

ToDo

프로젝트 기본 설정

❖ 프로젝트 생성 및 기본 설정 및 실행

- ✓ 프로젝트 생성: npx react-native ReactNativeToDo
- ✓ 안드로이드 실행 – npm run android
- ✓ iOS 실행 – npm run ios



Step One

Edit App.js to change this screen and then come back to see your edits.

See Your Changes

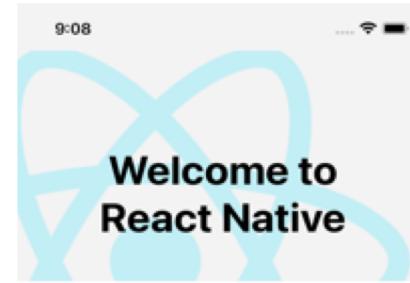
Double tap R on your keyboard to reload your app's code.

Debug

Press Cmd or Ctrl + M or Shake your device to open the React Native debug menu.

Learn More

Read the docs to discover what to do next:



Step One

Edit App.js to change this screen and then come back to see your edits.

See Your Changes

Press Cmd + R in the simulator to reload your app's code.

Debug

Press Cmd + D in the simulator or Shake your device to open the React Native debug menu.

Learn More

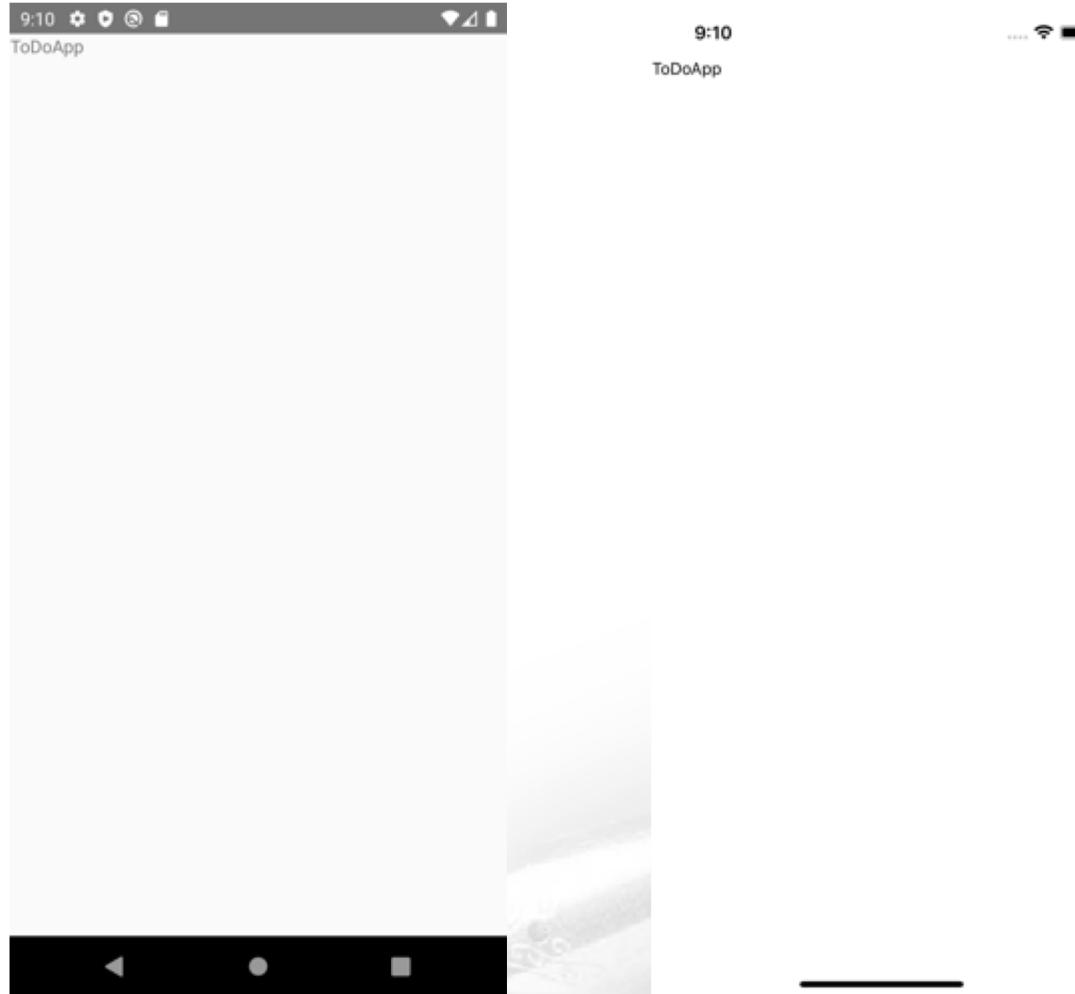
Read the docs to discover what to do next:

The Basics

Explains a Hello World for

프로젝트 기본 설정

- ❖ 프로젝트 생성 및 기본 설정 및 실행



프로젝트 기본 설정

- ❖ 프로젝트 생성 및 기본 설정 및 실행

- ✓ App.js 파일 수정

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

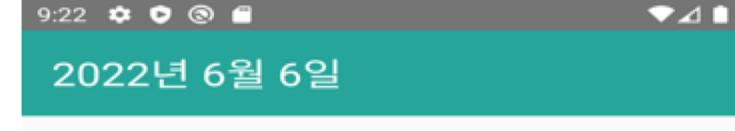
function App(){
  return (
    <SafeAreaView>
      <View>
        <Text>ToDoApp</Text>
      </View>
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({});

export default App;
```

메인 화면 구성

❖ 현재 날짜 출력



메인 화면 구성

❖ 현재 날짜 출력

- ✓ components 디렉토리를 생성하고 그 안에 DataHead.js 파일을 생성하고 작성

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

```
function DateHead({date}) {
  const year = date.getFullYear();
  const month = date.getMonth() + 1;
  const day = date.getDate();

  const formatted = `${year}년 ${month}월 ${day}일`;

  return (
    <View style={styles.block}>
      <Text style={styles.dateText}>
        {formatted}
      </Text>
    </View>
  );
}
```

메인 화면 구성

❖ 현재 날짜 출력

- ✓ components 디렉토리를 생성하고 그 안에 DataHead.js 파일을 생성하고 작성

```
const styles = StyleSheet.create({
  block: {
    padding: 16,
    backgroundColor: '#26a69a',
  },
  dateText: {
    fontSize: 24,
    color: 'white',
  },
});
export default DateHead;
```

메인 화면 구성

- ❖ 현재 날짜 출력

- ✓ App.js 파일 수정

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

import DateHead from './components/DateHead'

function App(){
  const today = new Date();

  return (
    <SafeAreaView>
      <View>
        <DateHead date={today} />
      </View>
    </SafeAreaView>
  );
}
```

메인 화면 구성

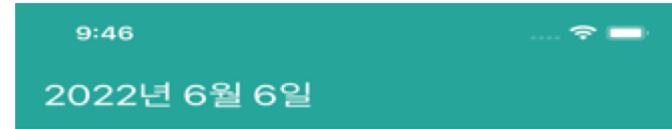
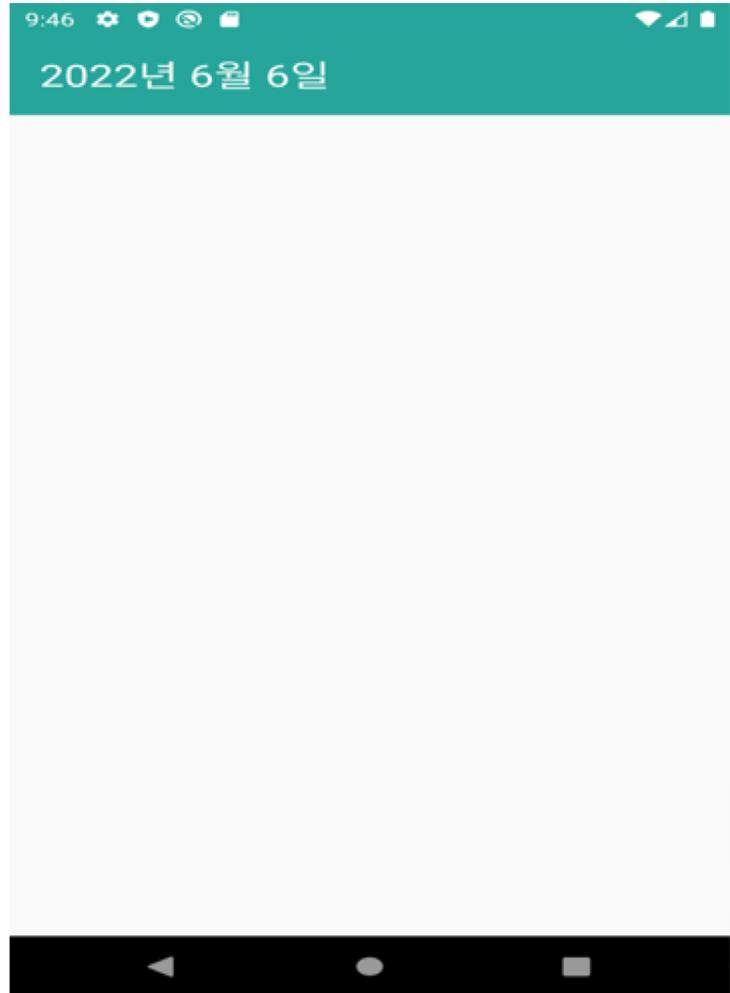
- ❖ 현재 날짜 출력
- ✓ App.js 파일 수정

```
const styles = StyleSheet.create({});
```

```
export default App;
```

메인 화면 구성

- ❖ StatusBar 색상 변경



메인 화면 구성

❖ 현재 날짜 출력

- ✓ DateHead.js 파일 수정

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

function DateHead({date}) {
    const year = date.getFullYear();
    const month = date.getMonth() + 1;
    const day = date.getDate();

    return (
        <>
        <StatusBar backgroundColor="#26a69a" />
        <View style={styles.block}>
            <Text style={styles.dateText}>
                {year}년 {month}월 {day}일
            </Text>
        </View>
        </>
    );
}
```

메인 화면 구성

❖ 현재 날짜 출력

- ✓ DateHead.js 파일 수정

```
const styles = StyleSheet.create({
  block: {
    padding: 16,
    backgroundColor: '#26a69a',
  },
  dateText: {
    fontSize: 24,
    color: 'white',
  },
});
export default DateHead;
```

메인 화면 구성

- ❖ 현재 날짜 출력



메인 화면 구성

❖ 현재 날짜 출력

- ✓ 특정 부분의 여백만 비활성화하고자 하는 경우 사용하는 react-native-safe-area-context 라이브러리 설치

yarn add react-native-safe-area-context

- iOS에서 아래와 같은 에러가 발생

error Could not find the following native modules: react-native-safe-area-context.
Did you forget to run "pod install" ?

- iOS에서 아래와 같은 에러가 발생

\$ cd ios

\$ pod install

\$ cd ../

\$ yarn ios

\$ yarn android

메인 화면 구성

- ❖ 현재 날짜 출력

- ✓ App.js 수정

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

```
import DateHead from './components/DateHead'
```

```
import {SafeAreaProvider, SafeAreaView} from 'react-native-safe-area-context';
```

```
function App(){
  const today = new Date();
```

```
  return (
    <SafeAreaProvider>
      <SafeAreaView edges={['bottom']}>
        <DateHead date={today} />
      </SafeAreaView>
    </SafeAreaProvider>
  );
}
```

메인 화면 구성

- ❖ 현재 날짜 출력

- ✓ App.js 수정

```
const styles = StyleSheet.create({});
```

```
export default App;
```



메인 화면 구성

❖ 현재 날짜 출력

- ✓ DateHead.js 수정 – StatusBar 의 높이를 구해서 여백 추가

```
import React from 'react';
import {View, Text, StyleSheet, StatusBar} from 'react-native';
import {useSafeAreaInsets} from 'react-native-safe-area-context';
```

```
function DateHead({date}) {
  const year = date.getFullYear();
  const month = date.getMonth() + 1;
  const day = date.getDate();
  const {top} = useSafeAreaInsets();

  return (
    <>
      <View style={[styles.statusBarPlaceholder, {height: top}]}/>
      <StatusBar backgroundColor="#26a69a" barStyle="light-content" />
      <View style={styles.block}>
        <Text style={styles.dateText}>
          {year}년 {month}월 {day}일
        </Text>
      </View>
    </>
  );
}
```

메인 화면 구성

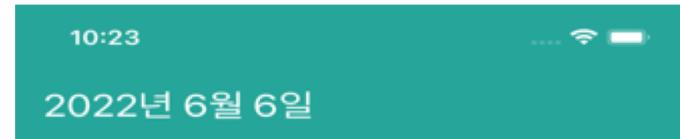
❖ 현재 날짜 출력

- ✓ DateHead.js 수정 – StatusBar 의 높이를 구해서 여백 추가

