BUILDING NATIVE MOBILE APPS WITH JAVASCRIPT

JOSH JENSEN ABOUT ME

MOBILE APPLICATION CONSULTANT,
ENTREPRENEUR, TECHNOLOGY ADDICT,
OCR ENTHUSIAST, BEER DRINKER.

@JOSHJ

WHAT THIS SESSION IS NOT

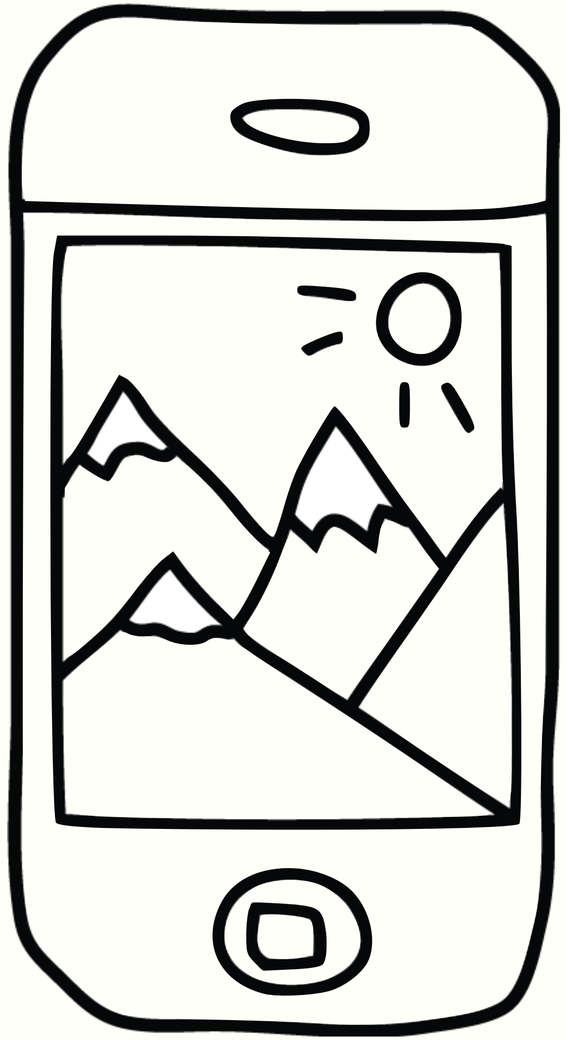
- A STUDY IN BEST PRACTICES
- A COMPREHENSIVE LOOK AT EACH PLATFORM
- A TIME TO BASH ONE PLATFORM OR ANOTHER

WHAT THIS SESSION IS

- A DEVELOPERS JOURNEY THROUGH 3 PLATFORMS
- INITIAL IMPRESSIONS AND RESULTS
- A TOOL FOR DECIDING WHAT IS RIGHT FOR YOU

MOBILE LANDSCAPE

NATIVE	OBJECTIVE C, SWIFT, JAVA
HYBRID	PHONEGAP/CORDOVA
TRANSPILED	XAMARIN
ABSTRACTED	NATIVESCRIPT, REACT NATIVE, TITANIUM



DEFINE AB·STRACT·ED

CONSIDER (SOMETHING)
THEORETICALLY OR SEPARATELY
FROM SOMETHING ELSE

DEFINE: ABSTRACTED PLATFORM

A platform where an API is abstracted away from or separated from the primary language by proxy and/or runtime.

OBJ C

```
UIView * myView = [[UIView alloc] init];
```

TITANIUM

```
var myView = Ti.UI.createView();
```

NATIVESCRIPT

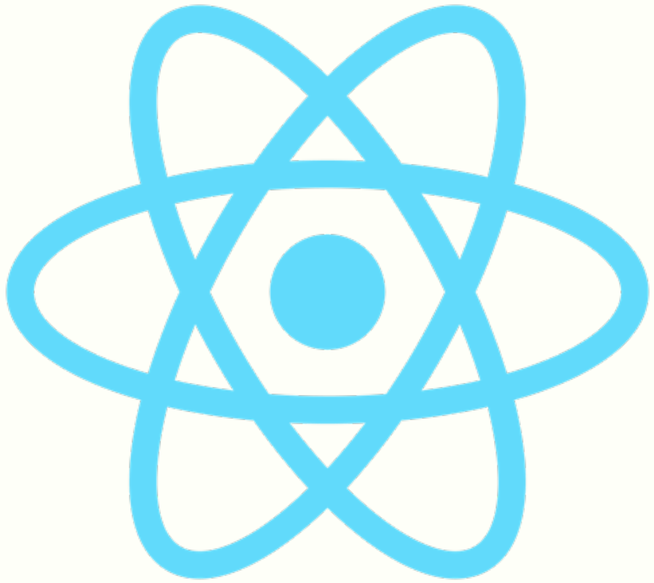
```
var myView = UIView.alloc().init();
```

REACT NATIVE

```
<View></View>
```

ABSTRACTED PLATFORMS

NATIVESCRIPT



REACT NATIVE

TITANIUM

PLATFORM PRICING

NativeScript	React Native	Titanium
OPEN SOURCE: FREE	OPEN SOURCE: FREE	OPEN SOURCE: FREE*
SEAT: $39/MO	FREE	INDIE: $39/MO
ADD-ONS: VARY	FREE	TEAM: $259/SEAT/MO
ENTERPRISE: CALL	FREE	ENTERPRISE: CALL

*Appcelerator's distributed open source offering is one to two version behind their commercial offering.

PLATFORM QUICK FACTS

	NativeScript	React Native	Titanium
OPEN SOURCE	✓	✓	✓
LICENCE	APACHE	BSD	APACHE / PROPRIETARY
IOS	✓	✓	✓
ANDROID	✓	✓	✓
WINDOWS	COMING SOON	NO	LIMITED

Source: https://github.com/skypanther/mobileframeworks

PLATFORM TOOLING

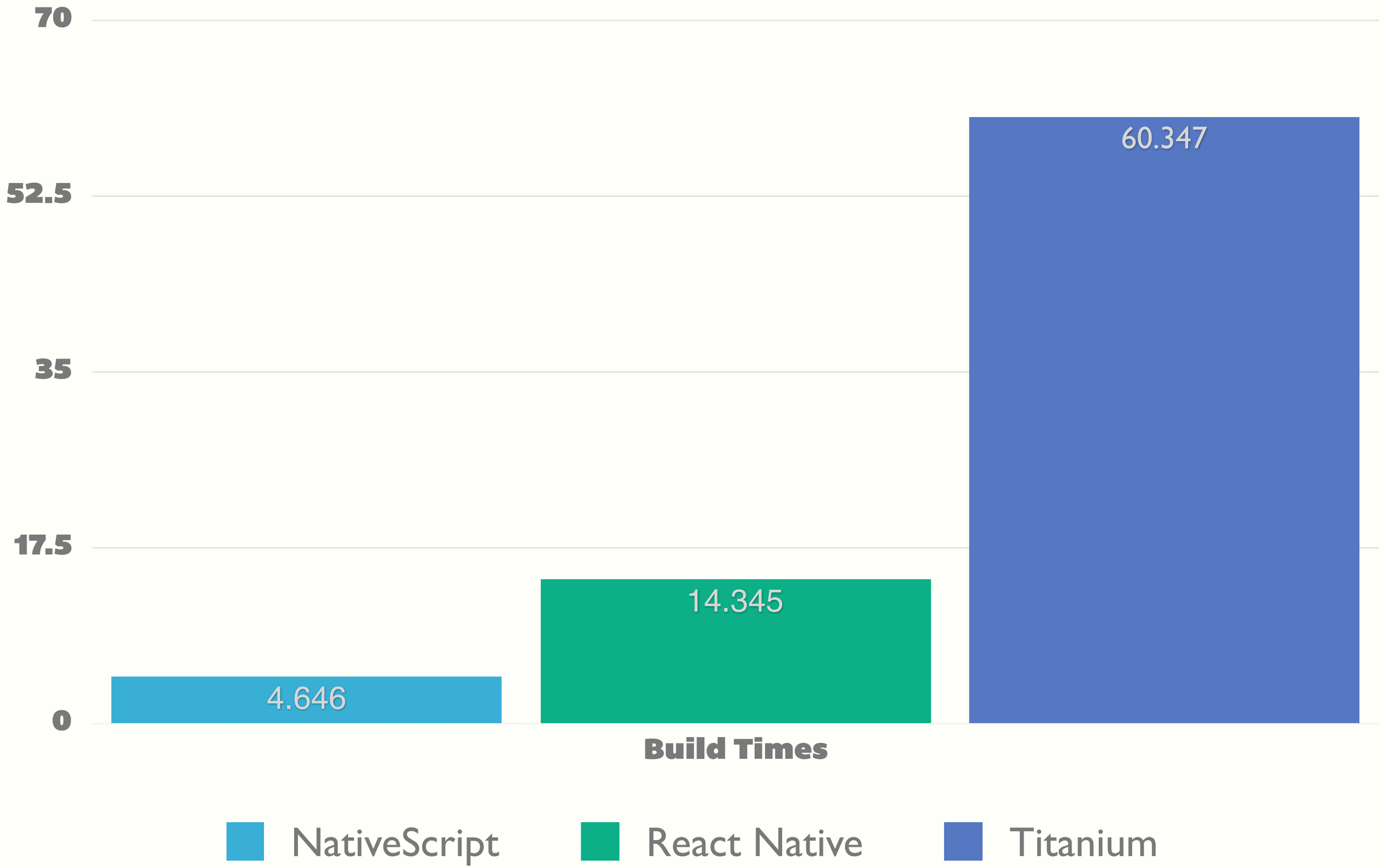
	NativeScript	React Native	Titanium
NODE.JS	0.12.X	4.X	0.12.X
XCODE	LATEST	6.3+	6.4.X
GRADLE	REQUIRED	N/A	N/A
ANT	N/A	N/A	REQUIRED
JDK	LATEST	LATEST	LATEST
HOMEBREW	RECOMMENDED	RECOMMENDED	N/A

