



# [스파르타코딩클럽] 프론트엔드의 꽃, 리액트 - 1주차



매 주차 강의자료 시작에 PDF파일을 올려두었어요!

## ▼ PDF 파일

### [수업 목표]

1. NVM으로 노드 버전을 관리할 수 있다.
2. npm과 yarn으로 react 패키지를 설치할 수 있다.
3. CRA(create-react-app)을 사용해 React를 설치한다.
4. React에서 쓰는 태그(JSX)를 사용해서 화면을 그릴 수 있다.
5. 클래스형 컴포넌트 vs 함수형 컴포넌트! 차이점을 안다.
6. 화면을 예쁘게 꾸미는 3가지 방법(CSS, SCSS, styled-component)중 CSS를 살펴본다.

### [목차]

- 00. 프론트엔드의 꽃, 리액트 강의는 앞으로,
- 01. 필수 프로그램 설치
- 02. 기초 체력을 쌓아보자! (1)
- 03. 기초 체력을 쌓아보자! (2)
- 04. 기초 체력을 쌓아보자! (3)
- 05. 첫 React 프로젝트 만들기
- 06. JSX
- 07. JSX 사용법
- 08. Quiz\_간단한 텍스트 입력 화면 만들어보기
- 09. Component (1)
- 10. Component (2)
- 11. Component (3)
- 12. Quiz\_버킷리스트 페이지를 만들어봅시다!
- 14. 화면을 예쁘게! React에서 CSS 사용하기
- 15. 끝 & 숙제 설명
- 16. 1주차 숙제 답안 코드



모든 토글을 열고 닫는 단축키

Windows : **Ctrl** + **alt** + **t**

Mac : **⌘** + **⌥** + **t**

## 00. 프론트엔드의 꽃, 리액트 강의는 앞으로,

- 우리는 앞으로 이렇게 배울 거예요!
  1. 오류를 많이 낼 거예요! 빨간 화면에 익숙해지시도록요!
  2. 모르는 단어를 잔뜩 들을 거예요! 스스로 공부하기 위한 좋은 키워드가 되어줍니다.
  3. 만든 코드를 주저없이 버리고 새로 시작할 거예요!

- 질문할 때는,

⚠ 그냥 모르겠어요! 라고 하시는 것보다, 어디를 모르는지 알려주시면 더 자세히, 빨리, 필요한 부분만 설명 드릴 수 있어요!

아래 세 가지를 함께 튜터에게 알려주시면 좋답니다 😊

아래 세 가지 항목을 잘 모르신다구요? 그럴 땐 그냥 도와주세요!라고 하세요! 괜찮아요!

1. 어떤 운영체제를 쓰는지, (윈도우10, 맥 등 간략하게 표기해주셔도 됩니다!)
2. 오류가 났다면 어떤 오류 메시지가 나오는지, (화면을 캡처해주세요!)
3. 어떤 것을 하고 싶으신지

## 01. 필수 프로그램 설치

- Github Desktop ([다운로드 링크](#)).
- Github 가입하기 ([회원가입 링크](#))
- Visual Studio Code 설치 ([다운로드 링크](#))
- (Windows 사용자만) Git bash ([설치 참고 링크](#))

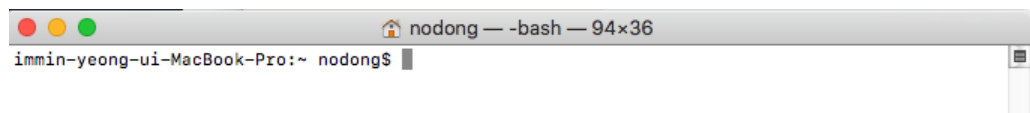
### ▼ NVM 설치

- Windows : ([다운로드 링크](#)), ([설치 참고 링크](#))
- Mac : ([설치 참고 링크](#))

### ▼ 설치 따라하기

#### ▼ 1. 내 터미널 확인하기

터미널을 켜고 내 터미널이 bash인지 zsh인 지 확인해주세요!(상단에 서 확인할 수 있어요.)



#### ▼ 2. nvm을 설치하자!

##### ▼ (추천) 브루를 통해 설치하기(homebrew를 통해서 설치하자)

1. homebrew를 설치하고([설치링크](#))
2. homebrew를 통해 nvm을 설치합시다.

```
brew install nvm
```

#### ▼ 바로 설치하기

아래 내용을 터미널에 복사 → 붙여넣기 → 엔터 해주세요

이때 git 명령어는 ~~ 이라면서 뭔가 설치하겠다고 할거예요! 설치해주세요.

```
sudo curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.1/install.sh | bash
```

#### ▼ 3. 설치 확인하기

아래 명령어를 입력해서 버전이 나오는 지 확인합니다. nvm: not found ~~가 나오면 4번으로 가서 환경변수를 설정하시고, 나오지 않는다면 설치 끝!

```
nvm --version
```

#### ▼ 4. 환경변수 설정하기

##### 1. 설정 파일 열기

vi 에디터로 설정 파일을 열어줍니다.

1. 터미널이 bash일 경우,

```
vi ~/.bash_profile
```

2. 터미널이 zsh일 경우,

```
vi ~/.zshrc
```

##### 2. 환경 변수를 추가하자 (잘못 만지면 안되는 중요한 파일이에요!! 조심조심하기! 기존에 있는 내용 지우면 절대절대절대 안됩니다!)

1. vi 에디터로 파일을 열면 방향키만 움직일 수 있을거예요!

**i** 키를 눌러서 입력 모드로 전환합니다.

2. 방향키를 아래로 쪽쪽 내리고 맨 밑에서 엔터 두번!

3. 아래 내용을 복사하고 붙여넣어줍니다.

```
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
```

##### 4. 저장하고 나가기

1. **esc** 키를 누르고,

2. **:wq** 를 입력합니다. (그리고 엔터!)

5. 추가한 내용을 적용하자!

1. 터미널이 bash일 경우,

```
source ~/.bash_profile
```

2. 터미널이 zsh일 경우,

```
source ~/.zshrc
```

6. 설치를 확인합니다.

```
nvm --version
```

## 02. 기초 체력을 쌓아보자! (1)

**A** 우리는 React의 깊은 부분까지 파보는 게 아니라, 꼭 필요한 부분만 배울 거예요!  
리액트를 설치하는 것부터 다른 사람들에게 내가 만든 웹 사이트를 보여주는 것까지 필요한 것을 배웁니다.

이 강의에서 중요한 건 **멋진 코드 짜기**보다 **내게 필요한 것을 찾는 능력 기르기**입니다. 마지막까지 끈질기게 해봅시다! 🙌

▼ 1) 서버와 클라이언트, 서버리스란?

▼ 1. 복습하자!

👉 이미 웹개발 종합반에서 서버와 클라이언트에 대해 배우신 분들은 아주 익숙하실 거예요. 아직 잘 모르셔도 괜찮습니다! 저랑 가볍게 한 번 훑어볼게요! 😊

👉 웹의 동작 개념 자료

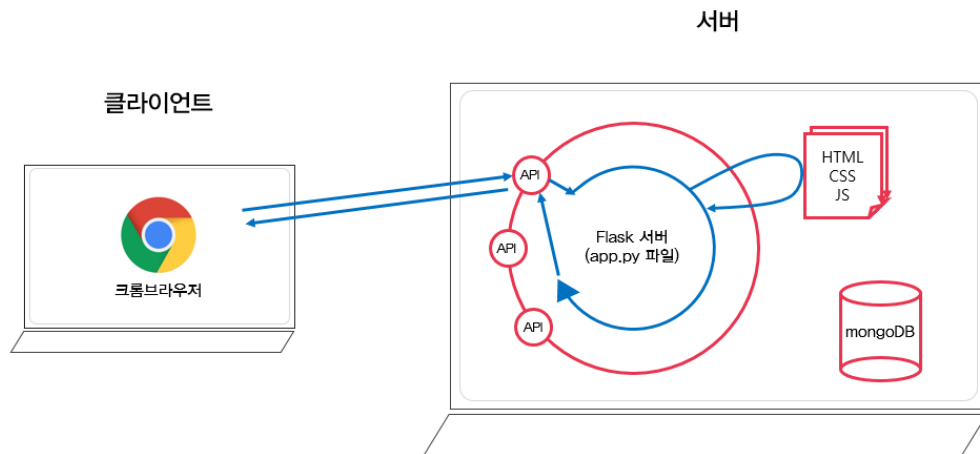
▼ 1. 웹의 동작 개념 (HTML을 받는 경우)

👉 네! 우리가 보는 웹페이지는 모두 서버에서 미리 준비해두었던 것을 "받아서", "그려주는" 것입니다. 즉, 브라우저가 하는 일은 1) 요청을 보내고, 2) 받은 HTML 파일을 그려주는 일 뿐이죠.

👉 근데, 1)은 어디에 요청을 보내냐구요? 좋은 질문입니다. 서버가 만들어 놓은 "API"라는 창구에 미리 정해진 약속대로 요청을 보내는 것이랍니다.

예) `https://naver.com/`

→ 이것은 "naver.com"이라는 이름의 서버에 있는, "/" 창구에 요청을 보낸 것!



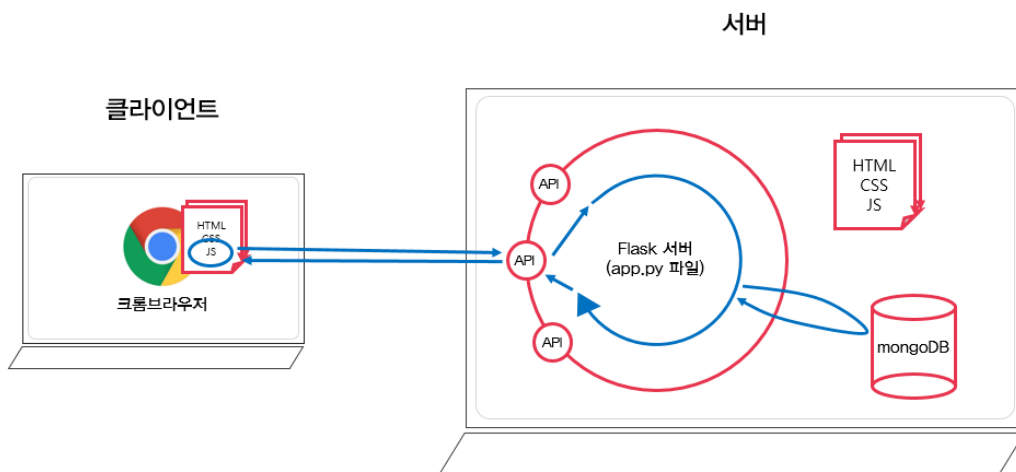
## ▼ 2. 웹의 동작 개념 (데이터만 받는 경우)

☞ 앳, 그럼 항상 이렇게 HTML만 내려주냐구요?  
아뇨! 데이터만 내려 줄 때가 더~ 많아요.

