



[스파르타코딩클럽] 리액트 기초반 - 1주차 (1)



매 주차 강의자료 시작에 PDF파일을 올려두었어요!

[수업 목표]

1. 프론트엔드 전반에 필요한 기본기를 다진다. (HTML, CSS, Javascript)
2. React 프로젝트 환경설정을 배워본다.
3. React에서 쓰는 태그(JSX)를 사용해서 화면을 그릴 수 있다.

[목차]

- 01. 프론트엔드의 꽃, 리액트 강의는 앞으로,
- 02. 필수 프로그램 설치
- 03. 프론트엔드 기초 지식
- 04. 알면 쉬운데 모르면 괴로운 (1) - HTML
- 05. 알면 쉬운데 모르면 괴로운 (2) - CSS
- 06. 자바스크립트 기초(1)
- 07. 자바스크립트 기초(2)
- 08. 자바스크립트 기초(3)
- 09. 첫 React 프로젝트 만들기
- 10. JSX
- 11. JSX 사용법
- 12. Quiz 간단한 텍스트 입력 화면 만들어보기
- 13. 끝 & 숙제 설명
- HW. 1주차 숙제 해설



모든 토글을 열고 닫는 단축키

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

Mac : **⌘** + **⌥** + **t**

01. 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 설명드릴 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

▼ 1~5주차에 배울 순서

- 1주차: React 개발 환경을 구성하는 방법, JSX



우리가 만들 웹사이트가 어떻게 동작하는지 웹사이트의 원리와 서버리스 환경에 대해 이론을 조금 알아보고,
NVM을 사용하는 방법과 CRA로 React 프로젝트를 만드는 방법을 배울거예요.
+) JSX를 배우고 써볼거예요.

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



리엑트를 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 매력...함께 하시죠!)

+)

React hook을 다뤄볼 거예요!

- 3주차: Redux, React-router



리엑트가 데이터를 어떻게 관리하는지 알아볼 거예요.

리엑트에서의 페이지 전환도 알아볼 거예요. (화면이 깜빡거리지 않고 이동하는 스무스한 전환, 기대 되죠!)

- 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 공부, AWS S3로 배포하기



리액트에서 개발을 더 편하게 해줄 공부 몇 가지를 배울 거예요.

우리가 만든 퀴즈 페이지를 친구들에게 공유할 수 있도록 AWS S3에 배포도 해봅니다.

02. 필수 프로그램 설치

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

▼ NVM 설치

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



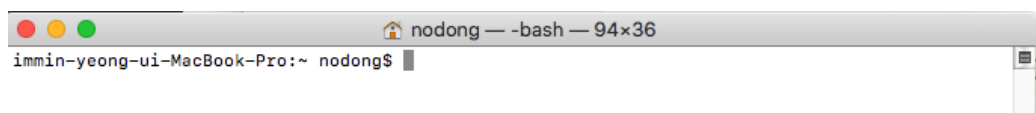
윈도우 사용자의 경우 꼭 NVM을 설치하지 않아도 괜찮습니다. node 공식 사이트에서 LTS 버전을 바로 다운로드해주세요. 😊

- 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!)

- a. vi 에디터로 파일을 열면 방향키만 움직일 수 있을거예요!

i 키를 눌러서 입력 모드로 전환합니다.

- b. 방향키를 아래로 쪽쪽 내리고 맨 밑에서 엔터 두번!

- c. 아래 내용을 복사하고 붙여넣어줍니다.

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

- d. 저장하고 나가기

- i. **esc** 키를 누르고,
- ii. **:wq** 를 입력합니다. (그리고 엔터!)

- e. 추가한 내용을 적용하자!

- i. 터미널이 bash일 경우,

```
source ~/.bash_profile
```

- ii. 터미널이 zsh일 경우,

```
source ~/.zshrc
```

f. 설치를 확인합니다.

```
nvm --version
```

03. 프론트엔드 기초 지식



프론트엔드의 꽃, 리액트!

리액트를 본격적으로 배우기 전에 꼭(X100) 알아야하는 기초 지식을 배워볼거예요.

빨리 리액트하러 가고 싶은 맘 저도 알고 있어요. 그래도 기초 지식은 소중한니까 꼭 듣고 가기! 😊

▼ 1) 서버와 클라이언트, 복습하자!

▼ 1. 서버와 클라이언트, 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)



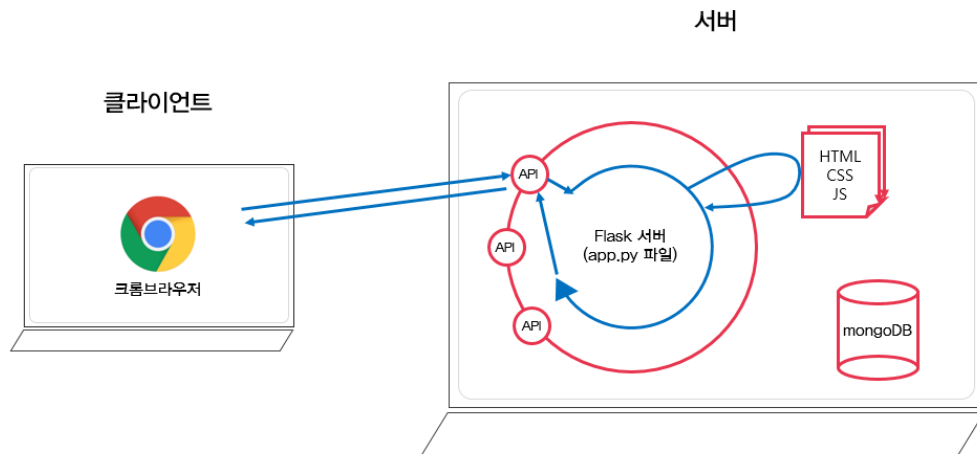
네! 우리가 보는 웹페이지는 모두 서버에서 미리 준비해두었던 것을 "받아서", "그려주는" 것입니다. 즉, 브라우저가 하는 일은 1) 요청을 보내고, 2) 받은 HTML 파일을 그려주는 일 뿐이죠.



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!



