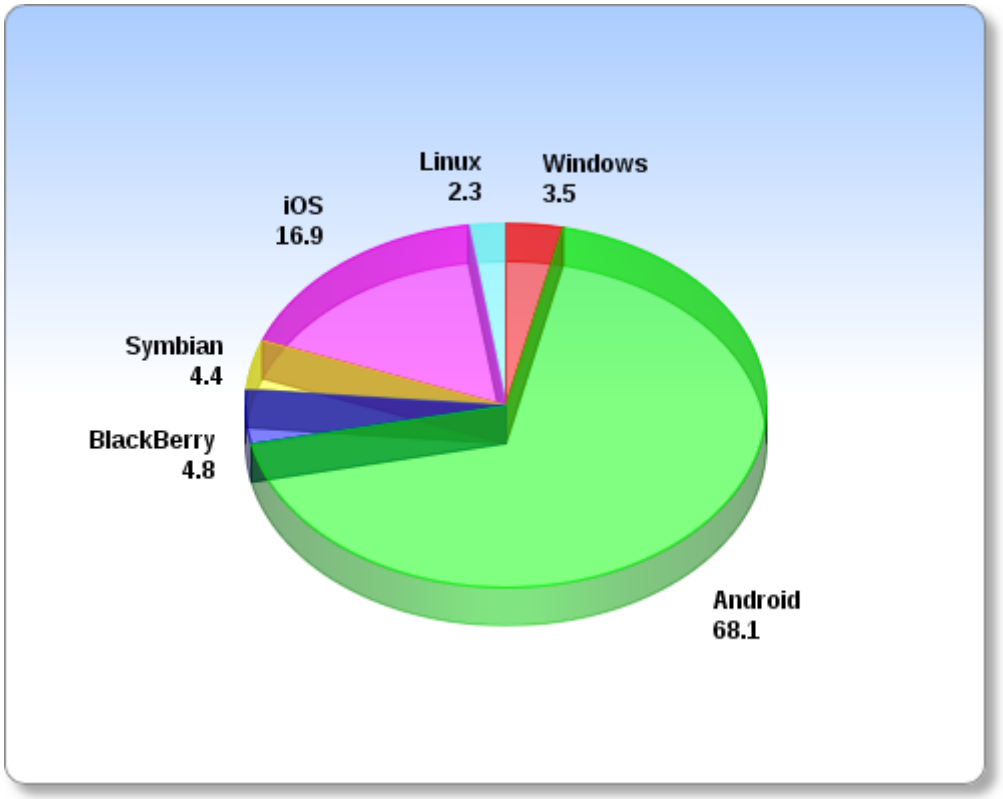
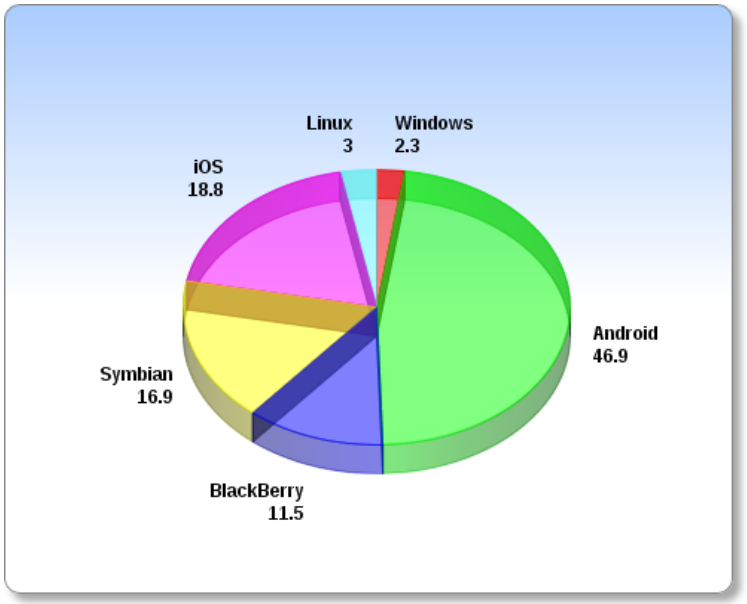
Cross-platform Mobile Development

Need an App ...

Smartphone OS Market Share





IDC, 2012 v. 2011

Cross-platform Mobile Development

Application types

- Native
- Web
- Hybrid
- Cross-compile

Native Apps

Built for a specific platform with the platform SDK, tools and languages, typically provided by the platform vendor

Native Apps

+

- Speed and reaction
- Native look'n'feel
- Native API
- Debug + Profile

-

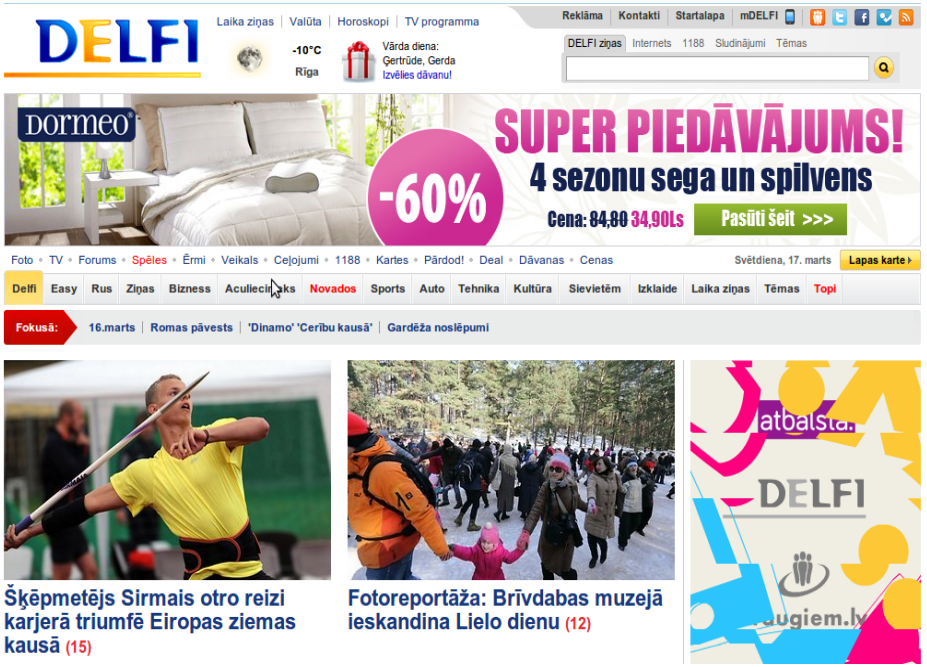
- Only 1 platform
- New language?
- Costs
- App review time

When to use native app ?

- Use native app if ...
  - You want to achieve native look and feel
  - You need top performance
  - You want to be on the app store(s)

The Mobile “Server-side” Web

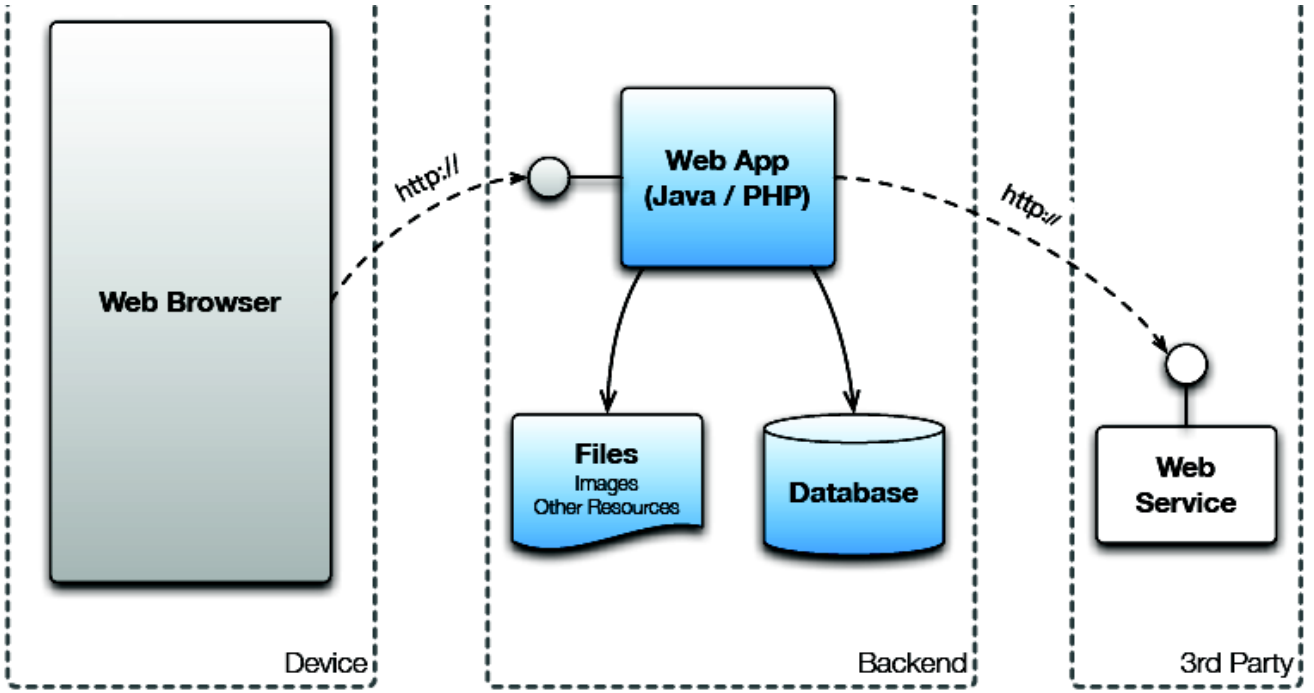
Uses HTML, JavaScript, CSS and runs in a devices browser