사실 HTML도 줄글로 쓰면 이게 다 '데이터' 아닌가요?

☞ 자, 공연 티켓을 예매하고 있는 상황을 상상해봅시다!  
좌석이 차고 꺼질때마다 보던 페이지가 리프레시 되면 난감하겠죠??

이럴 때! 데이터만 받아서 받아 끼우게 된답니다.



☞ 데이터만 내려올 경우는, 이렇게 생겼어요!  
(소곤소곤) 이런 생김새를 JSON 형식이라고 합니다.

```
openapi.seoul.go.kr:8088/6d4d7 x +
< > ↻ 주의 요함 | openapi.seoul.go.kr:8088/6d4d776b466c656533356a4b4b5872/json/RealtimeCityAir/1/99
{
  - RealtimeCityAir: {
    list_total_count: 25,
    - RESULT: {
      CODE: "INFO-000",
      MESSAGE: "정상 처리되었습니다"
    },
    - row: [
      - {
        MSRDT: "202004241900",
        MSRRGN_NM: "도심권",
        MSRSTE_NM: "중구",
        PM10: 44,
        PM25: 20,
        O3: 0.039,
        NO2: 0.02,
        CO: 0.4,
        SO2: 0.003,
        IDEX_NM: "보통",
        IDEX_MVL: 59,
        ARPLT_MAIN: "PM10"
      },
      - {
        MSRDT: "202004241900",
```

### ▼ 3. 그래서 서버랑 클라이언트가 뭐라고?

👉 클라이언트는 우리가 웹사이트를 보는 도구(휴대폰, PC, 아x패드 ...)가 모두 클라이언트입니다. 서버는 우리가 보는 웹사이트에 뿌려줄 것(html이나 데이터)을 만들어서 클라이언트에 전달해주는 친구겠죠!

### ▼ 2. 서버리스(serverless)란?

- 흔히 하는 오해 한 가지!

👉 Q. 서버리스? 서버가 없어요? DB도 백엔드 개발도 안해도 되는건가?  
A. 으악... 아니요! 서버는 필요하죠! 단지 **서버를 내가 만들 필요 없다!**가 맞아요.

- 서버... 있어? 그럼 서버리스가 뭘까?

👉 우리가 서버를 구성할 때 EC2를 사고, 서버 설정을 해줬던 걸 기억하실 거예요! 그 설정을 미리 해둔 어떤 서버를 빌려다 쓰는 겁니다!

즉, 서버의 사양, 네트워크 설정 같은 게 미리 되어 있는 서버를 빌려 쓰니, 인프라 작업을 내가 안 해도 된다는 겁니다.

+) 자세한 개념은 4주차에 예제와 함께 배울 거예요.  
오늘은 **서버리스는 백엔드리스가 아니라는 것만** 알고 있으면 굳!! 👍

### ▼ 3. 1~5주차에 배울 순서!

- 1주차: React 개발 환경을 구성하는 방법, JSX, Component

👉 우리가 만들 웹사이트가 어떻게 동작하는지 웹사이트의 원리와 서버리스 환경에 대해 이론을 조금 알아보고,  
NVM을 사용하는 방법과 CRA로 React 프로젝트를 만드는 방법을 배울거예요.  
+) JSX 이론과 컴포넌트를 만들고 쓰는 법을 맛볼 거예요.

- 2주차: Component 가지고 놀기, event listener, react hook

👉 리액트를 잘 쓰려면 꼭 알아야 하는 컴포넌트 라이프 사이클을 배우고 본격적으로 컴포넌트를 가지고 놀아볼 거예요. 😊

마우스 클릭, 키보드 입력, 터치 등 웹 사이트에서 일어나는 이벤트를 다루는 방법을 배울 거예요.  
(프론트엔드가 줄 수 있는 사용성이란 마력...함께 하시죠!)  
+) React hook을 다뤄볼 거예요!

- 3주차: Redux, React-router

👉 리액트가 데이터를 어떻게 관리하는지 알아볼 거예요.  
리액트에서의 페이지 전환도 알아볼 거예요. (화면이 깜빡거리지 않고 이동하는 스무스한 전환, 기대되죠!)

- 4주차: keyframe, firebase, firebase + React

👉 클릭 한 번에 바로바로 색이 바뀌고, 이미지가 롤링되면 재미없죠? 색은 서서히 바꾸고 이미지는 부드럽게 돌아가게 해주는 애니메이션 효과를 배울 거예요.

👉 백엔드 환경을 firebase로 만들고 React에서 불러오는 방법을 알아볼 거예요.

- 5주차: React 콤수, AWS S3로 배포하기

👉 리액트에서 개발을 더 편하게 해줄 콤수 몇 가지를 배울 거예요.  
우리가 만든 퀴즈 페이지를 친구들에게 공유할 수 있도록 AWS S3에 배포도 해봅니다.

## ▼ 2) 안녕 DOM!

👉 DOM을 아시나요?  
DOM은 html 단위 하나하나를 객체로 생각하는 모델입니다. 예를 들면, 'div태그'라는 객체는 텍스트 노드, 자식 노드 등등, 하위의 어떤 값을 가지고 있겠죠? 이런 구조를 트리 구조라고 합니다.  
네, 맞습니다! DOM이 트리구조란 소리입니다.

## 03. 기초 체력을 쌓아보자! (2)

- ⚠️ 우리가 배울 React는 javascript 라이브러리입니다.  
리액트를 다루기 위해 수업에 필요한 만큼만 ES6를 조금 알아볼까요! 🙌

**ES6**가 뭐냐구요?

자바스크립트 표준 문법 중 하나로 가장 보편화된 친구라고 생각해주세요!

### ▼ 3) 자바스크립트 알면 재미있는 기초 이론



#### 여기서 잠깐!

기초 이론은 지금 다 이해할 필요 없습니다.

5주 동안 천천히 익숙해질테니 지금은 맛만 본다고 생각하시고 편안하게 들어주세요. 😊

### ▼ (1) Class



#### 클래스란?

객체 지향 프로그래밍에서 클래스는 특정 객체를 생성하기 위해 변수와 함수를 정의하는 일종의 틀을 말해요. 객체를 정의하기 위한 상태와 함수로 구성되어 있죠!

객체 단위로 코드를 그룹화하고 쉽게 재사용하려고 사용합니다.

### ▼ -1) 클래스를 구성하는 것



클래스 안에는 객체를 정의하기 위한 상태(property)와 함수가 있다고 했죠!  
클래스가 어떻게 생겼는 지 보면서 클래스를 구성하는 것을 알아봐요.

```
class Cat {  
  // 생성자 함수  
  constructor(name) {  
    // 여기서 this는 이 클래스입니다.  
    this.name = name;  
  }  
  
  // 함수  
  showName(){  
    console.log(this.name);  
  }  
}  
  
let cat = new Cat('perl');  
cat.showName();  
console.log(cat);
```

- 생성자 함수: 클래스 인스턴스를 생성하고 생성한 인스턴스를 초기화(초기 값을 설정한다고 생각하세요!) 하는 역할을 합니다.
- 함수: 어떤 일을 하는 함수입니다.

### ▼ -2) 클래스를 상속하려면?



클래스를 상속한다는 건, 이미 만들어 둔 어떤 클래스를 가지고 자식 클래스를 만든다는 거예요.



```

class Cat {
  // 생성자 함수
  constructor(name) {
    // 여기서 this는 이 클래스입니다.
    this.name = name;
  }

  // 함수
  showName(){
    return this.name;
  }
}

// extends는 Cat 클래스를 상속 받아 온단 뜻입니다.
class MyCat extends Cat {
  // 생성자 함수
  constructor(name, age) {
    // super를 메서드로 사용하기
    super(name);
    this.age = age;
  }

  // 부모 클래스가 가진 것과 같은 이름의 함수를 만들 수 있습니다.
  // 오버라이딩한다고 해요.
  showName(){
    // super를 키워드로 사용하기
    return '내 고양이 이름은 '+super.showName()+ '입니다. ';
  }

  showAge(){
    console.log('내 고양이는 '+this.age+'살 입니다!');
  }
}

let my_cat = new MyCat('perl', 4);
my_cat.showName();
my_cat.showAge();

```

- super 키워드
  - 메소드로 사용할 수 있다.(constructor 안에서)
    - 부모의 constructor를 호출하면서 인수를 전달한다.
    - this를 쓸 수 있게 해준다.
  - 키워드로 사용할 수 있다.
    - 부모 클래스에 대한 필드나 함수를 참조할 수 있다.

## ▼ (2) let, const와 Scope



스코프(Scope)가 뭘까?

우리가 어떤 변수를 선언했을 때, 그 변수를 사용할 수 있는 유효범위를 스코프라고 불러요.

**변수에 접근할 수 있는 범위죠!**

- var: 함수 단위
- let: block 단위(변수: let으로 선언한 변수는 값이 변할 수 있습니다.)
- const: block 단위(상수: 한번 선언한 값은 바꿀 수 없습니다.)

```

function scope(){
  const a = 0;
  let b = 0;
  var c = 0;

  // {} 중괄호 안에 든 내용을 블록이라고 표현해요.

