**면접**

**Common Questions (컬쳐핏)**

전체 공통

**- 이직 사유에 대해서 말씀해주세요.**

**- 코드 리뷰 경험이 있는지?**

> 납득하기 어려운 코드가 있다면 어떤식으로 코드 리뷰를 진행할것인지?

**JavaScript**

- **자바스크립트의 데이터 타입의 프리미티브 타입(원시타입)과 레퍼런스(참조타입)의 특징을 비교해주세요.**

> 원본을 변경불가를 위해 어떻게 해야하나요?

**프리미티브 타입(원시타입):**

**종류: Number, String, Boolean, Undefined, Null, BigInt, Symbol.**

**메모리에 고정된 크기로 저장되며, 값을 직접 참조합니다.**

**불변성: 값을 변경할 수 없습니다. 새로운 값을 할당하면 새로운 메모리 공간에 저장됩니다.**

원본 변경 방지: 원시 타입은 기본적으로 불변이기 때문에 특별한 조치가 필요하지 않습니다.

**레퍼런스(참조타입):**

**종류: Object, Array, Function 등.**

**메모리에 크기가 동적으로 저장됩니다. 변수는 참조값을 가지며, 이 참조값을 통해 실제 객체에 접근합니다.**

원본 변경 방지: 깊은 복사(deep copy)를 통해 원본 객체의 변경을 방지할 수 있습니다.

**- Object.Assign과 Spread 등 객체를 복사하는 방법들이 가진 문제점이 무엇인가요?**

**> 어떤 단계까지 immutable 한가요?**

**> 참조 없이 완전히 객체를 복사하기 위한 방법은 무엇이 있나요?**

이 방법들은 \*\*얕은 복사(shallow copy)\*\*를 수행합니다.

복사된 객체의 내부 객체(중첩된 객체)는 원본 객체와 동일한 참조를 공유하므로, 복사된 객체에서 내부 객체를 변경하면 원본 객체도 영향을 받습니다.

깊은 복사(deep copy)를 위해 JSON.parse(JSON.stringify(object)) 방법이나 라이브러리를 사용할 수 있습니다.

**- 자바스크립트에서 변수가 유효한 범위는 어떻게 결정이 되나요? var, let, const는 어떻게 다른가요?**

var: 함수 스코프(function scope)를 가집니다. 호이스팅이 발생합니다.

let: 블록 스코프(block scope)를 가집니다. 호이스팅이 발생하지만 TDZ(Temporal Dead Zone)로 인해 선언 전에 참조하면 오류가 발생합니다.

const: 블록 스코프를 가지며, 한번 할당된 값을 변경할 수 없습니다(불변은 아님).

**- 자바스크립트의 호이스팅에 대해서 설명해주세요**

**> 변수를 선언하기 전에 참조하려면 어떤 현상이 일어나나요? 그 현상은 var, let, const가 동일한가요?**

변수와 함수의 선언이 그들이 포함된 스코프의 최상단으로 끌어올려지는 것을 의미합니다.

var로 선언된 변수: 선언은 호이스팅되나, 초기화는 호이스팅되지 않습니다.

let, const로 선언된 변수: TDZ로 인해 선언 전에 참조하면 오류가 발생합니다.

**- 함수의 블록이 실행완료 되면 내부의 변수들은 어떻게 되나요?**

**> 가비지 컬렉터는 실제로 어떤식으로 동작할까요? 아시는대로 최대한 설명해주세요**

함수 실행 컨텍스트가 종료되면 내부의 로컬 변수들은 메모리에서 제거됩니다.

가비지 컬렉터: 참조되지 않는 객체를 메모리에서 자동으로 제거하는 기능입니다. 주로 참조 카운팅과 마크-앤-스위프 방식을 사용합니다.

**- Promise 객체는 어떤 기능을 하나요?**

**> 생성된 (return new 된) Promise 객체는 어떠한 상태를 가지고 있나요?**

비동기 작업의 최종 완료 또는 실패를 나타내는 객체입니다.

상태: pending, fulfilled, rejected.

**- ES7(ES2016)의 async/await 키워드는 무엇이고 어떤 기능을 하나요?**

**> async 함수는 무엇을 반환하나요?**

비동기 작업을 동기적인 방식으로 작성할 수 있게 해줍니다.

async 함수: 항상 Promise를 반환합니다.

