프론트엔드 면접질문 및 답변사항 리스트

브라우저의 렌더링 과정에 대해서 상세하게 설명해달라

브라우저 주소창에 www.naver.com 치면 -> 네이버 서버를 찾아간다. -> DNS(실제 서버가 어디에있는지 알고 있는 서버)가 연결해줄 곳을 찾음 -> (여기서 주소 앞에 https가 붙었다면 https방식으로 통신하겠다.)-> 서버의 기본설정이 대부분 index.html되어 있어 서버에서 이파일을 클라이언트로 보냄 -> 브라우저는 텍스트로 이루어진 index.html 파일을 파싱한다. -> 한줄한줄 읽으면서 DOM트리를 만들어나감. -> 중간에 link태그를 만나 css요청이 발생하면, 요청과 응답과정을 거치고 css를 파싱함 -> CSS파싱이 끝나면 중단된 html을 다시읽고 DOM트리를 완성 -> 완성된 DOM트리와 CSSOM트리를 합쳐 Render Tree를 만들고 그린다.javascript는? -> 중간에 HTML파서는 Script태그를 만나게 되면 javascript 코드를 실행하기 위해 파싱을 중단 -> 제어권한을 자바스크립트 엔진에게 넘기고, 자바스크립트 코드 또는 파일을 로드해서 파싱하고 실행

Http와 Https 통신 방식의 차이?

결정적 차이는 보안이다. 1. http방식은 네트워크상에서 정보를 누군가가 마음대로 열람, 수정이 가 / https는 누가 볼수없도록 막음.2. http방식이 https방식보다 빠르다. 3. Http방식은 민감한정보를 다룰 때 항상 변조, 해킹 가능성을 생각해야한다. Https는 설치 및 인증서를 유지하는데 추가적인 비용이 발생. -> 따라서, 민감한 정보가 있는 페이지의 경우 Https 그럴필요가없으면 http로 만들면 된다.

OOP에 특징에 대해 설명해달라(상속, 캡슐화 등등...)

Object Oriented Programming 객체지향 프로그래밍이라고한다. 특징은 크게 4가지가 있다.
1. 상속 : 클래스개념에서 상위 클래스(부모)로 부터 하위 클래스(자식)이 유산을 물려받는것과 같이, 부모의 메소드나 변수를 사용할 수 있는 것을 말함.
2. 다형성 : 같은 함수가 있다고 칠대 그 함수가 매개변수에 따라 다른 역할을 할 수 도 있다.
3. 캡슐화 : 보통 데이터를 은닉시킨다고 표현하는데, 외부에서 쉽게 데이터를 접근할 수 없게 만들기도하고, 데이터 구조와 데이터를 다루는 방법들을 한데다 묶는것.
4. 추상화 : 공통적인 속성이나 기능을 묶어서 이름을 붙이는 것 (a b d 이런게있다고 치면 이런건 알파벳이라고 묶을 수 있다)

함수형 프로그래밍(Function Programming)

함수형 프로그래밍에 대해 설명해달라

함수형 프로그래밍은 순수함수와 보조 함수의 조합을 통해 로직내에 존재하는 조건문과 반복문을 제거하여 복잡성을 해결하고 변수의 사용을 억제하여 상태 변경을 피하려는 프로그래밍 패러다임이다.

함수형 프로그래밍에 개념에서 순수함수란 무엇인가

순수함수는 같은 입력이 주어지면, 같은 출력을 반환해야하고, side effect(부작용) 이 없어야한다.

결국, 함수형 프로그래밍은 순수함수를 통해 sideeffect를 최대한 억제하여 오류를 피하고 프

로그램의 안정성을 높이려는 노력의 한 방법
OOP와 함수형 프로그래밍의 가장 큰 차이점은 무엇인가
객체지향은 객체 안에 상태를 저장하고, 이 상태를 이용해서 메소드를 추가하고 상태변화를 설정하고 조정하기위해 다양한 기능을 사용한다. 이에 반해 함수형 프로그래밍은 상태를 제어하는것보다 상태를 저장하지 않고 없애는데 주력한다.
예를들면, 객체 지향은 상태를 저장하는 필드와 그 필드들을 이용해 기능을 제공하는 메소드를 만들고 클래스를 만듭니다. 반면 함수형은 몇몇 자료구조(list, map, set) 등을 이용해 최적화된 동작을 만들어낸다.

