CS 696 Emerging Web and Mobile Technologies
Spring Semester, 2011
Doc 15 Socket.IO & Sencha Touch
Mar 10, 2011

References

Sencha Touch API Documentation, http://dev.sencha.com/deploy/touch/docs/, http://www.sencha.com/learn/Sencha_Touch

Socket.IO, http://socket.io/

Various web pages as noted on individual slides.

Socket.IO

Problems with Ajax

Client has to request data

Comet allows directional communication

WebSockets simplifies the bi-directional connections

But not available on all browsers

Socket.IO

JavaScript library for Client server communication

Detects and uses the most capable transport supported by browser

- WebSocket
- Adobe® Flash® Socket
- AJAX long polling
- AJAX multipart streaming
- Forever Iframe
- JSONP Polling

Supported Browsers

Internet Explorer 5.5 - 8
Safari 3 - 5
Google Chrome 4 - 6
Firefox 3-4
Opera 10.61

iPhone Safari
iPad Safari
Android WebKit
WebOs WebKit

Client side communication

```
var socket = new io.Socket(node_server_url);
socket.connect();

socket.send(message);

// callback when connection is made
socket.on('connect', function(){ … })

// callback on receiving messge from server
socket.on('message', function(){ … })

// when disconnected
socket.on('disconnect', function(){ … })

socket.disconnect();
```

Async Communication

```
 var socket = new io.Socket(node_server_url);
 socket.connect();

socket.send("Hi Mom");

 socket.on('message', function(messageFromServer){
     alert(messageFromServer);
})
```

Response to send
Come in on('message")

Server can send message anytime

Server Side

Server has to be ready to handle variety of connections

- WebSocket
- Adobe® Flash® Socket
- AJAX long polling
- AJAX multipart streaming
- Forever Iframe
- JSONP Polling

Servers Support in Socket.IO

Node.js

Tornado (Python)

https://github.com/MrJoes/tornadio

socketio-java

Implemented in Servlets

http://code.google.com/p/socketio-java/

Go

https://github.com/madari/go-socket.io

Rack (Ruby Weberver)

https://github.com/markjeee/Socket.IO-rack

Example

Client connects to Server

Every second server sends integer - number messages sent

Client Side

```
<!doctype html>
<html>
  <head>
    <title>Timer client test</title>
    <script src="http://127.0.0.1:8080/json.js"></script>
    <script src="http://127.0.0.1:8080/socket.io/socket.io.js"></script>
  </head>
  <body>
    <script>
      var socket = new io.Socket(null, {port: 8080});
      socket.connect();
      socket.on('message', function(message){
                document.getElementById('text').value = message.announcement;
      });
    </script>
    <h1>Sample client</h1> <input type="text" id="text">
  </body>
</html>
```

Server Side using Node.js

```
var http = require('http') , io = require('../'), url = require('url') , fs = require('fs')
  , timer = require('timers') , server;

server = http.createServer(function(request, response){
 var path = url.parse(request.url).pathname;
  switch (path){
    case '/json.js':
    case '/chat.html':
      console.log("get file: " + path );
      fs.readFile(__dirname + path, function(err, data){
        if (err) return send404(res);
        response.writeHead(200, {'Content-Type': path == 'json.js' ? 'text/javascript' : 'text/html'})
        response.write(data, 'utf8');
        response.end();
      });
      break;
  }}
),

server.listen(8080);
```

Server Side using Node.js

```
var io = io.listen(server)
var count = 1;

io.on('connection', function(client){
  client.send({ announcement: count++ });
  client.broadcast({ announcement: client.sessionId + ' connected' });
      count = 1;
});

timer.setInterval(clock, 1000);
function clock() {
    io.broadcast({ announcement: "" + count++ });
}
```

