Cross Plattform App Developement

Simon Groth

1x

Warum?

Cross Plattform App Devel

SAMSUNG bada

MeeGo™

iOS 4

Windows Mobile

RIM BlackBerry

palm webOS™

android

Motivation

- 74% Entwickeln oder Planen native Anwendung
  - 72% Android
  - 66% iOS
  - 33% BlackBerry
- 48% entwickeln die App für jedes BS einzeln
- 48% kennen keine X-Plattform Frameworks
- 38% fürchten wegen X-P. nicht für den App-Store zugelassen zu werden

Motivation

- 17% gaben an, dass ihre App nicht als Web-Anwendung umsetzbar wäre

- ca. 150 Befragte, 3% davon waren weniger als ein Jahr in der mobilen Entwicklung tätig.

http://www.slideshare.net/dvdh/umfrageergebnisse-crossplatform-entwicklung-mobiler-anwendungen

Wie?

Web-App

portierte Web-App

Native-App

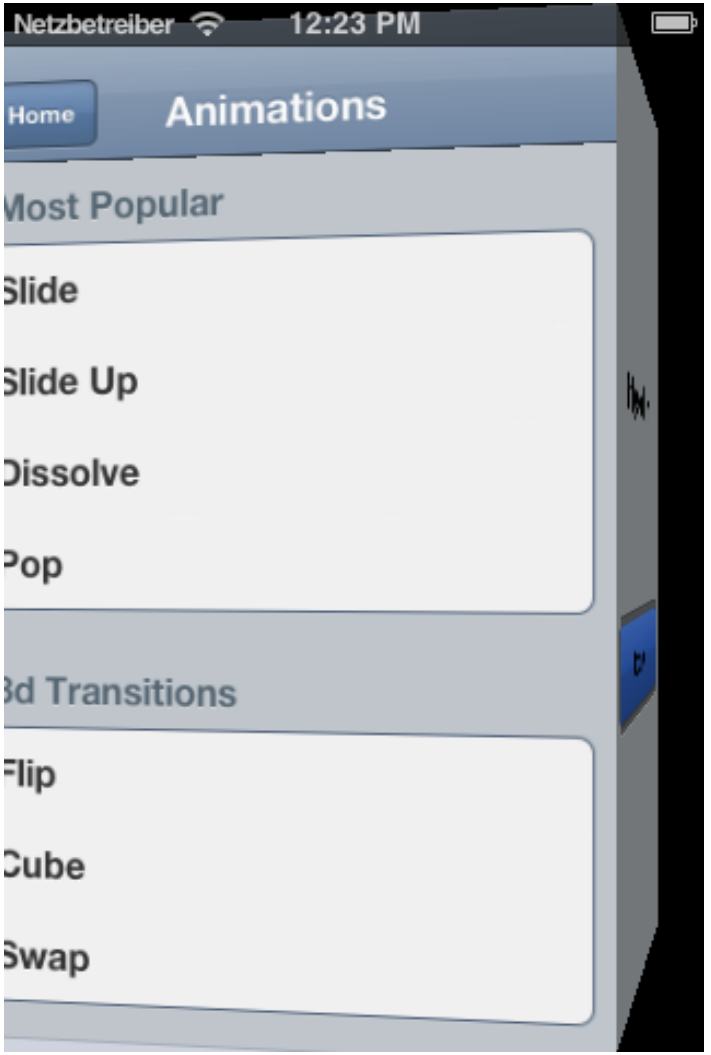
Web-App

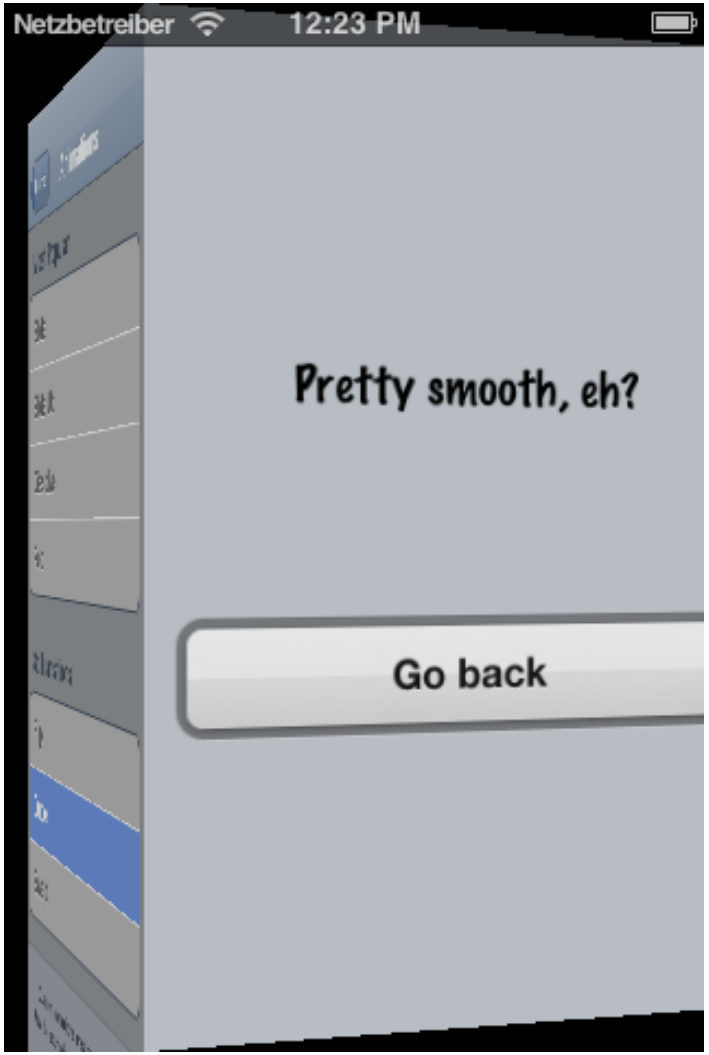
- Einmal entwickeln
  - HTML, CSS, JS
- Ohne App Store
  - Kaum Kosten
  - Kein Abweisen
  - Sofort Update
- Kein Zugriff das Gerät
  - Kamera, Sensoren, Kontakte...

Framework jQ-Touch

Framework jQ-Touch

Framework jQ-Touch





Web-App portieren mit PhoneGap

- Kostenlos
- Keine Anmeldung
- Fast alle Plattformen
- Zugriff auf Gerätefunktionen

Motivation

- 75% halten eine Bestmögliche User-Experience für wichtiger als eine einheitliche User-Experience

Native-App's?

Frameworks

MoSync

Appcelerator Titanium

- OpenSource-App
  - Free
- ClosedSource-App
  - 500$ pro Jahr
- Registrierung!

