JAVA DAVE ENVIRONMENT

Introduction to Ext JS & more…;

Java workshop training, August, 2017.

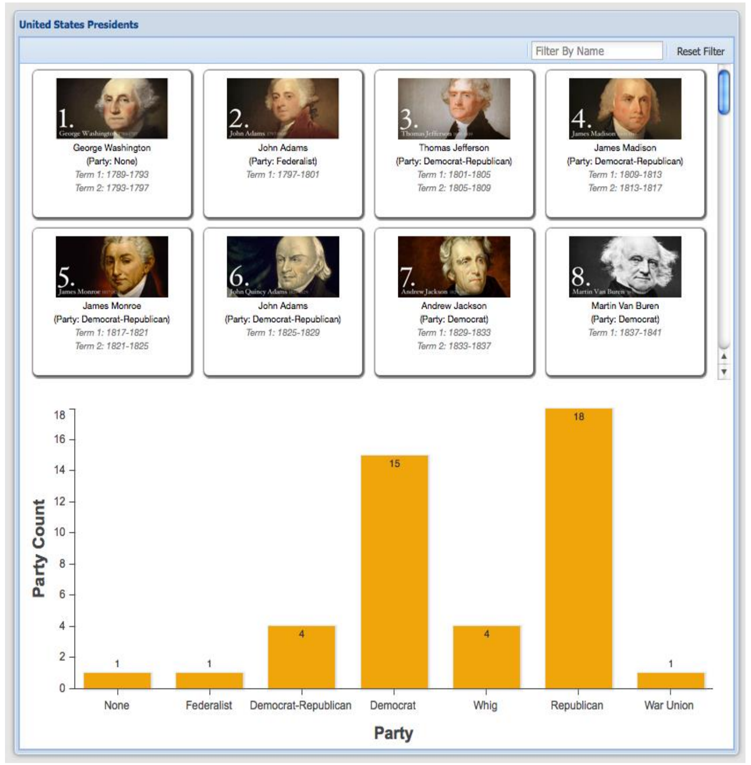
Dr. Kishore Biswas (Forrest/柯修)

PhD. Artificial Intelligence—Natural Language Processing.

CanWay IT Training ®.

CONTENTS

EXTJS 4.0


United States Presidents

Filter By Name

Reset Filter

1. George Washington
(Party: None)
Term 1: 1789-1793
Term 2: 1793-1797

2. John Adams
(Party: Federalist)
Term 1: 1797-1801

3. Thomas Jefferson
(Party: Democrat-Republican)
Term 1: 1801-1805
Term 2: 1805-1809

4. James Madison
(Party: Democrat-Republican)
Term 1: 1809-1813
Term 2: 1813-1817

5. James Monroe
(Party: Democrat-Republican)
Term 1: 1817-1821
Term 2: 1821-1825

6. John Adams
(Party: Democrat-Republican)
Term 1: 1825-1829

7. Andrew Jackson
(Party: Democrat)
Term 1: 1829-1833
Term 2: 1833-1837

8. Martin Van Buren
(Party: Democrat)
Term 1: 1837-1841

JavaScript MVC Framework

What is ExtJS?

- ExtJS is a MVC Framework
  - Model-View-Control
  - Unlike JQuery and other major JS library providers

- Ext JS is a popular JavaScript framework which provide rich UI for building web applications with cross browser functionality. Ext JS is basically used for creating desktop applications It supports all the modern browsers as IE6+, FF, Chrome, safari 6+, opera 12+ etc. Whereas another product of sencha, sencha touch is used for mobile applications.
- Ext JS is based on MVC/ MVVM architecture. The latest version of Ext JS 6 is a single platform which can be used for both desktop and mobile application without having different code for different platform.

WHO PROVIDES EXTJS

- ExtJS is provided by Sencha (www.sencha.com)
  - Sencha Touch
  - GWT
  - CSS Animator
  - IO (Cloud Data Management)

FEATURES

- These are the highlighted features of Ext JS

- Customizable UI widgets with collection of rich UI such as Grids, pivot grids, forms, charts, trees.
- Code compatibility of new versions with the older one.
- A flexible layout manager helps to organize the display of data and content across multiple browsers, devices, and screen sizes.
- Advance data package decouples the UI widgets from the data layer. The data package allows client-side collection of data using highly functional models that enable features such as sorting and filtering.
- It is protocol agnostic, and can access data from any back-end source.
- Customizable Themes Ext JS widgets are available in multiple out-of-the-box themes that are consistent across platforms.

WHY

- This is really what matters (MVC)
    - Easy Client-side data modeling
        - Relational Models
    - Simple to use GUI widgets
    - Full robustness of EcmaScript control

- Benefits
- Sencha Ext JS is the leading standard for business-grade web application development. Ext JS provides the tools necessary to build robust applications for the desktop and tablets.
- Streamlines cross-platform development across desktops, tablets, and smartphones — for both modern and legacy browsers.
- Increases the productivity of development teams by integrating into enterprise development environments via IDE plugins.
- Reduces the cost of web application development.
- Empowers teams to create apps with a compelling user experience.
- It has set of widgets for making UI powerful and easy.
- It follows MVC architecture so highly readable code.

TOOLS

These are the tools provided by sencha used for Ext JS application development mainly for production level.

Sencha Cmd

- Sencha CMD is a tool which provides the features of Ext JS code minification, scaffolding, production build generation.

Sencha IDE Plugins

- Sencha IDE plugins which are integrates Sencha frameworks into IntelliJ, WebStorm IDEs. Which helps in improving developer's productivity by providing features such as code completion, code inspection, code navigation, code generation, code refactoring, template creation, and spell-checking etc.

Sencha Inspector

- Sencha Inspector is a debugging tool which helps debugger to debug any issue while development.

MVC

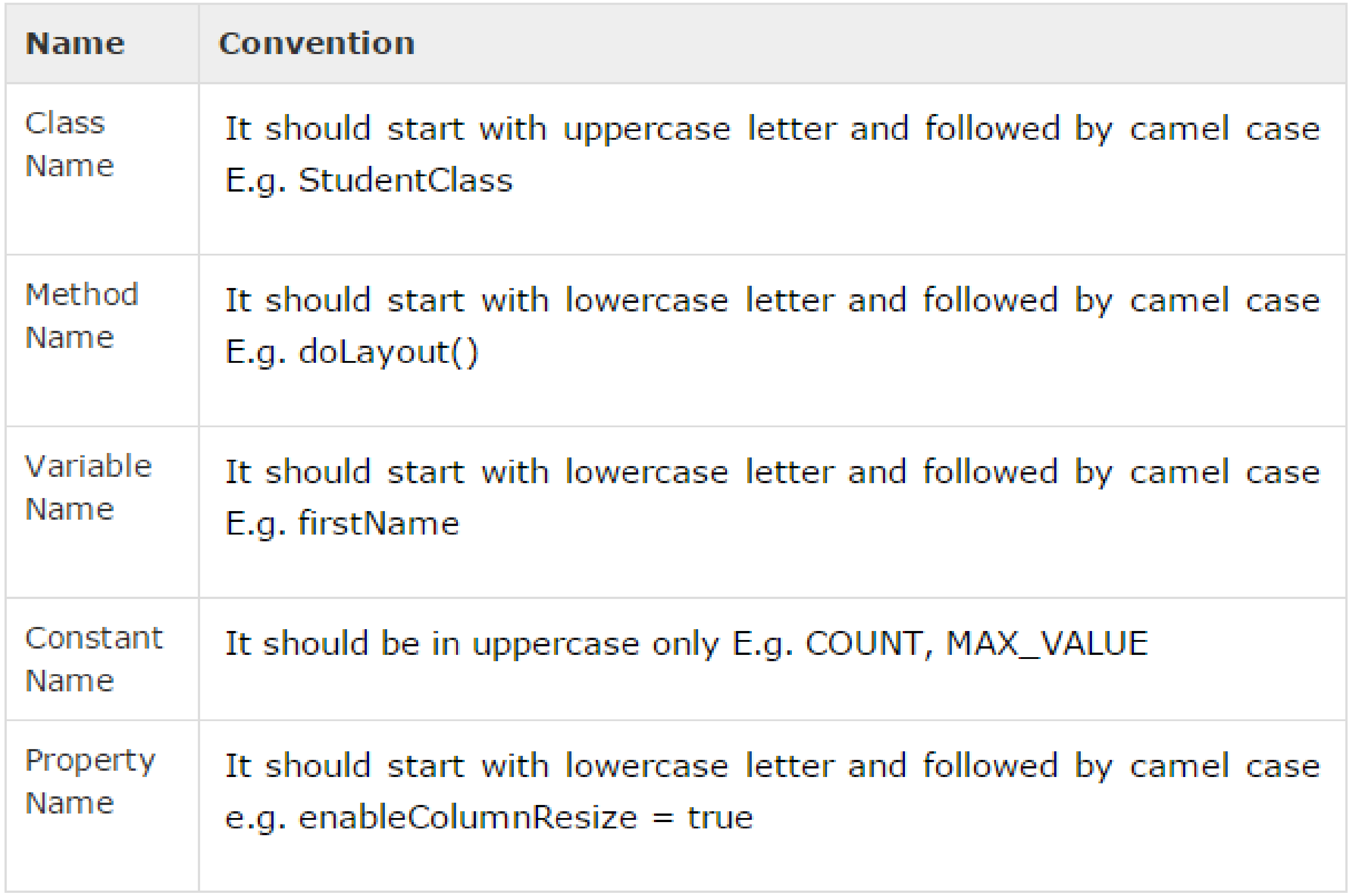
- Why is MVC so important?
  - In this case, it is because it is 100%, agent-based, client side code
  - This means typical MVC on the server is not needed
    - Good or Bad? Design decision