PLATFORM SETUP

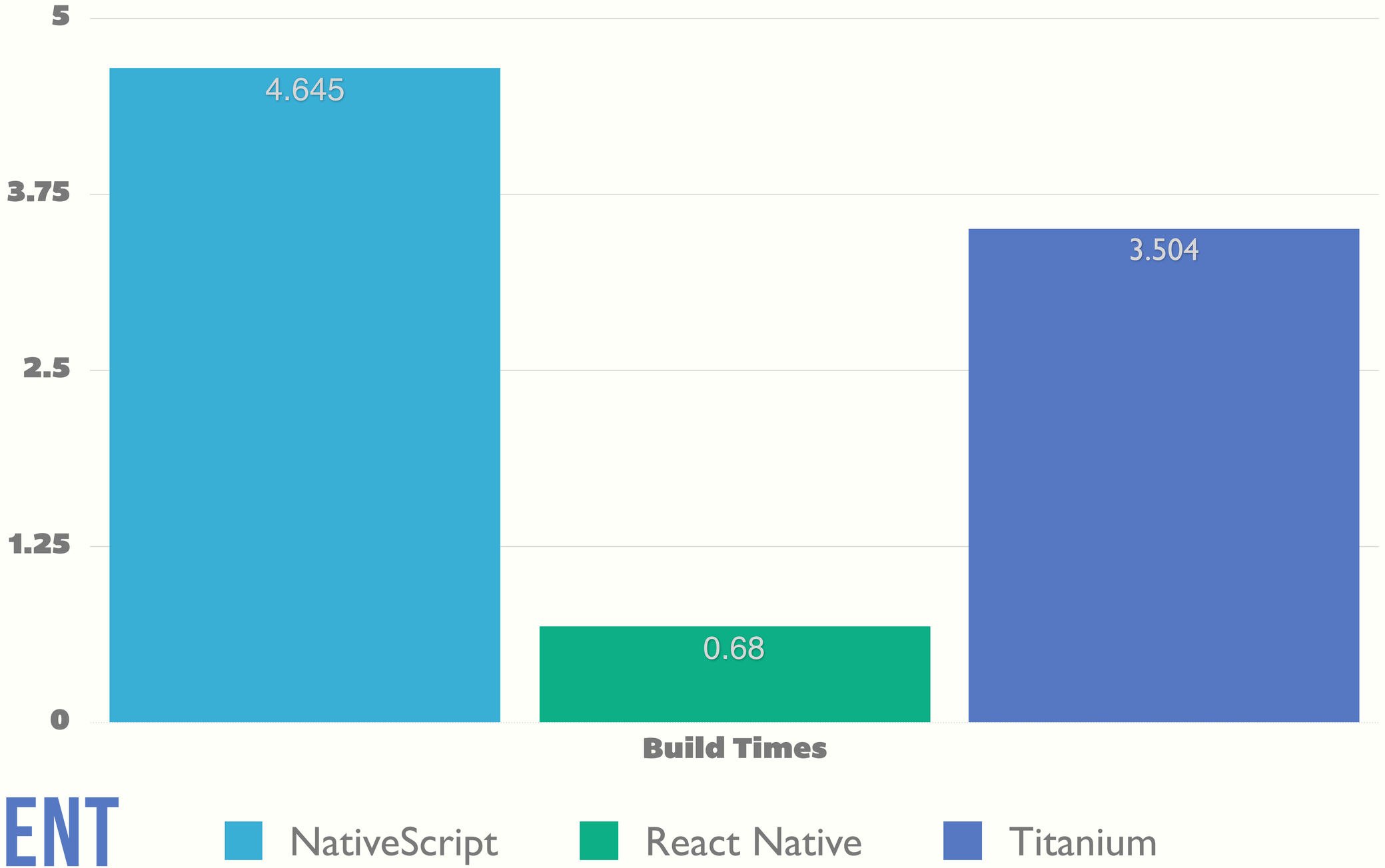
- HTTPS://FACEBOOK.GITHUB.IO/REACT-NATIVE/DOCS/GETTING-STARTED.HTML
- HTTPS://DOCS.NATIVESCRIPT.ORG/GETTING-STARTED#INSTALL-NATIVESCRIPT-AND-CONFIGURE-YOUR-ENVIRONMENT
- HTTP://DOCS.APPCELERATOR.COM/PLATFORM/LATEST/#!/GUIDE/INSTALLATION_AND_CONFIGURATION

BUILD TIMES

INITIAL



BUILD TIMES SUBSEQUENT



BUILD TIMES: LIVE RELOAD

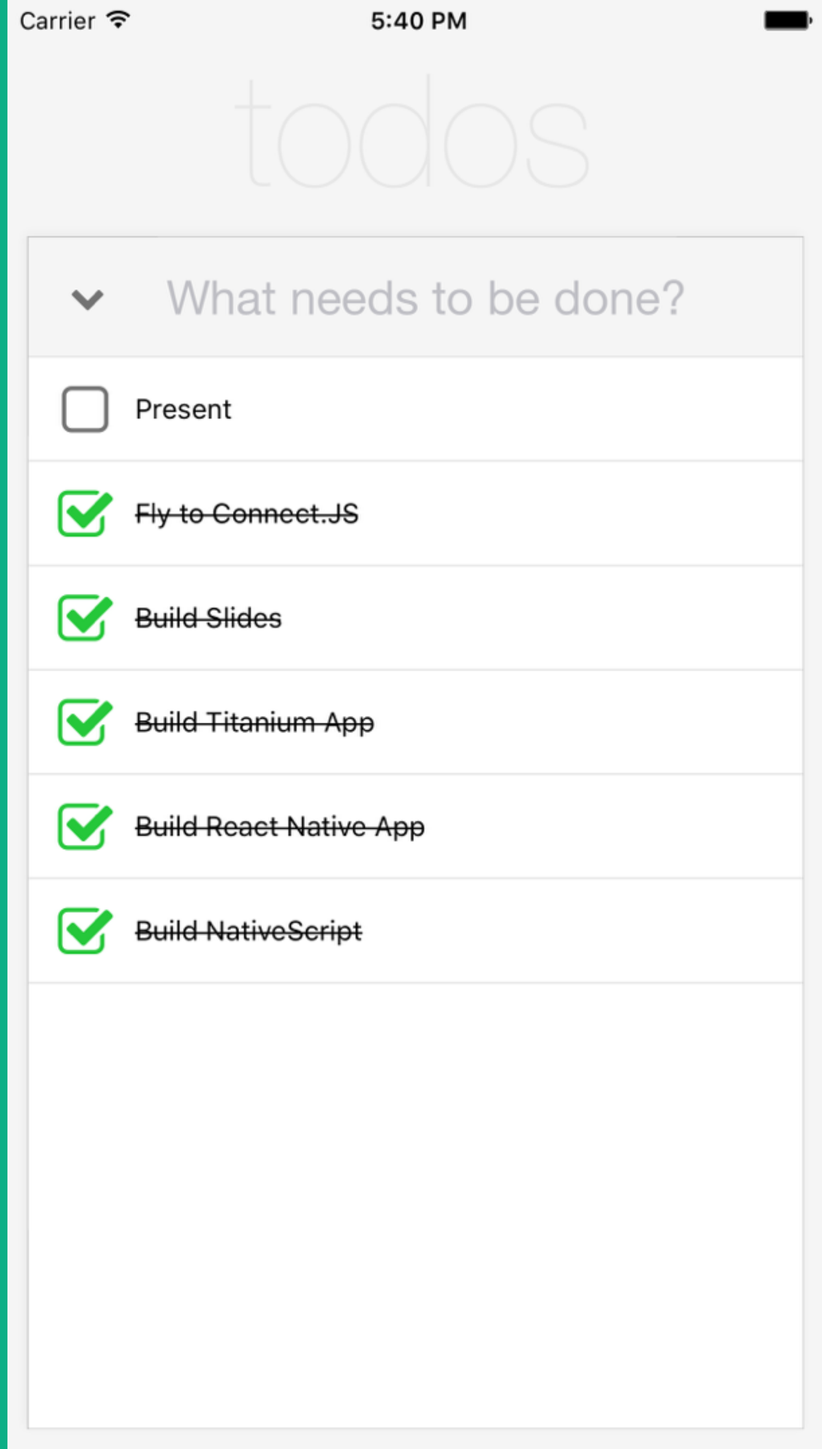
NATIVESCRIPT: INSTANT

REACT NATIVE: INSTANT

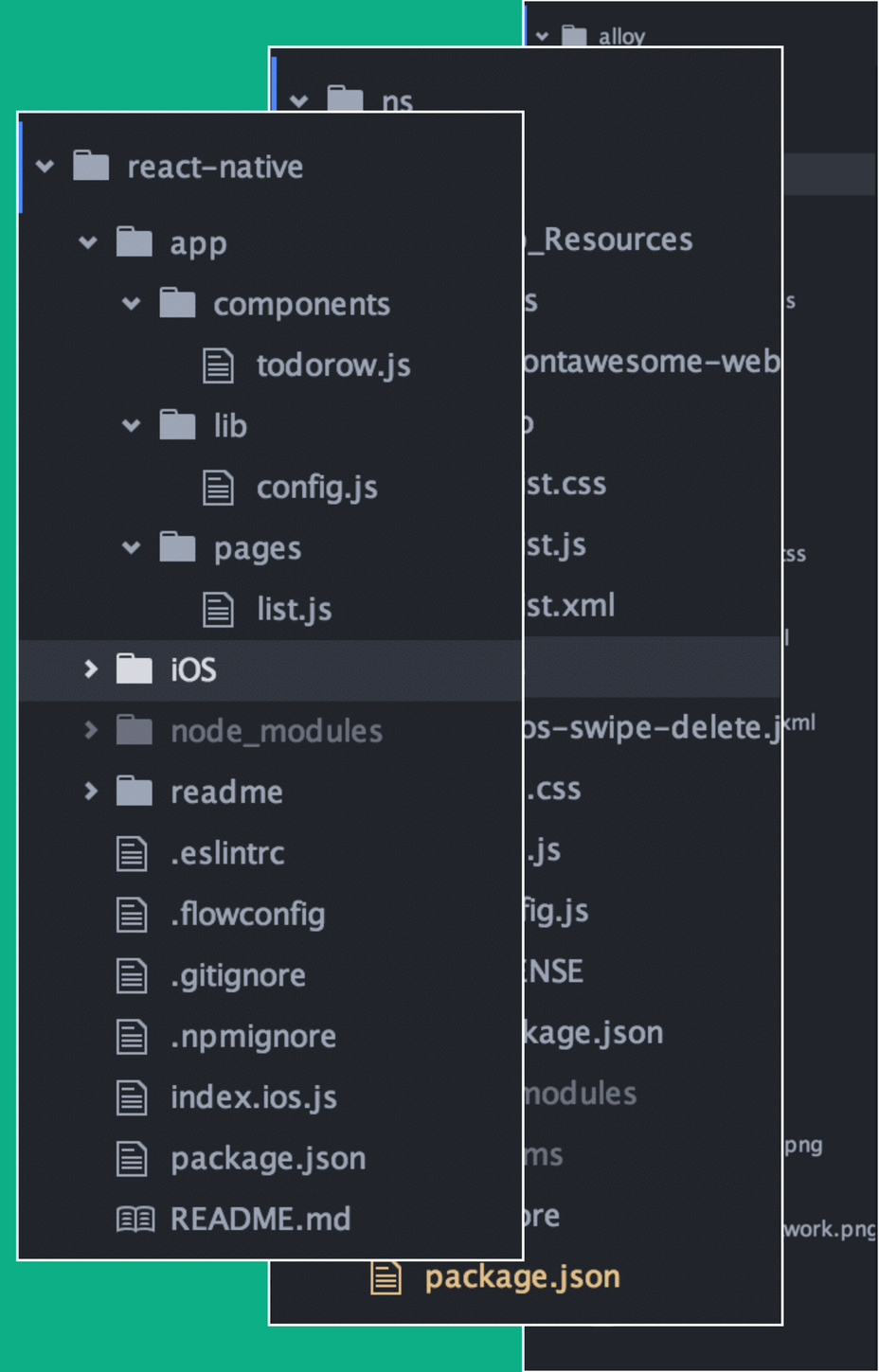
TITANIUM: 3+ SECONDS (REQUIRES SUBSCRIPTION OR 3RD PARTY TOOLING)

