ALL ABOUT FRONTEND_소프트웨어공학과 노욱진

1. 프론트엔드란
    (1) 클라이언트 사이드:사용자 측 처리,CSLvsSSR
    (2) 반응형 웹:브라우저, 기기, 스크린 사이즈에 따른 지원
    (3) 웹뷰 구현:프론트엔드의 웹, 앱환경 담당
    (4) 디자인 구현: 웹 퍼블리셔와 커넥
    (5) 기능 개발:사용자가 주로 사용하는 기능을 구현
    (6) 어플리케이션(웹앱) 최적화:로딩, 반응속도, 불필요한 데이터 제거, 비동기 프로그래밍 등 사용자가 최적화된 웹을 경험할 수 있도록
    (7) 다양한 브라우저, 운영체제 지원:크롬, 웨일, 사파리, 파이어폭스 등 브라우저별 지원 상이
    (8) 프론트엔드VS백엔드
        a. 프론트엔드
            - 사용자가 보는 화면, 사용자 인터페이스 담당
            - html, css, js 등
        b. 백엔드
            - 사용자가 보지 못하는 영역인 서버, 데이터베이스 등을 담당
            - asp, jsp, php등
        c. FE+BE+DB = FULL STACK
    (9) 프론트엔드 더 깊이
        a. 데이터를 잘(예쁘게)사용자에게 보여주기
            - 디자인을 html, css로 구현
            - 기기, 브라우저, 화면에 맞게 디자인이 제대로 보이도록
        b. 데이터(화면)를 조작
            - 현재의 화면을 데이터로 구성
            - 사용자의 이벤트를 감지 및 분류
            - 행동에 맞는 적절한 API 동작
            - 새로운 데이터 생성
            - 생성된 데이터를 화면에 출력
        c. 서버에 데이터 보내기
            - API 문서 숙지
            - 서버에서 정의한 schema에 맞게 내부적으로 type 설정
            - 화면에 있는 데이터를 API에서 사용하는 구조로 변경
            - 백엔드에서 지정한 API의 양식에 맞춰 URL, Method, Header, Param, Body등을 만들어 전달
        d. 서버에서 받은 데이터를 다루기
            - 서버에 있는 데이터를 전달받아 화면에 필요한 데이터로 전환하여 변환하는 과정
            - 비동기성 특징으로 인한 로딩, 중복방지 등 중간과정 처리
            - 서버응답에 따른 문제 원인 확인 및 문제 해결
2. 프론트엔드 장단점
    (1) 장점
        a. 눈에 보인다 ⇒결과물을 실시간으로 확인 가능
        b. 언어에 대한 고민이 적음: js⇒수많은 라이브러리, 다양한 프레임워크
        c. 혼자 프로젝트 시작하기 용이(백엔드 처리firebase, AWS lambda etc,,)
            - BE As a Services는 많으나 반대는 적음

- 백엔드 서비스(BasS):개발자가 웹 또는 모바일 애플리케이션의 모든 뒷면을 아웃소싱하여 프론트엔드만 작성하고 유지관리하면 되는 클라우드 서비스 모델

d. 서버가 터졌을 때(배초안정화이후):긴급 대응이 거의 없음

(2) 단점

a. 급변하는 기술 트렌드
b. 중간계(디자이너, 프론트엔드, 백엔드)