Web verses Application development

Web-based approaches

jQuery

- Javascript & CSS libraries for Web

jQuery Mobile, JQTouch

- Just Extend the libraries for Mobile browsers
- Mobile Look and Feel

PhoneGap

- Native application running Web view
- Just web page in native app

Web-based approaches - Benefits

Use Web skills & developers

Cross platform

Web app and native app

Designers & their tools

Web-based approaches - Cons

Performance

Emulate native widgets

HTML & CSS limitations

Access to platform features

Different style of development

Browser

Interpreter for applications

Application-based Approaches

Sencha Touch

Webpages and mobile apps

JavaScript development

MVC

JavaScript classes for Widgets

JavaScript create DOM objects

Titanium Appcelerator

Mobile apps only

Javascript development

MVC

JavaScript UI widgets

Control native widgets

Application-based Approaches

More like traditional application development

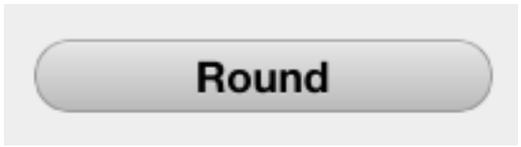
Construct interface by creating objects

Object interact

Library of UI widgets

- Panels
- Containters
- Buttons
- Layout managers

Button - jQuery Mobile

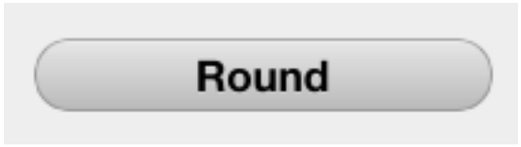


HTML button/element

CSS - looks

JavaScript added to element
onclick

Button - Sencha Touch

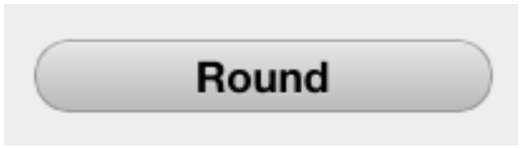


HTML button/element
CSS - looks

JavaScript Button object

Creates HTML button
Contains JavaScript logic
Interacts with other JavaScript objects
Can use some native widgets

Button - Appcelerator



Native Button

JavaScript Button

Contains JavaScript logic
Interacts with other JavaScript objects

If you only have a hammer everything looks like a nail

Sencha Touch

Sencha Touch http://www.sencha.com/products/touch/

Supports

- iOS (iPhone, iPod Touch, iPad)
- Android
- Blackberry 6 (soon)

Native iOS look

Themes for Android

Documentation

http://www.sencha.com/learn/Sencha_Touch

The Sencha Touch Web Page

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <meta name="viewport" content="width=device-width; initial-scale=1.0; maximum-
scale=1.0; minimum-scale=1.0; user-scalable=0;" />
    <title>Button</title>
    <link rel="stylesheet" href="sencha-touch-debug.css" type="text/css">
    <script type="text/javascript" src="sencha-touch-debug-w-comments.js"></script>
    <script type="text/javascript" src="index.js">
    </script>
</head>
<body></body>
</html>
```


I am using the debug version of the CSS & JavaScript libraries

Hello World Example

index.js

```
Ext.setup({
onReady: function(){
  new Ext.Panel({
        fullscreen: Ext.is.iPhone,
        items:[
          new Ext.Button({
            text: 'Hello World'
          })
        ]
      });
  }
});
```

Overview

