

기업 면접 질문 리스트

신입 개발자 기술 면접 준비하기

목 차

01

개발상식

개발상식
자료구조
운영체제

02

백엔드

네트워크
프로그래밍언어
데이터베이스

03

프론트엔드

Web Browser
HTML
CSS

04

기타질문

인성질문
손코딩 테스트



개발상식 01

개발상식

개발상식

자료구조

운영체제

01 개발상식

1) 개발상식

.....

객체지향 프로그램이란?

절차지향 프로그래밍은 오로지 컴퓨터 관점에서의 프로그래밍 패러다임이라면 객체지향은 인간이 구분할 수 있는 요소를 객체로 표현한 인간 중심적 프로그래밍 패러다임이다.

객체는 기억 장소와 이 기억장소의 값을 변경할 수 있는 연산의 집합이 정의되면 객체를 선언할 수 있다. 많은 객체 지향 언어에서 객체는 클래스로 그룹화 된다. 생성된 클래스로 객체의 특정 예를 생성 하는데 이를 인스턴스라고 부른다. 객체 간의 정보 교환이 모두 메시지 교환을 통해 일어난다.

객체 지향 프로그래밍은 4가지 특징을 가진다.

- 추상화 : 공통된 특성을 파악하고 불필요한 특성은 제거한다.
- 캡슐화 : 객체는 상태와 동작을 가지며 객체 스스로 상태를 책임지도록 한다.
- 상속성 : 상위 객체를 상속 받을 수 있도록 한다.
- 다형성 : 동일한 요청에 다른 방식으로 처리할 수 있도록 한다.

01 개발상식

1) 개발상식

.....

함수형 프로그래밍이란?

함수형 프로그래밍은 순수 함수의 조합으로 소프트웨어를 설계하는 방식을 말한다.

순수 함수란 함수의 동작으로 인해 부수 효과(메모리의 값을 직접 수정하는 행위)가 발생하지 않는 함수를 말한다. 객체 지향 언어의 유일한 단점은 객체가 상태를 가지고 있다는 것이다.

따라서 객체는 프로그래머가 의도하지 않은 상태를 가질수도 있다.

함수형 프로그래밍에서 모든 입력은 새로운 출력을 가지므로 부수 효과를 가지지 않는다.

함수형 프로그래밍이 언제나 높은 효율을 보이는 것은 아니다.

상황에 따라 다르며 특히 맵리듀스를 해야하는 상황이라면 효율적이다.

01 개발상식

1) 개발상식

.....

REST(Representational State Transfer) API란?

REST는 자원(Resource), 행위(Verb), 표현(Representations)으로 구성된 API 아키텍처이다.

웹의 장점과 HTTP의 우수성을 적극 활용할 수 있는 아키텍처로 URI 를 통해서 자원을 명시하고 POST , GET , PUT , PATCH , DELETE 등의 Method 를 통해서 해당 자원의 행위를 지정한다.

코드의 재사용성을 높일 수 있으며 프론트엔드와 백엔드의 완전한 분업이 가능해지는 등 장점을 가진다.

TDD란 무엇이며 어떠한 장점이 있는가?

테스트 주도 개발(Test Driven Development), 테스트를 먼저 만들고 테스트를 통과하기 위한 코드를 작성하는 것을 의미하며 모듈화가 자연스럽게 잘 이루어지면서 개발이 진행된다. 테스트 커버리지가 높아져 리팩토링과 유지보수가 쉬워진다.

01 개발상식

1) 개발상식

.....

MVC 패턴이란 무엇인가?

모델(Model), 뷰(View), 컨트롤러(Controller)가 분리된 형태의 아키텍처이다. 세가지가 결합된 형태에서는 어플리케이션의 확장이 어렵다. 모델은 데이터 처리, 뷰는 사용자 인터페이스 처리, 컨트롤러는 비즈니스 로직을 처리하는 등 각각의 요소가 하나의 역할만 담당한다.

Git과 GitHub에 대해서

Git은 분산 버전 관리 시스템으로 소스 코드와 파일의 변경 이력을 관리하는 도구다. Git은 SVN 과 같은 중앙 집중식 버전 관리 시스템에 비해서 가볍고 브랜치 관리가 유연하다는 특징을 가진다. GitHub는 Git 기반의 서비스이다.

Docker와 VM의 차이점

- Docker은 Virtual Machine에서 사용되는 Guest OS 의 이미지를 사용하지 않기 때문에 가볍다는 장점이 있다.
- App을 실행하는 방식에 있어 Docker 방식에서는 호스트 OS 위에 어플리케이션의 이미지를 배포하기만 하면 되지만, VM은 어플리케이션을 실행하기 위해서 VM을 띄우고 자원을 할당한 다음, Guest OS를 부팅하므로 훨씬 복잡하고 무겁게 실행해야 한다.



개발상식 01

개발상식

개발상식

자료구조

운영체제

01 개발상식

2) 자료구조

.....

순차 자료구조 vs 연결 자료구조

순차 자료구조는 메모리 상에서 일렬로 나열된 데이터형이며 연결 자료구조는 메모리 상에서는 분산되어 있지만 하나의 노드가 다음 노드로 이어지는 포인터를 가지고 있어 연속적으로 접근이 가능한 데이터형이다. 삽입과 읽기를 비교해보면 아래와 같다.

데이터 삽입

- 순차 자료구조 : 마지막 인덱스에 데이터를 삽입하는 경우엔 항상 빠르다. 처음 혹은 중간에 삽입하는 경우 자리교환으로 인한 오버헤드가 발생하므로 느리다.
- 연결 자료구조 : 데이터를 어디에 삽입하던 해당 노드까지 액세스하는 시간이 소요된다. 자료 추가시 연결된 노드의 링크만 교체하면 되므로 빠르다.

데이터 읽기

- 순차 자료구조 : 탐색하려는 위치를 알고 있다면 즉시 액세스 할 수 있으므로 빠르다. 위치를 모른다고 하더라도 메모리 상에서 근접한 데이터의 접근이 더 빠르므로 연결 자료구조보다 빠르다.
- 연결 자료구조 : 위치를 알던 모르던 관계없이 헤더 혹은 테일부터 찾으려는 위치까지 탐색이 필요하며 메모리에서 노드가 분산되어 있으므로 순차 자료구조에 비해 느리다.

01 개발상식

2) 자료구조

.....

트리

트리는 계층 구조로 구성된 자료구조이다. 트리의 모든 노드는 최대 하나의 부모 노드를 가진다.

부모 노드를 가지지 않는 최상위 노드를 루트 노드, 자식 노드가 없는 노드를 리프 노드라고 한다. 트리의 크기를 제한하면 트리의 연산이 단순해지고 명확해지는데 차수를 2개 이하로 정의한 것이 이진 트리이다. 이진 트리의 종류로는 스택 이진 트리, 이진 탐색 트리, AVL 트리 등이 있다.

힙(Heap)

힙은 이진트리의 한 종류로 나열한 두 가지 조건이 성립하는 이진 트리를 의미한다.

- 완전 이진 트리여야 한다.
- 부모 노드와 자식 노드간에 크기 관계가 성립해야 한다.

루트 노드가 가장 크고 자식 노드가 부모 노드보다 작으면 최대 힙이라 부르고, 반대의 경우는 최소 힙이라 부른다. 힙은 최댓값과 최솟값에 접근하기 위해 사용하며 성능이 매우 빠르다.

01 개발상식

2) 자료구조

.....

그래프

정점과 에지로 이루어진 형태의 자료구조다. 에지의 방향성의 존재 유무에 따라서 유향 그래프와 무향 그래프로 분리되며, 에지가 가중치를 가지고 있다면 가중치 그래프라고 부른다. 그래프는 행렬과 연결리스트를 활용하여 구현할 수 있는데 행렬의 경우 정점의 존재 여부와 상관없이 항상 n^2 의 공간 복잡도를 가진다.

해쉬

해쉬는 임의의 크기를 가진 데이터를 고정된 크기의 값으로 변환시키는 것을 말한다. 해쉬를 이용하여 임의의 데이터를 숫자로 변경하는 해쉬 함수를 정의하면 배열의 인덱스를 원하는 데이터 값으로 저장하거나 찾을 수 있다. 기존에는 탐색을 위한 시간이 소모됨에 반해 해쉬를 이용 하면 즉시 데이터에 액세스 할 수 있다. 단, 다른 입력이 같은 입력을 생성하는 현상을 해시 충돌 이라고 하는데 해시 함수는 해시 충돌을 염두하여 구현하는 것이 중요하다.

01 개발상식

2) 자료구조

.....

Array와 LinkedList의 차이가 무엇인가요? (N사 전화면접)

Array는 Random Access를 지원한다. 요소들을 인덱스를 통해 직접 접근할 수 있다. 따라서 특정 요소에 접근하는 시간복잡도는 $O(1)$ 이다. 반면 LinkedList는 Sequential Access를 지원한다. 어떤 요소를 접근할 때 순차적으로 검색하며 찾아야 한다. 따라서 특정 요소에 접근할 때 시간복잡도는 $O(N)$ 이다.

저장방식도 배열에서 요소들은 인접한 메모리 위치에 연이어 저장된다. 반면 LinkedList에서는 새로운 요소에 할당된 메모리 위치 주소가 linkedlist의 이전 요소에 저장된다.

배열에서 삽입과 삭제는 $O(N)$ 이 소요되지만, LinkedList에서 삽입과 삭제는 $O(1)$ 이 소요된다.

배열에서 메모리는 선언 시 컴파일 타임에 할당이 된다.(정적 메모리 할당) 반면 LinkedList에서는 새로운 요소가 추가될 때 런타임에 메모리를 할당한다.(동적 메모리 할당) 배열은 Stack 섹션에 메모리 할당이 이루어 진다. 반면 LinkedList는 Heap 섹션에 메모리 할당이 이루어진다.

Stack과 Queue의 차이점은 무엇인가요? (N사 전화면접)

스택은 쌓아 올리는 자료구조이다. 같은 구조와 크기의 자료를 정해진 방향으로 쌓을 수 있고, 한방향으로만 접근할 수 있다. top을 통해서 push, pop을 하면서 삽입과 삭제가 일어난다. 후입선출 구조이다. DFS나 재귀에서 사용된다.

큐는 원소의 줄을 세우는 자료구조이다. 큐는 한 쪽 끝에서 삽입 작업을, 다른 쪽 끝에서 삭제 작업을 진행한다. 선입선출 구조이다. 주로 데이터가 입력된 시간 순서대로 처리되어야 하는 경우 사용 한다. BFS나 캐시를 구현할 때 사용한다.

01 개발상식

2) 자료구조

.....

