hybrid

AKHIL S
akhil.new10@gmail.com

1 Feb

sencha

(part 1)

products:

- Sencha Platform

> Comprehensive platform for Web Application Lifecycle Management .

- Sencha Touch

> Cross-platform mobile web application framework based on HTML5 and JavaScript for creating universal mobile apps.

- Sencha Cmd

> Cornerstone for building Sencha Ext JS and Sencha Touch applications.

- Web Application Manager

> Enables teams to get their web applications to market faster and more securely by eliminating the need for native packagers and app store publication.

- Sencha Architect

> Empowers to build HTML5 applications using drag-and-drop features, hence spend less time on manual coding and application code is optimized for high performance.

Sencha Touch:

> mobile web framework- for creating web pages for mobile devices-looks like native apps.

Sencha Touch { (javascript+css) ~ creates HTML struct

> runs inside browsers, like other frameworks.

> based on HTML5 ,CS3, Javascript

> don't have access to native API's.

basic concepts:

- **Classes**

> Javascript dont have classes.

: define prototype/class } Prototype - objects used to create other objects, instead of using class.

syn: Ext.define(className, members, onClasscreated);

eg: Ext.define('Animal', {
config: {
name:null
},
constructor:function(config) {

```
this.initConfig(config);
},
speak:function() {
alert('grunt');
}
});
```

: Create Object from Prototype }

```
eg: var bob= Ext.create('Animal',{
name:'Bob'
});
bob.speak();              //alerts 'grunt'
```

- **Inheritance**

  > Create Human Prototype which inherits from Animal Prototype

  ```
  eg:Ext.define('Human', {
  extend:'Animal',
  speak:function() {
  alert(this.getName());   //override speak() function
  }
  });
  ```

  > Create specific Human

  ```
  eg: var bob= Ext.create('Human', {
  name:'Bob'
  });
  bob.speak(); //alerts 'Bob'
  ```

- **Config**

  > placing variable under config - automaticaly create following functions

- getName()- that returns the current value of variable.
- setName()- that sets a new value to variable and calls applier and updater functions.
- applyName()- called at the beginning of the setter function BEFORE the value is changed.

Setter function will save value returned by applier function.

If applier function returns false value will not be changed.

- updateName()- called at the end of the setter function AFTER the value has been changed.

It is called ONLY if the value has been changed; if new value is the same as last one it is not called.

- **Creating Objects**

> two types- Pure JSON & Ext.create()

- **Pure JSON:**

> each object is defined with curly brackets {}

> contains simple key-value pairs where key is a simple string and value can be

○ string

○ array designated using square brackets []

○ function designated using function() {...}

○ other object designated using curly brackets {} as mentioned before

eg: //CREATE BUTTON.

```
var button= {
xtype :'button',
text :'Button',
listeners : {
tap :function() {
alert("Button was pressed");
}};
```

- **Ext.create():**

> first parameter defines type of Object.

> second parameter is Object in JSON syntax defining additional properties.

eg: //CREATE BUTTON USING FUNCTION.

```
var button2 =Ext.create('Ext.Button',{
text :'Button',
listeners : {
tap :function() {
alert("Button was pressed");
}}
});
```

basic building blocks for creating an interface **~ Container & Components**

Containers :

types:

> **Container**: Ext.Container is basic Container

> **Panel:** Ext.Panel seems to have the same functionality as Ext.Container, additionally having a toolbar.

> **Tabbed Panel**: Ext.tab.Panel Container uses card layout by default.

: It contains toolbar with buttons to switch between items.

> **Carousel**: Ext.Carousel container uses card layout by default.

: You can switch between items by swiping with your finger horizontally (default) or vertically using direction parameter.

.●direction :'horizontal'

●direction :'vertical'

: Indicator at the bottom shows how many items there are and on which item we are on.

> **Navigation View:** Ext.navigation.View container uses card layout by default.

: Back button is also automatically added to get you back to the previous screen.

: By using push() function you can add new card which automatically comes into view.

Layouts:

types:

> **Auto Layout :** displays multiple items.

: Each item fills the entire available width but it only occupies as much height as required to show the content.

> **Fit Layout:** displays single item at the time.Shown item fills the entire available space, both horizontally and vertically.

: There is no way of changing which item is active since it always displays the last one.

> **Card Layout:** displays single item at the time.

: Shown item fills the entire available space, both horizontally and vertically.

: Card layout provides programmatic interface for switching between items.

> **vbox Layout**: arranges items vertically.

: Displays all items each taking full width but only as much height as required by the content.

Additional Parameters :

* pack ~ controls vertical positioning of the items.

* align ~ controls horizontal positioning of the items, without this parameter components take full width.

* flex ~ defines proportion of vertical space that each item will occupy.

> **hbox Layout:** arranges items horizontally.

: Displays all items each taking full height but only as much width as required by the content.

Additional Parameters :

* pack ~ controls horizontal positioning of the items.

* align ~ controls vertical positioning of the items, without this parameter components take full width.

* flex ~ defines proportion of horizontal space that each item will occupy.

GUI Components:

type:

> **Label:** Ext.Label is used to write some text.

> **FieldSet**: Ext.form.FieldSet is used to visually separate elements of a form by dividing them into group.

: Field Set can optionally have a title at the top and instructions at the bottom.

> **TextField:** Ext.field.Text is the basis for most of the input fields in Sencha Touch.

: It provides shared functionality such as: input validation, standard events, state management and look and feel.

> **Number field:** Ext.field.Number field is used to input numbers.

: It provides shared functionality such as: input validation, state management and look and feel.

> **Radio field:** Ext.field.Radio field allows user to choose one option.

> **Select field:** Ext.field.Select field allows user to choose one or more items from a drop down menu.

> **Toggle field:** Ext.field.Toggle is field with ON/OFF position.

: By default the toggle component can be switched between the values of 0 and 1.

> **Checkbox field**: Ext.field.Checkbox field allows user to choose multiple options.

> **TextArea Field:** Ext.field.TextArea creates an HTML Text Area Field on the page useful for entering large amounts of text.

> **List:** Displays list of items.

: Ext.data.Store ~ contains data which should be listed.

: ItemTpl specifies which data from the store will be presented and in what format.

: onItemDisclosure() ~ function which should be called when user presses arrow on the right.

: record.getId(): ~ gets value of 'id' field from the data store for the selected record/item.

> Button: Ext.Button creates a button.

> Toolbar: Ext.Tool bar creates toolbar with title and buttons.

: If title is too long it will overlap with button which can be avoided by using Titlebar.

> TitleBar: Ext.TitleBar creates toolbar with title and buttons where title will not overlap button if it is too long.

> Alert: Displays popup window.

> Prompt: Displays popup window with text field and two buttons.

> Confirm: Displays popup window with title, text and two buttons.

.