```
const styles = StyleSheet.create({
  statusBarPlaceholder: {
    backgroundColor: '#26a69a',
  },
  block: {
    padding: 16,
    backgroundColor: '#26a69a',
  },
  dateText: {
    fontSize: 24,
    color: 'white',
  },
});
export default DateHead;
```

메인 화면 구성

❖ 레이아웃 준비



현재는 할 일이 없습니다.



현재는 할 일이 없습니다.

메인 화면 구성

❖ 레이아웃 준비

- ✓ 하단에 보일 영역의 레이아웃을 위한 AddToDo.js 파일을 생성하고 작성

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

function AddToDo({onInsert}) {
  return (
    <View style={styles.block}>
      </View>
    );
}

const styles = StyleSheet.create({
  block: {
    backgroundColor: 'red',
    height: 64
  }
});

export default AddToDo;
```

메인 화면 구성

❖ 레이아웃 준비

- ✓ App.js 파일을 수정해서 AddToDo 컴포넌트를 하단에 배치

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

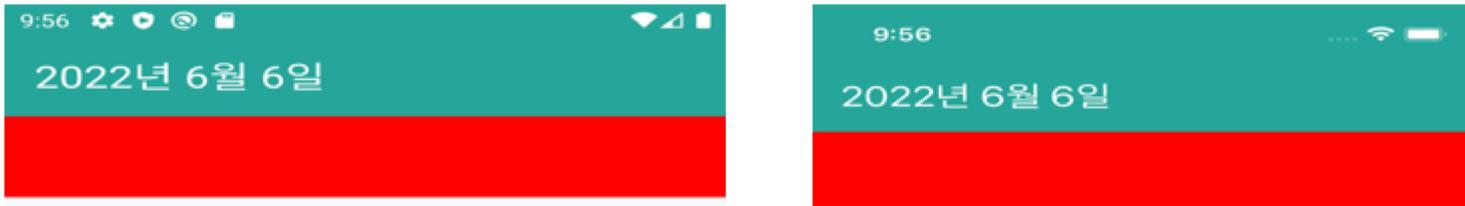
function AddToDo({onInsert}) {
  return (
    <View style={styles.block}>
      </View>
    );
}

const styles = StyleSheet.create({
  block: {
    backgroundColor: 'red',
    height: 64
  }
});

export default AddToDo;
```

메인 화면 구성

- ❖ 레이아웃 준비
- ✓ App.js 파일을 수정해서 AddToDo 컴포넌트를 하단에 배치



메인 화면 구성

❖ 레이아웃 준비

- ✓ 항목이 하나도 없을 때 보여질 컴포넌트를 Empty.js 파일을 component 디렉토리에 생성하고 작성

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

```
function Empty() {
  return (
    <View style={styles.block}>
      <Text style={styles.description}>현재는 할 일이 없습니다.</Text>
    </View>
  );
}
```

메인 화면 구성

❖ 레이아웃 준비

- ✓ 항목이 하나도 없을 때 보여질 컴포넌트를 Empty.js 파일을 component 디렉토리에 생성하고 작성

```
const styles = StyleSheet.create({
  block: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  description: {
    fontSize: 24,
    color: '#9e9e9e',
  },
});
export default Empty;
```

메인 화면 구성

❖ 레이아웃 준비

- ✓ App.js 파일을 수정해서 항목이 하나도 없을 때 보여질 컴포넌트를 배치

```
import React from 'react';
import {
  StyleSheet,
} from 'react-native';
import DateHead from './components/DateHead'
import {SafeAreaProvider, SafeAreaView} from 'react-native-safe-area-context';

import AddToDo from './components/AddToDo'
import Empty from './components/Empty'
```

메인 화면 구성

❖ 레이아웃 준비

- ✓ App.js 파일을 수정해서 항목이 하나도 없을 때 보여질 컴포넌트를 배치

```
function App(){
    const today = new Date();

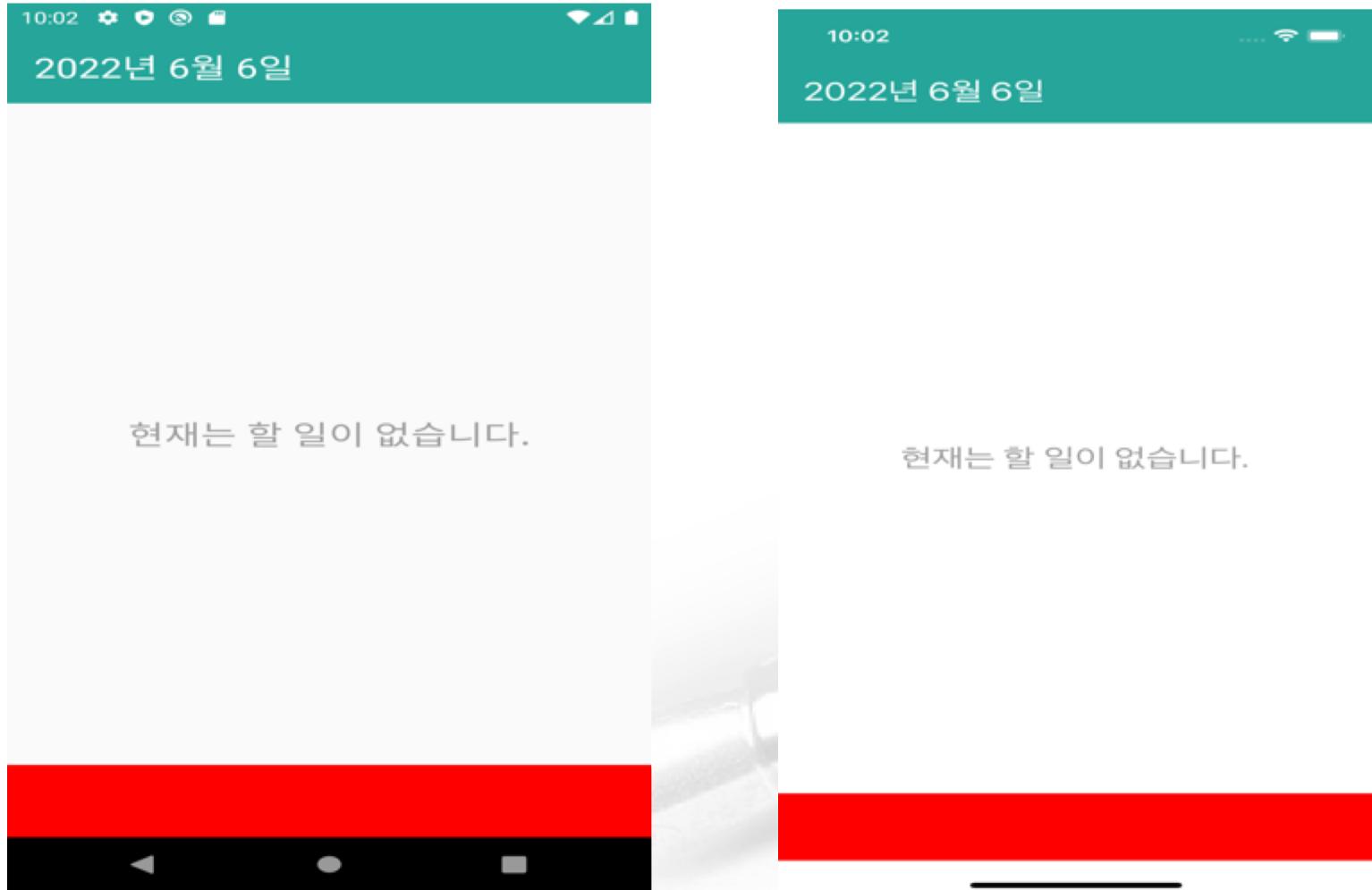
    return (
        <SafeAreaProvider>
            <SafeAreaView edges={['bottom']} style={styles.blck}>
                <DateHead date={today} />
                <Empty />
                <AddToDo />
            </SafeAreaView>
        </SafeAreaProvider>
    );
}

const styles = StyleSheet.create({
    blck:{
        flex:1
    }
});

export default App;
```

메인 화면 구성

- ❖ 레이아웃 준비
- ✓ App.js 파일을 수정해서 항목이 하나도 없을 때 보여질 컴포넌트를 배치



메인 화면 구성

❖ 레이아웃 준비

- ✓ 이미지를 사용하기 위해서 출력할 이미지를 assets 디렉토리를 생성하고 그 안에 images 디렉토리를 생성한 후 복사 – circle.png, [circle@2x.png](#), [circle@3x.png](#), young_and_happy.png

메인 화면 구성

❖ 레이아웃 준비

- ✓ 이미지 출력을 위해서 Empty.js 파일 수정

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

function Empty() {
  return (
    <View style={styles.block}>
      <Image source={require('../assets/images/young_and_happy.png')} style={styles.image}/>
      <Text style={styles.description}>현재는 할 일이 없습니다.</Text>
    </View>
  );
}
```

메인 화면 구성

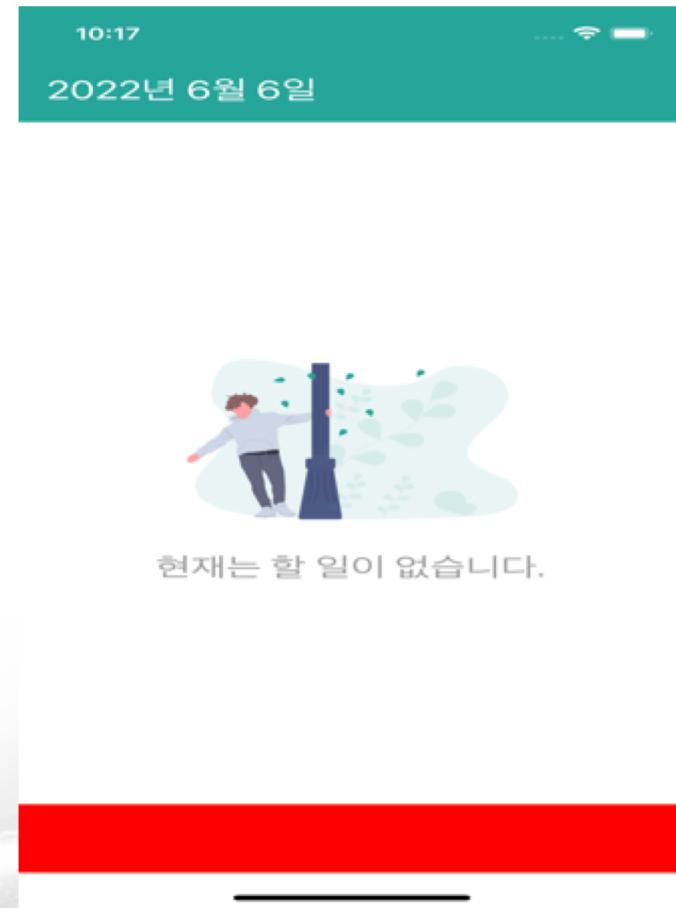
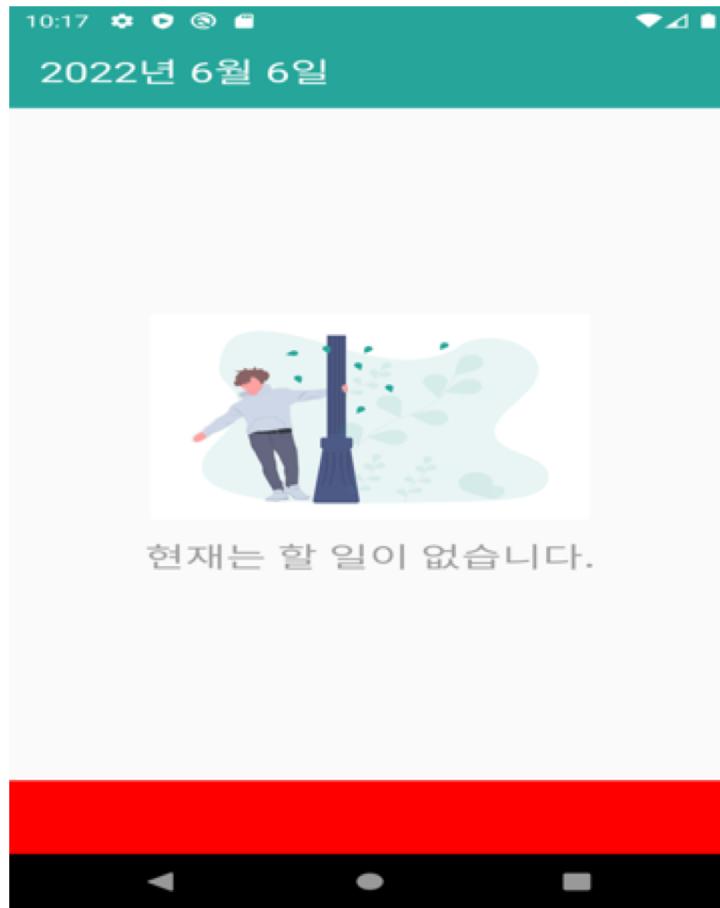
❖ 레이아웃 준비

