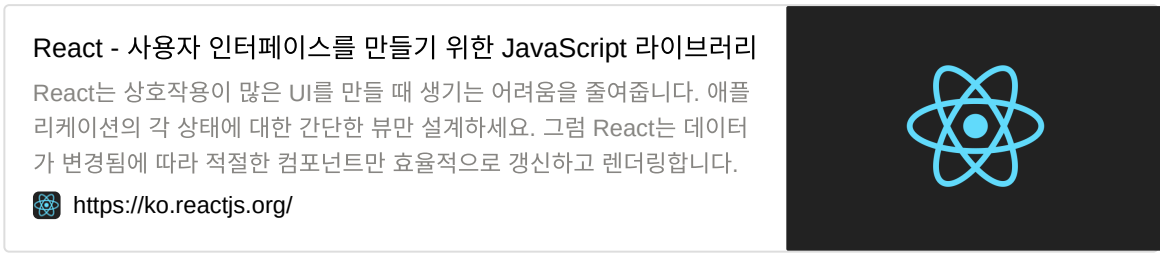
09.11_실습 강의자료

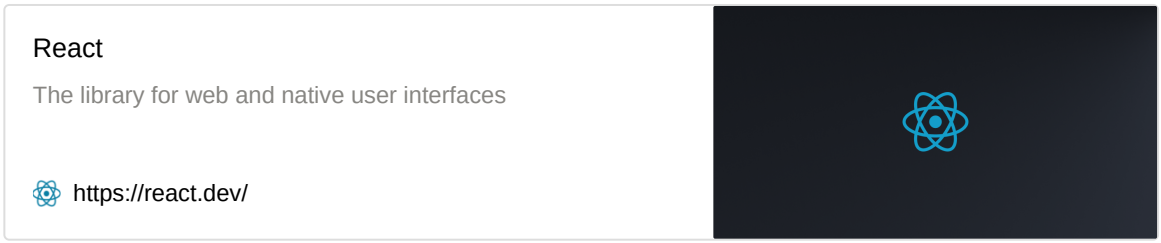
0교시

React란?

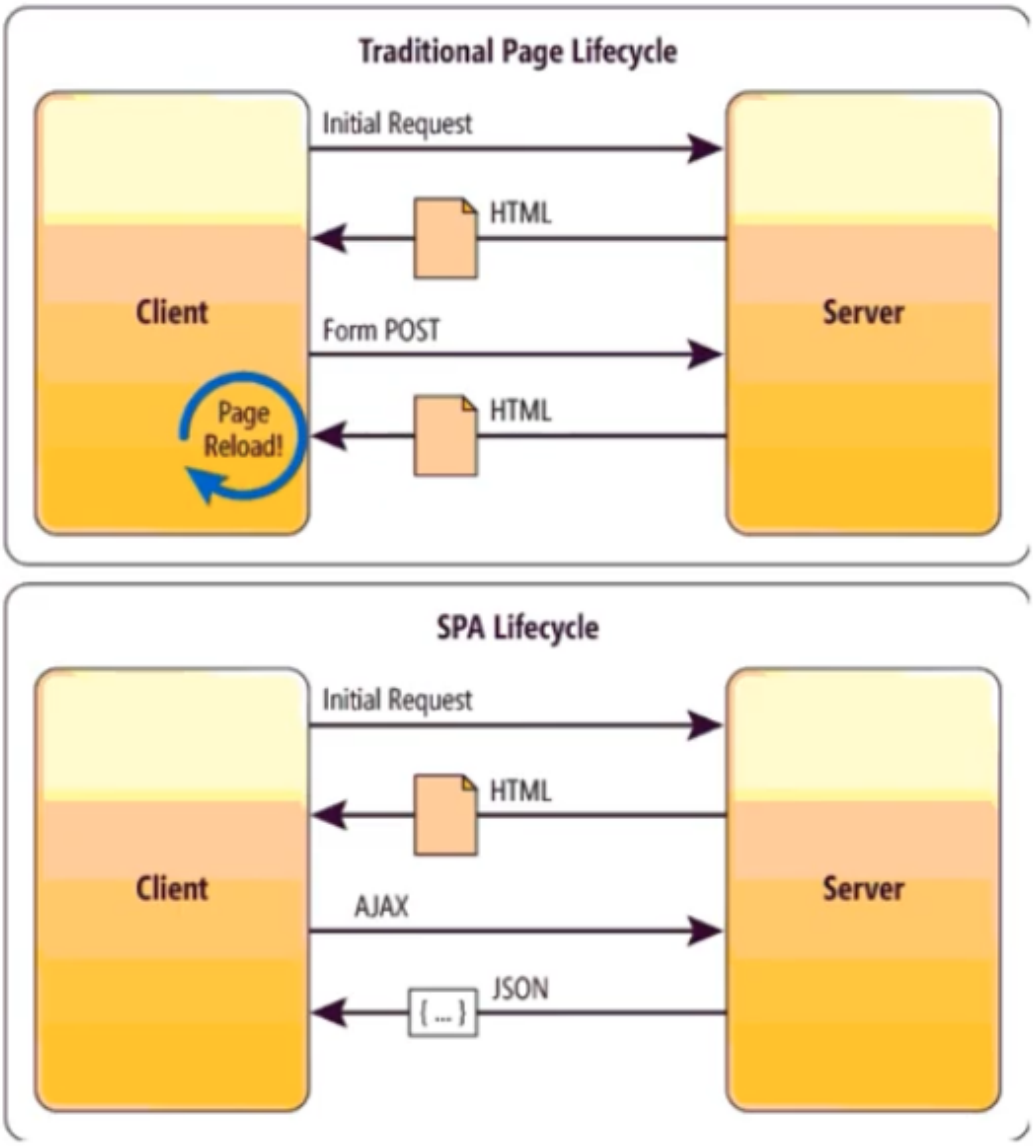
먼저 공식문서부터 확인해 볼까요?