The Mobile “Server-side” Web

- **Responsive Design**
  - Dynamically loads HTML content and assets
  - Serves same HTML content and assets for every device and then scales via CSS using media queries
    - e.g. @media screen and (max-width: 650px)

The Mobile “Server-side” Web



The Mobile “Server-side” Web

+

- Existing tools
- No license fees
- All platforms with browser, even your old Nokia
- Search engine

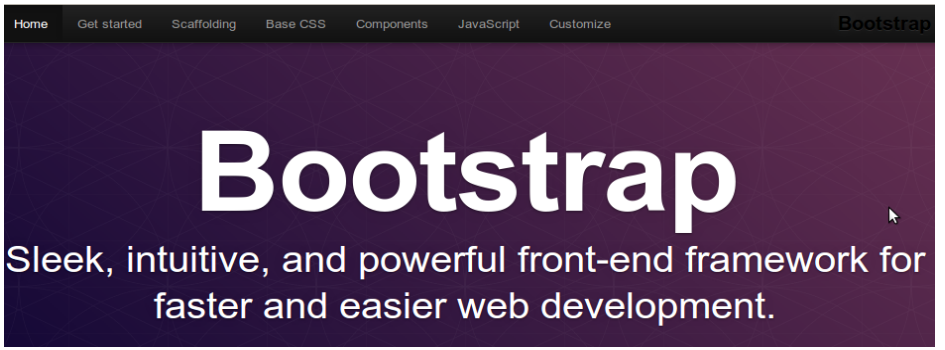
-

- No app stores
- Limited platform access (camera, calendar, etc)
- As fast as your browser, network latency

The Mobile “Server-side” Web

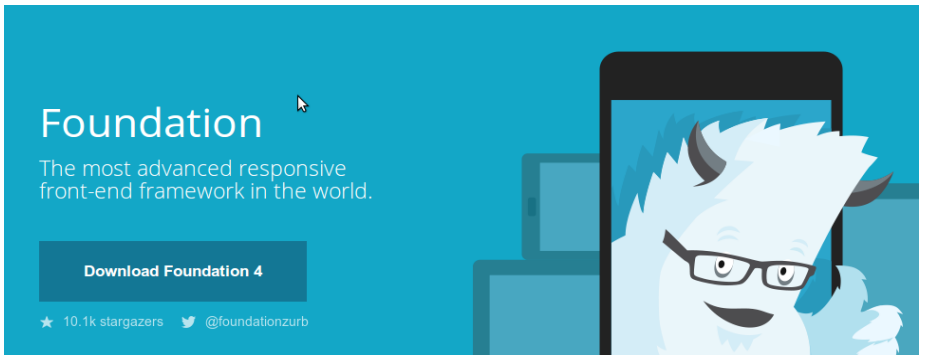
- **Twitter Bootstrap**

  http://twitter.github.com/bootstrap/



- **zurb Foundation**

  http://foundation.zurb.com/



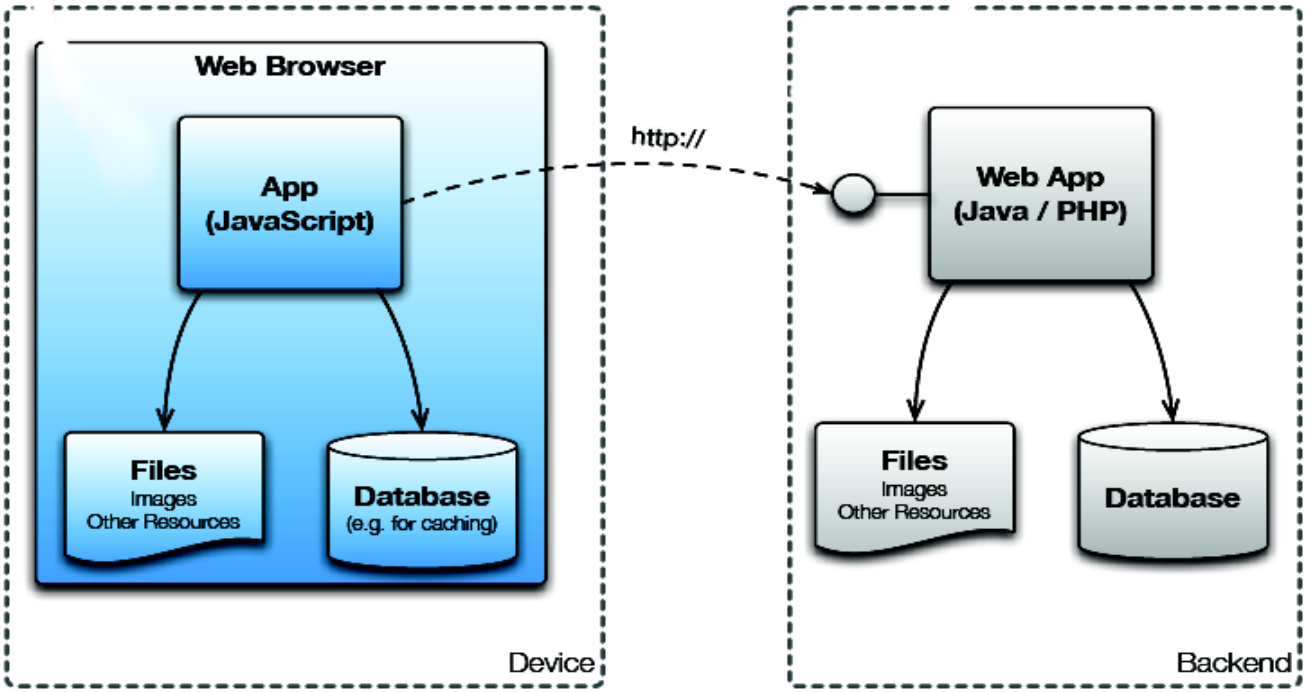
- **Other**

The Mobile “Server-side” Web

- Use the mobile web if ...
  - You need a mobile landing page for your company
  - You need something that runs on almost every platform

The Mobile “Client-side” Web

Uses HTML5 + JavaScript and runs in a devices browser



The Mobile “Client-side” Web

- **Sencha Touch**
- JQueryMobile
- JQTouch
- KendoUI
- GWT Mobile

The Mobile “Client-side” Web

+

- Existing tools
- No license fees
- All platforms with browser
- Search engine

-

- No app stores
- Limited platform access (camera, calendar, etc)
- As fast as your browser, network latency

The Mobile “Client-side” Web

- Use the client-side web if ...
  - Need something that looks like an app, but care less for the sensors of your phone
  - You want to prototype a service
  - You need crossplatform app (not native)

The Mobile “Client-side” Web

- Desktop browser != Mobile browser
- Safari != Chrome
- Offline app storage limits
- Screen rotation animation

