

React Native

React Native란?

- iOS와 안드로이드에서 동작하는 네이티브 모바일 앱을 만드는 자바 스크립트 프레임워크
- 리액트 네이티브는 페이스북의 자바스크립트 라이브러리인 리액트에 기반을 두고 있다.
- 자바스크립트 라이브러리를 이용해 겉모습과 실제 동작까지 진짜 네이티브인 모바일 앱을 만들 수 있다.

React Native 이점

- 리액트 네이티브는 코도바(cordova)나 아이오닉(Ionic)과 같은 기존의 크로스 플랫폼과는 다르게 대상 플랫폼의 표준 렌더링 API를 사용한다는 것이다.
- 기존의 방식은 html과 css를 사용하고, 웹뷰(webview)를 사용하였다.
- 기존의 방식은 성능이 떨어지고, 실제 플랫폼을 흉내 낸 UI를 사용하기 때문에 어색하다.
- 리액트 네이티브는 작성한 마크업을 플랫폼에 따라 그에 상응하는 진짜 네이티브 엘리먼트로 전환한다. 리액트는 메인 UI쓰레드와 분리되어 실행되기 때문에 앱의 역량을 줄이지 않아도 빠른 성능을 유지할 수 있다.
- 리액트 네이티브의 렌더링 갱신주기는 리액트와 같다. props나 state가 변경될 때 리액트 네이티브는 뷰를 다시 렌더링 한다.

React Native 단점

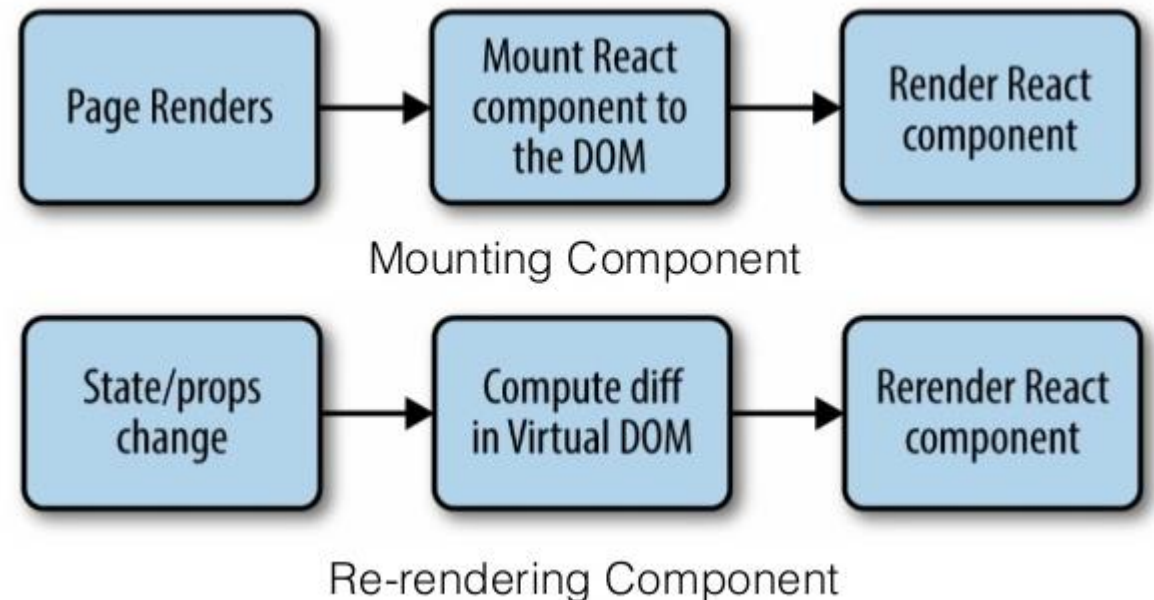
- 리액트 네이티브는 추가적인 레이어가 있기 때문에 디버깅이 간단하지 않다.
- 리액트와 대상 플랫폼 사이에서 발생하는 문제의 디버깅은 더 더욱 어렵다.
- 대상 플랫폼의 새로운 버전이 공개되었을 때 이를 지원하기 까지 시간이 걸린다.

React Native의 동작

- 리액트가 가상 DOM을 가지고 브라우저에 보여줄 수 있는 DOM으로 랜더링 하는 것처럼, React Native는 가상 DOM을 대상 플랫폼에 보여줄 수 있는 형태로 랜더링 한다.
- iOS의 경우 가상 DOM을 오브젝트 C API를 호출하여 iOS컴포넌트로 랜더링하고, 안드로이드의 경우 자바 API를 호출하여 안드로이드 컴포넌트로 랜더링한다.
- 가상 DOM이라는 추상화 레이어 덕분에 리액트 네이티브는 다른 플랫폼까지 지원할 수 있다.
- 리액트 네이티브로 윈도우와 우분투 데스크톱 앱을 만들 수 있게 해주는 오픈소스도 있다.

React Native 렌더링 라이프 싸이클

- 리액트 네이티브에서도 라이프사이클은 웹에서의 렌더링과 같지만 렌더링 절차는 조금 다르다. 그 이유는 리액트 네이티브는 브리지에 의존하기 때문이다.
- 브리지는 자바스크립트에서 발생하는 호출을 대상 플랫폼의 API와 UI엘리먼트에 연결한다. 리액트 네이티브는 메인 UI스레드에서 동작하지 않기 때문에 사용자 경험에 영향을 주지않고 비동기적으로 실행 될 수 있다.



create-react-native-app 과
Expo를 이용한 개발

개발환경:Create React Native App

- React Native App을 만들기 위한 도구인 create-react-native-app 을 생성한다.

```
npm install -g create-react-native-app
```

```
yarn global add create-react-native-app
```


첫번째 앱 만들기

- create-react-native-app을 이용하여 다음과 같이 앱을 생성한다. 아래의 명령을 실행하면 hello-project폴더가 생성된다.

```
create-react-native-app hello-project
```

- Expo CLI를 설치한다는 메시지가 나오면 y를 눌러준다.
- 템플릿을 선택 할 수 있는데 blank 페이지를 만들 것 인지, tabs로 만들 것 인지 메시지가 보여진다. 위아래 화살표를 이용해서 선택할 수 있다.
Hello-project에서는 blank페이지 생성을 선택한 후 엔터를 입력한다.