[웹 프로토콜]
웹 프로토콜이란?
웹 프로토콜은 웹에서 쓰이는 통신규약입니다. 통신규약이라는 것은 쉽게 설명하면, 통신을 할때 내가 이렇게 할게 너는 이렇게 해줘 라고 약속하는 것입니다.

Http 통신이란?
웹 프로토콜중 하나로 HTTP가 가장 많이 쓰이는데 Hyper text Transfer Protocol의 약자입니다.
쉽게말하면, 인터넷에서 데이터를 주고 받을 수 있는 통신규약 정도로 보시면됩니다. 요청과 응답으로 이루어져있어 어떤 데이터 주세요하고 요청하면, 이 데이터 줄게요 라고 응답합니다.

Http 1.1과 2.0의 차이는?
가장 큰 차이는 속도이다. 2.0같은 경우는 헤더를 압축해서 보내기도하고, 한번의 연결로 동시에 여러 메시지를 주고 받을 수도 있다.

[비동기 프로그래밍(Asynchronous)]
AJAX란 무엇인가
자바스크립트를 이용해 비동기적으로 서버와 브라우저가 데이터를 교환할 수 있는 통신 방식
보통은 서버로부터 웹페이지가 반환되면 전체를 갱신해야하는데 / AJAX를 사용하면, 페이지 일부만을 갱신하고도 동일한 효과를 볼 수 있다. 즉, 갱신이 필요한 부분만 로드하여 갱신하면 되므로 빠르고, 부드러운 화면효과를 기대할 수 있음

Promise와 Callback의 차이점은 무엇이며 각각의 장단점에 대해 설명해달라
Promise란 무엇이며 코드가 어떻게 구성되어 있는가

둘 다 자바스크립트에서 비동기처리를 위해서 사용되는 패턴이며,
Callback 같은 경우 함수의 처리 순서를 보장하기 위해서 함수를 중첩하게 사용되는 경우가 발생해 콜백헬이 발생하는 단점과 에러처리가 힘들다라는 단점이 있다.
그래서 나온게 Promise이며 ES6부터 정식 채택되어 사용중이다.
Promoise 생성자 함수를 통해 인스턴스화하며,

```
// Promise 객체의 생성
const promise = new Promise((resolve, reject) => {
  // 비동기 작업을 수행한다.

  if (/* 비동기 작업 수행 성공 */) {
    resolve('result');
  }
  else { /* 비동기 작업 수행 실패 */
    reject('failure reason');
  }
});
```

비동기 처리에 성공하면 resolve메소드를 호출해서 비동기 처리 결과를 후속처리 메소드로 전달한다.
비동기 처리에 실패하면 reject메소드를 호출해서 에러메시지를 후속처리 메소드로 전달한다.
후속처리메소드는 then과 catch가 있다. 둘다 Promise를 반환한다.
then 을 가지고 메소드 체이닝을 통하여서 콜백헬 문제를 해결 할 수 있다.

Async, Await가 무엇이며, 사용해본 경험이 있는가
Async, Await와 Promise의 차이는
Promise를 더욱 쉽게 사용할 수 있도록 ES2017(ES8) 문법이다.
함수의 앞부분에 async 키워드를 추가하고, 함수 내부에서 Promise의 앞부분에 await 키워드를 사용한다.
async, await를 사용할 경우 코드가 간결해지지만, 에러처리를 잡기 위해 try catch를 사용해야한다. 동기적인 코드흐름으로 개발이 가능하다.

[자바스크립트의 타입]
자바스크립트의 Number Type은 다른 언어들과 차이점이 무엇인가, 왜 하나만 존재하나.
다른 언어에는 int double 등 숫자타입의 다양함이 있지만, number는 하나만 있다.
정수만을 위한 타입이 없고, 모든 수를 실수를 처리한다.
자바스크립트의 원시 타입은 몇가지인가? 종류는?
boolean, string, number, undefined , null , symbol 이렇게 6가지 종류
undefined는 선언만 되어있고 값은 없는 상태 , null은 자료형이 객체이며 빈값을 의미