MOBILEJS.IO TODO APP

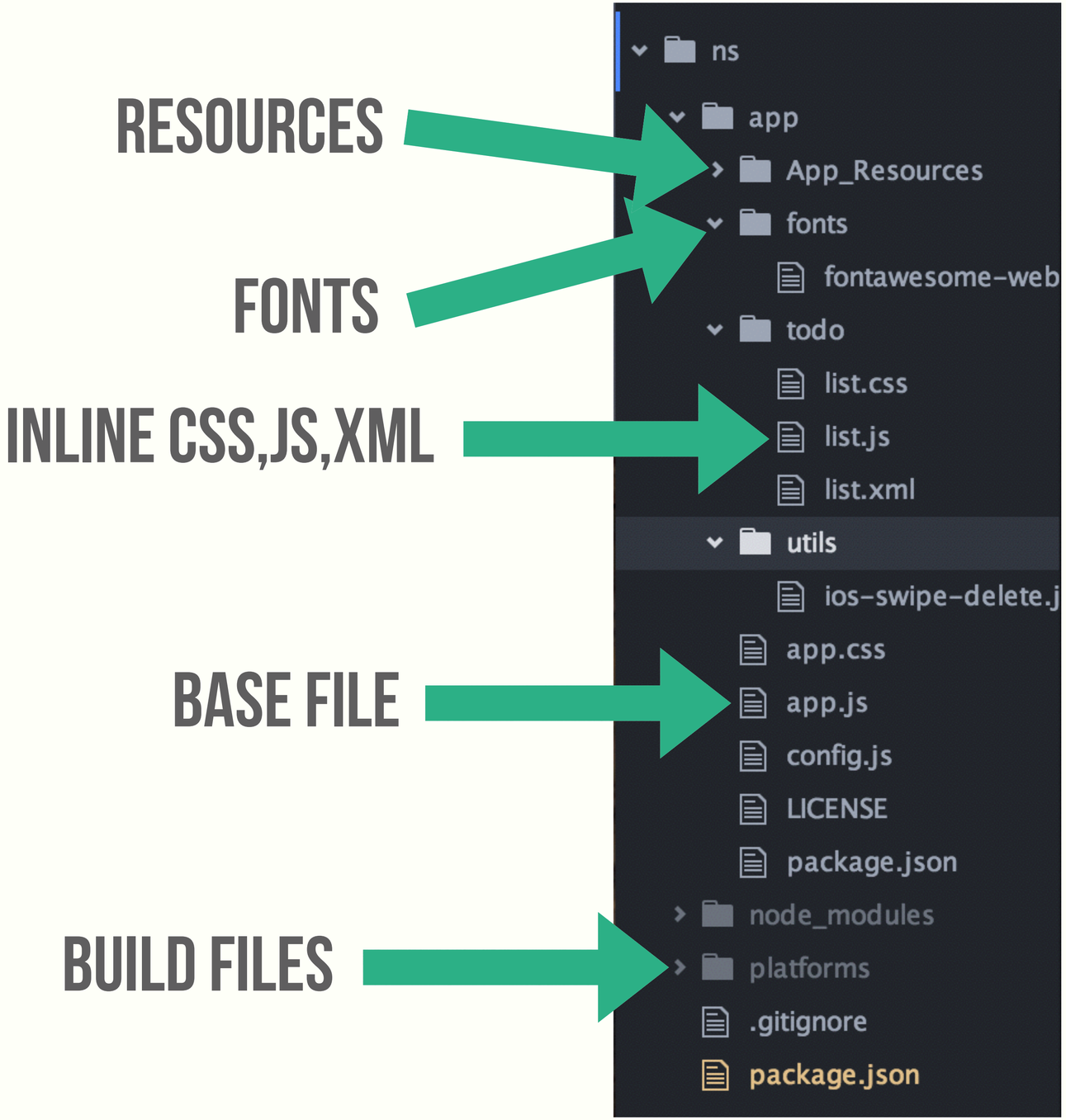
HTTP://MOBILEJS.IO - SOURCE



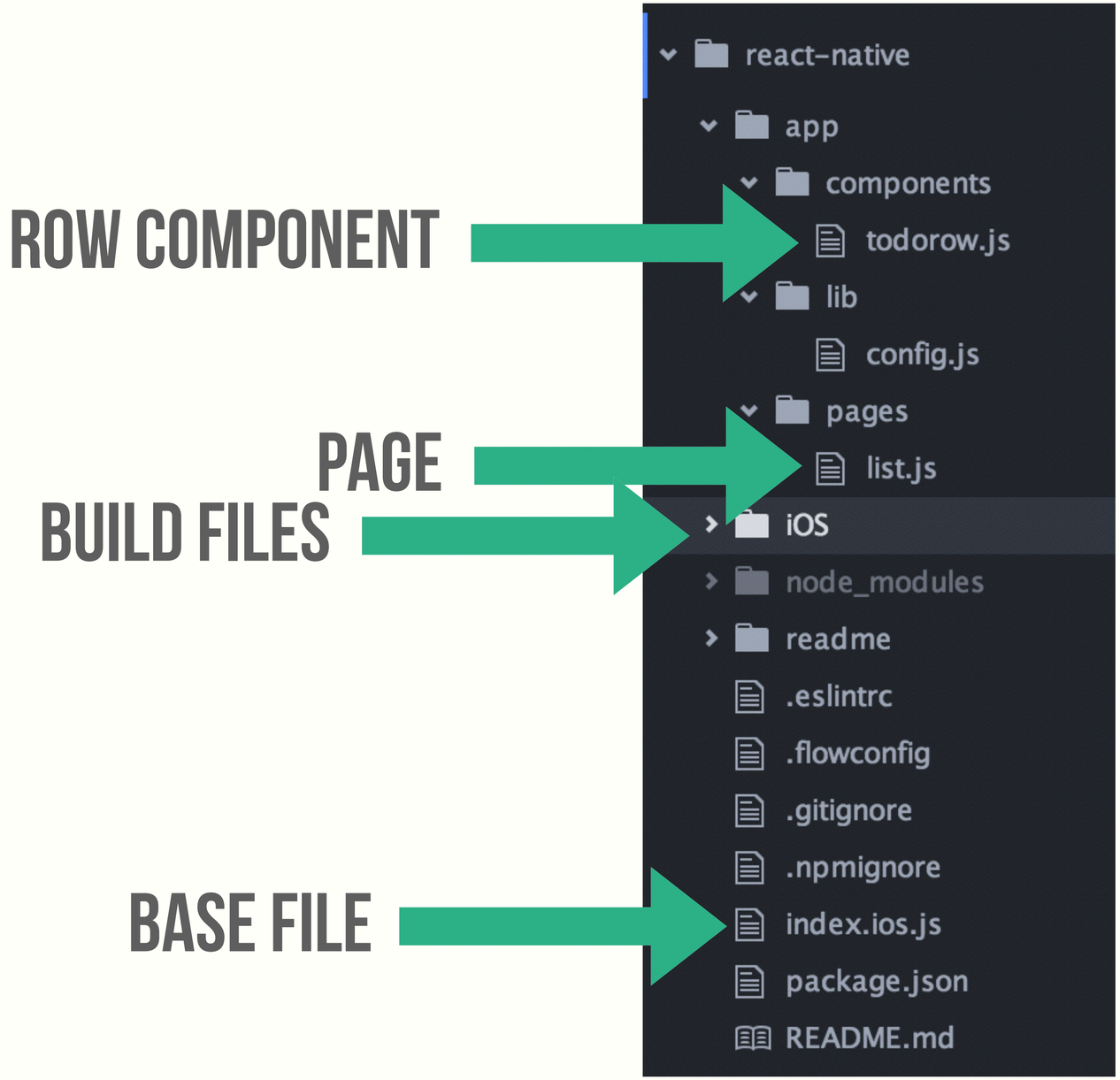
APPLICATION STRUCTURE



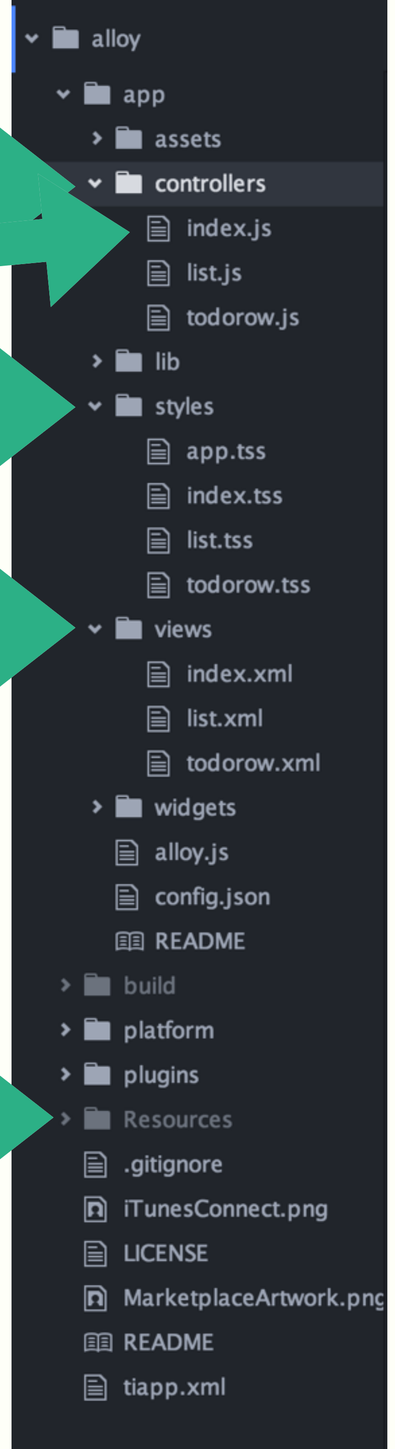
APPLICATION STRUCTURE NATIVESCRIPT



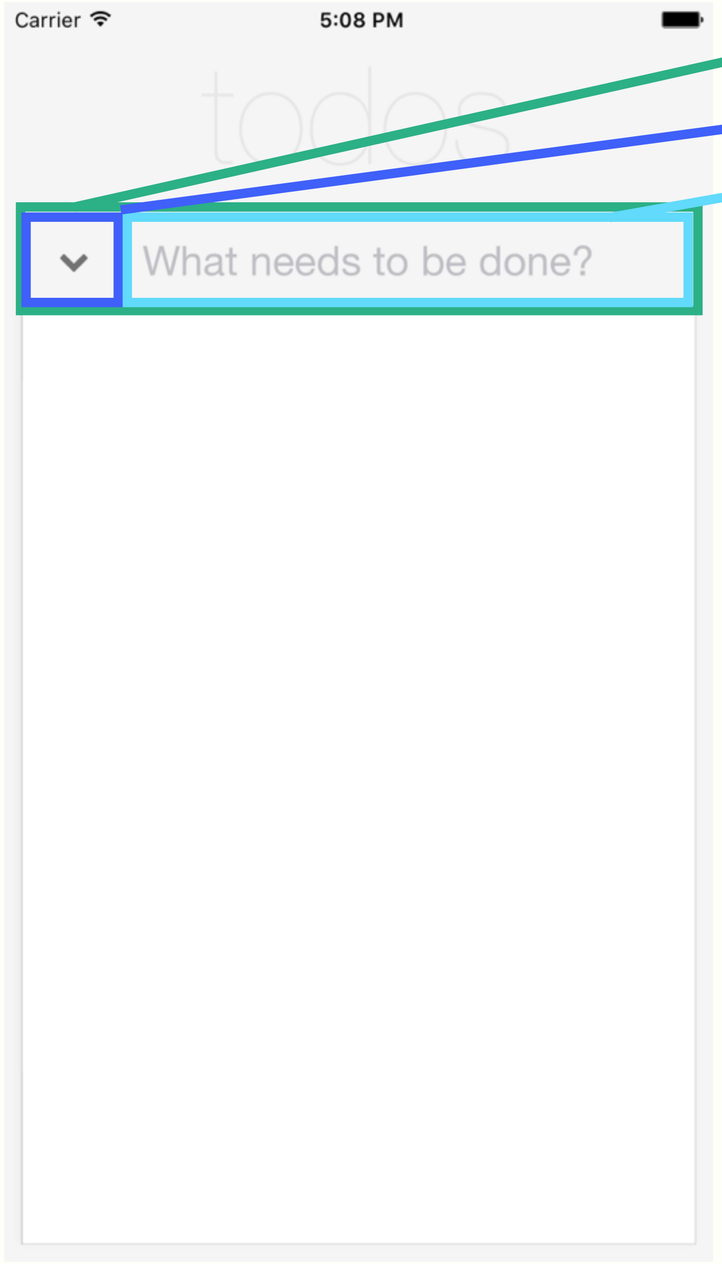
APPLICATION STRUCTURE
REACT NATIVE



APPLICATION STRUCTURE TITANIUM



LAYOUT METHODS




```
<StackLayout orientation="horizontal" id="formWrapper">
  <Label id="selectAllIcon" />
  <TextField id="textInput" />
</StackLayout>
```

LAYOUTS

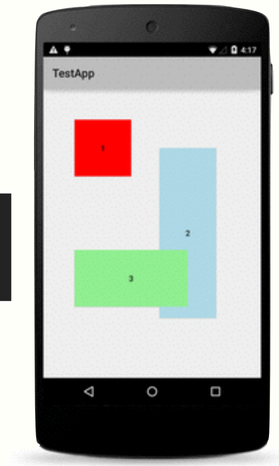
NATIVESCRIPT

```
#formWrapper {
  margin-right: 10;
  margin-top: 10;
  margin-left: 10;
}

#textInput {
  height: 60;
  padding: 10;
}

#selectAllIcon {
  margin-left: 18;
 width: 30;
  height: 30;
}
```

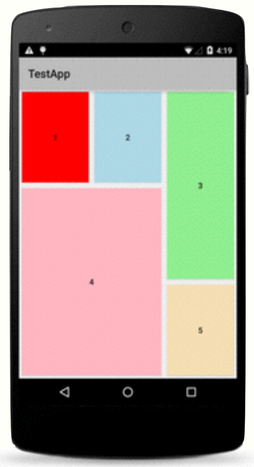
LAYOUTS

NATIVESCRIPT