- Web-App
- Nativ mit JavaScript

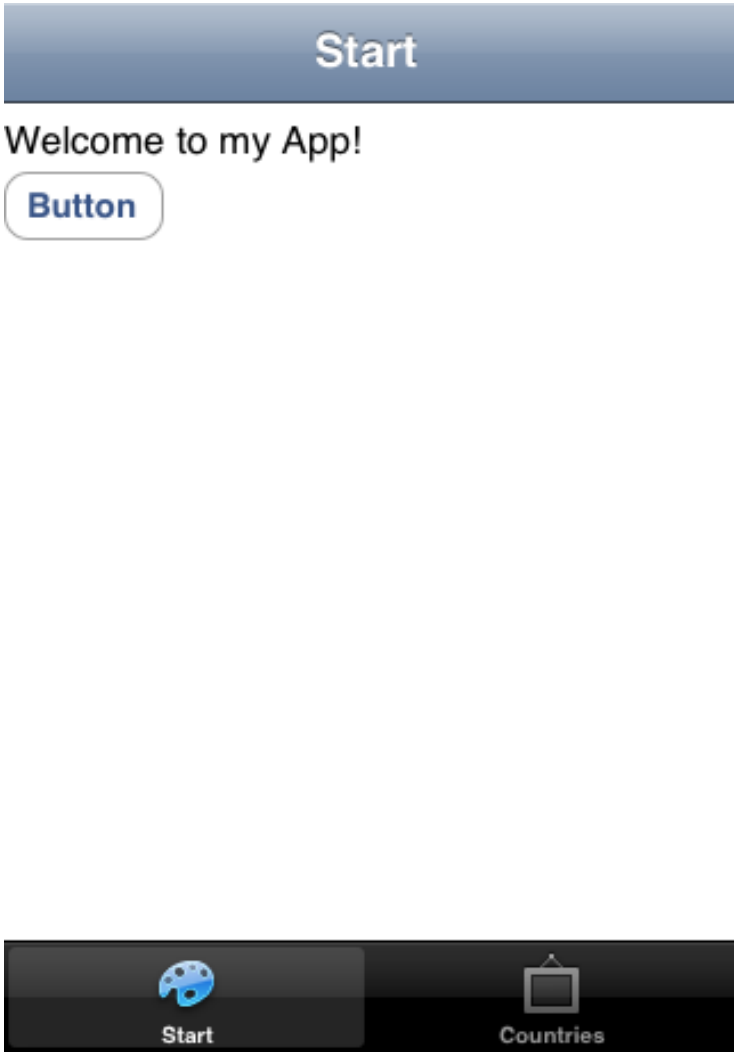
Appcelerator Titanium: **app.js**

```
var tabGroup = Titanium.UI.createTabGroup();
var win0 = Titanium.UI.createWindow({
    url: 'win0.js',    title:'Start'
    });
var tab0 = Titanium.UI.createTab({
    icon:'icon0.png',    title:'Start',    window:win0
    });
tabGroup.addTab(tab0);
// tab1 analog.
tabGroup.open();
```

Appcelerator Titanium: **win1.js**

```
var win = Titanium.UI.currentWindow;
var label = Ti.UI.createLabel({
    text:"Welcome to my App!"
    });
win.add(label);
var button = Ti.UI.createButton({
    title:"Button"
    });
win.add(button);
```