최근 버전



SPA방식은 기존의 방식인 MPA 새로HTML을 받아와 리로딩을하는게 아닌 JSON형태로 변경된 부분만을 받아와 다시 그리게 됩니다. 좋아요를 누른다고 페이지가 새로고침이 되지않는 이유이지요!



React를 구성요소

React를 구성하는 요소들 중 큰 축을 간단하게 설명드리겠습니다.

- Component

  각요소를 독립적인 단위로 쪼개서 구현(레고블럭이라고 생각하면 편합니다.)

- Virtual DOM

  가상적인 DOM을 메모리에 저장하고 실제 DOM과 비교하여 바뀐부분을 실제 DOM에 동기화 해주는 프로그래밍 개념

이걸 사용하지 않는다면 직접 변경되는 요소를 찾아서 지워주고 새로 만들어주는 작업을 해야하지만 가상DOM을 통해 React가 대신 해줍니다.

- JSX

  JS내에서 HTML을 작성할 수 있게 해주는 템플릿 라이브러리입니다. 생산성이 높아지겠죠?

React를 배워야 하는 이유

- 1등!

  FE진영에서 가장 많은 사용자를 보유하고 있으며 가장 큰 커뮤니티가 형성되어있습니다. 당연히 취업시장에서의 수요도 가장 많구요!

  가장 큰 커뮤니티를 가지고 있는게 왜 좋으냐! 바로 많은 자료와 라이브러리를 얻을 수 있기 때문입니다.

- 생산성과 재사용성

  Compoents 단위로 개발을 하기 때문에 독립적인 요소로서 재사용성이 높아집니다.

- 활용성!

  React를 이용해서 React-Native를 통해 하이브리드 모바일 앱, SSR을 위한 NEXT.js 등 다양한 곳에서 사용하실 수 있습니다.

React 프로젝트를 만드는 방법

여러가지 방법이 있지만 CRA(create-react-app)을 사용하셔서 학습하시길 바랍니다.

CRA로 생성을 하시게 되면 webpack, babel같은 여러가지 설정을 미리 해줍니다.

CRA는 react프로젝트를 쉽게 생성할 수 있는 보일러플레이트입니다. 이 중 reack를 만든 facebook이 직접 만들었기때문에 가장 많이 쓰입니다.

먼저 react 프로젝트를 생성하기 위해서 설치해야되는것들이 있습니다.

첫번째로 node가 필요한데요 이거는 여러분 다들 설치를 하셨던 기억이 있으시죠?

node가 필요한 이유는 CRA가 node.js 환경안에서 생성되기 때문입니다.

이때 npm이라는 노드패키지매니저를 아실 필요가 있습니다. npm은 node.js에서 사용하는 각종 패키지들을 관리하는 저장소로 node.js를 설치시 자동으로 설치됩니다. 따라서 여러분이 설치를 하지 않았지만 쓸 수 있는것이죠 이 npm을 통해 라이브러리 설치, 서비 실행 등 여러가지 명령어를 사용할 수 있습니다. npm과 비슷한 패키지 매니저로 facebook에서 개발한 yarn이라는 것도 있는데요 보안과 속도 층면에서 npm보다 뛰어납니다. 기능 차이는 별루없으니 아무거나 쓰셔도 돼요 :) 근데 저는 yarn 씁니다ㅎ

그 다음으로는 create-react-app이라는 것을 다운받으셔야합니다.

