네트워크/브라우저/API

브라우저 렌더링 과정을 설명하세요

- HTML, CSS를 파싱하여 각각 DOM, CSSOM 트리를 만듭니다.
- Javascript 실행
- DOM트리와 CSSOM트리를 조합하여 렌더 트리(Render Tree)를 만듭니다. 그 다음
- 렌더 트리를 화면에 어떻게 배치해야 하는지 노드의 정확한 위치와 크기를 계산합니다. (Layout/Reflow 단계)
- 계산한 위치/크기를 기반으로 화면에 그림 (Paint 단계)

주소창에 URL을 입력하면 어떤 일이 생길까요?

- 사용자가 브라우저 주소창에 URL을 입력하면
- 브라우저가 URL의 IP주소를 찾기 위해 캐시에서 DNS 기록을 확인합니다. DNS 기록을 찾기 위해 브라우저는 브라우저 캐시, OS 캐시, 라우터 캐시, ISP 캐시를 차례대로 확인합니다.
- 만약 요청한 URL이 캐시에 없다면 ISP의 DNS 서버가 URL을 호스팅하는 IP 주소를 찾습니다.
- 브라우저가 해당 웹 서버와 TCP 연결을 시작합니다.
- TCP 연결이 설정되면 브라우저가 웹 서버에 HTTP 요청을 처리하고 응답을 보냅니다.
- 응답을 보내면 브라우저가 HTML 컨텐츠를 보여주게 됩니다.

CORS란? 대응 방법은?

- 서버와 브라우저간에 서로 통신을 할 때 서버는 브라우저가 자원에 접근하는 것에 대해 권한을 부여하고 알려줘야합니다. 그렇지 않을 시 브라우저 측에선 해당 자원에 접근하는 것이 안전하지 않다고 판단하여 통신을 차단해버리는데 이 때 발생하는 것이 CORS 오류입니다. 이에 대응하기 위해 서버 측에서 브라우저에게 접근에 대한 권한이 있음을 알리는 방법과 응답 헤더의 Access-Control-Allow-Origin에 접근을 허용할 출저를 반환하는 방법이 있습니다.

RESTful API란?

- RESTful API는 HTTP URL을 통해 사용자가 받고자하는 리소스를 명시하고 POST, GET, PUT, DELETE와 같은 HTTP Method를 통해 서버에 요청을 보내면 서버는 리소스를 확인하여 응답하는 방식인 REST 규격을 준수하는 API를 의미합니다.

GET과 POST의 차이, PUT과 PATCH의 차이점은?

- GET은 서버로부터 데이터를 받아오기 위한 메서드이고, POST는 서버에 데이터를 전송하기위한 메서드입니다. PUT은 리소스의 모든 것을 업데이트하지만, PATCH는 리소스의 일부를 업데이트합니다.

HTTP와 HTTPS 통신의 차이

- HTTP는 서버와 클라이언트 간 보안 절차가 없고 HTTPS는 암호화, 복호화와 같은 보안 절차를 거친 후 요청과 응답을 한 후 종료합니다.

localStorage, sessionStorage, cookie의 차이

- localStorage는 같은 도메인에 대해 같은 정보를 저장하며, 창을 닫거나 탭을 닫아도 정보가 지워지지 않습니다.
- sessionStorage는 같은 도메인이더라도 탭, 창별로 서로 다른 정보를 저장하며 세션이 만료되면 데이터도 사라집니다.

- Cookie는 용량이 상당히 작은 텍스트파일로 만료기간을 가져 일정 시간이 지나면 만료되어 데이터가 삭제됩니다.
- **localStorage:** 자동로그인
- **sessionStorage:** 임시 저장 용도
- **cookie:** 팝업창 안보게 하는 용도

SEO란?

- 검색 엔진 최적화로 검색 엔진이 컨텐츠를 잘 이해하고 사용자에게 제공할 수 있도록 도와주어 사이트를 개선하는 프로세스를 말합니다.
- 사이트의 디렉토리 구조가 단순하고 의미가 명확한지 독창적인 메타 데이터와 URL 구조, 사이트 접근과 크롤링에 대한 처리를 담당하는 robot.txt의 구성, 페이지 순위, 이미지 최적화 등을 고려해볼 수 있습니다.

CSR과 SSR의 차이점은?