첫번째 앱 만들기

```
C:\wdevel\hybridapp_projects>create-react-native-app hello-project
'expo-cli'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.
This command requires Expo CLI.
Do you want to install it globally [Y/n]? y
Installing the package 'expo-cli'...
Does not seem like WSL enabled on this machine. Download a Linux distro from the Windows Store, run it at least once and
then make sure to run in an admin powershell:
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
Expo CLI installed. You can run `expo --help` for instructions.
? Choose a template: blank
[18:26:47] Downloading project files...
[18:26:49] Extracting project files...
[18:27:13] Customizing project...

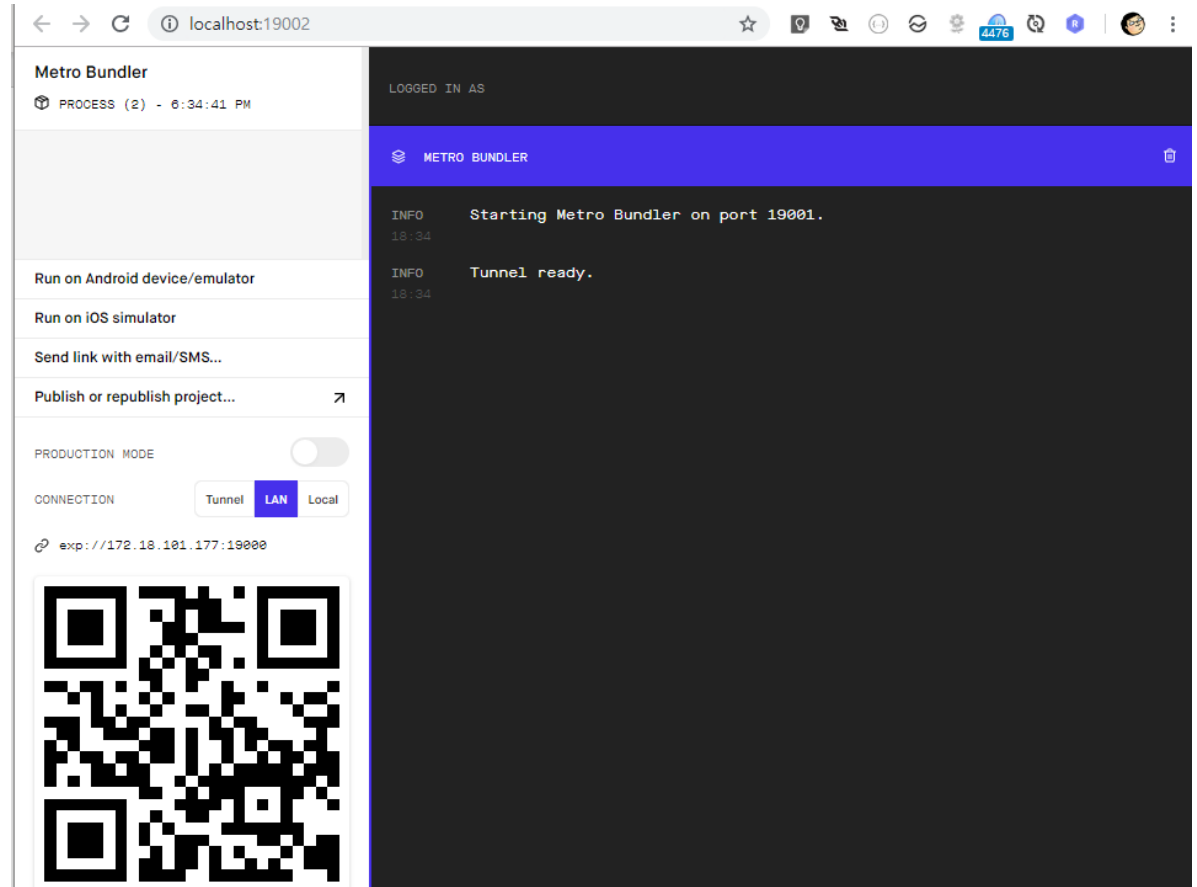
Your project is ready at C:\wdevel\hybridapp_projects\hello-project
To get started, you can type:

  cd hello-project
  expo start
```

첫번째 앱 만들기

cd hello-project

expo start



첫번째 앱 만들기

- 안드로이드, 아이폰 앱 스토어에서 Expo앱을 다운로드 받는다.
- QR코드를 Expo앱으로 스캔한다.
- * 컴퓨터와 스마트폰은 같은 네트워크를 사용하고 있어야한다.

첫번째 앱 만들기

```
App.js
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 export default class App extends React.Component {
5   render() {
6     return (
7       <View style={styles.container}>
8         <Text>Hello World!</Text>
9       </View>
10    );
11  }
12 }
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#fff',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21 });
22
```

SKT 10% 오후 6:44

Hello World!

개발환경설치 - 전형적인 개발

리액티 네이티브를 개발하기 위해 필요한 도구

- node.js
- 리액트 네이티브
- iOS 개발 환경(XCode)
- 안드로이드 개발 환경(JDK, 안드로이드 SDK, 안드로이드 스튜디오)