```
Ext.setup({
onReady: function(){
  new Ext.Panel({
        fullscreen: Ext.is.iPhone,
        items:[
          new Ext.Button({
            text: 'Hello World'
          })
        ]
    });
  }
});
```

Ext

- JavaScript object
- Namespace/Model
- Contains Sencha Touch library

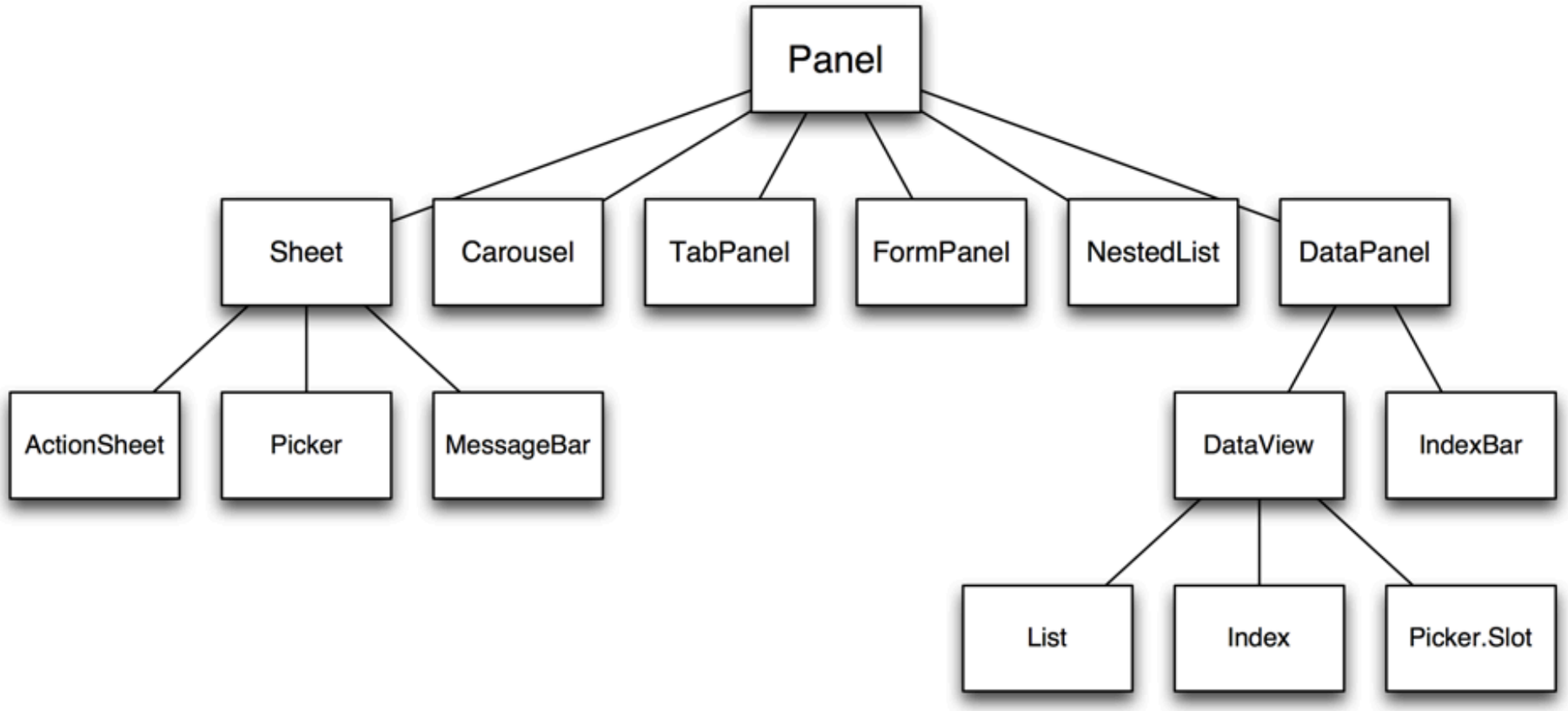
Entire interface & logic

- JavaScript
- Using Library objects

Sencha Touch Major parts

Panels
Layouts
Templates
Form Elements
Data Elements
Listeners

Panels



source: https://github.com/nelstrom/Sencha-Touch-panels-demo

Useful Panel Properties

fullscreen

- Should panel take up entire available area

layout

- How to layout contents
- Specify which Layout manager to use

items

- Item(s) to add to panel
- Object or string literal

dockedItems

- Items to be added boundry of panel
- Toolbars & tab bars

html

- Added to panel

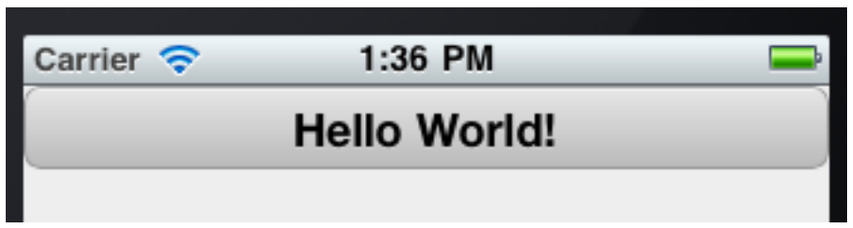
there are a lot more properties

Object Verses String Representation

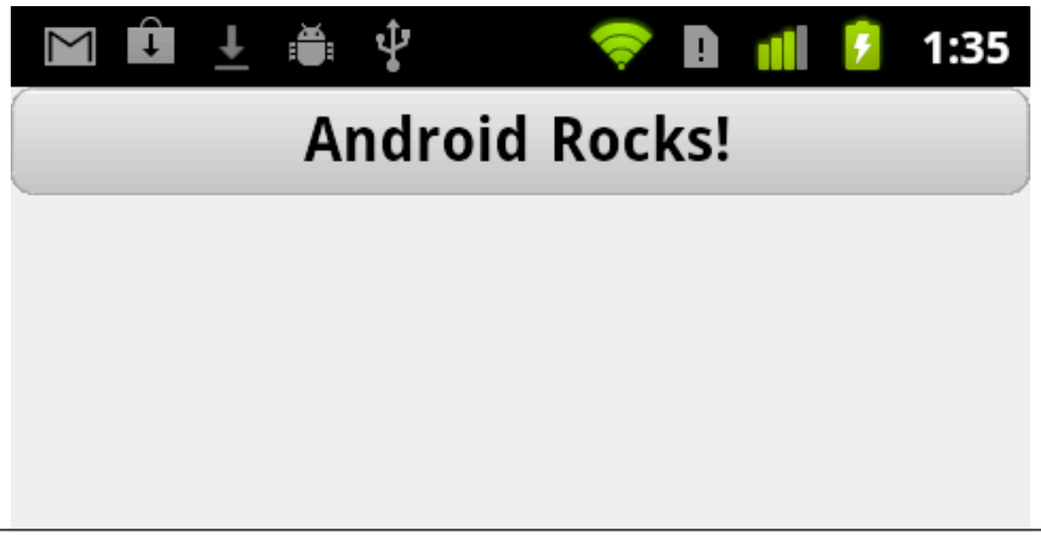
```
Ext.setup({
onReady: function(){
  new Ext.Panel({
      fullscreen: Ext.is.iPhone,
      items:[
        new Ext.Button({
          text: 'Hello World'
        })
      ]
    });
  }
});
```