```

```

if(a === 0){
  const a = 1;
  let b = 1;
  var c = 1;
  console.log(a, b, c);
}
// 앗! c는 값이 변했죠?
// 그렇습니다. var는 함수 단위라서 if문 밖에서 선언한 값이 변했어요.
// let과 const로 선언한 값은 어떤가요? if문 안쪽 내용이 바뀔 내용에 영향을 끼치지 않죠?
console.log(a, b, c);
}

```

### ▼ (3) =과 ==과 ===

- =

☞ =는 할당을 뜻합니다.  
어떤 변수에 값을 할당할 때 써요.

- ==

☞ ==는 등차입니다.  
유형을 비교하지 않는 등차예요. 변수 값을 기반으로 비교합니다.  
(ex. 0 == "0"은 true를 반환합니다.)

- ===

☞ ===도 등차입니다!  
유형도 비교하는 등차예요. 엄격한 비교죠!  
(ex. 0 === "0"은 false를 반환합니다.)

### ▼ (4) Spread 연산자 (Spread 문법)

☞ 어떤 객체 안에 있는 요소들을 객체 바깥으로 꺼내주는 친구입니다.

```

let array = [1,2,3,4,5];
// ... <- 이 점 3개를 스프레드 문법이라고 불러요.
// 배열 안에 있는 항목들(요소들)을 전부 꺼내준다는 뜻입니다.
// 즉 [...array]은 array에 있는 항목을 전부 꺼내
// 새로운 배열([] => 이 껍데기가 새로운 배열을 뜻하죠!)에 넣어주겠다 말입니다!
let new_array = [...array];

console.log(new_array);

```

### ▼ (5) 조건부 삼항 연산자

☞ 삼항 연산자는 if문의 단축 형태입니다.

사용법:  
**조건 ? 참일 경우 : 거짓일 경우**

```
let info = {name: "mean0", id: 0};

let is_me = info.name === "mean0"? true : false;

console.log(is_me);
```

## 04. 기초 체력을 쌓아보자! (3)

**A** 우리가 앞으로 정말정말(x100) 자주 쓸 Array의 내장 함수 4개를 미리 알아봅시다! 🙌

### ▼ 4) Array 내장 함수

#### ▼ (1) map

👉 map은 배열에 속한 항목을 변환할 때 많이 사용합니다.  
어떤 배열에 속한 항목을 원하는 대로 변환하고, 변환한 값을 **새로운 배열**로 만들어줍니다.  
즉, 원본 배열은 값이 변하지 않아요!

```
const array_num = [0, 1, 2, 3, 4, 5];

const new_array = array_num.map((array_item) =>{
  return array_item + 1;
});
// 새 배열의 값은 원본 배열 원소에 +1 한 값입니다.
console.log(new_array);
// 원본 배열은 그대로 있죠!
console.log(array_num);
```

#### ▼ (2) filter

👉 filter는 어떤 조건을 만족하는 항목들만 골라서 새 배열로 만들어주는 함수입니다.  
원본 배열은 변하지 않고, 원하는 배열을 하나 더 만들 수 있다니 (최고)죠!

```
const array_num = [0, 1, 2, 3, 4, 5];

// forEach(콜백함수)
const new_array = array_num.filter((array_item) => {
  // 특정 조건을 만족할 때만 return 하면 됩니다!
  // return에는 true 혹은 false가 들어가야 해요.
  return array_item > 3;
});

console.log(new_array);
```

#### ▼ (3) concat

👉 concat은 배열과 배열을 합치거나 배열에 특정 값을 추가해주는 함수입니다!  
**원본 배열은 변하지 않아요!**

```
const array_num01 = [0, 1, 2, 3];
const array_num02 = [3, 4, 5];

const merge = array_num01.concat(array_num02);

// 중복 항목(숫자 3)이 제거되었나요? 아니면 그대로 있나요? :)
console.log(merge);
```



**concat은 중복 항목을 제거해주지 않아요!**

다른 내장함수와 함께 사용해서 제거해야 합니다!

자바스크립트를 조금 다룰 줄 아는 분들을 위한 팁으로 제가 자주 사용하는 방법을 살짝 남겨둘게요.



아직 자바스크립트에 익숙하지 않으신 분들은 그냥 이렇게 있구나 하고 넘어가도 됩니다!

```
const array_num01 = [0, 1, 2, 3];
const array_num02 = [3, 4, 5];
// Set은 자바스크립트의 자료형 중 하나로,
// 중복되지 않는 값을 가지는 리스트입니다. :)!
// ... <- 이 점 3개는 스프레드 문법이라고 불러요.
// 배열 안에 있는 항목들(요소들)을 전부 꺼내준다는 뜻입니다.
// 즉 [...array_num01]은 array_num01에 있는 항목을 전부 꺼내
// 새로운 배열([] 이 겹데기가 새로운 배열을 뜻하죠!)에 넣어주겠다 말입니다!
const merge = [...new Set([...array_num01, ...array_num02])];

// 중복 항목(숫자 3)이 제거되었나요? 아니면 그대로 있나요? :)
console.log(merge);
```

#### ▼ (4) from



from은 쓰임새가 다양한 친구입니다. 😊

-1) 배열로 만들고자 하는 것이나 **유사배열**을 복사해서 새로운 배열로 만들 때

-2) 새로운 배열을 만들 때 (초기화한다고도 표현해요.)

주로 사용합니다!

**유사배열이 뭘까?**

[ 어떤 값들... ] 이 모양으로 생겼지만 배열의 내장 함수를 사용하지 못하는 친구들입니다. DOM nodelist같은 게 유사배열이에요.

```
// 배열화 하자!
const my_name = "mean0";
const my_name_array = Array.from(my_name);
console.log(my_name_array);

// 길이가 문자열과 같고, 0부터 4까지 숫자를 요소로 갖는 배열을 만들어볼게요.
const text_array = Array.from('hello', (item, idx) => {return idx});

console.log(text_array);

// 새 배열을 만들어 보자! (=> 빈 배열을 초기화한다고도 해요.)
// 길이가 4고, 0부터 3까지 숫자를 요소로 갖는 배열을 만들어볼게요.
const new_array = Array.from({length: 4}, (item, idx)=>{ return idx;});

console.log(new_array);
```

👉 우리가 다뤄본 6가지 내장 함수 외에도 엄청 엄청 많은 내장함수가 있어요!  
javascript 배열 내장함수로 검색해서 한 번 뭐가 있는 지 둘러보시면 정말 좋겠죠? 😎  
+) 특히 저희는 이번 강의에서 다루지 않지만, reduce()라는 내장함수는 정말 유용한 친구예요.  
꼭꼭 찾아보기!

## ▼ 5) Quiz\_Array 내장 함수

👉 크롬 브라우저를 열고 콘솔 창에서 해봅시다! 😊

### ▼ (1) 배열에서 고양이가 몇 마리인지 세기 - map으로 해보자!

👉 아래 for문을 map으로 바꿔봅시다.

```
const animals = ["강아지", "고양이", "햄스터", "강아지", "고양이", "고양이", "토끼"];

let count = 0;
for (let i = 0; i < animals.length; i++) {
  let animal = animals[i];
  if (animal === "고양이") {
    count += 1;
  }
}
console.log(count);
```

### ▼ 정답 확인하기

```
const animals = ["강아지", "고양이", "햄스터", "강아지", "고양이", "고양이", "토끼"];

let count = 0;
animals.map((animal) => {
  if (animal === "고양이") {
    count += 1;
  }
});
console.log(count);
```

### ▼ (2) 배열에서 filter로 해보자!

👉 고양이들만 새 배열에 넣어볼까요?  
아래 for문을 보고 filter로 바꿔봐요!

```
const animals = ["복색 강아지", "검정 고양이", "노란 햄스터", "강아지", "노랑 고양이", "고양이", "흰 토끼"];

let cats = [];
for (let i = 0; i < animals.length; i++) {
  let animal = animals[i];
  // indexOf는 파라미터로 넘겨준 텍스트가 몇 번째 위치에 있는 지 알려주는 친구입니다.
  // 파라미터로 넘겨준 텍스트가 없으면 -1을 반환해요!
  // 즉 아래 구문은 고양이라는 단어를 포함하고 있나? 라고 묻는 구문이지요!
  if (animal.indexOf("고양이") !== -1) {
    cats.push(animal);
  }
}
```

```

    }
  }
  console.log(cats);

```

#### ▼ 정답 확인하기

```

const animals = ["복슬 강아지", "검정 고양이", "노란 햄스터", "강아지", "노랑 고양이", "고양이", "흰 토끼"];

let cats = animals.filter((animal) => {
  return animal.indexOf("고양이") !== -1;
});

console.log(cats);

```

#### ▼ (3) 두 배열을 합쳐보자! - concat으로 해보자!

👉 아래 두 배열을 하나로 합쳐보세요!

```

const dogs = ["검은 강아지", "노란 강아지", "흰 강아지"];
const cats = ["검은 고양이", "복슬 고양이", "노란 고양이"];

```

#### ▼ 정답 확인하기

```

const dogs = ["검은 강아지", "노란 강아지", "흰 강아지"];
const cats = ["검은 고양이", "복슬 고양이", "노란 고양이"];

const animals = dogs.concat(cats);

console.log(animals);

```

👉 어떠셨나요? 배열의 내장 함수 사용하기 어렵지 않죠? 😎

#### ▼ 6) 특정 과일 갯수 세기 문제

##### ▼ 배열에서 특정 원소 갯수 구하기 - map 를 함수를 사용해서 해결하기

👉 다음에서 '딸기'는 몇 개일까? - 이번엔 자바스크립트 콘솔창에서!

```

let fruit_list = ['사과', '감', '감', '배', '포도', '포도', '딸기',
  '포도', '감', '수박', '딸기'];

let count = 0;
for (let i = 0; i < fruit_list.length; i++) {
  let fruit = fruit_list[i];
  if (fruit == '딸기') {
    count += 1;
  }
}
console.log(count);


```

#### ▼ map 해답

```
let fruit_list = ['사과', '감', '감', '배', '포도', '포도', '딸기', '포도', '감', '수박', '딸기']
let count = 0;
fruit_list.map((f)=>{
  if(f == "딸기") count += 1
})

console.log(count)
```


## 05. 첫 React 프로젝트 만들기

 NVM과 VSCode를 설치하셨나요? 안하셨다면 지금 바로 설치해주세요!