리액트 네이티브 개발을 위한 환경설정

- 환경변수 ANDROID_HOME 을 지정

Windows

C:\Users\계정명\AppData\Local\Android\Sdk

macOS

/Users/계정명/Library/Android/sdk

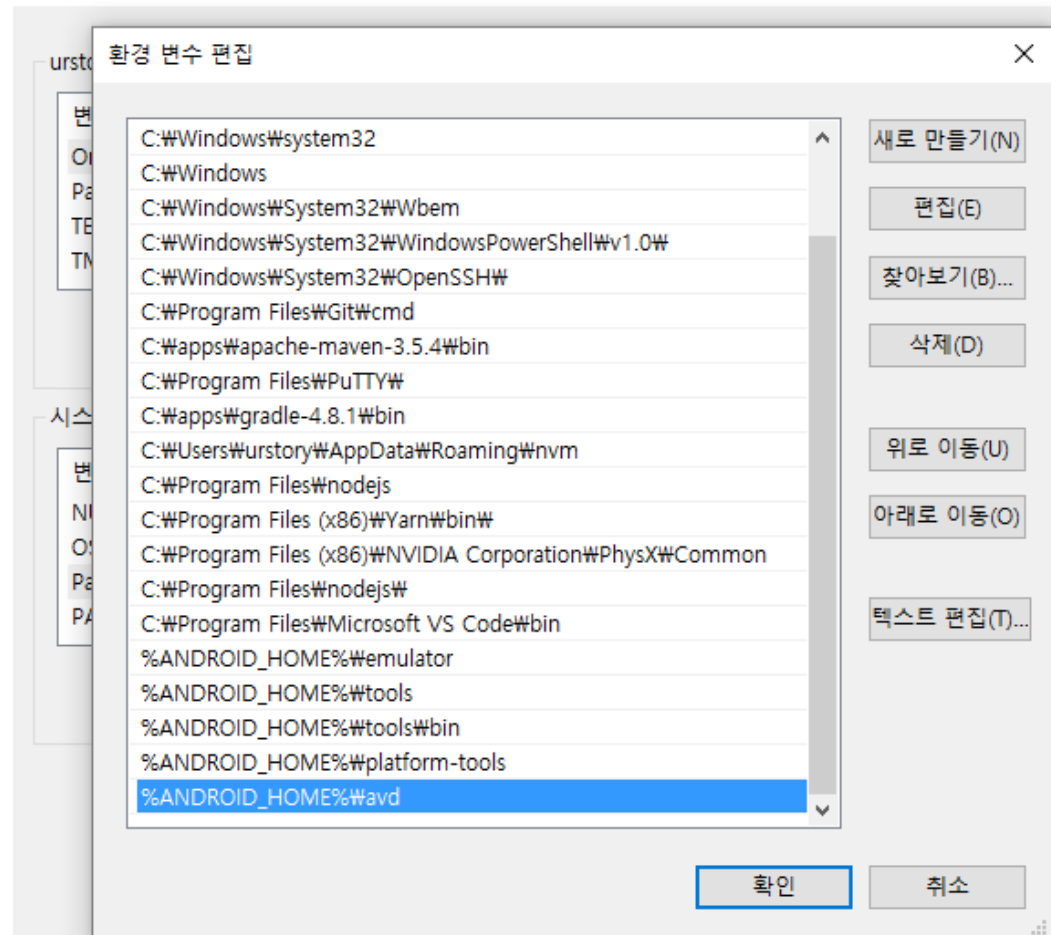
Linux

home/계정명/Android/Sdk

리액트 네이티브 개발을 위한 환경 설정

```
export
ANDROID_HOME=$HOME/Library/Android/sdk
export PATH=$PATH:$ANDROID_HOME/emulator
export PATH=$PATH:$ANDROID_HOME/tools
export PATH=$PATH:$ANDROID_HOME/tools/bin
export PATH=$PATH:$ANDROID_HOME/platform-
tools
export PATH=$PATH:$ANDROID_HOME/avd
```

환경 변수



react-native 명령어를 이용한 앱 생성하기

- react-native CLI 도구 설치

```
npm install -g react-native-cli
```

```
yarn global add react-native-cli
```

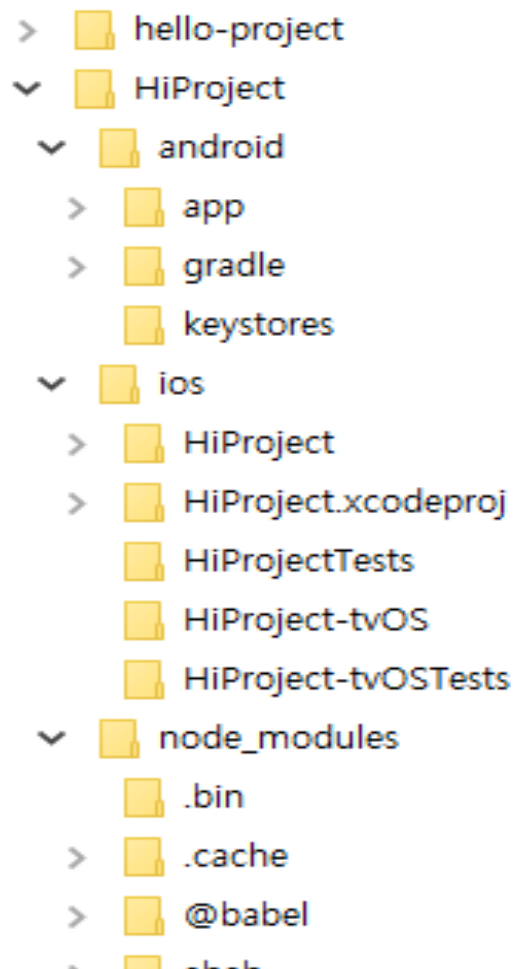
- react-native를 이용한 프로젝트 생성

```
react-native init HiProject
```

- react-native 앱 실행

```
cd HiProject
```

```
react-native run-android
```



오류

- react-native run-android 을 실행 했는데 오류가 발생했다는 것은 에뮬레이터가 실행되지 않았다는 것을 의미한다.

```
* Get more help at https://help.gradle.org
```

```
BUILD FAILED in 4s
```

```
Could not install the app on the device, read the error above for details.
```

```
Make sure you have an Android emulator running or a device connected and have  
set up your Android development environment:
```

```
https://facebook.github.io/react-native/docs/getting-started.html
```

오류

- avd 의 목록을 살펴본다.