**- Generator가 무엇인지 설명 해주세요**

**> 본인이 Generator로 async / await와 같은 코드를 작성한다면?**

**> Babel이 async / await 을 어떤것으로 대체 해두었는지 확인해본적이 있나요?**

함수의 실행을 중간에 일시 중지하고 재개할 수 있는 기능을 제공하는 함수입니다.

yield 키워드를 사용하여 값을 반환하고 일시 중지합니다

async/await와 같은 코드를 작성하기 위해 Generator와 함께 co 라이브러리 등을 사용할 수 있습니다.

Babel은 async/await를 Generator와 Promise로 변환하여 ES5 호환 코드를 생성합니다.

**프론트엔드 전반**

**- CSR과 SSR의 차이점**

CSR(Client-Side Rendering): 초기 페이지 로드 시에는 빈 문서나 로딩 스크린만 받아옵니다. 그 후 JavaScript가 동작하여 데이터를 가져와 페이지를 완성합니다.

SSR(Server-Side Rendering): 서버에서 페이지를 완전히 렌더링한 후에 해당 페이지를 클라이언트에게 전달합니다.

CSR은 초기 로드 속도가 느릴 수 있으나, 이후의 인터랙션에는 빠르게 반응합니다. SEO에 불리할 수 있습니다.

SSR은 초기 로드 속도가 빠르며, SEO에 유리합니다. 그러나 각 요청마다 페이지를 새로 렌더링 해야 하므로 서버에 부하가 갈 수 있습니다.

**- 로컬 스토리지와, 세션 스토리지의 차이**

로컬 스토리지: 브라우저를 종료하더라도 데이터가 유지됩니다. 수동으로 삭제하기 전까지 데이터가 보존됩니다.

세션 스토리지: 브라우저 탭 또는 창을 종료할 때 데이터가 삭제됩니다.

**- 브라우저의 자바스크립트 엔진에서 콜 스택, 태스크 큐, 이벤트 루프에 대해서 설명해주세요**

콜 스택: 실행 중인 함수들의 스택입니다. 함수가 호출되면 스택에 추가(push)되고, 함수의 실행이 완료되면 스택에서 제거(pop)됩니다.

태스크 큐(Event Queue): 비동기 작업(예: setTimeout, 이벤트 콜백)의 결과나 콜백 함수 등을 대기시키는 큐입니다.

이벤트 루프: 콜 스택이 비어 있을 때, 태스크 큐에서 다음 작업을 콜 스택으로 전달하는 역할을 합니다.

**> 어떤 경우에 태스크 큐에 태스크가 등록 되고 이러한 설계의 장점은 무엇인가요?**

태스크 큐에 태스크가 등록되는 경우: 비동기 작업이 완료될 때나, 이벤트(클릭, 키 입력 등)가 발생할 때입니다.

장점: 비동기 작업을 통해 메인 스레드를 차단하지 않고, 사용자와의 상호작용을 계속 유지할 수 있습니다.

**- 브라우저의 주소창에 www.fastcampus.co.kr 엔터키를 치면 일어나는 일들을 "아는대로" 설명해주세요.**

DNS 조회: 도메인 이름을 IP 주소로 변환합니다.

HTTP 요청: 해당 IP 주소로 HTTP GET 요청을 보냅니다.

서버 응답: 서버는 요청받은 페이지의 HTML을 응답으로 전송합니다.

브라우저 렌더링: 받아온 HTML, CSS, JavaScript 등을 해석하고 렌더링합니다.

**- HTTP Method에 어떤것이 있는지, 어떠한 용도로 사용되는지 설명해주세요.**

GET: 리소스를 조회합니다.

POST: 리소스를 생성하거나 데이터를 서버로 보냅니다.

PUT: 리소스를 전체 수정합니다.

PATCH: 리소스의 일부를 수정합니다.

DELETE: 리소스를 삭제합니다.

**- HTTP 상태 코드는 어떤 식으로 이루어져 있나요? 2xx, 3xx, 4xx, 5xx는 각각 어떤것을 의미하나요?**

2xx: 성공

3xx: 리다이렉션

4xx: 클라이언트 에러

5xx: 서버 에러

**HTTP 상태 코드**

**- HTTP Cookie 사용법과 필요성, 장/단점에 대해 알려주세요.**

사용법: 서버가 응답 헤더를 통해 클라이언트에게 쿠키를 설정할 수 있으며, 이후 클라이언트는 해당 쿠키 정보를 매 요청마다 서버에 전송합니다.