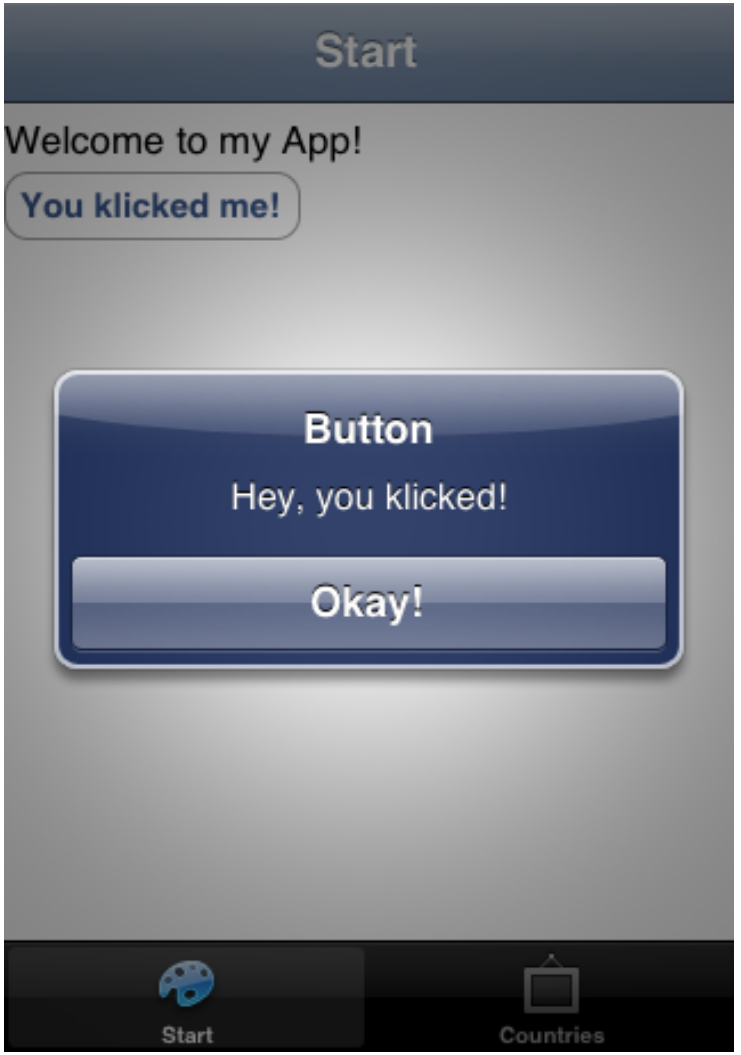
Appcelerator Titanium

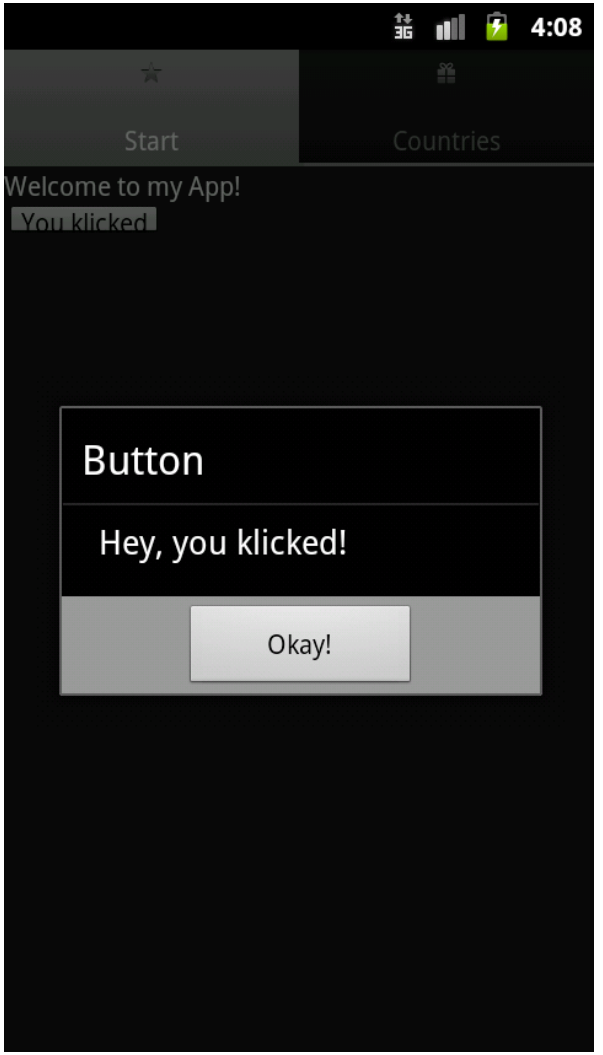


Appcelerator Titanium: win0.js

```
button.addEventListener('click', function(){
    button.title = 'You clicked me!';
    var a = Titanium.UI.createAlertDialog({
    title:'Button',
    message:'Hey, you klicked!'
    });
    a.show();
});
```