실행 컨텍스트(Execution Context)에 대해 설명해달라
자바스크립트의 코드들이 실행되기 위한 환경
전역 컨텍스트 ,함수 컨텍스트 2가지 존재
전역 컨텍스트 하나 생성 후에 함수 호출할 때마다 함수 컨텍스트가 생성
컨텍스트를 생성시에 변수객체, 스코프 체인, this가 생성된다.
컨텍스트 생성 후 함수가 실행되는데 사용되는 변수들은 변수 객체 안에서 값을 찾고 없다면 스코프체인을 따라 올라가며 찾음.
함수 실행이 마무리되면 해당 컨텍스트는 사라짐. 페이지가 종료되면 전역 컨텍스트가 사라짐

즉, 자바스크립트의 코드가 실행되기 위해서는 변수객체, 스코프체인, this 정보들을 담고 있는 곳을 실행컨텍스트라고 부른다.

자바스크립트의 호이스팅(Hoisting)은 어떻게 이루어져 있는가

변수를 선언하고 초기화 했을때 선언부분이 최상단으로 끌어올려지는 현상

예를들어, 코드 상단에서 console.log(a)를 찍고 하단에서 var a=1; 이라고 하였을때 a는 undefined라고 나온다. 이런 현상을 호이스팅이라고한다. 함수의 경우 함수표현식은 호이스팅이 적용되지 않으나 일반 함수선언문은 함수 호이스팅이 적용된다.

클로저(Closure)란 무엇이며, 왜 이러한 패턴을 사용하는가

반환된 내부함수가 자신이 선언됐을때의 환경인 스코프를 기억하여 자신이 선언되었을 때의 환경 밖에서 호출되어도 그 환경에 접근할 수 있는 함수, 자신이 생성될때의 환경을 기억하는 함수

사용 하는 이유 :

1) 현재 상태를 기억하고 변경된 최신 상태를 유지하기 위해
2) 전역 변수의 사용을 억제 하기위해
3) 정보를 은닉하기 위해

가비지컬렉터의 역할은? 어떻게 동작?

메모리 할당을 추적하고 할당된 메모리 영역이 필요하지 않은 영역일 경우를 판단해서 회수하는 것.

자바스크립트에서 변수는 직접적으로 참조 값(문자열, 객체, 배열 등)을 담고 있지 않고, 해당 값을 메모리 상에 저장 된다. 그래서 참조 값을 생성하고나서 더이상 참조할 것이 없거나 비어졌을 때 가비지 컬렉터가 동작해서 메모리가 반환됨.(메모리를 다시 재사용할 수 있는 상태가 된다)

자바스크립트의 순환참조란? 어떤게 문제이고 해결방법은?

```
function f(){
  var o = {};
  var o2 = {};
  o.a = o2; // o는 o2를 참조한다.
  o2.a = o; // o2는 o를 참조한다.

  return "azerty";
}

f();
```

위와 같이 서로 순환되어서 참조되어져서 가비지컬렉터가 작동하지 않고 메모리 누수가 발생된다.

null을 할당해서 연결을 끊는 방법을 사용한다.

대부분의 브라우저에서는 Mark and sweep알고리즘을 사용. 그래서 가비지컬렉터가 참조되지 않는 객체가 있을 때 동작하는 것이 아니라 접근 할 수 없는(닿을 수 없는) 객체 일 때 동

작한다.

자바스크립트의 배열이 실제 자료구조 배열이 아닌데 그 이유는?
자바스크립트의 배열은 실제 자료구조의 배열과 다르게 HashMap으로 구현되어있다. 이 HashMap을 구현하기 위해서는 연결리스트로 구현하게 되는데 연결리스트에서 값을 찾기 위해서는 탐색해나가면서 값을 찾는 불상사가 발생한다. 이를 해결하기 위해서 타이핑된배열(Int8Array,Float32Array 등) 이 추가되고 있다.

이벤트 루프에 대해서 설명, 동시성 모델에 대해서 설명
자바스크립트는 싱글 스레드 기반 언어이다. 함수를 실행하면 함수 호출이 스택에 순차적으로 쌓이고 스택의 맨위에서부터 아래로 한번에 하나의 함수만 처리 할 수 있다.
하지만, 자바스크립트에는 이벤트 루프라는것을 통해 동시성을 지원한다. (동시에 일어나는 것이 아니라 동시에 일어나는 것처럼 보이게 하는것이다!)
이벤트 루프는 콜 스택에서 실행 중인 게 있는지 확인하고, Event queue에 작업이 있는지 확인해서 콜스택이 비어있다면 이벤트큐 내의 작업이 콜스택으로 이동되어서 실행된다.