BST와 Binary Tree에 대해서 설명하세요. (N사 전화면접)

이진탐색트리(Binary Search Tree)는 이진 탐색과 연결 리스트를 결합한 자료구조이다. 이진 탐색의 효율적인 탐색 능력을 유지하면서, 빈번한 자료 입력과 삭제가 가능하다는 장점이 있다. 이진 탐색 트리는 왼쪽 트리의 모든 값이 반드시 부모 노드보다 작아야 하고, 반대로 오른쪽 트리의 모든 값이 부모 노드보다 커야 하는 특징을 가지고 있어야 한다. 이진 탐색 트리의 탐색, 삽입, 삭제의 시간복잡도는 $O(h)$ 이다. 트리의 높이에 영향을 받는데, 트리가 균형이 맞지 않으면 워스트 케이스가 나올 수 있다. 그래서 이 균형을 맞춘 구조가 AVL Tree이다.

PriorityQueue의 동작 원리가 어떻게 되나요? (N사 전화면접)

우선순위 큐는 힙이라는 자료구조를 가지고 구현한다. top이 최대면 최대힙, top이 최소면 최소힙 으로 표현한다. 힙으로 구현된 이진 트리는 모든 정점이 자신의 자식 요소보다 우선순위가 높다는 성질을 가지고 있다. 이 성질을 통해 삽입과 삭제 연산을 모두 $O(\log N)$ 으로 수행할 수 있다.

01 개발상식

2) 자료구조

.....

해시테이블(HashTable)에 대해서 설명해주세요.

해시테이블은 효율적인 탐색을 위한 자료구조로 key값을 value에 대응시킨다. 해시테이블을 구현 하기 위해서는 연결 리스트와 해쉬 함수가 필요하다. 해싱은 임의의 길이의 값을 해쉬 함수를 통해 고정된 크기의 값으로 변환하는 작업을 말하는데, 키 값을 해시 코드로 변환한 후 해당 해시 코드로 배열의 인덱스를 참조하여 값을 찾는다. 충돌이 발생할 수 있으며, 최악의 경우 $O(N)$, 일반적으로 잘 구현된 경우는 $O(1)$ 의 시간 복잡도를 가지게 된다. 충돌은 Chaining, Open addressing 등의 방식으로 해결할 수 있다.

해시테이블은 균형 이진 탐색 트리로도 구현할 수 있다. 이 경우는 탐색 시간이 $O(\log N)$ 이 된다.

이 방법은 크기가 큰 배열을 미리 할당해 놓지 않아도 되기 때문에 잠재적으로 적은 공간을 사용한 다는 장점이 있다.

최소 스패닝 트리(Minimum Spanning Tree)에 대해서 설명해주세요.

그래프 G 의 스패닝 트리 중 edge weight 값이 최소인 스패닝 트리를 말한다. 스패닝 트리란 그래프 G 의 모든 vertex가 cycle 없이 연결된 형태를 말한다. n 개의 vertex를 가지는 그래프에서 반드시 $(n-1)$ 개의 edge만을 사용해야 하며 사이클이 포함되어서는 안 된다. Kruskal과 Prim을 통해서 MST 를 구현할 수 있다. Kruskal의 경우 그래프의 간선들을 오름차순으로 정렬하여 가장 낮은 가중치의 간선부터 차례로 MST에 집합체 추가하는 그리디 알고리즘 방식을 사용한다. Prim의 경우는 시작 정점부터 단계적으로 트리를 확장하는 방법이다.



개발상식 01

개발상식

개발상식

자료구조

운영체제

01 개발상식

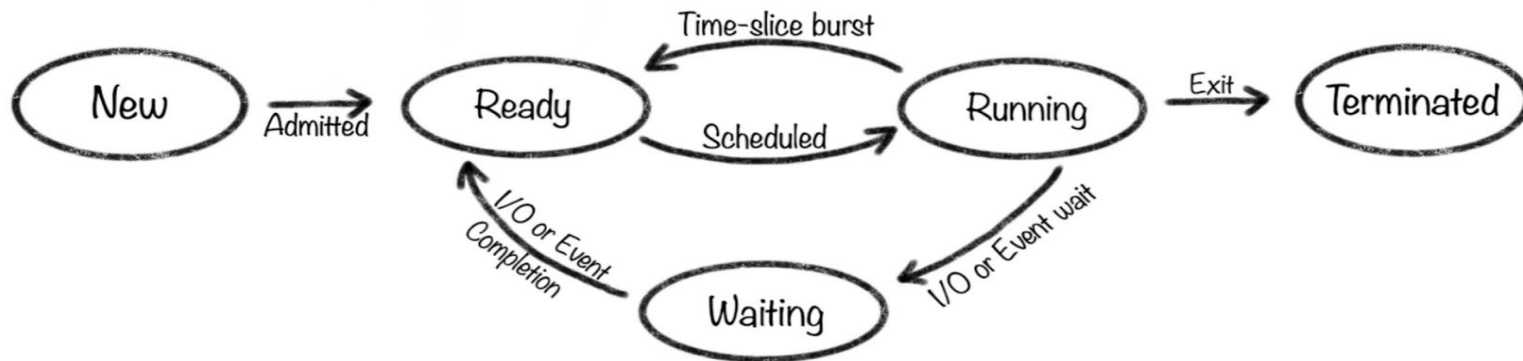
3) 운영체제

스케줄러의 종류

- 장기 스케줄러: 메모리로 프로세스를 적재할지 결정하는 스케줄러로, 디스크에서 메모리로의 스케줄링을 담당한다.
- 단기 스케줄러: 실행 가능한 프로세스 중 어떤 프로세스를 선택하여 CPU를 할당할지 결정 하는 스케줄러로, CPU 스케줄러라고도 한다.

CPU 스케줄러

- 다중 프로그램 OS의 기본으로, 여러 프로세스들이 CPU를 교환하며 보다 생산적으로 동작한다.
- CPU를 선점한 프로세스가 대기하는 시간을 보다 효율적으로 사용하기 위하여 사용한다.
- 기본적으로 다음에 실행될 프로세스를 정하고, 프로세스들을 실행가능한 상태로 만든다.



01 개발상식

3) 운영체제

.....

동기와 비동기의 차이

- 동기: 작업을 순차적으로 진행하며, 하나의 작업이 끝나야 다음 작업을 수행하는 방식. 호출된 함수가 반환될 때까지 대기한다.
- 비동기: 작업을 순차적으로 기다리지 않고, 요청한 작업을 다른 스레드 또는 프로세스에게 위임하고 즉시 다음 작업을 수행하는 방식. 작업 완료 여부를 확인하기 위해 콜백 또는 알림 메커니즘을 사용한다.

멀티스레드

- 하나의 프로세스 내에서 여러 스레드를 생성하여 작업을 동시에 수행하는 것을 말한다. 이로 인해 자원 공유 및 작업 처리 성능이 향상되며, 병렬 처리가 가능해진다.

프로세스 동기화

- 멀티스레드 환경에서 여러 스레드가 공유 자원에 동시에 접근할 때, 이를 조절하여 데이터의 무결성과 일관성을 유지하는 기술이다. 잘못된 동기화는 경쟁 상태와 교착상태를 유발할 수 있다.

메모리 관리 전략

- 프로세스가 메모리에 적재되고 실행되는 방식을 관리하는 기법들이다. 주요 전략에는 단일 고정 분할, 가변 분할, 페이지 기반 가상 메모리 등이 있다.

가상 메모리

- 물리적인 실제 메모리보다 큰 프로세스를 실행하기 위해 디스크 공간을 일부 마치 메모리인 것처럼 사용하는 기술이다. 이를 통해 여러 프로세스가 동시에 실행될 수 있고, 프로세스간 메모리 보호가 가능해진다.

01 개발상식

3) 운영체제

.....

캐시(Cache)의 지역성(Locality)

- 캐시가 효율적으로 동작하려면, 캐시의 적중율(Hit-rate)를 극대화 시켜야 한다.
- 캐시에 저장할 데이터가 지역성(Locality)을 가져야 한다.
- 지역성이란, 데이터 접근이 시간적, 혹은 공간적으로 가깝게 일어나는 것을 의미한다.
- 지역성의 전제 조건으로 프로그램은 모든 코드나 데이터를 균등하게 Access하지 않는다는 특성을 기본으로 한다.
- 즉, 지역성(Locality)이란 기억장치 내의 정보를 균일하게 Access하는 것이 아닌 어느 한 순간에 특정 부분을 집중적으로 참조하는 특성이다.

지역성(Locality)의 종류

시간적 지역성

- 최근에 참조된 주소의 내용은 곧 다음에 다시 참조되는 특성.
- 특정 데이터가 한번 접근되었을 경우, 가까운 미래에 또 한번 데이터에 접근할 가능성이 높은 것
- 메모리 상의 같은 주소에 여러 차례 읽기 쓰기를 수행할 경우, 상대적으로 작은 크기의 캐시를 사용해도 효율성을 꺾을 수 있다.

공간적 지역성

- 대부분의 실제 프로그램이 참조된 주소와 인접한 주소의 내용이 다시 참조되는 특성
- 특정 데이터와 가까운 주소가 순서대로 접근되었을 경우, CPU 캐시나 디스크 캐시의 경우 한 메모리 주소에 접근할 때 그 주소뿐 아니라 해당 블록을 전부 캐시에 가져오게 된다.
- 이때 메모리 주소를 오름차순이나 내림차순으로 접근한다면, 캐시에 이미 저장된 같은 블록의 데이터를 접근하게 되므로 캐시의 효율성이 크게 향상된다.

01 개발상식

3) 운영체제

.....

컴파일러와 인터프리터의 차이가 무엇인가요? (N사 전화면접)

컴파일러와 인터프리터 모두 고레벨 언어를 기계어로 변환하는 역할을 수행하지만 차이점은 컴파일러의 경우 전체 코드를 보고 명령어를 수집하고 재구성하는 반면, 인터프리터는 소스코드의 각행을 연속적으로 분석하며 실행한다. 인터프리터는 고레벨 언어를 중간 레벨 언어로 한 번 변환하고 이를 각 행마다 실행하기 때문에 일반적으로 컴파일러가 인터프리터보다 실행 시간이 빠른 경우가 많다. java의 경우 .java 파일을 .class 파일로 자바 컴파일러가 컴파일을 하고, .class 파일을 기계어로 인터프리터가 변환하는 것이다.

프로세스와 스레드의 차이가 무엇인가요? (N사 전화면접)

프로세스는 운영체제로부터 CPU 자원을 할당받는 작업의 단위이다. 그리고 메모리에 올라와 실행 되고 있는 프로그램의 인스턴스(독립적인 개체)를 의미한다. Code, Data, Stack, Heap의 구조로 되어 있다, 이러한 구조로 된 독립된 메모리 영역을 할당받는다.