C:\Users\사용자 계정

\AppData\Local\Android\Sdk\tools\bin> avdmanager.bat
list avd

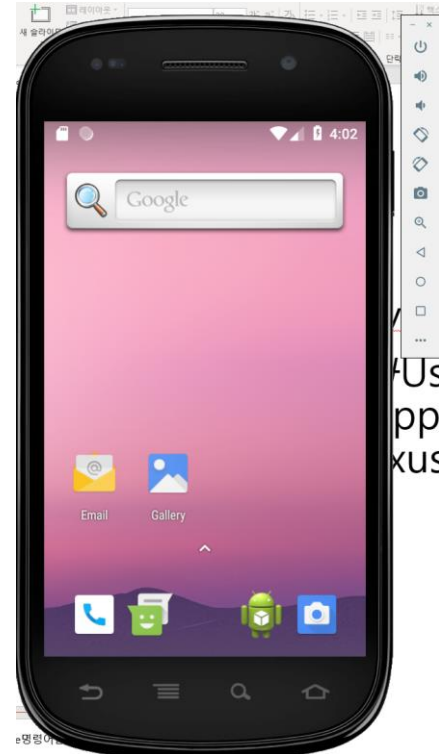
```
C:\Users\urstory\AppData\Local\Android\Sdk\tools\bin>avdmanager.bat list avd
Parsing C:\Users\urstory\AppData\Local\Android\Sdk\build-tools\27.0.3\package.xmlParsing C:\Users\urstory\AppData\Local
\Android\Sdk\build-tools\28.0.3\package.xmlParsing C:\Users\urstory\AppData\Local\Android\Sdk\emulator\package.xmlParsi
ng C:\Users\urstory\AppData\Local\Android\Sdk\extras\android\m2repository\package.xmlParsing C:\Users\urstory\AppData\L
ocal\Android\Sdk\extras\google\m2repository\package.xmlParsing C:\Users\urstory\AppData\Local\Android\Sdk\extras\intel\H
ardware_Accelerated_Execution_Manager\package.xmlParsing C:\Users\urstory\AppData\Local\Android\Sdk\extras\m2repositor
y\com\android\support\constraint\constraint-layout-solver\1.0.2\package.xmlParsing C:\Users\urstory\AppData\Local\Andro
id\Sdk\extras\m2repository\com\android\support\constraint\constraint-layout\1.0.2\package.xmlParsing C:\Users\urstory\A
ppData\Local\Android\Sdk\patcher\v4\package.xmlParsing C:\Users\urstory\AppData\Local\Android\Sdk\platform-tools\packag
e.xmlParsing C:\Users\urstory\AppData\Local\Android\Sdk\platforms\android-22\package.xmlParsing C:\Users\urstory\AppData
\Local\Android\Sdk\platforms\android-27\package.xmlParsing C:\Users\urstory\AppData\Local\Android\Sdk\platforms\androi
d-28\package.xmlParsing C:\Users\urstory\AppData\Local\Android\Sdk\sources\android-27\package.xmlParsing C:\Users\ursto
ry\AppData\Local\Android\Sdk\sources\android-28\package.xmlParsing C:\Users\urstory\AppData\Local\Android\Sdk\system-im
ages\android-22\google_apis\x86\package.xmlParsing C:\Users\urstory\AppData\Local\Android\Sdk\system-images\android-27\
default\x86_64\package.xmlParsing C:\Users\urstory\AppData\Local\Android\Sdk\system-images\android-27\google_apis_pla
ystore\x86\package.xmlParsing C:\Users\urstory\AppData\Local\Android\Sdk\tools\package.xmlAvailable Android Virtual Devic
es:
  Name: Nexus_S_API_27
  Device: Nexus S (Google)
  Path: C:\Users\urstory\AppData\Local\Android\Sdk\avd\Nexus_S_API_27.avd
  Target: Default Android System Image
    Based on: Android API 27 Tag/ABI: default/x86_64
  Skin: nexus_s
  Sdcard: 512M
C:\Users\urstory\AppData\Local\Android\Sdk\tools\bin>
```

오류

- avd 를 다음과 같이 실행한다.

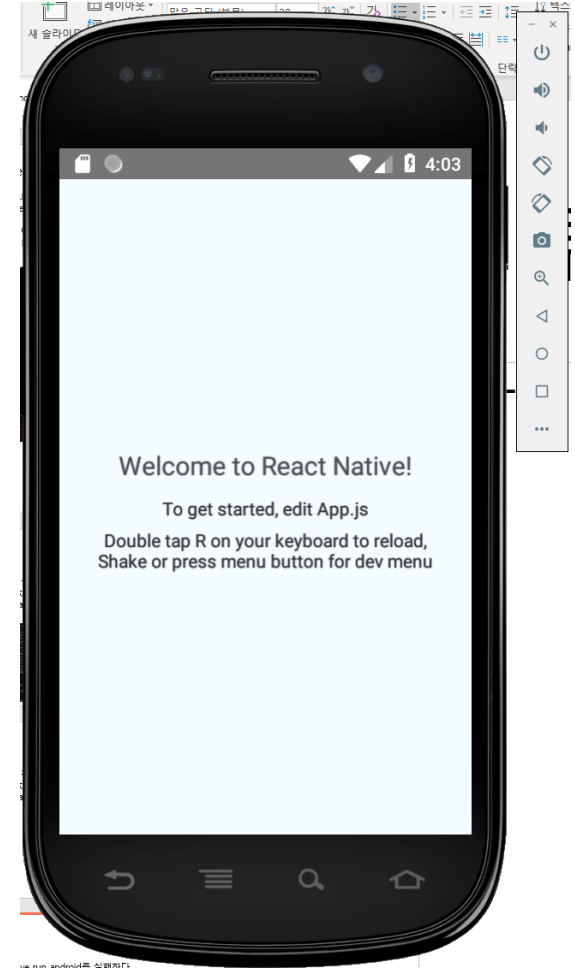
C:\Users\사용자계정

\AppData\Local\Android\Sdk\tools>emulator.exe -avd
Nexus_S_API_27



오류

- 다시 react-native run-android를 실행한다.