프로토타입이란?
자바스크립트는 프로토타입을 기반으로 상속을 구현하여 불필요한 중복을 제거(중복 제거 방법은 기존의 코드를 재사용하는것!!)
즉, 생성자 함수가 생성할 모든 인스턴스가 공통적으로 사용할 프로퍼티나 메소드를 프로토타입에 미리 구현해 놓음으로써 또 구현하는것이 아니라 상위(부모) 객체인 프로토타입의 자산을 공유하여 사용할 수 있다.
__proto__ 접근자 프로퍼티로 자신의 프로토타입, 즉 Prototype 내부슬롯에 접근 할 수 있음.

프로토타입체인이란?
객체의 프로퍼티에 접근하려고 할때 객체에 접근하려는 프로퍼티가 없으면, __proto__접근자 프로퍼티가 가리키는 링크를 따라 자신의 부모역할을 하는 프로토타입의 프로퍼티를 순차적으로 검색한다. 프로로타입체인의 최상위 객체는 Object.prototype이다. 이 객체의 프로퍼티와 메소드는 모든 객체에게 상속된다.
prototype 프로퍼티 는 생성자함수가 생성할 인스턴스의 프로토타입을 가르킨다.

[This]
자바스크립트에서 This는 몇가지로 추론 될수 있는가, 아는대로 말해달라
일반함수의 this와 화살표 함수의 this는 어떻게 다른가?
자바스크립트의 내부함수는 일반 함수, 메소드, 콜백함수 어디에서 선언되었든지 this는 전역객체를 가르킴
일반함수의 this는 window(전역)을 가르키며, 화살표 함수의 this는 언제나 상위스코프의 this를 가르킴

Call, Apply, Bind 함수에 대해 설명해달라
3가지 방법은 this를 바인딩하기 위한 방법이다.
Call은 this를 바인딩하면서 함수를 호출하는 것, 두번째 인자를 apply와 다르게 하나씩 넘기는 것
Apply는 this를 바인딩하면서 함수를 호출하는 것, 두번째인자가 배열
Bind는 함수를 호출하는 것이 아닌 this가 바인딩 된 새로운 함수를 리턴함.

use strict모드에서의 this?

일반함수에서의 this는 undefined가 바인딩 됨.

[ES6]
크롬 정도의 브라우저를 제외하곤 ES6 스펙에 대한 지원이 완벽하지 않다. 해결방법은 무엇인가
Babel을 사용한다. ES6이상의 문법의 코드들을 브라우저가 이해할 수 있게끔 ES5이하의 문법으로 변환

Babel이란?
babel은 컴파일러 인가 ? 트랜스파일러인가?
트랜스파일러이다. 컴파일은 한 언어로 작성된 소스 코드를 다른 언어로 바꾸는것(C-> 어셈블리어)
트랜스파일러는 한언어로 작성된 소스코드를 비슷한 수준의 추상화를 가진 다른 언어로 변환 (C++>C, ES6->ES5)

ES6 에서 추가된 스펙에 대해 아는대로 다 말해달라(let, const, rest parameter, class, arrow function...)
let, const, 화살표함수, 클래스, 프로미스, 스프레드 연산자 등

var 와 let, const의 차이점은 무엇인가 (function scope와 block scope의 개념에서)
var은 함수 레벨 스코프를 지원, let,const는 블록레벨 스코프를 지원한다.
하여, 다음과 같이 블록레벨에 foo를 456으로 재선언하는 경우 foo를 456으로 인식
하지만, let이나 const는 전역 변수를 읽는다 블록 안에 있는것을 읽지 않고
```
var foo = 123; // 전역 변수

console.log(foo); // 123

{
  var foo = 456; // 전역 변수
}

console.log(foo); // 456
```

Class 는 무엇이고, Prototype, fucntion의 ES5 스펙만으로 Class를 구현할수 있는가
구현가능하다. 자바스크립트에는 프로토타입이라는 것이 존재하여 클래스처럼 구현할 수 있다.클래스는 자바스크립트의 프로토타입 기반 패턴의 문법적 설탕이다.