- 폴더부터 만들게요 😊
  - [sparta\_react]라는 폴더를 만들어주세요!
  - windows: C드라이브 아래에!
  - osx: macintosh HD → 사용자 → [내 컴퓨터 이름] 아래에!

### ▼ 7) NVM으로 노드 버전을 관리해보자

- NVM(Node Version Manager)을 왜 써야할까?

 nvm은 Node.js의 버전 관리자입니다.  
우리 컴퓨터에 node를 설치하는 툴인데 수많은 버전을 마음대로 골라 설치할 수 있게 해주는 멋진 친구입니다. 😊

시스템(우리 컴퓨터)에 Node.js를 직접 설치하다보면 다른 버전을 설치하게 되는 경우가 많은데, 여러 버전의 Node.js를 관리하는 건 굉장히 귀찮은 일이지. 😞

nvm을 설치하고 설치한 nvm을 통해 node를 설치하면 나중에 생길 귀찮음을 방지할 수 있습니다!


- nvm 설치 확인하기

VSCode에서 터미널을 열고 아래와 같이 타이핑 해봅니다. nvm이 잘 설치 되었다면 설치한 nvm 버전이 나올 거예요.

```
nvm --version
```

```
immin-yeong-ui-MacBook-Pro:react_example nodong$ nvm --version
0.33.1
```

- nvm으로 노드 설치하기

 이제 Node.js를 설치해봅시다!  
노드 공식 사이트에서 안정적인 버전(버그가 적은 버전! LTS라고 불러요.)을 확인해보고 LTS 버전을 설치해봅시다!

공식 사이트에는 최신 LTS 버전과 최신 버전을 바로 안내해주고 있습니다.  
(아래 이미지의 초록 박스 확인)



Node.js®는 **Chrome V8 JavaScript 엔진**으로 빌드된 JavaScript 런타임입니다.

#BlackLivesMatter

New security releases are available

다운로드 - macOS (x64)

12.18.4 LTS

안정적, 신뢰도 높음

14.13.0 현재 버전

최신 기능

[다른 운영 체제](#) | [변경사항](#) | [API 문서](#)

[다른 운영 체제](#) | [변경사항](#) | [API 문서](#)

LTS 일정은 [여기서](#) 확인하세요.

12.18.4가 최신 LTS네요! 아래 명령어를 입력해서 node.js를 설치해볼까요?

12.18.4와 14.13.0 두 버전을 다 설치해봅시다.

```
nvm install [설치할 버전]
```

설치가 끝났다면 터미널에 아래 명령어를 입력해서 잘 설치 되었는 지 확인해봅시다!

```
nvm ls # nvm으로 설치한 노드 버전 리스트 확인 명령어  
node -v # 노드 버전 확인 명령어
```



```

immin-yeong-ui-MacBook-Pro:react_example nodong$ nvm ls
v10.15.0
v10.16.3
v12.18.4
-> v14.13.0
system
default -> lts/* (-> v12.18.4)
node -> stable (-> v14.13.0) (default)
stable -> 14.13 (-> v14.13.0) (default)
iojs -> N/A (default)
lts/* -> lts/erbium (-> v12.18.4)
lts/argon -> v4.9.1 (-> N/A)
lts/boron -> v6.17.1 (-> N/A)
lts/carbon -> v8.17.0 (-> N/A)
lts/dubnium -> v10.22.1 (-> N/A)
lts/erbium -> v12.18.4
immin-yeong-ui-MacBook-Pro:react_example nodong$ node -v
v14.13.0
immin-yeong-ui-MacBook-Pro:react_example nodong$

```

👉 두번 째로 설치한 14.13.0 버전에 화살표가 붙어 있는 게 보이시나요?  
 사용 중인 노드 버전을 표시해주는 거예요!  
 node -v를 입력해서 나온 버전과 같은 지 확인합니다.

+) nvm에서 사용 중인 노드 버전 바꾸기

⚠️ 앗! LTS는 12.18.4인데, 사용 중인 노드 버전을 바꾸고 싶을 땐 어떻게 할까요?  
 아래 명령어를 입력해 사용할 노드 버전을 바꾸고,  
 다시 node -v 명령어로 노드 버전을 확인해봅시다!

```
nvm use [사용할 노드 버전]
```

## ▼ 8) npm으로 yarn을 설치해보자

- NPM(Node Package Manager)은 무수히 많은 third-party 패키지를 활용할 수 있게 해줍니다!

💡 비슷한 친구로는 yarn이 있습니다.  
 둘 다 "프론트엔드 의존성"을 관리하기 위한 "패키지 매니저"입니다.  
 → "누가 만들어 놓은 좋은 것"(= 패키지)을 가져다 쓰기 편하게 도와줍니다.

👉 npm은 노드를 설치하면 함께 설치되서 따로 설치하지 않아도 됩니다!

- npm으로 yarn을 설치해보자!

아래 명령어를 입력해서 yarn을 설치해봅시다.

```

# npm으로 패키지를 설치할 때는 아래 명령어를 사용해요!
# 옵션은 필요한 경우에만 적어줍니다.
# npm install [옵션] [설치할 패키지 이름]

npm install -g yarn

```

```
# 이 명령어는 "npm으로 yarn을 컴퓨터 전체에 설치한다"는 뜻입니다.  
# -g 옵션은 컴퓨터 전체에서 쓸 수 있게 한다는 뜻입니다.
```

잘 설치되었다면 아래처럼 yarn -v 명령어로 yarn의 버전을 확인하실 수 있을거예요!

```
immin-yeong-ui-MacBook-Pro:react_example nodong$ npm install -g yarn  
  
> yarn@1.22.10 preinstall /Users/nodong/.nvm/versions/node/v12.18.4/lib/node_modules/yarn  
> ; (node ./preinstall.js > /dev/null 2>&1 || true)  
  
/Users/nodong/.nvm/versions/node/v12.18.4/bin/yarn -> /Users/nodong/.nvm/versions/node/v12.18.4/lib/node_modules/yarn/bin/yarn.js  
/Users/nodong/.nvm/versions/node/v12.18.4/bin/yarnpkg -> /Users/nodong/.nvm/versions/node/v12.18.4/lib/node_modules/yarn/bin/yarn.js  
+ yarn@1.22.10  
added 1 package in 0.904s  
immin-yeong-ui-MacBook-Pro:react_example nodong$ yarn -v  
1.22.10
```

👉 yarn으로 패키지를 설치할 때는?  
yarn으로 패키지를 설치할 때는 아래 명령어를 사용합니다. 명령어가 조금 더 직관적이죠!

```
yarn add [옵션] [설치할 패키지 이름]
```

## ▼ 9) CRA(create-react-app)으로 시작하는 리액트

- yarn으로 CRA를 설치하자!

```
# 옵션 global은 전역에 이 패키지를 깔겠다는 뜻입니다.  
yarn add global create-react-app
```

- CRA가 뭘까?

👉 React는 레고같은 친구입니다. 어린이들이 레고로 성을 만드는 것처럼 우리는 React로 웹사이트를 만들 거예요.

마트에서 레고를 보신 분들은 아시겠지만, 레고는 네모 블럭, 긴 블럭, 혹은 귀여운 캐릭터만 따로 구매할 수 있죠? 만들고 싶은 성 모양에 맞춰 누구는 네모 블럭만 잔뜩 살 것이고 누구는 성문과 대포까지 살 겁니다.

하나씩 구매하기 번거롭다면요? [해리x터 성만들기]같은 패키지를 사겠죠!

React도 마찬가지입니다! 우리가 웹사이트에 만들기 위해 필요한 것들을 하나씩 설치할 수 있습니다. (webpack, babel 같은 녀석들을 배워야 하지만요.)

**CRA(Create React App)는 웹사이트를 만들 때 필요한 것을 몽땅 때려넣어 만든 패키지입니다.** 레고의 해리x터 성만들기 같은 거라고 생각하시면 좋아요.

- 우리의 첫번째 리액트 프로젝트를 만들어요!

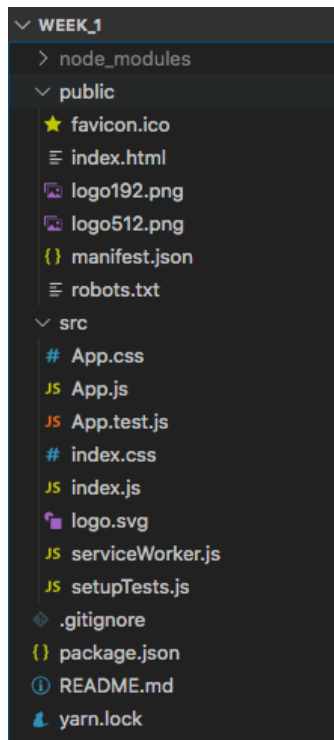
다시 터미널로 돌아가서 우리의 첫 리액트 프로젝트를 만들어 봅시다!

```
# yarn create react-app [우리의 첫 리액트 프로젝트 이름]  
# 우리가 설치한 create-react-app 패키지를 써서 프로젝트를 만들어요.  
# 주의! 꼭 sparta_react 폴더 경로에서 입력해주세요!  
yarn create react-app week-1
```

👉 React에서는 프로젝트를 앱이라고 불러요. 리액트 프로젝트와 리액트 앱은 같은 말이니, 편하신 쪽으로 말씀하시면 됩니다.

🔥 프로젝트가 생성이 되면, sparta\_react 폴더 아래에 week-1이라는 폴더가 생길 거예요. VSCode를 열어서, [폴더 열기] → sparta\_react → week-1 폴더를 선택해 열어주세요!

VSCode에서 폴더 구조를 확인해 봅시다.