이때 npx를 통해서 받는것을 추천드립니다. npx는 npm 패키지를 한번만 다온받아 실행할 때 사용하는 명령어 입니다. npm으로 create-react-app을 설치하시게 되면 create-react-app의 버전이 업그레이드 될때마다 다시 설치해주셔야겠죠? 하지만 npx를 사용한다면 가장 최신의 create-react-app을 다운받아 react 프로젝트를 만들고 creat-react-app은 다시 사라집니다.

이제 react 프로젝트가 생성되었습니다. `npm start` 을 통해서 로컬에서 한 번 실행을 해보겠습니다.

기본적으로 뭐가 많이 설치되어있죠? 이게CRA의 장점입니다.

디렉토리별 어떤 기능들이 있는지 살펴보겠습니다.

1교시

JSX문법

JS과 HTML이 합쳐진거라고 생각하시면 편합니다.

우리가 원래 작성하던 방식으로HTML을 JS에서 작성해주는 것이죠

원래 JS에서 HTML을 작성하려면 어떻게 해야했죠? element 생성해주고, 속성 뭐 넣을지 적어주고, 어디 들어갈지 정해주고, … 코드가 꽤 필요했었죠?

하지만 JSX문법을 사용하게 된다면 그냥 return 부분에 HTML 코드를 작성하듯 작성하면 됩니다. 나머지 변환과정은 react가 알아서 해줘요🙂

HTML을 작성할 때와 몇가지 다른 점이 있긴합니다. 예를들어 속성값을 camel case로 작성해야하며, class 대신 className을 사용하는 것 등 몇가지 차이가 있습니다.

JSX문법의 유의점들을 알아보겠습니다.

JSX는 Babel에 의해 자동으로 JS로 변환이 됩니다. 우리가 이전에 작성했었던 createElement를 이용해서 DOM요소들 직접만드는 것이죠 이거를 react에서 알아서 해주니 우리는 HTML을 작성하는것처럼 코드를 작성하면 됩니다.

이때 JSX의 특징을 생각해야 하는데요

1. HTML내에서 JS사용가능
    - JSX :
    - JS : react.createElement(”div”, {className: “App”)
2. class ⇒ className으로
3. style을 object로, style의 Property는 camelCase로 작성합니다.
4. 닫는태그가 필수입니다.
5. 최상단 element는 하나만 있어야 합니다. (React.Fragment사용하면 편합니다.)

참고 : https://react.dev/learn/writing-markup-with-jsx

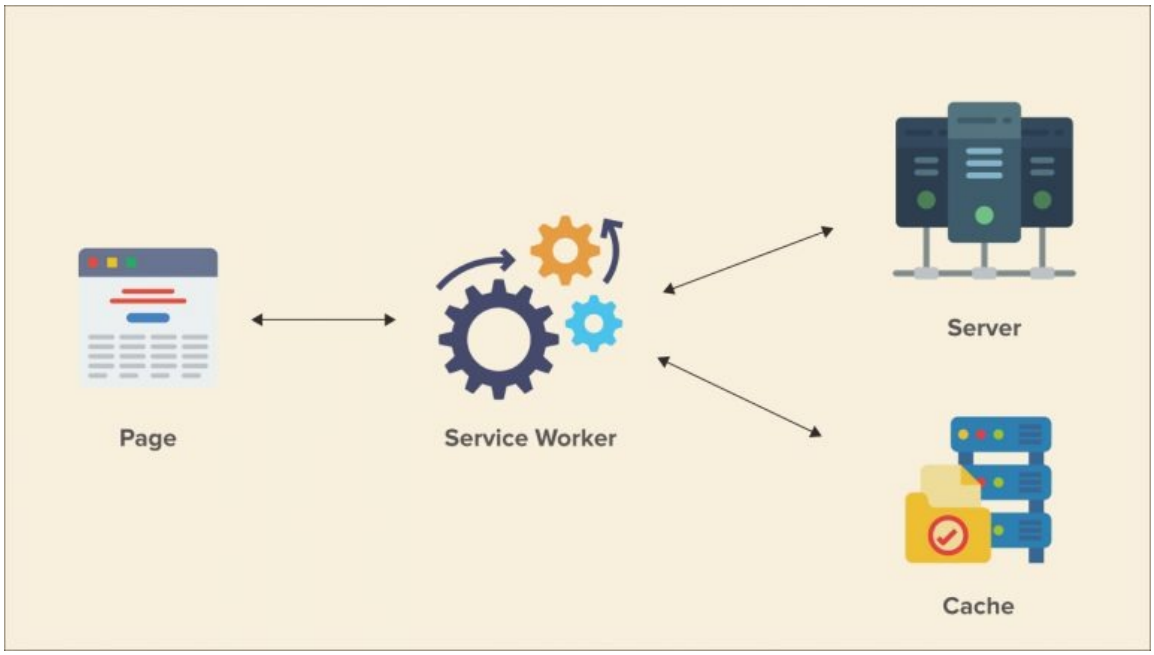
▼ 01. HTML을 JSX로 변환하기

```
const element = (
  <div>
    <h2>코더랜드에 오신것을 환영합니다:)</h2>
    <h2>즐거운 React! 함께 공부해봐요~</h2>
  </div>
);
```