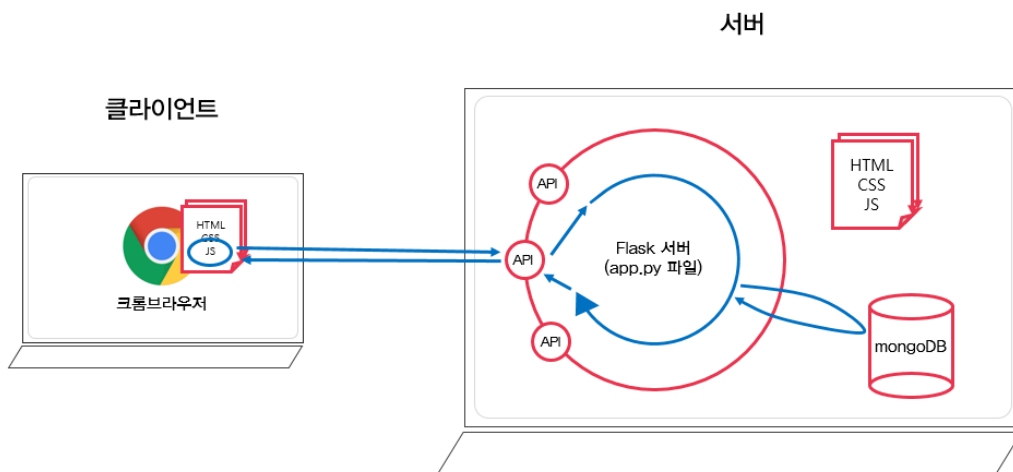
▼ 2. 웹의 동작 개념 (데이터만 받는 경우)

☞ 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 한답니다.

```

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",

```

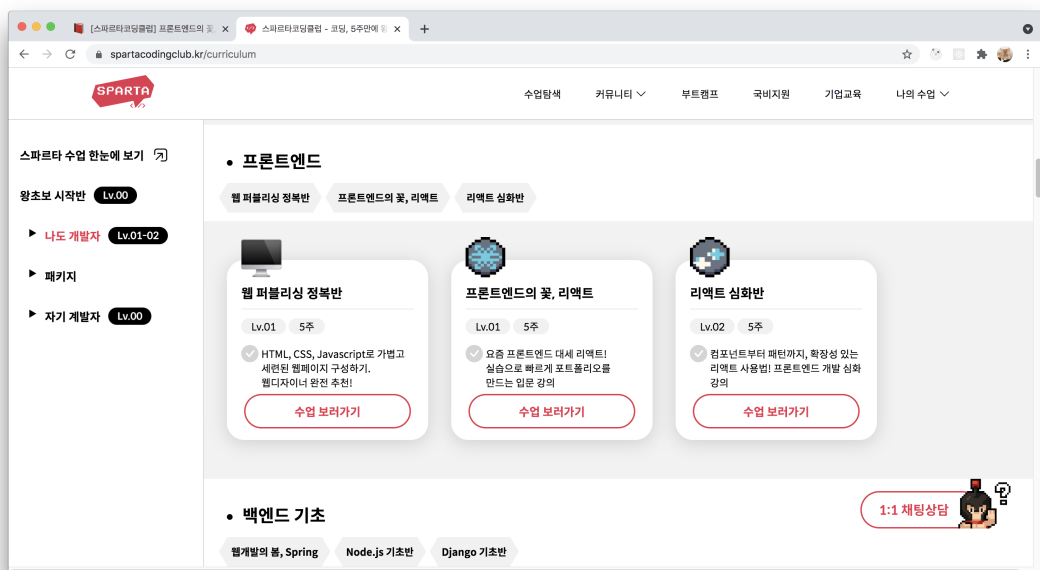
▼ 3. 그래서 서버랑 클라이언트가 뭐라고?



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

▼ 2) 우리는 무슨 일을 할까?

▼ 1. 화면을 그리고

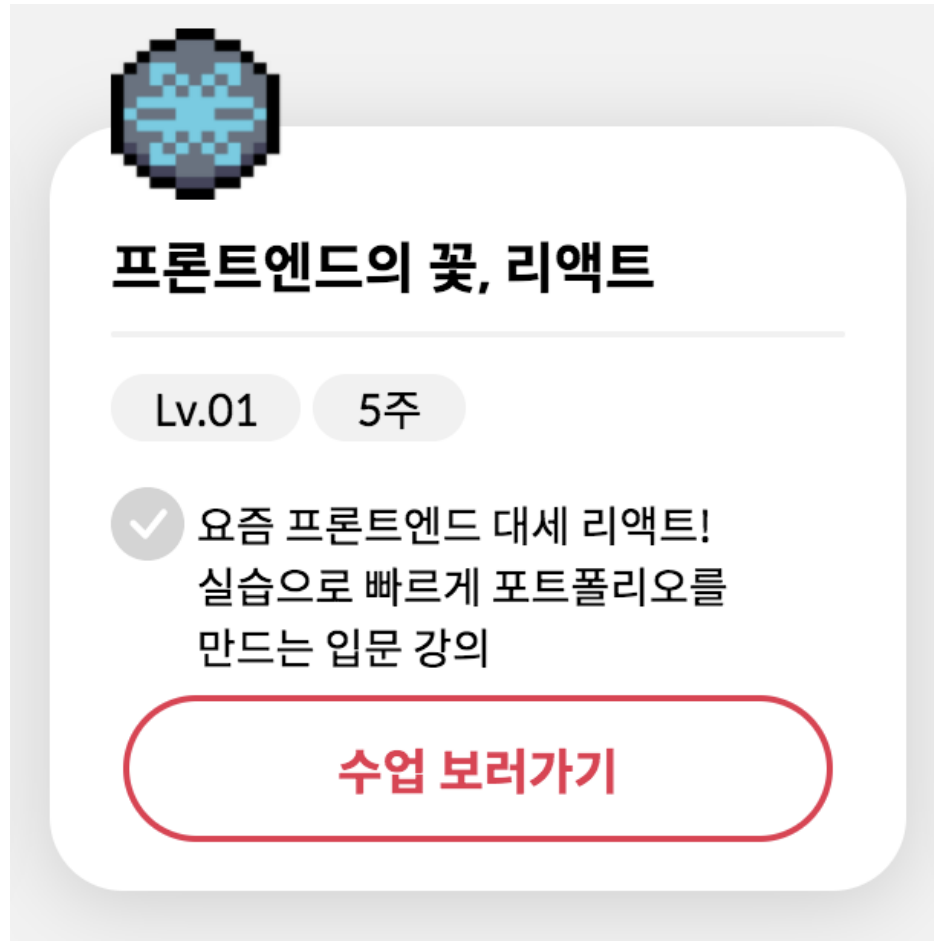




말그대로 화면을 그립니다. 눈에 보이는 부분을 만들어요.