오류

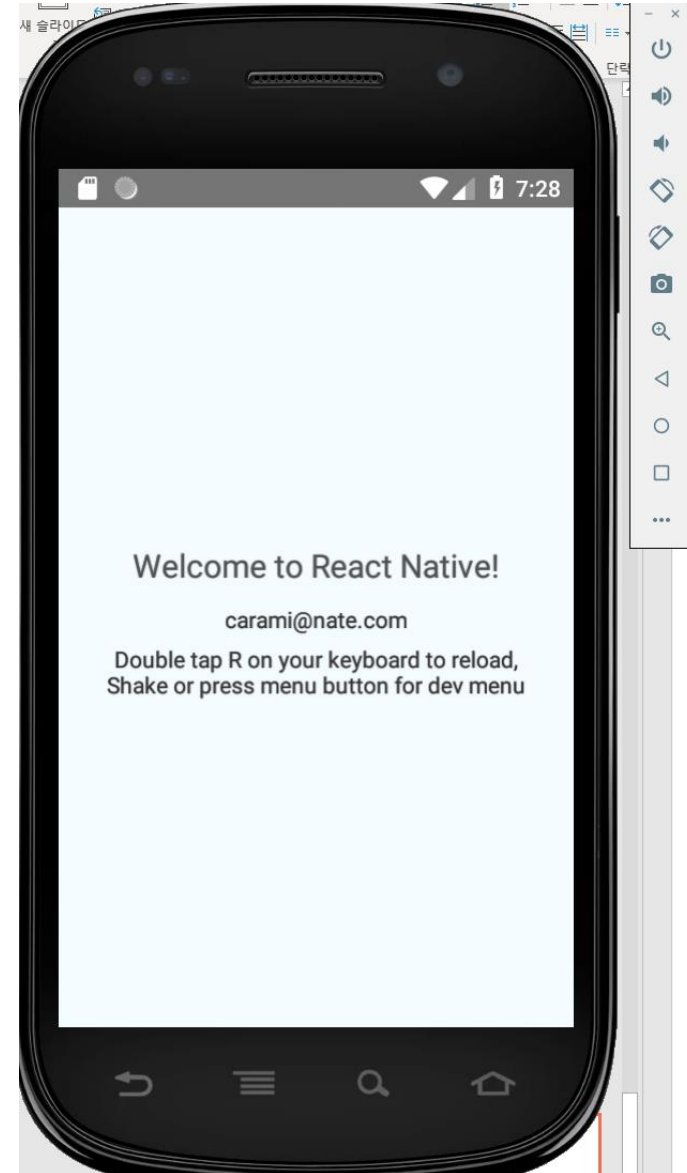
- ADB 안드로이드 에뮬레이터 응답이 없을 경우

adb kill-server

adb start-server

App.js 수정하기

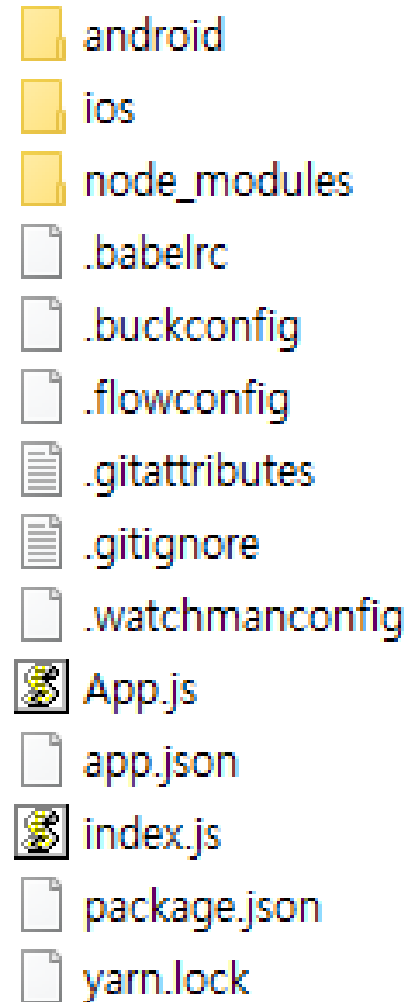
- 안드로이드의 경우 키보드의 R키를 더블클릭하면 App.js의 변경된 내용이 바로 반영된다.



Sample 코드 살펴보기

디렉토리 구조

- android : 안드로이드 프로젝트
- ios : ios프로젝트



android
ios
node_modules
.babelrc
.buckconfig
.flowconfig
.gitattributes
.gitignore
.watchmanconfig
App.js
app.json
index.js
package.json
yarn.lock

A vertical list of files and directories. The first three items (android, ios, node_modules) are preceded by yellow folder icons. The remaining items are preceded by white file icons with a folded top-right corner. The file icons for .babelrc, .buckconfig, .flowconfig, .gitattributes, .gitignore, .watchmanconfig, app.json, package.json, and yarn.lock are standard white. The file icons for App.js and index.js feature a small yellow and black robot icon on the left side.

index.js

```
/** @format */  
  
import {AppRegistry} from 'react-native';  
import App from './App';  
import {name as appName} from './app.json';  
  
AppRegistry.registerComponent(appName, () => App);
```

- 예전 버전에는 index.ios.js와 index.android.js 2가지로 구성이 되어 있었는데, 리액트 네이티브가 버전업이 되면서 파일이 하나로 바뀌었다.

App.js 1/2

- import {Platform, StyleSheet, Text, View} from 'react-native';

네이티브에서 사용하는 모듈을 명시적으로 import해야한다.

- Platform.select()는 플랫폼에 따라 다른 문자열이 사용되게 도와준다.
- 기존 리액트 프로그래밍과 다른점은 div대신에 View를 사용하는 것.
style을 사용하는 것

```
import React, {Component} from 'react';
import {Platform, StyleSheet, Text, View} from 'react-native';

const instructions = Platform.select({
  ios: 'Press Cmd+R to reload,\n' + 'Cmd+D or shake for dev menu',
  android:
    'Double tap R on your keyboard to reload,\n' +
    'Shake or press menu button for dev menu',
});

type Props = {};
export default class App extends Component<Props> {
  render() {
    return (
      <View style={styles.container}>
        <Text style={styles.welcome}>Welcome to React Native!</Text>
        <Text style={styles.instructions}>carami@nate.com</Text>
        <Text style={styles.instructions}>{instructions}</Text>
      </View>
    );
  }
}
```

App.js 2/2

- StyleSheet.create()
는 스타일과 관련된
정보를 생성한다.

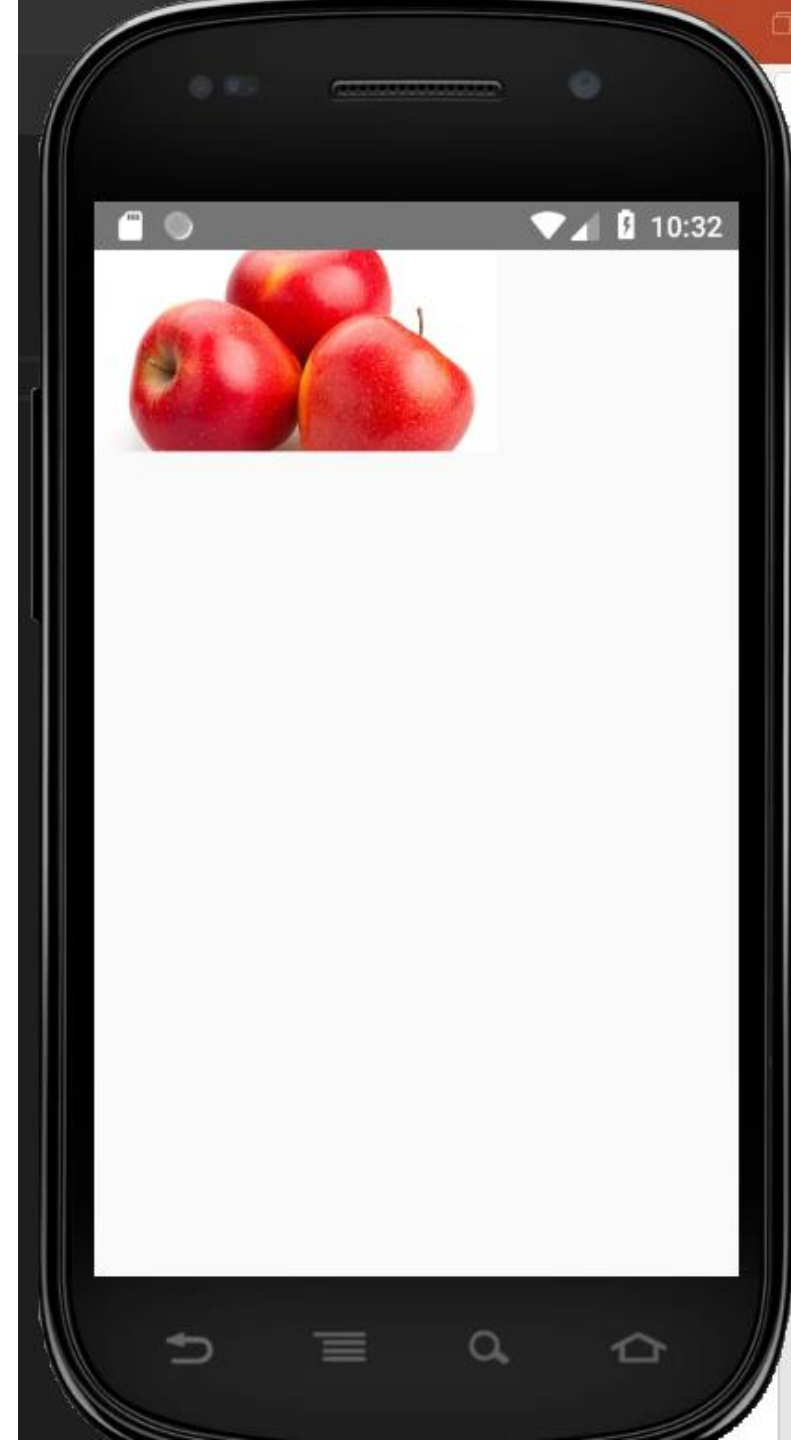
```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#F5FCFF',
  },
  welcome: {
    fontSize: 20,
    textAlign: 'center',
    margin: 10,
  },
  instructions: {
    textAlign: 'center',
    color: '#333333',
    marginBottom: 5,
  },
});
```

