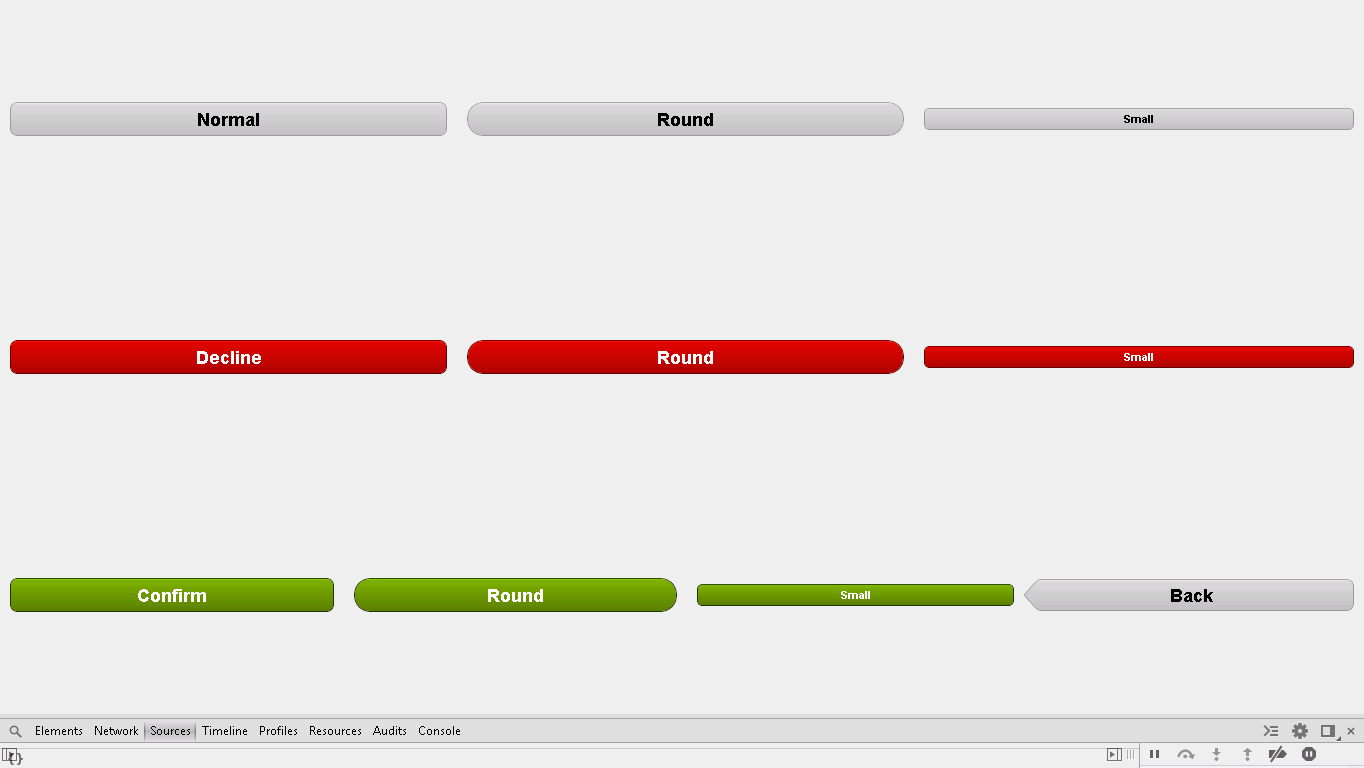
**Buttons**:

This example shown in snapshot1.1 below shows different types of buttons which can be used at appropriate place where required:



**Snapshot 1.1**

**Code**

**Following is code snippet for buttons example of FOS:**

* ­­Buttons view extends to super class of Container; Container is class here, which holds layouts of button application.

**extends: ’Ext.Container’**

* Button application requires VBox supper class to use it in our application, for vertical box layout.

**requires: ’Ext.layout.VBox’**

* Any sencha code starts with Config{} block which contains actual code.
* Layout is provided to complete application using following code:

layout: {

type : 'vbox',

pack : 'center',

align: 'stretch'

},

* cls : ’card1’ : cls here stands for class given a name as ‘card1’
* Following code will give layout to application which will overlap previous layout

This layout is HBox that mean horizontal box upon vertical box which gives grid layout as whole.

