



[스파르타코딩클럽] 프론트엔드의 꽃, 리액트 - 4주차



매 주차 강의자료 시작에 PDF파일을 올려두었어요!

▼ PDF 파일

[수업 목표]

1. keyframes를 설치하고 애니메이션 효과를 넣어본다.
2. 서버와 서버리스에 대해 이해한다.
3. realtime database를 어렵풋 알아본다.
4. firebase를 이용한 BaaS 환경을 설정하자.
5. React-firebase 사용법을 익힌다.

[목차]

- 01. [keyframes](#)
- 02. [버킷리스트에 프로그래스바 달기](#)
- 03. [스크롤바 움직이기](#)
- 04. [Quiz_버킷리스트 좀 더 예쁘게!](#)
- 05. [서버와 서버리스](#)
- 06. [Firebase란?](#)
- 07. [Firebase 설정하기](#)
- 08. [Firestore 설정하기](#)
- 09. [리액트에 Firebase 연동하기](#)
- 10. [Firestore 데이터 가지고 놀기](#)
- 11. [끝 & 숙제 설명](#)
- 12. [4주차 숙제 답안 코드](#)



모든 토글을 열고 닫는 단축키

Windows : `Ctrl` + `alt` + `t`

Mac : `⌘` + `⌥` + `t`

01. keyframes

▼ 1) keyframes 사용하기

☞ keyframes는 styled-components안에 이미 들어있습니다!
웹에서 애니메이션을 구현할때는 transition과 animation이라는 스타일 속성을 많이 사용합니다.

☞ transition은 단순한 엘리먼트 상태변화에 쓰기 좋고,
animation은 다이나믹한 효과를 주는데 쓰기 좋아요!

Keyframes은 animation에서 사용하는 속성 중 하나입니다!

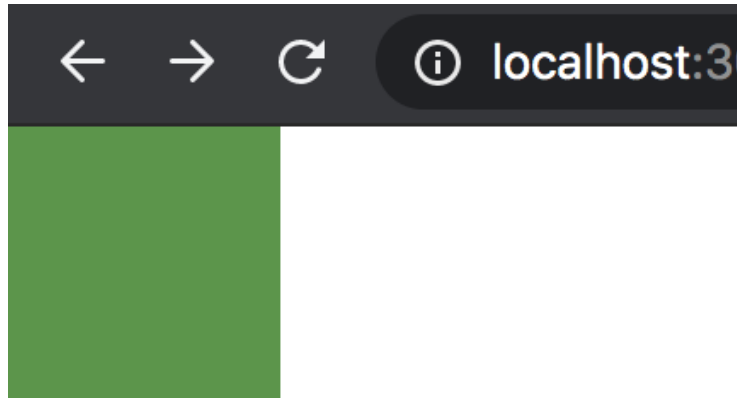
▼ css에서는 이런 식으로 keyframes를 씁니다.

```
.box {  
  width: 100px;  
  height: 100px;  
  background: #444;  
}  
.box.active {  
  animation: boxFade 2s 1s infinite linear alternate;  
}  
@keyframes boxFade {  
  0% {  
    opacity: 1;  
  }  
  50% {  
    opacity: 0;  
  }  
  100% {  
    opacity: 1;  
  }  
}
```

- 프로젝트를 하나 새로 만들어서 시작해볼까요!

☞ 새 프로젝트에 styled-components를 설치해주세요!

- 네모 박스를 하나 만들어주세요.



```
import React from 'react';
import './App.css';

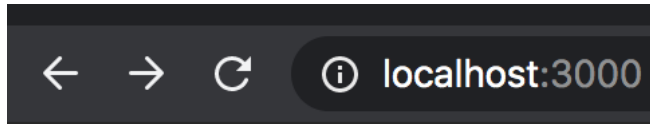
// styled와 keyframes를 불러옵니다!
import styled, {keyframes} from "styled-components";

function App() {
  return (
    <div className="App">
      <Box></Box>
    </div>
  );
}
// 박스를 먼저 하나 만들어줍니다
const Box = styled.div`
  width: 100px;
  height: 100px;
  background: green;
`;

export default App;
```

▼ 2) keyframes으로 움직이는 동그라미 만들기

- Box를 동그랗게 만들고,



```
...
const Box = styled.div`
  width: 100px;
  height: 100px;
  border-radius: 50px;
  background: green;
  animation: ${boxFade} 2s 1s infinite linear alternate;
`;
...
```

- position을 준 다음,

```
...
const Box = styled.div`
  width: 100px;
  height: 100px;
  border-radius: 50px;
  background: green;
  position: absolute;
  top: 20px;
  left: 20px;
  animation: ${boxFade} 2s 1s infinite linear alternate;
`;
...
```

- 위 아래로 움직이게 해보자!

```
...
// 이런식으로 동시에 여러가지 애니메이션을 넣어줄 수 있어요!
const boxFade = keyframes`
  0% {
    opacity: 1;
    top: 20px;
  }
  50% {
    opacity: 0;
    top: 400px;
  }
  100% {
    opacity: 1;
    top: 20px;
  }
`
```

};
...



이거 재미있죠? styled-components와 keyframes로 할 수 있는 건 훨씬 많아요!
여러 가지 애니메이션 효과를 찾아서 넣어보세요. 즐거울거예요.

02. 버킷리스트에 프로그래스바 달기

▼ 3) 모양보기

내 버킷리스트

영화관 가기

매일 책읽기

수영 배우기

일출보러가기

일출보러가기

추가하기

▼ 4) 버킷리스트에 프로그래스 바 달기

- 상태값 형태부터 바꾸자!



기본 값을 딕셔너리로 만들어서, 버킷리스트 텍스트하고 완료 여부를 넣어볼게요.
list: [{text: '버킷리스트', completed: true}, ...] 형태가 되겠죠!

▼ [코드스니펫] - bucket.js

```
// Actions
const LOAD = "bucket/LOAD";
const CREATE = "bucket/CREATE";
const DELETE = "bucket/DELETE";
```

```

const initialState = {
  // list: ["영화관 가기", "매일 책읽기", "수영 배우기"],
  list: [
    { text: "영화관 가기", completed: false },
    { text: "매일 책읽기", completed: false },
    { text: "수영 배우기", completed: false },
  ],
};

// Action Creators
export const loadBucket = (bucket) => {
  return { type: LOAD, bucket };
};

export const createBucket = (bucket) => {
  return { type: CREATE, bucket };
};

export const deleteBucket = (bucket) => {
  return { type: DELETE, bucket };
};

// Reducer
export default function reducer(state = initialState, action) {
  console.log(action);
  switch (action.type) {
    // do reducer stuff
    case "bucket/LOAD":
      return state;

    case "bucket/CREATE":
      console.log(state, action);
      const new_bucket_list = [...state.list, {text: action.bucket, completed: false}];
      return { list: new_bucket_list };

    case "bucket/DELETE":
      const bucket_list = state.list.filter((l, idx) => {
        if(idx !== action.bucket){
          return l;
        }
      });
      return {list: bucket_list};

    default:
      return state;
  }
}

```

▼ [코드스니펫] - App.js

```

import React from "react";

import { withRouter } from "react-router";
import { Route, Switch } from "react-router-dom";

// import [컴포넌트 명] from [컴포넌트가 있는 파일경로];
import BucketList from "./BucketList";
import styled from "styled-components";
import Detail from "./Detail";
import NotFound from "./NotFound";

// 리덕스 스토어와 연결하기 위해 connect라는 친구를 호출할게요!
import {connect} from 'react-redux';
// 리덕스 모듈에서 (bucket 모듈에서) 액션 생성 함수 두개를 가져올게요!

```

```

import {loadBucket, createBucket} from './redux/modules/bucket';

// 이 함수는 스토어가 가진 상태값을 props로 받아오기 위한 함수예요.
const mapStateToProps = (state) => ({
  bucket_list: state.bucket.list,
});

// 이 함수는 값을 변화시키기 위한 액션 생성 함수를 props로 받아오기 위한 함수예요.
const mapDispatchToProps = (dispatch) => ({
  load: () => {
    dispatch(loadBucket());
  },
  create: (new_item) => {
    console.log(new_item);
    dispatch(createBucket(new_item));
  }
});

// 클래스형 컴포넌트는 이렇게 생겼습니다!
class App extends React.Component {
  constructor(props) {
    super(props);
    // App 컴포넌트의 state를 정의해줍니다.
    this.state = {

    };
    // ref는 이렇게 선언합니다!
    this.text = React.createRef();
  }

  componentDidMount() {

  }

  addBucketList = () => {
    const new_item = this.text.current.value;
    this.props.create(new_item);
  };

  // 렌더 함수 안에 리액트 엘리먼트를 넣어줍니다!
  render() {
    return (
      <div className="App">
        <Container>
          <Title>내 버킷리스트</Title>
          <Line />
          {/* 컴포넌트를 넣어줍니다. */}
          {/* <컴포넌트명 [props명]={넘겨줄 것(리스트, 문자열, 숫자, ...)}> */}
          {/* Route 쓰는 법 2가지를 모두 써봅시다! */}
          <Switch>
            <Route path="/" exact component={BucketList} />
            <Route path="/detail/:index" component={Detail} />
            <Route component={NotFound} />
          </Switch>
        </Container>
        {/* 인풋박스와 추가하기 버튼을 넣어줬어요. */}
        <Input>
          <input type="text" ref={this.text} />
          <button onClick={this.addBucketList}>추가하기</button>
        </Input>
      </div>
    );
  }
}

const Input = styled.div`
  max-width: 350px;
  min-height: 10vh;
  background-color: #fff;

```

```

padding: 16px;
margin: 20px auto;
border-radius: 5px;
border: 1px solid #ddd;
`;

const Container = styled.div`
max-width: 350px;
min-height: 60vh;
background-color: #fff;
padding: 16px;
margin: 20px auto;
border-radius: 5px;
border: 1px solid #ddd;
`;

const Title = styled.h1`
color: slateblue;
text-align: center;
`;

const Line = styled.hr`
margin: 16px 0px;
border: 1px dotted #ddd;
`;
// withRouter 적용
// connect로 묶어줬습니다!
export default connect(mapStateToProps, mapDispatchToProps)(withRouter(App));

```

▼ [코드스니펫] - BucketList.js

```

// 리액트 패키지를 불러옵니다.
import React from "react";
import styled from "styled-components";

// redux hook을 불러옵니다.
import {useDispatch, useSelector} from 'react-redux';

const BucketList = (props) => {
  // 버킷리스트를 리덕스 훅으로 가져오기
  const bucket_list = useSelector(state => state.bucket.list);

  console.log(bucket_list);

  return (
    <ListStyle>
      {bucket_list.map((list, index) => {
        return (
          <ItemStyle
            className="list_item"
            key={index}
            onClick={() => {
              // 배열의 몇번째 항목을 눌렀는 지, url 파라미터로 넘겨줍니다.
              props.history.push("/detail/"+index);
            }}
          >
            {list.text}
          </ItemStyle>
        );
      })}
    </ListStyle>
  );
};

const ListStyle = styled.div`