스레드는 프로세스 내에서 실행되는 여러 흐름의 단위이다. 스레드는 프로세스 내에서 각각 Stack 만 따로 할당받고 Code, Data, Heap 영역은 공유한다.

멀티 프로세스에 비해 멀티 스레드가 가지는 장점은 먼저 첫 번째로 프로세스를 생성하여 자원을 할당하는 시스템 콜이 감소함으로써 자원을 효율적으로 관리할 수 있다는 점이다. 두 번째로 프로세스 간 통신(IPC)보다 스레드 간의 통신 비용이 더 적게 발생한다. 다만 멀티 스레드 사용 시에는 공유 자원으로 인한 문제를 해결하기 위해 동기화를 신경써 주어야 한다.

01 개발상식

3) 운영체제

.....

LRU에 대해서 설명해 보세요. (N사 전화면접)

LRU는 페이지 교체 알고리즘 중 하나로, 최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 방식이다.

Thread-safe의 개념에 대해 설명하세요. (K사 전화면접)

Thread safe는 멀티 스레드 프로그래밍 환경에서 일반적으로 어떤 함수나 변수, 혹은 객체가 여러 스레드로부터 동시에 접근이 이루어져도 프로그램의 실행에 문제가 없는 것을 말한다.

Thread-safe한 코드를 만들기 위해서는 Critical Session을 통해 스레드 내부에서 처리되는 연산들을 직렬화 하여 한 번에 한 스레드에서 연산이 수행되도록 만들어 주어야 한다.

GPU와 CPU의 차이에 대해 설명하세요. (K사 전화면접)

CPU는 입출력장치, 기억장치, 연산장치 등을 포함한 컴퓨터 리소스를 관리하는 최상위 계층의 중앙처리장치이다. 따라서 데이터 처리와 더불어 프로그램에서 작업의 우선순위를 지정하고 전환하며 가상 메모리를 관리하는 등의 역할을 한다. CPU는 직렬 처리에 최적화된 몇 개의 코어로 구성 되어 있다.

GPU는 반복적이고 비슷한 대량의 연산을 수행하며 이를 병렬적(parallel)으로 나누어 작업하기 때문에 CPU에 비해 속도가 압도적으로 빠르다. GPU는 병렬 처리용으로 설계된 수 천 개의 보다 소형이고 효율적인 코어로 구성되어 있다.

01 개발상식

3) 운영체제

.....

교착상태(데드락)가 무엇이고 발생하는 조건을 설명해 주세요.

교착 상태는 프로세스가 자원을 얻지 못해 다음 처리를 하지 못하는 상태로, 시스템적으로 한정된 자원을 여러 곳에서 사용하려고 할 때 발생한다.

상호 배제, 점유 대기, 비선점, 순환 대기 네 가지 조건을 모두 만족해야 교착 상태가 발생한다. 순환 대기의 경우 점유 대기 와 비선점 조건을 만족해야 성립하므로 4가지 조건은 완전히 서로 독립 적이지 않다.

페이지 폴트가 무엇인가요?

가상 메모리의 페이지 테이블에는 페이지가 물리 메모리에 있는지, 스왑 영역에 있는지 표시하는 유효 비트를 사용한다. 프로세스가 페이지를 요청했을 때 그 페이지가 메모리에 없는 경우를 페이지 폴트라고 한다. 페이지 폴트가 발생하면 프레임을 새로 할당 받아야 하며, 프로세스가 해당 페이지를 사용할 수 있도록 스왑 영역에서 물리 메모리로 옮겨야 한다. 그리고 페이지 테이블을 재구성 하고, 프로세스의 작업을 재 시작한다.

가상 메모리(Virtual Memory)에 대해 설명해 주세요.

가상 메모리는 멀티 프로세스 환경에서 프로세스마다 충분한 메모리를 할당하기에 물리 메모리의 한계가 있어서 나타난 개념이다. 가상 메모리에서 프로세스는 가상 주소를 사용하고, 실제 해당 주소에서 데이터를 읽고 쓸 때 물리 주소로 바꿔주게 된다. MMU(Memory Management Unit)를 통해 CPU에서 코드 실행 시, 가상 메모리 접근이 필요할 때, 해당 주소를 물리 주소로 변환해 준다.

01 개발상식

3) 운영체제

.....

문맥 전환(Context Switching)이 무엇인가요?


멀티 프로세스 환경에서 CPU가 어떤 하나의 프로세스를 실행하고 있는 상태에서, 인터럽트 요청에 의해 다음 우선 순위의 프로세스가 실행되어야 할 때 기존의 프로세스의 상태 또는 레지스터 값 (context)을 저장하고 CPU가 다음 프로세스를 수행하도록 새로운 프로세스의 상태 또는 레지스터 값(context)을 교체하는 작업을 context switching이라고 한다.

OS에서 context는 CPU가 해당 프로세스를 실행하기 위한 해당 프로세스의 정보들이다. 이 context는 프로세스의 PCB(Process Control Block)에 저장된다.

메모리가 어떻게 구성되어 있는지 설명해 주세요.

메모리는 크게 코드, 데이터, 스택, 힙 영역으로 나누어져 있다. 코드 영역은 실행될 프로그램의 코드가 저장되어 있는 영역이다. 데이터 영역은 전역 변수와 정적 변수가 저장되어 있는 영역이다.

스택 영역은 지역변수와 매개 변수가 저장되어 있으며, 함수의 호출과 함께 할당되는 영역이다. 힙영역은 사용자에게 의해 동적으로 할당되고 해제될 수 있는 메모리 영역이다. 스택 영역은 컴파일 타임에 크기가 결정되고, 힙 영역은 런 타임에 크기가 결정된다.



백엔드 02

네트워크

프로그래밍언어

데이터베이스

02 백엔드

1) 네트워크

.....

GET, POST 방식의 차이점

- GET : 클라이언트는 GET 요청시 URI 와 Header 에만 데이터를 담아서 전송할 수 있으므로 데이터 크기가 제한적이며 URI 에 포함된 데이터는 사용자에게 그대로 노출된다. GET 요청은 캐싱이 가능하므로 단순 조회이거나 데이터가 사용자에게 노출되어도 상관없다면 적극 활용하는게 좋다.
- POST : 데이터를 URI 와 Header 그리고 Body 에 담아서 전송할 수 있으므로 GET 방식보다 전송할 수 있는 데이터가 현저히 크다Body 데이터는 일반 사용자에게 노출되지 않으므로 안정적으로 데이터를 전송할 수 있다.

TCP 3-way-handshake

- Client는 Server에 접속 요청 메시지(SYN)을 전송하고 SYN_SEND 상태가 된다.
- Server는 SYN 요청을 받고 Client에 요청을 수락(SYN+ACK)하고 SYN_RECEIVED 상태가 된다.
- Client는 Server에게 수락 확인(ACK)를 보내고 Server는 ESTABLISHED 상태가 된다.

TCP와 UDP의 차이점

- TCP : 신뢰성과 순차적인 전달이 필요한 경우 사용한다. TCP 서비스는 송신자와 수신자 모두가 소켓이라고 부르는 종단점을 생성함으로써 이루어진다. TCP는 멀티캐스팅이나 브로드 캐스팅을 지원하지 않는다.
- UDP : 비연결형 프로토콜이며 손상된 세그먼트의 수신에 대한 재전송을 하지 않는다. UDP 를 사용하는 것에는 DNS가 있다. 사전에 설정이 필요하지 않으며 그 후에 해제가 필요하지 않다.

02 백엔드

1) 네트워크

.....

HTTP와 HTTPS의 차이점

HTTP(Hypertext Transfer Protocol)는 클라이언트와 서버 간 통신을 위한 통신 규칙 세트 또는 프로토콜이다. 사용자가 웹 사이트를 방문하면 사용자 브라우저가 웹 서버에 HTTP 요청을 전송하고 웹 서버는 HTTP 응답으로 응답한다. 웹 서버와 사용자 브라우저는 데이터를 일반 텍스트로 교환한다. 간단히 말해 HTTP 프로토콜은 네트워크 통신을 작동하게 하는 기본 기술이다. 이름에서 알 수 있듯이 HTTPS(Hypertext Transfer Protocol Secure)는 HTTP의 확장 버전 또는 더 안전한 버전이다. HTTPS에서는 브라우저와 서버가 데이터를 전송하기 전에 안전하고 암호화된 연결을 설정한다.

DNS round robin 방식

DNS를 이용해서 하나의 서비스에 여러 대의 서버를 분산 시키는 방법이다. 동일한 이름으로 여러 레코드를 등록 시키면 질의 할 때마다 다른 결과를 반환하며, 이 동작을 이용함으로써 여러 대의 서버에 처리를 분산 시킬 수 가 있다. 단점은 아래와 같다.

- 서버의 수 만큼 공인 IP 주소가 필요하다.
- DNS 질의 결과 캐싱으로 인해 균등하게 분산되지 않는다.
- 서버가 다운되어도 확인이 어렵다.


02 백엔드

1) 네트워크

.....

웹 통신의 흐름

- 사용자의 요청: 사용자가 웹 브라우저를 통해 특정 웹 페이지나 자원에 접속하면, 해당 요청이 클라이언트로부터 HTTP 프로토콜을 통해 서버로 전달된다.
- DNS 조회: 클라이언트가 요청한 웹 페이지의 도메인 이름을 IP 주소로 변환하기 위해 DNS 서버에 조회를 요청한다. (DNS는 ISP, 호스팅 회사, 인터넷 기업 등 다양한 조직이 제공)
- 서버 접속: DNS 조회를 통해 얻은 서버의 IP 주소를 사용하여 클라이언트가 웹 서버에 요청 한다.
- 서버 처리: 웹 서버는 클라이언트의 요청을 받아들이고, 요청을 처리한다.
- 응답 전송: 서버가 클라이언트의 요청을 처리한 결과를 HTTP 응답 헤더와 바디에 담아 클라이언트로 전송한다. 헤더에는 요청 결과에 대한 상태나 콘텐츠에 대한 정보가 포함되며 바디에는 콘텐츠의 실질적인 내용이 포함된다. (내용은 페이지 혹은 데이터, 정적인 파일 등이될 수 있다.)
- 클라이언트 처리: 브라우저는 서버로부터 받은 응답을 해석하여 사용자에게 보여준다. 페이지가 렌더되는 경우 페이지에 명시된 추가적인 자원을 요청하여 페이지를 완성한다.