어디에는 글이 들어가고, 어디에는 아이콘을 넣어주고 ..., 더 나아가서는 마우스를 올릴 때만 글자색을 바꿔주는 등의 작업(인터랙션을 준다고 해요!)도 해줍니다.

▼ 2. 데이터를 끼얹는다!



서버에서 데이터를 가지고 와서 만든 화면에 착착 넣어주는 거예요. 생각만해도 벌써 두근두근하죠!

▼ 3) 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 안해도 된다는 겁니다.

+)

자세한 개념은 4주차에 예제와 함께 배울 거예요.

오늘은 **서버리스는 백엔드리스가 아니라는 것만** 알고 있으면 굳!! 👍

04. 알면 쉬운데 모르면 괴로운 (1) - HTML



HTML(Hypertext Markup Language)

HTML(Hypertext Markup Language)은 마크업 언어예요.

마크업이 뭐냐구요? 마크업은 이름 그대로 표시하는 거예요.

우리가 보는 웹페이지는 그림도 있고, 글도 있고 표도 있고, 여러가지 요소가 잔뜩 들어있죠!

"여기는 글자 영역이고 여기는 이미지 영역이야!"라고 표시해서 브라우저가 웹페이지를 잘 그릴 수 있도록 하는 게 HTML입니다. 😊

▼ 4) 🛠️ HTML로 오늘의 TODO 리스트 만들기



이미 여러분에겐 쉬운 일이죠! 손가락도 풀어볼 겸 간단한 HTML 페이지를 만들어봅시다.

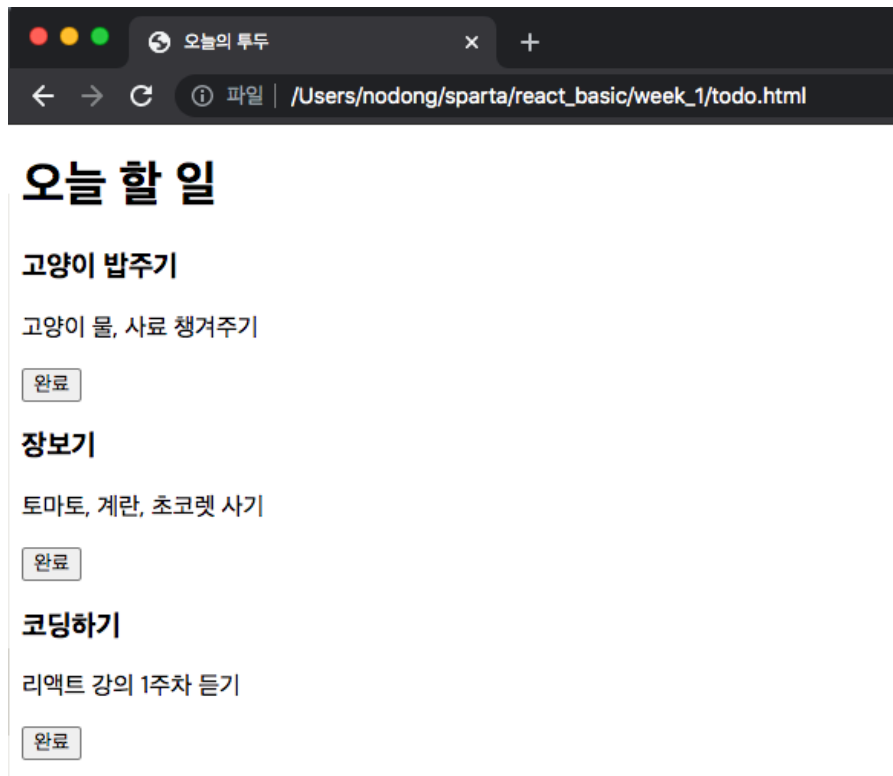


앞으로 실습이 필요한 부분이 나오면 "🛠️"모양으로 알려줄게요!

퀴즈설명 영상을 먼저 보고 → 정해진 시간동안 혼자 한다음 → 함께하기 영상을 보세요!

▼ Q. 퀴즈설명

▼ 모습 보기



힌트 : 사이트 제목을 바꿀 땐 <head>에서 뭘 바꿔주면 될까요?
기억이 잘 나지 않는다면 호다닥 구글에게 물어보기!

▼ A. 함께하기(완성본)



어때요, 할만했나요? 조금씩 다른 방법으로 해결하셨더라도 OK!
다만 아직 예쁘게 꾸미거나, 이런 저런 인터랙션을 넣진 마세요! 다음 강의에서 같이 할거니까요. (소근

▼ [코드스니펫] 오늘의 TODO HTML

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>오늘의 투두</title>
</head>
<body>
  <h1>오늘 할 일</h1>
  <div>
    <h3>고양이 밥주기</h3>
    <p>고양이 물, 사료 챙겨주기</p>
    <button>완료</button>
  </div>

  <div>
    <h3>장보기</h3>
    <p>토마토, 계란, 초코렛 사기</p>
    <button>완료</button>
  </div>

  <div>
    <h3>코딩하기</h3>
  </div>
</body>
</html>
```

```

    <p>리액트 강의 1주차 듣기</p>
    <button>완료</button>
  </div>
</body>
</html>

```

▼ 5) 안녕 DOM!



이번에는 프론트엔드 전반에 꼭 필요한 html 개념을 배울거예요.



<div>안녕</div>, <p>하세요</p>, <button>확인</button> ... 등등!

Q. 이 친구들, 뭐라고 부르는 지 알고 있나요?

→ A. Tag? 비슷해요! 하지만, 땡! 틀렸습니다. 이 친구들은 요소(elements)라고 불러요.

Q. 엇, 그럼 tag는 뭔가요?

→ A. tag는 요소를 만들 때 사용하는 <></> 이 친구를 말해요.

▼ (1) DOM



DOM(문서객체모델)을 아시나요?

DOM은 html 단위 하나하나를 객체로 생각하는 모델입니다. 예를 들면, 'div'라는 객체는 텍스트 노드, 자식 노드 등등, 하위의 어떤 값을 가지고 있겠조? 이런 구조를 트리 구조라고 합니다.

네, 맞습니다! DOM이 트리구조란 소리입니다.

▼ (2) dom 트리 확인하기



브라우저의 개발자 도구에서 하나씩 확인해봅시다.

([Cmd + Opt + i](맥), [Ctrl + Shift + i / F12](윈도우))를 눌러 개발자 도구를 열 수 있어요.)