```



```

display: flex;
flex-direction: column;
height: 100%;
overflow-x: hidden;
overflow-y: auto;
`;

const ItemStyle = styled.div`
padding: 16px;
margin: 8px;
background-color: aliceblue;
`;

export default BucketList;

```

▼ [코드스니펫] - Detail.js

```

// 리액트 패키지를 불러옵니다.
import React from "react";

// redux hook을 불러옵니다.
import { useDispatch, useSelector } from "react-redux";
// 내가 만든 액션 생성 함수를 불러옵니다.
import { deleteBucket } from "../redux/modules/bucket";

const Detail = (props) => {
  const dispatch = useDispatch();

  // 스토어에서 상태값 가져오기
  const bucket_list = useSelector((state) => state.bucket.list);
  // url 파라미터에서 인덱스 가져오기
  let bucket_index = parseInt(props.match.params.index);

  console.log(props);
  return (
    <div>
      <h1>{bucket_list[bucket_index].text}</h1>
      <button onClick={() => {
        // dispatch(); <- 괄호안에는 액션 생성 함수가 들어가야겠죠?
        // 예를 들면 이렇게요.
        dispatch(deleteBucket(bucket_index));
        props.history.goBack();
      }}>삭제하기</button>
    </div>
  );
};

export default Detail;

```

- [완료하기] 버튼 추가



완료하기 버튼을 달고, 버킷 모듈에도 완료 여부를 바꿔주도록 넣어봅시다!

- (1) 뷰를 먼저 만들어요! (버튼 먼저 만들기)
- (2) 액션 먼저 만들기
- (3) 액션 생성 함수 만들고,

- (4) 리듀서까지 만든다.
- (5) 이제 버튼을 누르면 액션을 호출하게 해볼까요?

▼ [코드스니펫] - Detail.js

```
// 리액트 패키지를 불러옵니다.
import React from "react";

// redux hook을 불러옵니다.
import { useDispatch, useSelector } from "react-redux";
// 내가 만든 액션 생성 함수를 불러옵니다.
import { deleteBucket } from "../redux/modules/bucket";

const Detail = (props) => {
  const dispatch = useDispatch();

  // 스토어에서 상태값 가져오기
  const bucket_list = useSelector((state) => state.bucket.list);
  // url 파라미터에서 인덱스 가져오기
  let bucket_index = parseInt(props.match.params.index);

  console.log(props);
  return (
    <div>
      <h1>{bucket_list[bucket_index].text}</h1>
      <button onClick={() => {
        // dispatch(); <- 괄호안에는 액션 생성 함수가 들어가야겠죠?
        // 예를 들면 이렇게요.
        dispatch(deleteBucket(bucket_index));
        props.history.goBack();
      }}>삭제하기</button>
    </div>
  );
};

export default Detail;
```

▼ [코드스니펫] - bucket.js

```
// Actions
const LOAD = "bucket/LOAD";
const CREATE = "bucket/CREATE";
const DELETE = "bucket/DELETE";
const UPDATE = "bucket/UPDATE";

const initialState = {
  // list: ["영화관 가기", "매일 책읽기", "수영 배우기"],
  list: [
    { text: "영화관 가기", completed: false },
    { text: "매일 책읽기", completed: false },
    { text: "수영 배우기", completed: false },
  ],
};

// Action Creators
export const loadBucket = (bucket) => {
  return { type: LOAD, bucket };
};

export const createBucket = (bucket) => {
  return { type: CREATE, bucket };
};
```

```

};

export const deleteBucket = (bucket) => {
  return { type: DELETE, bucket };
};

export const updateBucket = (bucket) => {
  return { type: UPDATE, bucket };
};

// Reducer
export default function reducer(state = initialState, action) {

  switch (action.type) {
    // do reducer stuff
    case "bucket/LOAD": {
      return state;
    }

    case "bucket/CREATE": {

      const new_bucket_list = [...state.list, action.bucket];
      return { list: new_bucket_list };
    }

    case "bucket/DELETE": {
      const bucket_list = state.list.filter((l, idx) => {
        if (idx !== action.bucket) {
          return l;
        }
      });
      return { list: bucket_list };
    }

    case "bucket/UPDATE": {
      const bucket_list = state.list.map((l, idx) => {
        if (idx === action.bucket) {

          return { ...l, completed: true };
        }

        return l;
      });
      return { list: bucket_list };
    }

    default:
      return state;
  }
}

```

- 완료하면 색을 바꿔주자!



아앗, 완료하기 버튼까지 달았는데, 티가 잘 안나네요.
완료한 리스트는 오렌지 색으로 바꿔봅시다.

- (1) styled-components에서 변수 받아오는 법! 기억하시죠?!

```

const ItemStyle = styled.div`
  padding: 16px;

```

```
margin: 8px;
background-color: ${props => props.color};
`;
```

▼ [코드스니펫] - BucketList.js

```
// 리액트 패키지를 불러옵니다.
import React from "react";
import styled from "styled-components";

// redux hook을 불러옵니다.
import {useDispatch, useSelector} from 'react-redux';

const BucketList = (props) => {
  // 버킷리스트를 리덕스 훅으로 가져오기
  const bucket_list = useSelector(state => state.bucket.list);

  return (
    <ListStyle>
      {bucket_list.map((list, index) => {
        return (
          <ItemStyle
            className="list_item"
            color={list.completed? "orange":"aliceblue"}
            key={index}
            onClick={() => {
              // 배열의 몇번째 항목을 눌렀는 지, url 파라미터로 넘겨줍니다.
              props.history.push("/detail/"+index);
            }}
          >
            {list.text}
          </ItemStyle>
        );
      })}
    </ListStyle>
  );
};

const ListStyle = styled.div`
display: flex;
flex-direction: column;
height: 100%;
overflow-x: hidden;
overflow-y: auto;
`;

const ItemStyle = styled.div`
padding: 16px;
margin: 8px;
background-color: ${props => props.color};
`;

export default BucketList;
```

- 이제 Progress 컴포넌트를 만들게요 (거의 다 왔어요!)

▼ App.js

```
import React from "react";

import { withRouter } from "react-router";
import { Route, Switch } from "react-router-dom";
```

```

// import [컴포넌트 명] from [컴포넌트가 있는 파일경로];
import BucketList from "./BucketList";
import styled from "styled-components";
import Detail from "./Detail";
import NotFound from "./NotFound";

// 리덕스 스토어와 연결하기 위해 connect라는 친구를 호출할게요!
import {connect} from 'react-redux';
// 리덕스 모듈에서 (bucket 모듈에서) 액션 생성 함수 두개를 가져올게요!
import {loadBucket, createBucket} from './redux/modules/bucket';
import Progress from "./Progress";

// 이 함수는 스토어가 가진 상태값을 props로 받아오기 위한 함수예요.
const mapStateToProps = (state) => ({
  bucket_list: state.bucket.list,
});

// 이 함수는 값을 변화시키기 위한 액션 생성 함수를 props로 받아오기 위한 함수예요.
const mapDispatchToProps = (dispatch) => ({
  load: () => {
    dispatch(loadBucket());
  },
  create: (new_item) => {
    console.log(new_item);
    dispatch(createBucket(new_item));
  }
});

// 클래스형 컴포넌트는 이렇게 생겼습니다!
class App extends React.Component {
  constructor(props) {
    super(props);
    // App 컴포넌트의 state를 정의해줍니다.
    this.state = {

    };
    // ref는 이렇게 선언합니다!
    this.text = React.createRef();
  }

  componentDidMount() {
  }

  addBucketList = () => {
    const new_item = { text: this.text.current.value, completed: false};
    this.props.create(new_item);
  };

  // 랜더 함수 안에 리액트 엘리먼트를 넣어줍니다!
  render() {
    return (
      <div className="App">
        <Container>
          <Title>내 버킷리스트</Title>
          <Progress/>
          <Line />
          {/* 컴포넌트를 넣어줍니다. */}
          {/* <컴포넌트 명 [props 명]={넘겨줄 것(리스트, 문자열, 숫자, ...)}> */}
          {/* Route 쓰는 법 2가지를 모두 써봅시다! */}
          <Switch>
            <Route path="/" exact component={BucketList} />
            <Route path="/detail/:index" component={Detail} />
            <Route component={NotFound} />
          </Switch>
        </Container>
        {/* 인풋박스와 추가하기 버튼을 넣어줬어요. */}
        <Input>

```

```

        <input type="text" ref={this.text} />
        <button onClick={this.addBucketList}>추가하기</button>
      </Input>
    </div>
  );
}
}

const Input = styled.div`
  max-width: 350px;
  min-height: 10vh;
  background-color: #fff;
  padding: 16px;
  margin: 20px auto;
  border-radius: 5px;
  border: 1px solid #ddd;
`;

const Container = styled.div`
  max-width: 350px;
  min-height: 60vh;
  background-color: #fff;
  padding: 16px;
  margin: 20px auto;
  border-radius: 5px;
  border: 1px solid #ddd;
`;

const Title = styled.h1`
  color: slateblue;
  text-align: center;
`;

const Line = styled.hr`
  margin: 16px 0px;
  border: 1px dotted #ddd;
`;
// withRouter 적용
// connect로 묶어줬습니다!
export default connect(mapStateToProps, mapDispatchToProps)(withRouter(App));

```

▼ Progress.js

```

// 리액트 패키지를 불러옵니다.
import React from "react";
import styled from "styled-components";

// redux hook을 불러옵니다.
import { useDispatch, useSelector } from "react-redux";
// 내가 만든 액션 생성 함수를 불러옵니다.
import { deleteBucket, updateBucket } from "../redux/modules/bucket";

const Progress = (props) => {
  // 스토어에서 상태값 가져오기
  const bucket_list = useSelector((state) => state.bucket.list);

  let count = 0;

  let goal_per = bucket_list.map((l, idx) => {
    if(l.completed){
      count++;
    }
  });

  return (

```

```

    <ProgressBar>
      <HighLight width={count / bucket_list.length} * 100 + "%"}></HighLight>
    </ProgressBar>
  );
};

const ProgressBar = styled.div`
  background: #eee;
  width: 100%;
  height: 40px;
`;

const HighLight = styled.div`
  background: orange;
  width: ${props => props.width};
  height: 40px;
`;
export default Progress;

```

03. 스크롤바 움직이기

▼ 5) 스크롤 바를 움직여보자

👉 window.scrollTo() → 이 녀석 어디선가 본 적 없으신가요? 😊

- 위로 가기 버튼을 만들어볼게요!