[REACT]
리액트의 상태관리에 대해 알고 있는가? Redux를 사용해봤다면, 그것에 대한 설명
리액트에서 전역의 상태를 관리하기 위해서 사용하는 방법이다. 컴포넌트간의 상태들을 한군데다가 모아놓고 공유해서 사용하는 방식.
리액트의 상태관리는 Context API, Redux, MobX 등의 상태관리가 있으며, Context API보다 Redux를 사용하는 이유는 대규모 개발에서 유지보수성이나 작업효율을 높이기에는 Redux를 사용하는것이 좋기 때문에 많은 사람들이 Redux를 사용한다. 리액트 16.3이후 버전에서는 그래도 Context API가 개선되어 사용하기 좋아졌다.
Redux는 사실 다른 곳에서도 많이 쓰이는 기술이었지만, react-redux라는 것이 있어서 react에서 사용하기 좋아졌다.
react-redux에서는 3가지 규칙을 지켜야하는데
1) 단일 스토어야 할것
2) 읽기전용상태여야 한다. 즉 기존의 객체는 건드리지 않고 새로운 객체를 생성해서 사용하여야한다.
3) 리듀서는 순수한 함수여야한다. 즉, 파라미터 외의 값에는 의존하지 않아야한다.

만드는 순서는 액션 타입을 정하고, 액션 생성 함수를 만들고, 이 액션들을 사용하는 리듀서 함수(초기상태 포함)를 만들고, index.js에서 스토어를 만들어 provider로 스토어를 props로 전달해준다.
프레젠테이셔널 컴포넌트와 컨테이너 컴포넌트를 분리하여, 컨테이너 컴포넌트에서 connect 함수를 사용해서 mapStateToProps(스토어 안의 상태를 컴포넌트의 props로 넘겨주기 위해 설정하는 함수),
mapDispatchToProps(액션 생성 함수를 컴포넌트의 Props로 넘겨주기 위해 사용하는 함수) 이 2가지를 다음과 같이 사용
connect(mapStateToProps, mapDispatchToProps)(타깃컴포넌트)

자세한 리덕스 내용은 아래를 참고
https://sunnykim91.tistory.com/entry/2020-01-30-%EB%A6%AC%EC%95%A1%ED%8A%B8-%EC%88%98%EC%97%85-Redux-%EC%97%85%EB%8D%B0%EC%9D%B4%ED%8A%B8%EC%A4%91

Context-API에 대해서 설명하세요.
리액트에서 전역상태를 관리하기 위한 기능

클래스형과 함수형의 차이는 무엇인가?
클래스형은 라이프사이클 메소드를 사용하고, 함수형에서는 useEffect등 Hook을 사용한다.

클래스형은 render함수가 반드시 필요, 함수형이 선언하기 더 간편하다.

라이프사이클 메소드에 대해서 설명하세요.
클래스형에 라이프사이클 메소드에는 크게
mount, update, unmount 3가지 과정으로 나뉜다.
자세하게는
constructor -> getDerivedStateFromProps -> render -> componentDidMount -> getDerivedStateFromProps -> shouldComponentUpdate -> render -> getSnapshotBeforeUpdate
-> componentDidupdate
mount에서 컴포넌트가 만들어질때 componetDidMount에서 비동기처리 같은것을 주로하고, shouldComponentUpdate에서 업데이트 직전에 랜더링시(상태가변경)에 조건으로 재랜더링을 하냐마냐 결정을 할 수 있고, componentDidUpdate 업데이트 직후에 호출되는 메소드이고 unmount에서 컴포넌트가 소멸된 시점에 타이머나 비동기 API를제거 하는 곳이다.

