7월 7일 기술면접 문제

1. 서버사이드 렌더링이란 무엇인가.

- 렌더링: 요청 받은 내용을 브라우저 화면에 표시하는 것

1. 로더(Loader)가 서버로부터 정보들을 불러옴

2. 파싱(Parsing)을 통해 문서를 DOM 트리를 만듦

3. DOM 트리가 구축되는 동안 브라우저는 렌더 트리를 구축한다.

4. CSS 설정/레이아웃 위치 지정

5. 렌더링 트리가 그려짐

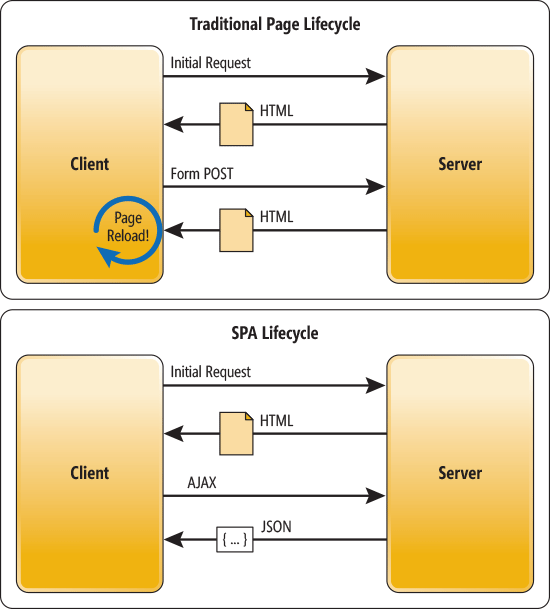
- 서버 사이드 렌더링(SSR)

* SSR : Server Side Rendering
* 전통적인 웹 어플리케이션은 서버 사이드 렌더링 방식
* 요청시마다 서버에서 처리 한 후 새로고침으로 페이지에 대한 응답(View)
* 웹에서 많은 정보와 기능이 많아지면서 SPA개념이 생김

출처: <https://linked2ev.github.io/devlog/2018/11/15/Javascript-2.-What-is-SSR/>

2. SPA란 무엇이고 어떤 장점이 있는가.

SPA는 Single Page Application의 약자로, SPA는 클라이언트 사이드 렌더링 방식을 채택한다. 여기서 클라이언트 사이드 렌더링 방식이란, 클라이언트에서 자바스크립트를 통해 렌더링 하는 방식을 의미하는데, SPA는 브라우저에 로드되고 난 뒤에 페이지 전체를 서버에 요청하는 것이 아니라 최초 한번 전체 페이지를 다 불러오고 HTML에 번들링된 자바스크립트가 실행되면서 페이지 렌더링을 한다. 그리고 렌더링 후 클라이언트 요청에 따라 응답 데이터만 다시 페이지의 특정 부분만 렌더링하는 웹 어플리케이션 개념이다.



[그림 - 전통적인 페이지 vs 단일 페이지 어플리케이션 비교]

그렇다면, 장단점은 어떻게 될까?

장점으로는,

1. 자연스러운 사용자 경험(UX)

2. 필요한 리소스만 부분적으로 로딩(성능)

3. 서버의 탬플릿 연산을 클라이언트로 분산(성능)

4. 컴포넌트별 개발 용이(생산성)

5. 모바일 앱 개발을 염두에 둔다면 동일한 API를 사용하도록 설계 가능(생산성)

단점으로는,

1. JavaScript 파일을 번들링해서 한 번에 받기 때문에 초기 구동 속도 느림

2. 검색엔진최적화(SEO)가 어려움

3. 보안 이슈 (프론트엔드에 비즈니스 로직 최소화)

가 있다.

출처: <https://linked2ev.github.io/devlog/2018/11/15/Javascript-2.-What-is-SSR/>

<https://medium.com/aha-official/%EC%95%84%ED%95%98-%ED%94%84%EB%A1%A0%ED%8A%B8-%EA%B0%9C%EB%B0%9C%EA%B8%B0-1-spa%EC%99%80-ssr%EC%9D%98-%EC%9E%A5%EB%8B%A8%EC%A0%90-%EA%B7%B8%EB%A6%AC%EA%B3%A0-nuxt-js-cafdc3ac2053>

3. 리액트에 사용되는 디자인패턴은 어떤 것들이 있는가. 각각 설명하시오.

Container + Presenter Pattern : container 컴포넌트와 presenter 컴포넌트를 분리시킨 패턴

- container 컴포넌트는 논리 담당, logic과 api를 갖고 있고,

- presenter 컴포넌트은 UI 담당, 사진 등을 갖고 있다.

ex) 홈 라우터를 예시로 보자

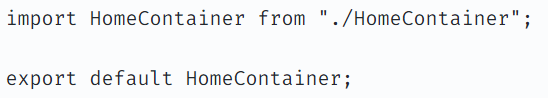
폴더 구조

Routes/Home/index.js

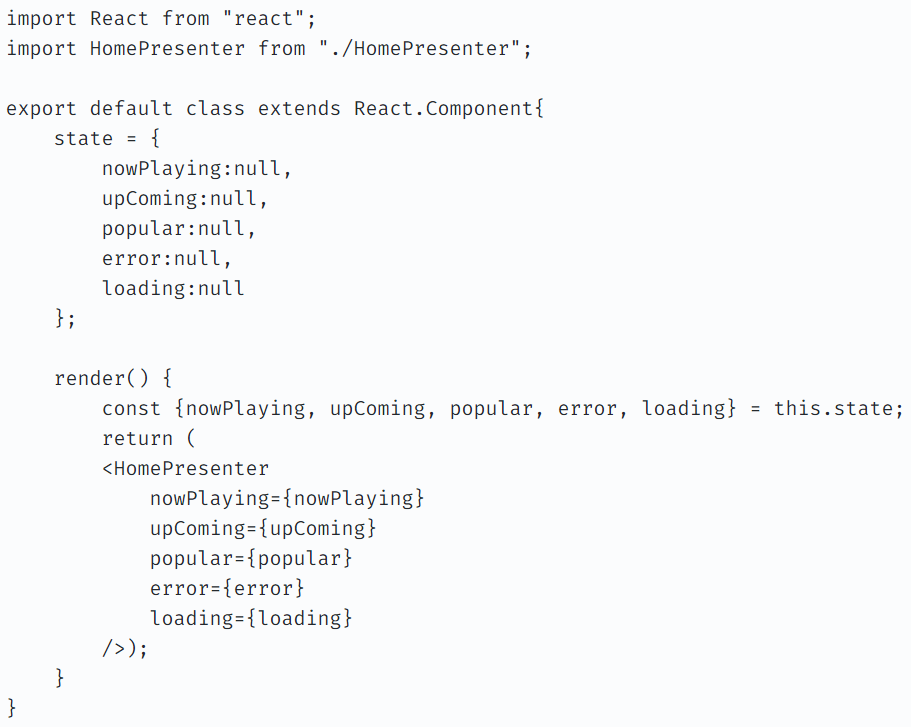
Routes/Home/HomePresenter.js

Routes/Home/HomeContainer.js

index.js (홈 라우터의 인덱스 파일은 홈컨테이너를 호출)



HomePresenter.js (홈컨테이너는 홈프리젠터를 렌더링)



HomePresenter.js (보이는 화면)



출처: https://velog.io/@mollang/19.12.05-리액트-디자인-패턴