▼ App.js

```

...
<button onClick={() => {
  window.scrollTo(0,0);
}}>위로가기</button>
...

```

- 리스트를 마구마구 늘리고, 위로가기 버튼을 눌러보세요!
- 앓, 스크롤이 아니라 획 가버렸죠?
- 이번엔 파라미터를 다르게 줘볼게요!

▼ App.js

```

...
button onClick={() => {
  window.scrollTo({ top: 0, left: 0, behavior: "smooth" });
}}>위로가기</button>
..

```

- 이번엔 스크롤 잘 움직이나요?

- 위로 가기 버튼이나, 특정 위치로 스크롤 하는 건 자주 쓰이는 트릭이에요.
- 좌표를 이용해서 이리 저리 움직일 수 있고, **ref**를 통해 특정 엘리먼트 위치를 받아다가 스크롤 시킬 수도 있습니다!

04. Quiz_버킷리스트 좀 더 예쁘게!

▼ 6) 🗒 버킷리스트를 조금 더 예쁘게 꾸며보자



버킷리스트 스타일을 예쁘게 만져봅시다!

꼭 해봐야하는 것 외에도 더 원하는 애니메이션 효과가 있다면 마음껏 넣어보세요. 😊

▼ Q. 퀴즈설명

▼ 모습 보기

내 버킷리스트

영화관 가기

매일 책읽기

수영 배우기



꼭 해보기!(키워드를 가지고 검색해서 아래 항목은 꼭꼭 스스로 찾아 해봅시다!) :

- 1) 버킷리스트 아이템을 추가하면, 리스트 항목이 있는 div에만 스크롤 생기게 하기
 - 키워드: overflow, max-height 혹은, div 넘치지 않게 하기
- 2) 프로그래스 바에 동그라미 달아보기
 - 키워드: flex, align-items, div 겹치려면?
- 3) input focus일 때 border 색상 바꿔보기
 - 키워드: input text focus border 색상 바꾸기

▼ A. 함께하기



어때요, 할만했나요? 재미있었죠!

이제 제가 어떻게 메인 화면을 꾸렸는 지 한번 함께 봅시다.

저와 다른 방식을 쓰셨어도 굳! 잘하셨습니다. 원하는 효과만 줄 수 있으면 됩니다! 😊

▼ [코드스니펫] - App.js

```
import React from "react";

import { withRouter } from "react-router";
import { Route, Switch } from "react-router-dom";

// import [컴포넌트 명] from [컴포넌트가 있는 파일경로];
import BucketList from "./BucketList";
import styled from "styled-components";
import Detail from "./Detail";
import NotFound from "./NotFound";

// 리덕스 스토어와 연결하기 위해 connect라는 친구를 호출할게요!
import { connect } from "react-redux";
// 리덕스 모듈에서 (bucket 모듈에서) 액션 생성 함수 두개를 가져올게요!
import { loadBucket, createBucket } from "../redux/modules/bucket";
import Progress from "./Progress";

// 이 함수는 스토어가 가진 상태값을 props로 받아오기 위한 함수예요.
const mapStateToProps = (state) => ({
  bucket_list: state.bucket.list,
});

// 이 함수는 값을 변화시키기 위한 액션 생성 함수를 props로 받아오기 위한 함수예요.
const mapDispatchToProps = (dispatch) => ({
  load: () => {
    dispatch(loadBucket());
  },
  create: (new_item) => {
    console.log(new_item);
    dispatch(createBucket(new_item));
  }
});

// 클래스형 컴포넌트는 이렇게 생성됩니다!
class App extends React.Component {
  constructor(props) {
    super(props);
    // App 컴포넌트의 state를 정의해줍니다.
```

```

    this.state = {

    };
    // ref는 이렇게 선언합니다!
    this.text = React.createRef();
  }

  componentDidMount() {
  }

  addBucketList = () => {
    const new_item = { text: this.text.current.value, completed: false };
    this.props.create(new_item);
  };

  // 렌더 함수 안에 리액트 엘리먼트를 넣어줍니다!
  render() {
    return (
      <div className="App">
        <Container>
          <Title>내 버킷리스트</Title>
          <Progress/>
          <Line />
          { /* 컴포넌트를 넣어줍니다. */ }
          { /* <컴포넌트 명 [props 명]={넘겨줄 것(리스트, 문자열, 숫자, ...)}> */ }
          { /* Route 쓰는 법 2가지를 모두 써봅시다! */ }
          <Switch>
            <Route path="/" exact component={BucketList} />
            <Route path="/detail/:index" component={Detail} />
            <Route component={NotFound} />
          </Switch>
        </Container>
        { /* 인풋박스와 추가하기 버튼을 넣어줬어요. */ }
        <Input>
          <input type="text" ref={this.text} />
          <button onClick={this.addBucketList}>추가하기</button>
        </Input>

        <button onClick={() => {
          window.scrollTo({ top: 0, left: 0, behavior: "smooth" });
        }}>위로가기</button>
      </div>
    );
  }
}

const Input = styled.div`
  max-width: 350px;
  min-height: 10vh;
  background-color: #fff;
  padding: 16px;
  margin: 20px auto;
  border-radius: 5px;
  border: 1px solid #ddd;
  display: flex;
  align-items: center;
  justify-content: space-between;
  & > * {
    padding: 5px;
  }

  & input {
    border-radius: 5px;
    margin-right: 10px;
    border: 1px solid #888;
    width: 70%;
    &:focus {
      border: 1px solid #a673ff;
    }
  }

```

```

    }
  }

  & button {
    width: 25%;
    color: #fff;
    border: 1px solid #a673ff;
    background-color: #a673ff;
  }
`;

const Container = styled.div`
  max-width: 350px;
  min-height: 60vh;
  background-color: #fff;
  padding: 16px;
  margin: 20px auto;
  border-radius: 5px;
  border: 1px solid #ddd;
`;

const Title = styled.h1`
  color: #673ab7;
  text-align: center;
`;

const Line = styled.hr`
  margin: 16px 0px;
  border: 1px dotted #ddd;
`;
// withRouter 적용
// connect로 묶어줬습니다!
export default connect(mapStateToProps, mapDispatchToProps)(withRouter(App));

```

▼ [코드스니펫] - BucketList.js

```

// 리액트 패키지를 불러옵니다.
import React from "react";
import styled from "styled-components";

// redux hook을 불러옵니다.
import {useDispatch, useSelector} from 'react-redux';

const BucketList = (props) => {
  // 버킷리스트를 리덕스 혹은로 가져오기
  const bucket_list = useSelector(state => state.bucket.list);

  return (
    <ListStyle>
      {bucket_list.map((list, index) => {
        return (
          <ItemStyle
            className="list_item"
            completed = {list.completed}
            key={index}
            onClick={() => {
              // 배열의 몇번째 항목을 눌렀는 지, url 파라미터로 넘겨줍니다.
              props.history.push("/detail/" + index);
            }}
          >
            {list.text}
          </ItemStyle>
        );
      })}
    </ListStyle>
  );
}

```

```

    </ListStyle>
  );
};

const ListStyle = styled.div`
  display: flex;
  flex-direction: column;
  height: 50vh;
  overflow-x: hidden;
  overflow-y: auto;
  max-height: 50vh;
`;

const ItemStyle = styled.div`
  padding: 16px;
  margin: 8px;
  font-weight: 600;
  color: ${props} => (props.completed ? "#fff" : "#212121");
  background-color: ${props} => (props.completed ? "#673ab7" : "#eee");
  box-sizing: border-box;
`;

export default BucketList;

```

▼ [코드스니펫] - Progress.js

```

// 리액트 패키지를 불러옵니다.
import React from "react";
import styled from "styled-components";

// redux hook을 불러옵니다.
import { useDispatch, useSelector } from "react-redux";
// 내가 만든 액션 생성 함수를 불러옵니다.
import { deleteBucket, updateBucket } from "../redux/modules/bucket";

const Progress = (props) => {
  // 스토어에서 상태값 가져오기
  const bucket_list = useSelector((state) => state.bucket.list);

  let count = 0;

  let goal_per = bucket_list.map((l, idx) => {
    if(l.completed){
      count++;
    }
  });

  return (
    <ProgressBar>
      <HighLight width={count / bucket_list.length * 100 + "%"}></HighLight>
      <Dot/>
    </ProgressBar>
  );
};

const ProgressBar = styled.div`
  background: #eee;
  width: 100%;
  height: 20px;
  display: flex;
  border-radius: 10px;
  align-items: center;
`;

const HighLight = styled.div`

```

```

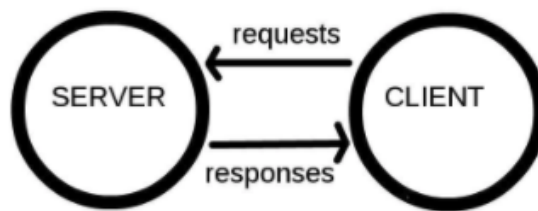
background: #673ab7;
width: ${props => props.width};
height: 20px;
border-radius: 10px;
transition: 2s;
`;

const Dot = styled.div`
background: #fff;
border: 5px solid #673ab7;
box-sizing: border-box;
margin: 0px 0px 0px -10px;
width: 40px;
height: 40px;
border-radius: 20px;
`;
export default Progress;

```

05. 서버와 서버리스

▼ 7) 웹의 동작방식



웹은 **요청과 응답**으로 굴러갑니다!

클라이언트가 서버에게 요청, 서버가 클라이언트에게 응답!

▼ 8) 서버가 하는 일



서버가 하는 일은 엄청 많아요.

우리가 알고 있는 것처럼 데이터도 관리하고, 분산처리도 하고, 웹 어플리케이션도 돌려야 하고...

서버가 할 일이 많다는 건,

서버가 하는 일을 우리가 전부 관리해줘야한다는 이야기이기도 해요. 😓

▼ 9) 서버리스가 뭔데?



서버리스란?

1주차에 잠깐 말씀드렸던 것처럼 **서버리스**는 **서버가 없다**가 아니라, **서버를 신경쓸 필요 없다**입니다.

이미 누군가가 구축해둔 서버의 일부분을 빌려서 쓸 수 있어요.

우리가 인프라를 구축하고, 서버 스펙을 고민할 필요 없다는 소리죠!

우린 그냥, **우리한테 필요한 서버를 필요한만큼만 빌려 쓰면 되니까요.**

06. Firebase란?

▼ 10) BaaS는 뭘 빌려오는 걸까?