```
Ext.setup({
onReady: function(){
  new Ext.Panel({
   fullscreen: true,
      items: [
          {text: "Hello World!", xtype:
"button" },
          {text: "How are You?", xtype:
"button" }
      ]
    });
  }
});
```

Selecting Platform




```
Ext.setup({
onReady: function(){
  new Ext.Panel({
        fullscreen: Ext.is.iPhone,
       items:{text: "Hello World!", xtype: "button" }
    });
  new Ext.Panel({
        fullscreen: Ext.is.Android,
          items:{text: "Android Rocks!", xtype: "button" }
      });
   }
});
```



can also use Ext.is.iOS, Ext.is.iPad, Ext.is.iPod

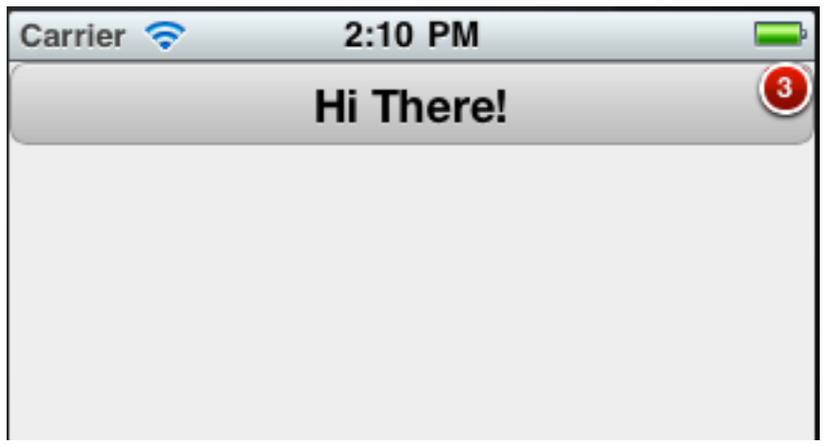
Startup images

```
 Ext.setup({
tabletStartupScreen: 'resources/img/tablet_startup.png',
phoneStartupScreen: 'resources/img/phone_startup.png',
icon: 'resources/img/icon.png',
glossOnIcon: false,

 onReady: function(){
   new Ext.Panel({
    fullscreen: true,
       items: {text: "Hello World!", xtype: "button" },
    });
  }
});
```


also tabletIcon and phoneIcon

Some Button Options

```
function react(button, event) {
        alert("you clicked");
}

var hello = new Ext.Button({
        text: "Hi There!",
        badgeText: "3",
        handler: react});