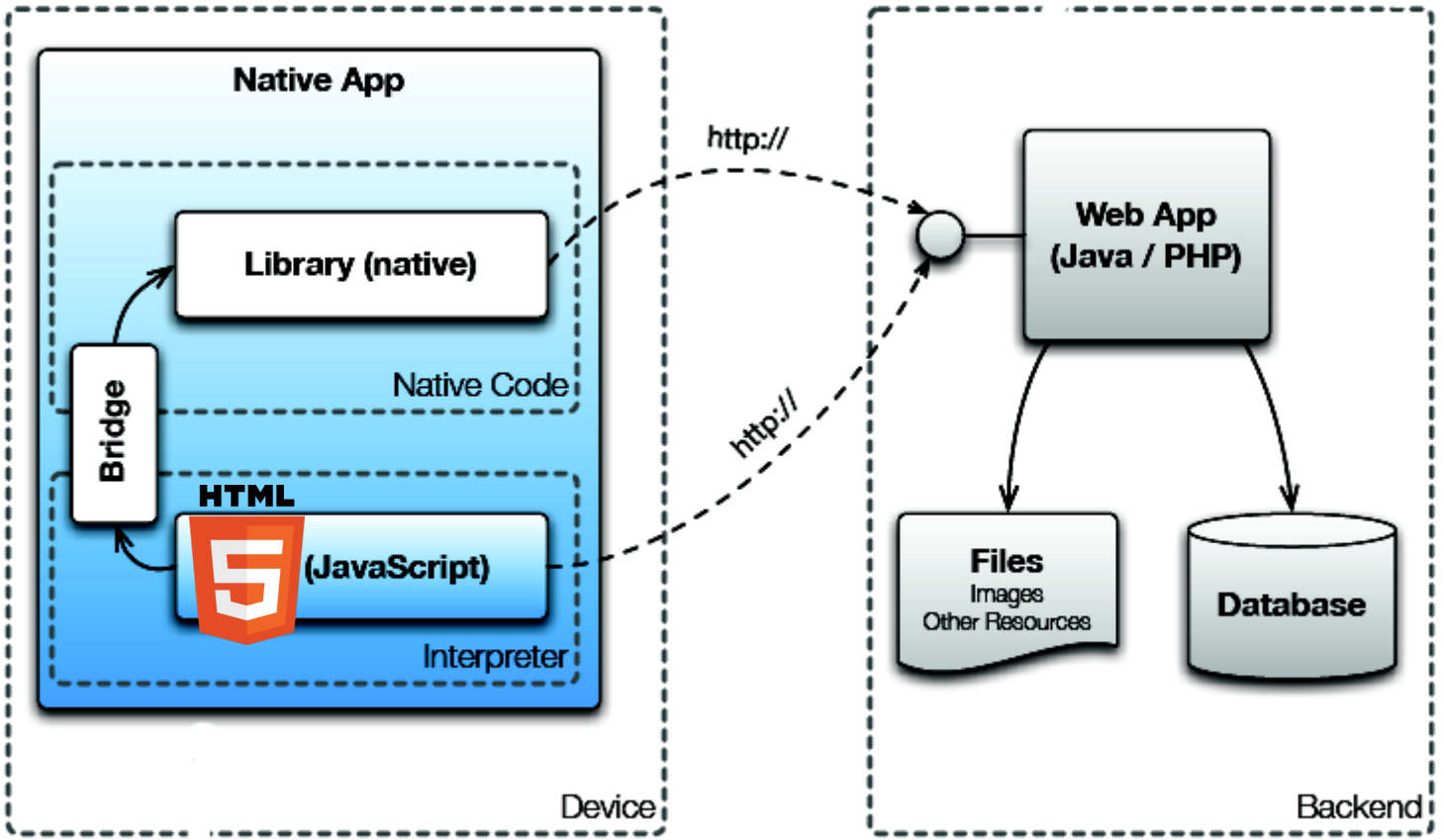
The Mobile “Client-side” Web

- Minimize frontend
- Optimize images
- Do not use jQuery only for $(“#mybutton”)
- Use HTML5 features (WebStorage, WebWorker, CSS3 animations)

Test on as much as possible devices !

Hybrid App

HTML5 + JavaScript with a wrapper that gives it native capabilities



Hybrid App

- Run inside a native container to leverage the device's browser engine (but not the browser)
- Web-to-native abstraction layer enables access to device capabilities

Hybrid App

- **WebView / UIWebView + self made protocol** (myapp://something)
- **Apache Cordova (PhoneGap)**
  - JavaScript abstraction layer over native APIs
  - An open source framework that allows you to access native APIs for iPhone, Android, BlackBerry, Windows Phone 7 + more
    - Accelerometer
    - Camera
    - Compass
    - Contacts
    - Geolocation
    - Notifications

Hybrid App

+

- Existing tools
- No license fees
- All platforms with browser
- Search engine

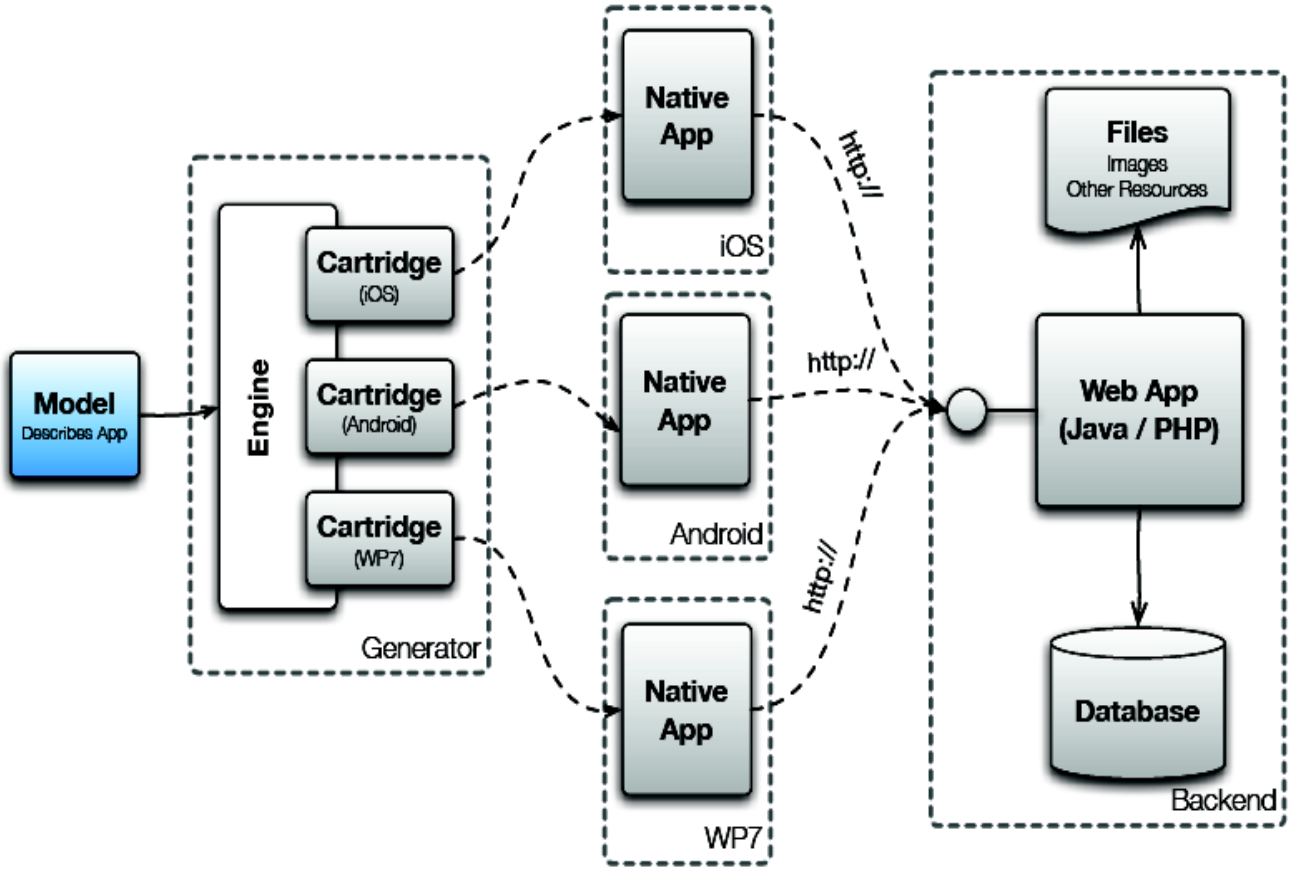
-

- As fast as your browser, network latency

Hybrid App

- Use hybrid approach if ...
  - You want to build your app based on HTML5
  - And need access to the phone's hardware