백엔드 02

네트워크

프로그래밍언어

데이터베이스

02 백엔드

2) 프로그래밍언어(Java)

jvm의 역할

- 자바 어플리케이션을 클래스 로더를 통해 읽어들이 자바 API와 함께 실행
- Java와 OS 사이에서 중개자 역할을 수행하여 Java가 OS에 구매받지 않고 재사용을 가능하게 함
- 프로그램에 필요한 메모리 할당, 관리
- GC(Garbage Collection) 수행

GC(Garbage Collection)의 역할

- 힙(Heap) 내의 객체 중 Garbage를 찾아낸다.
- 찾아낸 Garbage 객체를 반환하여 메모리를 회수한다.

래퍼(wrapper) 클래스

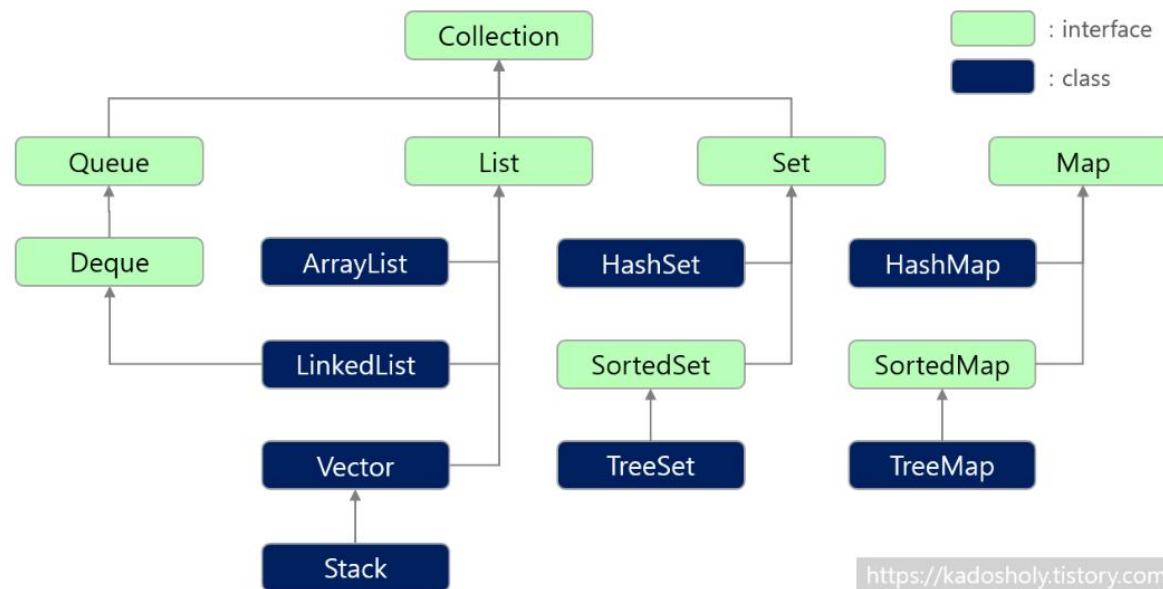
자바의 자료형은 크게 기본 타입(primitive type)과 참조 타입(reference type)으로 나뉘어진다. 대표적으로 기본 타입은 char, int, float, double, boolean 등이 있고 참조 타입은 class, interface 등이 있는데 프로그래밍을 하다 보면 기본 타입의 데이터를 객체로 표현해야 하는 경우가 종종 있다. 이럴 때에 기본 자료타입(primitive type)을 객체로 다루기 위해서 사용하는 클래스들을 래퍼 클래스(wrapper class)라고 한다. 자바는 모든 기본타입(primitive type)은 값을 갖는 객체를 생성할 수 있다. 이런 객체를 포장 객체라고도 하는데 그 이유는 기본 타입의 값을 내부에 두고 포장하기 때문이다. 래퍼 클래스로 감싸고 있는 기본 타입 값은 외부에서 변경할 수 없다. 만약 값을 변경하고 싶다면 새로운 포장 객체를 만들어야 한다.

02 백엔드

2) 프로그래밍언어(Java)

컬렉션(collection)이란?

컬렉션(collection)이란 많은 수의 데이터를 그 사용 목적에 적합한 자료구조로 묶어 하나로 그룹화한 객체를 말합니다. 자바에서는 이러한 컬렉션을 위한 인터페이스와 클래스들이 있으며 자주 사용되는 클래스들의 종류와 상속 계층도는 아래와 같습니다.



<https://kadosholy.tistory.com>

Collection 상속도

02 백엔드

2) 프로그래밍언어(Java)

제네릭

제네릭은 다양한 타입의 객체들을 다루는 메서드나 컬렉션에 컴파일 시 타입 체크를 해주는 기능이다. 클래스 내부에서 사용할 데이터 타입을 나중에 인스턴스를 생성할 때 확정하는 것을 제네릭이라 한다. 객체의 타입을 컴파일 시에 체크하기 때문에 객체의 타입 안정성을 높이고 형변환의 번거로움이 줄어든다.

파이널

원시 변수에 final 선언 시 immutable 하여 상수로 사용 가능하다. 객체 변수에 final 선언시 레퍼런스 변경이 불가능하다. 하지만 객체 자체는 mutable 하다. 메서드에 final 선언 시 override 가 불가능 하다. 클래스에 final 선언 시 상속이 불가능하다.

오버로딩 오버라이딩

- 오버로딩 : 같은 이름의 메소드를 여러 개 가지는 것이다. 이때 매개변수가 서로 달라야 한다.
- 오버라이딩 : 상속 시 부모 클래스의 메소드를 상속 받은 자식 클래스에서 재정의 하는 것이다.

02 백엔드

2) 프로그래밍언어(Java)

어노테이션이란? (Annotation)

어노테이션은 다른 프로그램에게 유용한 정보를 제공하기 위해 사용되는 것으로 주석과 같은 의미를 가진다.

어노테이션의 역할

- 컴파일러에게 문법 에러를 체크하도록 정보를 제공한다.
- 프로그램을 빌드할 때 코드를 자동으로 생성할 수 있도록 정보를 제공한다.
- 런타임에 특정 기능을 실행하도록 정보를 제공한다.

어노테이션은 @를 사용하여 작성하며, 해당 타겟에 대한 동작을 수행하는 프로그램 외에는 다른 프로그램에게 영향을 주지 않는다.

접근제어자(access modifier)

클래스 멤버와 함께 사용할 수 있는 접근제어자는 public, private, protected 와 default 네가지입니다.

- public 멤버 : public 은 '공개'를 나타내며, 모든 클래스에서 접근이 가능합니다 (패키지와 상관없음)
- private 멤버 : private은 '비공개'를 나타내며, 같은 클래스안에 있는 멤버들만 접근이 가능합니다.
- protected 멤버 : 같은 패키지안의 모든 클래스와, 다른 패키지의 자식 클래스에서 접근이 가능합니다.
- default(생략) 멤버 : 접근지정자가 없으면 default 멤버로, 같은 패키지안의 클래스에서만 접근이 가능합니다.

02 백엔드

2) 프로그래밍언어(Java)

Multi-Thread 환경에서의 개발

서로 다른 스레드가 데이터와 힙 영역을 공유하기 때문에 어떤 스레드가 다른 스레드에서 사용중인 변수나 자료 구조에 접근하여 엉뚱한 값을 읽어오거나 수정할 수 있다.

그렇기 때문에 멀티스레딩 환경에서는 동기화 작업이 필요하다. 동기화를 통해 작업 처리 순서를 컨트롤 하고 공유 자원에 대한 접근을 컨트롤 하는 것이다. 하지만 이로 인해 병목 현상이 발생하여 성능이 저하될 가능성이 높다. 그러므로 과도한 락(lock)으로 인한 병목 현상을 줄여야 한다. 공유 자원이 아닌 부분은 동기화 처리를 할 필요가 없다.

즉, 동기화 처리가 필요한 부분에만 synchronized 키워드를 통해 동기화하는 것이다. 불필요한 부분까지 동기화를 할 경우 현재 스레드는 락(lock)을 획득한 스레드가 종료하기 전까지 대기해야한다. 그렇게 되면 전체 성능에 영향을 미치게 된다. 즉 동기화를 하고자 할 때는 메소드 전체를 동기화 할 것인가 아니면 특정 부분만 동기화할 것인지 고민해야 한다.

02 백엔드

2) 프로그래밍언어(Javascript)

자바스크립트 이벤트 루프

이벤트 루프는 테스크 큐와 콜 스택을 수시로 감시하여 콜 스택에 처리할 함수가 없고, 테스크 큐에 처리할 함수가 존재하면 테스크 큐에 있는 함수를 콜 스택으로 전달하는 역할을 담당한다.

예를들어 setTimeout을 호출한 경우 WebAPI에서 처리한 뒤 프로그래머가 지정한 작업(콜백 함수)이 테스크 큐에 등록된다. 이후 이벤트 루프에 의해 콜 스택으로 전달되어 처리된다.

Prototype

자바스크립트는 프로토타입 기반의 객체 지향 프로그래밍을 지원하며, 자바스크립트의 대부분은 객체이고, 모든 객체는 부모 객체와 연결되어 있다. 이때 부모 객체를 프로토타입이라 한다.

자바스크립트에서는 프로토타입 체인을 이용하여 상속을 구현한다. 인스턴스의 프로퍼티 혹은 메서드를 호출한 경우 인스턴스에 해당 프로퍼티 혹은 메서드가 존재하지 않으면 프로토타입 내부 슬롯의 참조로 이동하여 과정을 반복한다.

프로토타입 프로퍼티는 함수 객체만 가지고 있으며 이는 생성될 인스턴스의 프로토타입을 가리 킨다. 프로토타입 내부 슬롯은 모든 객체가 가지며 프로토타입 객체의 참조를 가진다. 대부분의 경우 프로토타입 체인의 종단점은 Object.prototype 이다.

02 백엔드

2) 프로그래밍언어(Javascript)

이벤트 버블링(Event Bubbling), 이벤트 캡처링(Event Capturing)에 대해서 설명하세요.

이벤트 버블링은 특정 화면 요소에서 이벤트가 발생했을 때 더 상위 요소들로 전달되어 가는 특성을 의미한다. 이벤트 캡처링은 이벤트 버블링과 반대로 상위 요소에서 하위 요소로 탐색하며 이벤트를 전파하는 방식이다.