사용해 보기

이미지 보여주기

```
import React, {Component} from 'react';
import {Image} from 'react-native';

export default class App extends Component {
  render() {
    const img = {
      uri : '이미지URI'
    };
    return (
      <Image source={img} style={{width: 200, height:100}} />
    );
  }
}
```



props

```
1  import React, {Component} from 'react';
2  import {Text, View} from 'react-native';
3
4  class Hello extends Component{
5    render(){
6      return (
7        <View style={{alignItems : 'center'}}>
8          <Text>Hello {this.props.name}!</Text>
9        </View>
10      );
11    }
12  }
13
14  export default class SayHello extends Component {
15    render() {
16      return (
17        <View style={{alignItems: 'center'}}>
18          <Hello name='카라미' />
19          <Hello name='봄봄이' />
20          <Hello name='뽀뽀' />
21        </View>
22      );
23    }
24  }
```



state

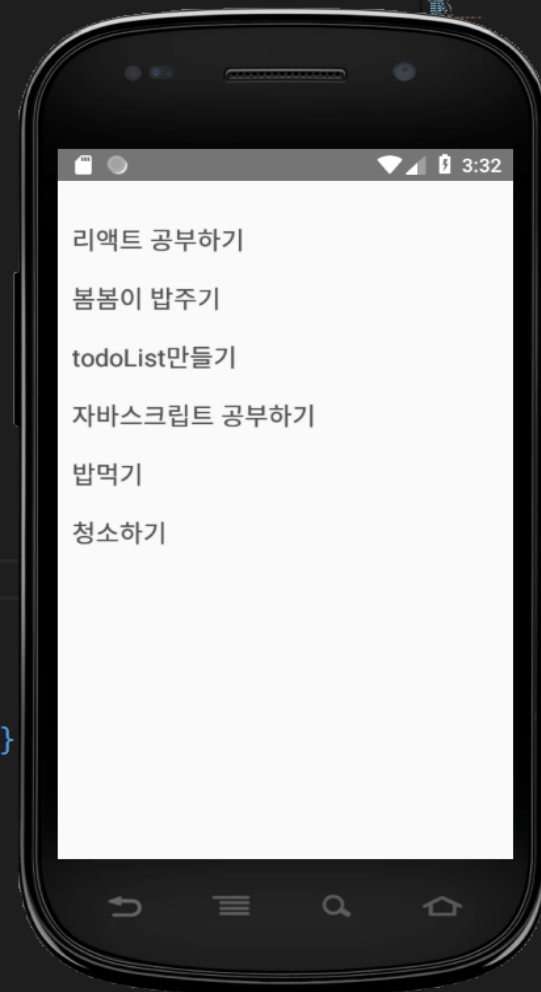
```
1  import React, {Component} from 'react';
2  import {View, Text} from 'react-native';
3
4  class StateExam extends Component{
5      state = {
6          name : 'kang'
7      }
8      render(){
9          return (
10             <View >
11                 <Text>Hello {this.state.name}!</Text>
12             </View>
13          );
14      }
15  }
16  export default StateExam;
```

scrollView

```
<ScrollView>  
  <Image  
    source={{uri: '이미지주소'}}  
    style={{width: 300, height:180}}  
  />  
  <Text>  
    긴문자열  
  </Text>  
</ScrollView>
```

List Views

```
1 import React from 'react';
2 import {FlatList,SectionList, StyleSheet, Text, View} from 'react-native';
3
4 export default class FlatListBasic extends React.Component{
5   render(){
6     const todos = [
7       {todo:'리액트 공부하기'},
8       {todo:'봄봄이 밥주기'},
9       {todo:'todoList만들기'},
10      {todo:'자바스크립트 공부하기'},
11      {todo:'밥먹기'},
12      {todo:'청소하기'}
13    ];
14    return (
15      <View style={styles.container}>
16        <FlatList
17          data={todos}
18          renderItem={({item})=>
19            <Text style={styles.item}>{item.todo}</Text>
20          }
21          keyExtractor = {(item, index) => item.toString()}
22        />
23      </View>
24    );
25  }
26 }
27
28 const styles = StyleSheet.create({
```