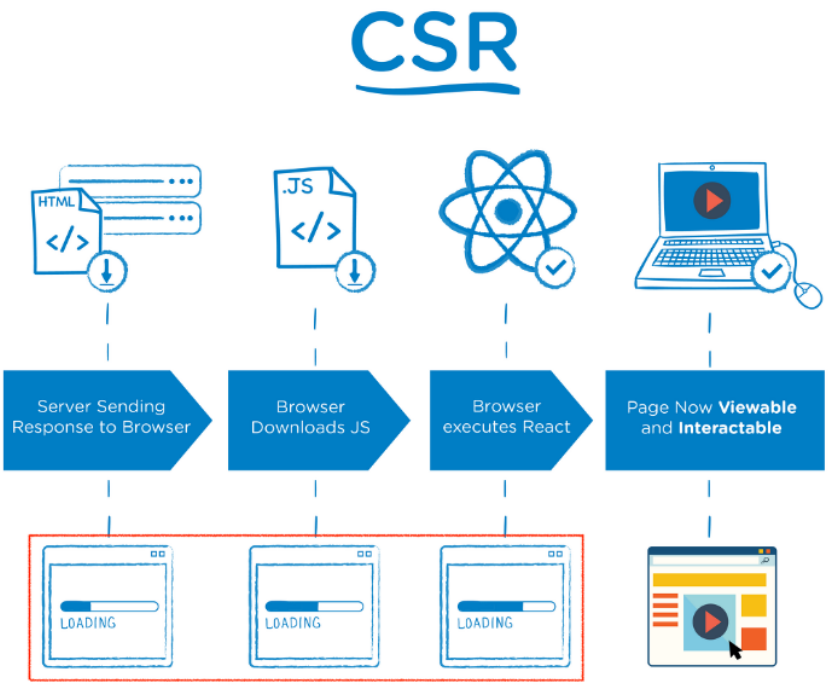
3. 프론트엔드의 언어, 프레임워크

(1) 웹 개발자 공통 필수 언어:HTML, CSS
(2) 언어:JS, 프레임워크:JS-TS/REACT-NEXT.JS

a. typescript:자바스크립트 프로그램+타입체커를 통과한 자바스크립트 프로그램
⇒타입 주입으로 안정성 보완
b. react:자바스크립트, html요소를 편하게 만들 수 있도록 해주는 자칭 라이브러리
- 다른 프렝ㅁ워크보다 리액트가 가장 자바스크립트 문법의 원형을 띔

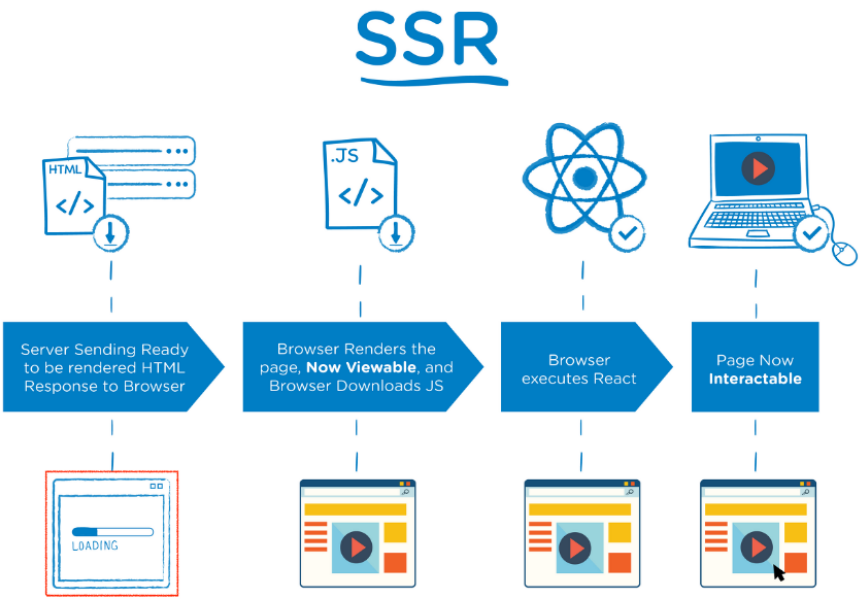
c. next.js
- Client Side Rendering
i)렌더링이 클라이언트 쪽에서 일어남, 서버는 요청을 받으면 클라이언트에게 html, js를 보내줌
→user가 website 요청을 보냄
→cdn이 html파일과 js로 접근할 수 있는 링크를 클라이언트로 보냄(CDN:AWS의 cloudflare, 엔드 유저의 요청에 물리적으로 가까운 서버에 요청을 응답)
→클라이언트는 html과 js를 다운로드 받음(유저 빈화면)
→다운 완료된 js 실행, 데이터를 위한 API 호출(유저 PLACEHOLDER 확인)
→서버가 API로부터 요청에 응답
→API로부터 받아온 data를 placeholder자리에 넣어줌

ii)



- Server Side Rendering
  i) 서버쪽에서 렌더링 준비를 끝 마친 상태로 클라이언트에게 전달
  →user가 website 요청을 보냄
  →sever는 즉시 렌더링 가능한 html파일을 만듬(리소스 체크, 컴파일 후 완성된 html 컨텐츠로 만듬)
  →클라이언트에 전다로디는 순간, 이미 렌더링 준비가 되어있기 때문에 html은 즉시 렌더링됨. 단, 사이트 자체 조작 불가능(js 읽히기 전)
  →클라이언트가 자바스크립트를 다운받음
  →다운 받아지고 있는 사이에 유저는 컨텐츠는 볼 수 있지만 사이트를 조작할 수는 없음
  → 브라우저가 js프레임워크를 실행
  → 웹 상호작용

iii)



(3) react vs next.js

a. 웹 페이지 로딩 시간
   - 첫 페이지 로딩 시간:CSR(한번에)/SSR(필요한 부분만)
   - 나머지 로딩시간:CSR(한번에)/SSR(정확히 다시 시작)

b. SEO 대응
   - 검색엔진은 자동화 로봇인 '크롤러'로 웹 사이트 리딩
   - CSR:JS실행시켜 동적으로 컨텐츠 생성
     →JS실행되어야meatadata 변화
   - SSR:서버사이드에서 컴파일되어 클라이언트로 넘어오기 때문에 대응 용이

c. 서버 자원 사용
   - CSR<SSR:매번 서버에 요청

4. 준비과정

(1) 기초 언어 학습
(2) 클론 코딩
(3) 프레임워크 및 라이브러리 학습
(4) 다양한 프로젝트 시도
(5) 교내 동아리, 대외활동, 부트캠프 활동
(6) 자바스크립트 동작 원리 공부