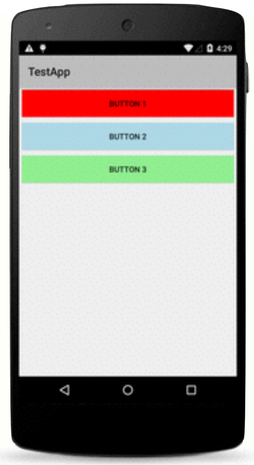
`<AbsoluteLayout />`



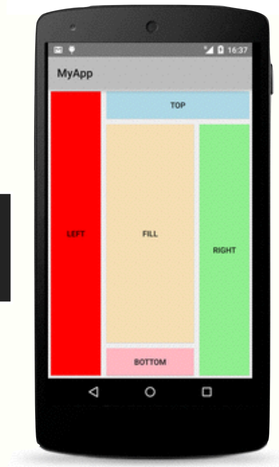
`<GridLayout />`



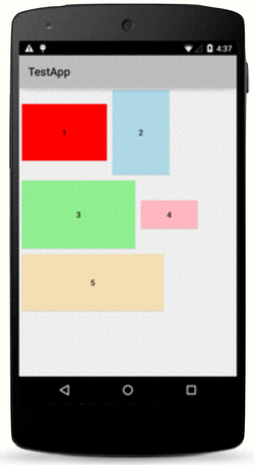
`<StackLayout />`

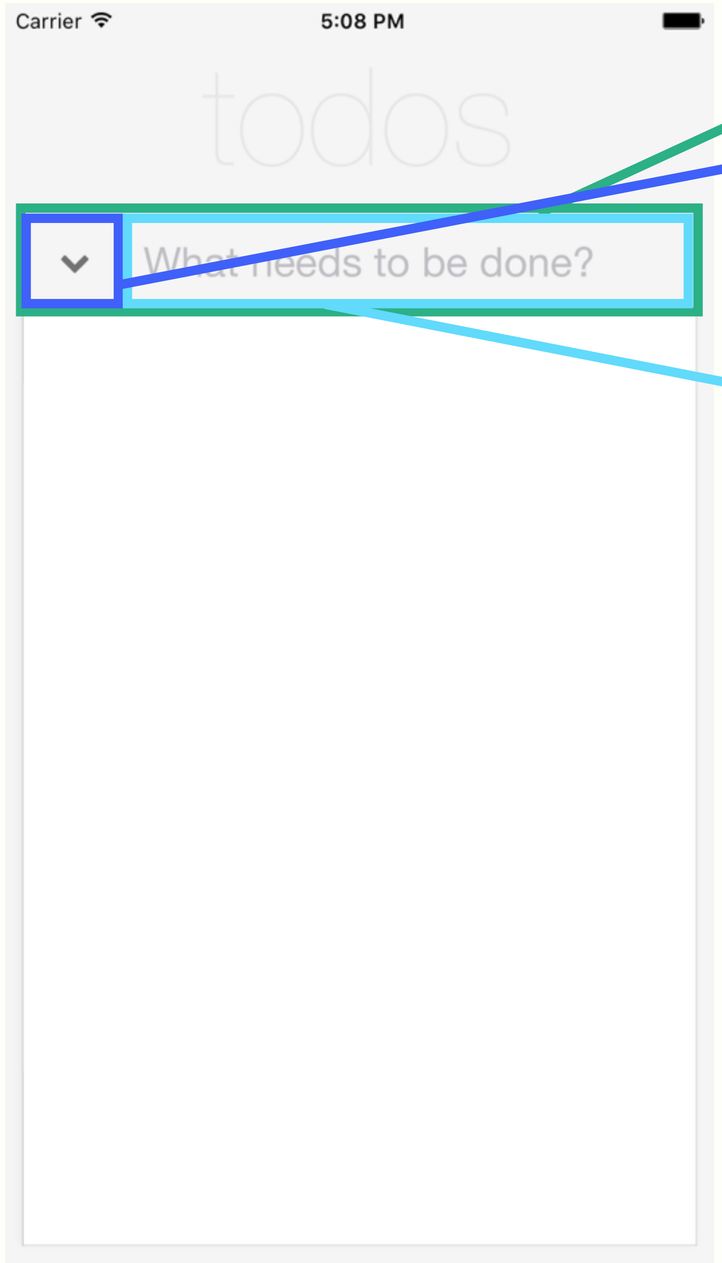


`<DockLayout />`



`<WrapLayout />`

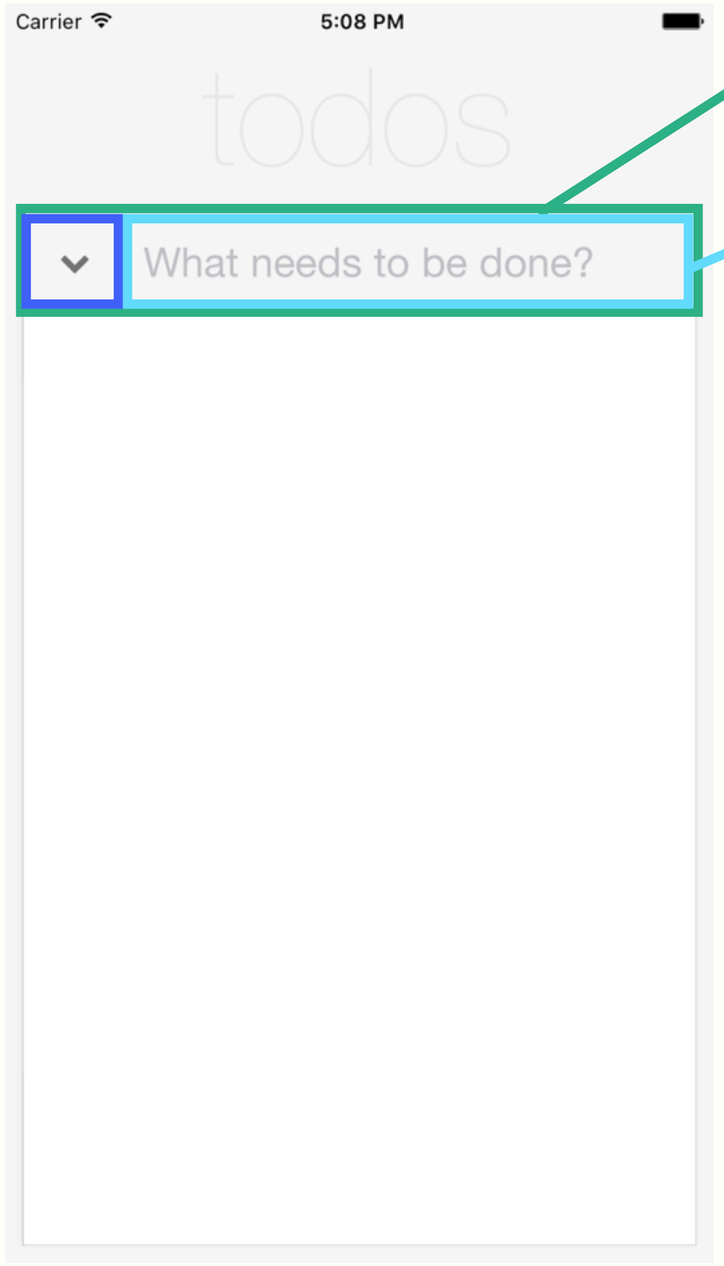





```
<View style={styles.formWrapper}>
  <Icon
    name='fontawesome|chevron-down'
    style={styles.selectAllIcon}
    />
  <TextInput
    ref="textInput"
    style={styles.textInput}
  />
</View>
```

```
formWrapper: {
  flexDirection: 'row'
},
selectAllIcon: {
  alignSelf: 'center',
  width: 40,
  height: 40
},
textInput: {
  flex: 1,
  height: 60,
  padding: 10,
}
```

LAYOUTS REACT NATIVE




```
<View id="formWrapper">
    <Label id="selectAllIcon"></Label>
    <TextField id="textInput" />
</View>
```