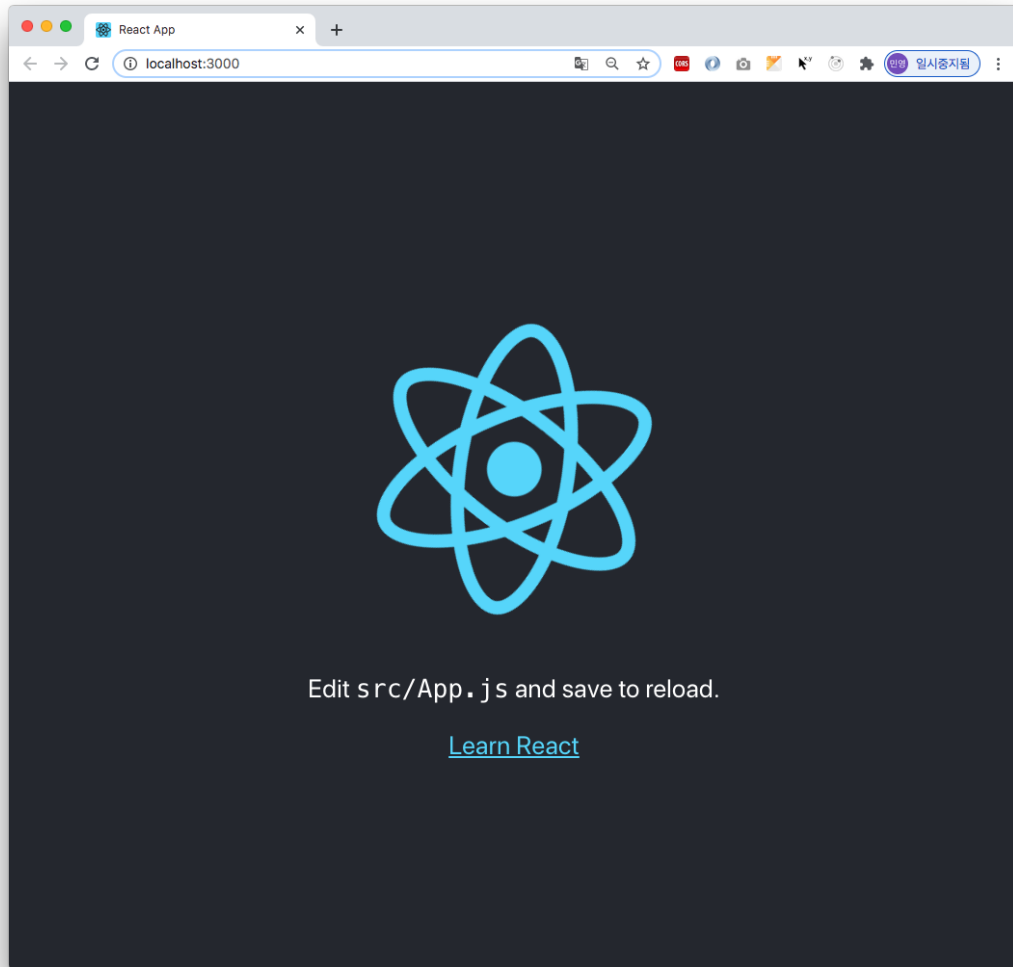


👉 지금은 모양이 이렇게 생겼지만 알아두시면 됩니다! 하나씩 천천히 배워가요!

설치가 끝났다면, 아래 명령어를 입력해서 우리의 첫 리액트 앱을 실행시켜봅시다!

```
cd week-1 # week-1 폴더로 이동합니다.  
yarn start
```

브라우저가 열리고 우리의 첫 리액트 앱이 실행되었습니다. 😊



## 06. JSX

### ▼ 10) JSX 가 뭘까?

🔥 VSCode에서 src폴더 아래의 App.js 파일을 열어봅시다.

자, App.js 코드를 뜯어보면서 jsx에 대해 알아보시다!

```
// 엇? 파이썬에서 뷰티풀스프를 불러올 때 본 것 같기도 하죠?
// *react*에서도 다른 패키지를 불러다 쓸 수 있습니다!
// import [패키지명] from [경로] 이 형식으로 불러와요.
import React from 'react';
// js 파일 뿐 아니라 이미지도 가능가능!
import logo from './logo.svg';
// css? 가능!
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
```


```


    <p>
      Edit <code>src/App.js</code> and save to reload.
    </p>
    <a
      className="App-link"
      href="https://reactjs.org"
      target=" blank"
      rel="noopener noreferrer"
    >
      Learn React
    </a>
  </header>
</div>
);
}

export default App;

```

- JSX

 리액트에서는 딱 하나의 html 파일만 존재합니다.  
(public 폴더 아래에 있는 index.html)

 그럼 리액트에서 어떻게 뷰를 그릴까요? App.js 파일에서 보이듯,  
**JSX 문법을 사용해서 React 요소를 만들고 DOM에 렌더링 시켜서** 그립니다.

- HTML을 품은 JS === JSX!

아래 같은 HTML 태그는 .js 파일 안에서 쓸 수 없어요.

```

<div>
  <h1>안녕하세요!</h1>
  <p>시작이 반이다!</p>
</div>

```


그래서 나온 게 JSX입니다.

자바스크립트 안에서 **html 태그같은 마크업**을 넣어 뷰(UI) 작업을 편하게 할 수 있죠!

```

const start_half = <div>
  <h1>안녕하세요!</h1>
  <p>시작이 반이다!</p>
</div>;

```

 어때요? 아직 아리송한가요?  
괜찮아요! 써보면 느낌이 올거니까!

## 07. JSX 사용법

### ▼ 11) JSX 사용법 훑어보기

👉 여기서부터는 따라해봅시다!

🔥 개발자 도구 여는 법! 기억하고 계시죠?  
개발자 도구를 열고 에러를 보는 거 잊지 마세요!  
(F12 혹은 [Cmd + Opt + i](맥), [Ctrl + Shift + i](윈도우)를 눌러 개발자 도구를 열 수 있어요.)

- JSX 규칙

- ▼ 1. 태그는 꼭 닫아주기

App.js 파일에서 실습합니다! (오류를 내면서 해보는 거예요!)

하이라이트 된 부분은 지워주세요.

```
JS App.js ×
src > JS App.js > App
1  import React from 'react';
2  import logo from './logo.svg';
3  import './App.css';
4
5  function App() {
6    return (
7      <div className="App">
8        <header className="App-header">
9          <img src={logo} className="App-logo" alt="logo" />
10         <p>
11           Edit <code>src/App.js</code> and save to reload.
12         </p>
13         <a
14           className="App-link"
15           href="https://reactjs.org"
16           target="_blank"
17           rel="noopener noreferrer"
18         >
19           Learn React
20         </a>
21       </header>
22     </div>
23   );
24 }
25
26 export default App;
27
```

```
// input 태그를 닫지 않고 넣어볼거예요!
function App() {
  return (
    <div className="App">
      <input type='text'>
    </div>
  );
}
```

## Failed to compile

```
./src/App.js
Line 9:11:  Parsing error: Unterminated JSX contents

   7 |     <div className="App">
   8 |       <input type="text">
>  9 |     </div>
      |         ^
    10 |   );
    11 | }
    12 |
```

This error occurred during the build time and cannot be dismissed.

JSX 문법에 맞지 않는다고 에러가 납니다! 아래처럼 /를 추가하고 브라우저를 새로고침 해봅시다.

```
<input type='text' />
```

### ▼ 2. 무조건 1개의 엘리먼트를 반환하기

이번엔 return 아래에 p태그를 하나 추가해볼까요?

```
return (
  <p>안녕하세요! 리액트 반입니다 :)</p>

  <div className="App">
    <input type='text' />
  </div>
);
```

## Failed to compile

```
./src/App.js
Line 9:5:  Parsing error: Adjacent JSX elements must be wrapped in an enclosing tag. Did you want a JSX fragment <>...</>?

   7 |     <p>안녕하세요! 리액트 반입니다 :)</p>
   8 |
>  9 |     <div className="App">
      |         ^
    10 |       <input type='text' />
    11 |     </div>
    12 |   );
```

This error occurred during the build time and cannot be dismissed.

앗! 또 에러가 나네요! 컴포넌트에서 반환할 수 있는 엘리먼트는 1개입니다. 아래와 같이 바꾸고 다시 새로고침 해보세요.

```
return (
  <div className="App">
    <p>안녕하세요! 리액트 반입니다 :)</p>
    <input type='text' />
  </div>
);
```

### ▼ 3. JSX에서 javascript 값을 가져오려면?

중괄호를 쓴다!

```
const cat_name = 'perl';
// return 부분만 복사해서 붙여넣고 크롬 브라우저로 돌아가 새로고침 해봅시다.
return (
```

```

    <div>
      hello {cat_name}!
    </div>
  );

```

👉 값을 가져올 때 뿐만 아니라, map, 삼항연산자 등 자바스크립트 문법을 JSX 안에 쓸 때도 {}를 이용할 수 있어요. 해볼까요? 😊

```

import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  const number = 1;

  return (
    <div className="App">
      <p>안녕하세요! 리액트 반입니다 :)</p>
      {/* JSX 내에서 코드 주석은 이렇게 씁니다 :) */}
      {/* 삼항 연산자를 사용했어요 */}
      <p>{number > 10 ? number+'은 10보다 크다': number+'은 10보다 작다'}</p>
    </div>
  );
}

export default App;

```

#### ▼ 4. class 대신 className!

처음부터 약간 거슬리던 친구가 있지 않나요?

```

<div className="App">

```

JSX로 작성하는 태그 내에서 클래스 명을 정해줄 땐 속성 값을 className으로 사용합니다. class대신에요! +)

그럼 id도 설마..? 하셨나요? id는 그냥 id로 씁니다.

#### ▼ 5. 인라인으로 style 주기

👉 html 태그에 스타일을 직접 넣던 거 기억하시나요? 거기에서 아주 조금 달라요! css 문법 대신 json 형식으로 넣어주면 끝!

#### HTML

```

<p style="color: orange; font-size: 20px;">orange</p>

```

#### JSX

```

// 중괄호를 두 번 쓰는 이유? 디셔너리도 자바스크립트니까요!
// 이렇게 쓰거나,
<p style={{color: 'orange', fontSize: '20px'}}>orange</p>

//혹은 스타일 디셔너리를 변수로 만들고 쓸 수 있어요!
function App() {
  const styles = {
    color: 'orange',
    fontSize: '20px'
  };
}

```



```

return (
  <div className="App">
    <p style={styles}>orange</p>
  </div>
);
}

```

## 08. Quiz\_간단한 텍스트 입력 화면 만들어보기

### ▼ 12) 🗨️ 간단한 텍스트 입력 화면 만들기

👉 JSX를 이용해서 간단한 텍스트 입력 화면을 만들어봅시다!