Ext.JS - Naming Convention…

- Naming convention is a set of rule to be followed for identifiers.
- It makes code more readable and understandable to the other programmers as well.
- Naming convention in Ext JS follows the standard JavaScript convention which is not mandatory but a good practice to follow.
- It should follows camel case syntax for naming the class, method, variable and properties.

- If name is combined with two words, second word will start with uppercase letter always e.g. doLayout(), StudentForm, firstName etc.

EXT.JS - NAMING CONVENTION



Name	Convention
Class Name	It should start with uppercase letter and followed by camel case E.g. StudentClass
Method Name	It should start with lowercase letter and followed by camel case E.g. doLayout()
Variable Name	It should start with lowercase letter and followed by camel case E.g. firstName
Constant Name	It should be in uppercase only E.g. COUNT, MAX_VALUE
Property Name	It should start with lowercase letter and followed by camel case e.g. enableColumnResize = true

Ext.JS - Architecture

- Ext JS follows MVC/ MVVM architecture.
- MVC – Model View Controller architecture (version 4)
- MVVM – Model View Viewmodel (version 5)
- This architecture is not mandatory for the program but it is best practice to follow this structure to make your code highly maintainable and organized.

```
•   <!DOCTYPE html>
•   <html>
•     <head>
•       <link href="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/classic/theme-classic/resources/theme-classic-all.css" rel="stylesheet" />
•       <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/extjs/6.0.0/ext-all.js"></script>
•       <script type="text/javascript">
•         Ext.onReady(function() {
•         Ext.create('Ext.Panel', {
•           renderTo: 'helloWorldPanel',
•           height: 200,
•           width: 600,
•           title: 'Hello world',
•           html: 'First Ext JS Hello World Program'
•           });
•         });
•       </script>
•     </head>
•     <body>
•       <div id="helloWorldPanel" />
•     </body>
•   </html>
```

Explanation

- Ext.onReady() method will be called once the Ext JS is ready to render the Ext JS elements.
- Ext.create() method is used to create object in Ext JS here we are creating an object of simple panel class Ext.Panel.
- Ext.Panel is the predefined class in Ext JS for creating a panel.
- Every Ext JS class has different properties to perform some basic functionalities.
- Ext.Panel class has various properties as:
- renderTo is the element where this panel has to be render. 'helloWorldPanel' is the div id in Index.html file.
- Height and width properties are for giving custom size of the panel.
- Title property is to provide the title to the panel.
- Html property is the html content to be shown in the panel.

Ext.JS - Class System

Ext JS is a JavaScript framework which has functionalities of object oriented programming. Ext is the namespace which encapsulates all the classes in Ext JS.

Defining a class in Ext JS

- Ext provides more than 300 classes which we can use for various functionalities.
- Ext.define() is used for defining classes in Ext JS.

SYNTAX:

Ext.define(class name, class members/properties, callback function);

- Class name is the name of class according to app structure e.g. appName.folderName.ClassName studentApp.view.StudentView.
- Class properties/members - which define the behavior of class.
- Callback function is optional. It is called when the class has loaded properly.

Example of Ext JS Class Definition

- Ext.define(studentApp.view.StudentDeatilsGrid, {
-   extend : 'Ext.grid.GridPanel',
-   id : 'studentsDetailsGrid',
-   store : 'StudentsDetailsGridStore',
-   renderTo : 'studentsDetailsRenderDiv',
-   layout : 'fit',
-   columns : [{
-     text : 'Student Name',
-     dataIndex : 'studentName'
-   },{
-     text : 'ID',
-     dataIndex : 'studentId'
-   },{
-     text : 'Department',
-     dataIndex : 'department'
-   }]
- });

Creating Objects

Like other OOPS based languages we can create objects in Ext JS as well. Different ways of creating objects in Ext JS:

1) Using new keyword:

- var studentObject = new student();
- studentObject.getStudentName();

2) Using Ext.create():

- Ext.create('Ext.Panel', {
-    renderTo : 'helloWorldPanel',
-    height : 100,
-    width : 100,
-    title : 'Hello world',
-    html :   'First Ext JS Hello World Program'
- });

USING INHERITANCE IN EXT JS

Inheritance is the principle of using functionality defined in class A into class B. In Ext JS inheritance can be done using two methods- Ext.extend:

- Ext.define(studentApp.view.StudentDetailsGrid, {
-    extend : 'Ext.grid.GridPanel',
-    ...
- });

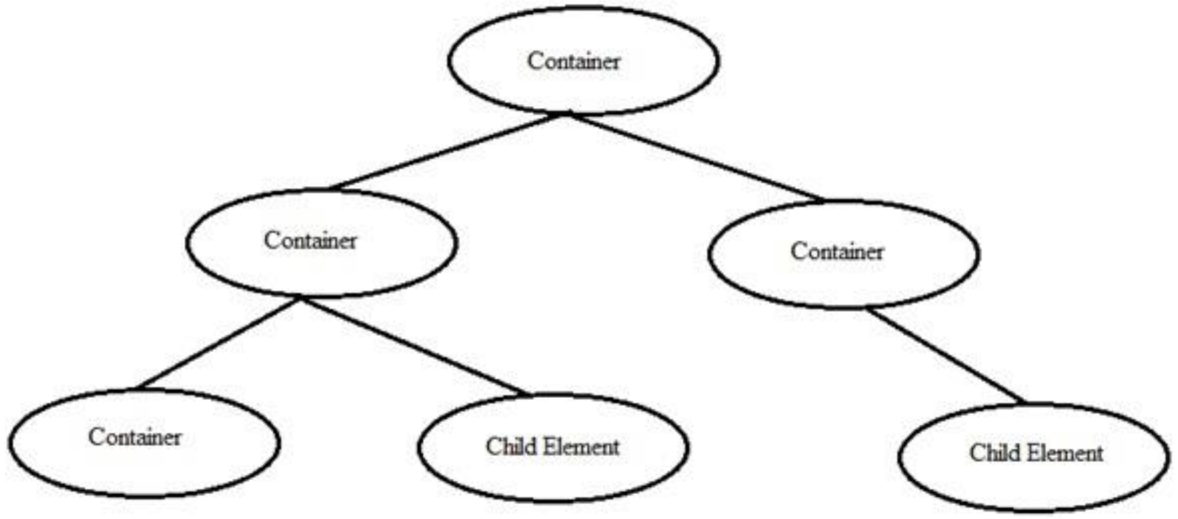
Here our custom class StudentDetailsGrid is using basic features of Ext JS class GridPanel.