BaaS는 Backend as a Service의 약자입니다.

→ 우리가 흔히 백엔드 하면 떠올리는 것들을 빌려오는거예요.

우리가 쓸 파이어베이스를 예로 들면,

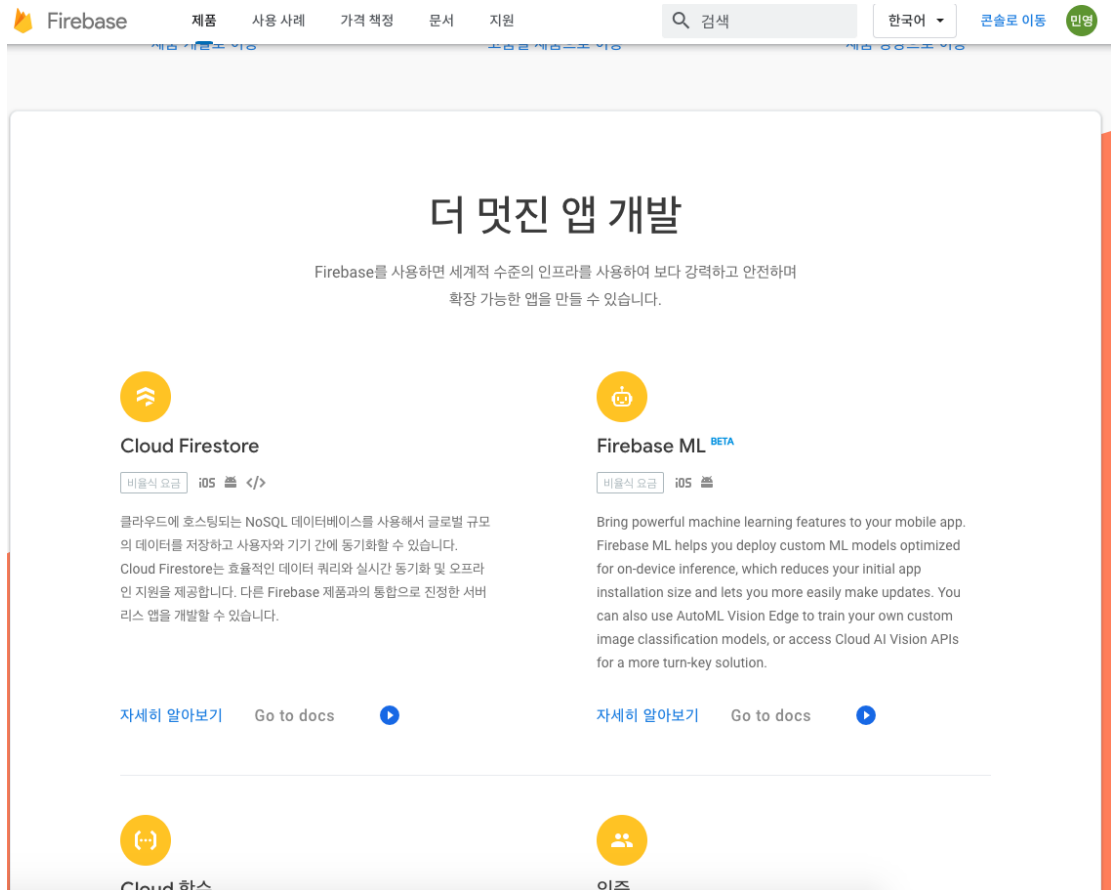
데이터 베이스, 소셜 서비스 연동(일종의 소셜 로그인), 파일시스템 등을 API 형태로 제공해 줍니다!

▼ 11) Firebase 간단 소개



사이트로 가볼까요? ([링크](#))

- 먼저 제품 탭으로 가볼까요?



벌써 뭐가 많죠! 머신러닝에, 인증에, 호스팅에 ...

파이어베이스는 굉장히 많은 기능을 제공하고 있습니다.



우리는 이중에서 **Firestore**를 사용할거예요!

Hosting(서버 없이 웹 서비스를 배포할 수 있도록 도와 주는 서비스)도 한번 써볼거예요

- firestore는 일명 realtime-database를 제공하는 서비스입니다.
- 실시간으로 데이터가 추가되고, 삭제되는 걸 우리 클라이언트에 알려줘요.
- 리액트와는 react-firebase라는 패키지를 통해 편리하게 연결할 수 있어요!
- 주의! 우리는 **spark** 요금제를 쓸거예요!(무료입니다!)

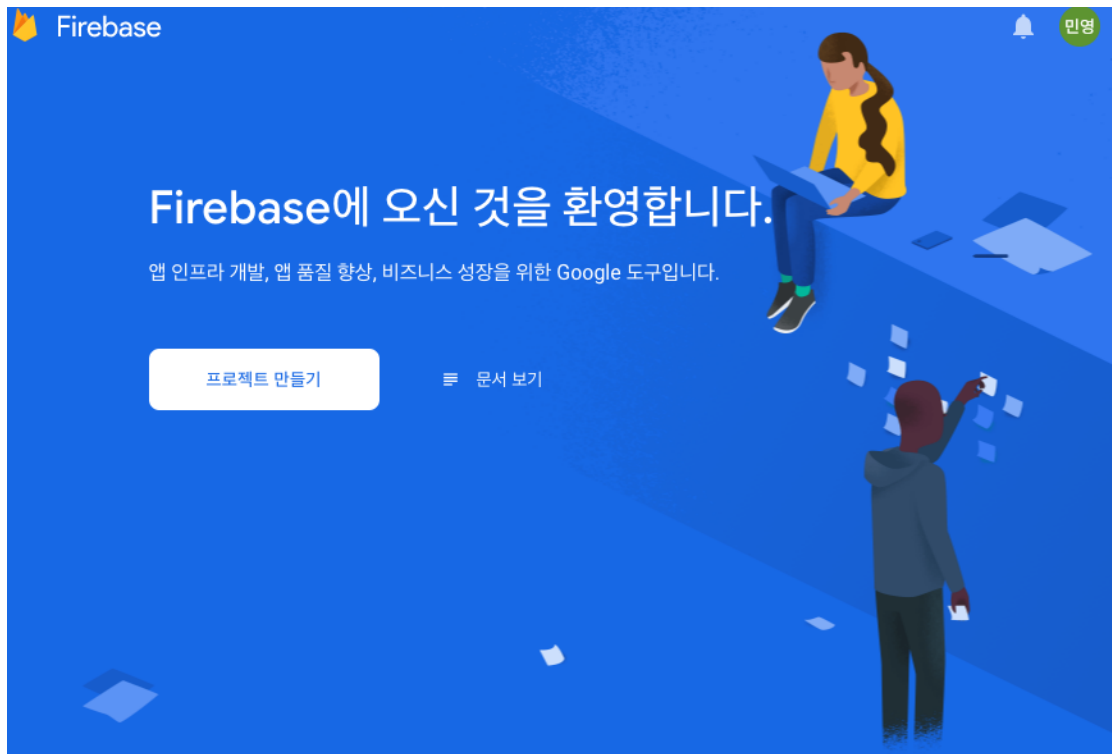
07. Firebase 설정하기

▼ 12) 파이어베이스 환경 설정하기



차근차근 하나씩 설정해봅시다!

▼ (1) 사이트에서 프로젝트 만들기 버튼 클릭



▼ (2) 프로젝트 이름을 정하고 약관에 동의해줍니다.

× 프로젝트 만들기(1/3단계)

프로젝트[?] 이름을 지정하여 시작하기

프로젝트 이름

sparta-react

 sparta-react

☐ [Firebase 약관](#)에 동의합니다.


계속


▼ (3) 프로젝트 애널리틱스 설정을 해줍니다.


Firestore 프로젝트를 위한 Google 애널리틱스


무제한 무료 애널리틱스 솔루션인 Google 애널리틱스를 사용하면 Firebase Crashlytics, 클라우드 메시징, 인앱 메시지, 원격 구성, A/B 테스트, 예측, Cloud Functions에서 타겟팅, 보고 등을 이용할 수 있습니다.


Google 애널리틱스를 통해 다음 기능을 이용할 수 있습니다.


 A/B 테스트 [?](#)

 장애가 발생하지 않은 사용자 [?](#)

 Firebase 제품 전반에서 사용자 세분화 및 타겟팅 [?](#)

 이벤트 기반 Cloud Functions 트리거 [?](#)

 사용자 행동 예측 [?](#)

 제한 없는 무료 보고 [?](#)

☒ 이 프로젝트에서 Google 애널리틱스 사용 설정 권장

[이전](#)

[계속](#)

▼ (4) 나라 선택 → 마음에 드는 나라로 선택하셔도 됩니다! → 프로젝트 만들기 버튼을 눌러주세요.



08. Firestore 설정하기

▼ 13) 파이어스토어란?

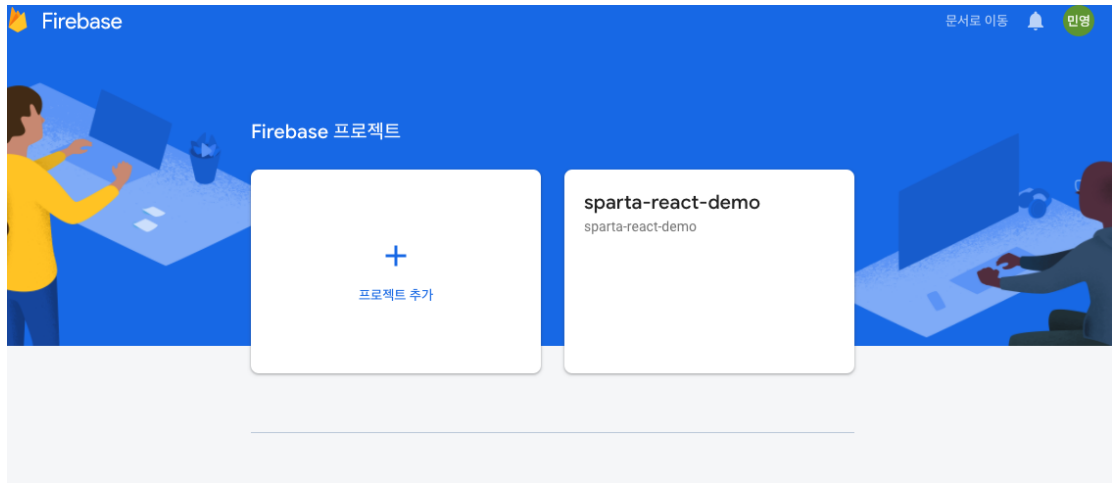


Firestore에 포함되어 있는 서비스 중 하나로
유연하고 확장 가능한 NoSQL 클라우드 데이터베이스입니다!

