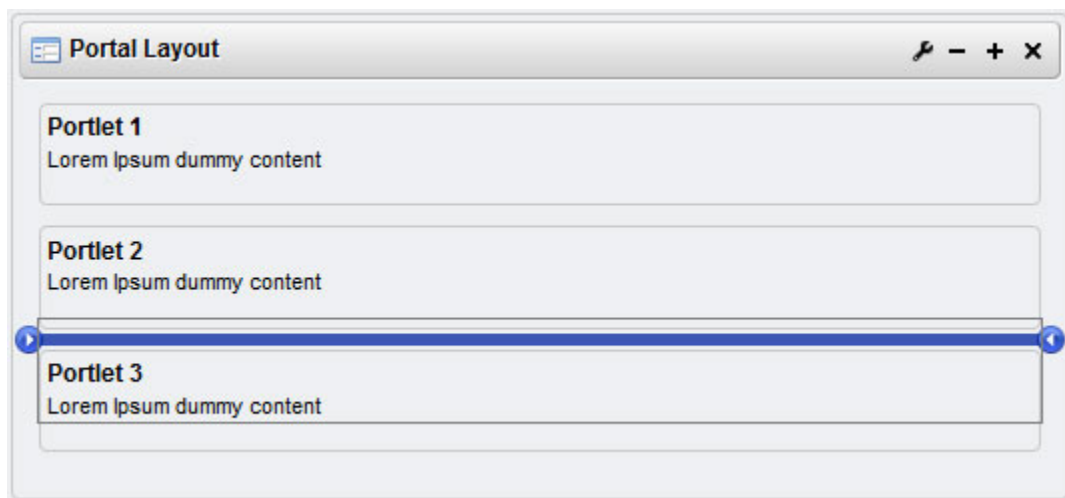


Figure 24 : Example of Portal Layout

Property List

Property List allows users to view and change properties from JavaScript objects. You can set an object's editor choice containing a new constructor. This option creates a checkbox input inside the object which can be toggled on/off within Property List.

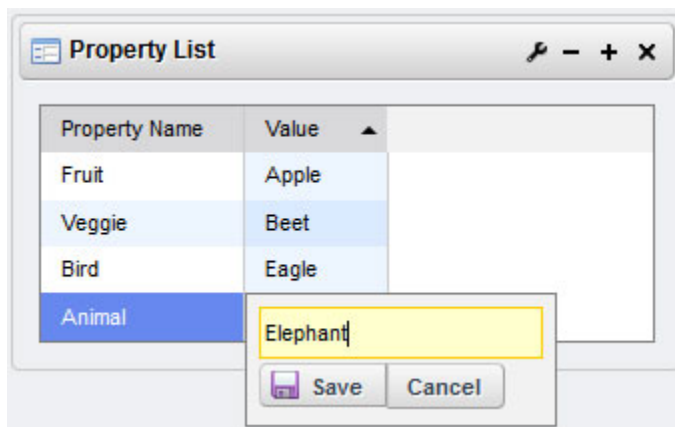


Figure 25 : Example of Property List

Rating

Rating allows users to set ratings for content, blog, wiki etc. you can pass the size option a whole number for how many options you want rating to have. You can also define Rating choices from HTML radio inputs, enabling control over individual title for each input. Rating can be defined as a thumb rating and star rating.

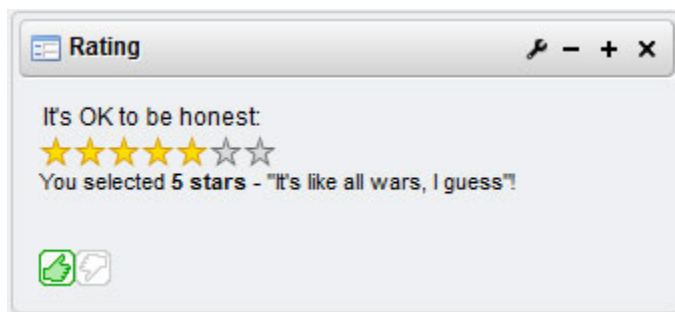


Figure 26 : Example of Rating

Scheduler

Scheduler allows user to access scheduler widget with built-in calendar. You can add other views to your views option with the Scheduler View constructors. This will permits you to view your events in different formats. You can also view Scheduler to start on.