LAYOUTS

TITANIUM

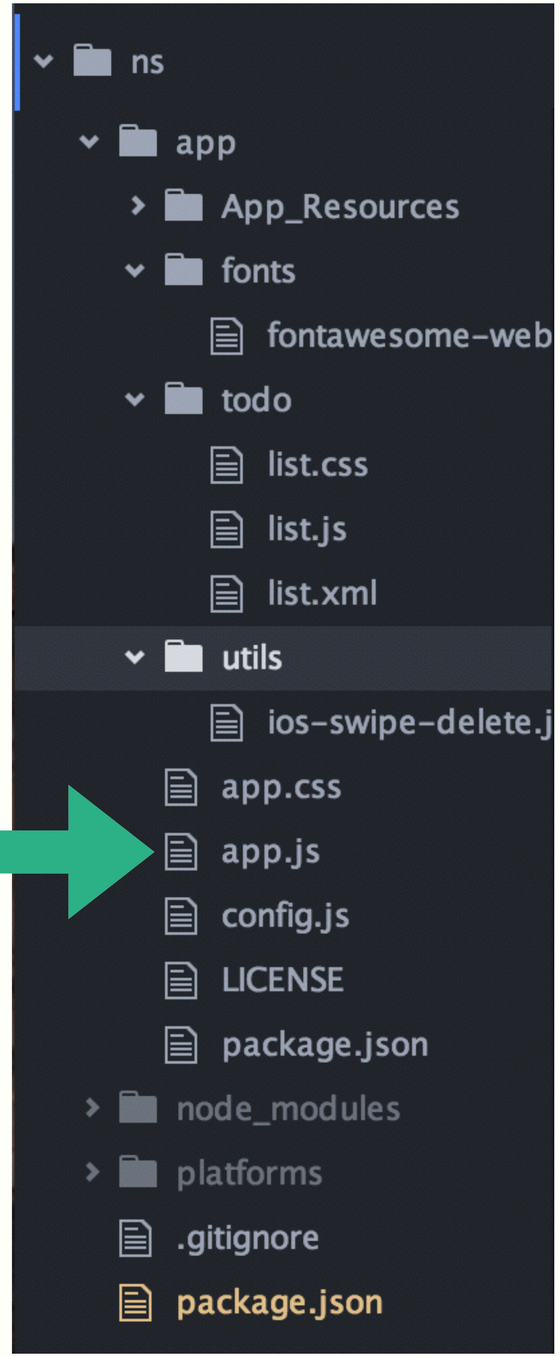
```
"#formWrapper": {
  top: 0,
  height: Ti.UI.SIZE,
  layout: "horizontal"
}

"#selectAllIcon": {
  left: 10,
  height: 60,
  width: 30
}

"#textInput": {
  width: Ti.UI.FILL,
  height: 60
}
```

INITIALIZING AN APPLICATION

```
var application = require('application');

application.mainModule = 'todo/list';
application.cssFile = './app.css';

application.start();
```

INITIALIZING AN APPLICATION NATIVESCRIPT

BASE FILE

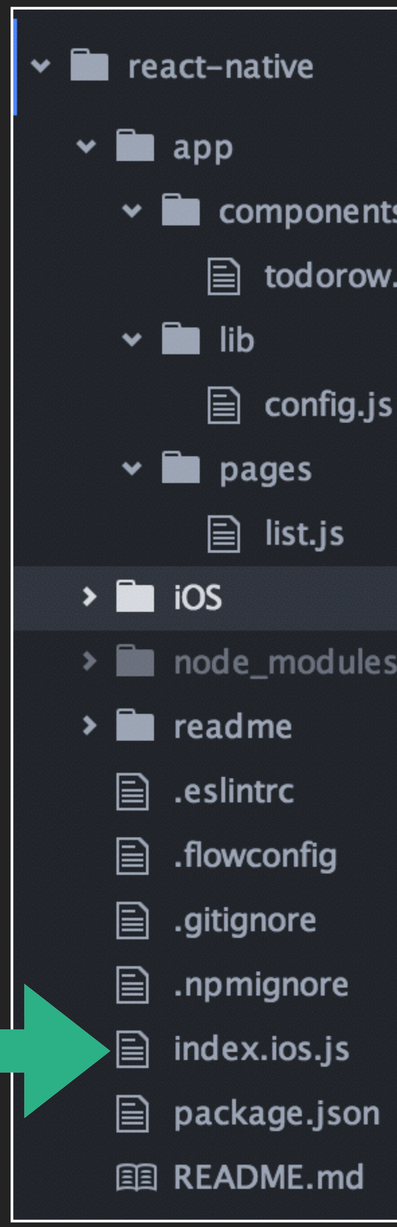


INITIALIZING AN APPLICATION

REACT NATIVE

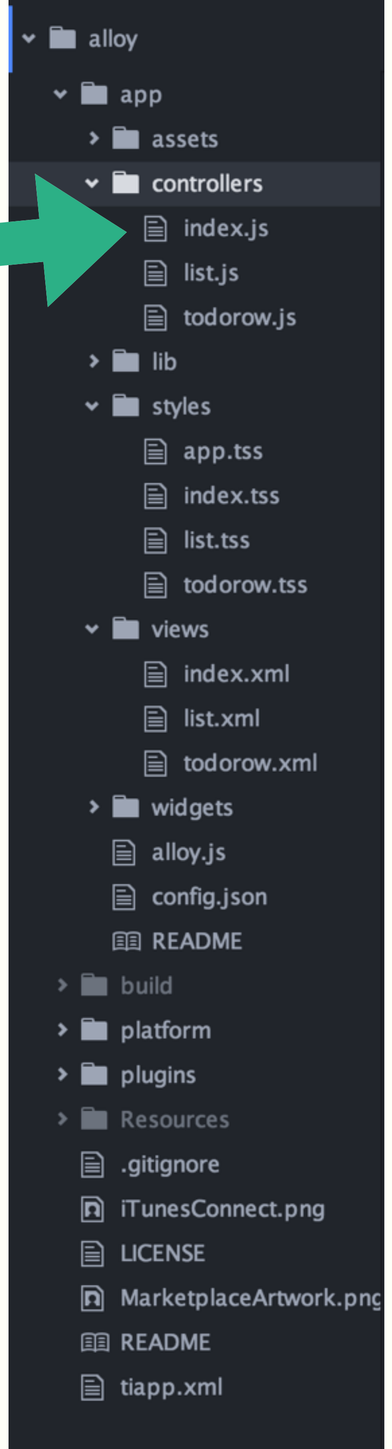
```
var React = require('react-native');
var ListPage = require('./app/pages/list');

var {AppRegistry, StyleSheet,
NavigatorIOS} = React;

var todomjs = React.createClass({
  render: function() {
    return (
      <NavigatorIOS
        style={styles.topLevelNavigator}
        navigationBarHidden={true}
        initialRoute={{
          title: '',
          component: ListPage
        }}
      />
    );
  }
});

AppRegistry.registerComponent('todomjs', () => todomjs);
```

BASE FILE



```
var application = require('application');

if (OS_IOS) {
  application.navWindow = $.navWindow;
}

$.navWindow.open();
application.init();
```

BASE FILE

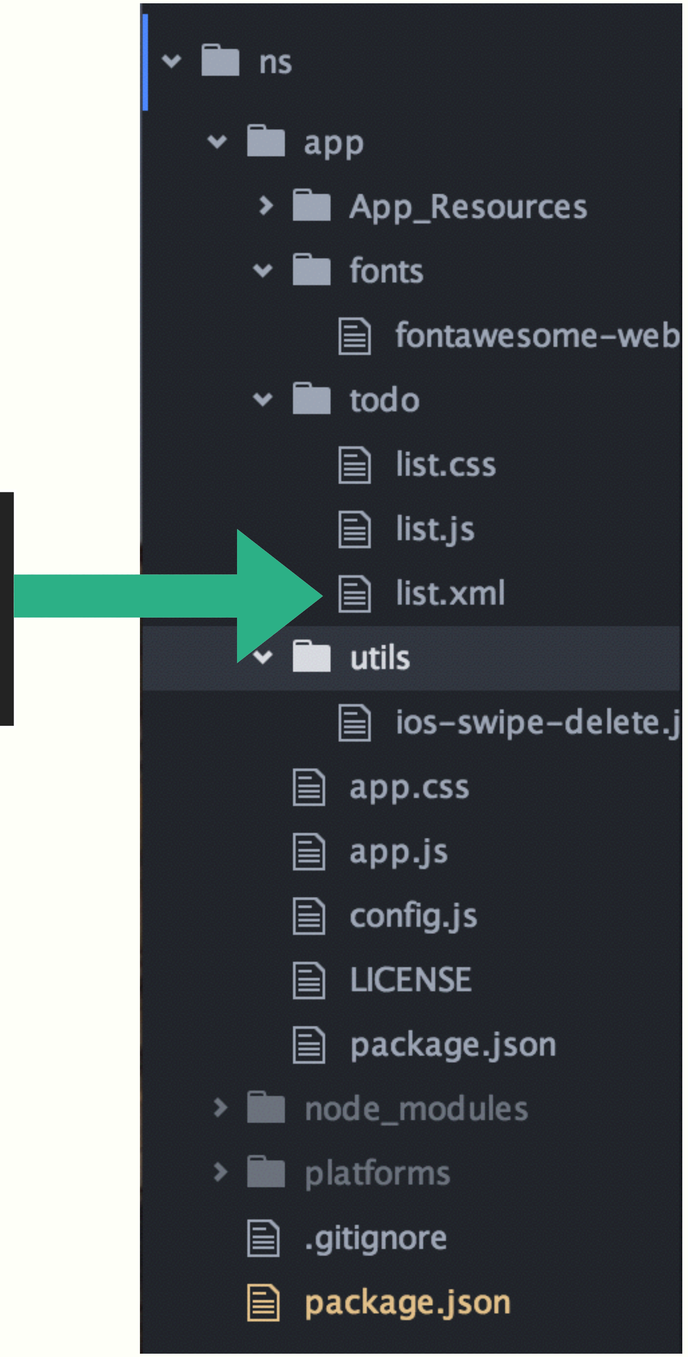


INITIALIZING AN APPLICATION

TITANIUM

GETTING TO HELLO WORLD

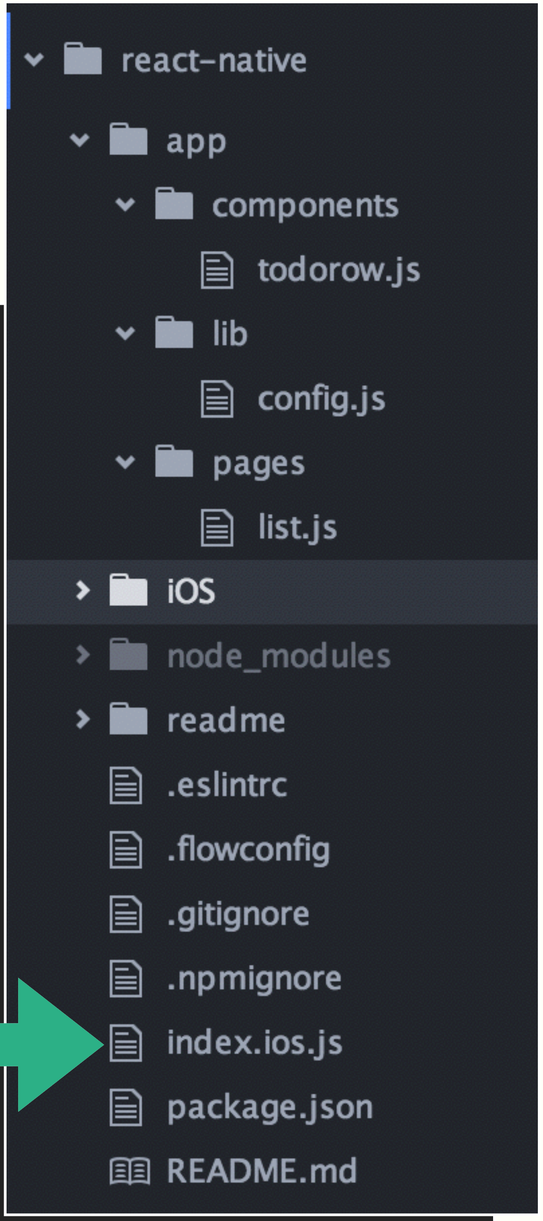
```
<Page xmlns="http://www.nativescript.org/tns.xsd">
  <Label text="Hello World"/>
</Page>
```