```

// 현재 dom 트리를 볼 수 있어요.
document

// dom 트리 중, body의 내용을 확인합니다.
document.body

// dom 트리 중, head의 내용을 확인합니다.
document.head

```

- document와 body, head는 부모 - 자식 관계예요. body랑 head는 무슨 사이냐고요?
형제 관계입니다. (sibling이라고도 하죠!)

▼ (3) 자식 요소에 접근하기



body 안에 들어 있는 요소에는 어떻게 접근하는 지 배워봅시다!

- childNodes

```
document.body.childNodes
```

- children

```
document.body.children
```

- `getElementsByName("태그 이름")`

```
document.getElementsByTagName("div")
```

- 이 외에도 `firstChild`, `lastChild` 등등 자식 노드에 접근할 수 있는 방법은 많아요.
MDN 문서([링크](#))에서 이런저런 접근 방법을 찾아보세요!

▼ [코드스니펫] MDN 문서 링크

```
https://developer.mozilla.org/ko/docs/Web/API/Document
```



[개발 이야기] HTML은 프로그래밍 언어인가?

→ 유명한 논쟁거리예요. 우리는 이미 답을 알고 있죠!

프로그래밍 언어는 어떤 연산을 수행하거나, 소프트웨어, 시스템을 동작하게 하는 언어고요. 즉, HTML은 프로그래밍 언어가 아니라 마크업 언어입니다.

05. 알면 쉬운데 모르면 괴로운 (2) - CSS



← CSS가 싫은 여러분,,

CSS는 꾸미기만 하면 장땡아닌가요? 맞아요! 하지만 덜 힘들게 잘 꾸며보려면 많이 아는 게 좋아요.
알아두면 좋은 몇 가지 개념만 빠르게 확인해봐요!

▼ 6) Selector



selector는 꾸밀 요소를 선택하는 선택자예요. 뭘 꾸밀 지 선택하고 속성을 넣어 예쁘게 꾸며주는겁니다.
앗? 속성이 어떤 게 있는 지 잘 모르겠다고요? 걱정마세요! 우리에게엔 MDN 문서([링크](#))가 있으니까요!

▼ [코드스니펫] MDN 문서 링크

```
https://developer.mozilla.org/ko/docs/Web/API/Document
```

```
/* id selector */
#id { ... }

/* class selector */
.class { ... }
```

```

/* tag selector */
tagName { ... }

/* 여러 요소 선택하기 */
#id, .class { ... }

/* 수도 클래스 선택자 */
/* 어떤 요소가 특정 상태(마우스 올림, 포커스 됨 등등)일 때만 선택하게 해주는 선택자예요. */
button:hover { ... }

```

▼ (1) 🗨️ 오늘의 TODO 꾸며보기



선택자를 마구마구 사용해서 예쁘게 꾸며봐요!
튜터와 달라도 OK!

▼ Q. 퀴즈설명

▼ 모습 보기

오늘 할 일

고양이 밥주기

고양이 물, 사료 챙겨주기

완료

장보기

토마토, 계란, 초코렛 사기

완료

코딩하기

리액트 강의 1주차 듣기

완료



힌트: css 파일을 만들어도 좋고, todo.html에 넣어서 하셔도 좋아요. 저는 todo.html에 넣어서 해볼게요.

▼ A. 함께하기(완성본)



생각보다 재미있죠!

▼ [코드스니펫] 오늘의 TODO HTML

```

<!DOCTYPE html>
<html lang="ko">
  <head>

```

```

<meta charset="UTF-8" />
<title>오늘의 투두</title>
</head>
<body>
  <style>
    #title {
      color: blue;
      text-decoration: underline;
    }

    .todo-card {
      border: 1px solid gray;
      border-radius: 5px;
      padding: 2em;
      margin: 1em;
    }

    button:hover {
      background-color: orange;
    }
  </style>
  <h1 id="title">오늘 할 일</h1>
  <div class="todo-card">
    <h3>고양이 밥주기</h3>
    <p>고양이 물, 사료 챙겨주기</p>
    <button>완료</button>
  </div>

  <div class="todo-card">
    <h3>장보기</h3>
    <p>토마토, 계란, 초코렛 사기</p>
    <button>완료</button>
  </div>

  <div class="todo-card">
    <h3>코딩하기</h3>
    <p>리액트 강의 1주차 듣기</p>
    <button>완료</button>
  </div>
</body>
</html>

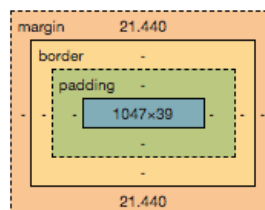
```

👉 #id, .class를 선택자라고 부르는데, { ... }은 뭐라고 부를까요?
선언부라고 부릅니다! 속성 : 값;으로 이루어져 있어요.

▼ 7) 그리드 시스템

👉 DOM 요소들은 기본적으로 박스형태로 표시됩니다.
p, div 할 것 없이 모두 네모난 영역을 가져요!

▼ (1) 박스모델



- margin box:
가장 바깥 영역입니다. margin 속성을 주면 이 영역이 바뀌어요.
주로 다른 요소들과 간격을 줄 때 사용해요!
- border box:
테두리 영역입니다. border 속성으로 테두리를 주면 이 영역이 바뀌어요.
- padding box:
테두리와 콘텐츠 사이 영역입니다. padding 속성을 주면 이 영역이 바뀌어요.
박스 내부의 간격을 줄 때 사용해요!
- contents box:
실제 콘텐츠 영역입니다. width, height 등으로 사이즈를 줄 수 있고, 따로 지정하지 않을 경우는 콘텐츠 내용(글이나 이미지 등)에 따라 임의로 사이즈가 잡혀요.

```
.todo-card {
  border: 1px solid gray;
  border-radius: 5px;
  padding: 2em;
  margin: 1em;
}
```

▼ (2) 그리드 시스템



그리드 시스템은 레이아웃을 잡는데 사용하는 거예요.

예를 들자면, todo-card를 두 줄로 보여주고 싶을 때처럼, 가로, 세로를 나누어 화면 배치를 만겨주는 거죠!

- flex로 그리드 잡기



todo-card를 두 줄로 만들어봅시다!

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8" />
    <title>오늘의 투두</title>
  </head>
  <body>
    <style>
      #title {
        color: blue;
        text-decoration: underline;
      }

      .wrap {
        display: flex;
      }

      .todo-card {
        border: 1px solid gray;
        border-radius: 5px;
        padding: 2em;
        margin: 1em;
        flex: 1 1 0;
      }

      button:hover {
        background-color: orange;
      }
    </style>
  </body>
</html>
```

```

</style>
<h1 id="title">오늘 할 일</h1>
<div class="wrap">
  <div class="todo-card">
    <h3>고양이 밥주기</h3>
    <p>고양이 물, 사료 챙겨주기</p>
    <button>완료</button>
  </div>

  <div class="todo-card">
    <h3>장보기</h3>
    <p>토마토, 계란, 초코렛 사기</p>
    <button>완료</button>
  </div>
</div>
<div class="wrap">
  <div class="todo-card">
    <h3>코딩하기</h3>
    <p>리액트 강의 1주차 듣기</p>
    <button>완료</button>
  </div>
  <div class="todo-card">
    <h3>코딩하기</h3>
    <p>리액트 강의 1주차 듣기</p>
    <button>완료</button>
  </div>
</div>
</body>
</html>

```



flex 하나만 알아보았지만, 레이아웃을 잡을 때는 사실 table, block, grid 등 아주 다양한 방법을 사용해요. 다른 방법도 꼭 검색해보기!