필요성: 상태가 없는 HTTP에서 상태 정보를 유지하기 위해 사용됩니다(예: 로그인 상태, 장바구니 정보).

장/단점:

장점: 사용자 식별, 세션 관리, 개인화 설정 등을 구현할 수 있습니다.

단점: 보안 문제, 용량 제한, 사용자의 브라우저에서 쿠키를 차단할 수 있는 가능성이 있습니다.

**> 쿠키와 서버 세션을 사용해 사용자 식별 문제를 해결하는 방법을 설명해주세요**

**- CORS 에 대해서 설명해주실수 있나요?**

Cross-Origin Resource Sharing의 약자로, 다른 출처의 리소스에 접근을 허용하도록 브라우저에게 알려주는 메커니즘입니다.

브라우저의 동일 출처 정책 때문에, 다른 출처에서의 요청은 기본적으로 차단됩니다.

> 관련해서 트러블슈팅 경험

서버 응답 헤더에 Access-Control-Allow-Origin 등의 CORS 관련 헤더를 추가하여 문제를 해결할 수 있습니다.

- 캐시된 에셋들로 인해 이슈를 겪었던 적이 있나요? 어떻게 해결하셨나요?

**> 캐싱과 관련된 HTTP 헤더는 어떤것들이 있나요?**

캐싱과 관련된 HTTP 헤더: Cache-Control, ETag, Last-Modified 등.

**- Etgg나 Cache-control이 어떤 일들을 하게 되나요?**

ETag는 리소스의 특정 버전을 나타내는 식별자입니다. 변경되면 다른 값을 갖게 됩니다.

Cache-control 헤더는 캐시의 동작을 제어하는 지시어들을 포함하며, max-age, no-store, public 등의 값을 포함할 수 있습니다.

- **웹 사이트의 주요 코드를 번들링 하는 이유는 무엇일까요? 구체적으로 어떤 효과를 볼 수 있나요?**

브라우저 로딩 성능 향상: 여러 개의 파일 대신 하나의 번들 파일만 로드합니다.

코드 최적화 및 압축: 불필요한 코드를 제거하고, 파일 크기를 줄일 수 있습니다.

**- Virtual Dom이 무엇인지 설명해주세요.**

브라우저의 실제 DOM 대신 메모리 상에서 표현되는 가상의 DOM입니다.

변경 사항이 발생하면 가상 DOM에서 먼저 이를 반영하고, 실제 DOM과의 차이를 계산(diffing)한 후 필요한 부분만 실제 DOM에 반영(patching)합니다.

**> 바닐라 JS로만 리스트를 만든다고 가정했을 떄. reflow가 한번만 일어나도록 하는 방법**

reflow 최소화: 바닐라 JS로 리스트를 만들 때는 문서의 구조를 바꾸는 대신, 변경 사항을 가상 DOM에서 먼저 반영하고 최종적으로 한 번만 실제 DOM에 반영하도록 설계하여 reflow를 줄일 수 있습니다.

**- 본인이 생각하는 React, Vue 라이브러리를 사용하며 느끼는 전역상태에 관리방법에 대해서 설명해주세요.**

React에서는 Context API, Redux 등을 사용하여 전역 상태를 관리할 수 있습니다.

Vue에서는 Vuex를 사용하여 전역 상태를 관리합니다.

컴포넌트 내부 상태: 각 컴포넌트에서만 관리되고 사용되는 상태입니다. 코드의 지역성이 높아 관리가 용이합니다.

**> 컴포넌트 내부에 상태를 정의 하는 것과, 외부에서 관리하는 방법의 장단점을 설명 해주세요.**

외부(전역) 상태 관리: 여러 컴포넌트에서 공유되는 상태를 효율적으로 관리할 수 있습니다. 복잡한 상태 로직을 centralize하여 관리할 수 있습니다.

- 프론트엔드 테스트 구성 경험이 있으시면 소개해주세요.

**- 소셜 로그인 구현 경험이 있다면 설명 해주세요.**

- Auth 2.0 프로토콜을 사용하여, Google, Facebook 등의 소셜 플랫폼과의 인증 및 인가 과정을 통해 사용자 인증을 구현할 수 있습니다.

**- 성능 최적화 경험이 있다면 알려주세요.**

Lazy loading, 코드 스플리팅, 이미지 최적화, 캐싱 전략 적용 등을 통해 웹 페이지의 로딩 속도와 반응성을 향상시킬 수 있습니다.