React Native

스타일링, 배포, 네이티브 연동 등

스타일링

스타일링

- CSS와 유사한 형태의 표현식 사용
  (React Native Style !== CSS)
- 전처리기(Sass, Less, Stylus) 사용 불가
- 인라인 스타일처럼 오브젝트 리터럴로 적용 가능
- 퍼포먼스를 위해 StyleSheet.create 를 통해 사용할 것을 권장

스타일링 선언

```
const styles = StyleSheet.create({
    container: {
        borderRadius: 4,
        borderWidth: 0.5,
        borderColor: '#d6d7da',
    },
    title: {
        fontSize: 19,
        fontWeight: 'bold',
    },
    activeTitle: {
        color: 'red',
    },
});
```

스타일링 적용

```
<View style={styles.container}>
    <Text style={[styles.title, this.props.isActive && styles.activeTitle]} />
</View>
```

Flatten

```
const styles = StyleSheet.create({
    listItem: {
        flex: 1,
        fontSize: 16,
        color: 'white',
    },
    selectedListItem: {
        color: 'green',
    },
});

StyleSheet.flatten([styles.listItem, styles.selectedListItem]);
// returns { flex: 1, fontSize: 16, color: 'green' }
```

메모리 최적화 #1

```
import { StyleSheet } from "react-native";

const styles = StyleSheet.create({
    container: {
        borderRadius: 4,
        borderWidth: 0.5,
        borderColor: '#d6d7da',
    },
    hGroup: {
        flexDirection: 'row',
    },
});
// 자주 사용하는 스타일은 전역적으로 사용한다

export default styles;
```

메모리 최적화 #2

```
<View style={styles.container}>
    {/*권장*/}
    <Text style={StyleSheet.flatten( [ styles.listItem, styles.selectedListItem ] )}/>
    {/*권장되지 않음*/}
    <Text style={[ styles.listItem, styles.selectedListItem ]}/>
</View>
```

배포 준비

Expo 를 통한 배포

- eject 를 하지 않은 Expo 앱의 배포
- 글로벌 패키지 exp 를 사용
  - npm i -g exp
  - exp build:android
  - exp build:ios

필요한 것

- expo.io 계정 필요
- 구글 플레이 배포용 계정
- 애플 개발자 계정
- 오랜 빌드 시간을 참을 수 있는 인내심..

안드로이드 코드 사이닝

- Expo 대행
  - 인증서 파일과 키스토어를 Expo에서 생성하고 관리 대행
- 직접 관리
  - 직접 생성한 인증서 파일과 키스토어 파일을 Expo에 업로드

iOS 프로비저닝

- Expo 대행
  - 애플 개발자 계정을 입력하면 Expo 에서 팀 생성, 프로비저닝 파일 생성, 빌드 등을 대행함
- 직접 관리
  - Expo는 빌드만 수행하고 나머지 정보는 직접 관리

app.json

```
{
  "expo": {
    "sdkVersion": "27.0.0",
    "name": "Thanos Roulette",
    "version": "1.0.0",
    "slug": "thanos-roulette",
    "ios": {
      "bundleIdentifier": "com.grotesq.thanos.roulette"
    },
    "android": {
      "package": "com.grotesq.thanos.roulette"
    }
  }
}
```

네이티브 연동

네이티브 연동

- 모듈
  - 비 화면 표시 요소에 대한 네이티브 통신
- UI 모듈
  - 네이티브 화면 요소를 직접 출력

샘플 라이브러리 (공식)

```
npm install -g react-native-create-library
react-native-create-library MyLibrary
```

안드로이드 모듈

```java
public class ToastModule extends ReactContextBaseJavaModule {
  public ToastModule(ReactApplicationContext reactContext) {
    super(reactContext);
  }
}
```

@ReactMethod

```java
@ReactMethod
public void show(String message, int duration) {
  Toast.makeText(getReactApplicationContext(), message, duration).show();
}
```

모듈 등록

```
public class CustomToastPackage implements ReactPackage {

  @Override
  public List<NativeModule> createNativeModules(
                                  ReactApplicationContext reactContext) {
    List<NativeModule> modules = new ArrayList<>();

    modules.add(new ToastModule(reactContext));

    return modules;
  }

}
```

모듈 등록

```
// MainApplication.java
protected List<ReactPackage> getPackages() {
    return Arrays.<ReactPackage>asList(
            new MainReactPackage(),
            new CustomToastPackage());
}
```

native.js

```
import {NativeModules} from 'react-native';
module.exports = NativeModules.ToastExample;
```

React Native 에서의 호출

```
import ToastExample from './native';

ToastExample.show('Awesome');
```

안드로이드 UI 모듈

```
// ReactImageView 라는 안드로이드 클래스가 이미 존재하는 상황
public class ReactImageManager extends SimpleViewManager<ReactImageView> {
  public static final String REACT_CLASS = "RCTImageView";

  @Override
  public String getName() {
    return REACT_CLASS;
  }
```