- ✓ 이미지 출력을 위해서 Empty.js 파일 수정

```
const styles = StyleSheet.create({
  block: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  description: {
    fontSize: 24,
    color: '#9e9e9e',
  },
  image:{
    width:240,
    height:179,
    marginBottom:16
  }
});
export default Empty;
```

메인 화면 구성

❖ 레이아웃 준비



메인 화면 구성

❖ 레이아웃 준비

✓ 이미지 출력 옵션

- 해상도에 따른 이미지 출력은 이미지 파일 이름 뒤에 @2x, @3x 등의 형태로 만들어 주면 됨
- resizeMode: 이미지 크기를 조정할 때 실제 이미지 크기 와 값이 다를 때 리사이징 하는 모드로 cover, contain, stretch, repeat, center 등이 있음
- 외부 이미지를 이용하고자 할 때는 source 속성에 {{uri:'이미지 url'}} 의 형태로 지정하면 됨

사용자 키보드 입력

- ❖ AddToDo 컴포넌트에 텍스트 입력 도구 배치

- ✓ AddToDo.js 파일 수정

```
import React from 'react';
import {
  View,
  StyleSheet,
  TextInput
} from 'react-native';
```

```
function AddToDo({onInsert}) {
  return (
    <View style={styles.block}>
      <TextInput placeholder='할 일을 입력하세요' style={styles.input} />
    </View>
  );
}
```

사용자 키보드 입력

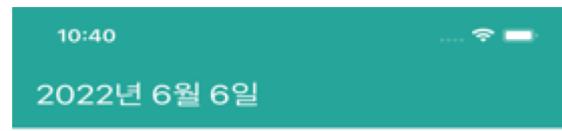
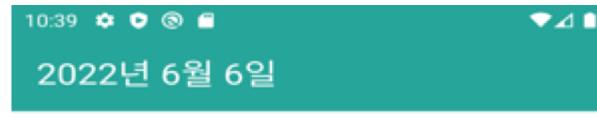
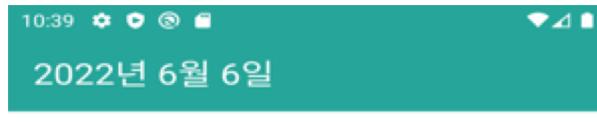
- ❖ AddToDo 컴포넌트에 텍스트 입력 도구 배치

- ✓ AddToDo.js 파일 수정

```
const styles = StyleSheet.create({
  block: {
    backgroundColor: 'white',
    height: 64,
    paddingHorizontal: 16,
    borderColor: '#bdbdbd',
    borderTopWidth: 1,
    borderBottomWidth: 1,
    justifyContent:'center'
  },
  input: {
    fontSize: 16,
    paddingVertical: 8,
  }
});
export default AddToDo;
```

사용자 키보드 입력

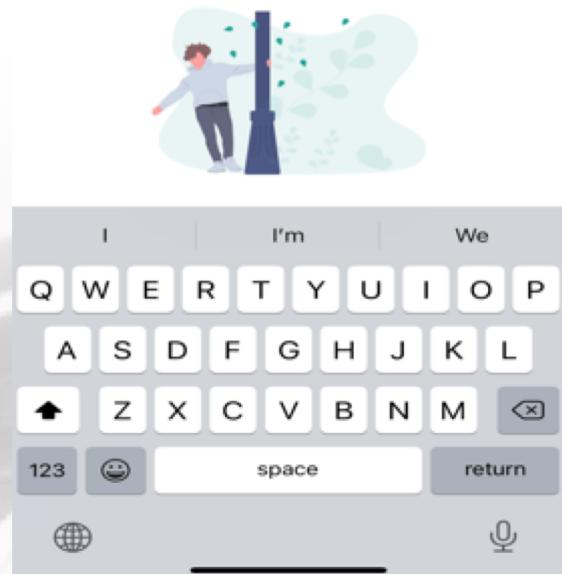
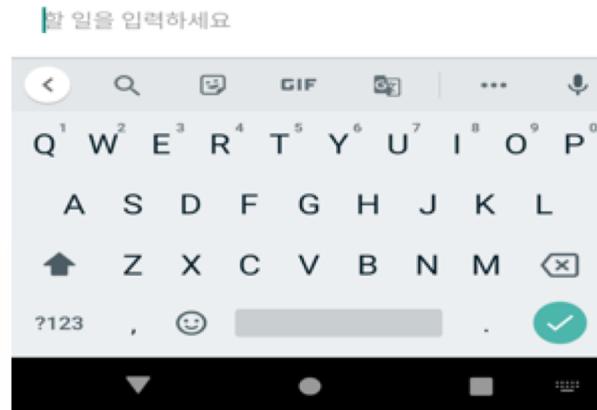
- ❖ AddToDo 컴포넌트에 텍스트 입력 도구 배치
- ✓ AddToDo.js 파일 수정



현재는 할 일이 없습니다.



현재는 할 일이 없습니다.



사용자 키보드 입력

- ❖ 키보드가 화면을 가리는 문제 해결

- ✓ AddToDo.js 파일 수정

```
import React from 'react';
import {
  StyleSheet, KeyboardAvoidingView, Platform
} from 'react-native';
import DateHead from './components/DateHead'
import {SafeAreaProvider, SafeAreaView} from 'react-native-safe-area-context';

import AddToDo from './components/AddToDo'
import Empty from './components/Empty'
```

사용자 키보드 입력

- ❖ 키보드가 화면을 가리는 문제 해결

- ✓ AddToDo.js 파일 수정

```
function App(){  
  const today = new Date();  
  
  return (  
    <SafeAreaProvider>  
      <SafeAreaView edges={['bottom']} style={styles.blck}>  
        <KeyboardAvoidingView  
          behavior={Platform.OS === 'ios' ? 'padding' : undefined} style={styles.avoid}>  
            <DateHead date={today} />  
            <Empty />  
            <AddToDo />  
            </KeyboardAvoidingView>  
        </SafeAreaView>  
    </SafeAreaProvider>  
  );  
};
```

사용자 키보드 입력

- ❖ 키보드가 화면을 가리는 문제 해결

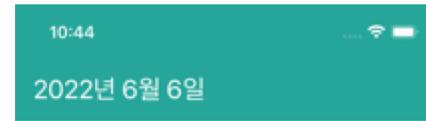
- ✓ AddToDo.js 파일 수정

```
const styles = StyleSheet.create({
  blk:{
    flex:1,
    backgroundColor:'white'
  },
  avoid: {
    flex:1
  }
});
```

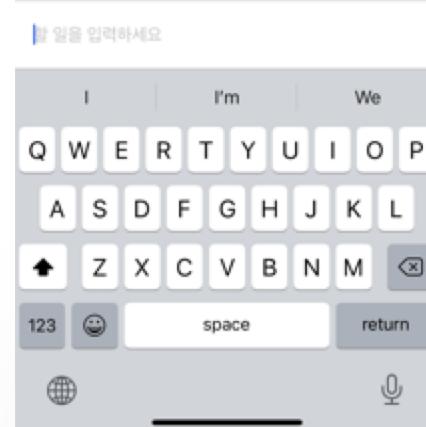
```
export default App;
```

사용자 키보드 입력

- ❖ 키보드가 화면을 가리는 문제 해결
- ✓ AddToDo.js 파일 수정



현재는 할 일이 없습니다.



사용자 키보드 입력

- ❖ useState를 이용한 상태 값 관리
- ✓ 문자열 타입의 상태 관리를 위해서 AddToDo.js 파일 수정

```
import React, {useState} from 'react';
import {
  View,
  StyleSheet,
  TextInput
} from 'react-native';

function AddToDo({onInsert}) {
  const [text, setText] = useState('')
  console.log(text)

  return (
    <View style={styles.block}>
      <TextInput placeholder='할 일을 입력하세요' style={styles.input} value={text}
      onChangeText={setText}/>
    </View>
  );
}
```

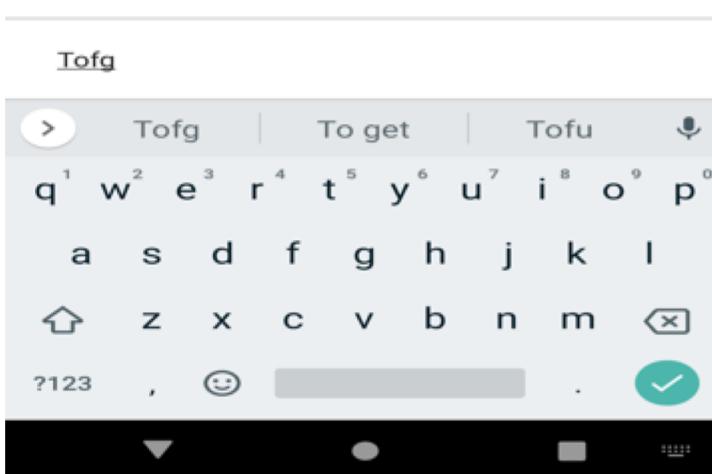
사용자 키보드 입력

- ❖ useState를 이용한 상태 값 관리
- ✓ 문자열 타입의 상태 관리를 위해서 AddToDo.js 파일 수정

```
const styles = StyleSheet.create({
  block: {
    backgroundColor: 'white',
    height: 64,
    paddingHorizontal: 16,
    borderColor: '#bdbdbd',
    borderTopWidth: 1,
    borderBottomWidth: 1,
    justifyContent:'center'
  },
  input: {
    fontSize: 16,
    paddingVertical: 8,
  }
});
export default AddToDo;
```

사용자 키보드 입력

- ❖ useState를 이용한 상태 값 관리
- ✓ 문자열 타입의 상태 관리를 위해서 AddToDo.js 파일 수정



사용자 키보드 입력

- ❖ 커스텀 버튼 만들기
- ✓ 프로젝트에 버튼에 사용할 이미지를 assets 디렉토리에 icons 디렉토리에 add_white 디렉토리에 복사

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기
- ✓ 커스텀 버튼을 만들기 위해서 AddToDo.js 파일을 수정

```
import React, {useState} from 'react';
import {
  View,
  StyleSheet,
  TextInput,
  Image
} from 'react-native';
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기

- ✓ 커스텀 버튼을 만들기 위해서 AddToDo.js 파일을 수정

```
function AddTodo({onInsert}) {
  const [text, setText] = useState('');

  return (
    <View style={styles.block}>
      <TextInput
        placeholder="할 일을 입력하세요."
        style={styles.input}
        value={text}
        onChangeText={setText}
      />
      <View style={styles.buttonStyle}>
        <Image source={require('../assets/icons/add_white/add_white.png')} />
      </View>
    </View>
  );
}
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기

- ✓ 커스텀 버튼을 만들기 위해서 AddToDo.js 파일을 수정

```
const styles = StyleSheet.create({  
  block: {  
    backgroundColor: 'white',  
    height: 64,  
    paddingHorizontal: 16,  
    borderColor: '#bdbdbd',  
    borderTopWidth: 1,  
    borderBottomWidth: 1,  
    alignItems: 'center',  
    flexDirection: 'row',  
  },  
  input: {  
    flex: 1,  
    fontSize: 16,  
    paddingVertical: 8,  
  },
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기
- ✓ 커스텀 버튼을 만들기 위해서 AddToDo.js 파일을 수정

```
buttonStyle: {  
    alignItems: 'center',  
    justifyContent: 'center',  
    width: 48,  
    height: 48,  
    backgroundColor: '#26a69a',  
    borderRadius: 24,  
}  
});
```