이벤트 위임 (event delegation) 에 대해 설명하세요. (D사 화상면접)

특정 요소 하나하나를 개별적으로 이벤트를 부여하는 것이 아니라, 하나의 부모에 이벤트를 등록 하여 부모가 이벤트를 위임하는 방식을 이벤트 위임이라고 한다. 이 방법은 동적인 요소들에 대한 처리가 수월하며 이벤트 핸들러를 더 적게 등록해 주기 때문에 메모리도 절약할 수 있다.

let, var, const의 차이점에 관해서 설명해주세요. (V사 기술면접)

var 선언은 전역 범위 또는 함수 범위이며, let 과 const 는 블록 범위이다. var 변수는 범위 내에서 업데이트 및 재선언할 수 있다. let 변수는 업데이트할 수 있지만, 재선언은 할 수 없다.

02 백엔드

2) 프로그래밍언어(Javascript)

prototype 기반 상속은 어떻게 하는지 설명해주세요.

자바스크립트 객체에는 Prototype이라는 내부 프로퍼티가 있고, 이는 다른 객체를 참조할 때 사용 한다. 자바스크립트에서 상속을 진행할 때는 프로토타입끼리 연결을 하는데, 부모 프로토타입을 create()나 setPrototypeOf() 메서드를 사용하여 자식 프로토타입과 연결한다.

null과 undefined의 차이점은 무엇인가요?

두 타입 모두 값이 없음을 의미한다. 둘 다 데이터 타입이자 그 변수의 값이다. 자바스크립트에서 변수를 선언하면 초기값으로 undefined를 할당하게 된다. 반면 null은 값이 비어있음을 나타내며 값이 없다는 값이 등록되어 있는 것이다.

익명함수(anonymous functions)는 주로 어떤 상황에서 사용하나요?

익명함수는 즉시 실행이 필요한 상황에서 사용한다.

02 백엔드

2) 프로그래밍언어(Javascript)

this에 대해서

객체는 상태와 동작을 가진 자료구조다. 동작은 상태를 변경할 수 있어야 한다. 그러려면 메서드는 객체를 참조할 수 있어야 하는데 이러한 자기 참조 변수를 대부분의 프로그래밍 언어에서는 this로 정의한다. 단, 자바스크립트에서 this는 언제나 자기 참조 변수를 의미하지 않는다. 문맥에 따라서 변한다.

1. 일반 함수에서 this : 비엄격 모드에서는 전역 객체로 바인딩된다. 엄격 모드에서는 undefined이다.
2. 생성자 함수에서 this : 미래에 생성될 인스턴스를 의미한다.
3. 메서드에서 this : 자기 참조 변수를 의미한다.
4. call, apply, bind로 호출된 함수에서 this : 첫번째 인자로 넘겨준 객체가 this로 바인딩된다.
5. 이벤트 핸들러에서 this : 이벤트를 위임한 객체(e.currentTarget)로 바인딩된다.

Promise

Promise는 자바스크립트에서 제공하는 비동기를 간편하게 처리할 수 있게 도와주는 객체이다. Promise 이전에 비동기 처리로 콜백 패턴을 주로 사용했었다. 그러나 콜백 지옥(Callback Hell)으로 인해 가독성도 나쁘고, 비동기 처리 중에 발생한 에러의 처리가 까다로웠다. Promise는 이러한 단점을 보완하기 위해 나온 대안이다.

02 백엔드

2) 프로그래밍언어(Javascript)

Callback과 Promise, async/await의 차이점에 대해서 설명해 주세요. (D사 화상 면접)

가장 먼저 나온 Callback은 비동기 처리를 구현하기 위해 만들어 졌다. 이 함수는 다른 함수에게 인자로 전달되어 어느 시점에 실행될 수 있도록 던져주는 함수이다. 하지만 콜백 지옥이라 불리는 중첩문이 발생하면서 에러처리 한계가 생기기 시작했고 이를 해결하기 위해 Promise가 나타났다.

Promise는 어떤 값이 생성 되었을 때 그 값을 대신하는 대리자이다. 비동기 연산이 종료된 이후에 그 결과 값이나 에러를 처리할 수 있도록 처리기를 연결하는 역할을 하는 객체이다. Promise 객체를 통해 성공, 실패, 오류에 따른 후속처리가 바로 가능해서 가독성도 좋고, 비동기 에러를 처리하기도 수월하다.

Async/await은 비동기 코드를 동기식으로 표현하는 더 나은 방법으로 ES2017에 등장하였다.

Async와 await는 항상 같이 붙어 있어야 한다. await 모드는 Promise 객체를 받아 처리하고, 만약 비동기 함수가 아닌 동기적 함수라면 리턴 값을 그대로 받는다. Async 함수는 Promise 객체를 통해 비동기적으로 처리된 내용을 동기적인 코드 진행 순서로 보여주는 역할을 한다.

Javascript Scope Chaining에 대해 설명해 주세요.

실행 컨텍스트 내에서 변수를 탐색할 때 중복되는 변수가 있더라도 먼저 탐색된 변수를 우선적으로 실행시키는 방식이다.

02 백엔드

2) 프로그래밍언어(Javascript)

use strict 은 무엇이고, 사용했을 때 장단점에 관해서 설명해주세요.

use strict는 엄격 모드로 전체 스크립트 또는 부분 함수에 적용이 가능하다. use strict를 사용하게 되면

1. 기존에 무시되던 에러들이 throw 되며
2. JS 엔진 최적화 작업을 어렵게 만드는 실수들을 바로 잡고
3. ECMAScript의 차기 버전에서 정의될 문법들을 금지하는 특성을 가지고 있다.

특히 엄격 모드에서는 이전에 허용되던 실수를 오류로 바꾸어 놓는다. 예를 들어 전역 변수 생성을 불가능하게 만든다든지, 아니면 NaN에 할당하는 것과 같은 일들을 엄격 모드에서는 그냥 넘어가지 않는다. 또한 삭제할 수 없는 프로퍼티를 삭제하려고 하면 예외를 발생시킨다.

AJAX에 관해 가능한 한 자세히 설명하세요. (D사 화상면접)

AJAX(Asynchronous JavaScript and XML)는 비동기 자바스크립트 xml의 약자이다. 쉽게 말하면 클라이언트와 서버가 xml 데이터를 주고 받는 기술이다. 기존에는 클라이언트에서 서버로 요청을 보내고 응답을 받으면 다시 화면을 갱신해야 했고 이 과정에서 많은 리소스가 낭비되었다. 이 문제를 해결하기 위해 ajax는 페이지에서 필요한 일부만 갱신할 수 있도록 XMLHttpRequest 객체를 서버에 요청한다. 이로 인해 자원과 시간을 많이 아낄 수 있다.

02 백엔드

2) 프로그래밍언어(Javascript)

Hoisting

자바스크립트는 소스코드를 실행하기 전에 소스코드 평가를 거쳐 실행 컨텍스트를 생성한다. 이과정에서 변수 선언문과 함수 선언문은 우선적으로 렉시컬 환경에 등록된다. 변수 선언문의 경우 선언과 초기화가 동시에 이루어지며 함수 선언문은 함수 객체로 등록된다. 따라서 소스코드 상에서는 변수가 선언되기 전이지만 프로그래머는 변수에 접근할 수 있다. 이러한 현상을 호이스팅이라 한다. 단, var의 경우에만 초기화가 진행되어 undefined로 접근되며, let 또는 const로 선언한 변수는 호이스팅되지만 호이스팅이 되지 않은 것처럼 동작한다는 점에 유의해야 한다.

클로저 (Closure)

클로저는 함수형 프로그래밍에서 중요하게 사용되는 개념으로 MDN 문서에서는 클로저를 함수와 함수가 선언된 렉시컬 환경의 조합으로 정의하고 있다. 외부 함수가 존재하고 외부 함수에 변수와 중첩 함수를 선언했을때 중첩 함수에서 외부 함수의 변수를 참조하는 상태로 중첩 함수가 반환이 되는 경우 외부 함수의 렉시컬 환경은 중첩 함수에 의해서 유지된다. 이러한 중첩 함수를 클로저라 한다. 클로저는 다음과 같은 상황에서 주로 사용된다.

- 상태 유지
- 정보 은닉
- 전역 변수 억제

02 백엔드

2) 프로그래밍언어(Javascript)

클로저(Closure)는 무엇이며, 어떻게/왜 사용하는지 설명해주세요.

클로저는 독립적인 변수를 가리키는 함수이다. 그리고 클로저 안에 정의된 함수는 만들어진 환경을 기억한다. 클로저를 통해 은닉화를 할 수 있으며, 콜백 함수 등을 사용할 때 발생할 수 있는 에러를 해결하는데도 유용하다.

다음 코드는 클로저에 대한 기본적인 예제이다. 대부분의 프로그래밍 언어에서는 makeFunc() 실행이 끝나면, 즉 displayName이 리턴되고 나면 name 변수에 더 이상 접근할 수 없다고 예상한다. 하지만 자바스크립트에서는 함수를 리턴하고, 리턴하는 함수가 클로저를 선언한다. 이 환경은 클로저가 생성된 시점의 유효 범위 내에 있는 모든 지역변수로 구성된다. 따라서 myFunc은 makeFunc 이 실행될 때 생성된 displayName 함수의 인스턴스에 대한 참조이다. displayName 인스턴스는 변수 name이 있는 어휘적 환경에 대한 참조를 유지한다.

```
function makeFunc() {  
  var name = "Mozilla";  
  function displayName() {  
    alert(name);  
  }  
  return displayName;  
}  
  
var myFunc = makeFunc();  
//myFunc변수에 displayName을 리턴함  
//유효범위의 어휘적 환경을 유지  
myFunc();  
//리턴된 displayName 함수를 실행(name 변수에 접근)
```

02 백엔드

2) 프로그래밍언어(Javascript)

모듈 패턴의 장단점을 말해달라.

모듈 패턴은 전역 영역에서 특정 변수영역을 보호하기 위해 단일 객체 안의 public/private의 변수를 포함할 수 있는 각 클래스 형식의 개념을 구현하는데 사용된다. 이 패턴으로 추가적인 자바스크립트 객체가 다른 스크립트의 객체와 충돌하는 것을 줄여줄 수 있다.

모듈 패턴의 장점

- 점점 더 늘어만 가는 코드를 정리할 때 널리 사용되며 자바스크립트 코딩패턴에서 널리 권장되는 방법이기도 하다.

모듈 패턴의 단점

- 전체적으로 코드량이 약간 더 많아지고 따라서 다운로드 해야 하는 파일크기도 늘어난다.
- 전역 인스턴스가 단 하나뿐이기 때문에 코드의 어느 한 부분이 수정되어도 전역 인스턴스를 수정하게 된다. 즉, 나머지 기능들도 갱신된 상태를 물려받게 된다.