Interpreted and cross-compile



Interpreted and cross-compile

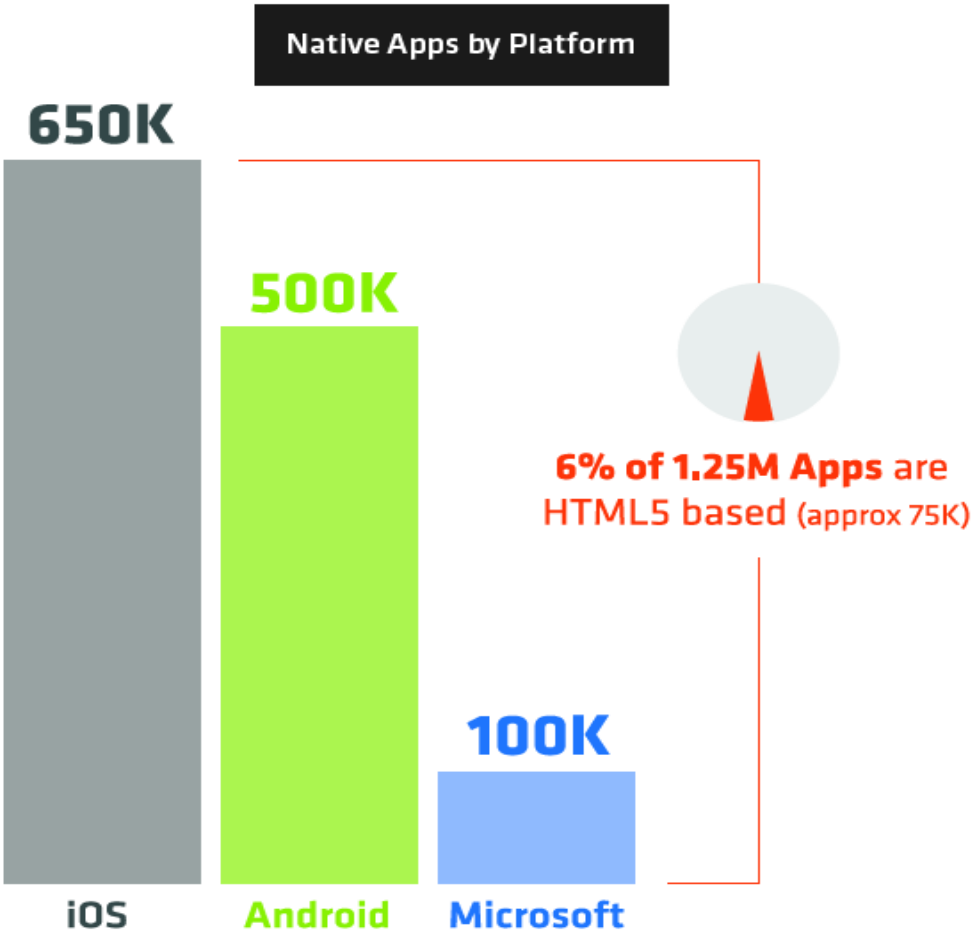
- Appcelerator Titanium
  - JavaScript -> Native + Hybrid JS
- MonoTouch
  - C# -> Native
- Apache Thrift
  - Thrift definition -> Native
- QT

Interpreted and cross-compile

- Use interpreted or cross-compiled app if ...
  - You want to use “one language to build them all”
  - Need access to the phone's hardware
  - Need semi- or truly native look and feel / performance

Decision

- **Factors**
  - App size and type
  - Platforms
  - Developers skills
  - Budget
  - Monetization
  - User experience
  - Importance of updates



Decision

Native vs. HTML5 Comparisons

	Native	HTML5
Rich user experience	●	
Performance	●	
App store monetization	●	
Cross platform deployment costs	HIGH	LOW
Fragmentation challenges	●	●
Availability of programming expertise		●
Immediate updates and distribution control		●
Timely access to new OS innovations	●	

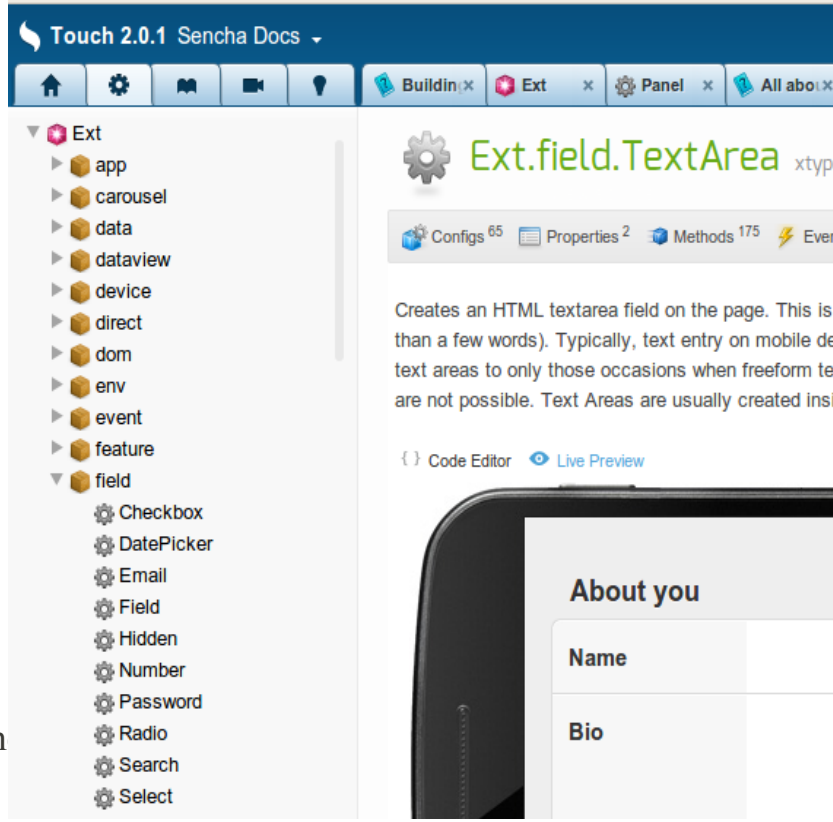
Sencha Touch

Sencha Touch

- JavaScript framework for building rich mobile apps with web standards
- Produce a native-app-like experience inside a browser
- Usually enables to quickly create HTML5 based mobile apps that work on Android, iOS and Blackberry devices

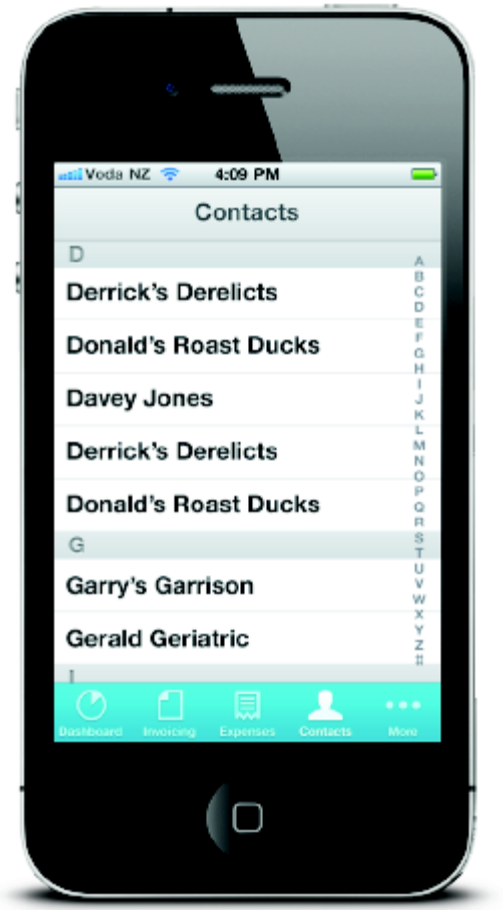
Features

- Cross-platform
- Looks native, feels native
- Usually faster, cheaper, easier to build with
- Highly customisable
- Great API documentation
- HTML5/CSS3