```
export default AddTodo;
```

사용자 키보드 입력

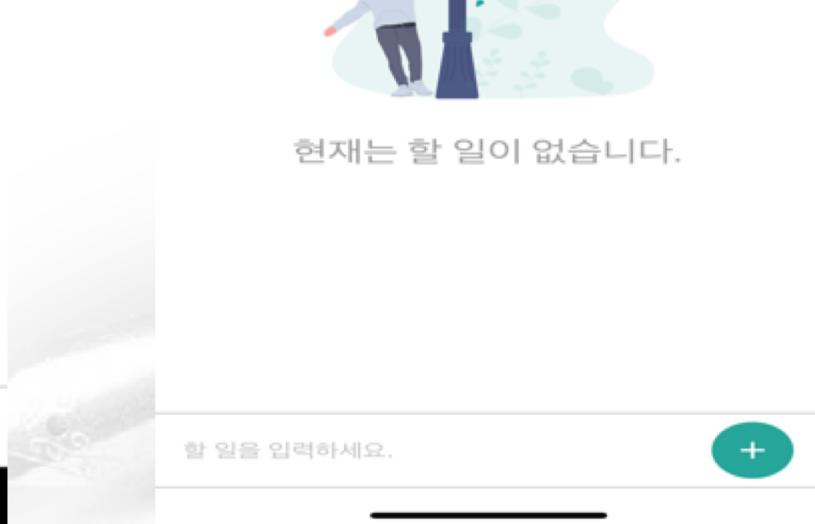
- ❖ 커스텀 버튼 만들기
- ✓ 커스텀 버튼을 만들기 위해서 AddToDo.js 파일을 수정



현재는 할 일이 없습니다.



현재는 할 일이 없습니다.



사용자 키보드 입력

- ❖ 커스텀 버튼 만들기

- ✓ 커스텀 버튼에서 터치를 하면 투명도가 조정되도록 AddToDo.js 파일을 수정

```
import React, {useState} from 'react';
import {
  View,
  StyleSheet,
  TextInput,
  Image,
  TouchableOpacity
} from 'react-native';
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기

- ✓ 커스텀 버튼에서 터치를 하면 투명도가 조정되도록 AddToDo.js 파일을 수정

```
function AddTodo({onInsert}) {
  const [text, setText] = useState('');

  return (
    <View style={styles.block}>
      <TextInput
        placeholder="할 일을 입력하세요."
        style={styles.input}
        value={text}
        onChangeText={setText}
      />
      <TouchableOpacity activeOpacity={0.5}>
        <View style={styles.buttonStyle}>
          <Image source={require('../assets/icons/add_white/add_white.png')} />
        </View>
      </TouchableOpacity>
    </View>
  );
}
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기

- ✓ 커스텀 버튼에서 터치를 하면 물결 효과가 나오도록 AddToDo.js 파일을 수정

```
import React, {useState} from 'react';
import {
  View,
  StyleSheet,
  TextInput,
  Image,
  TouchableOpacity,
  Platform,
  TouchableNativeFeedback,
} from 'react-native';
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기

- ✓ 커스텀 버튼에서 터치를 하면 물결 효과가 나오도록 AddToDo.js 파일을 수정

```
function AddTodo({onInsert}) {  
  const [text, setText] = useState("");  
  
  return (  
    <View style={styles.block}>  
      <TextInput  
        placeholder="할 일을 입력하세요."  
        style={styles.input}  
        value={text}  
        onChangeText={setText}  
      />  
  );  
}
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기

- ✓ 커스텀 버튼에서 터치를 하면 물결 효과가 나오도록 AddToDo.js 파일을 수정

```
{Platform.OS === 'ios' ? (
  <TouchableOpacity activeOpacity={0.5}>
    <View style={styles.buttonStyle}>
      <Image source={require('../assets/icons/add_white/add_white.png')} />
    </View>
  </TouchableOpacity>
):(
  <TouchableNativeFeedback>
    <View style={styles.buttonStyle}>
      <Image source={require('../assets/icons/add_white/add_white.png')} />
    </View>
  </TouchableNativeFeedback>
)
}
</View>
);
}
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기

- ✓ 커스텀 버튼에서 터치를 하면 물결 효과가 나오도록 AddToDo.js 파일을 수정

```
const styles = StyleSheet.create({
  block: {
    backgroundColor: 'white',
    height: 64,
    paddingHorizontal: 16,
    borderColor: '#bdbdbd',
    borderTopWidth: 1,
    borderBottomWidth: 1,
    alignItems: 'center',
    flexDirection: 'row',
  },
  input: {
    flex: 1,
    fontSize: 16,
    paddingVertical: 8,
  },
})
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기

- ✓ 커스텀 버튼에서 터치를 하면 물결 효과가 나오도록 AddToDo.js 파일을 수정

```
buttonStyle: {  
    alignItems: 'center',  
    justifyContent: 'center',  
    width: 48,  
    height: 48,  
    backgroundColor: '#26a69a',  
    borderRadius: 24,  
}  
});
```

```
export default AddTodo;
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기
 - ✓ 물결 효과가 원형 안에서만 나타나도록 AddToDo.js 파일을 수정

```
import React, {useState} from 'react';
import {
  View,
  StyleSheet,
  TextInput,
  Image,
  TouchableOpacity,
  Platform,
  TouchableNativeFeedback,
} from 'react-native';
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기

- ✓ 물결 효과가 원형 안에서만 나타나도록 AddToDo.js 파일을 수정

```
function AddTodo({onInsert}) {
  const [text, setText] = useState('');

  return (
    <View style={styles.block}>
      <TextInput
        placeholder="할 일을 입력하세요."
        style={styles.input}
        value={text}
        onChangeText={setText}
      />
      {Platform.OS === 'ios' ? (
        <TouchableOpacity activeOpacity={0.5}>
          <View style={styles.buttonStyle}>
            <Image source={require('../assets/icons/add_white/add_white.png')} />
          </View>
        </TouchableOpacity>
      ):(
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기
- ✓ 물결 효과가 원형 안에서만 나타나도록 AddToDo.js 파일을 수정

```
<View style={styles.circleWrapper}>
  <TouchableNativeFeedback>
    <View style={styles.buttonStyle}>
      <Image source={require('../assets/icons/add_white/add_white.png')} />
    </View>
  </TouchableNativeFeedback>
</View>
)
}
</View>
);
}
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기

- ✓ 물결 효과가 원형 안에서만 나타나도록 AddToDo.js 파일을 수정

```
const styles = StyleSheet.create({
  block: {
    backgroundColor: 'white',
    height: 64,
    paddingHorizontal: 16,
    borderColor: '#bdbdbd',
    borderTopWidth: 1,
    borderBottomWidth: 1,
    alignItems: 'center',
    flexDirection: 'row',
  },
  input: {
    flex: 1,
    fontSize: 16,
    paddingVertical: 8,
  },
})
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기
- ✓ 물결 효과가 원형 안에서만 나타나도록 AddToDo.js 파일을 수정

```
buttonStyle: {  
    alignItems: 'center',  
    justifyContent: 'center',  
    width: 48,  
    height: 48,  
    backgroundColor: '#26a69a',  
    borderRadius: 24,  
},  
circleWrapper:{  
    overflow: 'hidden',  
    borderRadius: 24  
}  
});  
  
export default AddTodo;
```

사용자 키보드 입력

❖ 커스텀 버튼 만들기

- ✓ 버튼을 눌렀을 때 텍스트를 비우기 위해서 AddToDo.js 파일을 수정

```
import React, {useState} from 'react';
import {
  View,
  StyleSheet,
  TextInput,
  Image,
  TouchableOpacity,
  Platform,
  TouchableNativeFeedback,
  Keyboard
} from 'react-native';
```

사용자 키보드 입력

❖ 커스텀 버튼 만들기

- ✓ 버튼을 눌렀을 때 텍스트를 비우기 위해서 AddToDo.js 파일을 수정

```
function AddTodo({onInsert}) {
  const [text, setText] = useState('');

  const onPress = () => {
    setText('');
    Keyboard.dismiss();
  }

  return (
    <View style={styles.block}>
      <TextInput
        placeholder="할 일을 입력하세요."
        style={styles.input}
        value={text}
        onChangeText={setText}
      />
    
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기

- ✓ 버튼을 눌렀을 때 텍스트를 비우기 위해서 AddToDo.js 파일을 수정

```
{Platform.OS === 'ios' ? (
  <TouchableOpacity activeOpacity={0.5} onPress={onPress}>
    <View style={styles.buttonStyle}>
      <Image source={require('../assets/icons/add_white/add_white.png')} />
    </View>
  </TouchableOpacity>
):(
  <View style={styles.circleWrapper}>
    <TouchableNativeFeedback onPress={onPress}>
      <View style={styles.buttonStyle}>
        <Image source={require('../assets/icons/add_white/add_white.png')} />
      </View>
    </TouchableNativeFeedback>
  </View>
)
}
</View>
);
}
```

사용자 키보드 입력

- ❖ 커스텀 버튼 만들기

- ✓ Enter를 눌렀을 때 도 onPress 를 호출하고 returnType 설정을 위해서 AddToDo.js 파일을 수정

```
<TextInput  
    placeholder="할 일을 입력하세요."  
    style={styles.input}  
    value={text}  
    onChangeText={setText}  
    onSubmitEditing={onPress}  
    returnType="done"  
/>
```

todos 상태 만들기 및 항목 출력

- ❖ todos 상태 만들기
- ✓ 데이터를 저장할 todos 배열을 App.js 파일에 추가

```
import React, {useState} from 'react';
```

...

```
function App(){  
  const today = new Date();  
  
  const [todos, setTodos] = useState([  
    {id: 1, text: '작업 환경 설정', done: true},  
    {id: 2, text: 'BackEnd - SpringBoot', done: true},  
    {id: 3, text: 'FrontEnd - ReactNative', done: false},  
  ]);  
}
```

todos 상태 만들기 및 항목 출력

- ❖ ToDoList 컴포넌트 생성

- ✓ FlatList를 사용해 여러 항목을 출력하기 위한 ToDoList 컴포넌트를 위해서 components 디렉토리에 ToDoList.js 파일을 생성하고 작성

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

```
function ToDoList({todos}) {
  return (
    <FlatList
      style={styles.list}
      data={todos}
      renderItem={({item}) => (
        <View>
          <Text>{item.text}</Text>
        </View>
      )}
      keyExtractor={item => item.id.toString()}
    />
  );
}
```

todos 상태 만들기 및 항목 출력

- ❖ ToDoList 컴포넌트 생성
 - ✓ FlatList를 사용해 여러 항목을 출력하기 위한 ToDoList 컴포넌트를 위해서 components 디렉토리에 ToDoList.js 파일을 생성하고 작성

```
const styles = StyleSheet.create({
  list: {
    flex: 1,
  }
});
```

```
export default ToDoList;
```

todos 상태 만들기 및 항목 출력

- ❖ ToDoList 컴포넌트 생성
- ✓ 데이터 출력을 위해서 App.js 파일 수정

```
import React, {useState} from 'react';
import {
  StyleSheet, KeyboardAvoidingView, Platform
} from 'react-native';
import DateHead from './components/DateHead'
import {SafeAreaProvider, SafeAreaView} from 'react-native-safe-area-context';

import AddToDo from './components/AddToDo'
import Empty from './components/Empty'
import ToDoList from './components/ToDoList';
```

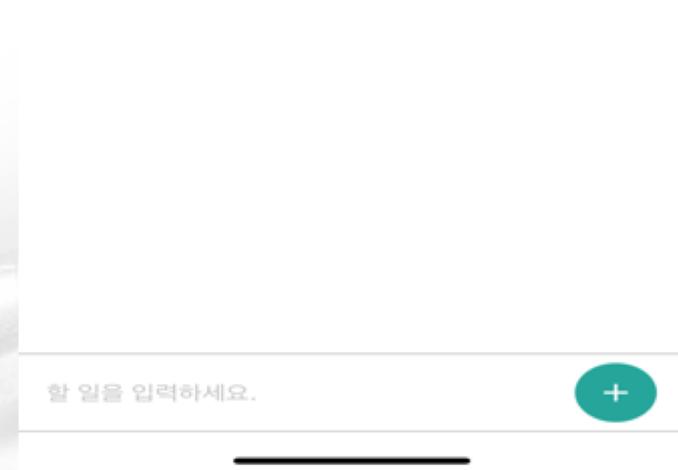
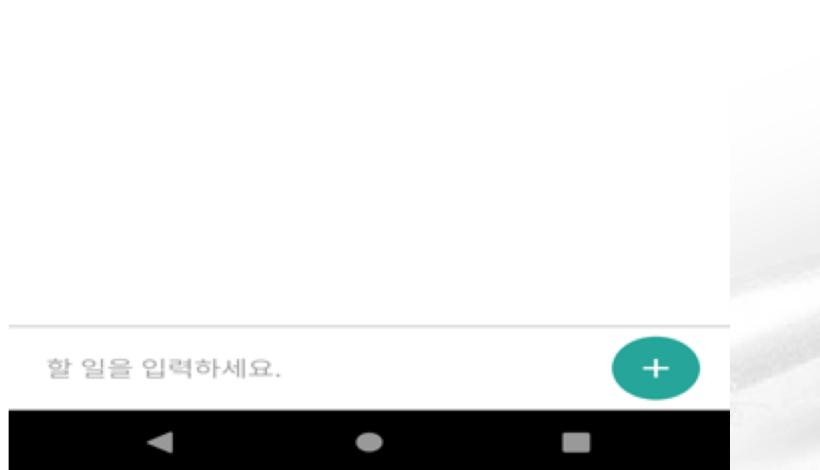
todos 상태 만들기 및 항목 출력

- ❖ ToDoList 컴포넌트 생성
- ✓ 데이터 출력을 위해서 App.js 파일 수정