GETTING TO HELLO WORLD NATIVESCRIPT

GETTING TO HELLO WORLD REACT NATIVE

```
var React = require('react-native');

var {
  AppRegistry,
  Text
} = React;

var todomjs = React.createClass({
  render: function() {
    return (
      <Text>Hello World</Text>
    );
  }
});

AppRegistry.registerComponent('todomjs', () => todomjs);
```

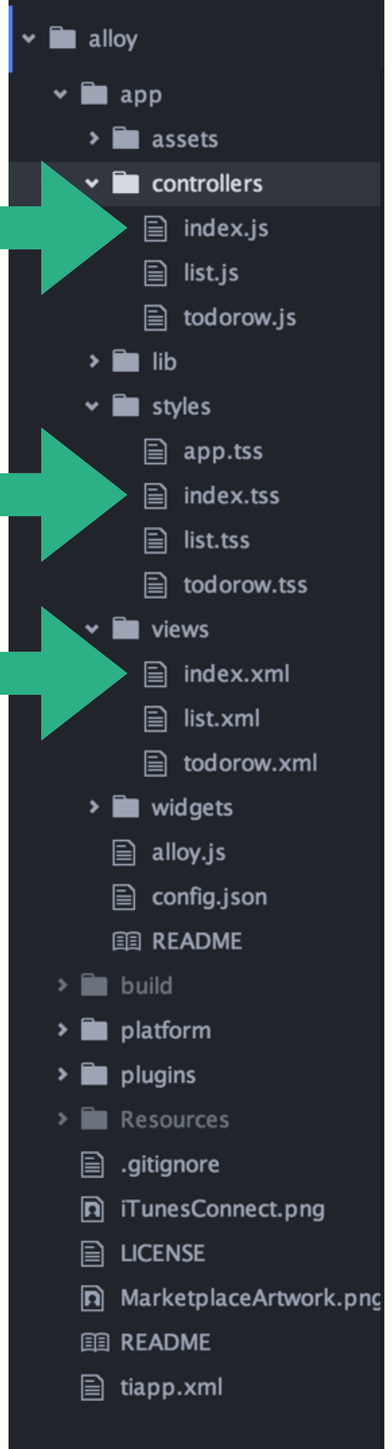


GETTING TO HELLO WORLD TITANIUM

```
$.getView().open();
```

```
"Window": {
  backgroundColor: "#fff"
}
```

```
<Alloy>
  <Window>
    <Label>Hello World</Label>
  </Window>
</Alloy>
```

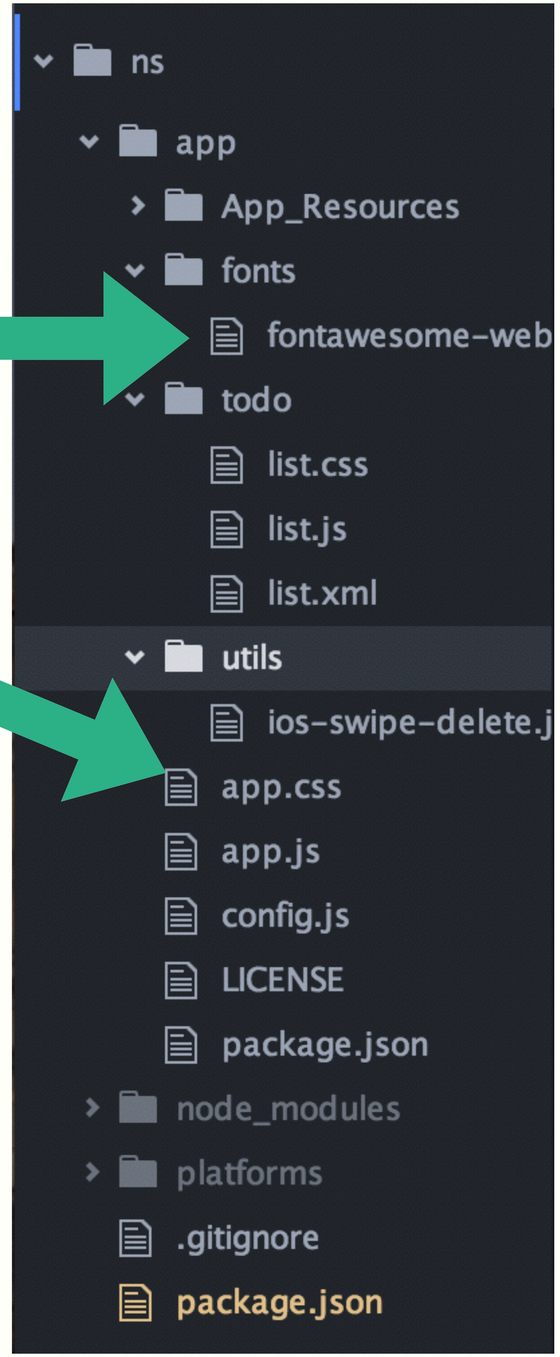


ADDING CUSTOM FONTS

ADDING CUSTOM FONTS NATIVESCRIPT

```
.fa {
  font-family: FontAwesome;
}
```

FONT FILE



ADDING CUSTOM FONTS REACT NATIVE

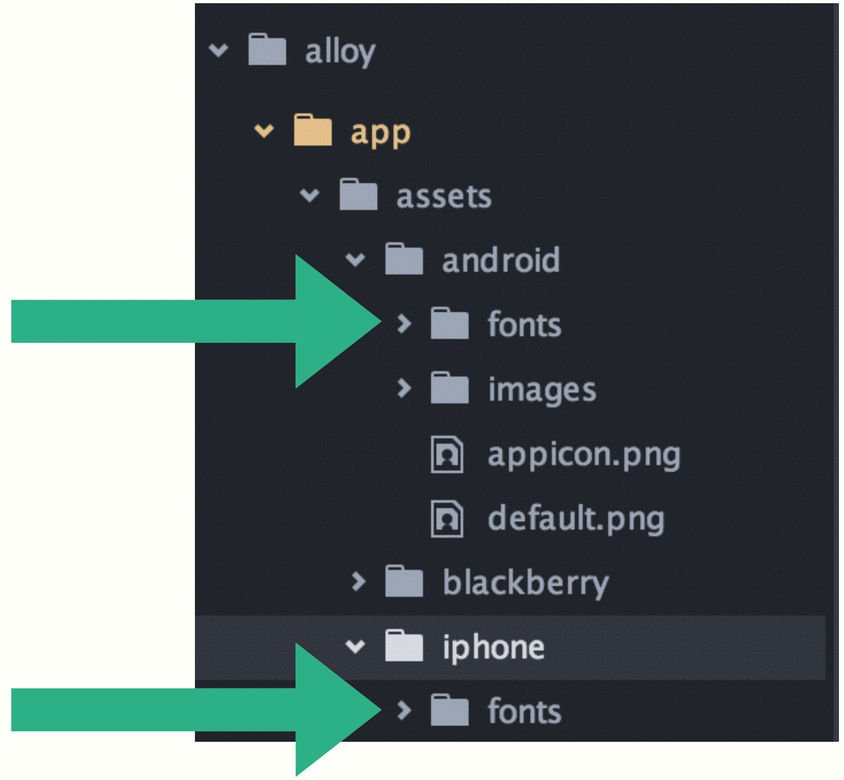
1. ADD THE FONT FILES TO YOUR RESOURCE FILES
2. EDIT YOUR INFO.PLIST: ADD A NEW ENTRY WITH THE KEY FONTS PROVIDED BY APPLICATION.
3. FOR EACH OF YOUR FILES, ADD THE FILE NAME TO THIS ARRAY

Main nib file base name	MainWindow
Status bar is initially hidden	☐
▼ Fonts provided by application	(4 items)
Item 0	DejaVuSansMono.ttf
Item 1	DejaVuSansMono-Bold.ttf
Item 2	DejaVuSansMono-Oblique.ttf
Item 3	DejaVuSansMono-BoldOblique.ttf

Source: http://stackoverflow.com/questions/4969329/how-to-include-and-use-new-fonts-in-iphone-sdk

ADDING CUSTOM FONTS TITANIUM

1. ADD THE FONT FILES TO APP > ASSETS
    1. > IPHONE > FONTS
    2. > ANDROID > FONTS



LABEL

LABEL

NATIVESCRIPT

```
<Label text="Todo" id=“header" />
```

```
#header {
  font-family: "HelveticaNeueThin";
  font-size: 80;
  text-align: center;
  color: #d2d2d2;
  right: 10;
}
```

LABEL

REACT NATIVE

```
<Text style={styles.header}>todos</Text>
```

```
header: {
  fontFamily: 'Helvetica Neue',
  fontWeight: '100',
  fontSize: 80,
  textAlign: 'center',
  color: '#e7e7e7',
  marginTop: 20
}
```

LABEL
TITANIUM

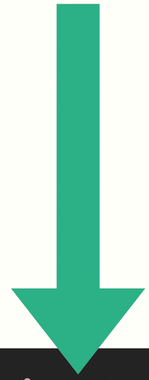
```
<Label id="header">Todos</Label>
```

```
"#header": {
  top: 20,
  color: "#e7e7e7",
  textAlign: "center",
   font: {
       fontFamily: "HelveticaNeue-Thin",
       fontSize: 80,
   }
}
```

LISTVIEW / TABLE VIEW

LISTVIEW

NATIVESCRIPT



LIST.XML

```
<Page xmlns="http://www.nativescript.org/tns.xsd" navigatedTo="navigatedTo">
</Page>
```

LIST.JS

```
exports.navigatedTo = function(args) {
  var pageData = new observableModule.Observable();
  page.bindingContext = pageData;
}
```

LISTVIEW

NATIVESCRIPT

LIST.JS

```
pageData.set("todoItems", todoItems);
```

LIST.XML

```
    <ListView items="{{ todoItems }}" id="listView" itemTap="rowOnPress"
separatorColor="#fff">
        <ListView.itemTemplate>
            <StackLayout orientation="horizontal" cssClass="todo-row">
                <Label text="{{ isChecked ? '' : '' }}" color="{{ iconColor }}"
tap="onPressCheckbox" cssClass="fa checkbox"  />
                <Label text="{{ text }}" />
            </StackLayout>
        </ListView.itemTemplate>
    </ListView>
```