- SSR은 사용자가 웹 페이지에 접근할 때 서버에 각각 페이지에 대해 요청하고 서버에서 HTML, JS 파일등을 모두 다운로드하여 화면에 렌더링 하는 방식입니다.
- CSR은 클라이언트 측에서 HTML을 반환한 후에 JS가 동작하면서 데이터만 주고받아 클라이언트에서 렌더링을 합니다.

fetch와 axios 차이

- axios는 promise API를 활용하는 http 비동기 통신 라이브러리입니다.
- fetch를 이용하면 데이터를 가져오는 과정에서 .json() 메소드를 한번 더 이용해야되서 axios가 좀 더 편합니다.

SPA, MPA

- SPA는 Single Page Application으로 단 하나의 HTML 문서로만 돌아가는 웹페이지로 사용자 요청에 따라 내용이 바뀝니다. 예로 리액트
- MPA는 Multi Page Application으로 여러 페이지로 구성된 웹 어플리케이션으로 주로 SSR을 사용합니다. 페이지의 링크를 클릭할 때마다 다른 페이지를 로드합니다.

프로세스와 스레드에 대해 설명

객체 지향 프로그래밍이란?

JavaScript

var, let, const 차이점

- var는 재선언과 재할당 모두 가능합니다. 호이스팅 현상으로 변수를 선언하기 이전에 참조가 가능합니다.
- let은 변수 재선언이 불가능하지만 재할당은 가능하고
- const는 재선언과 재할당 모두 불가능 합니다.
- let과 const 둘 다 블록 레벨 스코프를 가집니다.

호이스팅이란? (선언 - 초기화 - 할당 / TDZ)

- 코드가 실행되면서 선언부를 코드의 최상단으로 끌어올리는 것처럼 보이게 합니다.

스코프란? (Lexical Scope)

- 변수, 함수를 선언했을 때 해당 변수 또는 함수가 유효한 범위를 의미합니다.
- 전역스코프, 함수스코프, 블록스코프 세가지가 있습니다. 전역은 모든 범위에서 이용가능하고, 함수는 함수안에서, 블록은 특정 블록 안에서만 사용 가능합니다.

콜백함수란?

- 콜백함수는 함수의 매개변수로 다른 함수를 전달하고 어떤 이벤트가 발생한 후 매개변수로 전달한 함수가 다시 호출되는 것을 의미합니다.
- 보통 비동기 처리 시 콜백함수를 사용하며 콜백지옥에 빠지면 가독성이 떨어지기 때문에 promise나 async/await를 사용합니다.

동기 비동기란?

- 동기는 요청을 보낸 후 해당 응답을 받을때까지 기다렸다가 다음 동작을 실행
- 비동기는 요청을 보낸 후 응답에 관계 없이 순차적으로 다음 코드를 먼저 실행하는 것

Promis란?

- promise 객체를 생성하여 비동기 작업이 완료된 후 성공, 실패 결과값을 받아 이후 쉽게 컨트롤 할 수 있습니다.
- 인자로 들어온 함수는 resolve 비동기처리 성공과 reject 비동기처리 실패 2개의 함수를 인자로 받아서 성공시 then 메소드, reject 시 catch 메소드를 실행합니다.

async await란?

- 함수 앞에 async를 선언하면 promise object가 생성됩니다.
- async를 선언한 함수안에서 await를 사용할 수 있습니다.
- await 함수의 작업이 끝나고 결과값을 반환할 때 까지 대기하고 결과값이 리턴되면 다음 작업으로 넘어갑니다.
- await는 실패하면 에러가 나고 코드가 멈추기때문에 이 때 try catch를 사용합니다.

얕은 복사와 깊은 복사의 차이

- 얕은 복사는 객체의 참조값(주소 값)을 복사하고, 깊은 복사는 객체의 실제 값을 복사합니다.
- 얕은 복사를 하면 원본이 변경되지만 깊은 복사를하면 원본이 변경되지않습니다.

이벤트 캡쳐링, 이벤트 버블링이란? (예시)

- 이벤트 흐름은 캡처링 단계, 타깃 단계, 버블링 단계로 나누어집니다.
- 이벤트 캡처링은 최상위 노드에서 시작해 해당 요소까지 이벤트가 전파되는 과정을 의미합니다.
- 캡처링 단계 이후 버블링이 발생합니다.
- 이벤트 버블링은 가장 최상단의 조상 요소를 만날 때가지 이어서 부모 요소에게 이벤트가 전파되는 과정을 의미합니다.