▼ 8) css 사칙연산



css도 연산이 된다는 거! 알고 계셨나요?
이번에는 button 크기를 연산해볼게요!

▼ (1) calc()



버튼 크기를 화면의 20%보다 20px 작게 바꿔봅시다!

```

<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8" />
    <title>오늘의 투두</title>
  </head>
  <body>
    <style>
      #title {
        color: blue;
        text-decoration: underline;
      }

      .wrap {
        display: flex;
      }
      .todo-card {
        border: 1px solid gray;

```



```

border-radius: 5px;
padding: 2em;
margin: 1em;
flex: 1 1 0;
}
button {
width: calc(20% - 20px);
}

button:hover {
background-color: orange;
}
</style>
<h1 id="title">오늘 할 일</h1>
<div class="wrap">
  <div class="todo-card">
    <h3>고양이 밥주기</h3>
    <p>고양이 물, 사료 챙겨주기</p>
    <button>완료</button>
  </div>

  <div class="todo-card">
    <h3>장보기</h3>
    <p>토마토, 계란, 초코렛 사기</p>
    <button>완료</button>
  </div>
</div>
<div class="wrap">
  <div class="todo-card">
    <h3>코딩하기</h3>
    <p>리액트 강의 1주차 듣기</p>
    <button>완료</button>
  </div>
  <div class="todo-card">
    <h3>코딩하기</h3>
    <p>리액트 강의 1주차 듣기</p>
    <button>완료</button>
  </div>
</div>
</body>
</html>

```



사칙 연산하면 제일 중요한 게 있죠! 네, 바로 연산 순서입니다.

calc()는 아래 순서로 연산을 해요.

- 왼쪽에서 오른쪽으로,
- 곱셈, 나눗셈을 먼저, 덧셈, 뺄셈은 그 다음에!
- 괄호가 있으면 괄호 안쪽을 먼저 계산



CSS는 이 밖에도 아주 많은 내용이 있어요.

오늘은 정말 꼭 알아야 하는 내용만 함께 알아보았지만, 틈이 나는대로 MDN에서 CSS 문서([링크](#))를 살펴보세요. 많은 속성을 미리 알아두수록 다양한 활용이 가능해요! 😊

06. 자바스크립트 기초(1)



이제 드디어 자바스크립트 시간입니다. 🤖

자바스크립트로 DOM을 조작하는 방법, 궁금하시죠!

바닐라 자바스크립트로 해봅시다!



저는 오늘의 TODO에서 <script></script> 태그 안에 코드를 작성할거예요.
여러분은 별도의 .js 파일을 만들어서 하셔도 괜찮아요. 😊

▼ 14) document로 DOM에 접근하기

▼ (1) document.getElementsByClassName



클래스 명으로 DOM 요소에 접근해봅시다!

```
const wraps = document.getElementsByClassName("wrap");
console.log(wraps);
```

- 콘솔에 찍힌 리스트! 확인하셨나요?

HTML Collection이라는 배열같은 게 하나 나오죠.

이게 바로 **자바스크립트 이론에서 배웠던 유사배열**입니다. **Array의 내장함수를 쓸 수 없다**고 말씀드린 그 친구예요. (소근)

▼ (2) document.getElementById



이번에는 id로 DOM 요소에 접근해볼거예요.

접근만 하려니 재미가 영 없죠? 🤔

그러니까 이번엔 자바스크립트를 써서 배경색도 한 번 넣어줘봅시다.

```
const title = document.getElementById("title");
// 자바스크립트에서는 어떤 객체의 속성에 접근할 때 .을 이용해 접근할 수 있어요.
// title이라는 요소의 style 속성에 접근하려면 아래와 같이 title.style로 접근할 수 있습니다!
// style 안에 있는 속성에 접근할 때도 마찬가지예요. style.backgroundColor로 배경색 속성에 접근할 수 있어요.
title.style.backgroundColor = "yellow";
```

▼ (3) document.getElementsByTagName



id와 class 명을 써보았으니 이번엔 태그 명도 써볼까요!

이번에도 그냥 접근만 하긴 재미가 없으니까, 버튼을 눌렀을 때 onClick 이벤트를 발생시켜 봅시다.

```
function sayHello (event) {
  console.log("hello");
}

const buttons = document.getElementsByTagName("button");

// 이 구문은 X! html collection은 유사 배열이기 때문에 array의 내장함수를 쓸 수 없어요!
//buttons.map(b => {
  //console.log(b);
//});

for (let i=0; i< buttons.length; i++){
  // .addEventListener()로 클릭 이벤트를 연결해줍니다.
  buttons[i].addEventListener("click", sayHello);
}
```

👉 어때요? 버튼을 누르면 콘솔 창에 hello라고 잘 나오고 있나요? 😊

▼ (4) 🛠️ 오늘의 TODO 완료하기

🔥 배운 내용을 바탕으로 오늘의 TODO에서 완료하기 버튼을 누르면 카드 색이 바뀌도록 해봅시다!

▼ Q. 퀴즈설명

▼ 모습 보기

오늘 할 일

고양이 밥주기

고양이 물, 사료 챙겨주기

완료

장보기

토마토, 계란, 초코렛 사기

완료

코딩하기

리액트 강의 1주차 듣기

완료

코딩하기

리액트 강의 1주차 듣기

완료



힌트 : 클릭 이벤트는 앞서 배운 것처럼 addEventListener를 사용해도 되지만, 아래와 같이 작성하셔도 됩니다!