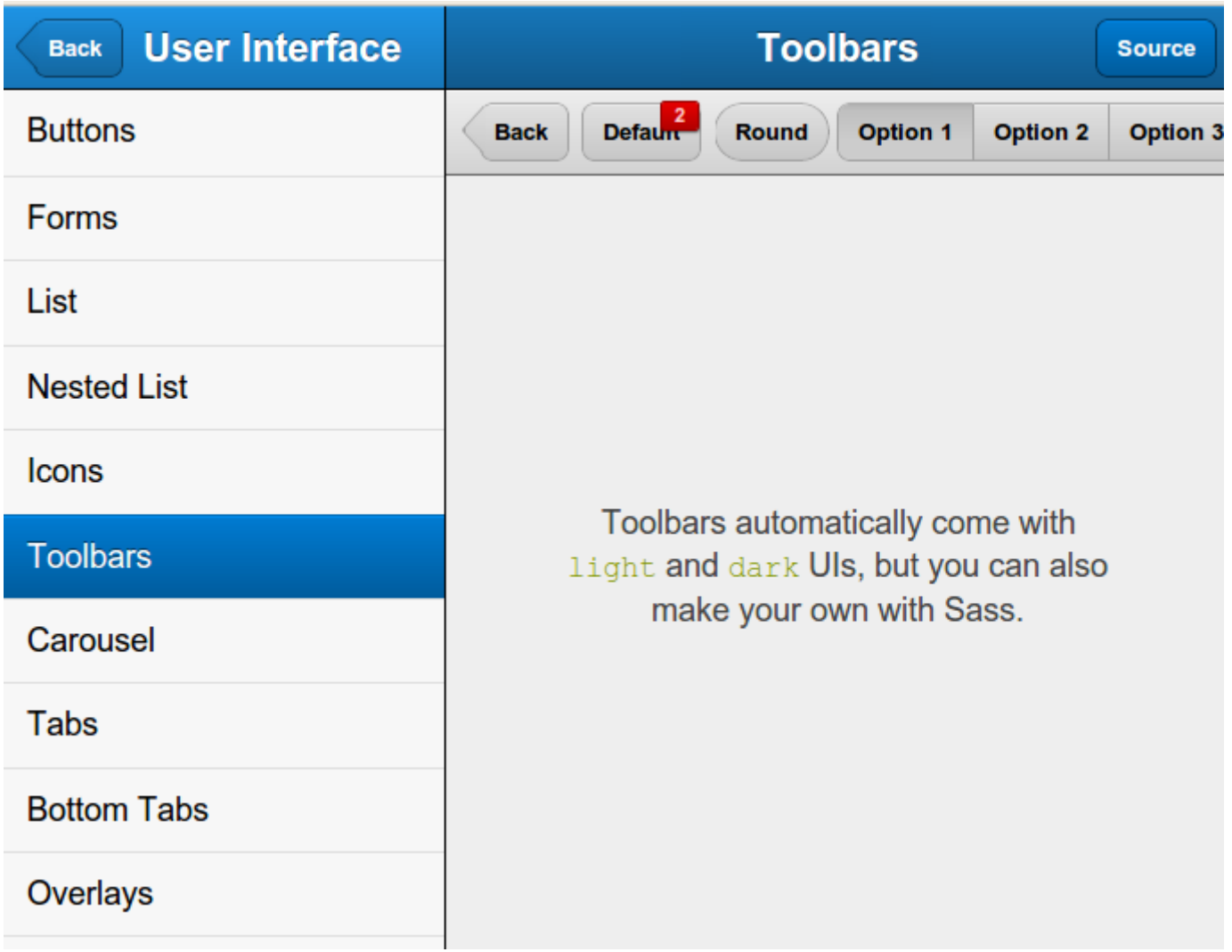


Features

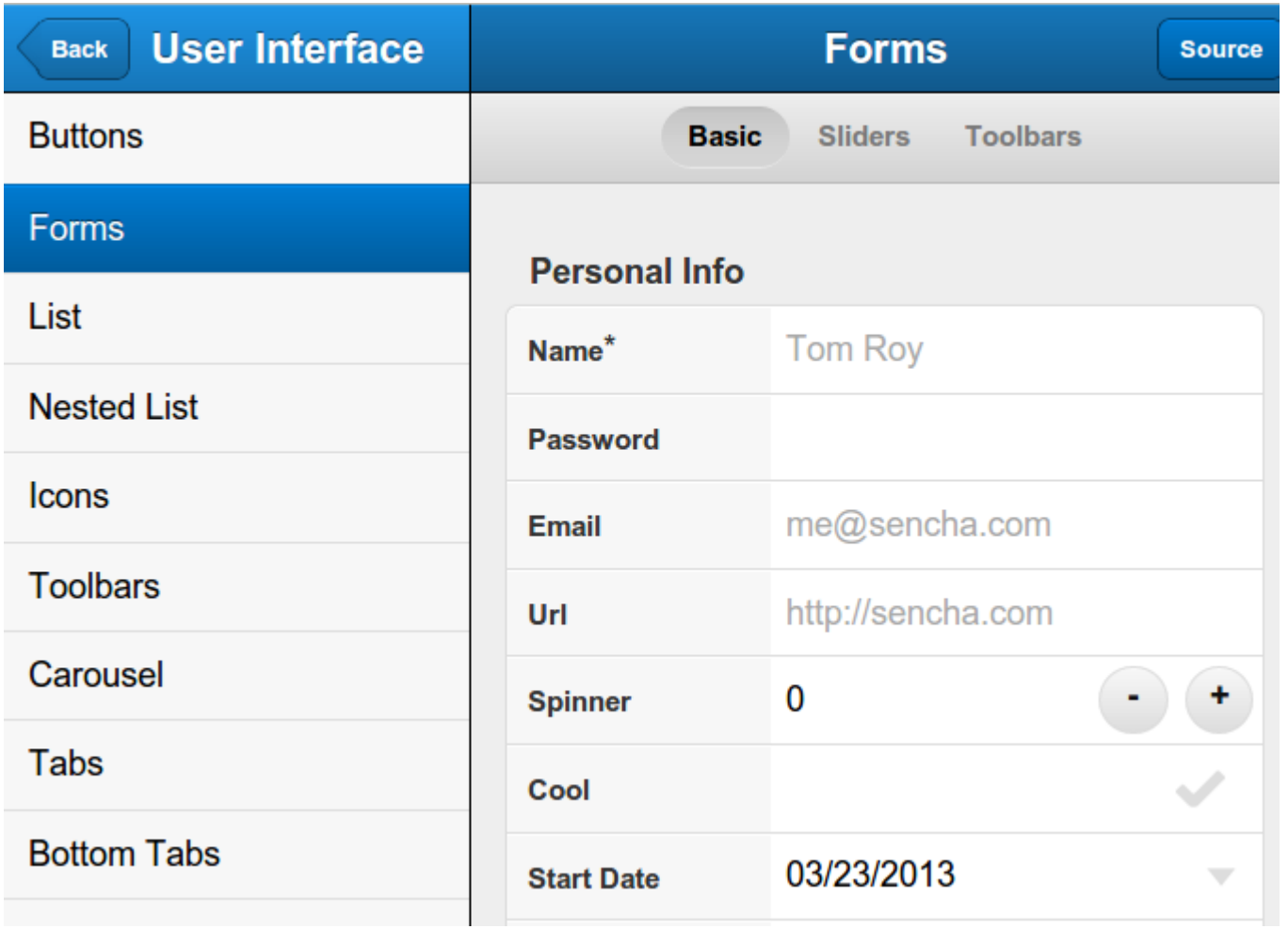
- Multi-touch and gesture support
- Flexible deployment
- Hardware accelerated, native & natural scrolling



