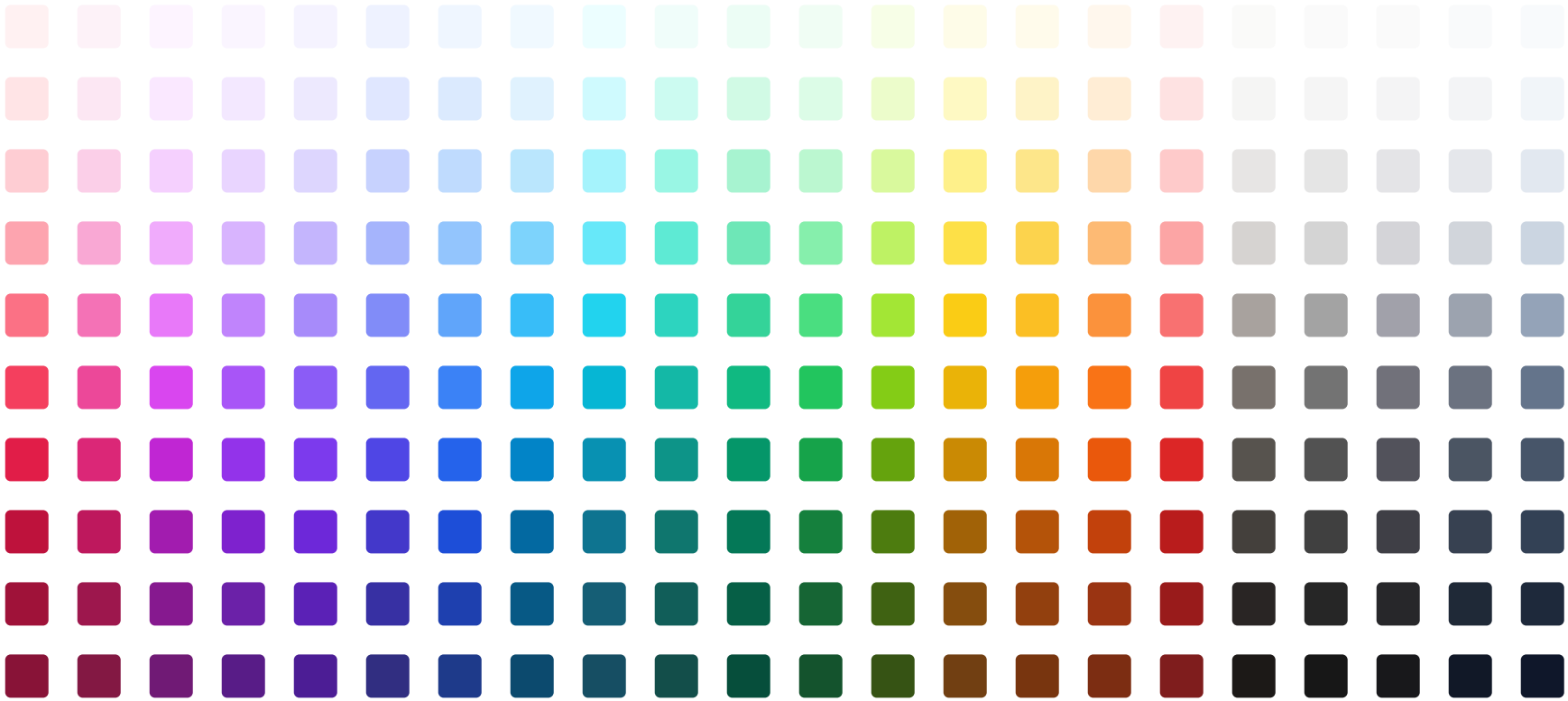
React
Reference Book



JSX

Javascript + XML을 합성어로, React에서는 이벤트가 처리되는 방식, 시간에 따라 state가 변하는 방식, 화면에 표시하기 위해 데이터가 준비되는 방식 등 렌더링 로직이 본질적으로 다른 UI 로직과 연결합니다.

```
ReactDOM.render(
  <h1>Hello, world!</h1>,
  document.getElementById('root')
);
```

h1태그 부분을 따로 변수로
저장해서 아래와 같이 쓸 수 도 있다.

```
const element = <h1>Hello, world!</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

- JSX에 표현식 포함하기 -

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};

const element = (
  <h1>Hello, {formatName(user)}!</h1>
);

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

중괄호를 이용해서 변수, 함수 등
여러 표현식을 사용 할 수 있다.

npm

NPM는 Node Package Manager의 약자이다. 자바스크립트 패키지 매니저이고 NodeJS에서 사용할 수 있는 모듈들을 패키지화하여 모아둔 저장소 역할을 하며 설치/관리를 수행할 수 있는 CLI를 제공한다.

```
npm -v
```

npm의 버전을 확인할 수 있습니다.

create-react-app

Create React App은 React 배우기에 간편한 환경이고, 시작하기에 최고의 방법은 새로운 싱글 페이지 애플리케이션 이다.

설치 명령어

```
npx create-react-app 파일이름(portfolio)
```

```
Happy hacking!
```

Component

컴포넌트는 프로그래밍의 한 부분을 의미하며 재사용이 가능한 최소 단위를 말한다. 그래서 객체지향언어를 사용할 때 자주 사용되며 재사용이 가능하기 때문에 컴포넌트 단위로 분류하거나 이동 가능하다는 특징이 있다.

< 컴포넌트의 특징 >

1. 컴포넌트는 독립적인 소프트웨어 모듈이다.
2. 컴포넌트는 구현, 명세화, 패키지화, 그리고 배포될 수 있어야 한다.
3. 하나의 컴포넌트는 하나 이상의 클래스들로 구성될 수 있다.
4. 컴포넌트는 인터페이스를 통해서만 접근할 수 있다.

```
class HelloMessage extends React.Component {
  render() {
    return <div>Hello <x-search>{this.props.name}</x-search>!</div>;
  }
}
```

웹 컴포넌트에서 나온 이벤트들은 React 렌더링 트리에 올바르게 전파되지 않을 수 있습니다. 이를 해결하기 위해 이벤트를 다루기 위한 핸들러를 React 컴포넌트 내에 각각 만들어야합니다.

Props

Props는 **상위 컴포넌트에서 하위 컴포넌트로 데이터를 전달할 때 사용**한다. 해당 Props 속성을 이용하여 전달할 수 있는 데이터는 객체, 배열 등 유형과 상관없이 모든 것이 전달 가능하다고 한다.

1) 함수형 컴포넌트에서 PROPS사용 하기

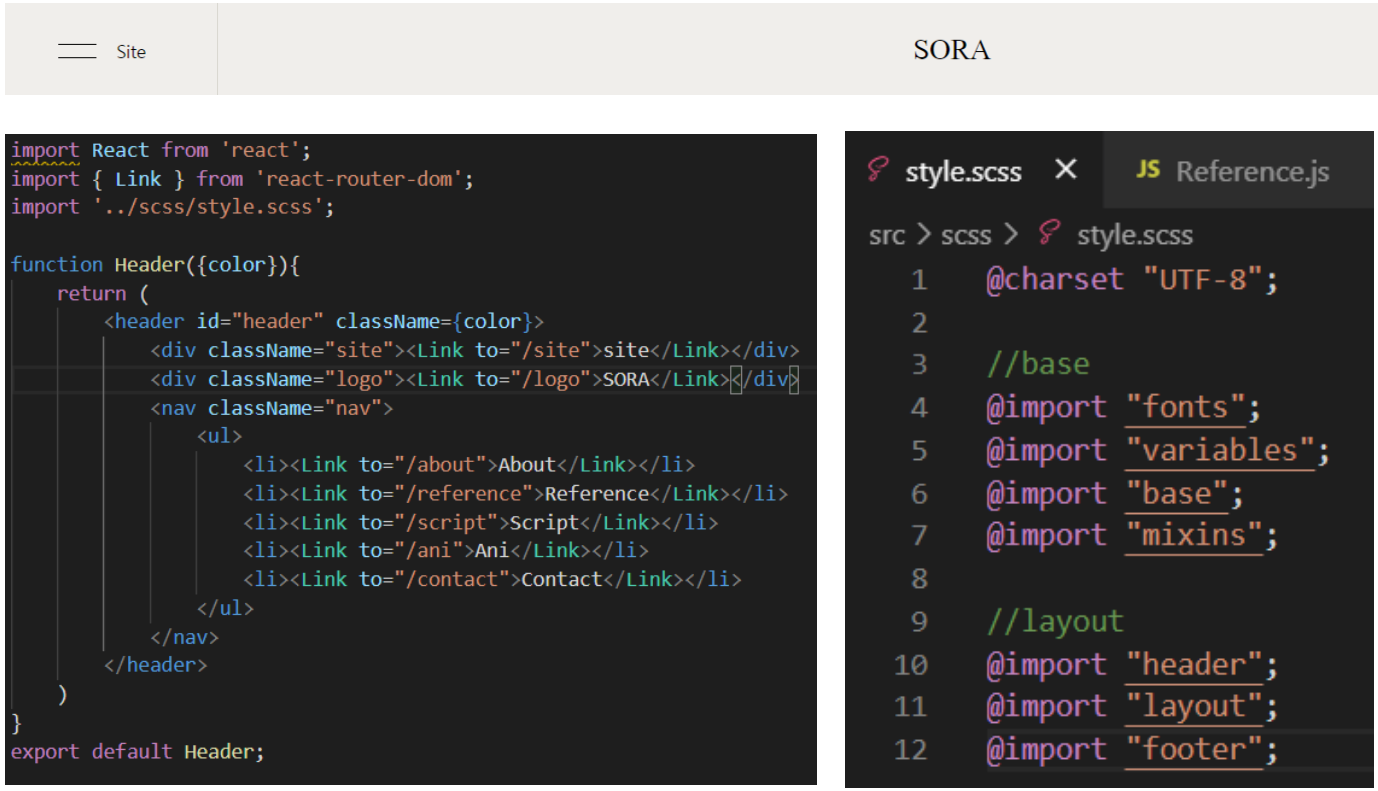
```
function Square(props){