안녕하세요!

이름을 입력해주세요.

- 회색 박스 안에 타이틀, 선, 일반 텍스트, 그리고 인풋 박스가 들어갔습니다.
- 위에서 배운대로 App.js 안에서 CSS를 사용하여 위 이미지처럼 만들어보세요!

### ▼ 정답 코드

⚠️ 다 만드시고 확인해보세요!  
꼭 스타일이 똑같지 않아도 괜찮아요! 같은 모양이 나왔다면 🙌!

### ▼ 확인하기

```

import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  const styles = {
    border: '1px solid #eee',
    padding: '20px',
    display: 'flex',
    width: '100vw',
    maxWidth: '400px',
    margin: '30px auto',
    flexDirection: 'column'
  };

  return (
    <div className="App">
      <div style={styles}>
        <h1 style={{ color: 'green' }}>안녕하세요!</h1>

```

```

    <hr style={{width: '100%'}}/>
    <p style={{ textAlign: 'left' }}>이름을 입력해주세요.</p>
    <input type="text"/>
  </div>
</div>
);
}

export default App;

```

## 09. Component (1)

- Component는 클래스형과 함수형이 있습니다.  
생김새와 용도가 다른데, React가 발전을 거듭하며 이전 생김새만 다르다고 봐도 무방합니다.  
우리는 이 강의에서 두 가지를 다 써볼 거예요. 😊

### ▼ 13) Component란?

위에서 리엑트는 레고라고 말씀 드렸죠? 컴포넌트는 블록입니다!

- 웹 사이트를 조각내자!  
컴포넌트를 이해하고 잘 써먹으려면 웹 사이트를 잘 조각낼 줄 알아야 합니다.  
가령 우리가 아래와 같은 웹 사이트를 만든다고 생각해봅시다!

SPARTA! 온라인 오프라인 기업교육 내 강의실

속속 따라하다 보면  
나만의 앱이 똑딱

광고 붙이고, 출시까지 한번에  
앱만들기 어렵다는 건 다 옛날 얘기!

이터분석 웹개발 앱개발 온라인 전시회

1. 즉문즉답  
첫번째 가장 다른 것은 즉문즉답입니다

스파르타 웹/앱개발

왕초보 코딩교육의 명가

스파르타코딩클럽은 사전 경험이 전혀 없는 입문자를 위해 고안된 수업입니다. 필요한 것만 짚고 굵게 배우고, 내 것을 만들면서 기본기를 다집니다. 웹/앱 서비스의 작동 방식을 이해하고, 코딩하는 사람으로서의 삶을 누리세요.

코딩 공부 어디서 뭐부터 시작해야하지?  
더 이상 고민하지 마세요!

문의/상담은 여기로

이 웹사이트를 HTML로 간단히 표현해보면 아래와 같을 겁니다.

```

<!DOCTYPE html>
<html lang="en">
  <head>
  </head>
  <body>
    <header>
      ...
    </header>
    <div class="container">
      <div id="image-banner">
        ...
      </div>
      <div id="contents-1">
        ...
      </div>
    </div>
    <footer>
      ...
    </footer>
  </body>
</html>

```

이 코드를 조각조각 내보면 아래와 같이 나눌 수 있을 거예요.  
나눈 조각 하나하나를 컴포넌트라고 불러요!

1. <header/>
2. <container/>
  1. <imagebanner/>
  2. <contents1/>
3. <footer/>



즉, 이 웹 사이트는,  
크게 <header/>, <container/>, <footer/> 세 개의 컴포넌트가 있고,  
<container/> 컴포넌트는,  
<imagebanner/>, <contents1/> 컴포넌트로 이루어져 있는 거죠! (쉽죠!)

- Component는 웹 사이트의 조각이고, 우리는 이 조각을 모아서 웹사이트에 뿌려준다.



웹 사이트를 잘 조각낼 줄 아는 사람 === 리액트를 잘 쓰는 사람

#### ▼ 14) State와 Props



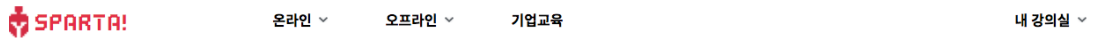
Component에서는 그럼 데이터 관리를 어떻게 할까요?  
데이터 관리를 아~주 살짝 맛만 볼게요. 2주차에 배울 **라이프 사이클**을 위한 선행이라고 생각하시고 가벼운 마음으로 들어보세요!

- State



state는 Component가 가지고 있는 데이터입니다.

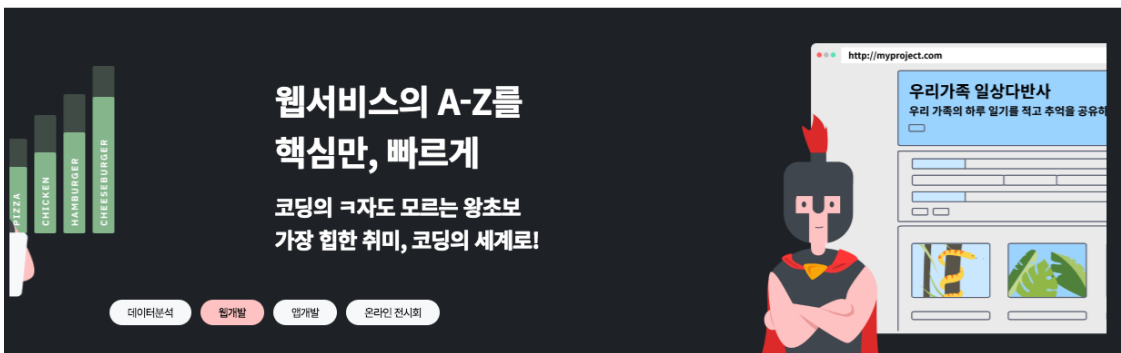
아까 스파르타코딩클럽 사이트 조각 중 <header/> 컴포넌트를 예로 들어볼게요!



- 이 헤더에 들어갈 데이터는 뭐가 있을까요?  
→ 로고 이미지 경로, 메뉴 이름(온라인, 오프라인, 기업교육, 내 강의실)이 있을 겁니다!
  - 이 데이터는 <container/>나 <footer/> 컴포넌트에서는 쓰지 않겠죠!
  - 즉, <header/> 컴포넌트에서만 쓰는 정보인 셈입니다.
  - state는 한 컴포넌트에서만 사용하는 정보를 주로 넣어놓고 생성, 수정하는 데이터입니다.
  - 생성도 수정도 오직 해당 컴포넌트 내에서만 이뤄집니다. 내꺼니까 생성도 수정도 자유롭죠!
- Props

👉 props는 Component가 부모 Component로부터 받아온 데이터입니다.

이번엔 <container/> 컴포넌트를 예로 들어볼게요!



스파르타 웹/앱개발

### 왕초보 코딩교육의 명가

스파르타코딩클럽은 사전 경험이 전혀 없는 입문자를 위해 고안된 수업입니다. 필요한 것만 짧고 굵게 배우고, 내 것을 만들면서 기본기를 다집니다. 웹/앱 서비스의 작동 방식을 이해하고, 코딩 하는 사람으로서의 삶을 누리세요.

코딩 공부 어디서 뭐부터 시작해야하지?  
더 이상 고민하지 마세요!

문의/상담은 여기로

```
<container>
  <imagebanner/>
  <contents1/>
</container>
```

<container/> 컴포넌트는 두 개의 자식 컴포넌트를 가지고 있죠?

- <container/> 컴포넌트만 <imagebanner/> 컴포넌트한테 필요한 이미지 경로를 가지고 있다고 가정합니다. (state로 가지고 있다고 가정합니다!)

- 이 때 <imagebanner/> 컴포넌트는 자신의 부모인 <container/>컴포넌트로부터 **이미지 경로**를 전달받아서 사용해야겠죠?
- <container/>가 가지고 있는 이미지 경로를 <imagebanner/>에게 전달해주면, 이 **이미지 경로**가 <imagebanner/> 컴포넌트의 props가 됩니다.
- 다시 말해, 부모 컴포넌트로부터 전달 받은 데이터를 props라고 합니다.
- 그럼 부모 컴포넌트가 가지고 있는 데이터를 <imagebanner/> 컴포넌트가 추가 하거나 수정할 수 있을까요?

**Props로 받은 데이터는 수정할 수 없습니다! 남의 것이니까요!**

## 10. Component (2)

### ▼ 15) 함수형 Component

👉 컴포넌트가 뭔지 이제 직접 만들어 봅시다!

- VSCode를 열고, src 폴더 안에 BucketList.js 파일을 하나 만듭니다.

⚠ 리액트 코딩 룰 1:  
폴더는 소문자로 시작하는 카멜케이스를 사용  
JS파일, 컴포넌트 이름은 대문자로 시작하는 카멜케이스를 사용

아래 코드를 BucketList.js에 붙여 넣어보세요.