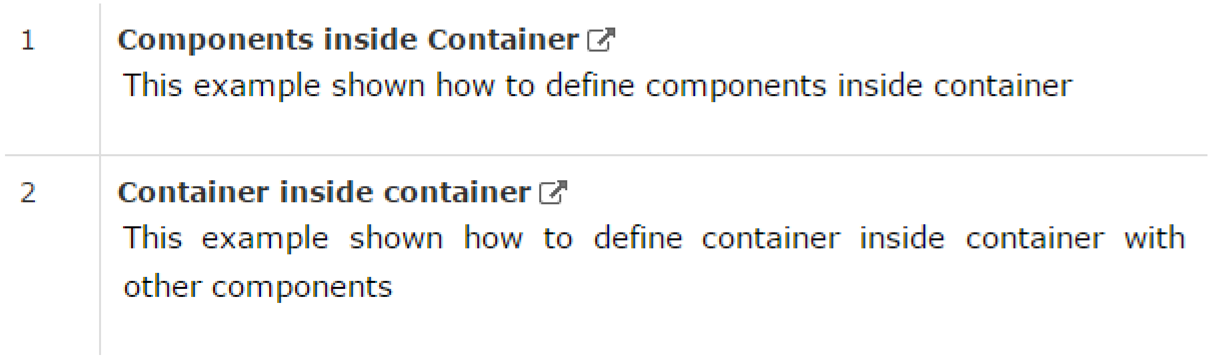
- Using Mixins: Mixins is the different way of using class A in class B without extend.

- mixins : {
-    commons : 'DepartmentApp.utils.DepartmentUtils'
- },
- Mixins we add in controller where we declare all the other classes such as store, view etc. In this way we can call DepartmentUtils class and use its functions in controller or in this application.

Ext.JS - Containers

- Container in Ext JS is the component where we can add other container or child components. These containers can have multiple layout to arrange the components in the containers. We can add or remove components from container and from its child elements. Ext.container.Container is the base class for all the containers in Ext JS.





1	**Components inside Container** This example shown how to define components inside container
2	**Container inside container** This example shown how to define container inside container with other components

Ext.JS - Layouts

- Layout is the way the elements are arranged in a container. That could be horizontal, vertical or any other. Ext JS has different layout defined in its library but we can always write custom layouts as well.

1) Absolute

- This layout allows to position the items using XY coordinates in the container.

2) Accordion

- This layout allows to place all the items in stack fashion (one on top of other) inside container.

3) Anchor

- This layout gives the privilege to the user to give the size of each element with respect to the container size.

4) Border

- In this layout various panels are nested and separated by borders.

5) Auto

- This is the default layout decides the layout of the elements based on the number of elements.

Ext.JS - Layouts

6) Card(TabPanel)

- This layout arranges different components in tab fashion. Tabs will be displayed on top of the container.Every time only one tab is visible and each tab is considered as different component.

7) Card(Wizard)

- In this layout every time the elements comes for full container space. There would be a bottom tool bar in the wizard for navigation.

8) Column

- This layout is to show multiple columns in the container. We can define fixed or percentage width to the columns. The percentage width will be calculated based on the full size of the container.

9) Fit

- In this layout the container is filled with a single panel and when there is no specific requirement related to the layout then this layout is used.

10) Table

- As name implies this layout arranges the components in container in the HTML table format.

11) vBox

- This layout allows the element to be distributed in the vertical manner. This is one of the mostly used layout.

12) hBox

- This layout allows the element to be distributed in the horizontal manner.

Ext.js - Components

1) Grid

- Grid component can be used to show the data in the tabular format.

2) Form

- Form widget is to get the data from the user.

3) Message Box

- Message box is basically used to show data in the form of alert box.

4) Chart

- Charts are used to reprent data in pictorial format.

5) Tool tip

- Tool tip is used to show some basic information when any event occurs.

6) Window

- This UI widget is to create a window which should pop up when any event occurs.

7) HTML editor

- HTML Editor is one of the very useful UI component which is used for styling the data which user enters in terms of fonts, color, size etc.

8) Progress bar

- To show the progress of the backend work

Drag & Drop

Drag and drop feature is one of the powerful feature added for making developers task easy.A drag operation, essentially, is a click gesture on some UI element while the mouse button is held down and the mouse is moved. A drop operation occurs when the mouse button is released after a drag operation.

Adding drag and drop class to the draggable targets.

- var dd = Ext.create('Ext.dd.DD', el, 'imagesDDGroup', {
-     isTarget: false
- });

Adding drag and drop target class to drappable target

- var mainTarget = Ext.create('Ext.dd.DDTarget', 'mainRoom', 'imagesDDGroup', {
-     ignoreSelf: false
- });

Ext.js - Themes

- Ext.js provides a number of themes to be used in your applications. You can add different theme inplace of classic theme and see the difference in output, this is done simply by replacing theme CSS file.

Ext.JS - Custom Events and Listeners

Events are something which get fired when something happens to the class. For example when a button is getting clicked or before/ after element is rendered.

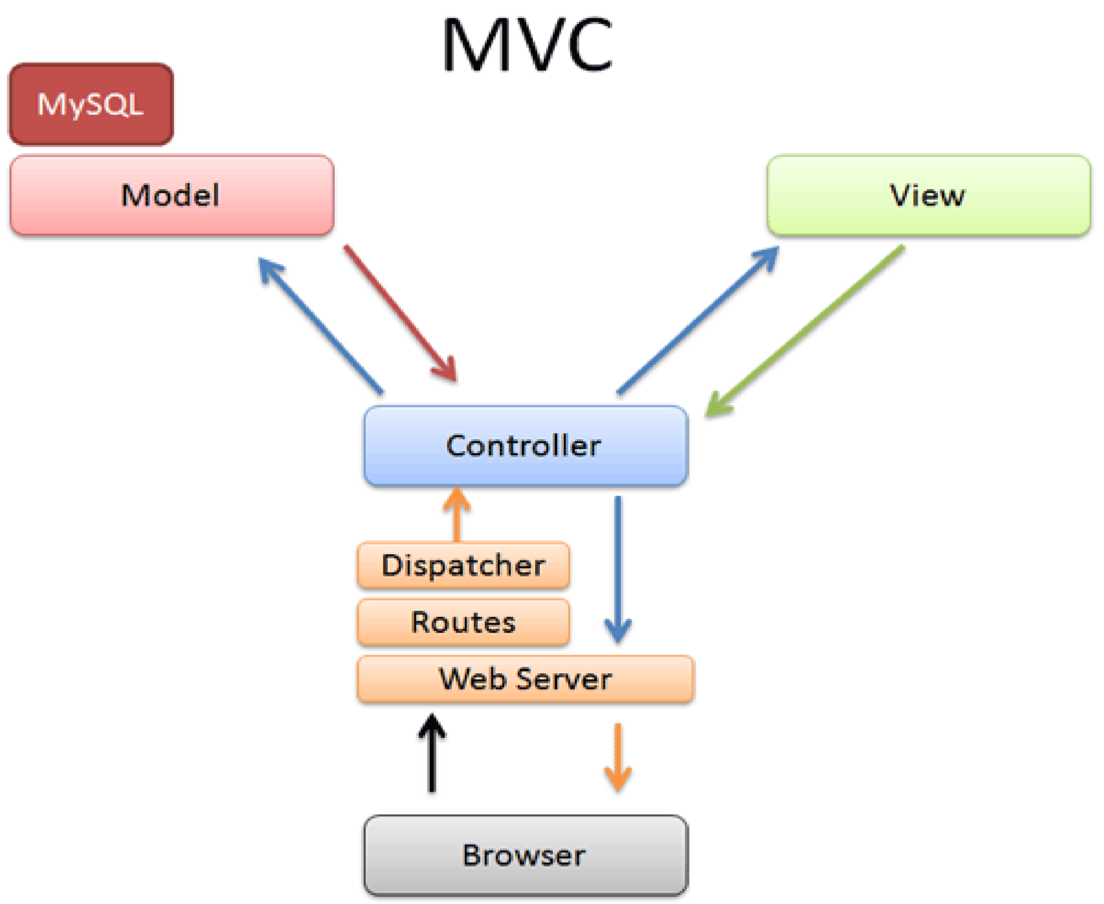
Methods of writing events:

- Built in events using listeners
- Attaching events later
- Custom events

Built in events using listeners

- Ext JS provides listener property for writing events and custom events in Ext JS files.
- Writing listener in Ext JS
- We will add the listener in the previous program itself by adding listen property to the panel a

SERVER SIDE MVC



Server Side Models

- Server Side Models are simple classes that house an 'instantiated' version of a resource record
  - Resource Records can be a row from a MySql Table, required parameters from another server public api, web service, etc
- These models should be transparent to the controller on how the raw data is represented, but rather be in a specified format for the controller

Server Side Models

- To facilitate how the model instantiates data, usually a map is present
- The Map is capable of understanding how to speak with the resource
  - "Select \`id\`, \`first\`, \`last\` from \`names\`......
- The model would then have member variables:
  - $model->id
  - $model->first
  - $model->last
  - ....