    return(

        <h1>{props.number} square is {props.number * props.number}</h1>
    )


}


ReactDOM.render(
    <Square number="2"></Square>

    , document.getElementById('root'));
```

2)클래스형 컴포넌트에서 props사용하기

```
class Square2 extends React.Component{

    render(){

        return(

            <h1>{this.props.number} square is {this.props.number * this.props.number}</h1>

        );
    }


}


ReactDOM.render(
    <div>
        <Square number="2"></Square>
        <Square2 number ="3"></Square2>
    </div>

    , document.getElementById('root'));
```

Header

React를 이용하여 Porfolio header영역을 코딩하였다.



```react_header.js
import React from 'react';
import { Link } from 'react-router-dom';
import '../scss/style.scss';

function Header({color}){
    return (
        <header id="header" className={color}>
            <div className="site">Link to="/site">site</Link></div>
            <div className="logo">Link to="/logo">SORA</Link></div>
            <nav className="nav">
                <ul>
                    <li>Link to="/about">About</Link></li>
                    <li>Link to="/reference">Reference</Link></li>
                    <li>Link to="/script">Script</Link></li>
                    <li>Link to="/ani">Ani</Link></li>
                    <li>Link to="/contact">Contact</Link></li>
                </ul>
            </nav>
        </header>
    )
}
export default Header;
```



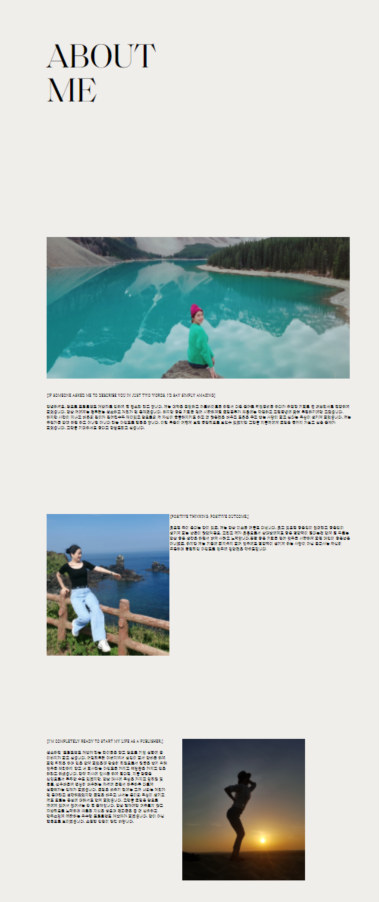
```style.scss
@charset "UTF-8";

//base
@import "fonts";
@import "variables";
@import "base";
@import "mixins";

//layout
@import "header";
@import "layout";
@import "footer";
```

```
<Header color="white" />
```

header.js를 만들어 header영역을 만들고 style.scss에 import시켜 주었다. 그리고 각 페이지마다 헤더 색이 다르므로 기본 header는 black으로 정의해놨으므로, white인 부분은 각각의 js영역에 <Header color="white" />로 정의해준다.

About

React를 이용하여 Porfolio about 페이지를 코딩하였다.




```
const abouMe= [
    {
        id:1,
        title:'[IF SOMEONE ASKED ME TO DES
        desc:'안녕하세요. 앞으로 프론트앤드
        imge:'../img/Mask1.jpg',
        imge2x:'../img/Mask1@2.jpg 2x, ../
        // imge3x:'../img/Mask1@3.jpg',
    },
    {
        id:2,
        title:'[POSITIVE THINKING, POSITIV
        desc:'웃으면 복이 온다는 말이 있죠
        imge:'../img/Mask2.jpg',
        imge2x:'../img/Mask2@2.jpg 2x, ../
        // imge3x:'../img/Mask2@3.jpg',
    },
    {
        id:3,
        title:'[I'M COMPLETELY READY TO ST
        desc:'생소하던 '프론트앤드 개발자'
        imge:'../img/Mask3.jpg',
        imge2x:'../img/Mask3@2.jpg 2x, ../
        // imge3x:'../img/Mask3@3.jpg',
    },
]
```

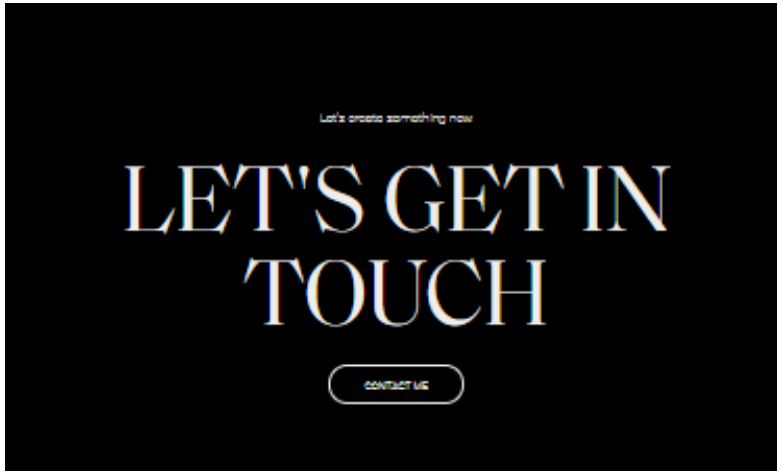
```
import React from 'react';
import Header from '../components/Header';
import Title from '../components/Title';
import Footer from '../components/Footer';

function AboutText({title, desc, imge, imge2x}){
    return (
        <div>
            <div className="aileft">
                <img src={imge} srcSet={imge2x} alt={title} />
                {/* <img src="../img/Mask1@2.jpg 2x"></img> */}
            </div>
            <div className="airight">
                <h2>{title}</h2>
                <p>{desc}</p>
            </div>
        </div>

    )
}
```

```
function About(){
    return (
        <div id="wrap">
            <Header color="white" />
            <main id="main">
                <section id="aboutCont">
                    <div className="container">
                        <div className="titleWrap">
                            <Title text={['About', 'Me', 'About Me']}/>
                        </div>
                        <div className="aboutCont">
                            {abouMe.map((txt)=>(
                                <AboutText key={txt.id} title={txt.title} desc={txt.desc} imge={txt.imge} imge2x={txt.imge2x}/>
                            ))}
                        </div>
                    </div>
                </section>
            </main>
            <Footer />
        </div>
    )
}
export default About;
```

react에서는 기존의 방식과는 다르게 페이지 안에 들어갈 내용들을 배열을 이용하여 좀 더 깔끔하게 코드를 만들수 있었다. 변수 aboutMe에 id, title, desc, image, image2x에 해당하는 항목을을 집어 넣은후, 각 아이디의 key 값을 불러 오기 위해 <AboutText key=>로 불러올 수 있었다. 그리고 layout.scss에 이미지와 해당 글귀들의 위치와 이동값들의 css속성을 넣어 페이지를 완성 할수 있었고, 메인페이지인 App.js에 클릭하면 해당 페이지로 이동할수 있도록 import시켜 페이지를 완성하였다.

Footer

React를 이용하여 Porfolio footer영역을 코딩하였다.