`<button onClick="[실행할 함수 이름()]">...</button>`

▼ A. 함께하기(완성본)



어때요, 할만했나요? 저는 getElementByClassName를 사용했지만, 앞서 배운 다른 걸 사용해서 해결해도 좋아요. 😊

▼ [코드스니펫] 오늘의 TODO HTML

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8" />
    <title>오늘의 투두</title>
  </head>
  <body>
    <style>
      #title {
        color: blue;
        text-decoration: underline;
      }

      .wrap {
        display: flex;
      }
    </style>
  </body>
</html>
```

```

}
.todo-card {
  border: 1px solid gray;
  border-radius: 5px;
  padding: 2em;
  margin: 1em;
  flex: 1 1 0;
}
button {
  width: calc(20% - 20px);
}

button:hover {
  background-color: orange;
}
</style>

<h1 id="title">오늘 할 일</h1>
<div class="wrap">
  <div class="todo-card">
    <h3>고양이 밥주기</h3>
    <p>고양이 물, 사료 챙겨주기</p>
    <button onClick="changeBackgroundColor(0)">완료</button>
  </div>

  <div class="todo-card">
    <h3>장보기</h3>
    <p>토마토, 계란, 초코렛 사기</p>
    <button>완료</button>
  </div>
</div>
<div class="wrap">
  <div class="todo-card">
    <h3>코딩하기</h3>
    <p>리액트 강의 1주차 듣기</p>
    <button>완료</button>
  </div>
  <div class="todo-card">
    <h3>코딩하기</h3>
    <p>리액트 강의 1주차 듣기</p>
    <button>완료</button>
  </div>
</div>

<script>
const title = document.getElementById("title");
// 자바스크립트에서는 어떤 객체의 속성에 접근할 때 .을 이용해 접근할 수 있어요.
// title이라는 요소의 style 속성에 접근하려면 아래와 같이 title.style로 접근할 수 있습니다!
// style 안에 있는 속성에 접근할 때도 마찬가지로요. style.backgroundColor로 배경색 속성에 접근할 수 있어요.
title.style.backgroundColor = "yellow";

const wraps = document.getElementsByClassName("wrap");
console.log(wraps);

function sayHello (event) {
  console.log("hello");
}

const buttons = document.getElementsByTagName("button");

// 이 구문은 X! html collection은 유사 배열이기 때문에 array의 내장함수를 쓸 수 없어요!
//buttons.map(b => {
//  console.log(b);
//});


for (let i=0; i< buttons.length; i++){
  // .addEventListener()로 클릭 이벤트를 연결해줍니다.
  buttons[i].addEventListener("click", sayHello);
}

function changeBackgroundColor (index) {
  // 가장 먼저 클래스 명으로 card들을 가지고 와봅시다! 아래 콘솔로 확인해보세요!
  console.log(document.getElementsByClassName("todo-card"));
}


```

```
// 몇 번째 카드의 배경색을 바꿀 지 정해주었어요.
let card = document.getElementsByClassName("todo-card")[index];
// 그럼 바꿔봅시다! 저장 -> 브라우저로 돌아가 새로고침하면 확인할 수 있어요. :)
card.style.backgroundColor = "gray";
}
</script>
</body>
</html>
```

▼ 15) document에 DOM 요소 추가하기

 이미 만들어진 것 말고도 새로운 요소를 추가하고 싶을 때 어떻게 하면 좋을 지 알아봅시다.

▼ (1) document.createElement()


 요소는 만들지만 하면 알아서 DOM에 뽕 생기지 않아요!
만들고 → DOM에 넣어주어야 합니다.

```
// 새 요소를 만들어요.
const new_div = document.createElement("div");

// 한 눈에 확인해볼 수 있도록 스타일을 조금 추가해줄게요. 그냥 div는 눈에 안보일테니까요!
new_div.style.backgroundColor = "green";
new_div.style.width = "100px";
new_div.style.height = "100px";


// 요소를 body에 추가해줍니다.
document.body.appendChild(new_div);
```

07. 자바스크립트 기초(2)

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

ES6가 뭐냐구요?
자바스크립트 표준 문법 중 하나로 가장 보편화된 친구라고 생각해주세요!

▼ 10) 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) 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);
  }
  // 앓! c는 값이 변했죠?
  // 그렇습니다. var는 함수 단위라서 if문 밖에서 선언한 값이 변했어요.
  // let과 const로 선언한 값은 어떤가요? if문 안쪽 내용이 바깥 내용에 영향을 끼치지 않죠?
  console.log(a, b, c);
}
```

▼ (2) 함수



함수는 `function do_something() { ... }` 처럼 생긴 친구들을 말합니다.

어떤 코드의 묶음이고, `do_something()` 처럼 `()` 를 붙여주면 미리 만들어둔 코드 묶음이 실행되는 거예요.

내장 함수는 우리가 만들지 않아도 자바스크립트가 쓰기 편하라고 **미리 만들어둔 코드 묶음들**이에요!

- 함수 선언식 :

```
function do_something() { ... }
```

- 함수 표현식 :

```
// 함수 이름은 생략해도 괜찮아요!
let do_something = function [함수 이름]() { ... }
```

- 화살표 함수

```
// 화살표 함수는 함수 표현식의 단축형입니다.
// function까지 생략이 되었죠!
let do_something = () => { ... }
```

▼ (3) 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);
  }
}

// 여기서 new는 키워드예요. 새로운 무언가를 만들기 위해서 생성자 함수와 함께 씁니다.
// 네, new와 생성자 함수는 세트예요 :) (소근
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(){
    console.log(this.name);
    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(){
  console.log(this.name);
  // 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.

▼ (4) =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.
→ 0은 숫자(number)고, "0"은 문자(string)이니까요!

▼ (5) 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 꺾쇠가 새로운 배열을 뜻하죠!)에 넣어주겠다 말입니다!
let new_array = [...array];

console.log(new_array);
```

▼ (6) 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);
```

08. 자바스크립트 기초(3)



우리가 앞으로 정말정말(x100) 자주 쓸 Array의 내장 함수 4개를 함께 써봅시다! 😎



[그 전에 잠깐! 함수가 뭐더라?]

함수는 `function do_something() { ... }` 처럼 생긴 친구들을 말합니다.

어떤 코드의 묶음이고, `do_something()` 처럼 `()` 를 붙여주면 미리 만들어둔 코드 묶음이 실행되는 거예요.

내장 함수는 우리가 만들지 않아도 자바스크립트가 쓰기 편하라고 **미리 만들어둔 코드 묶음들이에요!**

▼ 11) 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 만들어볼거예요.
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 만들어볼거예요.
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!