```
function App(){  
    const today = new Date();  
  
    const [todos, setTodos] = useState([  
        {id: 1, text: '작업 환경 설정', done: true},  
        {id: 2, text: 'BackEnd - SpringBoot', done: true},  
        {id: 3, text: 'FrontEnd - ReactNative', done: false},  
    ]);  
  
    return (  
        <SafeAreaProvider>  
            <SafeAreaView edges={['bottom']} style={styles.blck}>  
                <KeyboardAvoidingView  
                    behavior={Platform.OS === 'ios' ? 'padding' : undefined} style={styles.avoid}>  
                    <DateHead date={today} />  
                    {todos.length === 0 ? <Empty /> : <ToDoList todos={todos} />}  
                    <AddToDo />  
                </KeyboardAvoidingView>  
            </SafeAreaView>  
        </SafeAreaProvider>  
    );  
};
```

todos 상태 만들기 및 항목 출력

- ❖ ToDoList 컴포넌트 생성



todos 상태 만들기 및 항목 출력

- ❖ ToDoItem 컴포넌트 생성

- ✓ ToDoList 의 하나의 항목이 될 컴포넌트를 위해서 ToDoItem.js 파일을 생성하고 작성

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

```
function ToDoItem({id, text, done}) {
  return (
    <View style={styles.item}>
      <View style={styles.circle} />
      <Text style={styles.text}>{text}</Text>
    </View>
  );
}
```

todos 상태 만들기 및 항목 출력

- ❖ ToDoItem 컴포넌트 생성

- ✓ ToDoList 의 하나의 항목이 될 컴포넌트를 위해서 ToDoItem.js 파일을 생성하고 작성

```
const styles = StyleSheet.create({
  item: {
    flexDirection: 'row',
    padding: 16,
    borderBottomColor: '#e0e0e0',
    alignItems: 'center',
  },
  circle: {
    width: 24,
    height: 24,
    borderRadius: 12,
    borderColor: '#26a69a',
    borderWidth: 1,
    marginRight: 16,
  },
  text: {
    flex: 1,
    fontSize: 16,
    color: '#212121',
  }
});
```

```
export default ToDoItem;
```

todos 상태 만들기 및 항목 출력

- ❖ ToDoItem 컴포넌트 생성

- ✓ ToDoList.js 수정

```
import React from 'react';
import {FlatList, View, Text, StyleSheet} from 'react-native';
import ToDoItem from './ToDoItem';

function ToDoList({todos}) {
  return (
    <FlatList
      style={styles.list}
      data={todos}
      renderItem={({item}) => (
        <ToDoItem id={item.id} text={item.text} done={item.done} />
      )}
      keyExtractor={item => item.id.toString()}
    />
  );
}
const styles = StyleSheet.create({
  list: {
    flex: 1,
  }
});
export default ToDoList;
```

todos 상태 만들기 및 항목 출력

- ❖ ToDoltem 컴포넌트 생성



- 작업 환경 설정
- BackEnd - SpringBoot
- FrontEnd - ReactNative



- 작업 환경 설정
- BackEnd - SpringBoot
- FrontEnd - ReactNative

할 일을 입력하세요.

+

할 일을 입력하세요.

+

todos 상태 만들기 및 항목 출력

- ❖ 항목 사이에 구분선 출력

- ✓ ToDoList.js 수정

```
import React from 'react';
import {FlatList, View, Text, StyleSheet} from 'react-native';
import ToDoItem from './ToDoItem';

function ToDoList({todos}) {
  return (
    <FlatList
      ItemSeparatorComponent={() => <View style={styles.separator} />}
      style={styles.list}
      data={todos}
      renderItem={({item}) => (
        <ToDoItem id={item.id} text={item.text} done={item.done} />
      )}
      keyExtractor={item => item.id.toString()}
    />
  );
}
```

todos 상태 만들기 및 항목 출력

- ❖ 항목 사이에 구분선 출력

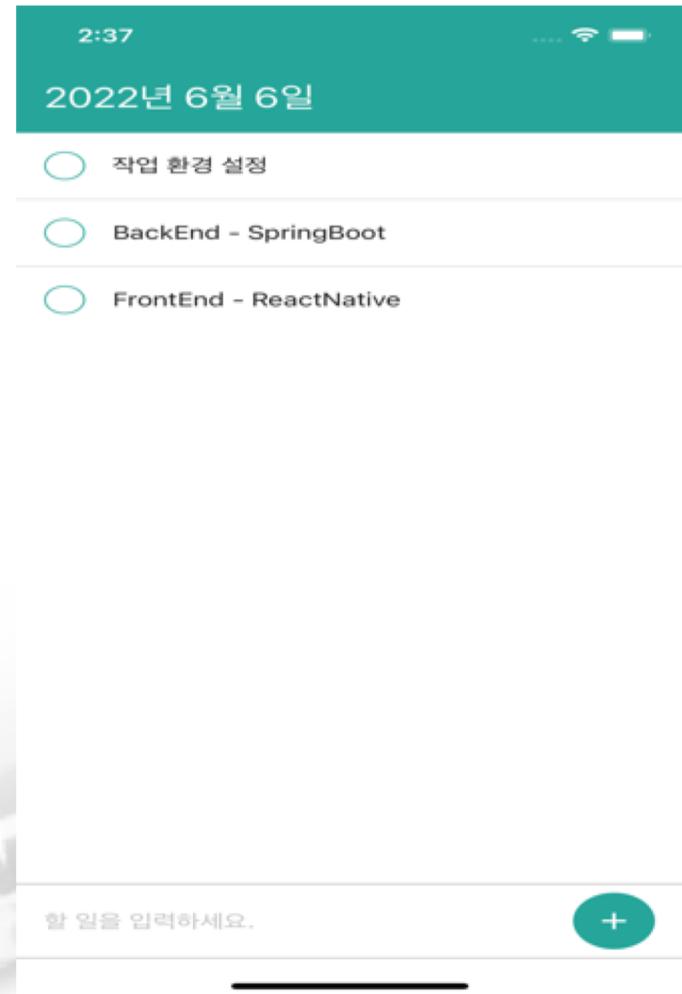
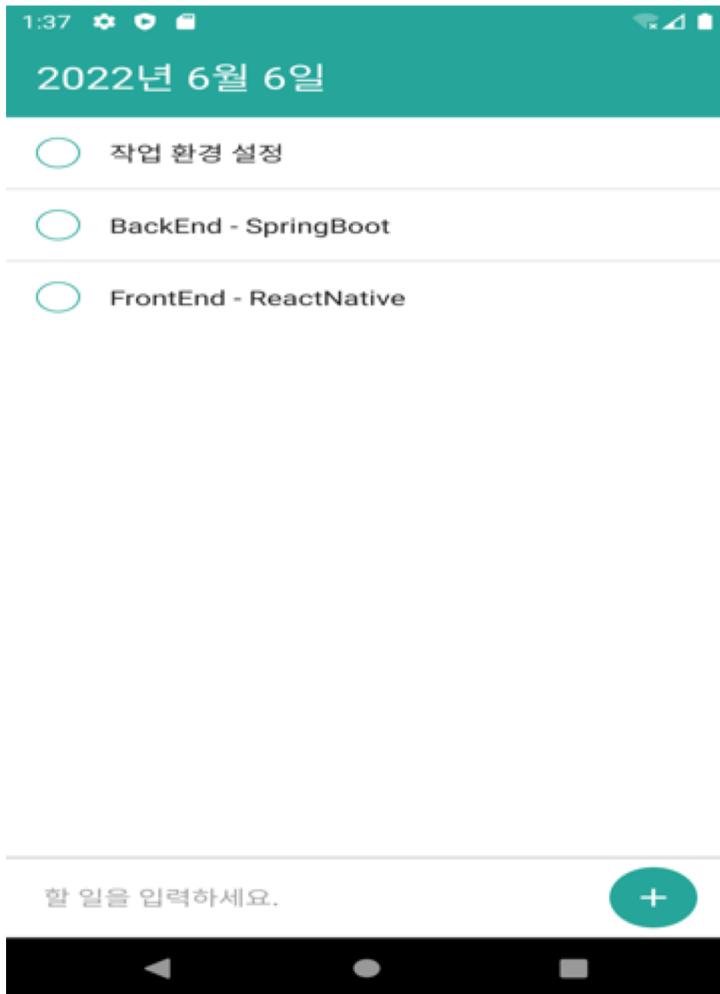
- ✓ ToDoList.js 수정

```
const styles = StyleSheet.create({
  list: {
    flex: 1,
  },
  separator: {
    backgroundColor: '#e0e0e0',
    height: 1
  }
});
```

```
export default ToDoList;
```

todos 상태 만들기 및 항목 출력

- ❖ 항목 사이에 구분선 출력



todos 상태 만들기 및 항목 출력

- ❖ 완료한 항목에 다른 스타일 적용
- ✓ 아이콘으로 사용할 check_white.png 파일을 assets/check_image 디렉토리에 복사

todos 상태 만들기 및 항목 출력

- ❖ 완료한 항목에 다른 스타일 적용

- ✓ ToDoItem.js 파일을 수정

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

function ToDoItem({id, text, done}) {
  return (
    <View style={styles.item}>
      <View style={[styles.circle, done && styles.filled]}>
        {done && (<Image source={require('../assets/icons/check_white/check_white.png')} />)}
      </View>
      <Text style={[styles.text, done && styles.lineThrough]}>{text}</Text>
    </View>
  );
}
```

todos 상태 만들기 및 항목 출력

- ❖ 완료한 항목에 다른 스타일 적용

- ✓ ToDoItem.js 파일을 수정

```
const styles = StyleSheet.create({
  item: {
    flexDirection: 'row',
    padding: 16,
    borderBottomColor: '#e0e0e0',
    alignItems: 'center',
  },
  circle: {
    width: 24,
    height: 24,
    borderRadius: 12,
    borderColor: '#26a69a',
    borderWidth: 1,
    marginRight: 16,
  },
  text: {
    flex: 1,
    fontSize: 16,
    color: '#212121',
  },
})
```