02 백엔드

2) 프로그래밍언어(Javascript)

Before module pattern

- 모듈 패턴을 사용하지 않으면 해당 객체가 모두 전역에서 인스턴스화 되어 다른 코드와 충돌 가능성이 높아진다.
- 객체 간의 연관성을 알기 어려워 코드관리가 어려우며 가용성이 떨어진다.

```
const count = 3
const publicMethod = function () {
  console.log('Public Method : ', count);
}
const publicMethod2 = function () {
  console.log('Public Method2 : ', count);
}
publicMethod();
publicMethod2();
```

After module pattern

- 은닉화, 다형성, 상속을 통해 객체지향적으로 코드를 구성할 수 있다.

```
class Module {
  private count: number = 3
  private privateMethod() {
    console.log('Private Method : ', this.count);
  }
  publicMethod() {
    console.log('Public Method : ', this.count);
  }
  public publicMethod2() {
    console.log('Public Method2 : ', this.count);
  }
}
const mod = new Module();
// mod.privateMethod(); // couldn't access
mod.publicMethod();
mod.publicMethod2();
```

02 백엔드

2) 프로그래밍언어(Javascript)

호스트 객체(Host Objects)와 네이티브 객체(Native Objects)의 차이점은 무엇 인가요?

호스트 객체는 브라우저 환경에서 제공하는 window, XMLHttpRequest, HTMLElement 등의 DOM 노드 객체와 같이 호스트 환경에 정의된 객체를 말한다. 예를 들어 브라우저에서 동작하는 환경과 브라우저 외부에서 동작하는 환경의 자바스크립트(Node.js)는 다른 호스트 객체를 사용할 수 있다.

브라우저에서 동작하는 환경의 호스트 객체는 전역 객체인 window, BOM(Browser Object Model) 과 DOM(Document Object Model) 및 XMLHttpRequest 객체 등을 제공한다.

네이티브 객체는 ECMAScript 명세에 정의된 객체를 말하며, 어플리케이션 전역의 공통 기능을 제공한다. 네이티브 객체는 어플리케이션 환경과 관계없이 언제나 사용할 수 있다. Object, String, Number, Function, Array, RegExp, Date, Math와 같은 객체 생성에 관계가 있는 함수 객체와 메소 드로 구분된다.

JSON이 무엇이며 사용하면 어떠한 장점이 있나요?

JSON은 경량화된 파일 형식이다. 자바스크립트에서 모든 객체는 연관배열(Key/Value)을 기초로 구성되어 있는데, JSON은 이러한 연관 배열을 표시하는데 매우 효과적인 방식이다. JSON은 텍스트 포맷이며 유니코드 인코딩이다. AJAX에서 JSON 방식을 사용하게 되면 eval() 메서드를 사용하지 않아도 키 값을 바로 불러올 수 있다. XML로도 데이터를 불러올 수 있지만, 데이터의 크기를 불필 요하게 증가시킬 수 있으며, 파싱 시에 브라우저 호환성도 신경을 써 주어야 한다는 단점을 가지고 있다.

02 백엔드

2) 프로그래밍언어(Javascript)

전역 scope를 사용했을 때 장단점에 관해 설명해주세요.

자바스크립트에는 스코프 체인(scope chain)이라는 개념이 있는데, 내부 함수에서는 외부 함수의 변수에 접근이 가능하지만 외부 함수에서는 내부 함수의 변수에 접근할 수 없다. 내부 함수에서는 변수를 먼저 해당 스코프에서 찾은 뒤, 만약 없으면 outer 스코프에서 찾고, 그렇게 타고 올라가다가 결국 전역 scope에서 찾는다. 따라서 전역 변수를 만들어서 작업하면 다른 개발자와 협업할 때우연히 같은 변수 이름을 사용해서 이전의 변수를 덮어쓰는 불상사가 발생할 수도 있다. 굳이 전역 변수의 장점을 꼽자면 함수 내의 지역 변수와 달리, 저장된 값이 사라지지 않는다는 점이다.

function foo() {}와 var foo = function() {}에서 foo 의 차이가 무엇인지 설명해 보세요.

전자는 함수 선언(function statement)이며 후자는 함수 표현(function literal)이다. 함수 선언은 코드 블록 자체가 실행 가능 코드가 아니라는 것이다. 해당 코드 블록을 콘솔에서 실행하여도 어떠한 결과가 리턴되지 않는다. Class와 동일한 개념으로 이해해도 된다. 함수 표현은 특정 변수에 할당되거나 즉시 실행가능한 코드 블록으로서 존재하는 함수를 의미하는 것이다. 둘은 호이스팅 관점에서 차이가 있다. 선언식은 호이스팅 되지만 표현식은 호이스팅 되지 않는다.

02 백엔드

2) 프로그래밍언어(Javascript)

JavaScript의 작동방식의 장단점에 관해 설명해주세요.

자바스크립트는 V8 엔진을 사용하며, 싱글 쓰레드 기반이고 콜백 큐를 사용한다. 엔진은 크게 두요소로 구분된다. 메모리 힙(Memory Heap)과 콜 스택(Call Stack). 메모리 힙에서는 메모리 할당이 일어나고, 콜 스택에서는 코드 실행에 따라 호출 스택이 쌓인다. 자바스크립트는 싱글 쓰레드 기반 언어이기 때문에 호출 스택이 하나이고 따라서 그 하나의 호출 스택에 실행된 코드가 하나씩 쌓이게 된다. 싱글 스레드이기 때문에 데드락 같은 문제나 복잡한 시나리오를 고민할 필요는 없으나, 호출 스택에 처리 시간이 어마어마하게 오래 걸리는 함수가 있다면 브라우저는 해당 함수가 실행 되는 동안 아무것도 하지 못하고 가만히 있게 된다.

함수형 프로그래밍에 대해 설명해 주세요.

함수형 프로그래밍은 계산을 수학적 함수의 조합으로 생각하는 방식을 말한다. 이것은 일반적인 프로그래밍 언어에서 함수가 특정 동작을 수행하는 역할을 담당하는 것과는 반대되는 개념으로, 함수를 수행해도 함수 외부의 값이 변경될 수 없다. 함수형 프로그래밍은 순수 함수(pure function)를 조합하고 공유 상태(shared state), 변경 가능한 데이터(mutable data) 및 부작용(side-effects)을 피하여 소프트웨어를 만드는 프로세스다. 함수형 프로그래밍은 명령형(imperative)이 아닌 선언형(declarative)이며 어플리케이션의 상태는 순수함수를 통해 전달된다.

02 백엔드

2) 프로그래밍언어(Javascript)

Callback을 왜 사용하나요?

디자인 패턴 중 하나인 옵저버(Observer) 패턴에서 나온 개념으로써 객체의 상태 변화(이벤트)가 발생 하였을 경우에 이러한 사실을 함수를 통해 전달하게 되는데, 이를 콜백 함수라고 한다. 콜백 함수를 사용하는 이유는 서버로 어떠한 요청을 보낸다고 가정할 때 응답을 기다리는 동안 가만히 시간을 버리는 것이 아닌, 다른 작업을 함으로써 동기적인 코드에 비해 성능을 향상시킬 수 있기 때문이다.

ES6에서 바뀐 문법을 생각나는 대로 이야기 해 주세요. (D사 화상면접)

Destructuring(var { name } = person;), Template Literal(`, 백틱), Spread Operator(...), 화살표 함수 (const a = () => {}), const&let(var와 다르게 block scope), import/export, Map/Set, Promise 등

ES6 이상의 버전을 브라우저에서 인식하지 못한다면 어떻게 해결해야 하는지? (D사 화상면접)

ES6 이상의 자바스크립트 버전은 브라우저 별로 지원률이 상이하기 때문에 트랜스파일러인 바벨을 사용하여 ES6+ → ES5로 변환한다.

02 백엔드

2) 프로그래밍언어(Javascript)

자바스크립트 메서드 .call()과 .apply(), .bind() 차이를 설명하시오. (D사 화상면접)

call과 apply는 함수를 호출하는 함수이다. 그냥 실행하는 것이 아닌 첫 번째 인자에 this로 setting 하고싶은 객체를 넘겨주어 this를 바꾸고 나서 실행한다. call과 apply의 차이점은 첫 번째 인자(this를 대체할 값)를 제외하고, 실제 함수 호출에 필요한 파라미터를 넣어야 한다. call과 다르게 apply 함수는 두 번째 인자부터 모두 배열에 넣어야 한다.

bind 함수가 call과 apply와 다른 점은 함수를 실행하지 않는다는 점이다. 대신 bound 함수를 리턴 한다. 이 bound 함수가 이제부터 this를 obj로 가지고 있기 때문에 나중에 사용해도 된다. bind에 사용하는 나머지 파라미터들은 동일하다.

ES6에서 화살표 함수와 일반 함수의 차이는?

일반 함수는 함수를 선언할 때 this에 바인딩할 객체가 정적으로 결정되는 것이 아니고, 함수를 호출할 때 함수가 어떻게 호출되었는지에 따라 this에 바인딩할 객체가 동적으로 결정된다. 반면에, 화살표 함수는 함수를 선언할 때 this에 바인딩할 객체가 정적으로 결정된다.

02 백엔드

2) 프로그래밍언어(Python)

Generator

제네레이터는 이터레이터를 생성해 주는 함수이다. 이터레이터는 `next()` 함수를 이용해 데이터에 순차적으로 접근이 가능하다. 제네레이터 함수가 실행 중 `yield` 를 만날 경우 함수는 그 상태로 정지되며 반환 값을 `next()` 를 호출한 쪽으로 전달한다. 이후 다시 제네레이터 함수가 실행되면 종료된 시점에서 다음 `yield` 까지 실행된다.

MRO(Method Resolution Order) : 메서드의 결정 순서

인스턴스의 메서드를 실행한다고 가정할 때 `__getattr__()` 로 bound 된 method 를 가져온 후 메서드를 실행한다. 상속할때 왼쪽에 가까운 순서대로 우선순위가 높아진다.

GIL과 그로 인한 성능 문제

GIL(Global Interpreter Lock)은 여러 스레드가 동시에 실행되는 걸 막는다. 덕분에 구현이 간단 하고 레퍼런스 카운팅 오버헤드가 적다. 수행시간에 CPU의 영향이 큰 작업(압축, 정렬, 인코딩)을 멀티 스레드로 수행하면 GIL 로 인하여 싱글 스레드일 때와 별반 차이가 나지 않는다. 따라서 CPU의 영향이 큰 작업(CPU Bound)을 진행할 경우에는 멀티 프로세스를 활용하는 것을 권장하지 않으며 파일 입출력, 네트워크 같은 입출력이 많은 작업(IO Bound)에 멀티 스레드를 사용하는 것이 적합하다.