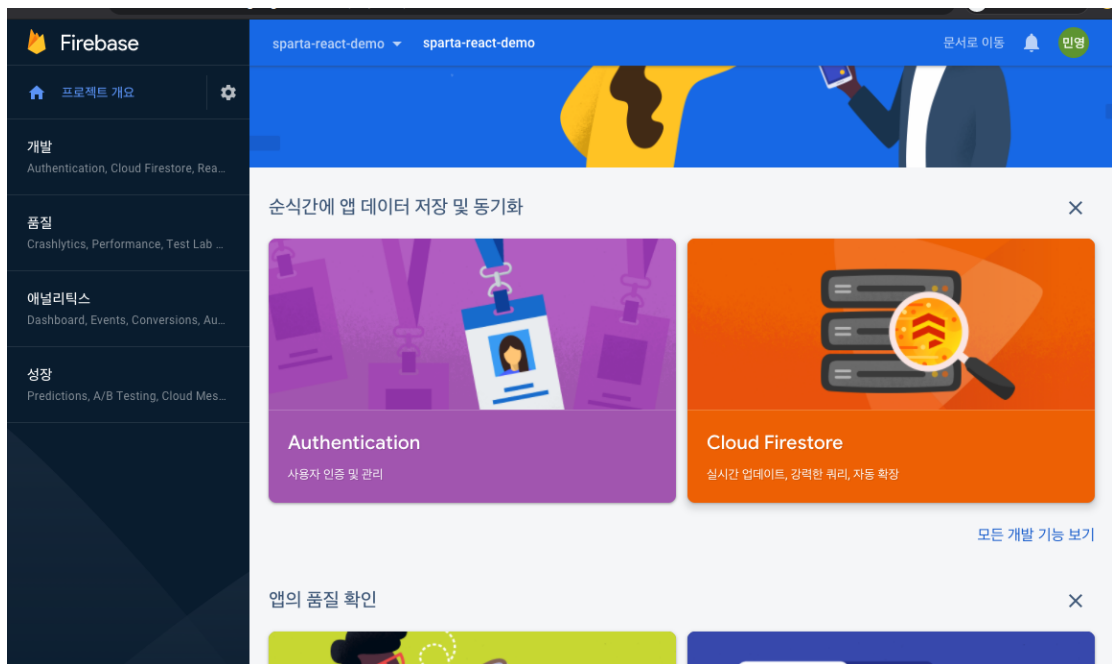
- 구조:
 - 1. Collection: 문서(다큐먼트)의 집합
 - 2. Document: JSON 형식으로 데이터를 저장할 수 있음

▼ 14) 파이어스토어 설정하기

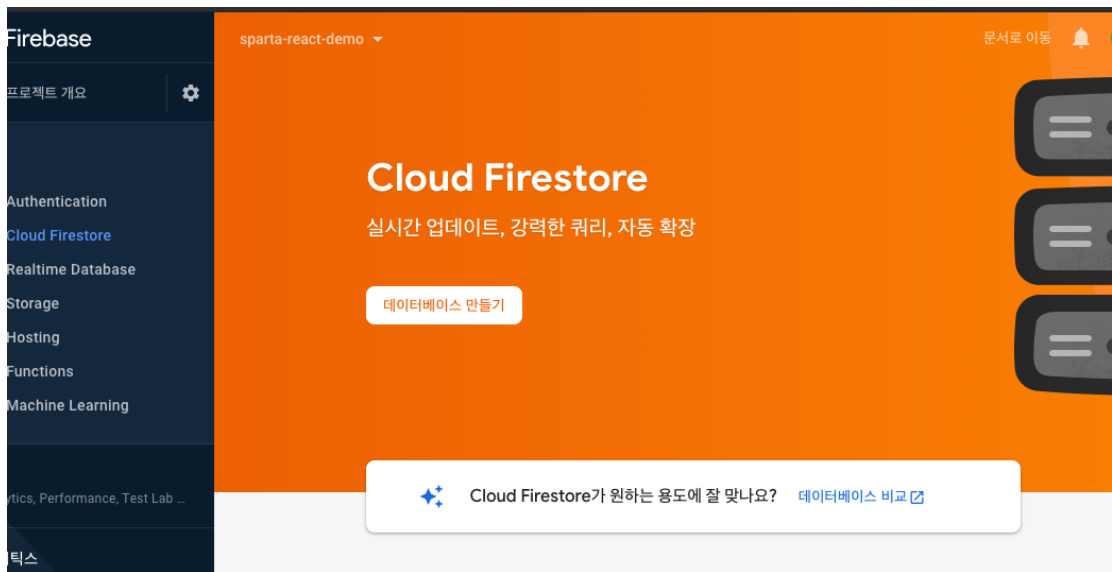
▼ (1) 생성된 프로젝트 클릭



▼ (2) Cloud Firestore 추가

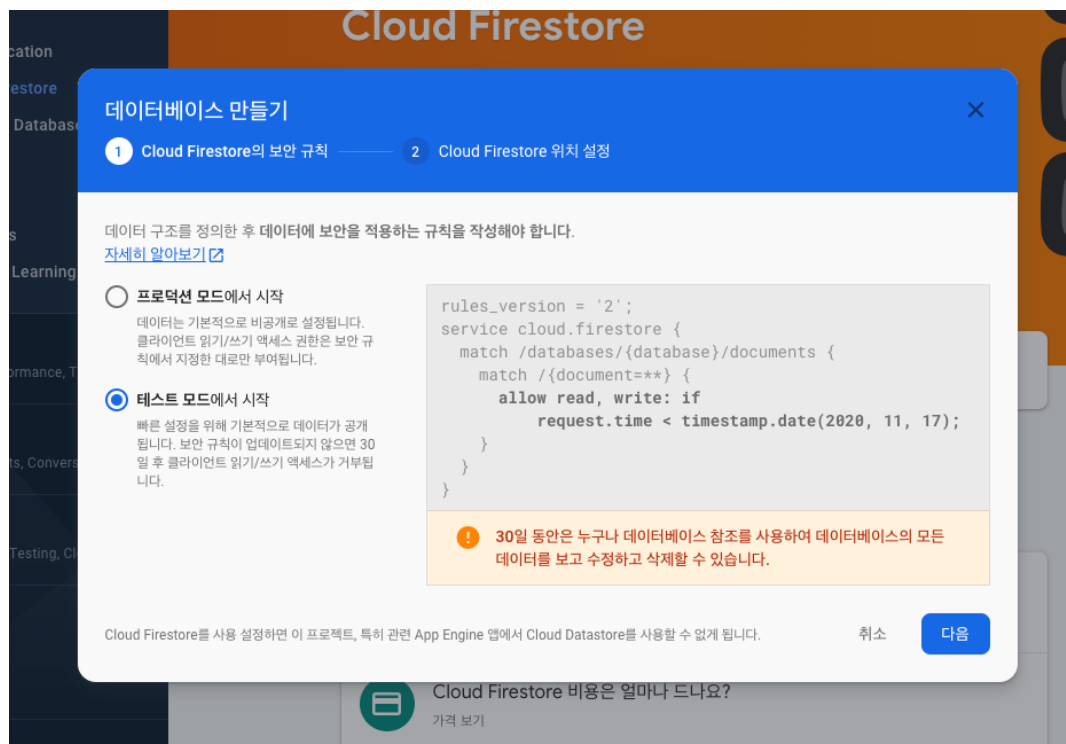


▼ (3) 데이터베이스 만들기 클릭

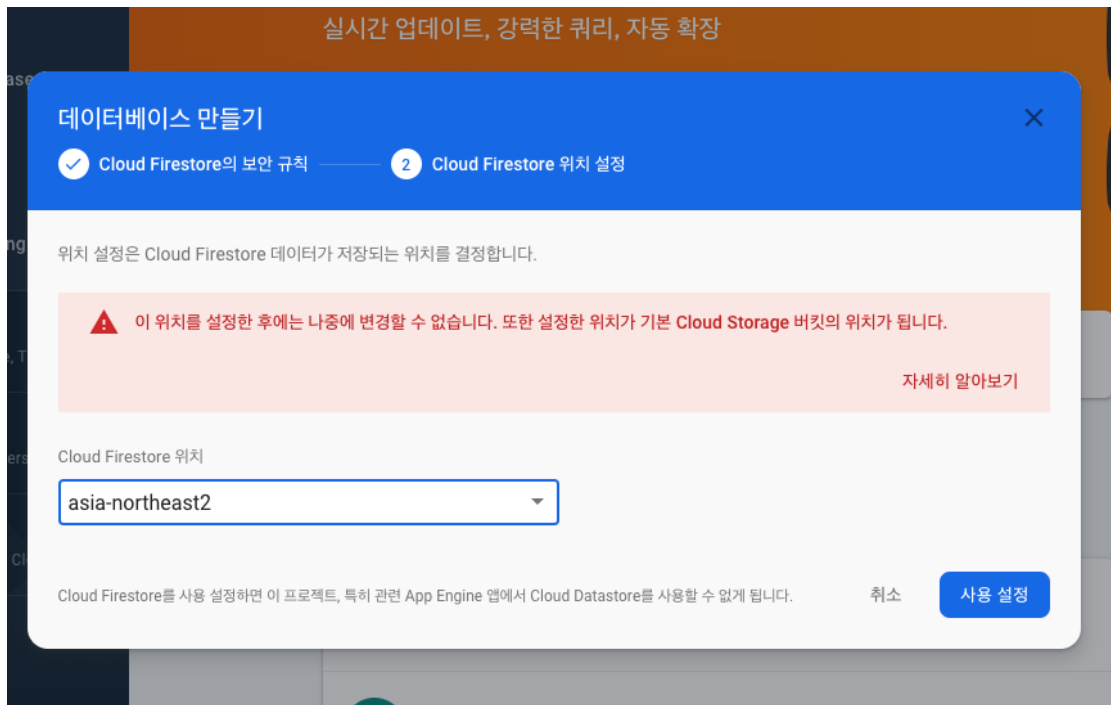


▼ (4) 보안규칙 설정

- test 모드로 하셔야 localhost에서 firestore로 데이터 요청이 가능해요!