todos 상태 만들기 및 항목 출력

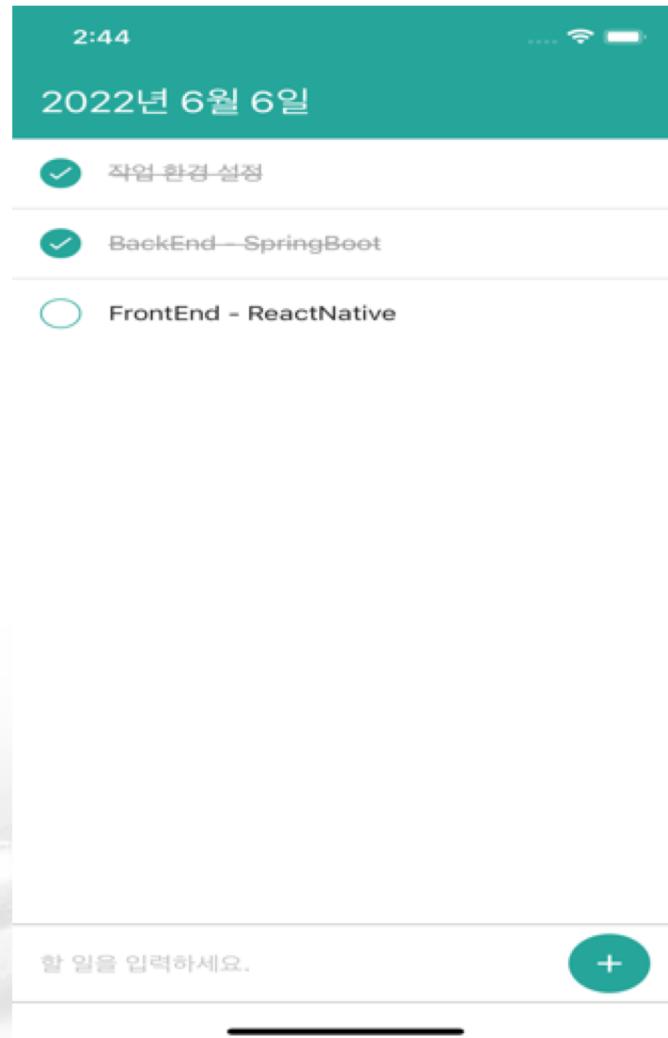
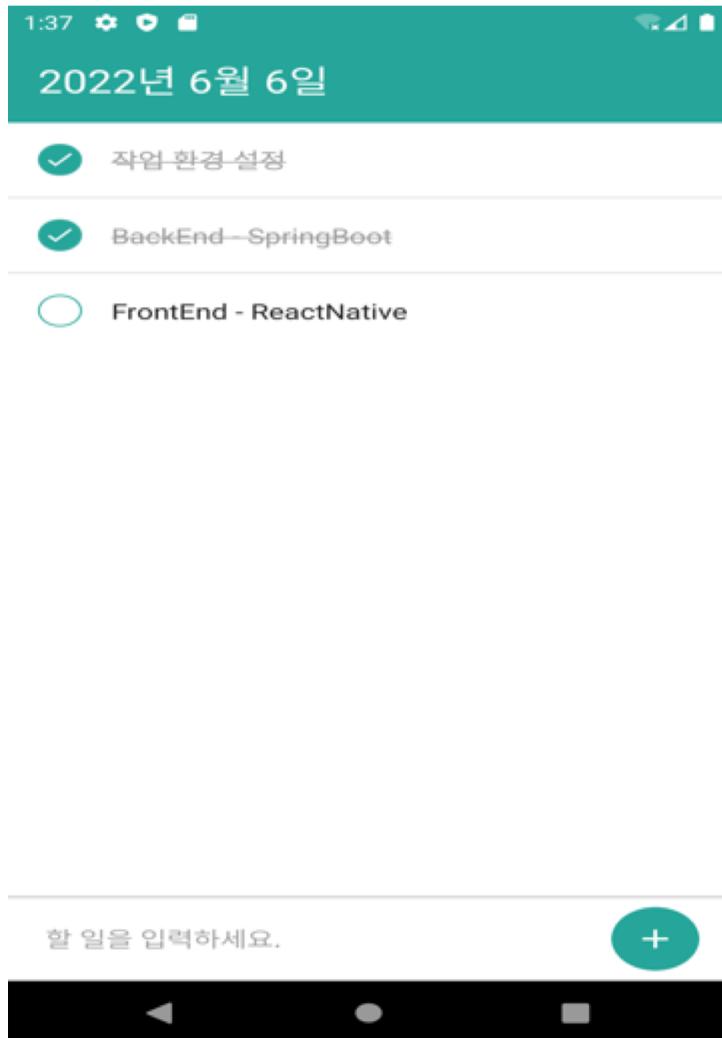
- ❖ 완료한 항목에 다른 스타일 적용

- ✓ ToDoItem.js 파일을 수정

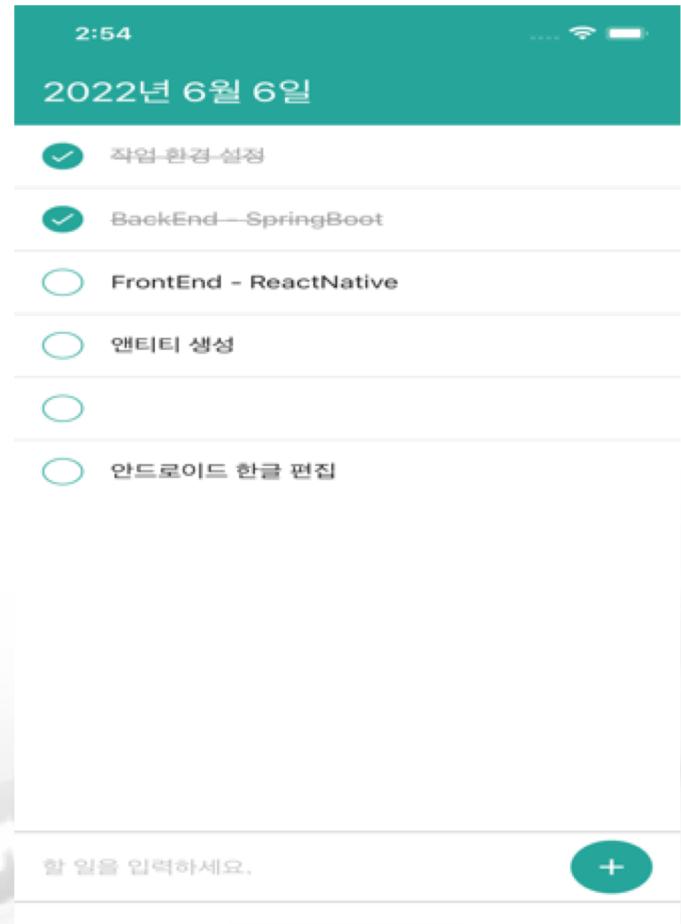
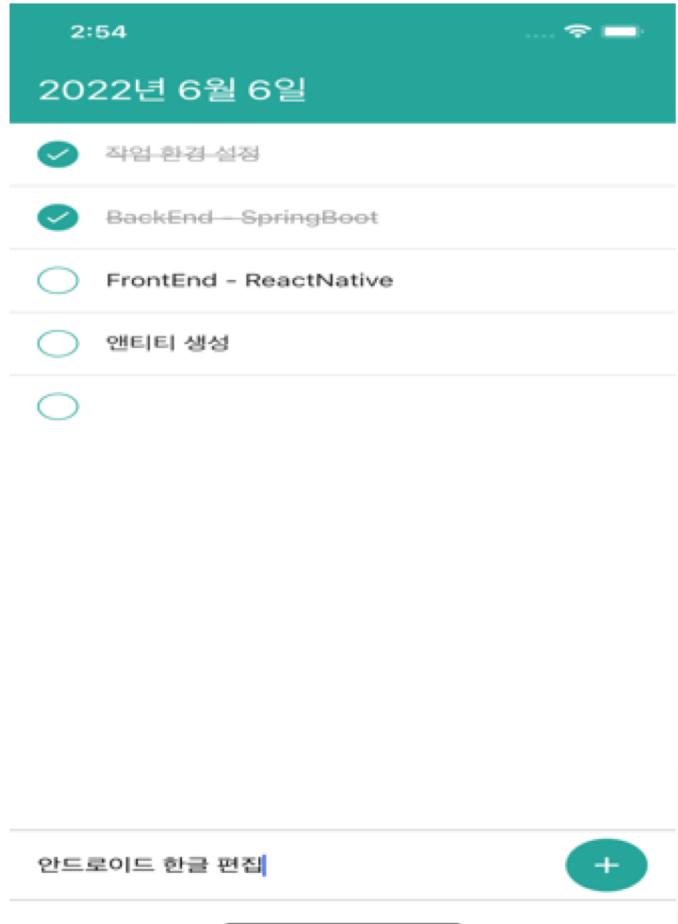
```
filled:{  
    justifyContent: 'center',  
    alignItems: 'center',  
    backgroundColor: '#26a69a'  
},  
lineThrough:{  
    color: '#9e9e9e',  
    textDecorationLine: 'line-through'  
}  
});  
  
export default ToDoItem;
```

todos 상태 만들기 및 항목 출력

- ❖ 완료한 항목에 다른 스타일 적용



새 항목 등록



새 항목 등록

- ❖ App.js 파일에 새 항목 등록에 사용할 함수 생성

```
import React, {useState} from 'react';
import {
  StyleSheet, KeyboardAvoidingView, Platform
} from 'react-native';
import DateHead from './components/DateHead'
import {SafeAreaProvider, SafeAreaView} from 'react-native-safe-area-context';

import AddToDo from './components/AddToDo'
import Empty from './components/Empty'
import ToDoList from './components/ToDoList';

function App(){
  const today = new Date();

  const [todos, setTodos] = useState([
    {id: 1, text: '작업 환경 설정', done: true},
    {id: 2, text: 'BackEnd - SpringBoot', done: true},
    {id: 3, text: 'FrontEnd - ReactNative', done: false},
  ]);
}
```

새 항목 등록

- ❖ App.js 파일에 새 항목 등록에 사용할 함수 생성

```
const onInsert = text => {
  // 새로 등록할 항목의 id를 구합니다.
  // 등록된 항목 중에서 가장 큰 id를 구하고, 그 값에 1을 더합니다.
  // 만약 리스트가 비어있다면 1을 id로 사용합니다.
  const nextId =
    todos.length > 0 ? Math.max(...todos.map(todo => todo.id)) + 1 : 1;
  const todo = {
    id: nextId,
    text,
    done: false,
  };

  setTodos(todos.concat(todo));
};
```

새 항목 등록

- ❖ App.js 파일에 새 항목 등록에 사용할 함수 생성

```
return (
  <SafeAreaProvider>
    <SafeAreaView edges={['bottom']} style={styles.blck}>
      <KeyboardAvoidingView
        behavior={Platform.OS === 'ios' ? 'padding' : undefined} style={styles.avoid}>
        <DateHead date={today} />
        {todos.length === 0 ? <Empty /> : <ToDoList todos={todos} />}
        <AddToDo onInsert={onInsert}>
      </KeyboardAvoidingView>
    </SafeAreaView>
  </SafeAreaProvider>
);
}
const styles = StyleSheet.create({
  blck: {
    flex: 1,
    backgroundColor: 'white'
  },
  avoid: {
    flex: 1
  }
});
export default App;
```

새 항목 등록

- ❖ AddToDo.js 파일에서 이벤트 처리 함수에 연결

```
import React, {useState} from 'react';
import {
  View,
  StyleSheet,
  TextInput,
  Image,
  TouchableOpacity,
  Platform,
  TouchableNativeFeedback,
  Keyboard
} from 'react-native';

function AddTodo({onInsert}) {
  const [text, setText] = useState('');

  const onPress = () => {
    onInsert(text)
    setText('');
    Keyboard.dismiss();
  }
}
```

할일 완료 상태 토글



할일 완료 상태 토글

- ❖ App.js 파일에 할일 완료 토글을 위한 함수 생성

```
import React, {useState} from 'react';
import {
  StyleSheet, KeyboardAvoidingView, Platform
} from 'react-native';
import DateHead from './components/DateHead'
import {SafeAreaProvider, SafeAreaView} from 'react-native-safe-area-context';

import AddToDo from './components/AddToDo'
import Empty from './components/Empty'
import ToDoList from './components/ToDoList';

function App(){
  const today = new Date();

  const [todos, setTodos] = useState([
    {id: 1, text: '작업 환경 설정', done: true},
    {id: 2, text: 'BackEnd - SpringBoot', done: true},
    {id: 3, text: 'FrontEnd - ReactNative', done: false},
  ]);
}
```

할일 완료 상태 토글

- ❖ App.js 파일에 할일 완료 토글을 위한 함수 생성

```
const onInsert = text => {
  // 새로 등록할 항목의 id를 구합니다.
  // 등록된 항목 중에서 가장 큰 id를 구하고, 그 값에 1을 더합니다.
  // 만약 리스트가 비어있다면 1을 id로 사용합니다.
  const nextId =
    todos.length > 0 ? Math.max(...todos.map(todo => todo.id)) + 1 : 1;
  const todo = {
    id: nextId,
    text,
    done: false,
  };
  setTodos(todos.concat(todo));
};

const onToggle = id => {
  const nextTodos = todos.map(todo =>
    todo.id === id ? {...todo, done: !todo.done} : todo,
  );
  setTodos(nextTodos);
};
```

할일 완료 상태 토글

- ❖ App.js 파일에 할일 완료 토글을 위한 함수 생성

```
return (
  <SafeAreaProvider>
    <SafeAreaView edges={['bottom']} style={styles.blck}>
      <KeyboardAvoidingView
        behavior={Platform.OS === 'ios' ? 'padding' : undefined} style={styles.avoid}>
        <DateHead date={today} />
        {todos.length === 0 ? <Empty /> : <ToDoList todos={todos} onToggle={onToggle} />}
        <AddToDo onInsert={onInsert}>
        </KeyboardAvoidingView>
      </SafeAreaView>
    </SafeAreaProvider>
  );
}
```