#### ▼ [코드스니펫] - 함수형 컴포넌트

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

// 함수형 컴포넌트는 이렇게 쓸 수도 있고
// function Bucketlist(props){
//   return (
//     <div>버킷 리스트</div>
//   );
// }

// 이렇게 쓸 수도 있어요. =>가 들어간 함수를 화살표 함수라고 불러요.
// 저희는 앞으로 화살표 함수를 사용할거예요.
// 잇 () 안에 props! 부모 컴포넌트에게 받아온 데이터입니다.
// js 함수가 값을 받아오는 것과 똑같이 받아오네요.
const BucketList = (props) => {

  // 컴포넌트가 뿌려줄 ui 요소(리액트 엘리먼트라고 불러요.)를 반환해줍니다.
  return (
    <div>
      버킷 리스트
    </div>
  );
}

// 우리가 만든 함수형 컴포넌트를 export 해줍니다.
// export 해주면 다른 컴포넌트에서 BucketList 컴포넌트를 불러다 쓸 수 있어요.
export default BucketList;
```

- 그리고 App.js로 돌아가서 BucketList 컴포넌트를 불러와 봅시다.

#### ▼ [코드스니펫] - 컴포넌트 불러오기

```
import React from 'react';
import logo from './logo.svg';
import './App.css';
// BucketList 컴포넌트를 import 해옵니다.
// import [컴포넌트명] from [컴포넌트가 있는 파일경로];
import BucketList from './BucketList';

function App() {

  return (
    <div className="App">
      <h1>내 버킷리스트</h1>
      {/* 컴포넌트를 넣어줍니다. */}
      <BucketList/>
    </div>
  );
}

export default App;
```

- 이제 크롬으로 돌아가 컴포넌트가 나오는 지 확인해봅시다.



## 11. Component (3)

### ▼ 16) 클래스형 Component



이번엔 함수형 컴포넌트를 클래스형 컴포넌트로 바꿔볼게요!

App.js를 클래스형으로 바꾸고, BucketList 컴포넌트에 데이터를 넘겨줘 봅시다!

- App.js를 클래스형으로 바꿔봅시다.  
아래 코드를 App.js에 붙여 넣고 브라우저에서 함수형일 때와 동일하게 동작하는 걸 확인해봅시다.

#### ▼ [코드스니펫] - 클래스형 컴포넌트

```
import React from 'react';
import logo from './logo.svg';
import './App.css';
// BucketList 컴포넌트를 import 해옵니다.
// import [컴포넌트명] from [컴포넌트가 있는 파일경로];
import BucketList from './BucketList';

// 클래스형 컴포넌트는 이렇게 생성합니다!
class App extends React.Component {

  constructor(props){
    super(props);
    // App 컴포넌트의 state를 정의해줍니다.
    this.state = {
      list: ['영화관 가기', '매일 책읽기', '수영 배우기'],
    };
  }

  // 렌더 함수 안에 리액트 엘리먼트를 넣어줍니다!
  render() {
    return (
      <div className="App">
        <h1>내 버킷리스트</h1>
        { /* 컴포넌트를 넣어줍니다. */ }
        <BucketList/>
      </div>
    );
  }
}

export default App;
```

#### ▼ Component에서 Component로 데이터를 넘겨주자!

👉 App 컴포넌트가 가지고 있는 state를 BucketList에 넘겨줍니다.

- 클래스형과 함수형 컴포넌트의 모양을 보고 눈치채셨을 수도 있겠지만, **함수형 컴포넌트에는 state가 없습니다.**  
(나중에 배울 React hook을 사용하면 함수형 컴포넌트에서도 state를 사용할 수 있지만, hook 없이 state를 사용할 수 없어요!)



왜냐구요? **가상돔**이라는 리액트가 DOM을 다루는 방식 때문입니다.  
가상돔은 다음 주에 라이프 사이클과 함께 알아보려요!

- state 값 사용하기

아래와 같이 입력해보고 콘솔을 확인해봅시다.

```
render() {
  console.log(this.state);
  ...
}
```

```
▼ {list: Array(3)} ⓘ
  ► list: (3) ["영화관 가기", "매일 책읽기", "수영 배우기"]
  ► __proto__: Object
```



### 알면 덜 찜찜해지는, 몰라도 되는 이야기 1:

this 키워드는 깊이 들어가면 **context 객체**라고 부르는 친구와 연관이 있어요.

우리는 함수나 클래스 안에서 사용하면 this를 쓴 위치(위의 경우에는 App 클래스)에 있는 값을 가지고 온다고 생각합니다.

ex) App 클래스 안에서 쓰면, this.[변수명]은 App 클래스 안에 있는 값을 가지고 옵니다.

- App 컴포넌트에서 BucketList 컴포넌트로 state를 넘겨주자

컴포넌트에 props를 넘겨줄 때는 아래와 같이 넘겨줍니다.

```
render() {
  // this 키워드를 통해 state에 접근할 수 있어요.
  console.log(this.state);

  return (
    <div className="App">
      <h1>내 버킷리스트</h1>
      {/* 컴포넌트를 넣어줍니다. */}
      {/* <컴포넌트 명 [props 명]={넘겨줄 것(리스트, 문자열, 숫자, ...)}> */}
      <BucketList list={this.state.list}/>
    </div>
  );
}
```

잘 넘어갔는 지 확인해볼까요?

BucketList.js 파일을 열고, 아래와 같이 입력합니다.

```
const BucketList = (props) => {

  console.log(props);

  // 컴포넌트가 뿌려줄 ui 요소(리액트 엘리먼트라고 불러요.)를 반환해줍니다.
  return (
    <div>
      버킷 리스트
    </div>
  );
}
```

이제 브라우저를 열어 콘솔에 찍히는 지 확인해봅시다.

좋아요! 이제 App에서 BucketList로 데이터를 전달해줬네요! 😊

```
▼ {list: Array(3)} App.js:22
  ▶ list: (3) ["영화관 가기", "매일 책읽기", "수영 배우기"]
  ▶ __proto__: Object

▼ {list: {}} BucketList.js:17
  ▶ list: {list: Array(3)}
  ▶ __proto__: Object
```

## 12. Quiz\_버킷리스트 페이지를 만들어봅시다!

- ▼ 17) 🎨 버킷리스트 뼈대 잡기(1)

▼ Q. 퀴즈설명: 코드를 수정해서 아래 모양을 만들어보세요!



# 내 버킷리스트

영화관 가기  
매일 책임기  
수영 배우기

## ▼ [코드스니펫] - 버킷리스트 컴포넌트 만들기

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

// 함수형 컴포넌트는 이렇게 쓸 수도 있고
// function Bucketlist(props){
//   return (
//     <div>버킷 리스트</div>
//   );
// }

// 이렇게 쓸 수도 있어요. =>가 들어간 함수를 화살표 함수라고 불러요.
// 저희는 앞으로 화살표 함수를 사용할거예요.
// 약 () 안에 props! 부모 컴포넌트에게 받아온 데이터입니다.
// js 함수가 값을 받아오는 것과 똑같이 받아오네요.
const BucketList = ({list}) => {

  // Quiz 1: my_list에 ['a', 'b', 'c'] 대신 부모 컴포넌트가 넘겨준 값을 넣으려면 어떻게 해야할까요?
  const my_lists = ['a', 'b', 'c'];

  // 컴포넌트가 뿌려줄 ui 요소(리액트 엘리먼트라고 불러요.)를 반환해줍니다.
  return (
    <div>
      {
        // js의 내장 함수 중 하나인 map입니다. 리스트의 갯수만큼 => 오른쪽 구문을 반복해요.
        // 자세한 사용법은 아래 링크를 확인해주세요.
        // https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array/map
        my_lists.map((list, index) => {
          // 콘솔을 확인해봅시다 :)
          console.log(list);
          return (<div key={index}>{list}</div>);
        })
      }
    </div>
  );
}

// 우리가 만든 함수형 컴포넌트를 export 해줍니다.
// export 해주면 다른 컴포넌트에서 BucketList 컴포넌트를 불러다 쓸 수 있어요.
export default BucketList;
```



힌트:

1. 부모 컴포넌트가 넘겨준 값은 props로 받아옵니다.
2. 자바스크립트의 내장함수 중 하나인 Map은 for문하고 비슷한 친구죠!

## ▼ A. 함께하기(완성본)

### ▼ [코드스니펫] - 버킷리스트(완성)

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

// 함수형 컴포넌트는 이렇게 쓸 수도 있고
// function Bucketlist(props){
//   return (
//     <div>버킷 리스트</div>
```

```
//     );
// }

// 이렇게 쓸 수도 있어요. =>가 들어간 함수를 화살표 함수라고 불러요.
// 저희는 앞으로 화살표 함수를 사용할거예요.
// 앳 () 안에 props! 부모 컴포넌트에게 받아온 데이터입니다.
// js 함수가 값을 받아오는 것과 똑같이 받아오네요.
const BucketList = (props) => {

  // Quiz 1: my_list에 ['a', 'b', 'c'] 대신 부모 컴포넌트가 넘겨준 값을 넣으려면 어떻게 해야할까요?
  const my_lists = props.list;

  console.log(props);
  // 컴포넌트가 뿌려줄 ui 요소(리엑트 엘리먼트라고 불러요.)를 반환해줍니다.
  return (
    <div>
      {
        // js의 내장 함수 중 하나인 map입니다. 리스트의 갯수만큼 => 오른쪽 구문을 반복해요.
        // 자세한 사용법은 아래 링크를 확인해주세요.
        // https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array/map
        my_lists.map((list, index) => {
          // 콘솔을 확인해봅시다 :)
          console.log(list);
          return (<div key={index}>{list}</div>);
        })
      }
    </div>
  );
}