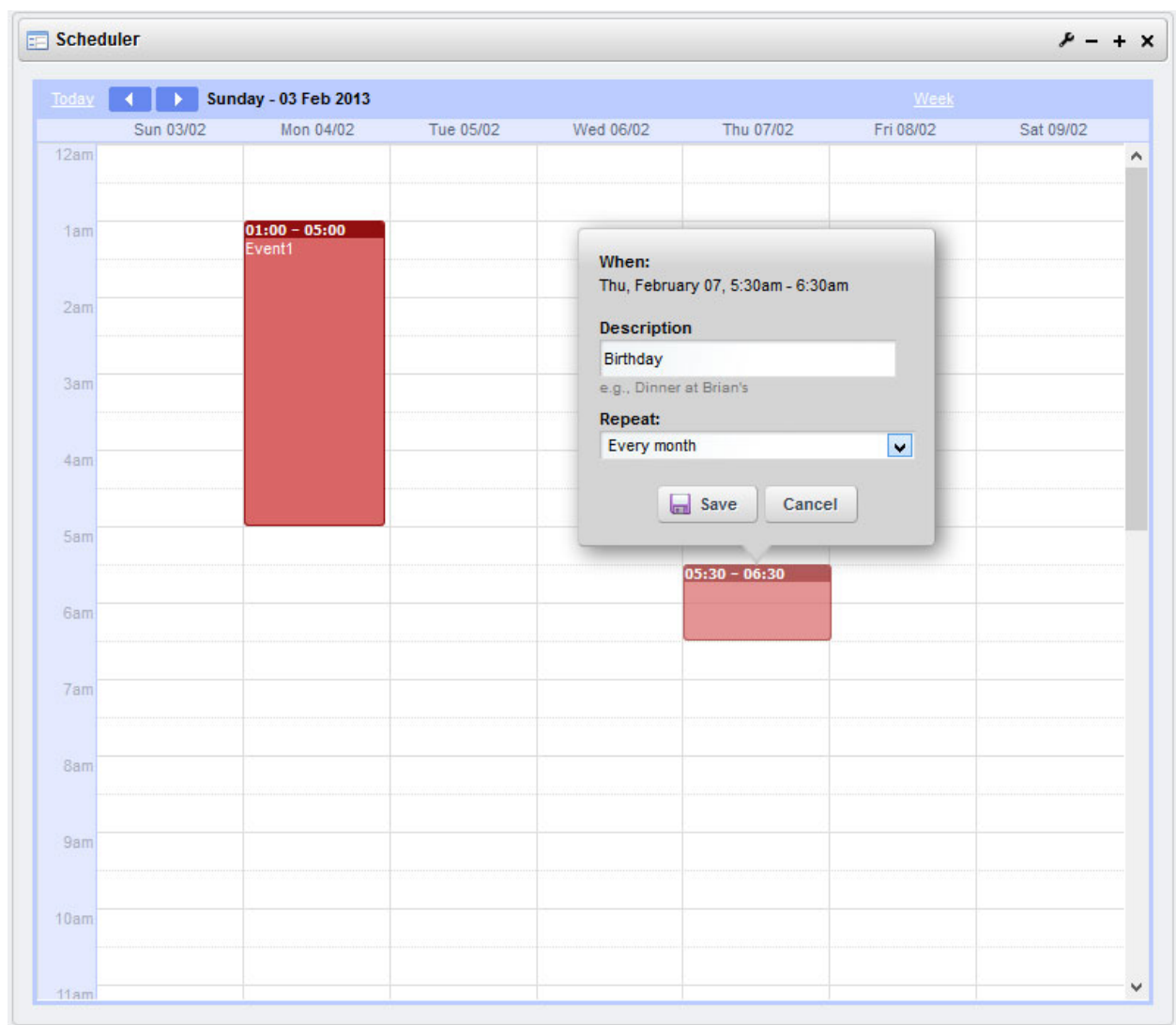


Figure 27 : Example of Scheduler

Viewport

Viewport allows us to create a cross-browser method of adapting web design to display size. Viewport allows you to modify your CSS for the four most normally used page widths: 960px, 720px, 480px, and 320px, 960 for desktops and laptops, 720 for tablets, 480 for smart phones, and 320 for mobile phones. It is very useful to create Responsive Design.

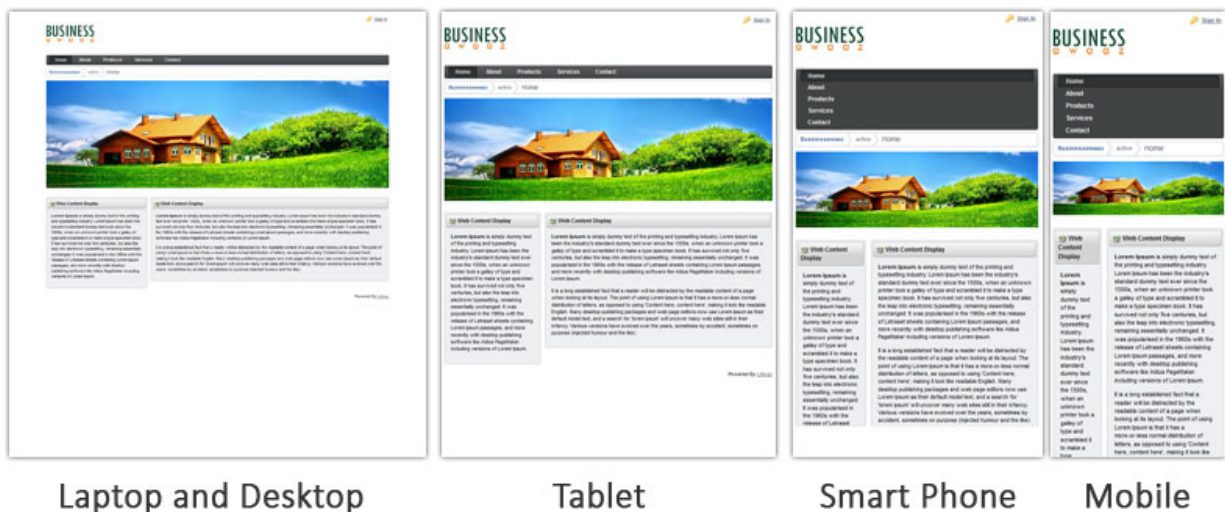


Figure 28 : Example of Viewport

Thank You for reading.....