할일 완료 상태 토글

- ❖ ToDoList.js 파일에 할일 완료 토클을 ToDoItem에서 사용하기 위해 수정

```
import React from 'react';
import {FlatList, View, Text, StyleSheet} from 'react-native';
import ToDoItem from './ToDoItem';

function ToDoList({todos, onToggle}) {
  return (
    <FlatList
      ItemSeparatorComponent={() => <View style={styles.separator} />}
      style={styles.list}
      data={todos}
      renderItem={({item}) => (
        <ToDoItem id={item.id} text={item.text} done={item.done} onToggle={onToggle} />
      )}
      keyExtractor={item => item.id.toString()}
    />
  );
}
```

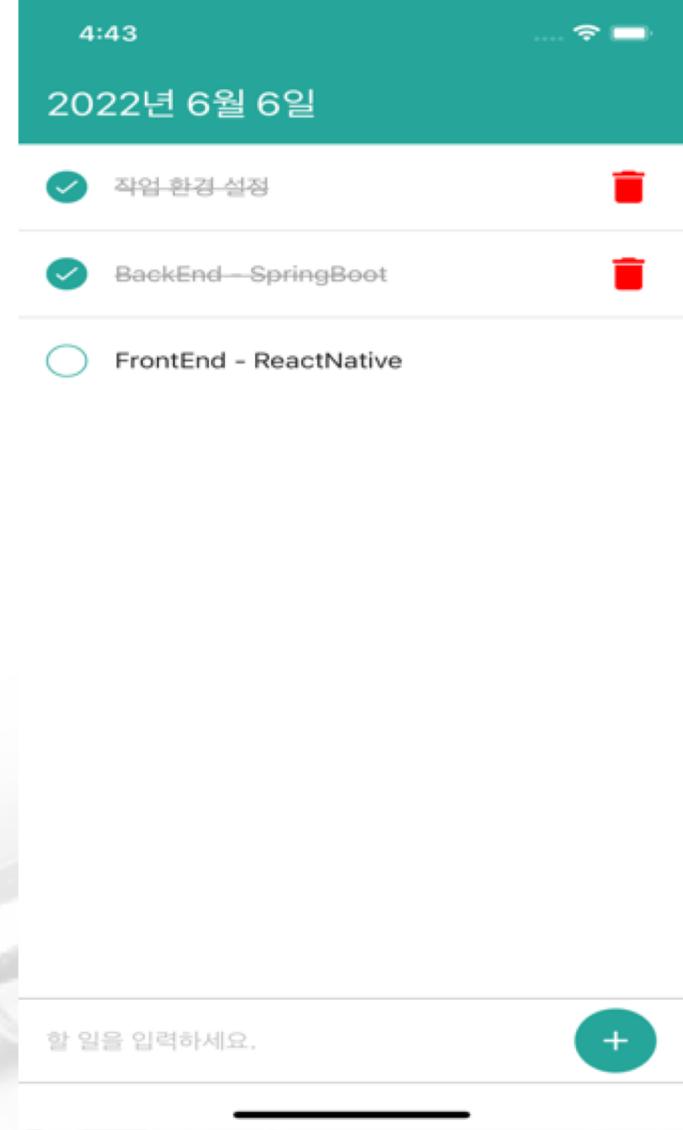
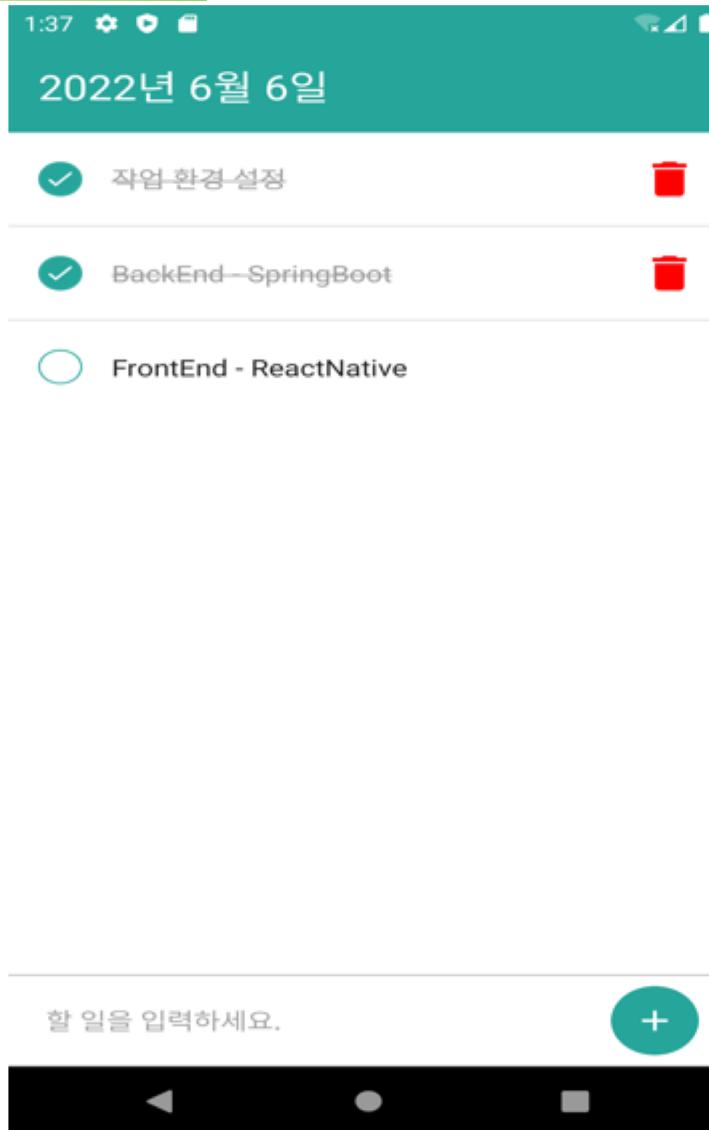
할일 완료 상태 토글

- ❖ ToDoItem.js 파일에 할일 완료 토글을 연결 위해 수정

```
import React from 'react';
import {
  View,
  Text,
  StyleSheet,
  Image,
  TouchableOpacity
} from 'react-native';

function ToDoItem({id, text, done, onToggle}) {
  return (
    <View style={styles.item}>
      <TouchableOpacity onPress={() => onToggle(id)}>
        <View style={[styles.circle, done && styles.filled]}>
          {done && (<Image source={require('../assets/icons/check_white/check_white.png')} />)}
        </View>
      </TouchableOpacity>
      <Text style={[styles.text, done && styles.lineThrough]}>{text}</Text>
    </View>
  );
}
```

항목 삭제



항목 삭제

- ❖ 벡터 아이콘 사용

- ✓ react-native-vector-icons 사용을 위해서 설치 - <https://oblador.github.io/react-native-vector-icons>

```
yarn add react-native-vector-icons
```

항목 삭제

- ❖ 벡터 아이콘 사용

- ✓ ios 디렉토리에서 pod 을 이용한 설치

- ```
cd ios
```

- ```
pod install
```

- ✓ ios/ToDoApp/Info.plist 파일을 열어서 속성 추가

- ```
<key>UIAppFonts</key>
<array>
 <string>MaterialIcons.ttf</string>
</array>
```

# 항목 삭제

- ❖ 벡터 아이콘 사용

- ✓ android/app/build.gradle 파일을 열고 맨 아래에 텍스트 추가

- ```
apply from: file("../node_modules/react-native-vector-icons/fonts.gradle")
```

- ✓ 안드로이드 다시 빌드

- ```
yarn android
```

# 항목 삭제

## ❖ 삭제 아이콘 출력

- ✓ ToDoItem.js 파일 수정

```
import React from 'react';
import {
 View,
 Text,
 StyleSheet,
 Image,
 TouchableOpacity
} from 'react-native';
```

```
import Icon from 'react-native-vector-icons/MaterialIcons'
```

# 항목 삭제

## ❖ 삭제 아이콘 출력

- ✓ ToDoItem.js 파일 수정

```
function ToDoItem({id, text, done, onToggle}) {
 return (
 <View style={styles.item}>
 <TouchableOpacity onPress={() => onToggle(id)}>
 <View style={[styles.circle, done && styles.filled]}>
 {done && (<Image source={require('../assets/icons/check_white/check_white.png')} />)}
 </View>
 </TouchableOpacity>
 <Text style={[styles.text, done && styles.lineThrough]}>{text}</Text>
 {done ? (
 <Icon name="delete" size={32} color="red" />
):(
 <View style={styles.removePlaceholder} />
)}
 </View>
);
}
```

# 항목 삭제

## ❖ 삭제 아이콘 출력

- ✓ ToDoItem.js 파일 수정

```
const styles = StyleSheet.create({
 item: {
 flexDirection: 'row',
 padding: 16,
 borderBottomColor: '#e0e0e0',
 alignItems: 'center',
 },
 circle: {
 width: 24,
 height: 24,
 borderRadius: 12,
 borderColor: '#26a69a',
 borderWidth: 1,
 marginRight: 16,
 },
 text: {
 flex: 1,
 fontSize: 16,
 color: '#212121',
 },
})
```

# 항목 삭제

- ❖ 삭제 아이콘 출력
- ✓ ToDoItem.js 파일 수정

```
filled:{
 justifyContent: 'center',
 alignItems: 'center',
 backgroundColor: '#26a69a'
},
lineThrough:{
 color: '#9e9e9e',
 textDecorationLine: 'line-through'
},
removePlaceholder:{
 width:32,
 height: 32
}
});

export default ToDoItem;
```

# 항목 삭제

- ❖ 항목 삭제 함수 생성

- ✓ App.js 파일 수정

```
import React, {useState} from 'react';
import {
 StyleSheet, KeyboardAvoidingView, Platform
} from 'react-native';
import DateHead from './components/DateHead'
import {SafeAreaProvider, SafeAreaView} from 'react-native-safe-area-context';

import AddToDo from './components/AddToDo'
import Empty from './components/Empty'
import ToDoList from './components/ToDoList';
```

# 항목 삭제

- ❖ 항목 삭제 함수 생성

- ✓ App.js 파일 수정

```
function App(){
 const today = new Date();

 const [todos, setTodos] = useState([
 {id: 1, text: '작업 환경 설정', done: true},
 {id: 2, text: 'BackEnd - SpringBoot', done: true},
 {id: 3, text: 'FrontEnd - ReactNative', done: false},
]);

 const onInsert = text => {
 // 새로 등록할 항목의 id를 구합니다.
 // 등록된 항목 중에서 가장 큰 id를 구하고, 그 값에 1을 더합니다.
 // 만약 리스트가 비어있다면 1을 id로 사용합니다.
 const nextId =
 todos.length > 0 ? Math.max(...todos.map(todo => todo.id)) + 1 : 1;
 const todo = {
 id: nextId,
 text,
 done: false,
 };

 setTodos(todos.concat(todo));
 };
```

# 항목 삭제

- ❖ 항목 삭제 함수 생성

- ✓ App.js 파일 수정

```
const onToggle = id => {
 const nextTodos = todos.map(todo =>
 todo.id === id ? {...todo, done: !todo.done} : todo,
);
 setTodos(nextTodos);
};

const onRemove = id => {
 const nextToDos = todos.filter(todo => todo.id !== id);
 setTodos(nextToDos);
}
```

# 항목 삭제

## ❖ 항목 삭제 함수 생성

### ✓ App.js 파일 수정

```
return (
 <SafeAreaProvider>
 <SafeAreaView edges={['bottom']} style={styles.blck}>
 <KeyboardAvoidingView
 behavior={Platform.OS === 'ios' ? 'padding' : undefined} style={styles.avoid}>
 <DateHead date={today} />
 {todos.length === 0 ? <Empty /> : <ToDoList todos={todos} onToggle={onToggle}
 onRemove={onRemove} />}
 <AddToDo onInsert={onInsert}/>
 </KeyboardAvoidingView>
 </SafeAreaView>
 </SafeAreaProvider>
);
};
```

# 항목 삭제

- ❖ 항목 삭제 함수 생성

- ✓ ToDoList.js 파일 수정

```
import React from 'react';
import {FlatList, View, Text, StyleSheet} from 'react-native';
import ToDoItem from './ToDoItem';

function ToDoList({todos, onToggle, onRemove}) {
 return (
 <FlatList
 ItemSeparatorComponent={() => <View style={styles.separator} />}
 style={styles.list}
 data={todos}
 renderItem={({item}) => (
 <ToDoItem id={item.id} text={item.text} done={item.done} onToggle={onToggle}
onRemove={onRemove} />
)}
 keyExtractor={item => item.id.toString()}
 />
);
}
```