```
import React from 'react';
import { Link } from 'react-router-dom';
function Footer (){
    return (
        <footer id="footer">
            <div className="container">
                <div class="footer">
                    <div>
                        <p>Let's create something new</p>
                        <h3>LET'S GET IN<br />TOUCH</h3>
                        <Link to="/contact" className="contact">Contact Me</Link>
                    </div>
                </div>
            </div>
        </footer>
    )
}
export default Footer;
```

```
</main>
<Footer />
/div>
```

```
#footer {
    background-color: $color-black;
    height: 100vh;
}

.footer {
    height: 100vh;
    display: flex;
    align-items: center;
    justify-content: center;
    text-align: center;
    color: $color-white;

    > div {
```

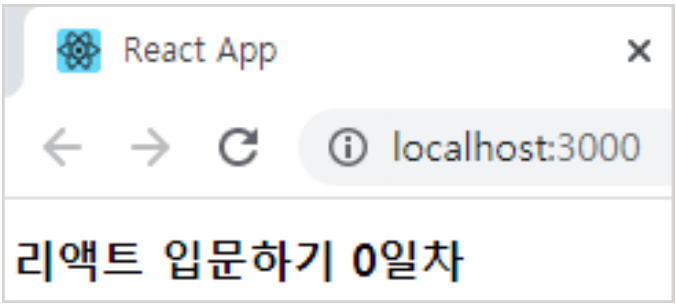
```
> div {

    p {
        font-family: $font-basic;
        font-size: 20px;
        font-weight: 200;
        margin-bottom: 2rem;
    }
    h3 {
        font-family: $font-main;
        font-size: 144px;
        line-height: 1;
        text-transform: uppercase;
        margin-bottom: 5rem;
    }
    a {
        color: $color-white;
        border: 1px solid $color-white;
        border-radius: 50px;
        padding: 18px 53px 18px 53px;
        font-size: 16px;
        font-weight: 300;
        font-family: $font-basic;
        text-transform: uppercase;

    }
}
```

footer영역을 만들기 위해 comfonents 파일에 footer.js를 새로 만들어 Link를 import 해주고, 안에 들어갈 문구들을 넣은후에 style.scss에도 import해준다. 그리고 footer.scss에 css 속성값을 넣어주어 위치와 폰트에 변화를 주어 디자인 해주고, header색과 같은 방식으로, footer가 들어가는 페이지에는 <Footer />를 넣어 페이지를 완성하였다.

State

setState()는 컴포넌트의 state 객체에 대한 업데이트를 실행합니다. state가 변경되면, 컴포넌트는 리렌더링되며, props는 (함수 매개변수처럼) 컴포넌트에 전달되는 반면 state는 (함수 내에 선언된 변수처럼) 컴포넌트 안에서 관리된다.

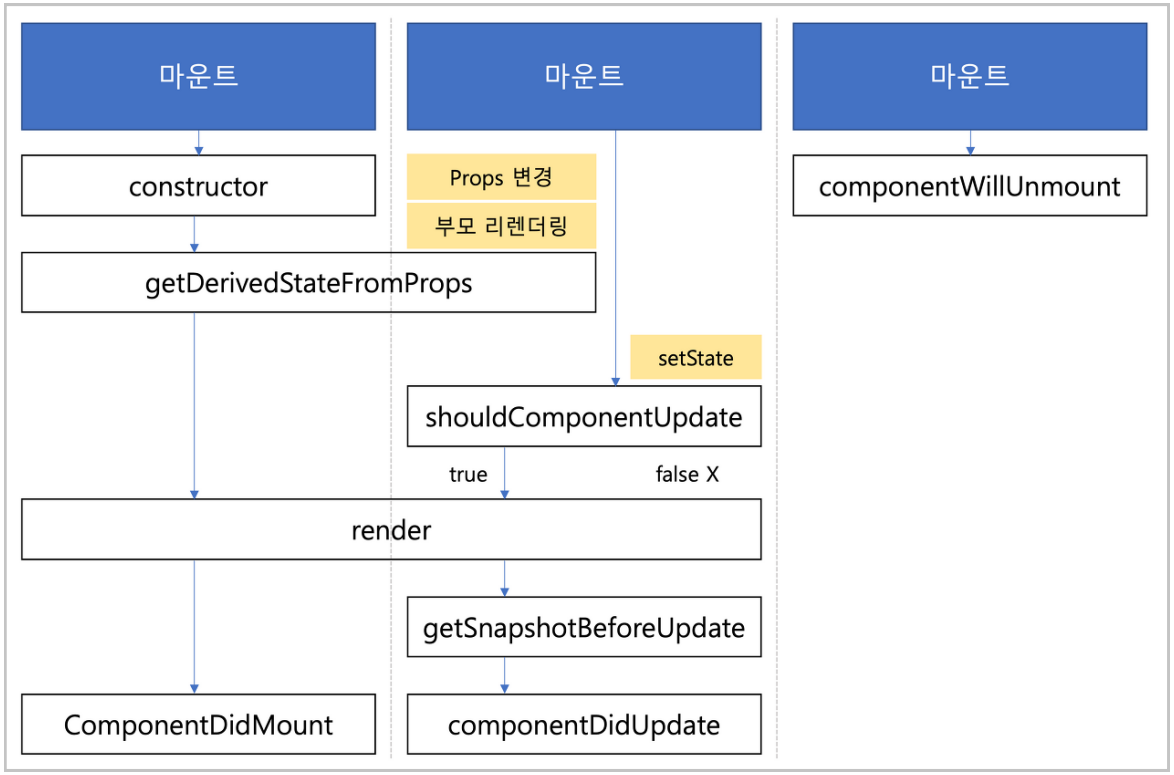
```
import React from 'react';

class App extends React.Component {
  state = {
    count: 0
  };
  render(){
    return <h1>리액트 입문하기 {this.state.count}일차</h1>
  }
}

export default App;
```



render() 내에서 클래스의 state를 사용하기 위해서는 이전과는 달리 this를 사용해주어야 합니다.

생명주기

컴포넌트가 실행되거나 업데이트되거나 제거될 때, 특정한 이벤트들이 발생하는데 이것을 생애 주기 또는 라이프사이클(Life cycle)이라고 한다.



< 라이프사이클 메서드의 이해 >

1. 마운트 : DOM이 생성되고 브라우저상에 나타나는것을 마운트라고 합니다.
2. 업데이트 : 업데이트가 발생하는 경우는 다음과 같습니다.
   - props가 바뀌는 경우
   - state가 바뀌는 경우
   - 부모컴포넌트가 리렌더링 되는 경우
   - this.forceUpdate로 강제 렌더링을 트리거하는 경우
3. 언마운트
   DOM에서 컴포넌트를 제거하는것을 언마운트라고 합니다.

axios

Axios는 HTTP통신을 하는데 매우 인기있는 Javascript라이브러리입니다. Axios는 브라우저와 Node.js 플랫폼에서 모두 사용할 수 있습니다. Axios는 Promise를 기반으로하여 async/await문법을 사용하여 XHR요청을 매우 쉽게 할 수 있습니다.