[기타 질문]
타입스크립트를 사용해본 경험이 있는가, 타입스크립트에 대한 본인의 생각과 도입시의 장점을 말해달라
Typescript는 동적타입언어인 Javascript의 약점을 보완하기 위해서 타입을 지정해주는 것이다. 타입이 필요한 이유는 결론은 메모리를 절약하기 위해서이다. 메모리에 저장된 것을 읽어들일때, 값을 메모리에 저장할때, 값이저장되어있는 것을 참조할때의 크기들을 알아야 하기 때문이다.
또한, 에러를 잡기가 쉬워지고, 다른 동료와 협업 할때 코드의 예측도 가능해지고, 코드에디터의 도움을 더 받을 수 있다. 리액트의 경우 (브라우저는 javascript밖에 모르기때문에) tsx파일을 javascript로 변환하는 트랜스파일링이 필요하다. 이때 변환하는 과정에서 에러를 잡을 수 가있다. 런타임에 오류를 잡는 것보다 훨~~좋다!
또한, Babel을 안써도 된다.

Angular와 React의 차이점은 무엇이라고 생각?
우선 Angular는 프레임워크이고, React는 라이브러리이다. Angular는 양방향 바인딩개념으로 Model과 view가 연결되어있어 데이터 값이 한쪽에서 변화하면 다른쪽에서도 바로 업데이트가 진행된다. 서비스라는 개념은 컴포넌트간의 의존성관리를 용이하게 해준다. Directive를 이용하여 커스텀 HTML태그를 작성할 수 있다. React는 Virtual DOM을 가지고 있다. 가상DOM이 있기때문에 상태를 비교하여 부분적으로 랜더링 할 수 있어 속도가 빠르다. 오직 UI 컴포넌트를 만들기 위한 라이브러리이다. Angular는 HTML스크립팅이 templete 기반, React는 JSX 이용 / Angular는 기본이 Typesciprt

라이브러리와 프레임워크에 대해서 설명
라이브러리와 프레임워크의 차이는 자유도의 차이 인것 같다. 프레임워크는 짜여진 패턴이나 틀 기반에서 내가 코딩을 하는 것이고, 라이브러리는 내가 가져다 사용해서 자유롭게 사용하는 방식이다.

두 명의 프론트엔드 개발자가 있다. git을 관리하는 방식?
git repository를 하나파서 다른 동료가 fork를 해서 사용하는 방식. PM과 팀원의 구조를 가질 수도있고 동시에 Pull request를 가능하게끔 권한을 줄 수 도 있다. 각자의 팀장의 레포가 origin이라하고, 팀원의 포크딴 레포를 rmorigin이라고 한다면 각자의 origin에서 develop브랜치에서 작업을 한뒤 최종 작업이 완료되면 팀장의 origin의 마스터로 push한다.

메소드 체이닝이란 무엇이며, 이것의 장단점은 무엇인가?
메소드 체이닝의 장점은 코드가 짧아진다는 장점이 있고, 단점은 에러가 났을때 어느 부분의 메소드에서 오류가 났는지 확인이 어렵다.

메모라이제이션이란?
불필요한 연산이나 계산을 하지 않고 기억을 해놓고 그 기억해놓은 것을 활용하는 방법

RESTful API 가 무엇인가, 아는대로 다 말해달라
REST API는 URI로 접근가능하고 내용이 JSON,XML 등으로 표현된 자원에 대한 행위를 HTTP Method로 정의한다.
RESTful하다라는 것은 REST API의 설계의도를 명확하게 지켜주는 것이다. 슬래시를 통해 계층관계를 표시한다던가 숫자는 id를 나타낸다든가 동사보단 명사를 위주로 쓴다든가 하는.

CORS(Cross-Origin Resource Sharing)는 무엇인가 왜 이러한 방법이 정의 되었으며, 본인이 코드를 작성하면서 CORS와 관련하여서 경험하였던 이슈는 무엇인가
CORS란 도메인 또는 포트가 다른 서버의 자원을 요청하면 발생하는 문제. 경험 없음.
웹 프론트 측에서 request header에 CORS 관련 옵션을 넣어주고, 서버에서는 해당 프론트 요청을 허용하면 됨

Eslint가 무엇인가요?
Eslint는 소스코드를 스캔하여 문법적오류나 잠재적 오류까지 찾아내고 오류의 이유를 볼 수 있게 해주는 도구

Prettier가 무엇인가요?
prettier는 정해진 규칙대로 코드를 이쁘게 할 수 있는 도구, 들여쓰기나 따옴표 등

Webpack이란?
webpack은 모듈 번들러로 파일 확장자에 맞는 로더에게 위임해서 하나로 묶어준다 최종 배포용 파일을 만들어줌
<script>태그가 여러개 있을 때 순서보장도 중요하기 때문에 이런것도 Webpack에서 해줌.