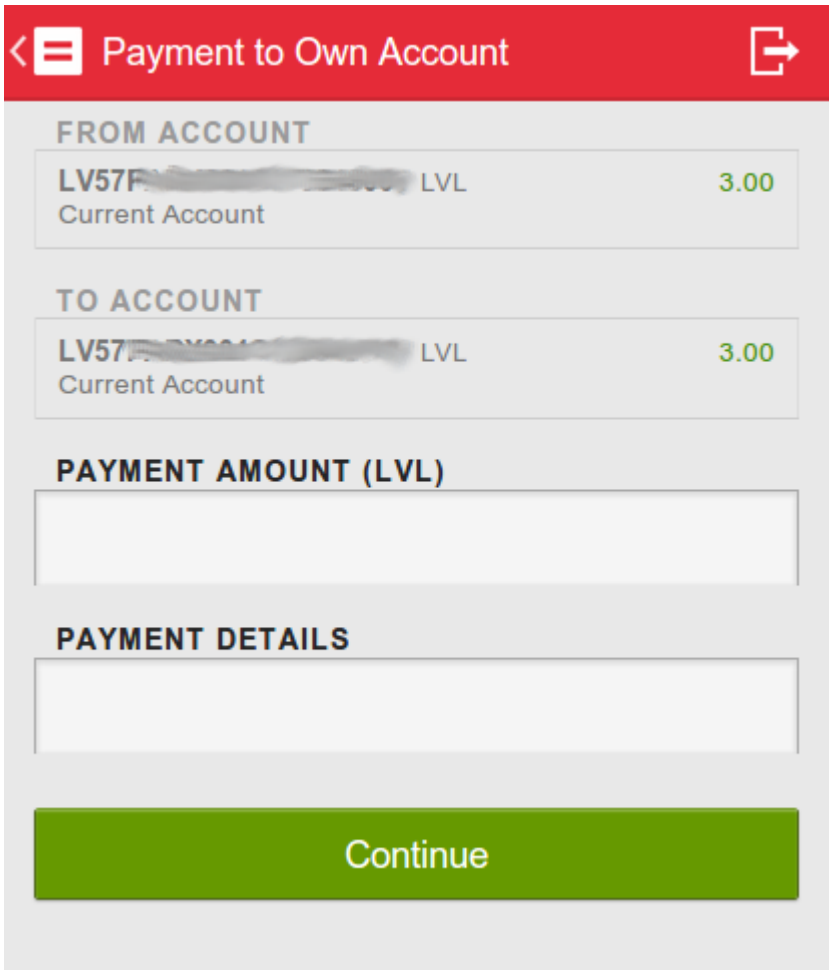
Sench Touch UI

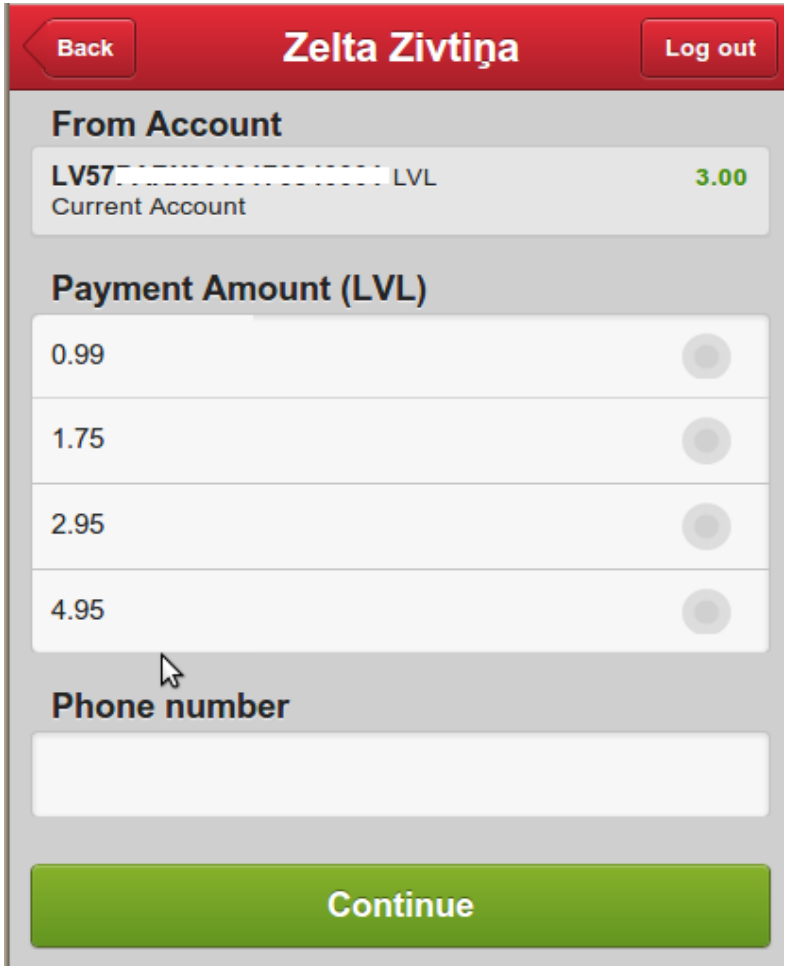


Sencha Touch UI



Sencha Touch UI



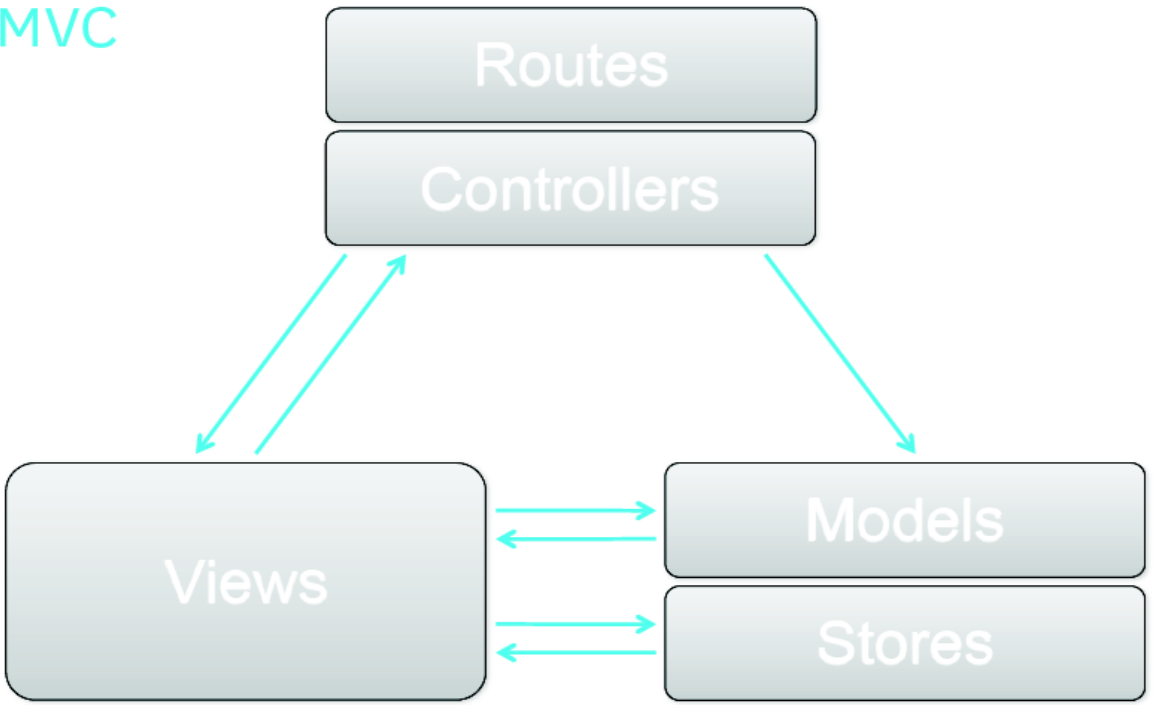


Javascript Spaghetti

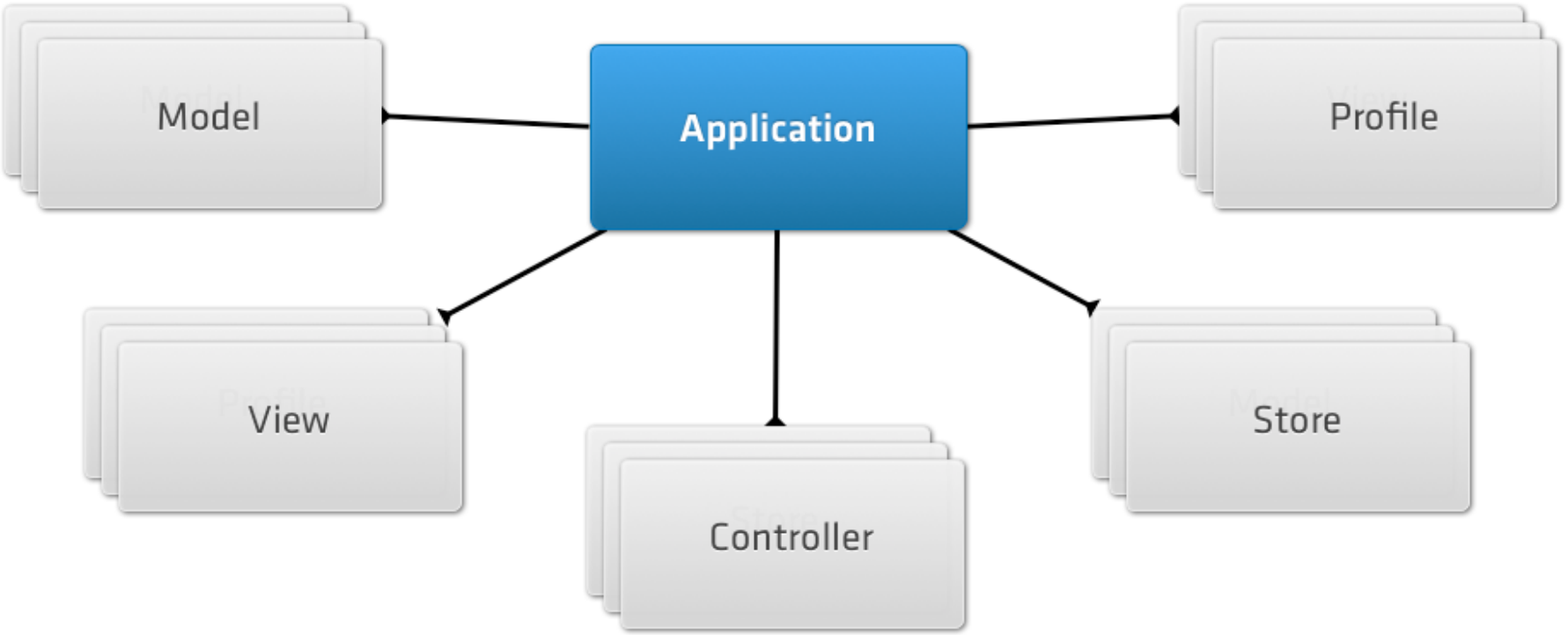
Sencha Touch Way

- Building applications, not websites
- Directory structure for code organization
- Pure JavaScript

- MVC Paradigm
- Based on ExtJS 4



Sencha Building Blocks



Sencha Touch SDK

Sencha Touch SDK Tools

Sencha Building Blocks

- Models: represent a type of object in your app (Users, Products)
- Views: are responsible for displaying data to your users and leverage the built in Components in Sencha Touch
- Controllers: handle interaction with your application, listening for user taps and swipes and taking action accordingly
- Stores: are responsible for loading data into your app and power Components like Lists and DataViews

Start using Sencha Touch

- Get a WebKit-based desktop browser
- Get some emulators & real devices
- Download the Sencha Touch 2
- Develop against a local web server
- http://www.sencha.com/products/touch

Resources for Practical Task

- **Sencha-generated.zip –** application stub generated by sencha sdk tools
  - Starting point for development
  - Includes sdk
- **Tomcat (or other web-server) + Sencha.xml**
  - http://tomcat.apache.org/download-70.cgi
  - Put Sencha.xml into tomcat/conf/Catalina/localhost
  - NB! change path

```
<Context path="/Sencha"