Server Side Models

- All of my models have key features
  - 1-to-1 resource mapping
  - $model->save()
  - $model->find()
  - $model->delete()
- Similar to CRUD operations except I leave save() to determine wether it is Create or Update
  - CRUD === 'Create Read Update Destroy'

Server Side Views

- Sever Side View classes, for most frameworks, take the model data and return the requested type of view
  - echo($view->buildTable(records));
- This buildTable() function is called by a controller, who then echo()'s the html generated by the view
- Has one major fault
  - What happens when I want to use this server side stack for mobile apps?
- Are there any performance flaws?

Server Side Control

- We have seen that how models and views work
  - These require some sort of delegation
- Controllers will receive the request from the client (old view), do any preprocessing, call the model (CRUD), use the model data, call the view, and return the html
- Within this return, we usually find JavaScript embedded as a code agent to 'enchance' our viewing pleasure.
- What if we mixed this up a bit and used JavaScript as our primary source of control?

Client Side JS with ExtJS

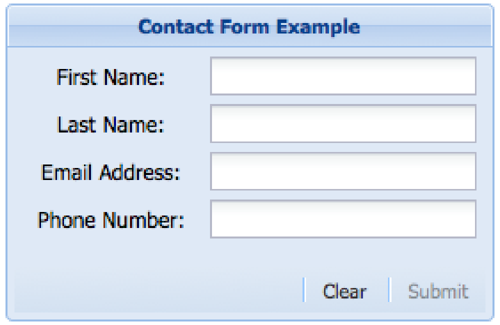
- MVC for JavaScript
- Exactly same process for server side stack, except we now try to use the server as little as possible
    - This allows for powerful client machines to do most of our processing and rendering
    - Only allow the client to manipulate data that can be considered hostile!

ExtJS Models

- The most important feature of ExtJS
  - Can have relational models!!!!!!!
  - Example:
    - Orders (a model) can have many Order Items (another model)
  - Each record of orders is stored in store
  - Each record of orders points to another store that has its Order items
  - This allows us to select an order, and then immediately have access to all its order items

ExtJS View

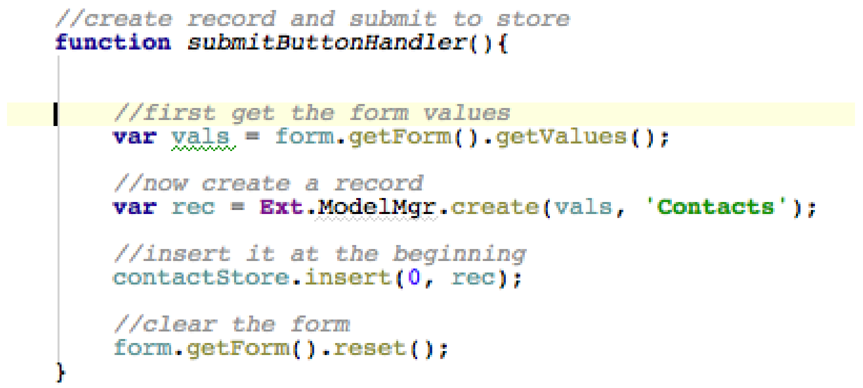
- Since this is JavaScript, we immediately expect robust GUI widgets



- Of course, you can add CSS and style them

ExtJS Control

- JavaScript is a functional language
  - This allows for very strong and very easy control logic



```
//create record and submit to store
function submitButtonHandler(){

    //first get the form values
    var vals = form.getForm().getValues();

    //now create a record
    var rec = Ext.ModelMgr.create(vals, 'Contacts');

    //insert it at the beginning
    contactStore.insert(0, rec);

    //clear the form
    form.getForm().reset();
}
```

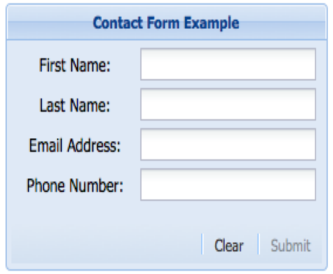
  - Of course, you can still use OOP style if desired

SO HOW DOES THIS ALL WORK?

- By using MVC on the client side:
  - We only need to contact the server when using CRUD operations
  - By control logic when otherwise needed
- Lets go through an example

Simple Form with Grid

- Our goal will be to make a form, that upon submit, updates a local store
- This store feeds a grid and will automatically update when a new record is inserted



Contact Form Example

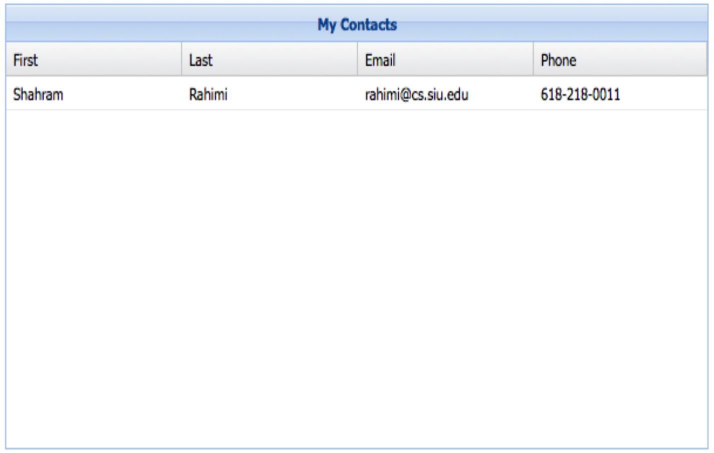
First Name:

Last Name:

Email Address:

Phone Number:

Clear | Submit

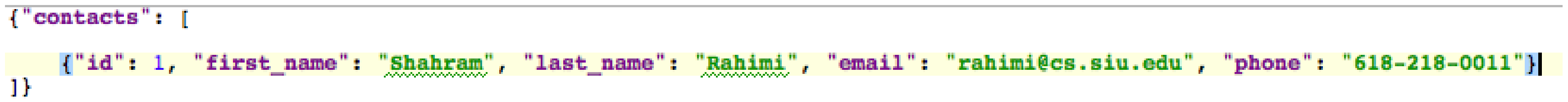


My Contacts

First	Last	Email	Phone
Shahram	Rahimi	rahimi@cs.siu.edu	618-218-0011

OUR DATA WE WILL WORK WITH

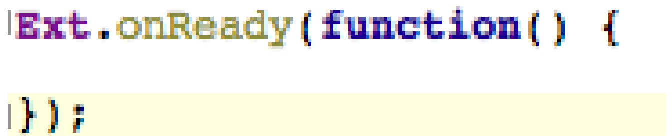
- JSON (JavaScript Object Notation)



```
{"contacts": [

    {"id": 1, "first_name": "Shahram", "last_name": "Rahimi", "email": "rahimi@cs.siu.edu", "phone": "618-218-0011"}
]}
```