모듈 등록

```
// 모듈 등록의 createNativeModules 와 유사
@Override
public List<ViewManager> createViewManagers(
                                ReactApplicationContext reactContext) {
  return Arrays.<ViewManager>asList(
    new ReactImageManager()
  );
}
```

React 컴포넌트 생성

```
// ImageView.js
import {requireNativeComponent} from 'react-native';

module.exports = requireNativeComponent('RCTImageView');
```

화면 배치

```
<View style={styles.container}>
    {/*
    import ImageView from './ImageView' 한 후
    */}
    <ImageView/>
</View>
```

iOS 모듈

```objc
// CalendarManager.h
#import <React/RCTBridgeModule.h>

@interface CalendarManager : NSObject <RCTBridgeModule>
@end
```

브릿지 모듈 추가

```
#import "CalendarManager.h"

@implementation CalendarManager

RCT_EXPORT_MODULE();

@end
```

메소드 추가

```objc
#import "CalendarManager.h"
#import <React/RCTLog.h>

@implementation CalendarManager

RCT_EXPORT_MODULE();

RCT_EXPORT_METHOD(addEvent:(NSString *)name location:(NSString *)location)
{
  RCTLogInfo(@"Pretending to create an event %@ at %@", name, location);
}
```

React Native 에서의 호출

```
import {NativeModules} from 'react-native';
const CalendarManager = NativeModules.CalendarManager;
CalendarManager.addEvent('Birthday Party', '4 Privet Drive, Surrey');
```

iOS UI 모듈

```
// RNTMapManager.m
#import <MapKit/MapKit.h>

#import <React/RCTViewManager.h>

@interface RNTMapManager : RCTViewManager
@end

@implementation RNTMapManager

RCT_EXPORT_MODULE()

- (UIView *)view
{
  return [[MKMapView alloc] init];
}

@end
```

React 컴포넌트 생성

```
// MapView.js
import { requireNativeComponent } from 'react-native';

module.exports = requireNativeComponent('RNTMap', null);
```

화면 배치

```
<View style={styles.container}>
    {/*
    import MapView from './MapView' 한 후
    */}
    <MapView />
</View>
```

인 앱 결제

인 앱 결제의 종류

- 소장형
  - 유료 광고 제거, 기간 무제한 아이템 구매 등
- 소비형
  - 게임 내 화폐 등
- 구독형
  - 정기 구독

사용 패키지

- react-native-iap 사용
  https://github.com/dooboolab/react-native-iap

초기화

```
try {
    await RNIap.initConnection();
    const products = await RNIap.getProducts(['item.id']);
    console.log( 'products', products );
} catch(err) {
    console.warn(err);
    Alert.alert( '결제 서버 통신 실패', error );
}
```

구매

```
// 구매
RNIap.buyProduct( data.item ).then(purchase => {
    console.log( 'purchase.transactionReceipt', purchase.transactionReceipt );

    // 소비
    RNIap.getAvailablePurchases().then( purchases => {
        purchases.forEach( async purchase => {
            await RNIap.consumePurchase(purchase.purchaseToken);
        });
    } );
}).catch(err => {
    console.warn(err); // standardized err.code and err.message available
    Alert.alert( '구매 실패', error );
})
```

코드푸시

(라이브 업데이트)

코드 푸시

```
import codePush from "react-native-code-push";

class App extends Component {
}

App = codePush(App);
```

데코레이터 사용시

```
import codePush from "react-native-code-push";

@codePush
class MyApp extends Component {
}
```

업데이트 확인 주기

```
const codePushOptions = { checkFrequency: codePush.CheckFrequency.ON_APP_RESUME };

class App extends Component {
}

App = codePush(codePushOptions)(App);
```

```
const codePushOptions = { checkFrequency: codePush.CheckFrequency.MANUAL };

class App extends Component {
}

App = codePush(codePushOptions)(App);
```

이미지 업데이트 범위

Component	Prop(s)
`Image`	`source`
`MapView.Marker` *(Requires react-native-maps `>=0.3.2`)*	`image`
`ProgressViewIOS`	`progressImage` , `trackImage`
`TabBarIOS.Item`	`icon` , `selectedIcon`
`ToolbarAndroid` *(React Native 0.21.0+)*	`actions[].icon` , `logo` , `overflowIcon`

업데이트 제약사항

Note: Any product changes which touch native code (e.g. modifying your AppDelegate.m/MainActivity.java file, adding a new plugin) cannot be distributed via CodePush, and therefore, must be updated via the appropriate store(s).

참고 *:* 네이티브 코드 *(*예 *: AppDelegate.m / MainActivity.java* 파일 수정*,* 새 플러그인 추가*)*를 변경한 제품 변경 사항은 *CodePush*를 통해 배포 할 수 없으므로 해당 스토어를 통해 업데이트해야합니다*.*