▼ serviceWorker란?

서비스 워커는 기본적으로 웹 애플리케이션, 브라우저 및 네트워크(사용 가능한 경우) 사이에 있는 프록시 서버 역할을 합니다. 쉽게 말해 client와 server 사이에 중간 다리 역할을 하는 것입니다.

이를 통해 오프라인 기능, 네트워크 요청을 가로채고 서버를 업데이트 서버의 변경사항에 따라 client에게 알람을 띄우는 푸시알림 및 백그라운드 동기화 API 여러가지 기능을 할 수 있습니다.



가장 대표적인 예시로 PWA를 만들때를 예시로 들 수 있는데요

PWA는 웹을 앱처럼 쓸 수 있게 해줍니다. 따라서 앱처럼 다운로드 기능과 캐싱을 통한 오프라인 기능, 서버의 변경사항에 따른 푸시알림 기능을 구현할 수 있습니다.

참고 : https://web.dev/learn/pwa/service-workers/

▼ 02. 함수에 JSX 활용하기

```
//인사문구를 출력하는 함수
//formatName()함수를 사용해 출력문구를 완성합니다.
function getGreeting(user) {
  return <h1>Hello, {formatName(user)}!</h1>;
}
```

```
//getGreeting()의 결과값을 element에 저장합니다.
const element = getGreeting(user);
```

2교시

React의 컴포넌트

UI기반으로 선언된 것을 component라고 합니다! react의 가장 작은 단위의 개발이며

쉽게 이해하기 위해서 레고로 자동차를 만든다고 생각해보겠습니다. 바퀴도 필요하고 창문도 필요하고 의자도 필요하고 여러가지가 필요하겠죠? 같은게 겹치는 것도 있을 것이구요.

그러한 하나의 단위를 컴포넌트로 만들면 재사용하기에도 편리하고 유지보수하기에도 편리합니다.

이전에는 class형 컴포넌트를 만들어서 사용했지만 최근에는 함수형 컴포넌트를 만들어 사용하고 React도 그것을 권장합니다. 따라서 함수형 컴포넌트를 만드는 방법에 대해서 알아보겠습니다.

component를 만들때도 몇가지 규칙이 있습니다.

1. 반드시 대문자로 시작해야합니다.
2. attrubute에 해당하는 부분을 props라고 합니다.
3. component안에 작성된 하위 element를 children이라고 합니다.
4. component를 다른 곳에서 사용하려면 export, import를 해줘야하고 HTML태그처럼 JSX문법을 이용해서 사용합니다.

▼ 03. 함수 컴포넌트 만들어보기

```
function Welcome() {
  return <h2> Welcome, Sara!</h2>;
}

const element = <Welcome />;
```

▼ 04. 컴포넌트 합치기

src/components/Profile.js

```
import React from 'react';

// Comment 컴포넌트 import
import { Comment } from './Comment.js';
// UserInfo 컴포넌트 import
import { UserInfo } from './UserInfo.js';

import '../index.css';

function Profile() {
  const user1 = {
    name: '엘리스 토끼',
    age: '12',
  };
  const text1 = 'React는 재밌다!!';

  return (
    <div className="profile">
      <Comment text={text1} />
      <UserInfo user={user1} />
    </div>
  );
}

export { Profile };
```

src/App.js

```
import { Profile } from './components/Profile.js';

function App() {
  const title = '사용자 프로파일';
  return (
    <div>
      <h1>{title}</h1>
      <Profile />
    </div>
  );
}
```