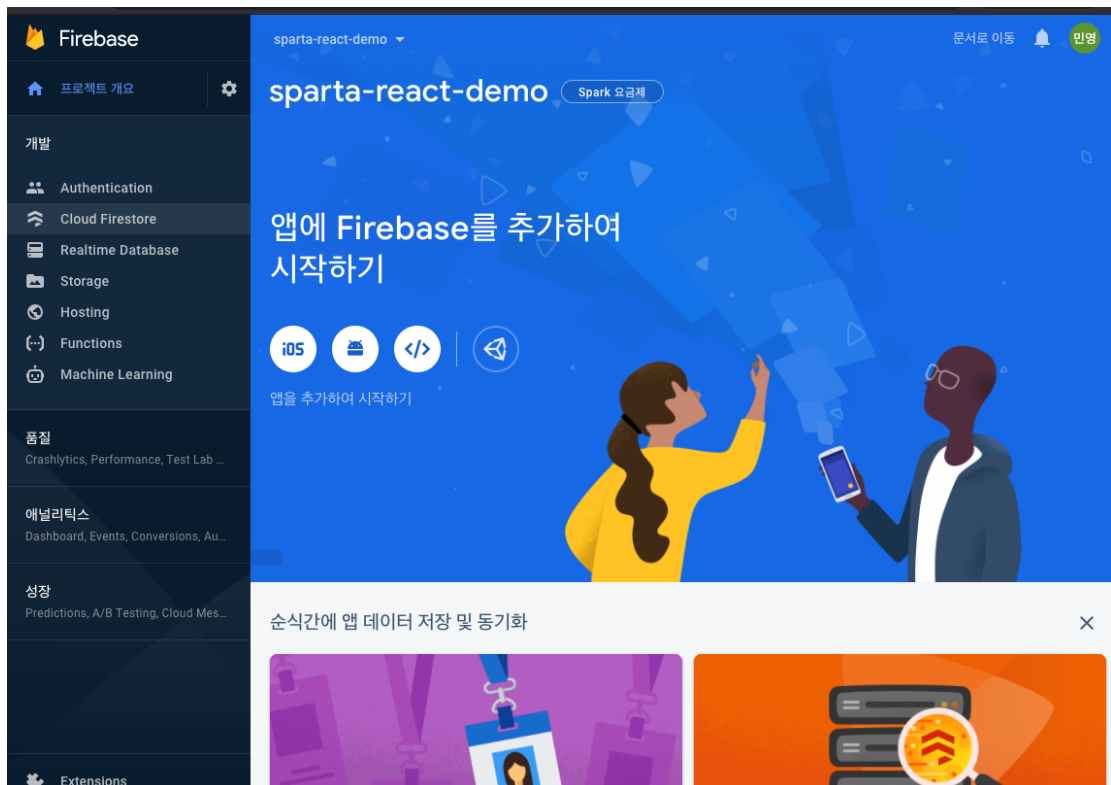


▼ (5) Cloud Firestore 위치 설정

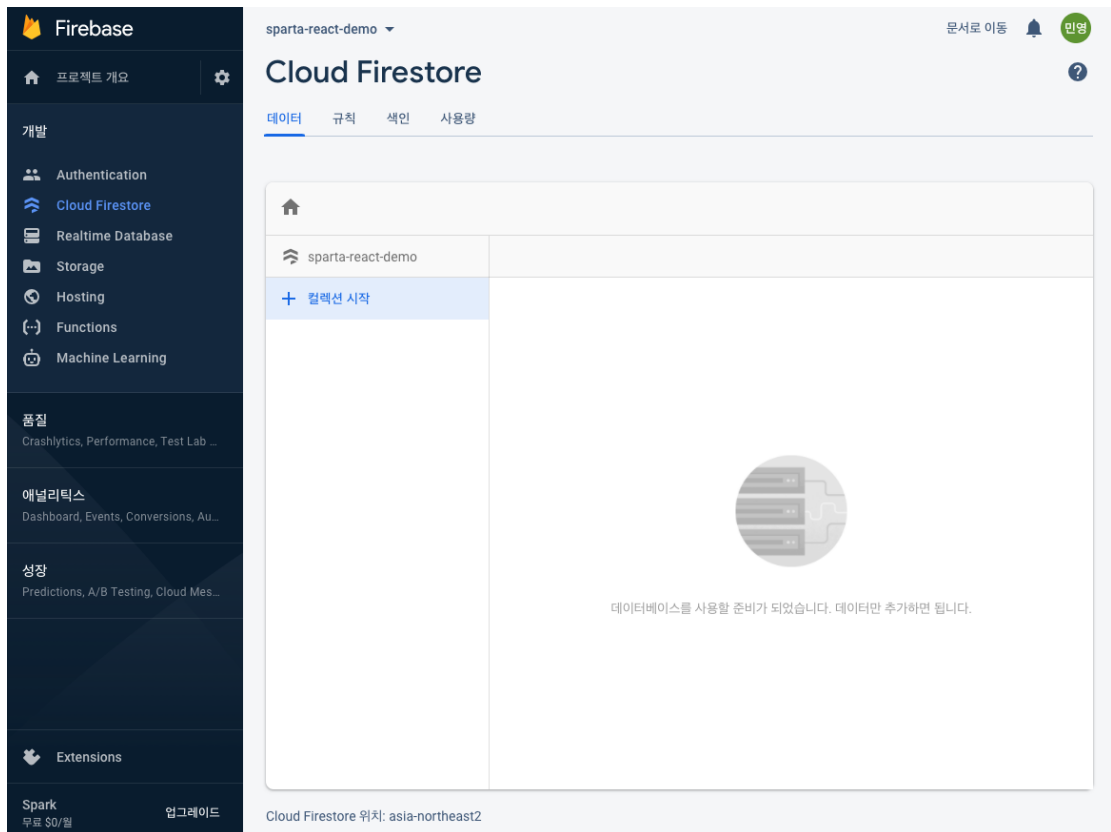


▼ 15) 대시보드에서 파이어스토어 데이터 넣기

▼ (1) 콘솔에서 내 프로젝트 → firestore 선택하여 대시보드로 진입



▼ (2) 대시보드에서 컬렉션 시작 버튼 클릭



▼ (3) 컬렉션을 만든다

컬렉션 시작

✓ 컬렉션 ID 지정

2 첫 번째 문서 추가

문서 상위 경로

/bucket

문서 ID

자동 ID

필수 항목입니다.

필드

=

boolean

값

false

+

취소

저장

▼ (4) 문서 ID를 지정하고 필드 값을 넣는다

컬렉션 시작

✓ 컬렉션 ID 지정

2 첫 번째 문서 추가

문서 상위 경로 ②

/bucket

문서 ID ②

bucket_item

필드

유형

값

text

=

string

⊖

completed

=

boolean

false

⊖

+ 필드 추가

취소

저장

09. 리액트에 Firebase 연동하기

▼ 16) 파이어베이스 패키지 설치



버킷 리스트 프로젝트에서 해봅시다!

```
yarn add firebase
```

▼ 17) config 가져오기

▼ (1) src 폴더 하위에 firebase.js 파일을 만들어주세요

```
//firebase.js
import firebase from "firebase/app";
import "firebase/firestore";

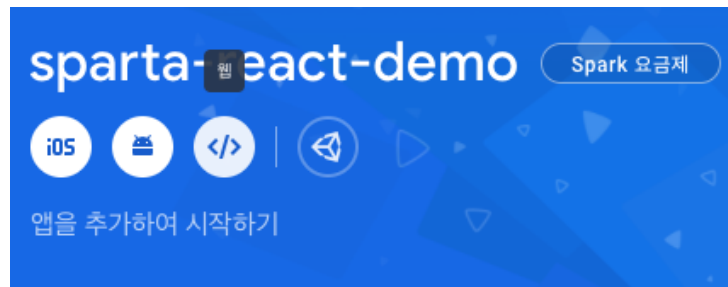
const firebaseConfig = {
  // firebase 설정과 관련된 개인 정보
};
```

```
// firebaseConfig 정보로 firebase 시작
firebase.initializeApp(firebaseConfig);

// firebase의 firestore 인스턴스를 변수에 저장
const firestore = firebase.firestore();

// 필요한 곳에서 사용할 수 있도록 내보내기
export { firestore };
```

▼ (2) firebase 대시보드에서 **웹**버튼을 눌러주세요.



▼ (3) 앱 이름을 적고, 잠시 기다리면 **sdk 추가영역**에 텍스트가 뜹니다.

- **firebaseConfig** 내용만 위의 **firebse.js**에 붙여넣어주세요!

× 웹 앱에 Firebase 추가

- ✓ 앱 등록
- 2 Firebase SDK 추가

스크립트를 복사하여 <body> 태그 하단에 붙여넣으세요. Firebase 서비스를 사용하기 전에 진행해야 합니다.

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/7.24.0/firebase-app.js"></scrip

<!-- TODO: Add SDKs for Firebase products that you want to use
https://firebase.google.com/docs/web/setup#available-libraries -->
<script src="https://www.gstatic.com/firebasejs/7.24.0/firebase-analytics.js"><

<script>
  // Your web app's Firebase configuration
  // For Firebase JS SDK v7.20.0 and later, measurementId is optional
  var firebaseConfig = {
    apiKey: "AIzaSyA9j51kHLPp9R0sqfFJa0ZnsLidX1BuSLY",
    authDomain: "sparta-react-demo.firebaseio.com",
    databaseURL: "https://sparta-react-demo.firebaseio.com",
    projectId: "sparta-react-demo",
    storageBucket: "sparta-react-demo.appspot.com",
    messagingSenderId: "63477404588",
    appId: "1:63477404588:web:dbf4d5b8dfb30ae1556009",
    measurementId: "G-E77R7639PC"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
  firebase.analytics();
</script>
```

웹용 Firebase 자세히 알아보기: [시작하기](#), [Web SDK API 참조](#), [샘플](#)

▼ (4) App.js 에서 firebase.js에서 내보낸 firestore 가져오기

```
import { firestore } from "./firebase";
```

▼ (5) componentDidMount에서 데이터 불러와보기

```
...
const bucket = firestore.collection("buckets");
// 하나만 확인하기
bucket
  .doc("bucket_item")
  .get()
  .then((doc) => {
    // .exists를 써서 데이터가 있는 지 없는 지 확인!
    if(doc.exists){
      // 데이터를 콘솔에 찍어보자!
      console.log(doc.data());
    }
  });
//전체 확인하기
bucket
  .get()
  .then((docs) => {
```

```

let bucket_data = [];
docs.forEach((doc) => {
  // 문서 객체를 확인해보자!
  console.log(doc);
  // 문서 데이터 가져오기
  console.log(doc.data());
  // 문서 id 가져오기
  console.log(doc.id);

  if (doc.exists) {
    bucket_data = [...bucket_data, { id: doc.id, ...doc.data() }];
  }
});

console.log(bucket_data);

...

```

10. Firestore 데이터 가지고 놀기



공식문서를 보면서 해봅시다!

▼ 18) app.js에서 firestore 데이터 가지고 놀기

▼ (1) 데이터 추가하기



컬렉션을 찾고 → add!
대시보드에서 잘 추가되었는지 확인해봅시다!