```
axios({
  url: 'https://test/api/cafe/list/today',
  method: 'get',
  data: {
    foo: 'diary'
  }
});
```

axios객체는 아래와 같이 간단하게 HTTP요청을 할 수 있다.

npm에서 설치 ⟶

```
npm install axios
```

< axios의 장점 >

- 구형브라우저를 지원합니다.
  (Fetch API의 경우는 폴리필이 필요합니다.)
- 요청을 중단시킬 수 있습니다.
- 응답 시간 초과를 설정하는 방법이 있습니다.
- CSRF 보호 기능이 내장되어있다.
- JSON 데이터 자동변환
- Node.js에서의 사용

async/await

Promise의 단점을 해결하기 위해 async/await 키워드가 추가되었다. async/await 키워드를 사용하면 비동기 코드를 마치 동기 코드처럼 보이게 작성할 수 있다. callback이나 promise와 같이 비동키 코드를 작성하는 새로운 방법이다.

< Async / Await을 이용한 비동기요청 처리 방법 >

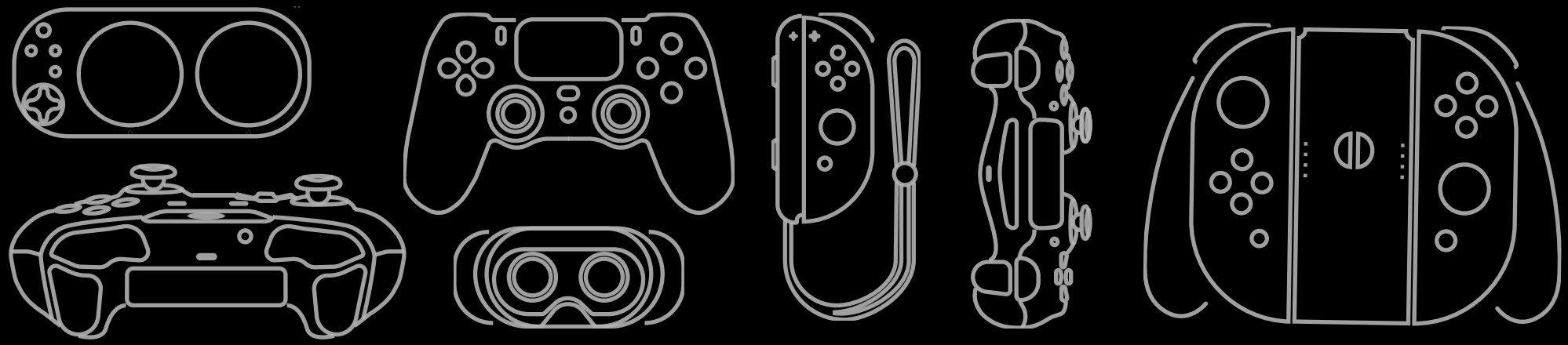
```
const users = async() => {
    console.log(await getData());
    return await getData();
}
```

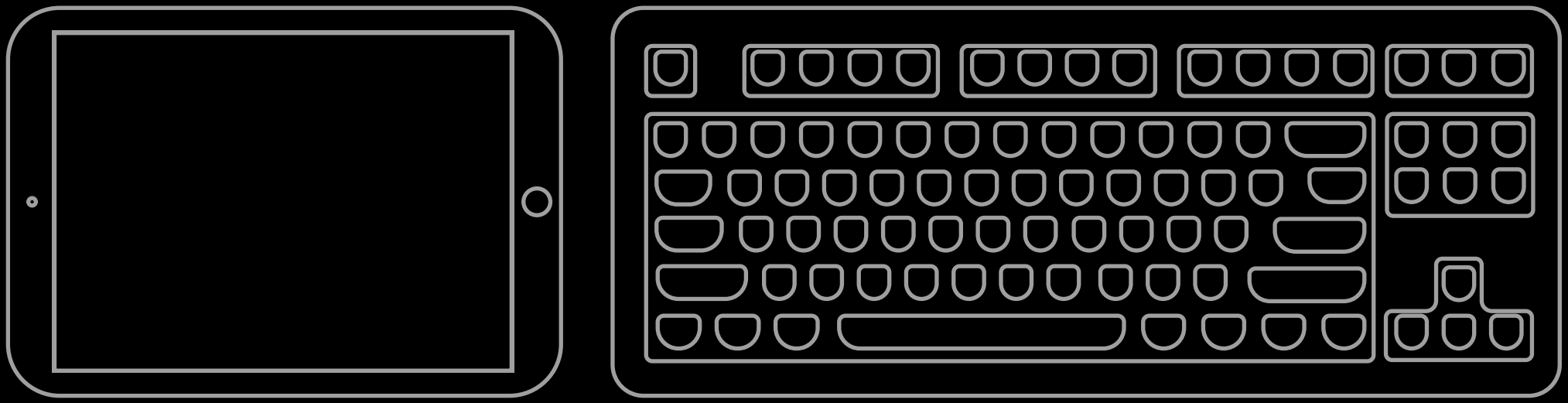
< Async / Await의 장점 >

- 코드가 간결해지고, 가독성이 높아진다.
- try / catch로 에러를 핸들링할 수 있다.
- error 확인이 쉽다.

VUE

REFERENCE BOOK





Vue란?

vue.js는 웹 페이지 화면을 개발하기 위한 프론트엔드 프레임워크이다.

< Vue의 특징 >

첫째, 배우기가 쉽습니다.

HTML, CSS, Javascript 의 기초만 아는 입문자라도 하루 안에 배울 수 있고, 익숙한 실무 개발자라면 몇 시간 이내에 배울 수 있습니다.

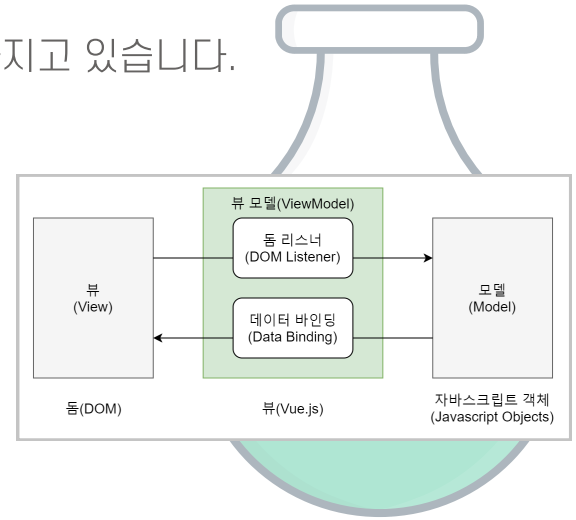
둘째, React나 Angular에 비해 성능이 우수하고 빠릅니다.

Vue 제작팀에서 React와 Angular를 가지고 같은 성능을 비교한 결과 뷰가 가장 빠른 것으로 나타났습니다.

셋째, React와 Angular의 장점을 모두 갖고있습니다.

Angular의 데이터 바인딩 특성과 React의 가상 돔 기반 랜더링 특징을 모두 가지고 있습니다.

용어	설명
뷰(View)	사용자에게 보이는 화면
돔(DOM) ***Document Object Model**	HTML 문서에 들어가는 요소(태그, 클래스, 속성 등)의 정보를 담고 있는 데이터 트리
돔 리스너(DOM Listener)	돔의 변경 내역에 대해 즉각적으로 반응하여 특정 로직을 수행하는 장치
모델(Model)	데이터를 담는 용기, 보통은 서버에서 가져온 데이터를 자바스크립트 객체 형태로 저장
데이터 바인딩(Data Binding)	뷰(View)에 표시되는 내용과 모델의 데이터를 동기화
뷰 모델(ViewModel)	뷰와 모델의 중간 영역, 돔 리스너와 데이터 바인딩을 제공하는 영역



Router

vue에서 라우팅 기능을 구현할 수 있도록 지원하는 공식 라이브러리로, 하나의 페이지로 동작하기 때문에 실제로 다른 주소로 접근하여 웹페이지를 리로드하지 않는 대신에 Router 설정이 반드시 필요하게 됩니다.

태그	설명
<router-link to="URL값">	페이지 이동 태그, 화면에선 <a> 태그로 치환됨
<router-view>	페이지 표시 태그 , 변경된 URL에 따라 해당 컴포넌트를 뿌려주는 영역

[뷰 라우터 기초]

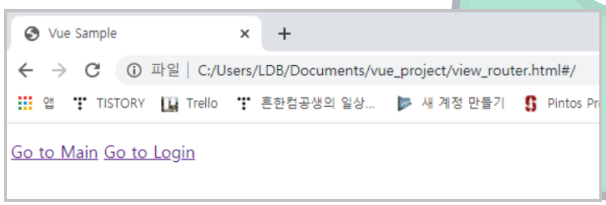
<router - link to = “URL” 태그

페이지 이동 태그, 화면에서는 <a>로 렌더링 되며, 클릭하면 to에 지정한 url로 이동.

<router-view>

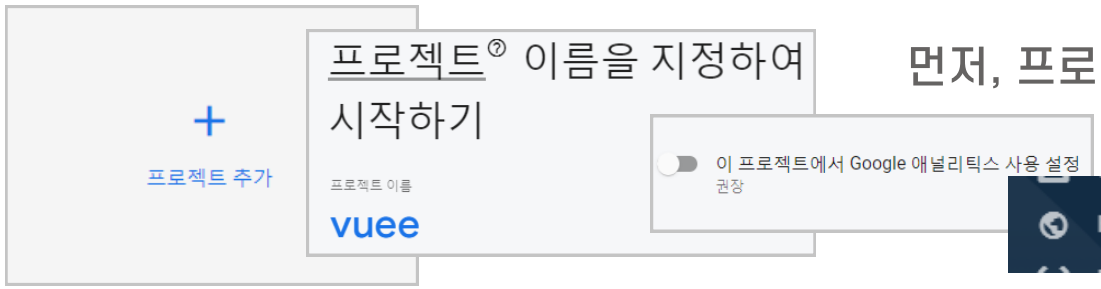
페이지 표시 태그, 변경되는 url에 따라 해당 컴포넌트를 화면에 그려지는 영역