This란? (call, apply, bind)

- this는 자신이 속한 객체나 자신이 생성한 인스턴스를 가리키는 자기 참조변수 입니다. 문맥에 따라서 다양한 값을 가지고 크게 4가지로 나뉩니다.
- 첫번째로 일반 함수로 호출된 경우 this는 전역 객체 입니다. 브라우저에서 window 객체가 됩니다.
- 두번째로 메소드로 호출되었다면 this는 메서드를 호출한 객체입니다.
- 세번째로 함수가 호출될 때나 생성될 때 apply, call 또는 bind가 사용되었다면 이 함수 안에서 this는 첫번째 인자값이 됩니다.
- 마지막으로 함수가 호출될 때 new 키워드를 사용했다면 이 함수 안의 this는 새로운 객체이기때문에 빈 객체일 것입니다.

실행 컨텍스트란?

- 실행할 코드에 제공할 환경 정보들을 모아놓은 객체입니다.
- 자바스크립트는 이러한 환경 정보들을 모은 실행 컨텍스트를 콜 스택에 쌓아올린 후 실행하여 코드의 환경과 순서를 보장합니다.

자바스크립트는 싱글스레드 기반 언어인데 비동기 프로그래밍은 어떻게 가능할까요?

- 자바스크립트 엔진은 하나의 Memory Heap과 하나의 Call Stack을 가지고 있습니다. 한 번에 하나의 함수만 동기적으로 실행 가능합니다. 그런데도 자바스크립트가 비동기적으로 작동할 수 있는 이유는 자바스크립트 엔진 밖에서도 자바스크립트 실행에 관여하는 요소 Web API, Task Queue, Event Loop가 존재하기 때문입니다. Call Stack에 하나의 테스크가 추가되고 테스크 요청 시간이 지나면 Task Queue에 인자로 받은 콜백함수를 전달합니다. event loop는 call stack이 비어져 있는 것을 확인하고 task queue에 있던 콜백함수를 call stack으로 옮기고 실행합니다.

클로저란?

- 클로저는 반환된 내부 함수가 자신이 선언되었을 때의 환경(렉시컬 환경)인 스코프를 기억하여 그 밖에서 호출되어도 해당 환경에 접근할 수 있는 함수를 의미합니다.

화살표 함수와 일반 함수의 차이점

- 화살표 함수는 함수를 간단하게 표현할 수 있는 ES6 문법입니다. 일반 함수는 this가 동적으로 바인딩되는 반면, 화살표 함수는 바로 상위 스코프의 this와 같습니다.

이벤트루프란? 동작 방식 설명

- 콜 스택과 테스크 큐를 주시하고 콜 스택이 비어있다면 테스크 큐에 있던 작업을 콜 스택으로 보내주는 역할을 합니다.

ES6 문법에 대해서 설명

const, let, arrow function, 탬플릿 리터럴 - 백틱사용하여 문자열 내에서 변수 사용 가능, 구조분해할당, import export, promise, spread 연산자가 추가되었습니다.

프로토타입에 대해 설명

불변성을 유지하려면 어떻게 해야하나요?

TypeScript

타입스크립트를 사용해 본 경험이 있는지? 어땠는지?

- 정적 타입으로 컴파일 단계에서 오류를 잡아낼 수 있는 장점이 있고 ES6의 새로운 기능을 바벨을 사용하지 않아도 기존의 자바스크립트 엔진에서 실행할 수 있습니다. 명시적인 정적 타입 지정은 코드의 가독성을 높이고 디버깅을 쉽게 할 수 있다는 장점이 있습니다.

타입스크립트의 특징

- TypeScript도 JavaScript에 속합니다. 하지만 TypeScript는 클래스와 인터페이스 그리고 정적 타이핑을 제공하는 Java와 같은 객체 지향 프로그래밍 언어이기도 합니다.
- JavaScript는 기본적으로 변수에 값이 할당될 때 변수의 자료형이 결정되는 동적 타이핑 언어입니다. 반면 TypeScript는 선언할 때 타입을 지정해줘야 하는 정적 타이핑 언어입니다.

인터페이스와 타입 차이

- interface는 같은 이름으로 중복 선엄을 함으로써 확장이 가능하지만 type의 경우 에러가 발생합니다.

제네릭에 대해 설명해주세요

- 제네릭은 타입을 파라미터처럼 사용하는 것을 의미합니다.
- <>안에 대표적으로 T를 써서 사용합니다.

undefined, null 차이

React

Virtual DOM 작동 원리

- 데이터를 업데이트하면 전체 UI를 Virtual DOM에 리렌더링 합니다. 이전. Virtual DOM에 있던 내용과 현재 Virtual DOM의 내용을 비교하여 바뀐 부분만 실제 DOM에 적용합니다.

Virtual DOM 이란?