Configure Script

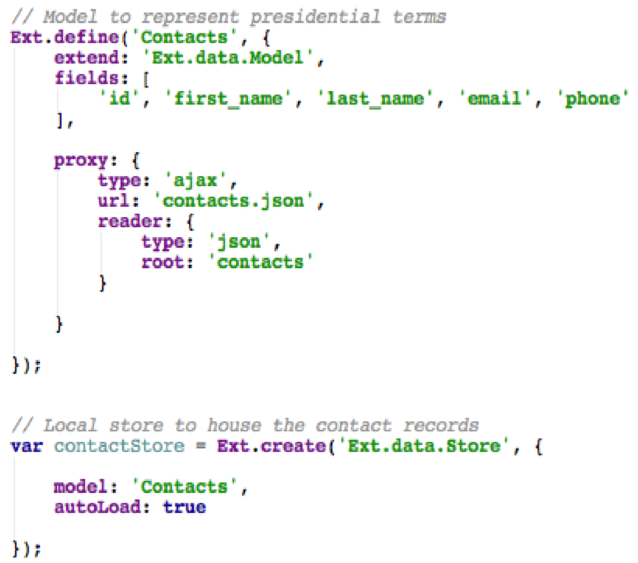
- We first must wrap all of our JavaScript within an onLoad() function



```
Ext.onReady(function() {

});
```

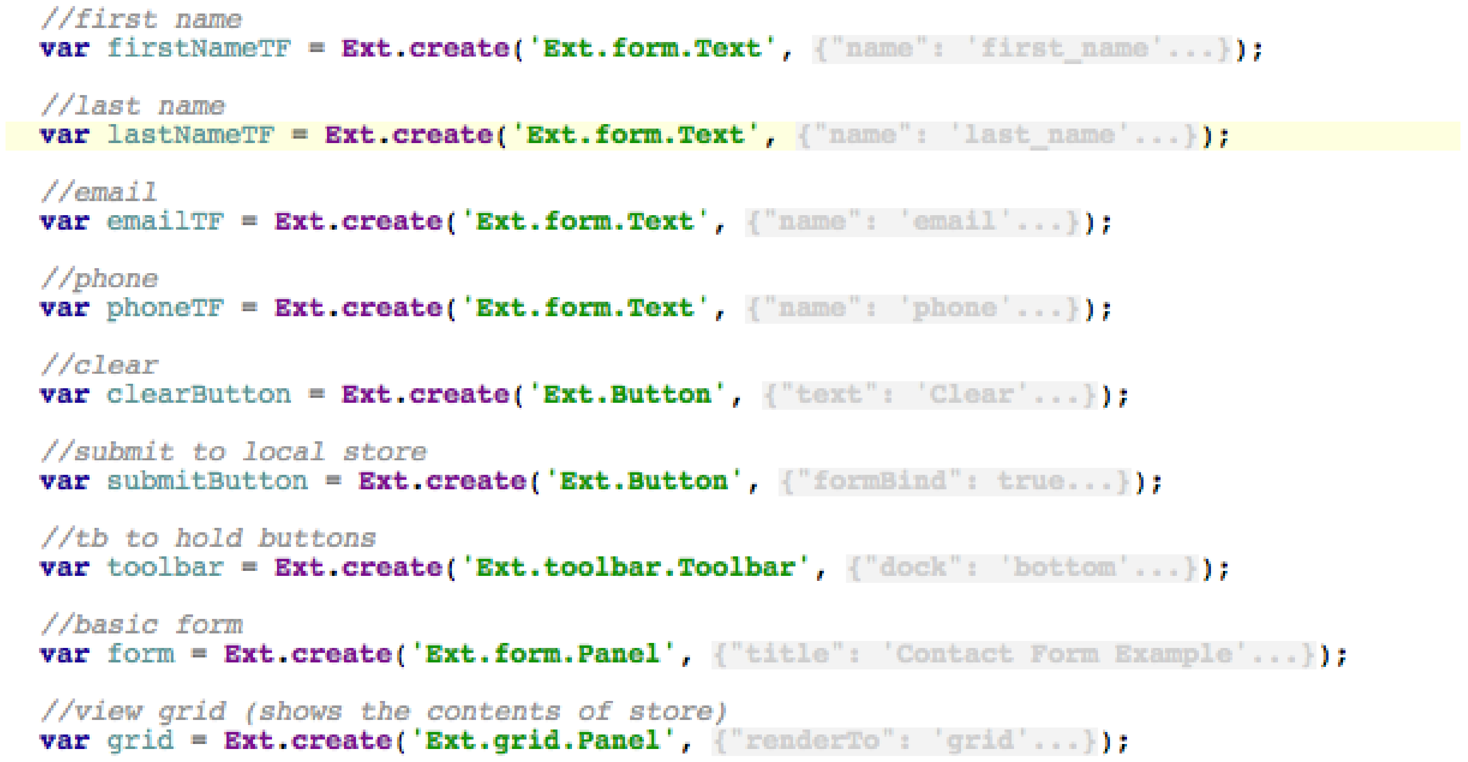
Create the Model

- Typically, we will always represent modeled data, so it seems wise to start there
- Similar to the server stack, we use the model to communicate with the resource
  - In this case, a flat file named contacts.json



```
// Model to represent presidential terms
Ext.define('Contacts', {
    extend: 'Ext.data.Model',
    fields: [
        'id', 'first_name', 'last_name', 'email', 'phone'
    ],

    proxy: {
        type: 'ajax',
        url: 'contacts.json',
        reader: {
            type: 'json',
            root: 'contacts'
        }

    }

});


// Local store to house the contact records
var contactStore = Ext.create('Ext.data.Store', {

    model: 'Contacts',
    autoLoad: true

});
```

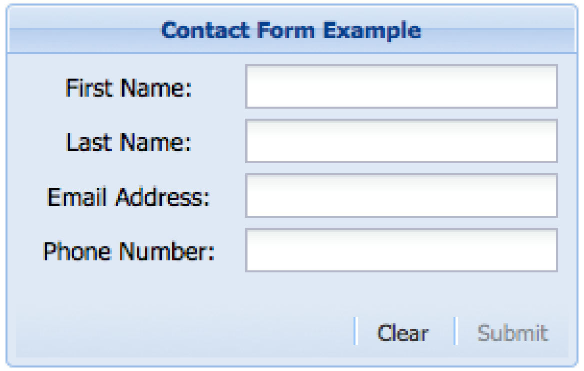
Create the View



```
//first name
var firstNameTF = Ext.create('Ext.form.Text', {"name": 'first_name'...});

//last name
var lastNameTF = Ext.create('Ext.form.Text', {"name": 'last_name'...});

//email
var emailTF = Ext.create('Ext.form.Text', {"name": 'email'...});

//phone
var phoneTF = Ext.create('Ext.form.Text', {"name": 'phone'...});

//clear
var clearButton = Ext.create('Ext.Button', {"text": 'Clear'...});

//submit to local store
var submitButton = Ext.create('Ext.Button', {"formBind": true...});

//tb to hold buttons
var toolbar = Ext.create('Ext.toolbar.Toolbar', {"dock": 'bottom'...});

//basic form
var form = Ext.create('Ext.form.Panel', {"title": 'Contact Form Example'...});

//view grid (shows the contents of store)
var grid = Ext.create('Ext.grid.Panel', {"renderTo": 'grid'...});
```

Create the View

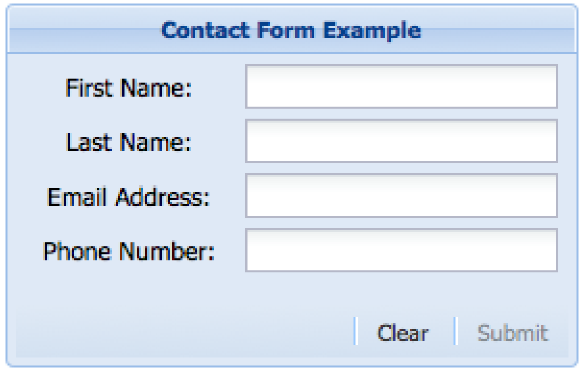
```
//first name
var firstNameTF = Ext.create('Ext.form.Text', {

    fieldLabel: 'First Name',
    name: 'first_name',
    validator: nameValidator

});

//last name
var lastNameTF = Ext.create('Ext.form.Text', {

    fieldLabel: 'Last Name',
    name: 'last_name',
    validator: nameValidator

});
```



Create the View

```
//email
var emailTF = Ext.create('Ext.form.Text', {

    fieldLabel: 'Email Address',
    name: 'email',
    validator: emailValidator

});
```

```
//phone
var phoneTF = Ext.create('Ext.form.Text', {

    fieldLabel: 'Phone Number',
    name: 'phone',
    validator: phoneValidator

});
```