```
<div id="app">
    <p>
        <router-link to="/main">Go to Main</router-link>
        <router-link to="/login">Go to Login</router-link>
    </p>
    <router-view></router-view>
</div>
```

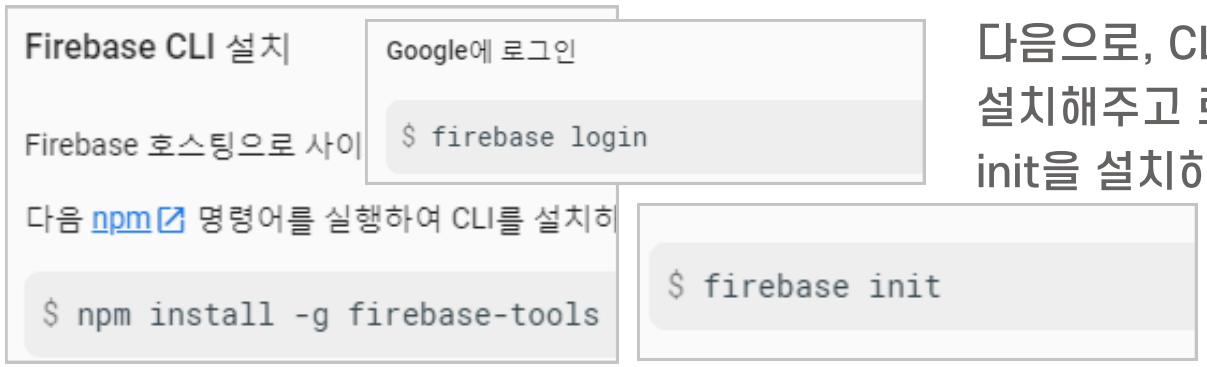


Firebase-Vue

FireBase는 모든 플렛폼의 프로젝트 구축 시 자동적으로 만들어 줍니다. 또한 서버를 구축하기 위해서 리눅스 명령어를 알 필요도 없으며 도메인을 구입할 필요도 없고 개발하는 동안에는 서버를 구입할 필요도 없습니다.



먼저, 프로젝트 추가하기를 누른 후, 이름을 지정해 주고, 사용설정 권장해제를 해준다.



다음으로, CLI명령어를 넣어 설치해주고 로그인을 한 후, init을 설치하고 deploy를 해주면 된다.

선언적 렌더링

Vue.js의 핵심은 간단한 템플릿 구문을 사용해 선언적으로 DOM에 데이터를 렌더링하는 것입니다.

```
<!doctype html>
<html lang="ko">
  <head>
    <meta charset="utf-8">
    <title>JavaScript</title>
  </head>
  <body>
    <script>
      var jbBtn = document.createElement( 'button' );
      var jbBtnText = document.createTextNode( 'Click' );
      jbBtn.appendChild( jbBtnText );
      document.body.appendChild( jbBtn );
    </script>
  </body>
</html>
```

React와 같은 형식의 DOM 을 사용하여 작성 한다. 변수 App 이 있는 부분이 바로 DOM의 형식으로 사용하는 방법이다. 이는 기존에 있는 jquery의 Document안에 사용하는 것이랑 똑같다고 보면된다. Data 에 담겨 있는 메세지를 이용하여 app이라는 id에 내용을 적용 시켜 줄 것이다. 이것이 바로 선언적 렌더링 이라는 것이다. 그냥 내용 복 붙이라고 생각하면 쉽다. {{ }} 안에 data의 이름과 똑같이 적어 주면 안에 있는 내용이 적용이 된다.

조건문

Vue에서는 v-if 디렉티브를 사용해 조건문을 작성하여, 여느 다른 언어의 조건문과 같이 v-if , v-else-if, v-else 사용한다.

```
<template v-if="ok">
   <h1>Title</h1>
   <p>Paragraph 1</p>
   <p>Paragraph 2</p>
</template>
```

v-if 는 디렉티브이기 때문에 하나의 엘리먼트에 추가 해야한다.

```
<div v-if="Math.random() > 0.5">
   you can see me
</div>
<div v-else>
   you can't see me
<div>
```

```
<div v-if="type === 'A'">
   A
</div>
<div v-else-if="type === 'B'">
   B
</div>
<div v-else-if="type === 'C'">
   C
</div>
<div v-else>
   Not A/B/C
</div>
```

v-else-if 는 v-if 에 대한 “else if 블록” 역할, 여러 개 사용 가능하며, 순서는 v-if > v-else-if > v-else 순이다.

반복문

데이터에서 배열이나, 객체의 내용을 반복해서 출력할 때 v-for을 사용한다.

```
<div id="app">
  <span v-for="n in 10">{{ n }} </span>
</div>

<script src="//unpkg.com/vue"></script>
<script>
new Vue({
  el: '#app'
})
</script>
```

Result

1 2 3 4 5 6 7 8 9 10

예를들어 배열의 이름이 stories이면 <li v-for="story in stories>로 stories의 내용을 각각 story로 하나씩 꺼내서 리스트로 나열하겠다는 뜻이며, 객체도 마찬가지로 사용이가능하다. 여기서 배열이나 객체의 이름을 복수형의 의미를 담은 단어로 지으면 직관적으로 알아보기 쉽게 사용할 수 있다. 배열에서 데이터의 index를 표시해야 한다면, <li v-for="(story,index)in stories>로 for문을 작성하고, {{index}}로 각각의 index를 출력할 수 있다.

컴포넌트

Vue.js의 핵심 기능으로, 기본 HTML 엘리먼트를 확장해 재사용이 가능하도록 캡슐화하는데 사용한다.

```
//HTML
<div class="container">
 <list></list>
</div>
//JavaScript
<script type="text/javascript">
 Vue.component('list',{
   template: '<h1>Hello, World!</h1>'
 });
 new Vue({
   el: '.container'
 })
</script>
```

1. HTML단에 새로 만든 컴포넌트 작성

```
//JavaScript
<script type="text/template" id="item-template">
 <h1>Hello, World!</h1>
</script>
<script type="text/javascript">
 Vue.component('list', {
   template: '#item-template'
 });
</script>
```

2. 스크립트단에서 모두 처리한다.

3. template태그를 사용한다.

```
//HTML
<template id='item-template'>
 <h1>Hello, World!</h1>
</template>
//JavaScript
<script type="text/javascript">
 Vue.component('list', {
   template: '#item-template'
 });
</script>
```

4. 같은 템플릿에서 다른 내용을 전달하고 싶을 때 props를 사용한다.

props

prop는 상위 컴포넌트의 정보를 전달하기위한 사용자 지정 특성입니다. 하위 컴포넌트는props 옵션을 사용하여 수신 할 것으로 기대되는 props를 명시적으로 선언해야합니다

```
<child message="안녕하세요!"></child>
<script>
Vue.component('child', {
  // props 정의
  props: ['message'],
  // 데이터와 마찬가지로 prop은 템플릿 내부에서 사용할 수 있으며
  // 인스턴스의 this.message로 사용할 수 있습니다.
  template: '<span></span>'
})
</script>
```

모든 컴포넌트 인스턴스는 자체 범위가 있어, 하위 컴포넌트의 템플릿에서 상위 데이터를 직접 참조할 수 없고, 데이터는 props옵션을 사용해서만 하위 컴포넌트로 전달 될 수 있습니다.

javascript에서 camelCased 컨벤션을 이용해 작성할 경우 HTML의 표현은 조금 다르게 작성해야하고, camelCased(대문자 사용) prop이름에 해당하는 HTML속성은 대소문자를 구분하지 않기 때문에 kebab-case (하이픈 구분)를 사용해야합니다.