- 웹 브라우저에서 DOM에 변화가 일어나면 브라우저 렌더링 과정을 거치게 되고 자주 조작하면 성능에 영향을 끼치기 때문에 속도가 느려집니다. Virtual DOM을 사용해 DOM 처리 횟수를 최소화하고 효율적으로 진행하게 해줍니다.

왜 React를 사용하나요?

- 컴포넌트 단위로 작성하게 되면 재사용이 가능해서 생산성과 유지보수가 용이합니다.
- Virtual DOM이 존재해서 DOM과 비교해 달라진 부분만 찾아 바꾸기 때문에 웹 반응성이 좋습니다.
- 마지막으로 커뮤니티입니다. 개발을 하다가 막히거나 오류가 있을 때 사용자가 많아서 자료가 많기때문입니다.

클래스 컴포넌트와 함수 컴포넌트의 차이

- 클래스 컴포넌트는 state의 사용이 가능하여 상태 저장이 가능하고 리액트 라이프 사이클 메서드를 사용합니다. 함수가 아닌 클래스이기 때문에 return문이 없고 render() 함수로 JSX를 반환합니다.
- 함수 컴포넌트는 클래스보다 선언하기 좀 더 편하고 함수는 한번 실행되고 나면 메모리 상에서 사라지기때문에 메모리 자원을 덜 사용하는 것이 장점입니다. 함수형 컴포넌트에서는 hook을 사용할 수 있고 return문을 사용합니다.

메모이제이션 설명

- 메모이제이션이란 계산된 값을 자료구조에 저장하고 이 후 같은 계산을 반복하지 않고 자료구조에서 꺼내 재사용하는 것을 말합니다.

useMemo() 란?

- Memoization된 값을 반환합니다.
- deps 배열의 요소가 변경되지 않는 이상 함수의 반환값을 새로 계산하지 않기 위하여 사용합니다.

useCallback() 란?

- Memoization된 함수를 반환하는 대표적인 hook 입니다.

- deps 배열의 요소가 변경되지 않는 이상 함수를 새로 생성하지 않기 위하여 사용합니다. 자식 컴포넌트에 함수를 넘겨줄 때 사용합니다.

useMemo/React.memo 차이

- React.memo는 HOC이고, useMemo는 훅입니다.
- React.memo는 컴포넌트 자체를 메모이제이션하는 용도로 사용하고 useMemo는 복잡한 계산의 결과값을 메모이제이션 하는 용도로 주로 사용합니다.

리액트 렌더링 성능 최적화 방법은 무엇이 있는지?

- state와 props의 변경을 최소화합니다.
- 컴포넌트를 map을 통해 매핑할 때 key 값으로 index를 사용하지 않습니다.
- useMemo, React.memo, useCallback 사용
- 첫번째로 useMemo()가 있습니다. useMemo()는 memoization된 값을 반환하기 때문에 연산의 결과값을 메모리에 저장해두고 동일한 입력이 들어오면 값을 재사용합니다. 이로 인해 함수 호출 시간도 줄이고 하위 컴포넌트에서 함수를 props하여 사용하고 있다면 재렌더링도 방지할 수 있습니다.
- 두번째로 React.memo가 있습니다. React.memo는 컴포넌트의 props가 바뀌지 않도록 하여 성능 최적화할 수 있습니다.
- 세번째로 useCallback()이 있습니다. useCallback()은 memoization된 함수를 반환합니다. 함수를 props로 넘겨줄 경우 함수는 객체이고, 새로 생성된 함수는 다른 참조 값을 가져 props가 변한 것이라고 인지합니다. 이런 경우 useCallback()으로 함수를 감싸면 의존성 배열 안에 넣어준 값이 바뀔 경우를 제외하고 항상 동일한 객체를 넘겨줌으로써 불필요한 재렌더링을 방지할 수 있습니다.

React, Vue, Angular의 차이는 무엇인가요?

- 3개의 공통점은 SPA 기반 프론트엔드 프레임워크 혹은 라이브러리입니다. react 는 단방향 바인딩으로 부모 컴포넌트에서 props가 자식 컴포넌트로 전달되고, vue, angular는 양방향 바인딩이 가능하다는 차이가 있습니다.

React의 라이프사이클에 대해 설명해주세요

- 각각의 컴포넌트에는 생명주기가 존재합니다. 컴포넌트의 생명은 보통 페이지에서 렌더링되기전부터 페이지에서 사라질 때까지입니다. 라이프 사이클은 크게 보면 컴포넌트가 처음 실행될 때인 mount, 데이터에 변화가 있을 때인 update, 컴포넌트가 제거될 때인 unmount 이렇게 3개로 나눌 수 있습니다.