Ext.setup({
        onReady: function(){
                new Ext.Panel({
                        fullscreen: Ext.is.iPhone,
                        items:[hello]
                });
    }
});
```



Configuration verses Methods

```
var hello = new Ext.Button({
        text: "Hi There!",
        badgeText: "3"});
```

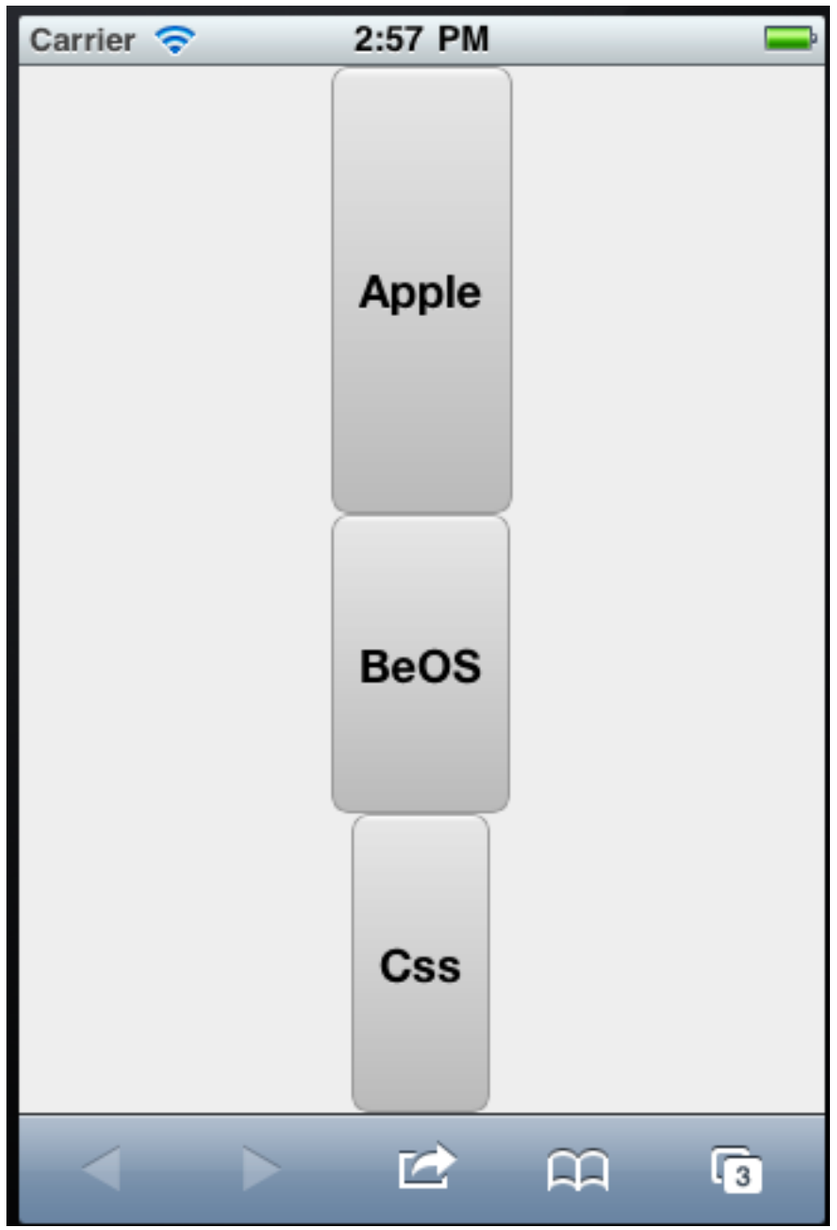
```
var hello = new Ext.Button();