LISTVIEW NATIVESCRIPT

LIST.JS

```
todoItems.unshift(_.extend({
  text: text,
  children: new observableArray.ObservableArray([])
}, config.rowTypes.notDone));
```

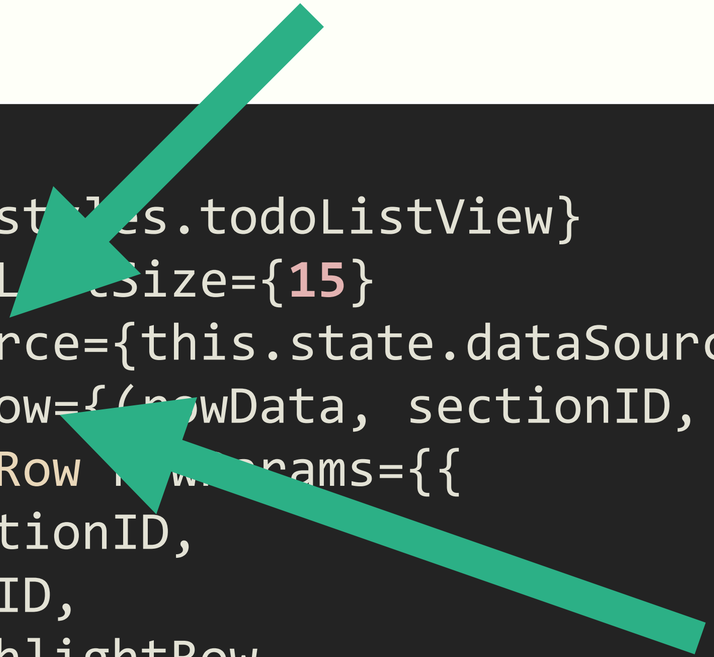
LIST.XML

```
    <ListView items="{{ todoItems }}" id="listView" itemTap="rowOnPress"
separatorColor="#fff">
        <ListView.itemTemplate>
            <StackLayout orientation="horizontal" cssClass="todo-row">
                <Label text="{{ isChecked ? '' : '' }}" color="{{ iconColor }}"
tap="onPressCheckbox" cssClass="fa checkbox"  />
                <Label text="{{ text }}" />
            </StackLayout>
        </ListView.itemTemplate>
    </ListView>
```

LISTVIEW REACT NATIVE

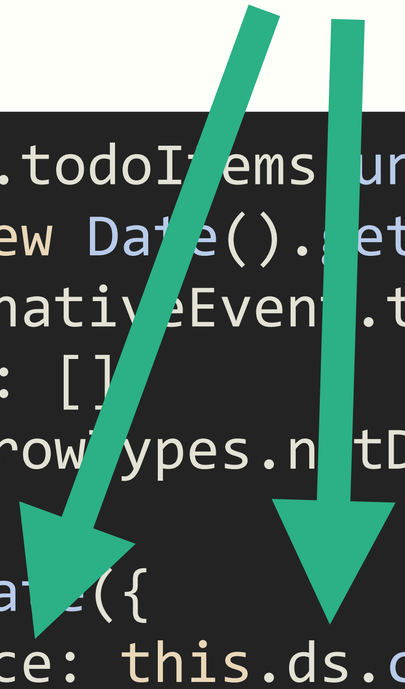
```
var ListPage = React.createClass({
  getInitialState: function() {
    this.ds = new ListView.DataSource({rowHasChanged: (r1, r2) =>
r1.rowID !== r2.rowID});

    return {
      todoItems: this.props.todoItems || [],
      dataSource: this.ds.cloneWithRows(this.props.todoItems || []),
    };
  }
}
```

LISTVIEW REACT NATIVE



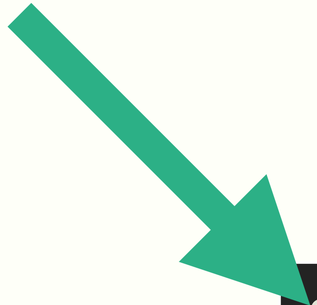
```
<ListView
  style={styles.todoListView}
  initialListSize={15}
  dataSource={this.state.dataSource}
  renderRow={(rowData, sectionID, rowID, highlightRow) => (
    <TodoRow rowParams={{
      sectionID,
      rowID,
      highlightRow,
      updateRow: this.updateRow,
      deleteRow: this.deleteRow,
      rowOnPress: this.rowOnPress
    }} rowData={rowData} />
  )}
 automaticallyAdjustContentInsets={false}
/>
```

LISTVIEW REACT NATIVE



```
this.state.todoItems.unshift(_.extend({
  rowID: new Date().getTime(),
  text: e.nativeEvent.text,
  children: []
}, config.rowTypes.notDone));

this.setState({
  dataSource: this.ds.cloneWithRows(this.state.todoItems)
}, function() {
  this.clearTextInput();
});
```

TABLEVIEW
TITANIUM



LIST.XML

```
<TableView id="todoTable" />
```

LIST.JS

```
$.todoItems = args.todoItems || [];

function createRow(e) {
  $.todoItems.unshift(_.extend({
    text: e.value,
    children: []
  }, config.rowTypes.notDone));

  $.todoTable.setData(buildRows($.todoItems));
}
```

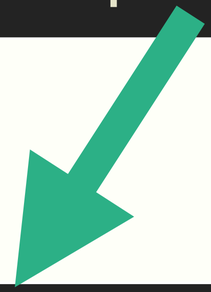
TAP EVENT HANDLERS

TAP EVENT HANDLERS

NATIVESCRIPT

LIST.XML

```
<Label text="back" tap="onBackTap" />
```



LIST.JS

```
exports.onBackTap = function() {
};
```

TAP EVENT HANDLERS

REACT NATIVE

LIST.JS

```
<TouchableOpacity
  onPress={() => this.backOnPress()}
  activeOpacity={0.2}
  >
  <Text>back</Text>
</TouchableOpacity>
```

TAP EVENT HANDLERS

TITANIUM

LIST.XML

```
<Label id="backButton"></Label>
```

LIST.JS

```
$.backButton.addEventListener('click', backOnPress);
```

TAP EVENT HANDLERS

NATIVESCRIPT: TAP

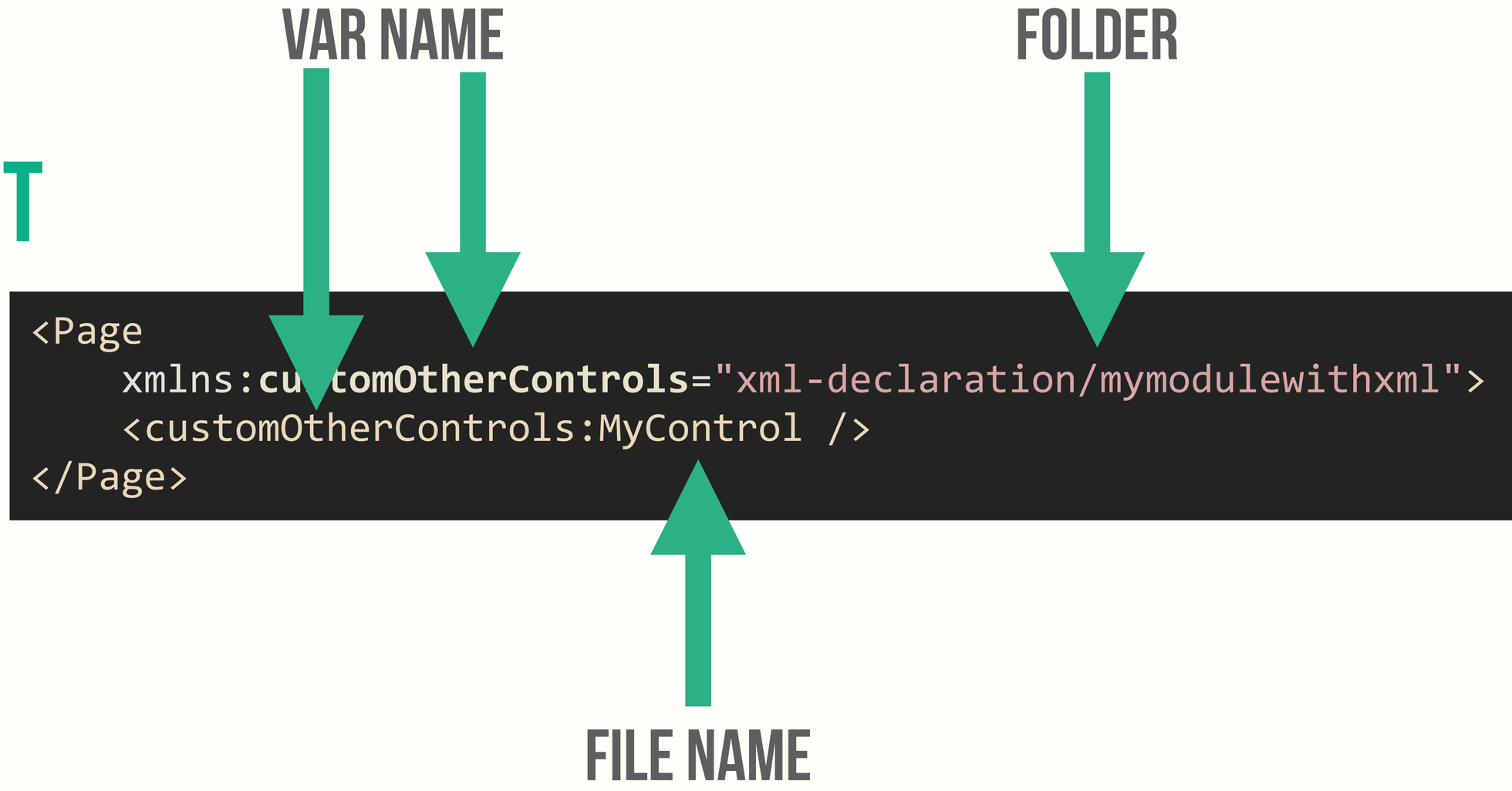
REACT NATIVE: PRESS

TITANIUM: CLICK OR SINGLETAP

CODE REUSE

CODE REUSE

NATIVESCRIPT



Source: http://docs.nativescript.org/ui-with-xml#example-custom-xml-based-component-with-code-file

CODE REUSE

REACT NATIVE

```
{this.renderBackButton()}
```

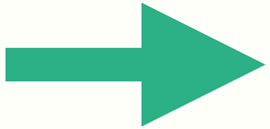
```
renderBackButton: function() {
  if (this.props.showBack) {
    return (
      <TouchableOpacity
        style={styles.touchableAreaBackIcon}
        onPress={() => this.backOnPress()}
        activeOpacity={0.2}
        >
        <Icon
          name='fontawesome|arrow-circle-o-left'
          size={30}
          color='#ead7d7'
          style={styles.backButtonIcon}
          />
      </TouchableOpacity>
    );
  }
}
```