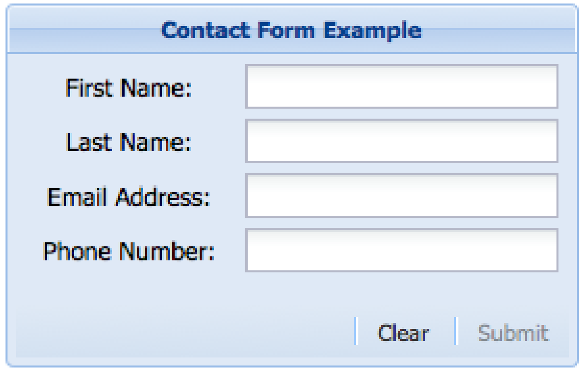


Contact Form Example

First Name:	
Last Name:	
Email Address:	
Phone Number:	

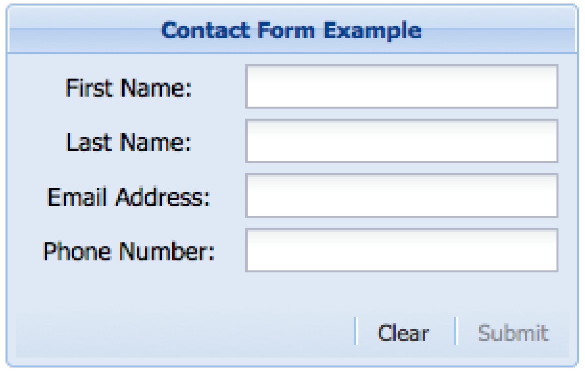
Create the View

```
...

//clear
var clearButton = Ext.create('Ext.Button', {

    text: 'Clear',
    handler: clearButtonHandler
});

//submit to local store
var submitButton = Ext.create('Ext.Button', {

    formBind: true,
    text: 'Submit',
    handler: submitButtonHandler
});
```

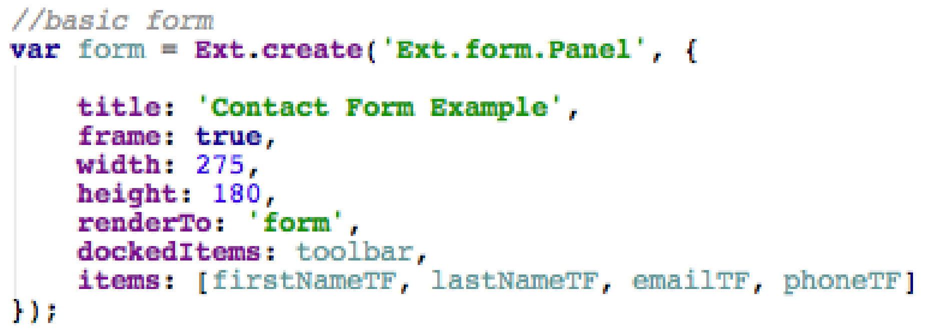


Create the View

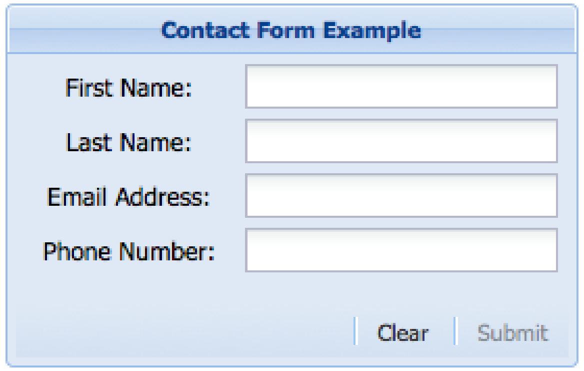
```
//tb to hold buttons
var toolbar = Ext.create('Ext.toolbar.Toolbar', {

    dock: 'bottom',
    items: ['->', '-', clearButton, '-', submitButton]
});
```



Create the View

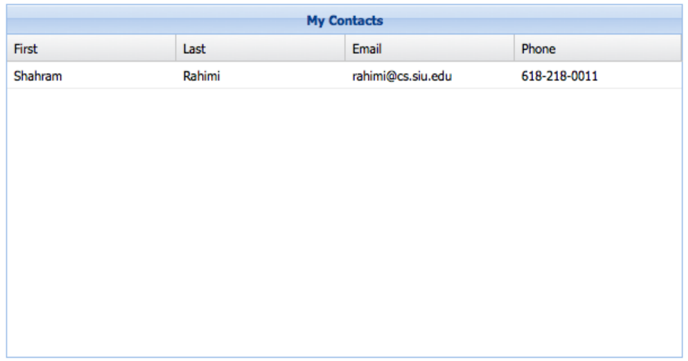


```
//basic form
var form = Ext.create('Ext.form.Panel', {

    title: 'Contact Form Example',
    frame: true,
    width: 275,
    height: 180,
    renderTo: 'form',
    dockedItems: toolbar,
    items: [firstNameTF, lastNameTF, emailTF, phoneTF]
});
```



Create the View

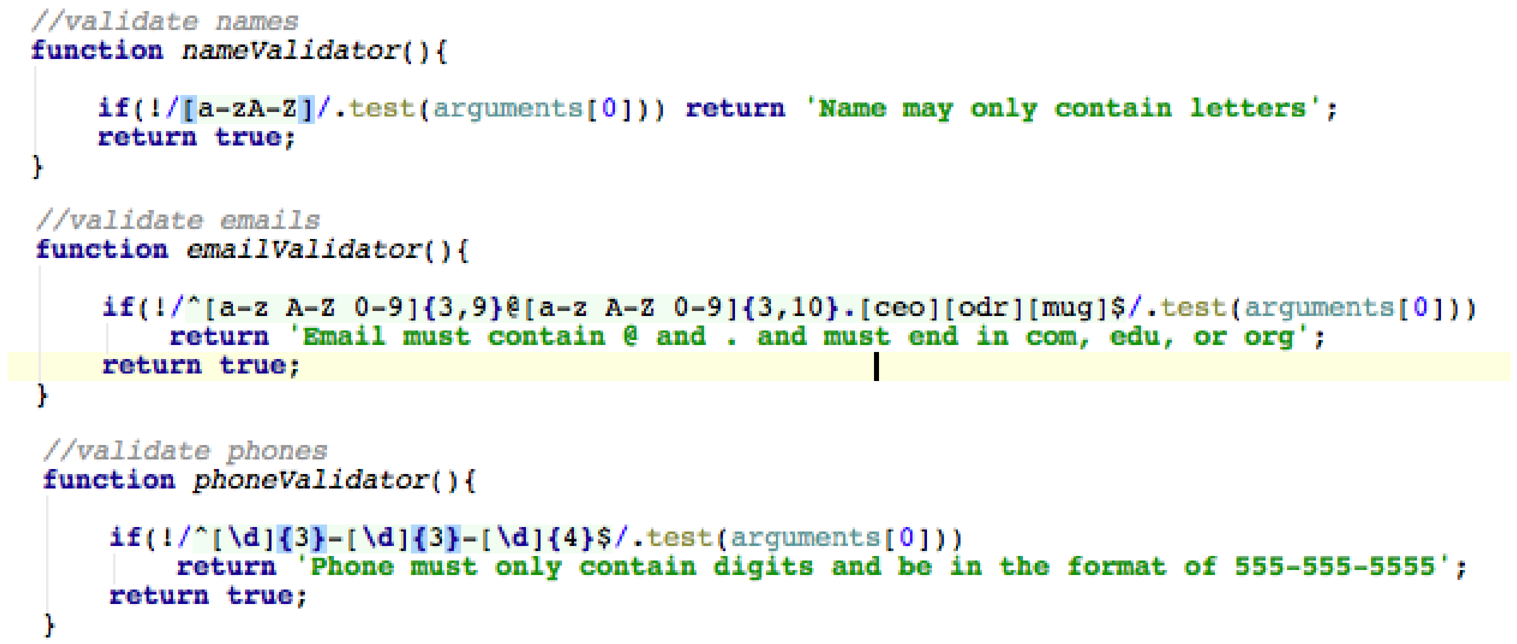
```
//view grid (shows the contents of store)
var grid = Ext.create('Ext.grid.Panel', {

    store: contactStore,
    renderTo: 'grid',

    width: 600,
    height: 300,
    autoScroll: true,

    title: 'My Contacts',
    columns:[
        {header: 'First', dataIndex: 'first_name', width: 150},
        {header: 'Last', dataIndex: 'last_name', width: 150},
        {header: 'Email', dataIndex: 'email', width: 150},
        {header: 'Phone', dataIndex: 'phone', flex: 1}
    ]
});
```