hello.text = "Hi There!";
hello.setBadge("2");
```

Layout

```
Ext.setup({
    onReady: function() {
        new Ext.Panel ({
            fullscreen: true,
            layout: {
                type: 'hbox',
                pack: 'center',
                align: 'center'
                },
            defaults: {xtype: 'button'},
            items: [
                {text: "Apple", flex: "3" },
                {text: "BeOS", flex: "2" },
                {text: "Css", flex: "2" }
            ]
        })
    }
});
```

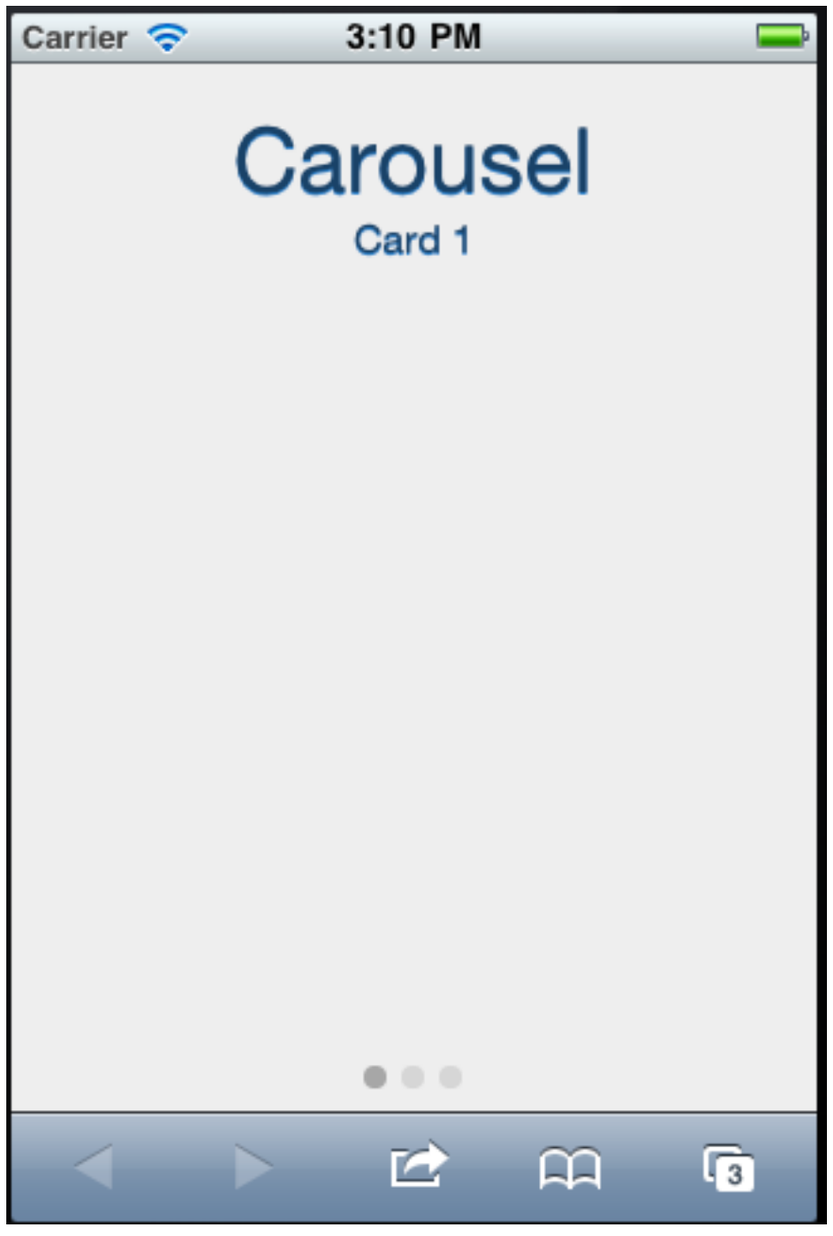
Vbox & hbox

```
Ext.setup({
    onReady: function() {
        new Ext.Panel ({
            fullscreen: true,
            layout: {
                type: 'vbox',
                pack: 'center',
                align: 'center'
                },
            defaults: {xtype: 'button'},
            items: [
                {text: "Apple", flex: "3" },
                {text: "BeOS", flex: "2" },
                {text: "Css", flex: "2" }
            ]
        })
    }
});
```



Carousel