▼ 12) 📌 Quiz_Array 내장 함수



크롬 브라우저를 열고 콘솔 창에서 해봅시다! 😊

▼ (1) 배열에서 고양이 몇 마리인지 세기 - map으로 해보자!



아래 for문을 map으로 바꿔봅시다.

▼ [코드스니펫] animals 배열 1

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

```
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!

▼ [코드스니펫] animals 배열 2

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

```

const animals = ["복슬 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 있냐? 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!

▼ [코드스니펫] concat 퀴즈

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

```
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? 😎

09. 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!

▼ 13) 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 일이지요. 😞

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!

```
nvm install 16.15.1
```

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

```
nvm ls # nvm으로 설치한 노드 버전 리스트 확인 명령어  
nvm use 16.15.1 # 16.15.1으로 노드 버전 적용 명령어  
node -v # 노드 버전 확인 명령어
```

```
~ nvm ls  
v16.13.0  
v16.14.0  
-> v16.15.1  
system  
default -> 16 (-> v16.15.1)  
iojs -> N/A (default)  
unstable -> N/A (default)  
node -> stable (-> v16.15.1) (default)  
stable -> 16.15 (-> v16.15.1) (default)  
lts/* -> lts/gallium (-> v16.15.1)  
lts/argon -> v4.9.1 (-> N/A)  
lts/boron -> v6.17.1 (-> N/A)  
lts/carbon -> v8.17.0 (-> N/A)  
lts/dubnium -> v10.24.1 (-> N/A)  
lts/erbium -> v12.22.12 (-> N/A)  
lts/fermium -> v14.19.3 (-> N/A)  
lts/gallium -> v16.15.1
```



갓 설치한 16.15.1 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 16.15.1인데, 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]

▼ 14) npm으로 yarn을 설치해보자



설치,, 그만! yarn을 꼭 써야만 하나?

꼭 사용하실 필요 없어요. npm을 그대로 사용하셔도 좋고, npx를 사용하셔도 좋습니다.
강의에서는 yarn을 사용하지만, 원하는 패키지 매니저를 사용하셔도 좋아요.

- 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 설치되어 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]
```

▼ 15) CRA(create-react-app)으로 시작하는 리액트

- yarn으로 CRA를 설치하자!



yarn을 설치하지 않은 분들은 아래 명령어 대신, npx나 npm을 사용해도 좋습니다.

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

- CRA가 뭘까?



React는 **레고**같은 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
```

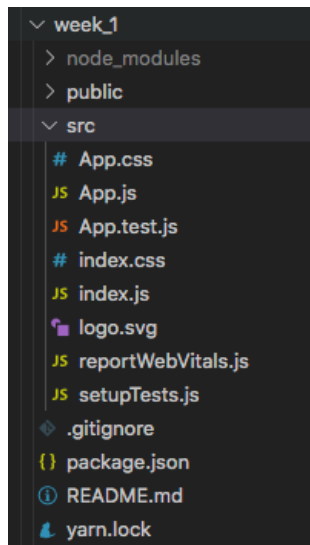


React에서는 프로젝트를 앱이라고 불러요. 리액트 프로젝트와 리액트 앱은 같은 말이니, 편하신 쪽으로 말씀하시면 됩니다.



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.

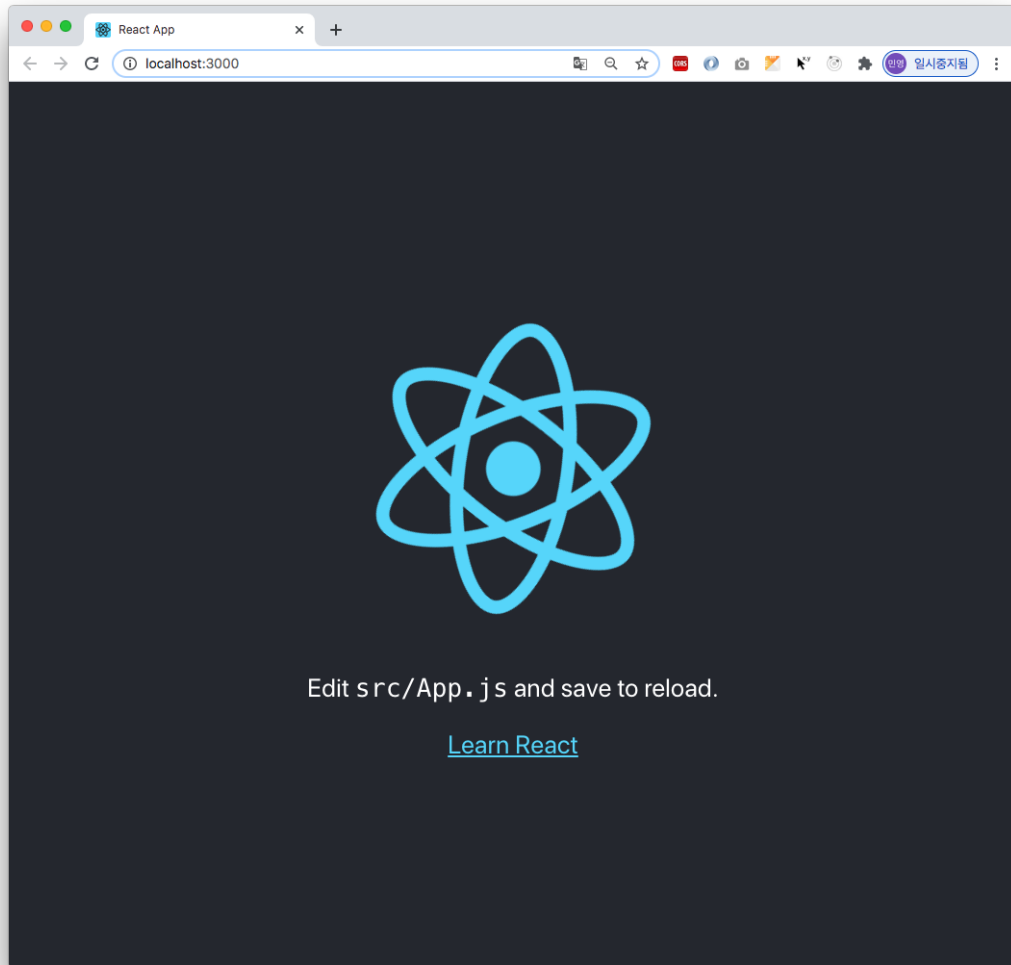


지금은 모양이 이렇게 생겼지만 알아두시면 됩니다! 하나씩 천천히 배워가요!

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. 😊



10. JSX

▼ 16) 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 가능해요!
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" />
      </header>
    </div>
  );
}
```

```

    <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>;

```



그럼 JSX에서 쓰는 <div>~~</div>는 DOM 요소인가요?