웹팩과 바벨에 대해서 설명해주세요

- 웹팩은 여러 개의 파일을 하나로 합쳐주는 모듈 번들러입니다. 웹팩은 모듈 지원, 파일 분할, css loader, jsx 변환 작업을 해줍니다. 그리고 여러 개로 나눠진 자바스크립트 파일을 html이 실행할 수 있는 하나의 자바스크립트 파일로 합쳐줍니다.
- 바벨은 대표적인 트랜스파일러인데 모든 브라우저가 ES6 문법을 제공하지 않기 때문에 ES5 문법으로 변환시켜줍니다.

React에서 렌더링이 일어나는 경우

- 컴포넌트의 state가 변경되었을 때
- 컴포넌트가 상속받은 props가 변경되었을 때
- 부모 컴포넌트가 리렌더링 된 경우 자식 컴포넌트는 모두 리렌더링

useEffect/useLayoutEffect 차이

- useEffect는 비동기적으로 동작하고 useLayoutEffect는 동기적으로 동작합니다.

- 리액트에서 useEffect는 렌더링이 끝나고 테스크를 수행하고, useLayoutEffect는 렌더링 전에 테스크를 수행합니다.
- 성능 모니터링이나 애니메이션 구현 등 즉시 반응이 필요한 경우에 useLayoutEffect를 사용하고 네트워크 요청, DOM 접근, 비동기 작업을 하는 경우에 useEffect를 사용하는 것이 좋습니다.

Redux 특징

- Redux는 자바스크립트 상태관리 라이브러리입니다.
- 모든 상태의 업데이트를 액션으로 정의하고 액션 정보에 기반하여 리듀서에서 상태를 업데이트하기 때문에 효율적으로 유지보수 할 수 있습니다.

Flux 패턴에 대해 설명

- Flux 패턴은 사용자 입력을 기반으로 Action을 만들고, Action을 Dispatcher에 전달하여, Store의 데이터를 변경한 뒤, View에 반영하는 단방향의 흐름으로 애플리케이션을 만드는 아키텍처입니다.

Context API와 Redux를 비교

- Context API는 prop drilling을 해결하기 위해 나온 도구이고 Redux는 상태를 좀 더 쉽게 관리하고자 나온 도구입니다.

React hooks 각각 설명

- useState, useEffect, useLayoutEffect, useRef, useMemo, useCallback, ...

props와 state 차이

- state는 현재 컴포넌트의 생명주기 동안에 변경될 수 있는 정보를 담고 있는 상태이고 props는 부모 컴포넌트에 자식 컴포넌트로 전달되는 데이터입니다.
- props는 함수 매개변수처럼 컴포넌트에 전달되는 반면 state는 함수 내에 선언된 변수처럼 컴포넌트 안에서 관리됩니다.

Prop Drilling에 대해 설명해주세요

- 상위 컴포넌트에서 하위 컴포넌트로 props를 내려주는데 이 깊이가 깊어질 때를 prop drilling이라고 합니다.

Prop Drilling를 어떻게 해결할 수 있나요?

- redux나 Context API를 사용해서 데이터를 전역적으로 관리하는 방법이 있습니다.

데이터를 자식에서 부모로도 전달할 수 있나요?

- 부모가 자식으로 setter 함수를 props 로 보내줍니다.

React의 불변성에 대해 설명

- state는 불변성을 유지해야해서 state를 직접 변경하지 않고 setState를 사용합니다.

key props를 사용하는 이유는?

- key는 Virtual DOM을 비교할 때 어떤 변화가 일어났는지 빠르게 알아낼 수 있도록 사용합니다.
- key는 엘리먼트에 안정적인 고유성을 부여하기 위해 배열 내부의 엘리먼트에 지정해야 합니다.

key props를 쓸 때 index를 쓰면 안되는 이유

React.Fragment에 대해 설명

- 리액트 컴포넌트는 1개의 엘리먼트를 리턴해야 하며 여러개의 엘리먼트를 리턴할 경우 에러가 발생합니다.
- Fragment로 엘리먼트들을 감싸면 레이아웃이나 스타일에 영향을 주지 않습니다.

React 18 버전 업데이트 내용

- Automatic batching: 상태 업데이트를 하나로 통합해서 배치처리한 뒤 리렌더링을 진행합니다.

React에서 JSX 문법이 어떻게 사용되나요?