docBase="YOURFOLDER/Sencha/" />
```

Resources for Practical Task

- **Sencha API documentation**

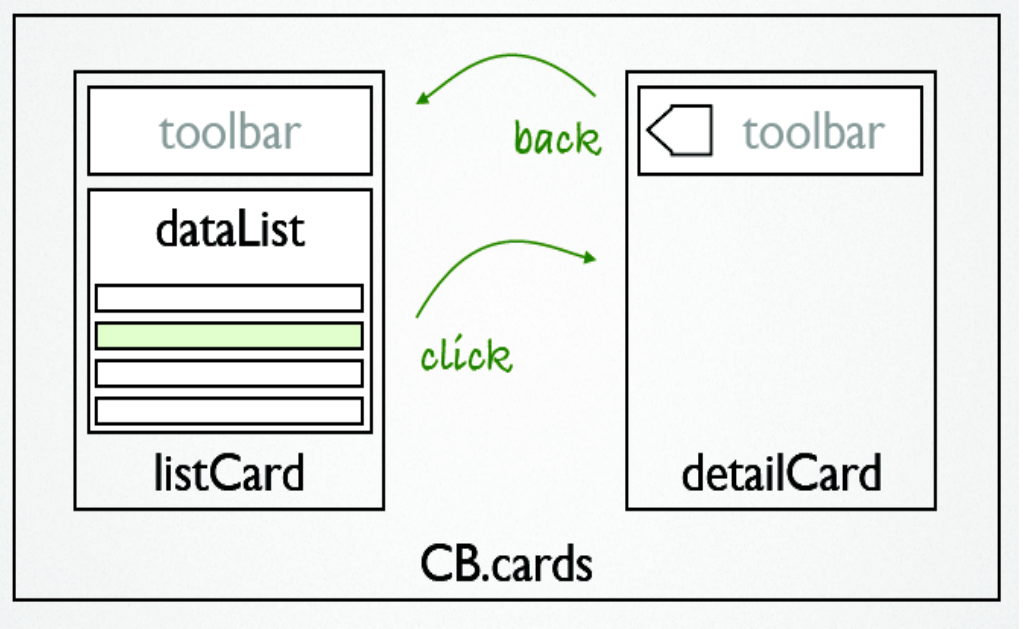
  http://docs.sencha.com/touch/2-0/#!/api

- **Navigation.js, purchases.js**

- **Sencha-minified.zip –** application minified bundle for Android hybrid app

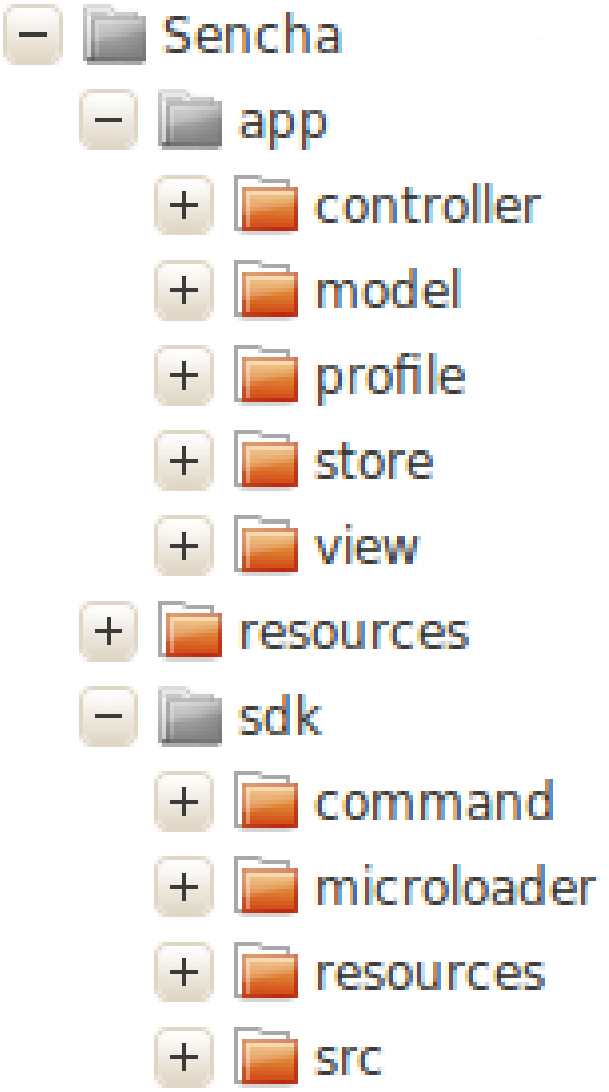
Development workflow

- App architecture
- UI structure
- Data Modelling
- List binding
- List event handler
- Details page
- Package



Sench Touch App Tree

- App
- Resources
- SDK (do not touch)
- app.js
- .json files
- index.html



Sencha Touch / Application

- The Application is usually the first thing you define in a Sencha Touch application

```
Ext.application({
    name: 'Sencha',

    requires: [
        'Ext.MessageBox'
    ],

    views: ['Main','Navigation','PurchaseList','PurchaseView','PurchaseEdit'],
    models: ['Purchase'],
    stores: ['Purchases'],
    controllers: ['PurchaseList','Purchases'],
```

Sencha Touch / Application

- Each Application can define a launch function, which is called as soon as all of your app's classes have been loaded and the app is ready to be launched

```
launch: function() {
    // Destroy the #appLoadingIndicator element
    Ext.fly('appLoadingIndicator').destroy();

    // Initialize the main view
    //Ext.Viewport.add(Ext.create('Sencha.view.Main'));

    Ext.Viewport.add({
        xtype : 'navigation',
        items : { xtype : 'purchaselistview' }
    });
},
```