```
Ext.setup({
    onReady: function() {
        var carousel1 = new Ext.Carousel({
            defaults: {cls: 'card'},
            items: [{html: '<h1>Carousel</h1><p>Card 1</p>'},
            {title: 'Tab 2', html: 'Card 2'},
            {title: 'Tab 3', html: 'Card 3'}]
        });
    new Ext.Panel({
            fullscreen: true,
            layout: {
                type: 'vbox',
                align: 'stretch'
            },
            defaults: { flex: 1},
            items: [carousel1]
        });
    }
});
```

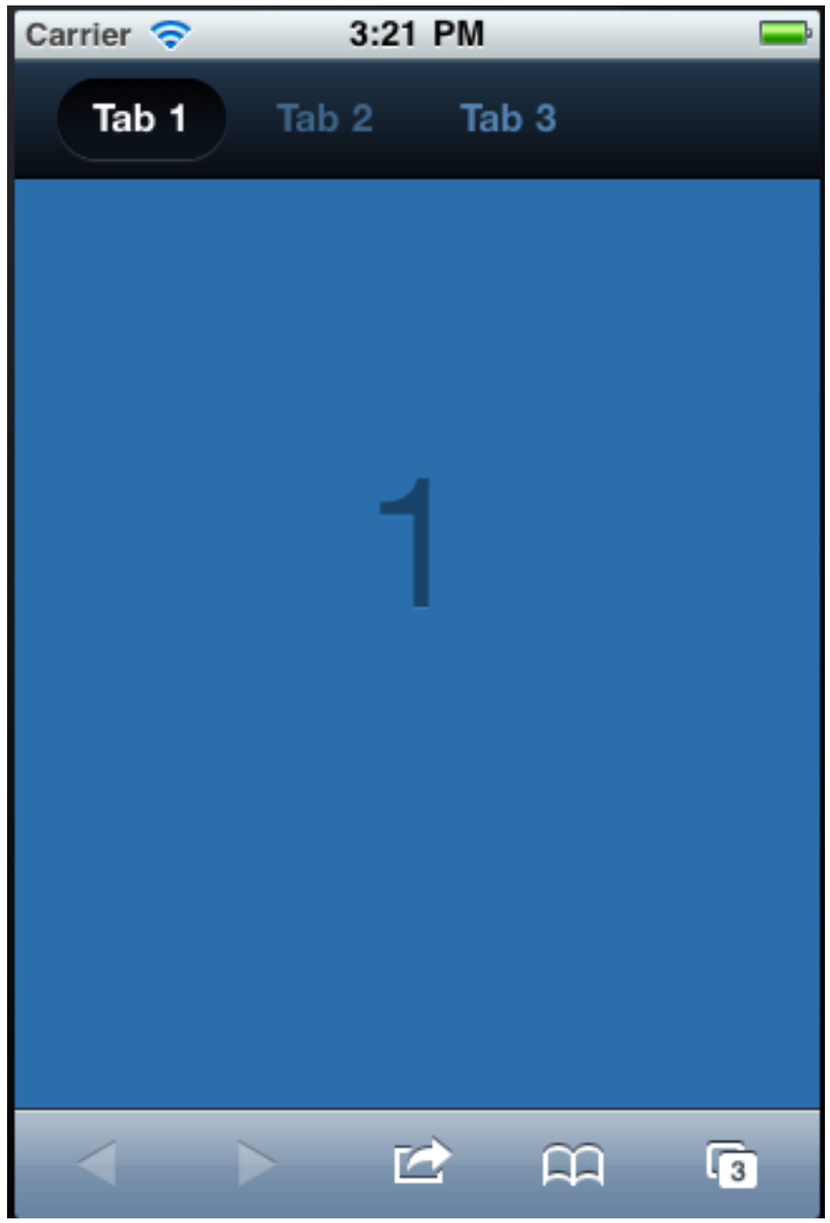


Styles used in Carousel

```
<style>
 body { background-color: #376daa; }
 .card {
     text-align: center;
     color: #204167;
     text-shadow: #3F80CA 0 1px 0;
     font-size: 72px;
     padding: 80px 40px;
 }

     .card p {
         font-size: 24px;
         line-height: 30px;
     }

     .x-phone .card p {
         font-size: 16px;
         line-height: 18px;
     }
</style>
```

Tabs

```
Ext.setup({
    onReady: function() {
        new Ext.TabPanel({
            fullscreen: true,
            type: 'dark',
            sortable: true,
            items: [{
                title: 'Tab 1', html: '1', cls: 'card1'
            }, {
                title: 'Tab 2', html: '2', cls: 'card2'
            }, {
                title: 'Tab 3', html: '3', cls: 'card3'
            }]
        });
    }
});
```



Tab Example CSS

```
.card1, .card2, .card3 {
            background-color: #376daa;
            text-align: center;
            color: #204167;
            text-shadow: #3F80CA 0 1px 0;
            font-size: 72px;
          padding-top: 100px;
        }
```