```
bucket.add({ text: "수영 배우기", completed: false });
```

▼ (2) 데이터 수정하기



컬렉션을 찾고 → 문서 id로 set!

```
bucket.doc("bucket_item").update({ text: "수영 배우기", completed: false });
```

▼ (3) 데이터 삭제하기



컬렉션을 찾고 → 문서 id로 delete!

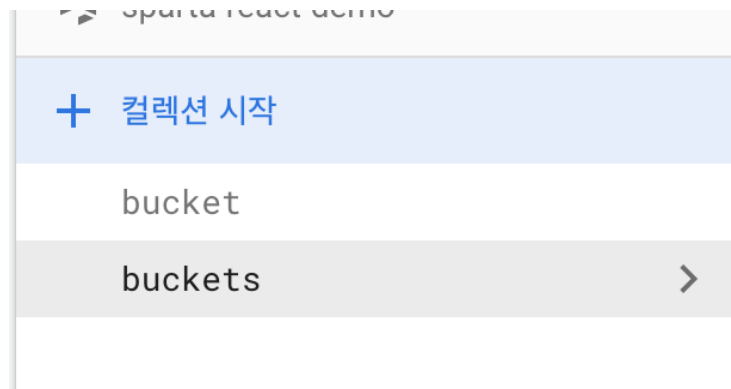
```
bucket.doc("bucket_item").delete([문서 ID]);
```

▼ 19) 컬렉션 이름을 바꾸면 어떻게 될까?

▼ (1) 컬렉션 이름을 바꿔서 추가하면 어떻게 될까?

👉 컬렉션이름만 바꿔서 해봅시다!

```
// bucket에서 buckets로 이름 바꾸기! 그리고 대시보드를 확인해보세요!  
const bucket = firestore.collection("buckets");  
bucket.doc("bucket_item").set({ text: "수영 배우기", completed: false });
```



👉 새로운 컬렉션이 생기죠?! 파이어베이스 어떠세요? 사용하기 정말 쉽죠? 😎

11. 끝 & 숙제 설명



[준비물]

AWS 계정 생성해오기
가비아에서 도메인 구입하기

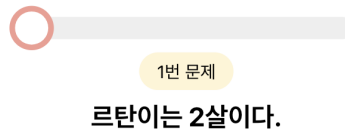
[숙제]

프로그래스바 만들기
파이어스토어 만들고 데이터 넣기! (redux에 있는대로!!)
랭킹 화면에 들어가면 스크롤을 움직여 내 랭킹으로 이동하게 해볼까요!

▼ 기획서(레이아웃) 보기



▼ 예시 화면



12. 4주차 숙제 답안 코드

▼ [코드스니펫] - 4주차 숙제 답안 코드

전체 코드

▼ quiz.js

▼ rank.js

```
// Actions

// 유저 이름을 바꾼다
const ADD_USER_NAME = "rank/ADD_USER_NAME";
// 유저 메시지를 바꾼다
const ADD_USER_MESSAGE = "rank/ADD_USER_MESSAGE";
// 랭킹정보를 추가한다
const ADD_RANK = "rank/ADD_RANK";
// 랭킹정보를 가져온다
const GET_RANK = "rank/GET_RANK";

const initialState = {
  user_name: "",
  user_message: "",
  user_score: "",
  score_text: {
    60: "우린 친구! 앞으로도 더 친하게 지내요! :)",
    80: "우와! 우리는 엄청 가까운 사이!",
    100: "둘도 없는 단짝이에요! :)",
  },
  ranking: [
    { score: 40, name: "임민영", message: "안녕 르탄아!" },
  ],
};

// Action Creators
export const addUserName = (user_name) => {
  return { type: ADD_USER_NAME, user_name };
};

export const addUserMessage = (user_message) => {
  return { type: ADD_USER_MESSAGE, user_message };
};

export const addRank = (rank_info) => {
  return { type: ADD_RANK, rank_info };
};

export const getRank = (rank_list) => {
  return { type: GET_RANK, rank_list };
};

// Reducer
export default function reducer(state = initialState, action = {}) {
  switch (action.type) {
    // do reducer stuff
    case "rank/ADD_USER_NAME": {
      return { ...state, user_name: action.user_name };
    }

    case "rank/ADD_USER_MESSAGE": {
      return { ...state, user_message: action.user_message };
    }

    case "rank/ADD_RANK": {
      return { ...state, ranking: [...state.ranking, action.rank_info] };
    }

    case "rank/GET_RANK": {
```

```

    return { ...state, ranking: action.rank_list };
  }

  default:
    return state;
  }
}

```

▼ configStore.js

```

import { createStore, combineReducers } from "redux";
import quiz from "../modules/quiz";
import rank from "../modules/rank";
import { createBrowserHistory } from "history";

export const history = createBrowserHistory();

const middlewares = [thunk];

const rootReducer = combineReducers({ quiz, rank });
const store = createStore(rootReducer);

export default store;

```

▼ index.js

```

import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { BrowserRouter } from "react-router-dom";

import { Provider } from "react-redux";
import store from "../redux/configStore";

ReactDOM.render(
  <Provider store={store}>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </Provider>,
  document.getElementById("root")
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();

```

▼ App.js

```

import logo from './logo.svg';
import './App.css';
import React from "react";
import { Route, Switch } from "react-router-dom";

import Start from "./Start";

```



```

import Quiz from "../Quiz";
import Score from "../Score";
import Message from "../Message";
import Ranking from "../Ranking";

import { withRouter } from "react-router";
// 리덕스 스토어와 연결하기 위해 connect라는 친구를 호출할게요!
import { connect } from "react-redux";

// 이 함수는 스토어가 가진 상태값을 props로 받아오기 위한 함수예요.
const mapStateToProps = (state) => ({
  ...state,
});

// 이 함수는 값을 변화시키기 위한 액션 생성 함수를 props로 받아오기 위한 함수예요.
const mapDispatchToProps = (dispatch) => ({
  load: () => {

  },

});

class App extends React.Component {
  constructor(props) {
    super(props);

    this.state = {

  };
}

render () {
  return (
    <div className="App">
      <Switch>
        <Route path="/quiz" component={Quiz} />
        <Route path="/" exact component={Start} />
        <Route path="/score" component={Score} />
        <Route path="/message" component={Message} />
        <Route path="/ranking" component={Ranking} />
      </Switch>
    </div>
  );
}
}

export default connect(mapStateToProps, mapDispatchToProps)(withRouter(App));

```

▼ Score.js

```

import React from "react";
import styled from "styled-components";

import { useSelector, useDispatch } from "react-redux";
import { addRank } from "../redux/modules/rank";

const Score = (props) => {
  const name = useSelector((state) => state.quiz.name);
  const score_texts = useSelector((state) => state.quiz.score_texts);

  const answers = useSelector((state) => state.quiz.answers);

  // 정답만 걸러내기
  let correct = answers.filter((answer) => {

```

```

    return answer;
  });

  // 점수 계산하기
  let score = (correct.length / answers.length) * 100;

  // 점수별로 텍스트를 띄워줄 준비!
  let score_text = "";

  // Object.keys는 딕셔너리의 키값을 배열로 만들어주는 친구예요!
  Object.keys(score_texts).map((s, idx) => {
    // 첫번째 텍스트 넣어주기
    if (idx === 0) {
      score_text = score_texts[s];
    }
    // 실제 점수와 기준 점수(키로 넣었던 점수) 비교해서 텍스트를 넣자!
    score_text = parseInt(s) <= score ? score_texts[s] : score_text;
  });

  return (
    <ScoreContainer>
      <Text>
        <span>{name}</span>
        퀴즈에 <br />
        대한 내 점수는?
      </Text>
      <MyScore>
        <span>{score}</span>점<p>{score_text}</p>
      </MyScore>

      <Button
        onClick={() => {
          props.history.push("/message");
        }}
        outlined
      >
        {name}에게 한마디
      </Button>
    </ScoreContainer>
  );
};

const ScoreContainer = styled.div`
  display: flex;
  width: 100vw;
  height: 100vh;
  overflow: hidden;
  padding: 16px;
  box-sizing: border-box;
  flex-direction: column;
  justify-content: center;
  align-items: center;
`;

const Text = styled.h1`
  font-size: 1.5em;
  margin: 0px;
  line-height: 1.4;
  & span {
    background-color: #fef5d4;
    padding: 5px 10px;
    border-radius: 30px;
  }
`;

const MyScore = styled.div`
  & span {

```

```

    border-radius: 30px;
    padding: 5px 10px;
    background-color: #fef5d4;
  }
  font-weight: 600;
  font-size: 2em;
  margin: 24px;

  & > p {
    margin: 24px 0px;
    font-size: 16px;
    font-weight: 400;
  }
`;
const Button = styled.button`
  padding: 8px 24px;
  background-color: ${props => (props.outlined ? "#ffffff" : "#dadafc")};
  border-radius: 30px;
  margin: 8px;
  border: 1px solid #dadafc;
  width: 80vw;
`;

export default Score;

```

▼ Quiz.js

```

import React from "react";
import styled from "styled-components";
import Score from "../Score";
import SwipeItem from "../SwipeItem";

import { useSelector, useDispatch } from "react-redux";
import { addAnswer } from "../redux/modules/quiz";

const Quiz = (props) => {
  const dispatch = useDispatch();
  const answers = useSelector((state) => state.quiz.answers);
  const quiz = useSelector((state) => state.quiz.quiz);

  const num = answers.length;

  const onSwipe = (direction) => {
    let _answer = direction === "left"? "0" : "X";

    if(_answer === quiz[num].answer){
      // 정답일 경우,
      dispatch(addAnswer(true));
    }else{
      // 오답일 경우,
      dispatch(addAnswer(false));
    }
  }

  if (num > quiz.length -1) {
    return <Score {...props}/>;
    // return <div>퀴즈 끝!</div>;
  }

  return (
    <QuizContainer>
      <p>
        <span>{num + 1}번 문제</span>
      </p>

```

```

    {quiz.map((l, idx) => {
      if (num === idx) {
        return <Question key={idx}>{l.question}</Question>;
      }
    })}

    <AnswerZone>
      <Answer>{"0 "}</Answer>
      <Answer>{" X"}</Answer>
    </AnswerZone>