CODE REUSE

REACT NATIVE

```
var TodoRow = require('./../components/todorow');
```

```
<TodoRow rowParams={{
  sectionID,
  rowID,
  highlightRow,
  updateRow: this.updateRow,
  deleteRow: this.deleteRow,
  rowOnPress: this.rowOnPress
}} rowData={rowData} />
```



```
this.props
```

CODE REUSE

TITANIUM

```
Alloy.createController('list', {});
```

```
var args = arguments[0] || {};
```

NAVIGATION

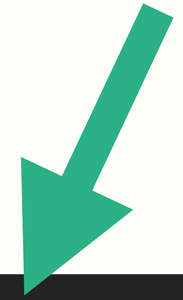
NAVIGATION

NATIVESCRIPT

```
var frameModule = require("ui/frame");

frameModule.topmost().navigate({
  moduleName: "todo/list",
  context: {
    rowID: args.index,
    updateRowChildren: updateRowChildren,
    todoItems: todoItems.getItem(args.index).children
  }
});
```

NAVIGATION / CODE REUSE

NATIVESCRIPT



```
<Page xmlns="http://www.nativescript.org/tns.xsd" navigatedTo="navigatedTo">
</Page>
```

```
exports.navigatedTo = function(args) {
  var page = args.object;
};
```

```
page.navigationContext
```

NAVIGATION

REACT NATIVE

```
this.props.navigator.push({
  title: '',
  component: require('./list'),
  passProps: {
    rowID: rowID,
    showBack: true,
    todoItems: this.state.todoItems[rowID].children,
    updateRowChildren: this.updateRowChildren
  }
});
```

NAVIGATION

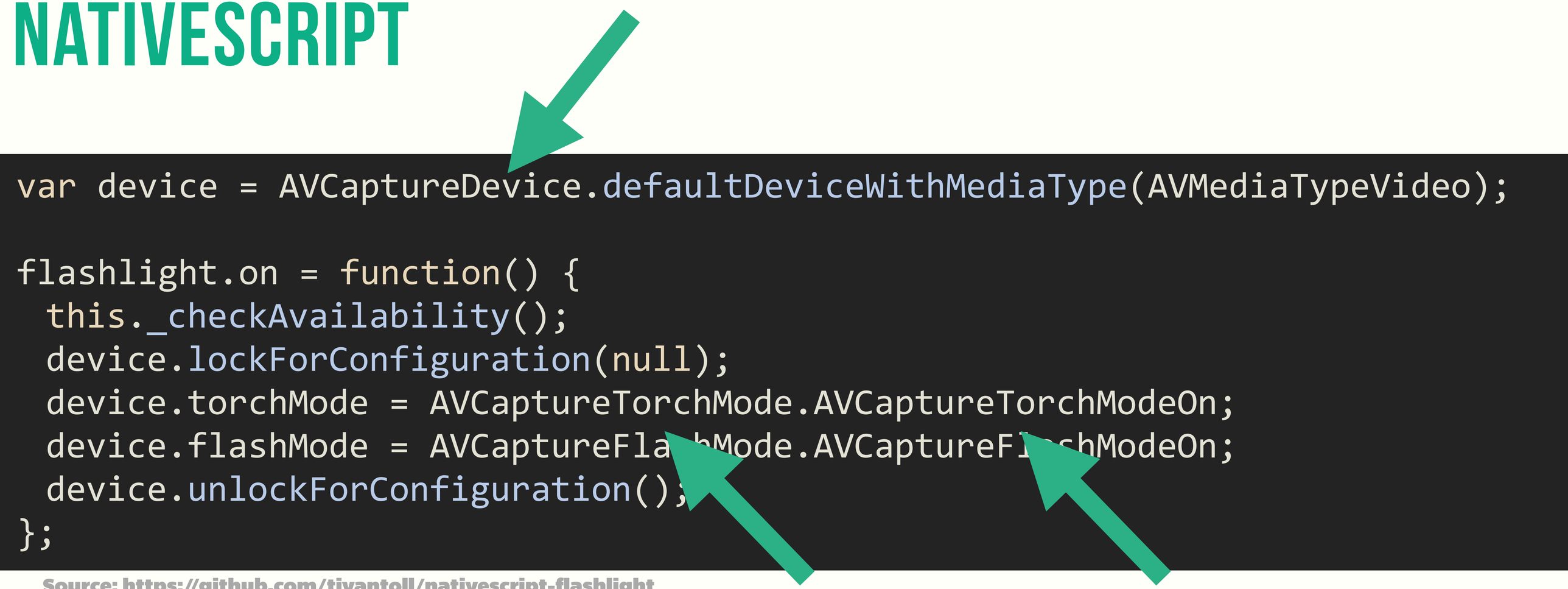
TITANIUM

```
var application = require('application');

if (OS_IOS) {
  application.navWindow = $.navWindow;
}
```

```
application.navWindow.openWindow(
  Alloy.createController('list', {
    rowID: e.index,
    todoItems: rowData.children || [],
    updateRowChildren: updateRowChildren
  }).getView()
);
```

EXTENDING EACH PLATFORM

EXTENDING NATIVESCRIPT



```
var device = AVCaptureDevice.defaultDeviceWithMediaType(AVMediaTypeVideo);

flashlight.on = function() {
  this._checkAvailability();
  device.lockForConfiguration(null);
  device.torchMode = AVCaptureTorchMode.AVCaptureTorchModeOn;
  device.flashMode = AVCaptureFlashMode.AVCaptureFlashModeOn;
  device.unlockForConfiguration();
};
```

Source: https://github.com/tjvantoll/nativescript-flashlight

EXTENDING REACT NATIVE

```
var React = require('react-native');

var { NativeModules } = React;

var { RNControlFlashlight } = NativeModules;

RNControlFlashlight.turnFlashlight("flashlightOn", function
errorCallback(results) {
      console.log('JS Error: ' + results['errMsg']);
  }, function successCallback(results) {
      console.log('JS Success: ' + results['successMsg']);
  }
);
```

Source: https://github.com/rhaker/react-native-control-flashlight-ios

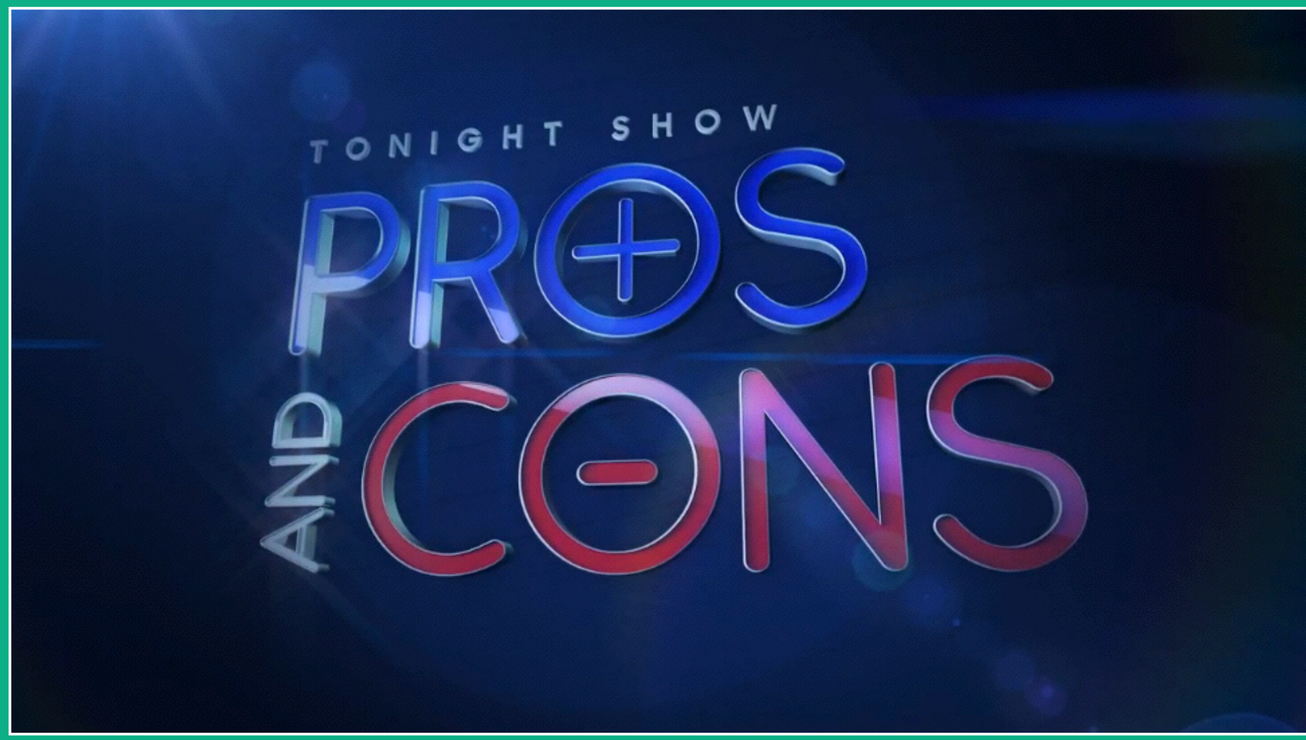
EXTENDING TITANIUM

```
-(void)turnFlashLightOn:(id)args
{
    if ([device hasTorch]) {
        [device lockForConfiguration:nil];
        [device setTorchMode:AVCaptureTorchModeOn];
        [device unlockForConfiguration];
    }
}
```

```
var NappFlashLight = require('NappFlashLight');

NappFlashLight.turnFlashLightOn()
```

Source: https://github.com/viezel/NappFlashLight

PROS & CONS



PROS & CONS NATIVESCRIPT

BEST FEATURE: DIRECT API ACCESS

Pros	Cons
DIRECT API ACCESS	NO JS ENCRYPTION*
EXCELLENT SUPPORT AND RESPONSE TIME TO TICKETS	RATHER YOUNG
OPEN SOURCE	YOU REALLY NEED TO KNOW TYPESCRIPT
CROSS-PLATFORM UI ABSTRACTIONS	LIMITED CROSS-PLATFORM UI ABSTRACTIONS
LARGE, STABLE, OLD COMPANY BACKING	DIFFICULT DOCUMENTATION

PROS & CONS REACT NATIVE

BEST FEATURE: LIVE RELOAD

Pros	Cons
LIVE RELOAD / NO COMPILE TIME	NO JS ENCRYPTION
GROWING COMMUNITY	YOUNG
COMPLETELY OPEN SOURCE	ES6 FLUENCY IS ALMOST REQUIRED
EXPRESSIVE MARKUP	REACT METHODOLOGY LEARNING CURVE
COMPONENT MODULARITY	LIMITED CROSS-PLATFORM APIS
	MODULES REQUIRED FOR NATIVE API ACCESS

PROS & CONS TITANIUM

BEST FEATURE: MATURITY

Pros	Cons
JS SOURCE ENCRYPTION	TECHNICAL DEBT
VERY MATURE PLATFORM	OSS VERSION IS BEHIND
LARGEST COMMUNITY	MODULES REQUIRED FOR NATIVE API ACCESS*
EXTENSIVE API DOCS	COMPILE TIME
EXTENSIVE CROSS-PLATFORM APIS	

NATIVE MOBILE APPS WITH JAVASCRIPT

THANK YOU

QUESTIONS?

646.876.2777
JOSH@PIXELFLAVOR.COM
@JOSHJ