```
<script>
Vue.component('child', {
  // JavaScript는 camelCase
  props: ['myMessage'],
  template: '<span></span>'
})
</script>
<!-- HTML는 kebab-case -->
<child my-message="안녕하세요!"></child>
```

SORA KIM

Web Publisher

신입 웹 퍼블리셔 김소라입니다.
잘 부탁드립니다.

경기도 안산시 고잔동 원고잔공원로22, 502호
sorakim911130@gmail.com | kakao ID : belle1130

Education

2020 - 21

라인 컴퓨터 아트 학원

웹표준기반 스마트UI/UX 디자인

Main Skills

Photoshop

Illustrator

HTML5

CSS3

Javascript

React | Vue | Php

Education

협성대학교

경영학과

2010.03 - 2016.02

Contact

김소라 | Sora Kim

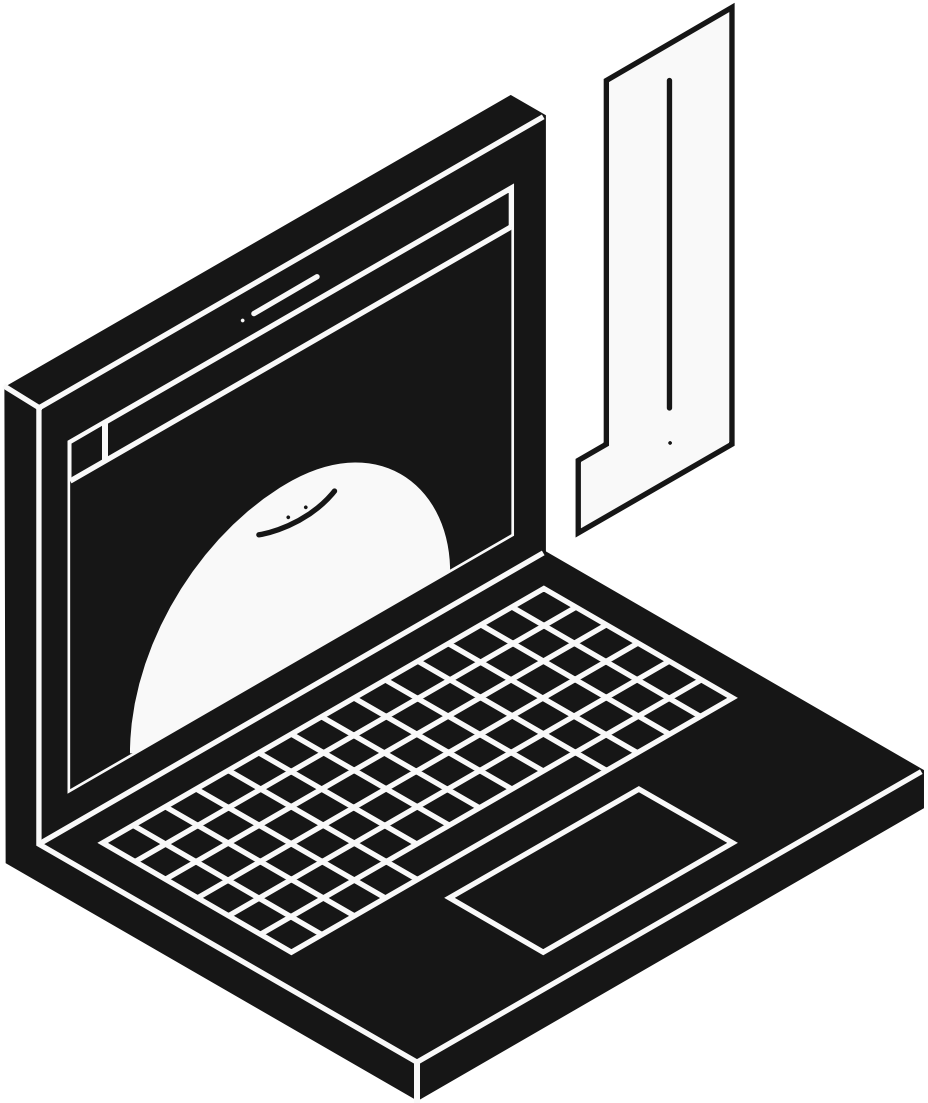
email : sorakim911130@gmail.com

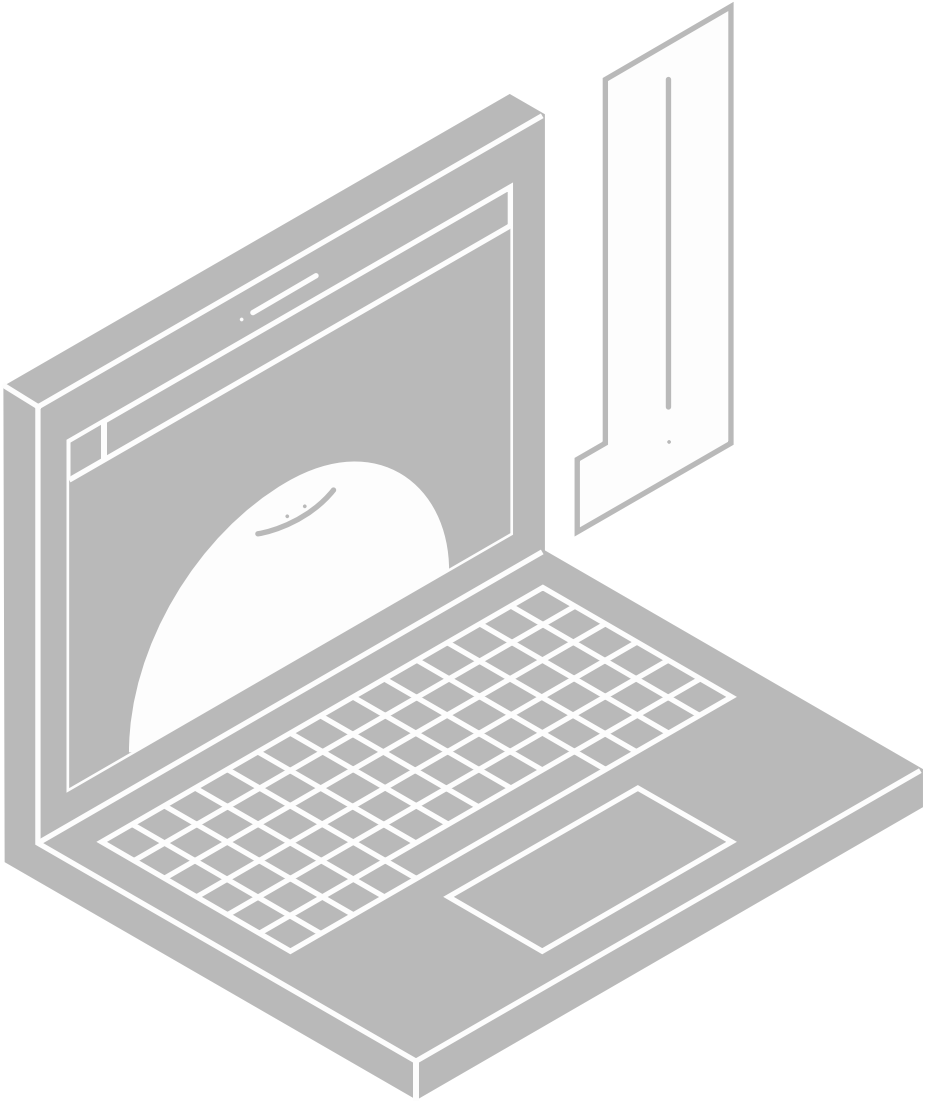
tell : 01031086418

kakao ID : belle1130

MYSQL

Reference Book





MYSQL

Reference Book

1. 데이터 베이스

컴퓨터 시스템의 정보 또는 데이터들의 집합을 말합니다.