02 백엔드

2) 프로그래밍언어(Python)

GC 작동 방식

파이썬은 객체를 레퍼런스 카운트를 통해 관리한다. 객체를 참조하는 횟수가 늘어날수록 해당 객체의 레퍼런스 카운트는 증가하고 참조 횟수가 줄어들수록 감소한다. 레퍼런스 카운트가 0이 되면 객체는 메모리에서 해제된다. 단, 레퍼런스 카운트가 0이 아닌 경우에도 자가 참조 혹은 삭제된 객체들이 순환 참조되어 도달할 수 없는 경우에도 메모리에서 해제된다.

가비지 컬렉터는 세대와 임계값을 통해 가비지 컬렉션의 주기를 관리한다. 0세대의 경우 메모리에 객체가 할당된 횟수에서 해제된 횟수를 뺀 값이 threshold 0 을 초과하면 실행된다. 다만 그이후 세대부터는 조금 다른데 0세대 가비지 컬렉션이 일어난 후 0세대 객체를 1세대로 이동시킨 후 카운터를 1 증가시킨다. 이 1세대 카운터가 threshold 1 을 초과하면 그때 1세대 가비지 컬렉션이 일어난다. 2세대도 1세대와 같은 방식으로 동작한다. 즉 1세대의 가비지 컬렉션 은 threshold 0 * threshold 1 주기로, 2세대 가비지 컬렉션은 threshold 0 * threshold 1 * threshold 2 주기로 발생한다.

새로운 객체가 만들어 질 때 파이썬은 `_PyObject_GC_Alloc()` 을 호출한다. 이 메서드는 객체를 메모리에 할당하고, 가비지 컬렉터의 0세대의 객체 할당 횟수를 증가시킨다. 그 다음 이 횟수 가 threshold 0 보다 큰지, `gc.enabled` 가 true 인지, threshold 0 이 0이 아닌지, 가비지 컬렉션이 진행중이 아닌지 확인하고, 모든 조건을 만족하면 `collect_generations()` 를 실행한다. 이때 가장 오래된 2세대 부터 역으로 검사하여 가비지 컬렉션을 진행한다.

02 백엔드

2) 프로그래밍언어(Python)

PyPy가 CPython 보다 빠른 이유

CPython 은 일반적인 인터프리터임에 반해 PyPy 는 JIT(Just In Time)를 겸비한 인터프리터이기 때문이다. JIT 는 인터프리터의 단점을 보완하기 위해 기계어 코드를 생성하면서 그 코드를 캐싱하여, 같은 함수가 여러 번 불릴 때 매번 기계어 코드를 생성하는 것을 방지한다.

Duck Typing

Duck typing이란 특히 동적 타입을 가지는 프로그래밍 언어에서 많이 사용되는 개념으로, 객체의 실제 타입보다는 객체의 변수와 메소드가 그 객체의 적합성을 결정하는 것을 의미한다. 하나의 외부 메서드를 통해서 서로 다른 객체의 인자의 같은 이름의 메서드를 호출할 수 있다.

02 백엔드

2) 프로그래밍언어(Python)

메모리 누수가 발생할 수 있는 경우

해제되지 않은 큰 객체가 메모리에 남아 있는 경우

도메인 컨트롤러가 tombstone lifetime보다 긴 시간 간격 동안 복제할 수 없는 경우 잔여 개체가 발생한다.

그런 다음 도메인 컨트롤러가 replication topology에 다시 연결한다.

도메인 컨트롤러가 오프라인 상태일 때 Active Directory 서비스에서 개체를 삭제하면 해당 개체가 도메인 컨트롤러에 남아 있는 개체로 유지된다. 메모리 누수의 발생으로 이어지는 공간을 소비하는 것은 바로 그 객체이다.

코드의 참조 스타일

참조 스타일에 따라 메모리 누출을 발생시킬지 또는 방지할지가 결정된다.

참조에는 참조 중인 개체에 대한 주소 및 클래스 정보가 있다. 참조를 할당해도 고유한 중복 개체가 생성되지 않는다. 그러나 개체가 더 이상 사용되지 않고 응용 프로그램 내의 다른 위치에서 참조되고 있기 때문에 가비지를 수집할 수 없는 경우에는 메모리 누수가 발생한다.

코드 참조에는 다양한 유형의 참조가 사용되며, 가비지 수집 기능이 다르다.

강한 참조 스타일은 일상 프로그래밍에서 가장 편리하게 사용할 수 있다. 하지만 강한 참조가 붙어 있는 물체는 가비지 컬렉터를 어렵게 만듭니다. 이런 경우 메모리 누수가 발생한다.

백엔드 02

네트워크

프로그래밍언어

데이터베이스

02 백엔드

3) 데이터베이스

데이터베이스를 사용하는 이유

데이터베이스를 사용하면 많은 양의 데이터를 효율적으로 사용할 수 있다. 업데이트 데이터를 쉽고 안정적으로 만들 수 있고 정확도를 보장하는 데도 도움이 된다. 정보에 대한 액세스를 제어하는 보안 기능을 제공하며 중복을 피하는 데 도움이 된다.

인덱스

데이터베이스 인덱스(index)는 추가적인 쓰기 작업과 저장 공간을 활용하여 데이터베이스 테이블에 저장된 데이터의 검색 속도를 향상시키기 위한 자료구조이다.

인덱스는 데이터베이스 내의 특정 컬럼(열)이나 컬럼들의 조합에 대한 값과 해당 값이 저장된 레코드(행)의 위치를 매핑하여 데이터베이스 쿼리의 성능을 최적화하는 데 중요한 역할을 한다.

트랜잭션

트랜잭션이란 데이터베이스의 상태를 변화시키기 위해 수행하는 작업의 논리적 단위

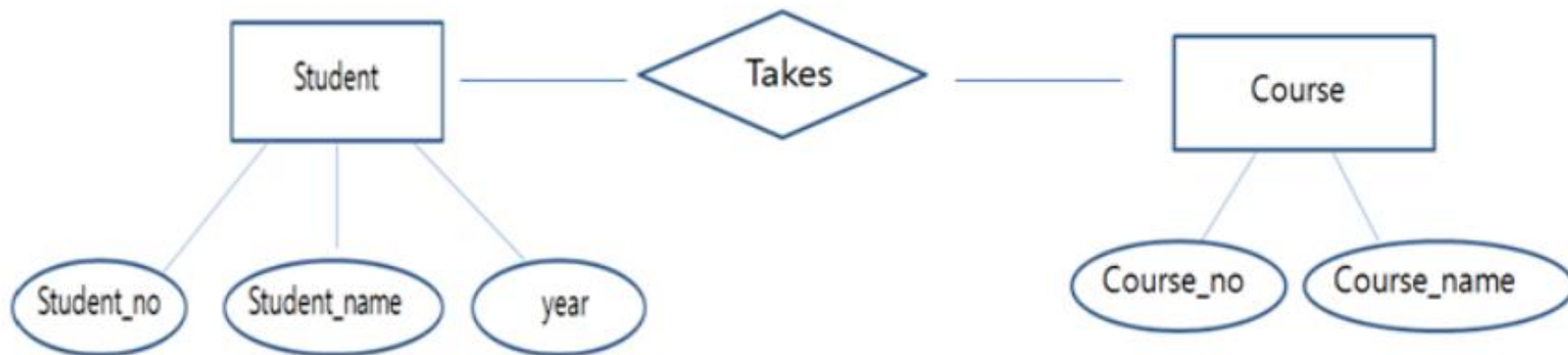
02 백엔드

3) 데이터베이스

PK, FK, ER 모델이란?

- PK (Primary) Key : 테이블에서 각 Row를 유일하게 구분하는 Column Key
- FK (Foreign) Key : 하나의 테이블에 있는 Column으로는 그 의미를 표현할 수 없는 경우 다른 테이블의 Primary-Key Column 의 값을 반드시 참조하여 표현해야 하는 Column
- ER 모델 (Entity Relation Model) : 요구사항으로부터 얻어낸 정보들을 개체 (Entity) 애트리뷰트 (Attribute), 관계성 (Relation) 으로 기술하는 데이터 모델

ex)



02 백엔드

3) 데이터베이스

-개체(Entity)

: 개체란 단독으로 존재하는 객체를 의미하며, 동일한 객체는 존재하지 않는다.

ex) 학생 정보가 학번,이름,학년이 있을 때, 3개의 정보가 모두 같은 학생이 오직 한 명이면 이를 개체라고 한다.

====> 즉, 학생 한명이 개체가 되는 것이다. (=튜플)

(1) 애트리뷰트, 속성(Attribute)

: 개체가 갖는 속성을 의미한다.

ex) Student에서 학번, 이름, 학년 같은 정보를 속성이라 한다.

- ER 다이어그램에서 Attribute는 원으로 표현

(2) 관계 (Relation)

: Entity Type간의 관계를 의미한다.

ex) 수강을 뜻하는 Takes는 학생과 과목간의 "수강"이라는 관계를 갖는다

이 때 Takes를 Relation Type이라고 하며, Relation Type 역시 속성을 가지고 있다.

- ER 다이어그램에서는 Relation은 마름모로 표현한다.

02 백엔드

3) 데이터베이스

참조 무결성이란?

- 기본키와 참조키 간의 관계가 항상 유지됨을 보장한다.
- 참조되는 테이블의 행을 이를 참조하는 참조키가 존재하는 한 삭제될 수 없고, 기본키도 변경할 수 없다.

ex) RESTRICTED(변경 또는 삭제 연산을 취소) , CASCADE(참조하고 있는 개체도 변경 또는 삭제) , SET NULL(참조하고 있는 개체의 값을 NULL로 설정)

RDBMS란?

- 모든 데이터를 2차원 테이블로 표현
- 테이블은 ROW(record , tuple)과 COLUMN(field , item)으로 이루어진 기본 데이터 저장 단위
- 상호관련성을 가진 테이블(table)의 집합
- 만들거나 이용하기도 비교적 쉽지만, 무엇보다 확장이 용이하다는 장점이 있음
- 데이터베이스의 설계도를 ER(Entity Relationship)모델
- ER모델에 따라, 데이터베이스가 만들어지며, 데이터베이스는 하나 이상의 테이블로 구성됨.
- ER모델에서 엔티티를 기반으로 테이블이 만들어짐

02 백엔드

3) 데이터베이스

DB에서의 commit와 rollback 이란?

commit은 쿼리문에서 update, delete, insert를 수행했을 때, 그 쿼리문 수행결과에 대해서 확정을 짓는 것.

Rollback은 사용자가 update, delete, insert등을 실수로 수행했을 때, 데이터를 원복시키기 위해 사용하는 것.