패키지매니저로 어떤거 사용?
npm

npm과 yarn은 어떤게 다른가?
과거 깃허브에서 받은 소스코드를 npm i로 패키지를 설치 했을 때 이전과 버전정보가 다른 환경문제가 생길 수 있었는데 이것을 처리하기 위해서 yarn.lock 파일에서 처리할 수 있었다. 하지만, 현재는 npm의 package-lock.json에서 이 처리를 동일하게 사용할 수 있다.

적응형과 반응형의 차이를 아는가?
반응형 웹은 하나의 템플릿을 사용해 모든 기기에 대응하는데 반해, 적응형 웹은 선별된 기기 유형에 따라 별도의 독립적인 템플릿이 요구

package.json파일의 역할을?
package.json에서 dependencies와 devDependencies의 차이는?
프로세스와 스레드의 차이
프로세스는 운영체제로부터 자원을 할당받는 작업의 단위이고 스레드는 프로세스가 할당받은 자원을 이용하는 실행 단위이다. 멀티프로세싱보다 멀티스레딩을 하는 이유는 운영체제로부터 자원을 할당 하는 프로세싱 작업을 많이 하는 것 보다 스레딩을 여러개 하는것이 훨씬 더 시스템 자원을 효율적으로 관리할 수 있다.

CSR과 SSR의 차이?
CSR의 과정 :
- 서버가 브라우저에게 응답을 보냄 -> 브라우저는 JS를 다운 받음 -> 브라우저는 리액트를 실행 -> 페이지가 보여지고 상호작용 함
SSR의 과정 :
- 서버가 브라우저에게 HTML 응답 랜더링하기 위한 준비가 되었다고 보냄 -> 브라우저가 페이지랜더링, 페이지가 보여지고 브라우저는 JS 다운받음 -> 브라우저 리액트 실행 -> 페이지 상호작용 가능

CSR은 마지막 단계 전까지 화면에 보여지지가 않고 로딩중 / SSR은 미리 페이지가 보여진다.
즉, CSR은 초기로딩속도가 느리긴하지만, 화면전환에 있어서 클라이언트에서 이루어져서 빠른 전환이 가능
SSR은 초기로딩속도가 빨라서 사용자가 느끼기엔 좋지만, 동작은 하지않음. 그리고 화면전환에 있어서 서버에 요청해야하므로 서버에 부담을 줄 수 있음.
어떤게 더 좋다보다 서비스마다 사용자의 요구마다 다름.

이벤트 위임이란?
이벤트 위임이란 자식 엘리먼트의 이벤트를 부모엘리먼트에서 감지할 수 있으니 이벤트를 하나하나 등록하는 것이 아니라 부모에게 이벤트를 위임 하는 방법

DOM을 건드리는 방식과 아닌 방식들의 차이

직접 DOM을 건드리는 경우 DOM의 구조를 파악하고 있어야하며, 클래스명이다 태그명이 바뀌는 경우 다시 DOM을 변경해야한다.
Angular의 경우 view와 model을 연결시키는 바인딩작업이 있고 변화감지를 통해서 상태를 보고 있다가 업데이트되는식이다.
React의 경우 가상 DOM이 있고, 가상 DOM이 실제 DOM과 비교하여 state가 변화되었는지 감지 한다.

반응형 프로그래밍?
반응형 프로그래밍이란 데이터 스트림이라는 하나의 일관된 형식으로 만들고, 이 데이터 스트림을 구독하여 데이터 스트림의 상태 변화에 반응하는 방식으로 동작하는 애플리케이션을 만드는 것
예를 들어, Tv랑 Tv방송국이 있다고 가정했을때, Tv방송국이 일정한 시간 단위로 영상에 대한 프레임을 계속해서 방출(emit)하고 TV는 방송국을 관찰하고 있다가 새로운 영상을 방출하면 이를 획득하는 방식이다. 여기서 방송국의 역할이 옵저버블, Tv가 옵저버, 영상프레임이 Notification이다.
Call by value & call by ref

call by value 는 인자로 값이 넘어올때 복사된 값이 넘어오기 때문에 중간에 어떤 연산을 해도 변하지 않는다.

```
var a = 1;
var fun = funcion(b) {
        b=b+1;
}
fun(a)
console.log(a) // 1
```