```
create database 파일이름;
```

```
mysql> create database firsdb;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| firsdb             |
| freeive            |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.00 sec)
```

```
show database;
```

```
mysql> drop database firsdb;
Query OK, 0 rows affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| freeive            |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

```
drop database 파일이름;
```

```
mysql> drop database firsdb;
Query OK, 0 rows affected (0.01 sec)
```

```
use 데이터베이스 파일이름;
```

2. 테이블

테이블은 데이터베이스 내에 정보를 담는 역할을 합니다.

```
create table myMember5(
myMemberID int unsigned auto_increment comment '고객의 고유번호',
useID varchar(15) not null comment '고객 아이디',
uname varchar(10) not null comment '고객명',
upassword varchar(30) not null comment '고객 비밀번호',
email varchar(30) not null comment '고객의 이메일 주소',
birthDay char(10) not null comment '고객의 생일',
gender enum('m','w','x') default 'x' comment '고객 성별 m은 남성 w은 여성 x는 선택하지 않음',
regTime datetime not null comment '회원가입 시간',
primary key(myMemberID))
charset=utf8 comment='고객 정보 테이블';
```

<-- 테이블 생성하기

create database이름 테이블명 (
필드명, 데이터형, 필드 설명
) 인코딩(언어) 설정 테이블설명

< 테이블 목록보기 >

```
show 테이블명(tables);
```

```
drop table 명;
```

< 테이블 보기 >

```
desc 테이블 명;
```

테이블 목록보기 -->

```
| Tables_in_myfirst |
+-------------------+
| mymember          |
+-------------------+
1 row in set (0.02 sec)

mysql> desc mymember
    -> ;
+------------+------------------+------+-----+---------+----------------+
| Field      | Type             | Null | Key | Default | Extra          |
+------------+------------------+------+-----+---------+----------------+
| myMemberID | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| useID      | varchar(15)      | NO   |     | NULL    |                |
| uname      | varchar(10)      | NO   |     | NULL    |                |
| upassword  | varchar(30)      | NO   |     | NULL    |                |
| email      | varchar(30)      | NO   |     | NULL    |                |
| birthDay   | char(10)         | NO   |     | NULL    |                |
| gender     | enum('m','w','x') | YES  |     | x       |                |
| regTime    | datetime         | NO   |     | NULL    |                |
+------------+------------------+------+-----+---------+----------------+
8 rows in set (0.05 sec)

mysql>
```

3. 테이블 필드 수정하기

필드는 DB Table에서 가로로 표시한 레코드 (Record)에 대한 개별적인 속성값을 표시한다.

< 필드 추가 하기 >

```
ALTER TABLE 테이블 명 추가할 필드명
ALTER TABLE mymember2 ADD phone varchar(13) not null comment '고객의 연락처'

//위치 설정하기
ALTER TABLE mymember2 ADD phone varchar(13) not null comment '고객의 연락처' AFTER;
```

```
ALTER TABLE mymember2 DROP phone
```

<-- 필드 삭제하기 -->

```
ex)
ALTER TABLE mymember2 DROP uname;
desc mymember2;
ALTER TABLE mymember2 DROP useID;
desc my member2;
```

```
ALTER TABLE 테이블명 MODIFY upassward varchar(15) not null comment '패스워드'
desc mymember2;
```

<-- 필드 수정 하기

4. 테이블에 데이터 입력하기

테이블에서 전체나, 부분적으로 검색하고 추가할때 사용한다.

< 데이터 입력하기 >

INSERT INTO 테이블명(필드명)VALUES(입력할 데이터)

```
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO mymember4(useID, uname, upassword, email, birthDay, gender, regTime) VALUES('min','민','1234','min@naver.com','1997-03-11','w',now());
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO mymember4(useID, uname, upassword, email, birthDay, gender, regTime) VALUES('heemwon','힘원','1234','heemwon@gmail.com','1994-06-03','w',no
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO mymember4(useID, uname, upassword, email, birthDay, gender, regTime) VALUES('heemwon','힘원','1234','heemwon@gmail.com','1994-06-03','w',no
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO mymember4(useID, uname, upassword, email, birthDay, gender, regTime) VALUES('paneoy','hyunseo','1234','paneoy@gmiail.com','1996-04-18','w',
Query OK, 1 row affected (0.03 sec)

mysql> SELECT * FROM mymember4;
+------------+-------+-------+----------+----------------+------------+-------+---------------------+
| myMemberID | useID | uname | upassword | email          | birthDay   | gender | regTime             |
+------------+-------+-------+----------+----------------+------------+-------+---------------------+
|          1 | webs  | 웹스  | 1234     | web@naver.com  | 1999-02-03 | m      | 2020-12-28 11:57:57 |
|          2 | lrrock | 원락 | 1234     | zxcv1685@naver.com | 1996-03-02 | m      | 2020-12-28 12:01:31 |
|          3 | min   | 민    | 1234     | min@naver.com  | 1997-03-11 | w      | 2020-12-28 12:02:03 |
|          4 | young | 영    | 1234     | young@naver.com | 1998-12-18 | w      | 2020-12-28 12:11:26 |
|          5 | min   | 민    | 1234     | min@naver.com  | 1997-03-11 | w      | 2020-12-28 12:11:39 |
|          6 | heemwon | 힘원 | 1234     | heemwon@gmail.com | 1994-06-03 | w      | 2020-12-28 12:11:47 |
|          7 | heemwon | 힘원 | 1234     | heemwon@gmail.com | 1994-06-03 | w      | 2020-12-28 12:12:20 |
|          8 | paneoy | hyunseo | 1234     | paneoy@gmiail.com | 1996-04-18 | w      | 2020-12-28 12:12:31 |
+------------+-------+-------+----------+----------------+------------+-------+---------------------+
8 rows in set (0.00 sec)

mysql>
```

< 데이터 불러오기 >

```
//회원아이디 1번 검색하기
SELECT * FROM mymember4 WHERE myMemberID = 1;

//회원 이메일중에 naver가 있는 메일 검색하기
SELECT * FROM mymember4 WHERE email LIKE 'naver';

//차이점
SELECT * FROM mymember4 WHERE email LIKE '%naver';

SELECT * FROM mymember4 WHERE email LIKE 'naver%';
뒤에 어떤 단어가 있어도 검색이 된다.
```

```
mysql> SELECT * FROM mymember4 myMemberID = 1
    -> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server v
mysql> SELECT * FROM mymember4 WHERE myMemberID = 1;
+------------+-------+-------+----------+----------------+------------+-------+---------------------+
| myMemberID | useID | uname | upassword | email          | birthDay   | gender | regTime             |
+------------+-------+-------+----------+----------------+------------+-------+---------------------+
|          1 | webs  | 웹스  | 1234     | web@naver.com  | 1999-02-03 | m      | 2020-12-28 11:57:57 |
+------------+-------+-------+----------+----------------+------------+-------+---------------------+
1 row in set (0.04 sec)

mysql>
```

5. 테이블 데이터 수정하기

입력된 테이블의 데이터 값을 변경하거나 레코드를 삭제할 때 사용한다.

< 데이터 값 변경하기 >

```
//회원 아이디1번 핸드폰 번호 변경
UPDATE mymember4 SET phone = '010-1234-5678' WHERE myMemberID = 1;
```

```
11 rows in set (0.00 sec)

mysql> UPDATE mymember4 SET phone ='010-9999-2222' WHERE myMemberID = 1;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM mymember4;
+-------------+---------+---------+----------+---------------+-------------------+------------+---------+---------------------+
| myMemberID | useID   | uname   | upassword | phone         | email             | birthDay   | gender | regTime             |
+-------------+---------+---------+----------+---------------+-------------------+------------+---------+---------------------+
|          1 | webs    | 웹스    | 1234     | 010-9999-2222 | web@naver.com     | 1999-02-03 | m      | 2020-12-28 11:5
|          2 | lrrock  | 원록    | 1234     |               | zxcv1685@naver.com | 1996-03-02 | m      | 2020-12-28 12:0
|          3 | min     | 민      | 1234     |               | min@naver.com     | 1997-03-11 | w      | 2020-12-28 12:0
|          4 | young   | 영      | 1234     |               | young@naver.com   | 1998-12-18 | w      | 2020-12-28 12:1
|          5 | min     | 민      | 1234     |               | min@naver.com     | 1997-03-11 | w      | 2020-12-28 12:1
|          6 | heemwon | 희원    | 1234     |               | heemwon@gmail.com | 1994-06-03 | w      | 2020-12-28 12:1
|          7 | heemwon | 희원    | 1234     |               | heemwon@gmail.com | 1994-06-03 | w      | 2020-12-28 12:1
|          8 | paneoy  | hyunseo | 1234     |               | paneoy@gmail.com  | 1996-04-18 | w      | 2020-12-28 12:1
|          9 | hong    | 홍기영  | 123456   |               | hgy@naver.com     | 2019-12-28 | m      | 2020-12-28 12:1
|         10 | young   | 채영    | 1234     |               | young@naver.com   | 1995-12-28 | w      | 2020-12-28 12:1
|         11 | boram   | 보람    | 1234     |               | br@naver.com      | 1999-04-05 | w      | 2020-12-28 12:1
+-------------+---------+---------+----------+---------------+-------------------+------------+---------+---------------------+
11 rows in set (0.00 sec)

mysql> UPDATE mymember4 SET phone = 010-9999-2222 WHERE myMemberID = 1;
```