Sencha Touch / Classes

- Uses the state of the art class system developed for Ext JS 4
  - easy to create new classes in JavaScript
  - providing inheritance, dependency loading, mixins, powerful configuration options, and lots more
- Class has some functions and properties attached to it

```
Ext.define('Animal', {
    config: {
        name: null
    },

    constructor: function(config) {
        this.initConfig(config);
    },

    speak: function() {
        alert('grunt');
    }
});
```

```
Ext.define('Human', {
    extend: 'Animal',

    speak: function() {
        alert(this.getName());
    }
});
```

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

bob.speak(); //alerts 'Bob'
```

Sencha Touch / Component

- Most of the visual classes in Sencha Touch are Components. Every Component in Sencha Touch is a subclass of Ext.Component
  - Render themselves onto the page using a template
  - Show and hide themselves at any time
  - Center themselves on the screen
  - Enable and disable themselves

```
//we can configure the HTML when we instantiate the Component
var panel = Ext.create('Ext.Panel', {
    fullscreen: true,
    html: 'This is a Panel'
});

//we can update the HTML later using the setHtml method:
panel.setHtml('Some new HTML');

//we can retrieve the current HTML using the getHtml method:
Ext.Msg.alert(panel.getHtml()); //alerts "Some new HTML"
```

Sencha Touch / xtype

- xtype is an easy way to create Components without using the full class name.
  - a shorthand way of specifying a Component
  - e.g., you can use xtype: 'panel' instead of typing out Ext.panel.Panel.

```
items: [
    {
        xtype: 'panel',
        html: 'This panel is created by xtype'
    },
    {
        xtype: 'toolbar',
        title: 'So is the toolbar',
        docked: 'top'
    }
]
```

Sencha Purchase List / Model

- Purchase.js

```
Ext.define('Sencha.model.Purchase', {
    extend : 'Ext.data.Model',

    config : {
        fields : [
            'product',
            'quantity',
            'list',
            'notes'
        ]
    }
});
```

Sencha Purchase List / Store

- Purchases.js

```
Ext.define("Sencha.store.Purchases", {
    extend : 'Ext.data.Store',

    config : {
        storeId : 'purchases',
        model : 'Sencha.model.Purchase',

        sorters : 'product',
        groupField : 'list',
        autoLoad : 'true',
```

Sencha Purchase List / Store

```
        data: [
         { "product": "Tomato", "quantity": "1kg
         { "product": "Bread", "quantity": "1",
         { "product": "Cabbage", "quantity": "3k
         { "product": "Jameson", "quantity": "1"
         { "product": "Tomato", "quantity": "1kg
         { "product": "Bread", "quantity": "1",
         { "product": "Cabbage", "quantity": "3k
         { "product": "Jameson", "quantity": "1"
        ],

//      proxy : {
//          type : 'ajax',
//
//          url : 'purchases.json',
//          reader : {
//              type : 'json',
//              rootProperty : 'purchases'
//          }
//      }
    }
});
```

Sencha Purchase List / View

- PurchaseList.js

```
Ext.define("Sencha.view.PurchaseList", {
    extend : 'Ext.dataview.List',
    xtype : 'purchaselistview',

    config : {
        title : 'Purchases',
        store : 'purchases',
        itemTpl : '<b>{product}</b> ({quantity})',
        grouped : true
    }
});
```

Sencha Purchase List / View

```
Ext.define("Sencha.view.PurchaseView", {
    extend : 'Ext.form.Panel',
    requires : ['Ext.form.FieldSet'],
    xtype : 'purchaseview',

    config : {
        purchaseitem : null,
        title : 'Details',
        id : 'purchase_details_form',
        items : [{
                    id : 'purchase_details',
                    xtype : 'fieldset',
                    title : 'Purchase Item info',
                    items : [{
                                xtype : 'textfield',
                                name : 'product',
                                label : 'Product'
                            }, {
                                xtype : 'textfield',
                                name : 'quantity',
                                label : 'Quantity'
                            }, {
```

Sencha Purchase List / View

```
updatePurchaseitem : function(newVal, oldVal) {
    var pdetails = Ext.getCmp('purchase_details');

    pdetails.down('[name = product]').setValue(newVal.get('product'));
    pdetails.down('[name = quantity]').setValue(newVal.get('quantity'));
    pdetails.down('[name = notes]').setValue(newVal.get('notes'));
}
```

Sencha Purchase List / Controller

- Controllers are responsible for responding to events that occur within your app.
- If your app contains a Logout button that your user can tap on, a Controller would listen to the Button's tap event and take the appropriate action.
- An Application usually consists of a number of Controllers, each of which handle a specific part of the app
- Refs and Control configuration
  - Refs used to gain references to Components inside your app and to take action on them
  - Control is the means by which your listen to events fired by Components.

Sencha Purchase List / Controller

```
Ext.define('Sencha.controller.PurchaseList', {
    extend: 'Ext.app.Controller',

    config: {
        refs: {
            navigation: 'navigation',
            list: 'purchaselistview'
        },

        control: {
            list: {
                itemtap: function(view, index, target, record, e, eOpts) {
                    view.select(record);

                    this.getNavigation().push({
                        xtype: 'purchaseview',
                        purchaseitem: record
                    });

                    return true;
                }
            }
```

Sencha Purchase List / Navigation

- NavigationView is basically - Ext.Container with a card layout, so only one view can be visible at a time. Extra functionality on top of this to allow you to push and pop views at any time

```
Ext.define("Sencha.view.Navigation", {
    extend : 'Ext.navigation.View',
    xtype : 'navigation',
    requires : ['Ext.TitleBar'],

    config : {
        iconCls : 'home',
        defaultBackButtonText : 'Back',

        navigationBar : {
            items : [{
                        id : 'btn-navigation-add',
                        xtype : 'button',
                        align : 'right',
                        text : 'Add'
                    }, {
                        xtype : 'button',
                        align : 'right',
                        text : 'Exit',
```

Hybrid Sencha Purchase List

- Embed Sencha App into Android V
- Use packaged / minimized app bu

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <LinearLayout
        android:id="@+id/webViewSencha"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true" />

</RelativeLayout>
```

Hybrid Sencha Purchase List

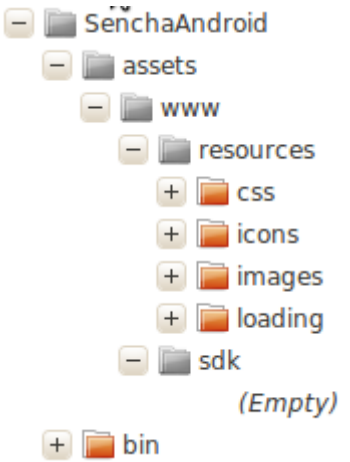
```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_sencha);

    webViewContainer = (LinearLayout)findViewById(R.id.webViewSencha);
    webView = new WebView(getApplicationContext());
    webView.setLayoutParams(new LinearLayout.LayoutParams(LayoutParams.MA
    webViewContainer.addView(webView);

    final WebSettings webSettings = webView.getSettings();
    webSettings.setJavaScriptEnabled(true);  S
    webSettings.setDomStorageEnabled(true);
    webSettings.setRenderPriority(WebSettings.RenderPriority.HIGH);

    webView.setVerticalScrollBarEnabled(false);
    webView.setHorizontalScrollBarEnabled(false);
    webView.setInitialScale(0);

    webView.addJavascriptInterface(new SenchaAndroidNativeInterface(), JS_SENCHA_ANDROID);
    webView.loadUrl(SITE);
}
```



Hybrid Sencha Purchase List

- **Calling native from Sencha (Android)**
  - webView.addJavascriptInterface(new SenchaAndroidNativeInterface(),”senchaAndroidNative”);

```
class SenchaAndroidNativeInterface {
    public void exitToMain() {
        SenchaActivity.this.finish();
    }
}
```

Hybrid Sencha Purchase List

- **Calling native from Sencha (Secha)**

```
}, {
    xtype : 'button',
    align : 'right',
    text : 'Exit',

    handler : function() {
        if (window.senchaAndroidNative != undefined) {
            senchaAndroidNative.exitToMain();
        }
    }
}]
```

Hybrid Sencha Purchase List

- NB! ProGuard
- proguard-properties.txt

  -keep public class com.example.senchaandroid.SenchaActivity$SenchaAndroidNativeInterface

  -keep public class * implements com.example.senchaandroid.SenchaActivity$SenchaAndroidNativeInterface

  -keepclassmembers class com.example.senchaandroid.SenchaActivity$SenchaAndroidNativeInterface { <methods>;}