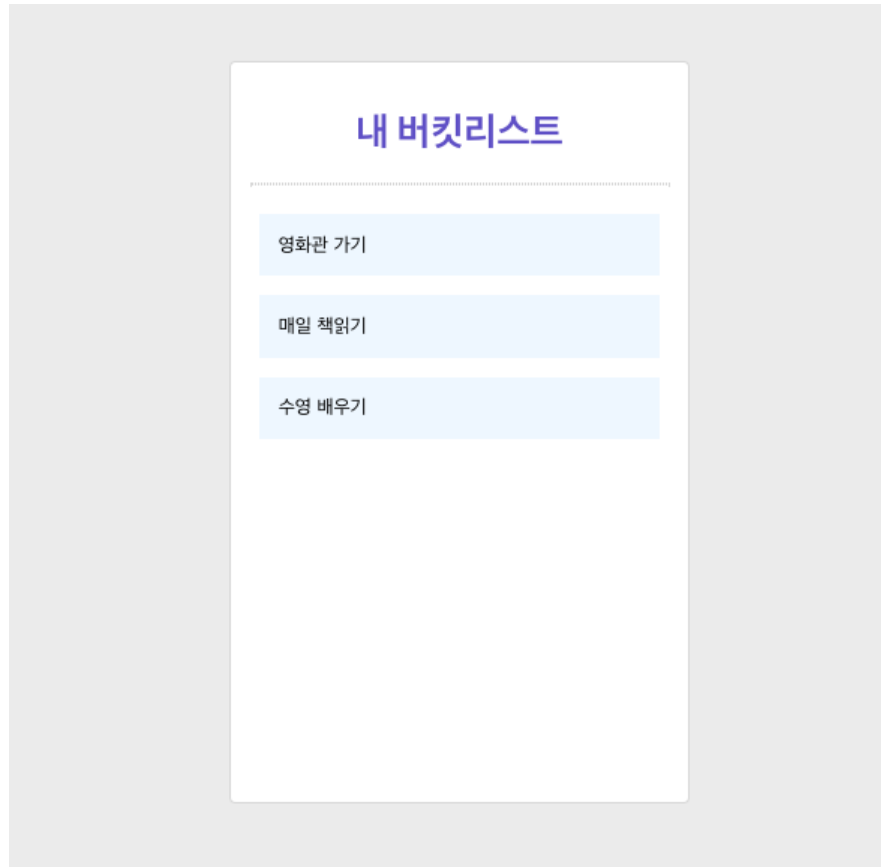
// 우리가 만든 함수형 컴포넌트를 export 해줍니다.
// export 해주면 다른 컴포넌트에서 BucketList 컴포넌트를 불러다 쓸 수 있어요.
export default BucketList;
```

## 14. 화면을 예쁘게! React에서 CSS 사용하기

### ▼ 18) CSS in React



이번에는 src폴더에 style.css 파일을 만들고 그 안에 스타일을 정의하고, 그 파일을 불러다 사용하는 방법을 배워볼게요!



#### ▼ (1) CSS 파일 만들기



class 대신 className을 주는 거 잊지 않으셨죠?  
아래에 만들어진 파일을 공유하지만, 꼭 혼자서도 만들어 보세요.

#### ▼ [코드스니펫] - App.js

```
import React from 'react';
import logo from './logo.svg';
// BucketList 컴포넌트를 import 해옵니다.
// import [컴포넌트명] from [컴포넌트가 있는 파일경로];
import BucketList from './BucketList';
import './style.css';

// 클래스형 컴포넌트는 이렇게 생겼습니다!
class App extends React.Component {

  constructor(props){
    super(props);
    // App 컴포넌트의 state를 정의해줍니다.
    this.state = {
      list: ['영화관 가기', '매일 책읽기', '수영 배우기'],
    };
  }

  // 랜더 함수 안에 리액트 엘리먼트를 넣어줍니다!
  render() {
    // this 키워드를 통해 state에 접근할 수 있어요.
    console.log(this.state);

    return (
      <div className="App">
        <div className="container">
          <h1 className="title">내 버킷리스트</h1>
```

```

        <hr className="line"/>
        { /* 컴포넌트를 넣어줍니다. */ }
        { /* <컴포넌트 명 [props 명]={넘겨줄 것(리스트, 문자열, 숫자, ...)}> */ }
        <BucketList list={this.state.list} />
      </div>
    </div>
  );
}
}

export default App;

```

## ▼ [코드스니펫] - BucketList.js

```

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

// 함수형 컴포넌트는 이렇게 쓸 수도 있고
// function BucketList(props){
//   return (
//     <div>버킷 리스트</div>
//   );
// }

// 이렇게 쓸 수도 있어요. =>가 들어간 함수를 화살표 함수라고 불러요.
// 저희는 앞으로 화살표 함수를 사용할거예요.
// 위 () 안에 props! 부모 컴포넌트에게 받아온 데이터입니다.
// js 함수가 값을 받아오는 것과 똑같이 받아오네요.
const BucketList = (props) => {

  // Quiz 1: my_list에 ['a', 'b', 'c'] 대신 부모 컴포넌트가 넘겨준 값을 넣으려면 어떻게 해야할까요?
  const my_lists = props.list;

  console.log(props);
  // 컴포넌트가 뿌려줄 ui 요소(리액트 엘리먼트라고 불러요.)를 반환해줍니다.
  return (
    <div className="lists">
      {
        // js의 내장 함수 중 하나인 map입니다. 리스트의 갯수만큼 => 오른쪽 구문을 반복해요.
        // 자세한 사용법은 아래 링크를 확인해주세요.
        // https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array/map
        my_lists.map((list, index) => {
          // 콘솔을 확인해봅시다 :)
          console.log(list);
          return (<div className="list-item" key={index}>{list}</div>);
        })
      }
    </div>
  );
}

// 우리가 만든 함수형 컴포넌트를 export 해줍니다.
// export 해주면 다른 컴포넌트에서 BucketList 컴포넌트를 불러다 쓸 수 있어요.
export default BucketList;

```

## ▼ [코드스니펫] - style.css

화면과 똑같지 않아도 괜찮아요! 원하는 모양으로 예쁘게 만들어주세요.

```

.App{
  width: 100vw;
  min-height: 100vh;
  background-color: #eee;
  padding: 32px;
  box-sizing: border-box;
}

.container{
  max-width: 350px;
  min-height: 80vh;
}

```

```

background-color: #fff;
padding: 16px;
margin: 20px auto;
border-radius: 5px;
border: 1px solid #ddd;
}

.title{
  color: slateblue;
  text-align: center;
}

.line{
  margin: 16px 0px;
  border: 1px dotted #ddd;
}

.lists{
  display: flex;
  flex-direction: column;
  height: 100%;
  overflow-x: hidden;
  overflow-y: auto;
}

.list-item{
  padding: 16px;
  margin: 8px;
  background-color: aliceblue;
}

```

## ▼ (2) CSS 파일을 가져다 쓰려면?

👉 import를 하자!

```

import React from 'react';
import logo from './logo.svg';
// BucketList 컴포넌트를 import 해옵니다.
// import [컴포넌트 명] from [컴포넌트가 있는 파일경로];
import BucketList from './BucketList';
// 이 import를 통해 css 파일을 불러다 씁니다!
import './style.css';

```



엇? App.js에서는 import를 했는데, 왜 BucketList.js에서는 style.css를 import하지 않나요?

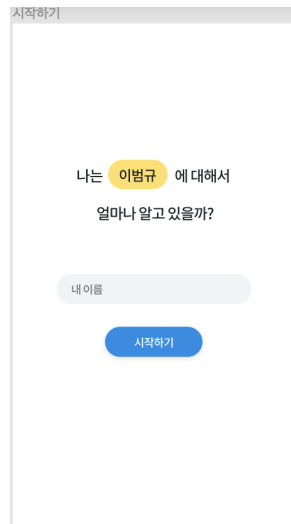
- 자식 컴포넌트는 부모 컴포넌트에 속해 있으니까요!
- 즉, 자식 컴포넌트는 return에 들어 있는 리액트 요소를 부모 컴포넌트에 가져다 줍니다.
- 부모와 같은 스타일 파일을 쓸 때는 부모 컴포넌트에만 import하면 됩니다.

## 15. 끝 & 속제 설명

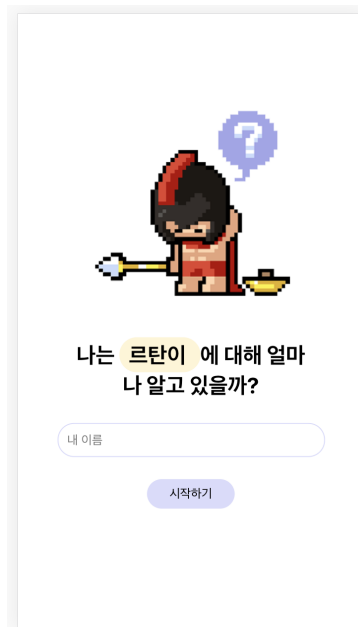


아래 기획서를 보고, 퀴즈 시작하기 화면을 만들어보세요!  
컴포넌트를 쪼개서 만들어봅시다!

### ▼ 기획서(레이아웃) 보기



#### ▼ 예시 화면



👉 우리가 배운 내용으로 위 페이지를 만들어봅시다.  
이미지도 넣어보고, 내 친구가 이름을 넣을 텍스트 입력 인풋과 시작하기 버튼을 만들어요.  
[나는 {} 에 대해서...]부분에 {}는 state에 넣고 prop로 넘겨서 해보세요!

## 16. 1주차 숙제 답안 코드

### ▼ [코드스니펫] - 1주차 숙제 답안 코드

#### 전체 코드

#### ▼ 이미지 파일



### ▼ App.js

```
import logo from './logo.svg';
import './App.css';
import React from "react";

import Start from "./Start";

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

    this.state = {
      name: "스파르타 코딩 클럽"
    };
  }

  render () {
    return (
      <div className="App">
        <Start name={this.state.name}/>
      </div>
    )
  }
}

export default App;
```

### ▼ Start.js

```
import React from "react";
import img from "./scc_img01.png";

const Start = (props) => {
  // 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",
        }}
      >
        <img alt="A pixel art character with a red and black headband, a red tunic, and a black skirt. The character is holding a yellow and black staff with a blue and white orb at the end. A purple speech bubble with a white question mark is above the character's head." data-bbox="396 96 656 273"/>
      </div>
    </div>
  )
}
```

```

        width: "100vw",
        overflow: "hidden",
        padding: "0px 10vw",
        boxSizing: "border-box",
        maxWidth: "400px",
    }}
    >
    <img src={img} style={{ width: "80%", margin: "16px" }} />
    <h1 style={{ fontSize: "1.5em", margin: "0px", lineHeight: "1.4" }}>
        나는{" "}
        <span
            style={{
                backgroundColor: "#fef5d4",
                padding: "5px 10px",
                borderRadius: "30px",
            }}
        >
            {props.name}
        </span>
        에 대해 얼마나 알고 있을까?
    </h1>
    <input
        type="text"
        style={{
            padding: "10px",
            margin: "24px 0px",
            border: "1px solid #dadafc",
            borderRadius: "30px",
            width: "100%",
            // backgroundColor: "#dadafc55",
        }}
        placeholder="내 이름"
    />
    <button
        style={{
            padding: "8px 24px",
            backgroundColor: "#dadafc",
            borderRadius: "30px",
            border: "#dadafc",
        }}
    >
        시작하기
    </button>
</div>
</div>
);
};

export default Start;

```

- 이미지는 src 폴더에 넣고 불러와주세요!

Copyright © TeamSparta All rights reserved.