< 데이터 값 삭제하기 >

```
DELETE FROM mymember5 WHERE mymemberID = 5;

//IN()
DELETE FROM mymember5 WHERE mymemberID IN(1,2,3);
```

```
mysql> DELETE FROM mymember5 WHERE mymemberID = 5;
Query OK, 1 row affected (0.04 sec)

mysql> SELECT * FROM mymember5;
+-------------+---------+---------+---------+---------+---------+---------+---------+---------+
| myMemberID | useID   | phone   | uname   | upassword | email   | birthDay | gender | regTime |
+-------------+---------+---------+---------+---------+---------+---------+---------+---------+
|          1 | min     | 010-2222-3333 | 민      | 1234    | min@naver.com | 1997-03-11 | w      | 2020-12-29 09:53:39 |
|          2 | young   | 010-2222-3333 | 영      | 1234    | young@naver.com | 1998-12-18 | w      | 2020-12-29 09:53:39 |
|          3 | jangsoewon | 010-2222-3333 | 스원    | 1234    | thldnjs@naver.com | 1996-10-16 | w      | 2020-12-29 09:53:39 |
|          4 | boram   | 010-2222-3333 | 보람    | 1234    | br@naver.com | 1999-04-05 | w      | 2020-12-29 09:53:39 |
|          6 | hong    | 010-2222-3333 | 홍기영  | 123456  | hgy@naver.com | 2019-12-28 | m      | 2020-12-29 09:53:40 |
|          7 | jjung   | 010-2222-3333 | 성현    | 1234    | jjung@naver.com | 2000-08-04 | w      | 2020-12-29 09:53:40 |
+-------------+---------+---------+---------+---------+---------+---------+---------+---------+
6 rows in set (0.00 sec)

mysql> DELETE FROM mymember5 WHERE mymemberID IN(1,2,3);
Query OK, 3 rows affected (0.06 sec)

mysql> SELECT * FROM mymember5;
+-------------+---------+---------+---------+---------+---------+---------+---------+---------+
| myMemberID | useID   | phone   | uname   | upassword | email   | birthDay | gender | regTime |
+-------------+---------+---------+---------+---------+---------+---------+---------+---------+
|          4 | boram   | 010-2222-3333 | 보람    | 1234    | br@naver.com | 1999-04-05 | w      | 2020-12-29 09:53:39 |
|          6 | hong    | 010-2222-3333 | 홍기영  | 123456  | hgy@naver.com | 2019-12-28 | m      | 2020-12-29 09:53:40 |
|          7 | jjung   | 010-2222-3333 | 성현    | 1234    | jjung@naver.com | 2000-08-04 | w      | 2020-12-29 09:53:40 |
+-------------+---------+---------+---------+---------+---------+---------+---------+---------+
3 rows in set (0.00 sec)
```

5. 테이블 합치기(join)

각각 다른 테이블을 합칠 때 join문을 이용하여 합칠 수 있다.

< 테이블 합치기 >

```
create table review(
reviewID int unsigned auto_increment comment '리뷰의 고유번호',
memberID int unsigned comment '리뷰를 작성한 회원번호',
reviewContent tinytext comment '리뷰 내용',
regTime datetime not null comment '리뷰 작성 시간',
primary key(reviewID))
charset=utf8 comment='상품 리뷰 테이블';
```

```
//리뷰목록
INSERT INTO review(memberID, reviewContent, regTime) VALUES(2, '이쁩니다.', NOW());
INSERT INTO review(memberID, reviewContent, regTime) VALUES(3, '사지마세요.', NOW());
INSERT INTO review(memberID, reviewContent, regTime) VALUES(4, '가성비 갑.', NOW());
INSERT INTO review(memberID, reviewContent, regTime) VALUES(5, '좀 아쉬워요.', NOW());
INSERT INTO review(memberID, reviewContent, regTime) VALUES(6, '추천합니다.', NOW());
```

```
SELECT m.userName, r.reviewContent, r.regTime
FROM member m JOIN review r
ON (m.memberID = r.memberID);
```

< 테이터 불러오기 >

```
mysql> SELECT m.userName, r.reviewContent, r.regTime AS reviewDate FROM member m JOIN review r ON (m.memberID = r.memberID);
+----------+----------------------+---------------------+
| userName | reviewContent        | reviewDate          |
+----------+----------------------+---------------------+
| 영       | 이 신발 너무 이쁩니다. | 2020-12-29 10:39:34 |
| 시원     | 이 신발 사지마세요.    | 2020-12-29 10:40:47 |
| 보람     | 이 신발 가성비 갑.     | 2020-12-29 10:40:47 |
| 지원     | 이 신발 좀 아쉬워요.   | 2020-12-29 10:40:48 |
| 홍기영   | 이 신발 추천합니다.    | 2020-12-29 10:40:48 |
+----------+----------------------+---------------------+
5 rows in set (0.00 sec)
```

< 필드명 변경하여 표시하기 >

< left join 하기 >

```
SELECT m.userName, r.reviewContent, r.regTime
FROM member m LEFT JOIN review r ON (m.memberID = r.memberID);
```

```
mysql> SELECT m.userName, r.reviewContent, r.regTime FROM member m LEFT JOIN review r ON (m.memberID = r.memberID);
+----------+----------------------+---------------------+
| userName | reviewContent        | regTime             |
+----------+----------------------+---------------------+
| 영       | 이 신발 너무 이쁩니다. | 2020-12-29 10:39:34 |
| 시원     | 이 신발 사지마세요.    | 2020-12-29 10:40:47 |
| 보람     | 이 신발 가성비 갑.     | 2020-12-29 10:40:47 |
| 지원     | 이 신발 좀 아쉬워요.   | 2020-12-29 10:40:48 |
| 홍기영   | 이 신발 추천합니다.    | 2020-12-29 10:40:48 |
| 민       | NULL                 | NULL                |
| 정현     | NULL                 | NULL                |
+----------+----------------------+---------------------+
7 rows in set (0.03 sec)

mysql>
```

6. 집계함수

< GROUP BY지정한 열에서 같은 값을 가진 행끼리 한데 묶어 그룹화한 집합을 넘겨줄수 있다. >

```
SELECT class, avg(react) FROM CodingRecord WHERE class IN(1,2,3,4) GROUP BY class;
```

```
//리액트 점수가 80점 이상만 출력 (기능사 시험문제 예시)
SELECT class, sum(react) FROM CodingRecord WHERE class IN(1,2,3,4)
GROUP BY class HAVING sum(react) >= 80;
```

< ORDER BY를 추가하여 지정된 컬럼을 기준으로 오름차순(ASC) 또는 내림차순(DESC) 두 가지 정렬이 가능하다. >

```
//내림차순
SELECT codingID, css FROM CodingRecord ORDER BY css DESC;
//오름차순
SELECT codingID, css FROM CodingRecord ORDER BY css ASC;
//반별로
SELECT class, sum(javascript) FROM CodingRecord WHERE class IN(1,2,3)
GROUP BY class ORDER BY sum(javascript) DESC;
```

< LIMIT을 이용하여 Select 문장에서 Limit 키워드를 사용하면 테이블 데이터 조회 시 한계를 지정할 수 있다. >

```
//제한
SELECT * FROM CodingRecord LIMIT 3;

SELECT * FROM CodingRecord ORDER BY html DESC LIMIT 5;
```