My Contacts

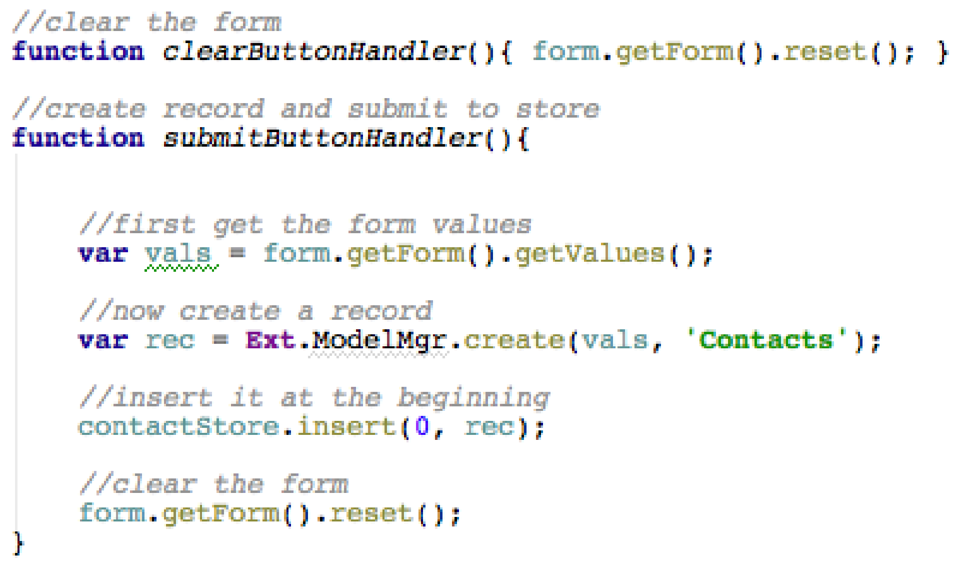
First	Last	Email	Phone
Shahram	Rahimi	rahimi@cs.siu.edu	618-218-0011

TIE IN THE CONTROL



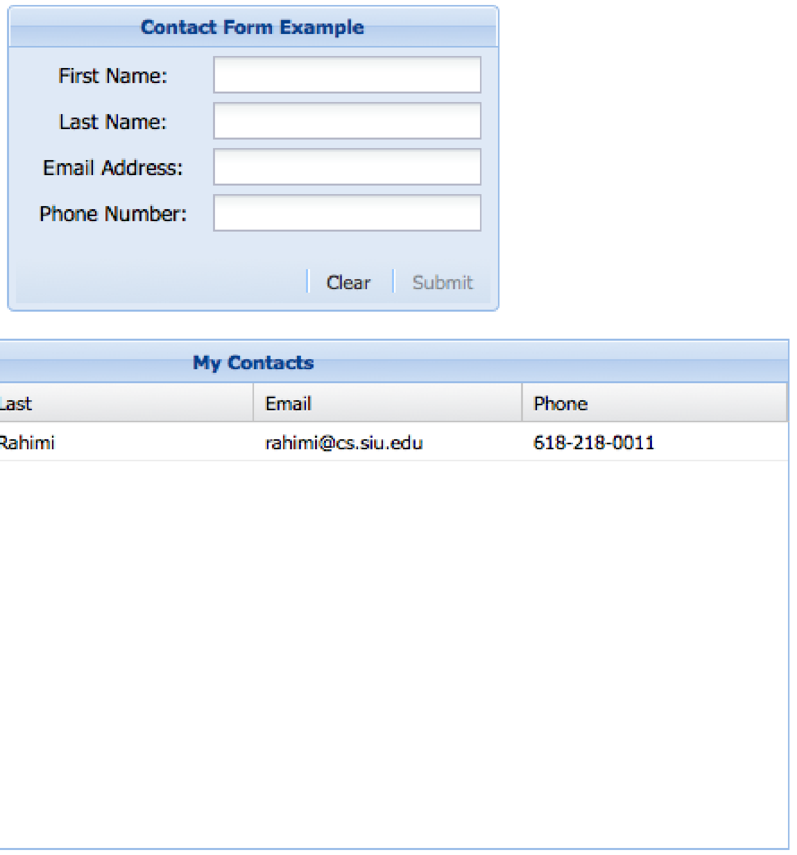
```
//validate names
function nameValidator(){

    if(!/[a-zA-Z]/.test(arguments[0])) return 'Name may only contain letters';
    return true;
}

//validate emails
function emailValidator(){

    if(!/^[a-z A-Z 0-9]{3,9}@[a-z A-Z 0-9]{3,10}.[ceo][odr][mug]$/.test(arguments[0]))
        return 'Email must contain @ and . and must end in com, edu, or org';
    return true;
}

//validate phones
function phoneValidator(){

    if(!/^[\d]{3}-[\d]{3}-[\d]{4}$/.test(arguments[0]))
        return 'Phone must only contain digits and be in the format of 555-555-5555';
    return true;
}
```

TIE IN THE CONTROL



```
//clear the form
function clearButtonHandler(){ form.getForm().reset(); }

//create record and submit to store
function submitButtonHandler(){


    //first get the form values
    var vals = form.getForm().getValues();

    //now create a record
    var rec = Ext.ModelMgr.create(vals, 'Contacts');

    //insert it at the beginning
    contactStore.insert(0, rec);

    //clear the form
    form.getForm().reset();
}
```

Final Product



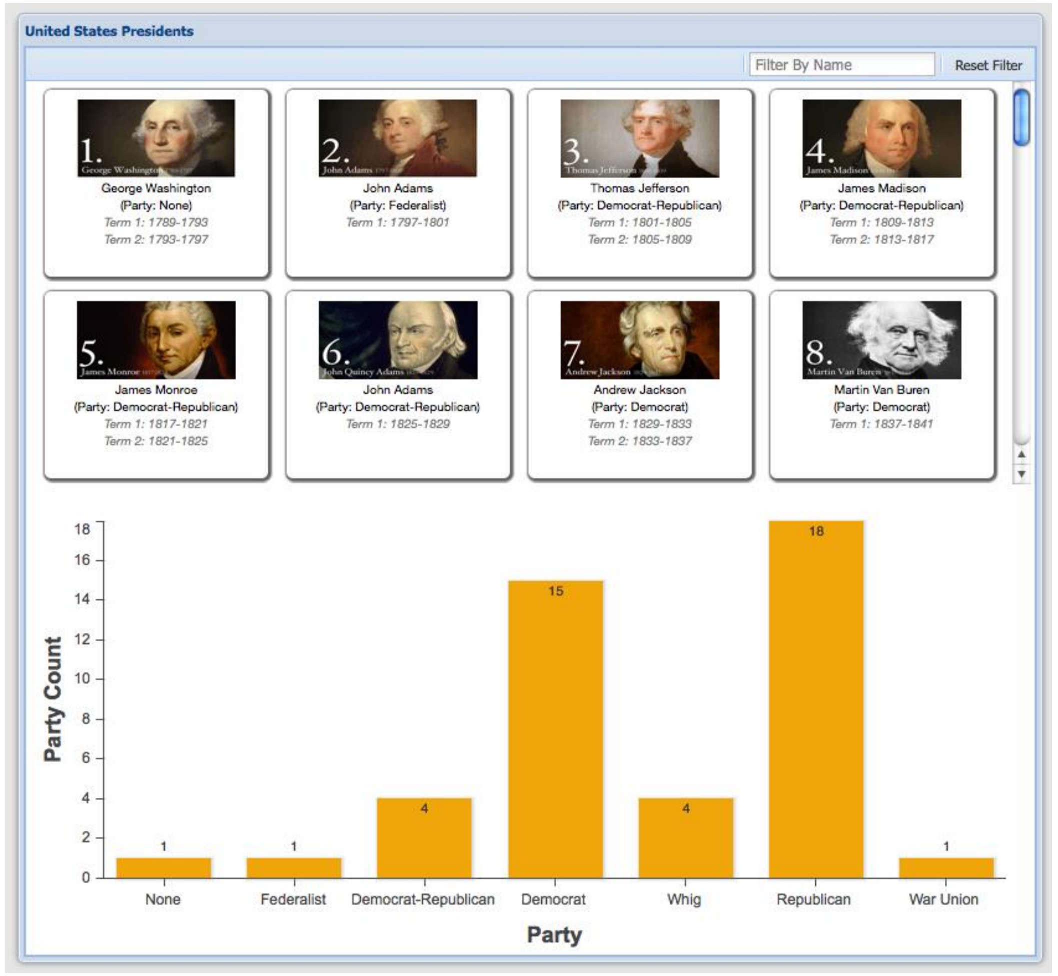
Contact Form Example

First Name:	
Last Name:	
Email Address:	
Phone Number:	