Appcelerator Titanium

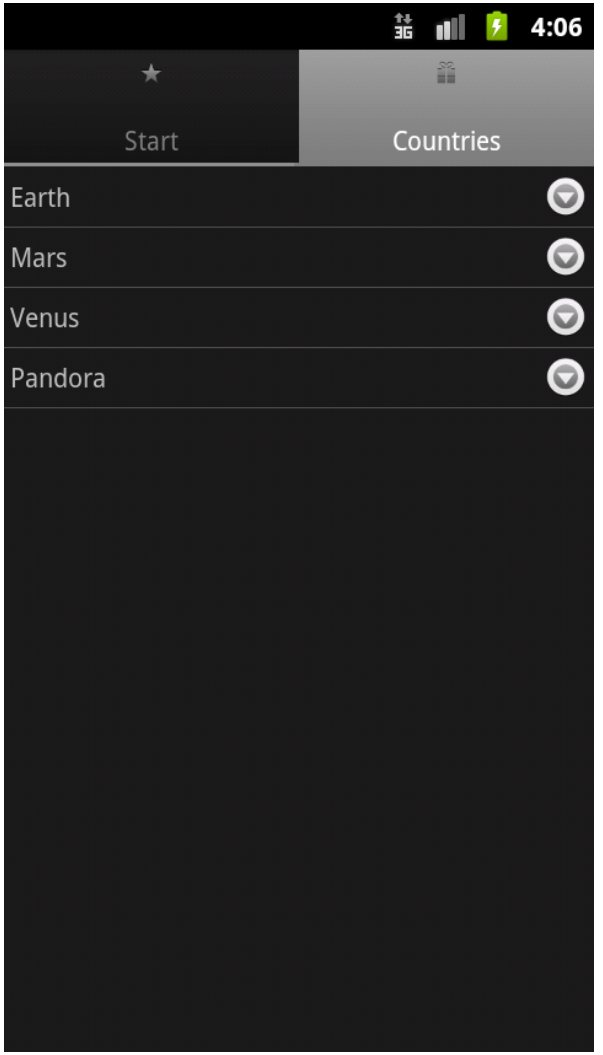




Appcelerator Titanium: **win1.js**

```
var data = [];
data.push(Ti.UI.createTableViewRow({
    title:'Earth'
    }));
    // Mars, Venus, Pandora
var tableView = Ti.UI.createTableView({
    style:Titanium.UI.iPhone.TableViewStyle.GROUPED,
    data:data
});
win.add(tableView);
```

Appcelerator Titanium



Appcelerator Titanium

Appcelerator Titanium

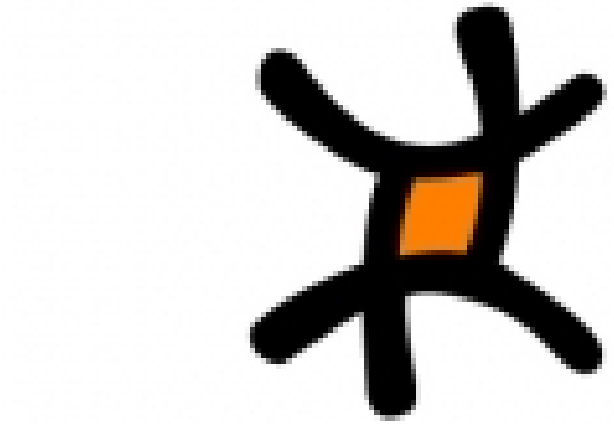
- Schnell Native UI-Komponenten
  - Android
  - iOS
  - BlackBerry
- Zugriff auf Gerätefunktionalität

Appcelerator Titanium

- Interpretiert JavaScript zur Laufzeit
  - Langsamer Aufbau von großen Tabellen
  - Zwischen 1s und 1.5s mit V8
  - Zwischen 2s und 3s in älteren Versionen
  - Nativ zwischen 8ms und 50ms

Mosync

- OpenSource-App
  - Free
- ClosedSource-App
  - 199$ pro Jahr
- Registrierung?!?
- Web-App mit C++ Anbindung (Wormhole)
- Nativ mit C++