정확히는 React 요소예요! 차이가 뭐냐구요?

이건 다음주차에서 **가상돔**을 배우면서 조금 더 자세히 이야기해볼테니, 지금은 리액트 돔을 구성하는 건 **리액트 요소! 돔을 구성하는 건 돔 요소!** 라고만 알아둡시다.



어때요? 아직 아리송한가요?

괜찮아요! 써보면 느낌이 올거니까!

11. JSX 사용법

▼ 17) 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  x
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

```
// 종괄호를 두 번 쓰는 이유? 덕셔너리도 자바스크립트니까요!
// 이렇게 쓰거나,
```

```
<p style={{color: 'orange', fontSize: '20px'}}>orange</p>

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

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

12. Quiz_간단한 텍스트 입력 화면 만들어보기

▼ 18) 📌 간단한 텍스트 입력 화면 만들기

👉 JSX를 이용해서 간단한 텍스트 입력 화면을 만들어봅시다!

- 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;

```

13. 끝 & 숙제 설명



오늘 숙제는 2개예요. 😊

1. 아래 기획서를 보고, 새로운 오늘의 TODO 페이지를 만들어보세요!

(HTML과 바닐라 JS, CSS만 사용해서 해봅시다.

- 필수) 글을 쓰고, 추가하기 버튼을 누르면 투두 카드가 추가되게 해주세요!
- 마음껏 예쁘게 꾸며도 보세요! 😊

2. CRA를 사용해 리액트 프로젝트를 새로 만들어보고, 내 이름을 화면에 하나, 콘솔에 하나 띄워서 캡처해주세요!

▼ 기획서(레이아웃) 보기

오늘 할 일

독서하기

완료

입력하기 ...

추가

독서하기

완료

▼ 오늘의 TODO 예시 화면

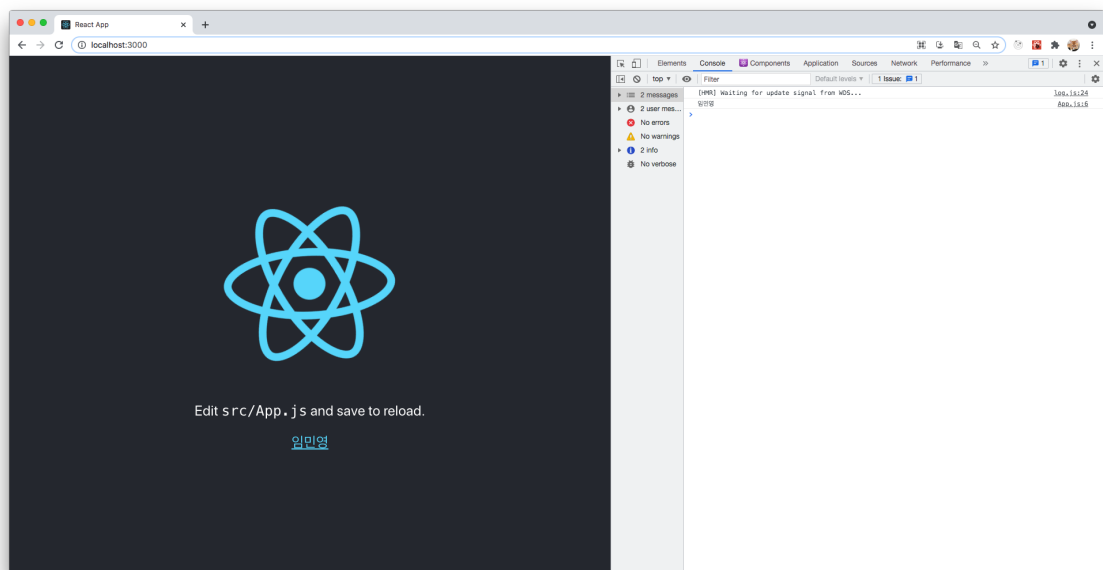
오늘 할 일

고양이 밥주기
고양이 물, 사료 챙겨주기

장보기
토마토, 계란, 초코렛 사기

추가하기

▼ 새 프로젝트 만들기 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!

HW. 1주차 숙제 해설

▼ [코드스니펫] - 1주차 숙제 답안 코드

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8" />
    <title>오늘의 투두</title>
  </head>
  <body>
    <style>
      #title {
        color: blue;
        text-decoration: underline;
      }

      .wrap {
        display: flex;
        align-items: center;
      }

      .todo-card {
        border: 1px solid gray;
        border-radius: 5px;
        padding: 2em;
        margin: 1em;
        flex: 1 1 0;
      }

      button {
        width: 100px;
      }

      button:hover {
        background-color: orange;
      }
    </style>

    <h1 id="title">오늘 할 일</h1>
    <div class="wrap">
      <div class="card-list">

      </div>

      <div class="add">
        <input id="add-input" />
        <button onClick="addCard()">추가하기</button>
      </div>
    </div>

    <script>

      // 완료하기 함수
      // button에 추가해줄 거예요!
      function completeTodo(e) {
        // 어떤 버튼을 눌렀는 지 타겟을 가져와요.
        console.log(e.target);
        // 부모(투두 카드겠죠!)를 찾아봅니다.
        console.log(e.target.parentElement);
        // 배경색을 바꿔줍니다.
        e.target.parentElement.style.backgroundColor = "green";
      }
    </script>
  </body>
</html>
```

```

    }

    // 투두 카드를 추가하는 함수
    function addCard() {
        // 투두를 감싸는 div 만들고 클래스 이름을 줍니다.
        const new_todo = document.createElement("div");
        new_todo.className = "todo-card";

        // 투두 안에 들어갈 타이틀과 버튼을 만들어요.
        const title = document.createElement("h3");
        title.textContent = document.getElementById("add-input").value;
        const button = document.createElement("button");
        button.textContent = "완료";

        // 만든 버튼에 완료하기 함수를 연결해줘요.
        button.addEventListener("click", completeTodo);
        // 타이틀과 버튼을 투두에 추가해주고,
        new_todo.appendChild(title);
        new_todo.appendChild(button);

        // 목록에 투두를 추가해줘요.
        document.getElementsByClassName("card-list")[0].appendChild(new_todo);
    }
</script>
</body>
</html>

```

Copyright © TeamSparta All rights reserved.