My Contacts

First	Last	Email	Phone
Shahram	Rahimi	rahimi@cs.siu.edu	618-218-0011

Now lets create this



1. George Washington
George Washington
(Party: None)
Term 1: 1789-1793
Term 2: 1793-1797

2. John Adams
John Adams
(Party: Federalist)
Term 1: 1797-1801

3. Thomas Jefferson
Thomas Jefferson
(Party: Democrat-Republican)
Term 1: 1801-1805
Term 2: 1805-1809

4. James Madison
James Madison
(Party: Democrat-Republican)
Term 1: 1809-1813
Term 2: 1813-1817

5. James Monroe
James Monroe
(Party: Democrat-Republican)
Term 1: 1817-1821
Term 2: 1821-1825

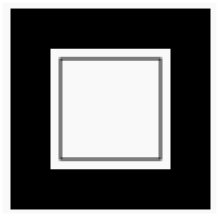
6. John Quincy Adams
John Adams
(Party: Democrat-Republican)
Term 1: 1825-1829

7. Andrew Jackson
Andrew Jackson
(Party: Democrat)
Term 1: 1829-1833
Term 2: 1833-1837

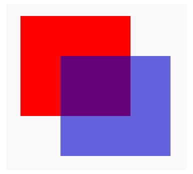
8. Martin Van Buren
Martin Van Buren
(Party: Democrat)
Term 1: 1837-1841

JavaScript Exercise:

1) Write a JavaScript program to draw the following rectangular shape.

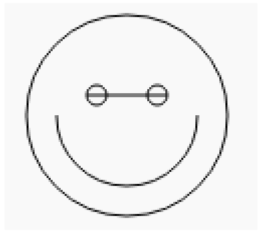


2) Write a JavaScript program to draw two intersecting rectangles, one of which has alpha transparency.

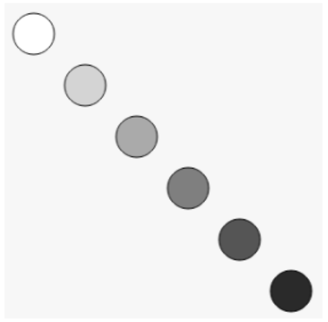


JavaScript Exercise:

- 3) Write a JavaScript program to draw the following diagram [use moveto() function].



- 4) Write a JavaScript program to draw the following diagram [diagonal, white to black circles]



JavaScript Exercise:

5) Write a JavaScript program to display the current day and time in the following format:

Sample Output : Today is : Friday.

Current time is : 4 PM : 50 : 22

JavaScript Exercise…

6) Write a JavaScript function to find the area of a triangle where lengths of the three of its sides are 5, 6, 7.

7) Write a JavaScript function to convert a string into camel case.
Test Data :
console.log(camelize("JavaScript Exercises"));
console.log(camelize("JavaScript exercises"));
console.log(camelize("JavaScriptExercises"));
"JavaScriptExercises"
"JavaScriptExercises"
"JavaScriptExercises"

JavaScript Exercise…

8) Write a JavaScript program to determine whether a given year is a leap year in the Gregorian calendar.

9) Write a JavaScript program to reverse a given string.

Input sample = “Forrest”

Output sample = “tserroF”

JavaScript Exercise…

10) Write a JavaScript program to calculate the factorial of a number.

(In mathematics, the factorial of a non-negative integer n, denoted by n!, is the product of all positive integers less than or equal to n. For example, 5! = 5 x 4 x 3 x 2 x 1 = 120)

JavaScript Exercise…

11) Write a JavaScript function that generates all combinations of a string.

- *Example string :* 'dog'
  Expected Output : d,do,dog,o,og,g

- 12) Write a JavaScript function to check to check whether a variable is numeric or not. *Test Data :*

```
console.log(is_Numeric(12));
console.log(is_Numeric('abcd'));
console.log(is_Numeric('12'));
console.log(is_Numeric(' '));
console.log(is_Numeric(1.20));
console.log(is_Numeric(-200));
```

Output :

```
true
false
true
false
true
true
```

JavaScript Exercise…

13) Write a JavaScript function to get all prime numbers from 0 to a specified number.

- Test Data :
- console.log(primeFactorsTo(5));
- [2, 3, 5]
- console.log(primeFactorsTo(15));
- [2, 3, 5, 7, 11, 13]

JavaScript Exercise.

14) Create a simple homepage with at least 2 child pages linked to it. Add the following components throughout the pages:

1) Grid 2) Form 3) Message Box 4) Charts

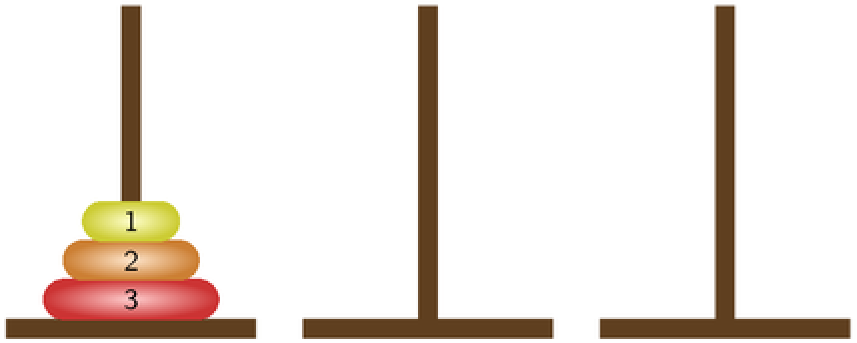
5) Tool tip , 6) Window

Try to design the pages as user friendly as possible using Ext JS. Add text and graphics contents. Chose a suitable built in theme in Ext Js or use customized theme.

Add events and event listener to the pages.

Java Exercises...

1) How to use method for solving Tower of Hanoi problem for 3 discs?



Sample input: Disk num: 3
Sample output:

```
Disk 1 from A to C
Disk 2 from A to B
Disk 1 from C to B
Disk 3 from A to C
Disk 1 from B to A
Disk 2 from B to C
Disk 1 from A to C
```

JAVA EXERCISES

- 2) Solve the producer–consumer problem, coding in Java.
- The problem describes two processes, the producer and the consumer, who share a common, fixed-size buffer used as a queue. The producer's job is to generate data, put it into the buffer, and start again. At the same time, the consumer is consuming the data (i.e., removing it from the buffer), one piece at a time. The problem is to make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer.
- The solution for the producer is to either go to sleep or discard data if the buffer is full. The next time the consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again. In the same way, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer. The solution can be reached by means of inter-process communication, typically using semaphores. An inadequate solution could result in a deadlock where both processes are waiting to be awakened. The problem can also be generalized to have multiple producers and consumers.

Sample output.

```
Producer #1 put: 0
Consumer #1 got: 0
Producer #1 put: 1
Consumer #1 got: 1
Producer #1 put: 2
Consumer #1 got: 2
Producer #1 put: 3
Consumer #1 got: 3
Producer #1 put: 4
….
….
….
```