MoSync

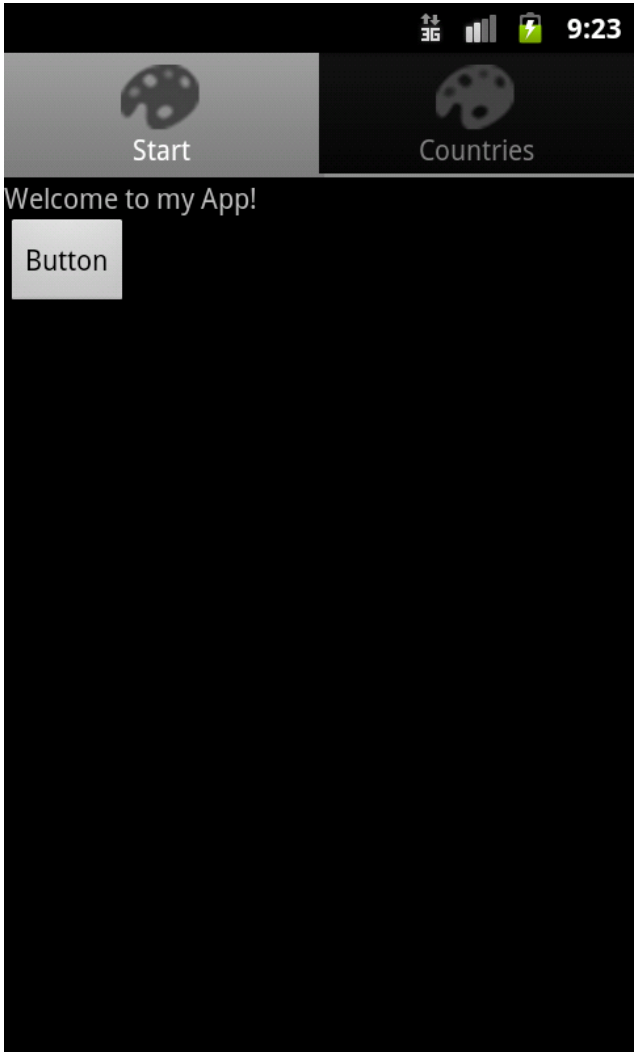
Mosync: Main.cpp

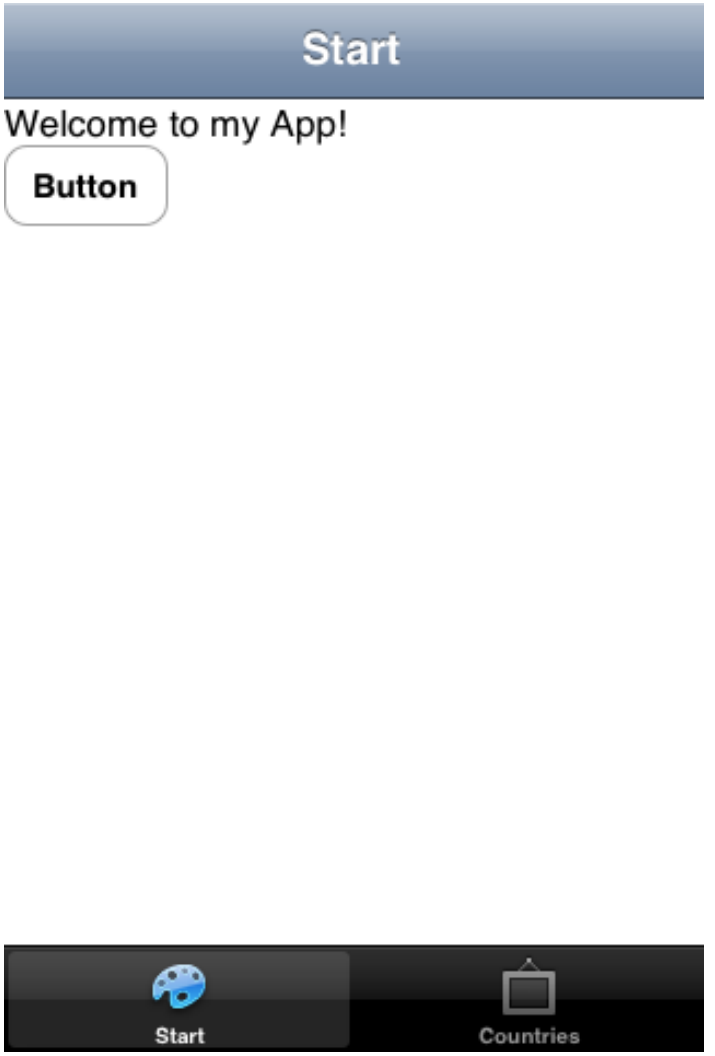
```
// includes, using namespace, syscall...
class ScreenMainWithTabs : public TabScreen {
ScreenMainWithTabs() : TabScreen() {
    mFirstScreen = new FirstScreen();
    mSecondScreen = new SecondScreen();
    // Add them as tabs.
    this->addTab(mFirstScreen);
    this->addTab(mSecondScreen);
}
// Destruktor, Variables...
}
```