자바스크립트는 기본적으로 원시값을 넘겨주면 call by value 로 작동
함수 내에서 값을 변경하면 함수에 전달된 데이터만 변경될 뿐 함수 전달된 원본 복사본에는 아무런 영향을 미치지 않는다.

call by reference는 인자로 레퍼런스가 넘어올때 가리키는 값을 복사하는 것이 아니라 참조값을 넘기는 것

참조형 데이터는 그 값의 주소를 말 그대로 참조 할 값의 복사본이나 값 자체가 할당되지 않는다. 참조에 의해 할당된 새 변수는 원본 변수가 가르키는 값과 동일한 값을 가르킨다. 원본 변수와 할당된 변수는 모두 동등하며, 값을 조작하는데 사용될 수 있다. 그래서 할당된 변수(참조)가 변경되면 원본 변수에서도 동일하게 변경된다.

```
var a = {};
var fun = funcion(b) {
        b.a=1;
}
```

fun(a)
console.log(a.a) // 1

null vs undefined ?
기본적으로 둘다 값이 없음을 나타낸다.
undefiend는 데이터 타입이자 값을 나타냄. 정의되지 않은 것
null은 명시적으로 값이 비어있음을 나타내는데 사용
undefined는 변수를 선언만 한더라도 할당되지만, null은 변수를 선언한 후에 null로 값을 바꾼다.

inline vs inline block ?
inline은 text크기만큼만 공간을 점유하고 줄바꿈을 하지않음.
inline-block은 inline속성과 block속성의 특징을 둘다 가지고 있다. inline속성과 다르게 width,height 적용 가능하고 line-height를 커스텀하게 적용할 수 있다.

vue와 React의 차이?
공통점으로 컴포넌트 기반이다. Virtual DOM 방식이다. 가볍고 빠르다.
vue는 단일 파일 컴포넌트이다. html, css, javascript코드가 하나의 파일에 모두 정의하는 방식이 기본이지만, 컴포넌트화 해서 사용할 수 있음!!
HTML 기반 템플릿 구문을 가진다. -> 배우기 쉬움

vue에서의 라이프사이클?
creation , mounting, updating, destruction으로 나눌 수 있다.
creation
  - 컴포넌트가 돔에 추가되기 전이다.
  - created : data와 events가 활성화 되어 접근 가능, 템플릿과 가상돔은 마운트 및 렌더링되지 않은 상태
  - 주로 초기에 세팅되어야할 데이터 fetch작업은 created단계 사용
mounting
  - mounted : 초기 렌더링 직전에 컴포넌트에 직접 접근 할 수 있다.
  - 컴포넌트, 템플릿, 렌더링된 돔에 접근 가능
updating
  - updated : 데이터가 변하여 재 렌더링이 일어난 후에 실행
destruction
  - destroyed : 뷰 인스턴스가 제거 된 후에 호출

가상돔 (virtual DOM)
Virtual DOM은 실제 DOM 변화를 최소화 시켜주는 역할을 합니다.

먼저 브라우저는 HTML 파일을 스크린에 보여주기 위해 DOM 노드 트리 생성, 렌더트리 생성, 레이아웃, 페인팅 과정을 거칩니다. DOM 노드는 HTML의 각 엘리먼트와 연관되어 있기 때문에 HTML 파일에 20개의 변화가 생기면 DOM 노드가 변경되고 그 이후의 과정역시 20회 다시 이루어 집니다. 작은 변화에도 매우 복잡한 과정들이 다시 실행되기 때문에 DOM 변화가 잦을 경우 성능이 저하됩니다.

Virtual DOM은 뷰에 변화가 있다면, 그 변화가 실제 DOM에 적용되기 전에 Virtual DOM에 적용시키고 최종 결과만 실제 DOM에 전달합니다. 따라서 20개의 변화가 있다면 Virtual DOM은 변화된 부분만 가려내어 실제 DOM에 전달하고 실제 DOM은 그 변화를 1회로 인식하여 단 한번의 렌더링 과정만 거치게 됩니다.