# 항목 삭제

- ❖ 항목 삭제 함수 생성

- ✓ ToDoItem.js 파일 수정

```
import React from 'react';
import {
 View,
 Text,
 StyleSheet,
 Image,
 TouchableOpacity
} from 'react-native';
```

```
import Icon from 'react-native-vector-icons/MaterialIcons'
```

# 항목 삭제

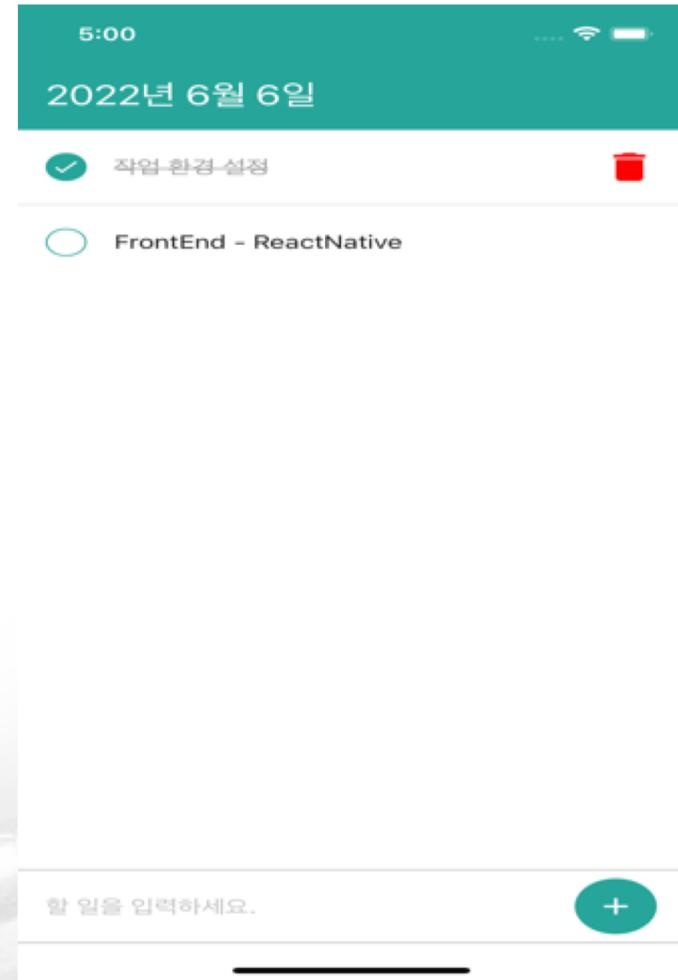
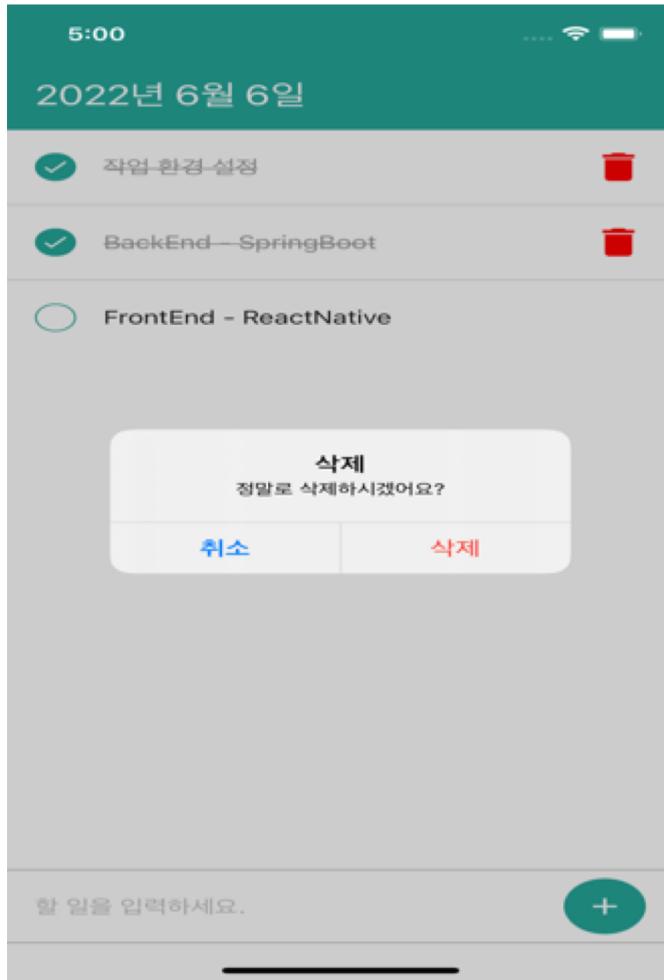
- ❖ 항목 삭제 함수 생성

- ✓ ToDoItem.js 파일 수정

```
function ToDoItem({id, text, done, onToggle, onRemove}) {
 return (
 <View style={styles.item}>
 <TouchableOpacity onPress={() => onToggle(id)}>
 <View style={[styles.circle, done && styles.filled]}>
 {done && (<Image source={require('../assets/icons/check_white/check_white.png')} />) }
 </View>
 </TouchableOpacity>
 <Text style={[styles.text, done && styles.lineThrough]}>{text}</Text>
 {done ? (
 <TouchableOpacity onPress={() => onRemove(id)}>
 <Icon name="delete" size={32} color="red" />
 </TouchableOpacity>
):(
 <View style={styles.removePlaceholder} />
)}
 </View>
);
}
```

# 항목 삭제

- ❖ 항목을 삭제하기 전에 한 번 물어보기



# 항목 삭제

- ❖ 항목을 삭제하기 전에 한 번 물어보기

- ✓ ToDoItem.js 파일 수정

```
import React from 'react';
import {
 View,
 Text,
 StyleSheet,
 Image,
 TouchableOpacity,
 Alert
} from 'react-native';
```

```
import Icon from 'react-native-vector-icons/MaterialIcons'
```

# 항목 삭제

- ❖ 항목을 삭제하기 전에 한 번 물어보기

- ✓ ToDoItem.js 파일 수정

```
function ToDoItem({id, text, done, onToggle, onRemove}) {
 const remove = () => {
 Alert.alert(
 '삭제',
 '정말로 삭제하시겠어요?',
 [
 {text: '취소', onPress: () => {}, style: 'cancel'},
 {
 text: '삭제',
 onPress: () => {
 onRemove(id);
 },
 style: 'destructive',
 },
],
 {
 cancelable: true,
 onDismiss: () => {},
 },
);
 };
}
```

# 항목 삭제

- ❖ 항목을 삭제하기 전에 한 번 물어보기
- ✓ ToDoItem.js 파일 수정

```
return (
 <View style={styles.item}>
 <TouchableOpacity onPress={() => onToggle(id)}>
 <View style={[styles.circle, done && styles.filled]}>
 {done && (<Image source={require('../assets/icons/check_white/check_white.png')} />)}
 </View>
 </TouchableOpacity>
 <Text style={[styles.text, done && styles.lineThrough]}>{text}</Text>
 {done ? (
 <TouchableOpacity onPress={remove}>
 <Icon name="delete" size={32} color="red" />
 </TouchableOpacity>
):(
 <View style={styles.removePlaceholder} />
)}
 </View>
);
}
```

# AsyncStorage를 이용한 데이터 유지

## ❖ AsyncStorage

- ✓ AsyncStorage는 리액트 네이티브에서 사용할 수 있는 key-value 형식의 저장소
- ✓ iOS에는 네이티브 코드로 구현되어 있으며 안드로이드에서는 네이티브 코드와 SQLite를 기반으로 구현되어 있음
- ✓ AsyncStorage는 브라우저에서 사용하는 localstorage 와 유사
- ✓ 값을 저장할 때는 문자열 타입으로 저장해야 하며 getItem, setItem, removeItem, clear 등 localstorage에서 사용하는 메서드와 같은 이름을 가진 메서드들도 존재
- ✓ localstorage와의 큰 차이점이라면 AsyncStorage는 비동기적으로 작동한다는 것으로 값을 조회하거나 설정할 때 Promise를 반환
- ✓ <https://reactnative.dev/docs/asyncstorage>
- ✓ 설치

```
yarn add @react-native-community/async-storage
```

```
cd ios
```

```
pod install
```

# AsyncStorage를 이용한 데이터 유지

## ❖ AsyncStorage 적용

- ✓ AsyncStorage를 사용하는 코드를 추상화하는 코드를 작성하기 위한 코드를 storages/todoStorage.js 파일을 생성하고 작성

```
import AsyncStorage from '@react-native-community/async-storage';
const key = 'todos';
const todosStorage = {
 async get() {
 try {
 const rawTodos = await AsyncStorage.getItem(key);
 const savedTodos = JSON.parse(rawTodos);
 return savedTodos;
 } catch (e) {
 throw new Error('Failed to load todos');
 }
 },
 async set(data) {
 try {
 await AsyncStorage.setItem(key, JSON.stringify(data));
 } catch (e) {
 throw new Error('Failed to save todos');
 }
 },
};
export default todosStorage;
```

# AsyncStorage를 이용한 데이터 유지

- ❖ AsyncStorage 적용
- ✓ App.js 파일 수정

```
import React, {useState, useEffect} from 'react';
import {StyleSheet, KeyboardAvoidingView, Platform} from 'react-native';
import {SafeAreaProvider, SafeAreaView} from 'react-native-safe-area-context';
import todosStorage from './storages/todosStorage';

import DateHead from './components/DateHead';
import AddToDo from './components/AddToDo';
import Empty from './components/Empty';
import ToDoList from './components/ToDoList';
```

# AsyncStorage를 이용한 데이터 유지

- ❖ AsyncStorage 적용

- ✓ App.js 파일 수정

```
function App() {
 const today = new Date();
 const [todos, setTodos] = useState([
 {id: 1, text: '작업환경 설정', done: true},
 {id: 2, text: '리액트 네이티브 기초 공부', done: false},
 {id: 3, text: '투두리스트 만들어보기', done: false},
]);

 // 불러오기
 useEffect(() => {
 todosStorage.get().then(setTodos).catch(console.error);
 }, []);

 useEffect(() => {
 todosStorage.set(todos).catch(console.error);
 }, [todos]);
}
```

# AsyncStorage를 이용한 데이터 유지

- ❖ AsyncStorage 적용
- ✓ App.js 파일 수정

```
const onInsert = text => {
 // 새로 등록할 항목의 id를 구합니다.
 // 등록된 항목 중에서 가장 큰 id를 구하고, 그 값에 1을 더합니다.
 // 만약 리스트가 비어있다면 1을 id로 사용합니다.
 const nextId =
 todos.length > 0 ? Math.max(...todos.map(todo => todo.id)) + 1 : 1;
 const todo = {
 id: nextId,
 text,
 done: false,
 };

 setTodos(todos.concat(todo));
};

const onToggle = id => {
 const nextTodos = todos.map(todo =>
 todo.id === id ? {...todo, done: !todo.done} : todo,
);
 setTodos(nextTodos);
};
```

# AsyncStorage를 이용한 데이터 유지

- ❖ AsyncStorage 적용
- ✓ App.js 파일 수정

```
const onRemove = id => {
 const nextTodos = todos.filter(todo => todo.id !== id);
 setTodos(nextTodos);
};
```

# AsyncStorage를 이용한 데이터 유지

- ❖ AsyncStorage 적용

- ✓ App.js 파일 수정

```
return (
 <SafeAreaProvider>
 <SafeAreaView edges={['bottom']} style={styles.block}>
 <KeyboardAvoidingView
 behavior={Platform.select({ios: 'padding'})}
 style={styles.avoid}>
 <DateHead date={today} />
 {todos.length === 0 ? (
 <Empty />
) : (
 <ToDoList todos={todos} onToggle={onToggle} onRemove={onRemove} />
)}
 <AddToDo onInsert={onInsert} />
 </KeyboardAvoidingView>
 </SafeAreaView>
 </SafeAreaProvider>
);
}
```

# AsyncStorage를 이용한 데이터 유지

- ❖ AsyncStorage 적용
- ✓ App.js 파일 수정

```
const styles = StyleSheet.create({
 block: {
 flex: 1,
 backgroundColor: 'white',
 },
 avoid: {
 flex: 1,
 },
});
```

```
export default App;
```