    {quiz.map((l, idx) => {
      if (idx === num) {
        return <SwipeItem key={idx} onSwipe={onSwipe}/>;
      }
    })}
  </QuizContainer>
);
};

const QuizContainer = styled.div`
  margin-top: 16px;
  width: 100%;
  & > p > span {
    padding: 8px 16px;
    background-color: #fef5d4;
    // border-bottom: 3px solid #ffd6aa;
    border-radius: 30px;
  }
`;

const Question = styled.h1`
  font-size: 1.5em;
`;

const AnswerZone = styled.div`
  width: 100%;
  display: flex;
  flex-direction: row;
  min-height: 70vh;
`;

const Answer = styled.div`
  width: 50%;
  display: flex;
  justify-content: center;
  align-items: center;
  font-size: 100px;
  font-weight: 600;
  color: #dadafc77;
`;

const DragItem = styled.div`
  display: flex;
  align-items: center;
  justify-content: center;
  position: fixed;
  top: 0;
  left: 0;
  width: 100vw;
  height: 100vh;

  & > div {
    border-radius: 500px;
    background-color: #ffd6aa;
  }
  & img {
    max-width: 150px;

```

```

    }
  };
  export default Quiz;

```

▼ Start.js

```

import React from "react";
import img from "./scc_img01.png";
import { useDispatch, useSelector } from "react-redux";
import { addUserName } from "./redux/modules/rank";

const Start = (props) => {
  const dispatch = useDispatch();
  const name = useSelector((state) => state.quiz.name);
  const input_text = React.useRef(null);

  // 컬러셋 참고: https://www.shutterstock.com/ko/blog/pastel-color-palettes-rococo-trend/
  return (
    <div
      style={{
        display: "flex",
        height: "100vh",
        width: "100vw",
        overflow: "hidden",
        padding: "16px",
        boxSizing: "border-box",
      }}
    >
      <div
        className="outter"
        style={{
          display: "flex",
          alignItems: "center",
          justifyContent: "center",
          flexDirection: "column",
          height: "100vh",
          width: "100vw",
          overflow: "hidden",
          padding: "0px 10vw",
          boxSizing: "border-box",
          maxWidth: "400px",
          margin: "0px auto",
        }}
      >
        <img
          src={img}
          style={{ width: "80%", margin: "-70px 16px 48px 16px" }}
        />
        <h1 style={{ fontSize: "1.5em", margin: "0px", lineHeight: "1.4" }}>
          나는{" "}
          <span
            style={{
              backgroundColor: "#fef5d4",
              padding: "5px 10px",
              borderRadius: "30px",
            }}
          >
            {name}
          </span>
          에 대해 얼마나 알고 있을까?
        </h1>
        <input
          ref={input_text}
          type="text"
          style={{

```

```

        padding: "10px",
        margin: "24px 0px",
        border: "1px solid #dadafc",
        borderRadius: "30px",
        width: "100%",
        // backgroundColor: "#dadafc55",
      }}
      placeholder="내 이름"
    />
    <button
      onClick={() => {
        // 이름 저장
        dispatch(addUserName(input_text.current.value));
        // 페이지 이동
        props.history.push("/quiz");
      }}
      style={{
        padding: "8px 24px",
        backgroundColor: "#dadafc",
        borderRadius: "30px",
        border: "#dadafc",
      }}
    >
      시작하기
    </button>
  </div>
</div>
);
};

export default Start;

```

▼ Ranking.js

```

import React from "react";
import styled from "styled-components";

import { useSelector, useDispatch } from "react-redux";
import { resetAnswer } from "../redux/modules/quiz";

const Ranking = (props) => {
  const dispatch = useDispatch();
  const _ranking = useSelector((state) => state.rank.ranking);

  React.useEffect(() => {
    // current 가 없을 때는 바로 리턴해줍니다.
    if (!user_rank.current) {
      return;
    }
    // offsetTop 속성을 이용해 스크롤을 이동하자!
    window.scrollTo({
      top: user_rank.current.offsetTop,
      left: 0,
      behavior: "smooth",
    });
  }, []);

  // 스크롤 이동할 div의 ref를 잡아줄거예요!
  const user_rank = React.useRef(null);

  // Array 내장 함수 sort로 정렬하자!
  // https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort

  const ranking = _ranking.sort((a, b) => {
    // 높은 수가 맨 앞으로 오도록!

```

```

    return b.score - a.score;
  });

  return (
    <RankContainer>
      <Topbar>
        <p>
          <span>{ranking.length}명</span>의 사람들 중 당신은?
        </p>
      </Topbar>

      <RankWrap>
        {ranking.map((r, idx) => {
          if (r.current) {
            return (
              <RankItem key={idx} highlight={true} ref={user_rank}>
                <RankNum>{idx + 1}등</RankNum>
                <RankUser>
                  <p>
                    <b>{r.name}</b>
                  </p>
                  <p>{r.message}</p>
                </RankUser>
              </RankItem>
            );
          }
          return (
            <RankItem key={idx}>
              <RankNum>{idx + 1}등</RankNum>
              <RankUser>
                <p>
                  <b>{r.name}</b>
                </p>
                <p>{r.message}</p>
              </RankUser>
            </RankItem>
          );
        })}
      </RankWrap>

      <Button
        onClick={() => {
          dispatch(resetAnswer());
          window.location.href = "/";
        }}
      >
        다시 하기
      </Button>
    </RankContainer>
  );
};

const RankContainer = styled.div`
  width: 100%;
  padding-bottom: 100px;
`;

const Topbar = styled.div`
  position: fixed;
  top: 0;
  left: 0;
  width: 100vw;
  min-height: 50px;
  border-bottom: 1px solid #ddd;
  background-color: #fff;
  & > p {
    text-align: center;
  }

```

```

    & > p > span {
      border-radius: 30px;
      background-color: #fef5d4;
      font-weight: 600;
      padding: 4px 8px;
    }
  `;

const RankWrap = styled.div`
  display: flex;
  flex-direction: column;
  width: 100%;
  margin-top: 58px;
`;

const RankItem = styled.div`
  width: 80vw;
  margin: 8px auto;
  display: flex;
  border-radius: 5px;
  border: 1px solid #ddd;
  padding: 8px 16px;
  align-items: center;
  background-color: ${(props) => (props.highlight ? "#ffd6aa" : "#ffffff")};
`;

const RankNum = styled.div`
  text-align: center;
  font-size: 2em;
  font-weight: 600;
  padding: 0px 16px 0px 0px;
  border-right: 1px solid #ddd;
`;

const RankUser = styled.div`
  padding: 8px 16px;
  text-align: left;
  & > p {
    &:first-child > b {
      border-bottom: 2px solid #212121;
    }
    margin: 0px 0px 8px 0px;
  }
`;

const Button = styled.button`
  position: fixed;
  bottom: 5vh;
  left: 0;
  padding: 8px 24px;
  background-color: ${(props) => (props.outlined ? "#ffffff" : "#dadafc")};
  border-radius: 30px;
  margin: 0px 10vw;
  border: 1px solid #dadafc;
  width: 80vw;
`;

export default Ranking;

```

▼ Message.js

```

import React from "react";
import img from "./scc_img01.png";
import { useDispatch, useSelector } from "react-redux";

```



```

import {addRank} from "../redux/modules/rank";

const Message = (props) => {
  const dispatch = useDispatch();
  const name = useSelector((state) => state.quiz.name);
  const answers = useSelector((state) => state.quiz.answers);
  const user_name = useSelector((state)=>state.rank.user_name);

  const input_text = React.useRef(null);
  // 정답만 걸러내기
  let correct = answers.filter((answer) => {
    return answer;
  });

  // 점수 계산하기
  let score = (correct.length / answers.length) * 100;

  // 컬러셋 참고: https://www.shutterstock.com/ko/blog/pastel-color-palettes-rococo-trend/
  return (
    <div
      style={{
        display: "flex",
        height: "100vh",
        width: "100vw",
        overflow: "hidden",
        padding: "16px",
        boxSizing: "border-box",
      }}
    >
      <div
        className="outter"
        style={{
          display: "flex",
          alignItems: "center",
          justifyContent: "center",
          flexDirection: "column",
          height: "100vh",
          width: "100vw",
          overflow: "hidden",
          padding: "0px 10vw",
          boxSizing: "border-box",
          maxWidth: "400px",
          margin: "0px auto",
        }}
      >
        <img
          src={img}
          style={{ width: "80%", margin: "-70px 16px 48px 16px" }}
        />
        <h1 style={{ fontSize: "1.5em", margin: "0px", lineHeight: "1.4" }}>
          <span
            style={{
              backgroundColor: "#fef5d4",
              padding: "5px 10px",
              borderRadius: "30px",
            }}
          >
            {name}
          </span>
          에게 한마디
        </h1>
        <input
          ref={input_text}
          type="text"
          style={{
            padding: "10px",
            margin: "24px 0px",
            border: "1px solid #dadafc",

```

```

        borderRadius: "30px",
        width: "100%",
      }}
      placeholder="한 마디 적기"
    />
    <button
      onClick={() => {

        let rank_info = {
          score: parseInt(score),
          name: user_name,
          message: input_text.current.value,
          current: true,
        };
        // 랭킹 정보 넣기
        dispatch(addRank(rank_info));
        // 주소 이동
        props.history.push('/ranking');

      }}
      style={{
        padding: "8px 24px",
        backgroundColor: "#dadafc",
        borderRadius: "30px",
        border: "#dadafc",
      }}
    >
      한마디하고 랭킹 보러 가기
    </button>
  </div>
</div>
);
};

export default Message;

```

▼ Progress.js

```

import React from "react";
import styled from "styled-components";

import { useSelector } from "react-redux";

const Progress = (props) => {
  // 퀴즈 리스트 가지고 오기
  const quiz_list = useSelector((state) => state.quiz.quiz);
  // 유저 답 리스트 가지고 오기
  const answers = useSelector((state) => state.quiz.answers);
  // 답 리스트 갯수 세기
  let count = answers.length;

  return (
    <ProgressBar>
      <HighLight width={({count / quiz_list.length) * 100 + "%"} />
      <Dot />
    </ProgressBar>
  );
};

const ProgressBar = styled.div`
  width: 80%;
  margin: 20px auto;
  background: #eee;
  // width: 100%;
  height: 20px;

```

```

    display: flex;
    align-items: center;
    border-radius: 10px;
    `;

const HighLight = styled.div`
  background: #df402c88;
  height: 20px;
  width: ${({props}) => props.width};
  transition: width 1s;
  border-radius: 10px;
  `;

const Dot = styled.div`
  background: #fff;
  border: 5px solid #df402c88;
  box-sizing: border-box;
  margin: 0px 0px 0px -10px;
  width: 40px;
  height: 40px;
  border-radius: 20px;
  `;

export default Progress;

```

Copyright © TeamSparta All rights reserved.