Mosync: FirstScreen.cpp

```
// includes, using namespace, syscall...
class FirstScreen : public Screen {
FirstScreen() : Screen() {
	mMainLayout = new VerticalLayout();
	mButton = new Button();
	mButton->setText("Button");
	mMainLayout->addChild(mButton);
}
// Destruktor, Variables...
}
```

Mosync


9:23
Start
Countries
Welcome to my App!
Button


Start
Welcome to my App!
Button
Start
Countries

Mosync: FirstScreen.cpp

```
// includes, using namespace, syscall...
class FirstScreen : public Screen, public ButtonListener {
FirstScreen() : Screen() {
	//...
	mButton->addButtonListener(this);
}
void FirstScreen::buttonClicked(Widget* button) {
	maAlert("Button", "Hey, you clicked!", "Okay");
	mButton->setText("You clicked me!");
}
```

Mosync


9:24
Start
Countries
Welcome to my App!
You clicked me!
Button
Hey, you clicked!
Okay


Start
Welcome to my App!
You clicked me!
Button
Hey, you clicked!
Okay
Start
Countries

Mosync: SecondScreen.cpp

```
// includes, using namespace, syscall...
class SecondScreen : public Screen {
SecondScreen() : Screen() {
	mListView = new ListView();
	for (int i = 0; i < sizeOf(data); i++) {
		ListViewItem* listItem = new ListViewItem();
		listItem->setText(data[i].name);
		mListView->addChild(listItem);
	}
	this->setMainWidget(mListView);
}
```