Here, we are going to design button so buttons are declared in default box as xtype.

defaults: {

xtype: 'container',

flex : 1,

layout: {

type : 'hbox',

align: 'middle'

},

defaults: {

xtype : 'button',

flex : 1,

margin: 10

}

},



**Snapshot 1.2**

**Following are items which are buttons as declared in default block**.

* When you provide ui as round it will give you normal button with no color(default clolor grey), you can also declare it as small to reduce size of button as required and respective tex using ‘text’ property.

Text:’Normal’ :-> Gives normal button with default grey colour and default size.

Ui:’round’ :-> Gives round corner button with default grey colour and default size.

Ui:’small’ :-> Gives normal small button with default grey colour and default size.

items: [

{

items: [

{text: 'Normal'},

{ui: 'round', text: 'Round'},

{ui: 'small', text: 'Small'}

] },

* When you provide ui as decline it will give you normal button with red colour(default colour grey), you can also declare it as small to reduce size of button as required and respective text using ‘text’ property.

ui: 'decline', text: 'Decline':-> Gives normal button with default red colour and default size.

ui: 'decline-round',text:'Round':->Gives round corner button with default red colour and default size.

ui: 'decline-small', text: 'Small':->Gives normal small button with default grey colour and default size.

{

items: [

{ui: 'decline', text: 'Decline'},

{ui: 'decline-round', text: 'Round'},

{ui: 'decline-small', text: 'Small'}

]

},

* When you provide ui as decline it will give you normal button with green colour(default colour grey), you can also declare it as small to reduce size of button as required and respective text using ‘text’ property.

ui: 'confirm', text: 'Confirm':-> Gives normal button with default green colour and default size.

ui: 'confirm-small', text: 'Round', text: 'Round':->Gives round corner button with default green colour and default size.

ui: 'confirm-small', text: 'Small':->Gives normal small button with default green colour and default size.

ui: 'back', text: 'Back':->Gives default back button with default grey color and normal size, its shape is quite different, it shows back arrow using button.

{ items:

[

{ui: 'confirm', text: 'Confirm'},

{ui: 'confirm-round', text: 'Round'},

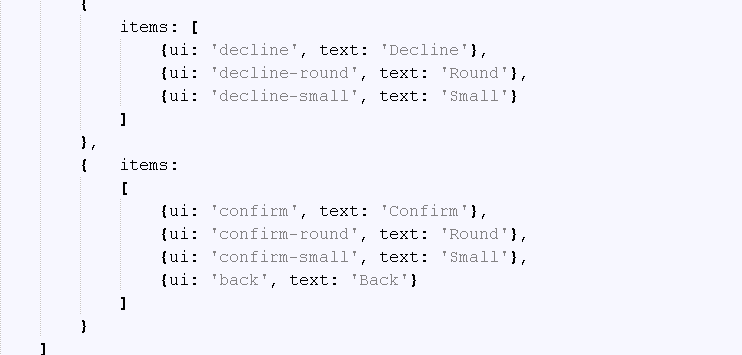
{ui: 'confirm-small', text: 'Small'},

{ui: 'back', text: 'Back'}

]

}

]



**Snapshot 1.3**

**FAQ:**

1. Is it necessary to use VBOX and HBOX in layout for using buttons?

Ans: No. It is not compulsory it depends on what layout you want.

1. For what FLEX is used?

Ans: Flex is the ratio to the width/height of each child.  
  
Your width of the wrapping Panel is 100%. Say your child items have flex specified as 1, 2, 3. Add them together to get 6 and take 100 divide by 6 you got from adding the flex together and you get 16.6667.  
  
Now when flex is equal to 1 the width is 16.6667% that of the wrapping Panel's width. Then your flex = 2 the width is 33.3333% and flex = 3 is about 50%. Add them up and you get 100%.

**CHECKPOINTS**:

1. Check carefully with your indentations as if you will miss any of the braces by mistake may reflect in compilation error.
2. See that your .js file starts with Ext.Define block and check that ends at the end of the program.
3. More than one items are separated by comma (,) separator and main program block of define is ended by semi colon (;)
4. Any value is assigned using colon (:) for example ui: 'confirm' and value is written inside single coats.
5. Items block starts with square brackets [], which contains multiple members and also sub items.
6. Don’t forget to add entries of your files in app.js.

**Source Code**

****