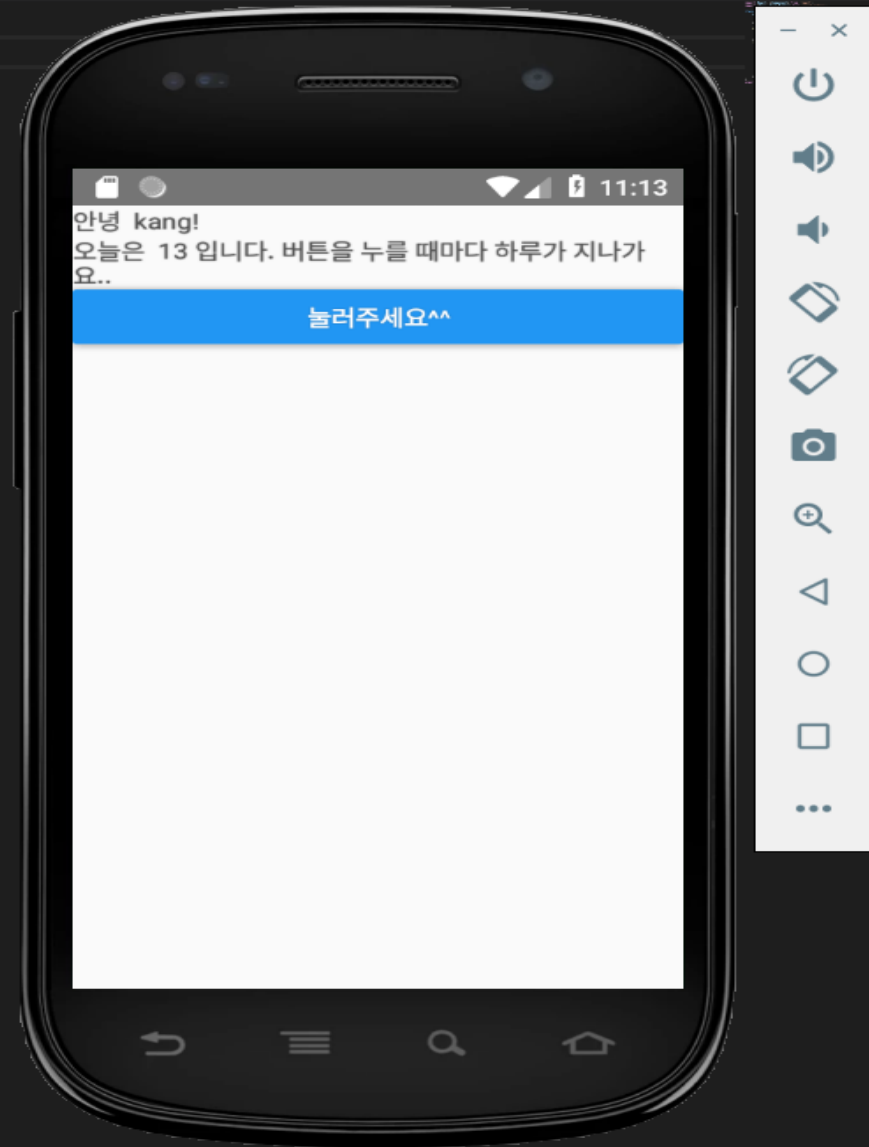
SelectionList

```
<SectionList
sections=[
  {title: 'Title1', data: ['item1', 'item2']},
  {title: 'Title2', data: ['item3', 'item4']},
  {title: 'Title3', data: ['item5', 'item6']},
]

renderItem=(({item, index, section}) => <Text>{item}</Text>
  renderSectionHeader=(({section: {title}}) => (
    <Text style={{fontWeight: 'bold'}}>{title}</Text>
  ))
keyExtractor={(item, index) => item + index}
/>
```

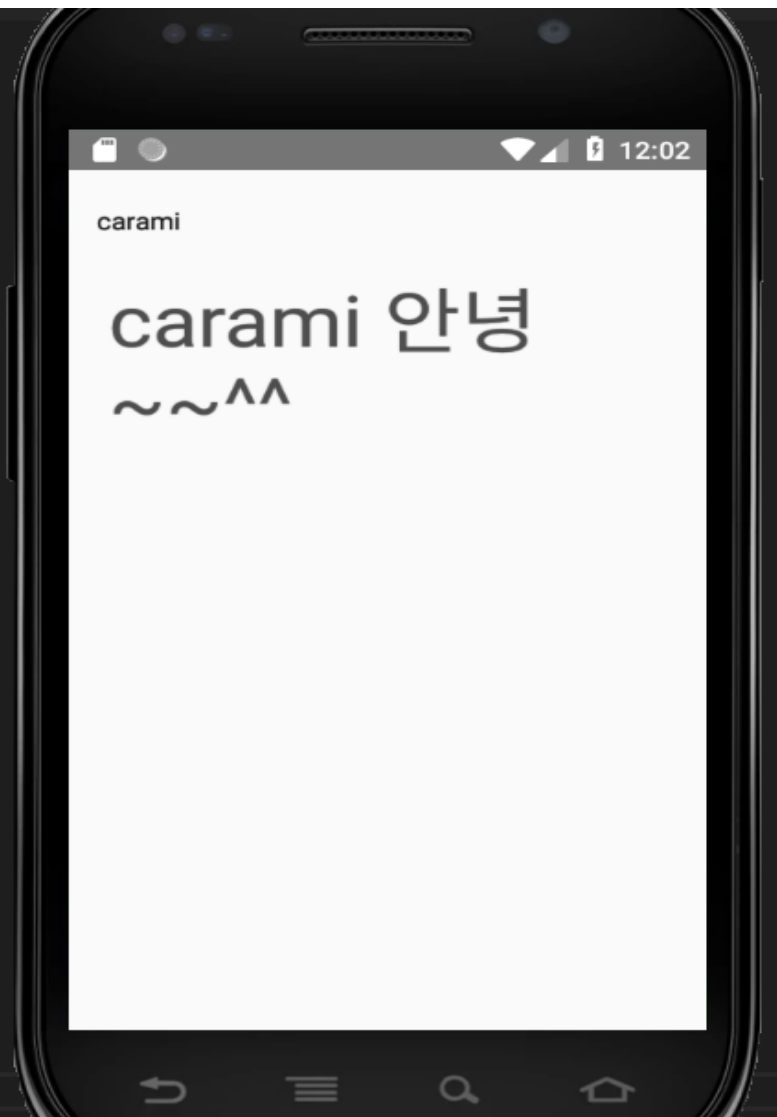
onPress

```
1 import React, {Component} from 'react';
2 import {View, Text, Button} from 'react-native';
3
4 class StateExam extends Component{
5   state = {
6     name : 'kang',
7     date : 13
8   }
9
10  onPressButton={() => {
11    this.setState({
12      date : this.state.date + 1
13    })
14  }}
15  render(){
16    return (
17      <View >
18        <Text>안녕 {this.state.name}!</Text>
19        <Text>오늘은 {this.state.date} 입니다.
20        버튼을 누를 때마다 하루가 지나가요..</Text>
21        <Button
22          onPress={this.onPressButton}
23          title="눌러주세요^^"
24        />
25      </View>
26    );
27  }
28 }
29 export default StateExam;
```