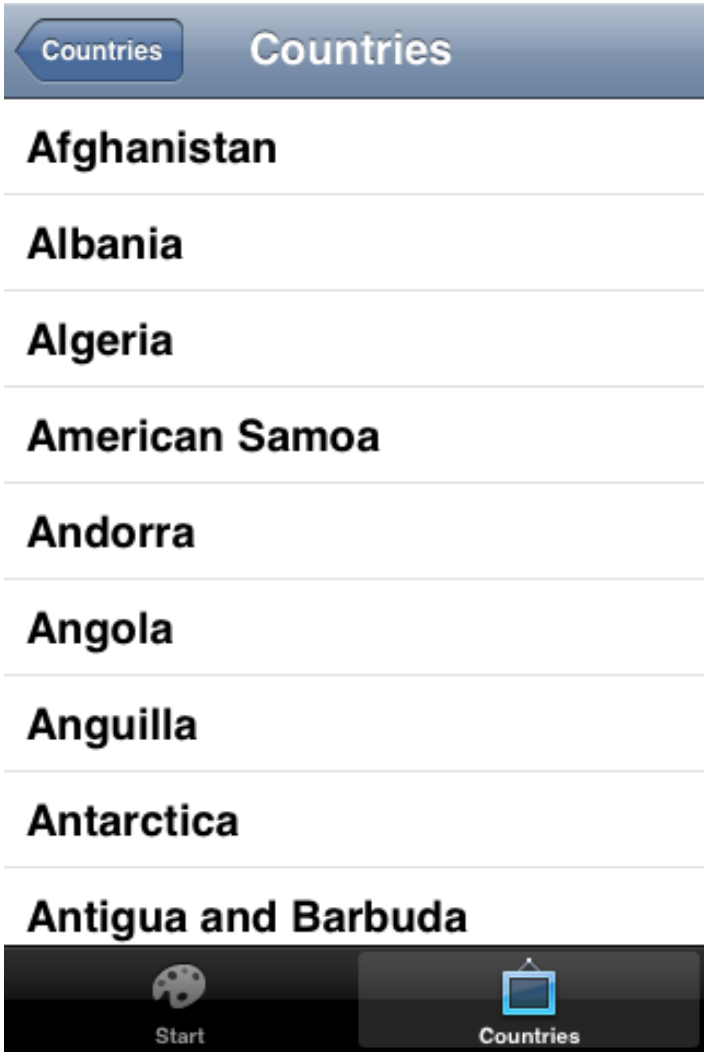
Mosync


9:24
Start
Countries
Earth
Mars
Venus
Pandora


Countries
Earth
Mars
Venus
Pandora
Start
Countries

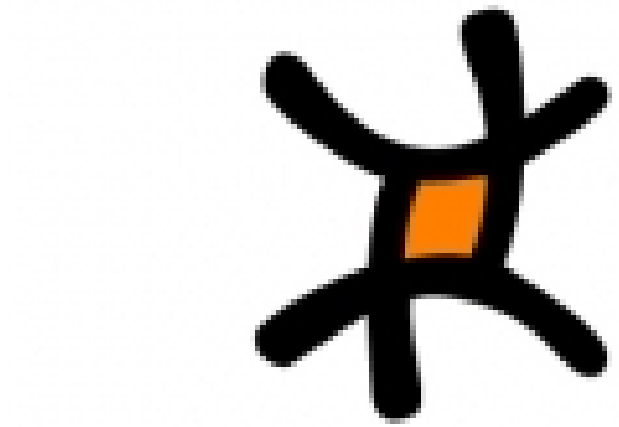
Mosync


9:24
Start
Countries
Afghanistan
Albania
Algeria
American Samoa
Andorra
Angola
Anguilla
Antarctica
Antigua and Barbuda
Argentina
Armenia
Aruba
Australia
Austria
Azerbaijan
Bahrain


Countries
Countries
Afghanistan
Albania
Algeria
American Samoa
Andorra
Angola
Anguilla
Antarctica
Antigua and Barbuda
Start
Countries

Mosync

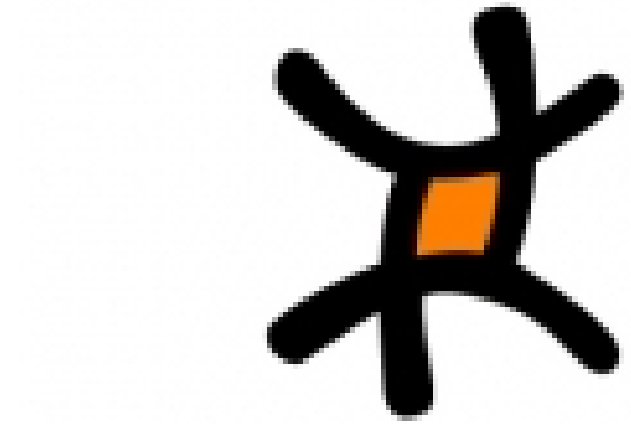
- Native UI mit C++
- Zugriff auf Gerätefunktionalität



MoSync

Mosync

- Mehr Gefrickel
- Mehr Freiheit
- Keine Garbage Collection
  - „Aus großer Kraft folgt große Verantwortung“
- Tabelle braucht 6s zum Öffnen



MoSync

Nachteile insgesamt

- Keine GUI-Builder (exkl. Web-App)
- Nicht für jede App verwendbar
- Nicht immer absolut auf das BS abstimmbar
  - oder viel Aufwand
- ClosedSource Kosten

Vorteile insgesamt

- (Fast) nur einmal Entwickeln
- Spart Zeit und Geld
- Sieht aus wie echt
- Native Extensions

Vortag insgesamt

- ist jetzt zu Ende.

Diskussion

- fängt jetzt an...

Danke für die Aufmerksamkeit!