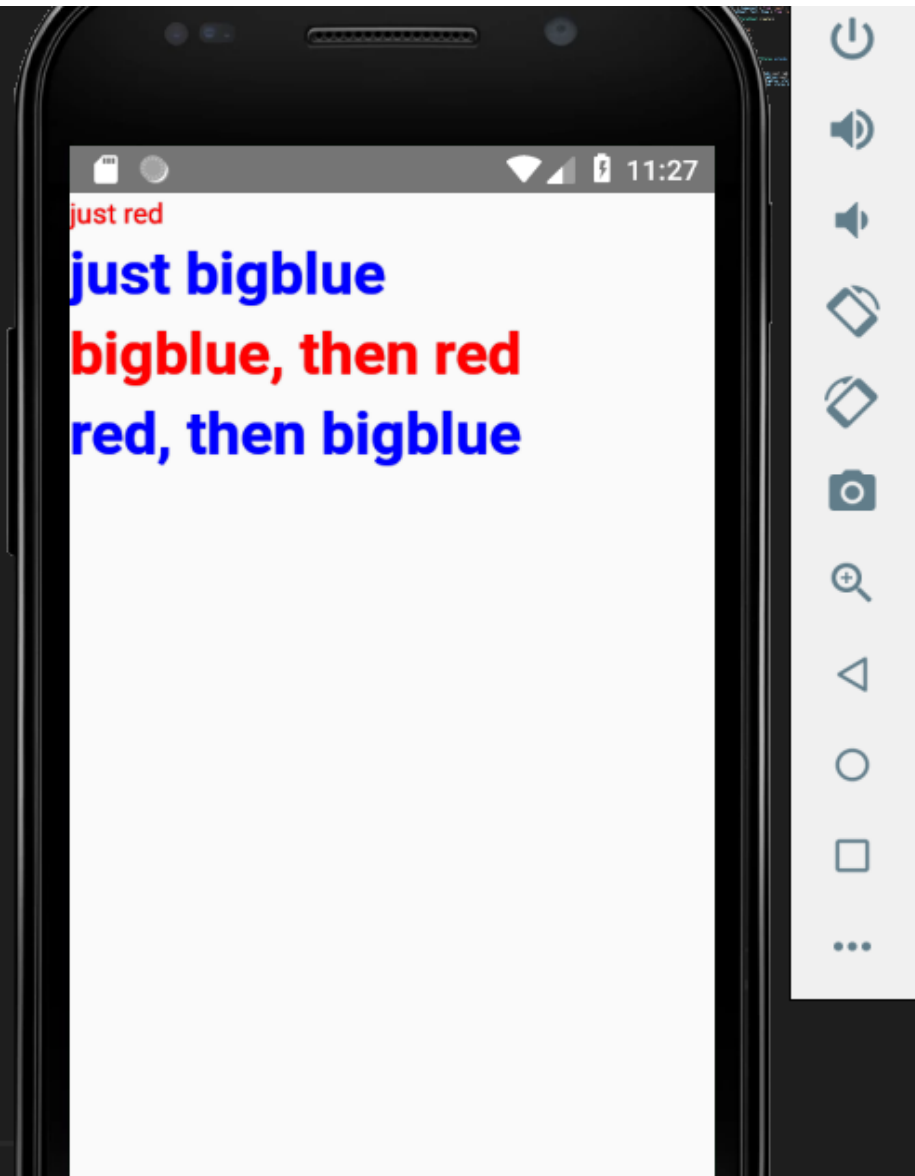
onChangeText

```
1  import React, { Component } from 'react';
2  import { Text, TextInput, View } from 'react-native';
3
4  export default class EventExam extends Component {
5    state = {
6      text : ''
7    }
8    onChange = (name) =>{
9      this.setState({
10        text : name
11      });
12    }
13    render() {
14      return (
15        <View style={{padding: 10}}>
16          <TextInput
17            style={{height: 40}}
18            placeholder="이름을 입력하세요."
19            onChangeText={(name) => this.onChange(name)}
20          />
21          <Text style={{padding: 10, fontSize: 42}}>
22            {this.state.text} 안녕~~^^
23          </Text>
24        </View>
25      );
26    }
27  }
28
```



스타일 적용하기

```
1 import React, { Component } from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const styles = StyleSheet.create({
5   bigblue: {
6     color: 'blue',
7     fontWeight: 'bold',
8     fontSize: 30,
9   },
10  red: {
11    color: 'red',
12  },
13 });
14
15 export default class LotsOfStyles extends Component {
16   render() {
17     return (
18       <View>
19         <Text style={styles.red}>just red</Text>
20         <Text style={styles.bigblue}>just bigblue</Text>
21         <Text style={[styles.bigblue, styles.red]}>bigblue, then red</Text>
22         <Text style={[styles.red, styles.bigblue]}>red, then bigblue</Text>
23       </View>
24     );
25   }
26 }
```



Flex 스타일 적용하기

```
<View style={{flex: 1}}>
  <View style={{flex: 1}}>
    <Text style={styles.red}>just red</Text>
    <Text style={styles.bigblue}>just bigblue</Text>
    <Text style={[styles.bigblue, styles.red]}>bigblue, then red</Text>
    <Text style={[styles.red, styles.bigblue]}>red, then bigblue</Text>
  </View>
  <View style={{flex: 1}}>
    <View style={{flex: 1, backgroundColor: 'skyblue'}} />
    <View style={{flex: 2, backgroundColor: 'blue'}} />
    <View style={{flex: 3, backgroundColor: 'darkblue'}} />
  </View>
</View>
```



ajax 이용한 비동기 통신

```
1  import React,{Component} from 'react';
2  import { Alert,ActivityIndicator,FlatList, Text, View} from 'react-native';
3
4  export default class FetchExam extends Component{
5    state = {
6      isLoading : true,
7      todos : []
8    }
9    componentDidMount(){
10     var request = new XMLHttpRequest();
11     request.onreadystatechange = (e) => {
12       if (request.readyState !== 4) {
13         return;
14       }
15       if (request.status === 200) {
16         Alert.alert('success',request.responseText);
17         this.setState({
18           isLoading: false,
19           todos: JSON.parse(request.responseText)
20         });
21         console.log('success', request.responseText);
22       } else {
23         Alert.alert('error');
24         console.warn('error');
25       }
26     };
27     request.open('GET', 'http://192.168.0.164:8080/todos/');
28     request.send();
29  }
```

```
30
31  render(){
32    if(this.state.isLoading){
33      return(
34        <View style={{flex:1,padding:20}}>
35          <ActivityIndicator/>
36        </View>
37      )
38    }
39    return(
40      <View style={{flex:1,paddingTop:20}}>
41        <FlatList
42          data={this.state.todos}
43          renderItem = { ({item})=><Text>{item.todo}</Text>}
44          keyExtractor={({id},index)=> id}
45        />
46      </View>
47    )
48  }
49 }
```