DBMS(DataBase Management System)

데이터베이스를 관리하며 응용 프로그램들이 데이터베이스를 공유하며 사용할 수 있는 환경을 제공하는 소프트웨어. 데이터베이스 내의 정보를 검색하거나 데이터베이스에 정보를 저장하기 편리하고 효율적인 환경을 제공하는 것이 목적이다.

DB정규화란?

자료의 손실이나 불필요한 정보의 도입 없이 데이터의 일관성, 데이터 중복을 최소화하고 최대의 데이터 안정성 확보를 위한 안정적 자료 구조로 변환하기 위해서 하나의 테이블을 둘 이상을 분리하는 작업이다.

DB정규화의 목적?

자료 저장에 필요한 저장 공간을 최소화하고 자료의 삽입, 갱신 및 삭제에 따른 이상 현상을 제거하고 (데이터 무결성 유지), 자료 구조의 안정성 최대화를 위해서다.

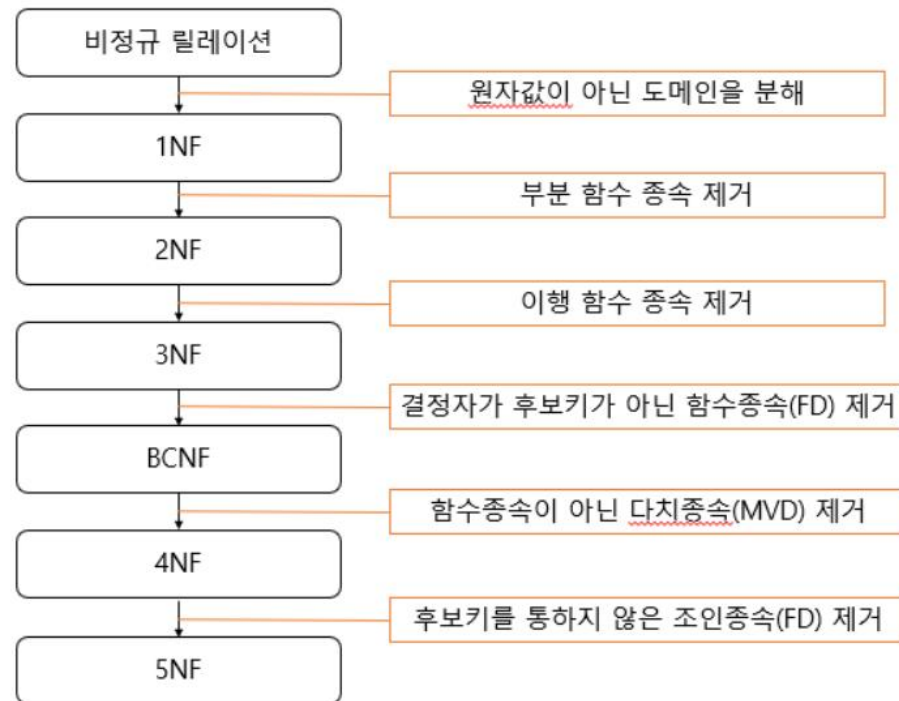
02 백엔드

3) 데이터베이스

데이터베이스의 정규화에 대하여 설명

데이터베이스 정규화는 논리적 데이터베이스 설계에 있어서 테이블들을 구조화 하는 기법 중 하나 이다.

어느 테이블이든 어느 정도는 정규화 될 수 있는데 데이터베이스 이론에서 테이블의 정규화된 정도를 정규형이라는 용어를 써서 표현한다.



02 백엔드

3) 데이터베이스

SQL?

SQL은 구조화 질의어라고 한다. 데이터 정의어(DDL), 데이터 조작어(DML), 데이터 제어어(DCL)를 포함한 데이터베이스용 질의 언어의 일종으로 데이터베이스를 사용할 때 데이터베이스에 접근할 수 있는 데이터베이스 하부 언어를 말한다.

PLSQL이란?

SQL이 확장된 개념으로 DML문과 Query문, 절차형 언어 등을 사용해서 절차적 프로그래밍을 가능하게 하는 트랜잭션 언어이다. 블록구조로 수행속도를 높일 수 있다

Join의 종류

- INNER JOIN(내부조인) : 키 값이 있는 테이블의 컬럼 값을 비교 후 조건에 맞게 값을 가져오는 것이다.
ex) EQUI JOIN
- CROSS JOIN(교차조인, Cartesian Product- 카디션 곱) : 조인되는 두 테이블을 곱집합을 반환한다.
- OUTER JOIN : 조인하는 여러 테이블에서 한 쪽에는 데이터가 있고 한 쪽에는 데이터가 없는 경우, 데이터가 있는 쪽 테이블의 내용을 전부 출력하는 방법이다.
ex) LEFT OUTER JOIN, RIGHT OUTER JOIN

02 백엔드

3) 데이터베이스

Statement vs PreparedStatement

Statement와 PreparedStatement의 아주 큰 차이는 바로 캐시 사용여부이다.

Statement를 사용하면 매번 쿼리를 수행할 때마다 계속적으로 단계를 거치면서 수행하지만 PreparedStatement는 처음 한번만 세 단계를 거친 후 캐시에 담아 재사용을 한다. 만약 동일한 쿼리를 반복적으로 수행한다면 PreparedStatement가 DB에 훨씬 적은 부하를 주며, 성능도 좋다.

NoSQL

NoSQL 데이터베이스는 특정 데이터 모델에 대해 특정 목적에 맞추어 구축되는 데이터베이스로서 현대적인 애플리케이션 구축을 위한 유연한 스키마를 갖추고 있습니다. NoSQL 데이터베이스는 개발의 용이성, 기능성 및 확장성을 널리 인정받고 있습니다.

프론트엔
|니

03

Web Browser

HTML

CSS

03 프론트엔드

1) Web Browser

.....

브라우저의 동작 원리를 간단하게 설명해 주세요.

브라우저의 기본적인 역할은 HTML, CSS 명세에 따라 HTML 파일을 해석해서 표시하는 것이다. 브라우저를 구성하는 요소는 사용자 인터페이스, 브라우저 엔진(크롬, 사파리는 Webkit, 파이어폭스는 Gecko), 렌더링 엔진, 통신, UI 백엔드, 자바스크립트 해석기, 자료 저장소 등이 있다. 렌더링 엔진은 먼저 HTML 문서를 파싱해서 DOM 트리를 구축한다. 그리고 CSS 마크업을 파싱해서 앞서 구축한 DOM 트리과 함께 렌더링 트리를 만든다. 렌더링 트리는 화면에 보여줄 것들만 가지고 있는 트리이므로, 구축이 되면 순차적으로 화면에 배치한다. 부모에서 자식 순서로 배치는 진행된다. 배치가 완료되면 그리기를 시작한다.

CORS가 무엇이며 어떻게 해결을 할 수 있는지 설명해 보세요. (D사 화상면접)

다른 도메인에서 리소스 요청 시 cross-origin HTTP 에 의해 요청을 하는데, 대부분의 브라우저는 보안 상의 이유로 이 요청을 제한한다. 이를 동일 오리진 정책(Same Origin Policy)이라고 한다. 요청을 보내기 위해서는 요청 보내는 대상과 프로토콜이 같아야 하고, 포트도 같아야 한다. JSONP(JSON-padding)을 통해 해결하거나 특정 HTTP 헤더를 추가하여 이 이슈를 해결할 수 있다. 이와 같이 타 도메인 간 자원을 공유할 수 있게 해주는 것을 Cross Origin Resource Sharing, 줄여서 CORS라고 한다.

03 프론트엔드

1) Web Browser

.....

크로스 브라우징이 무엇인가요?

크로스 브라우징은 웹 표준에 따라 서로 다른 OS 또는 플랫폼에 대응하는 것을 말한다. 브라우저별 렌더링 엔진이 다른 상황 등 어떠한 상황 속에서도 문제 없이 동작하게 하는 것을 목표로 한다.

프론트엔드 개발자는 여러가지 전략을 세울 수가 있는데, feature detection(기능 탐지)을 사용해서 해당 기능이 해당 브라우저에 있는지를 확인하는 방법을 사용할 수도 있다. 특히 한 쪽 환경에 최적화를 하는 것 보다, 전체적인 웹 표준을 지키는 데에 노력해야 한다.

서버 사이드 렌더링과 클라이언트 사이드 렌더링의 차이에 대해 설명하십시오.

서버 사이드 렌더링은 전통적인 웹 방식을 의미하며 페이지가 새로고침 될 때마다 서버로부터 리소스를 전달받아 화면에 렌더링 하는 방식이다. 하지만 React, Vue 등의 라이브러리가 등장하면서 훨씬 더 좋은 성능의 SPA 방식의 개발 환경을 선호하기 시작하였다. CSR에서 서버는 단지 JSON 파일을 보내주는 역할만 할 뿐이며, html을 그리는 역할은 클라이언트에서 수행한다. 하지만, CSR 은 자바스크립트가 모든 동작을 수행한 후에 화면에 내용이 나타나므로 초기 구동속도는 SSR에 비해 느리다는 단점이 있다. SEO를 할 수 없고 보안적으로 취약하다는 문제점도 나타난다.

빈 화면에서 렌더링이 완료 되기까지 너무 오래 걸린다는 피드백이 있을 때, 어떻게 하면 이 문제를 해결할 수 있을까요?(단, 캐시는 이미 적용 됨) (V사 필기 시험)

script 파일을 body tag 가장 하단에 위치시키거나 script 태그에 async 속성을 부여한다. 또는 네트워크 리소스 블라킹을 통해 불필요한 무거운 파일들을 제한한다.

프론트엔
|니

03

Web Browser

HTML

CSS

03 프론트엔드

2) HTML

.....

쿠키와 로컬 스토리지, 세션 스토리지의 차이를 설명해 주세요.

기본적으로 쿠키와 로컬 스토리지, 세션 스토리지는 모두 브라우저에서 데이터 저장소의 역할을 하는 것들이다. 웹에서 로그인 하기 위해서는 토큰을 발급받아 API를 호출해야 한다. 하지만 반복되는 작업을 계속 하게 되는 것이 비효율적이고, 이를 보완하기 위해 쿠키를 서버와 클라이언트에 생성해서 토큰 발급 없이 쿠키만 가지고 서버에 요청을 할 수 있게 된다. 쿠키는 저장 공간이 4KB로 작은 편인데 이러한 단점을 보완하여 만든 것이 웹 스토리지이다.

웹 스토리지는 서버에 클라이언트 데이터를 저장하지 않는다. 웹 스토리지에는 로컬 스토리지와 세션 스토리지가 있는데 로컬 스토리지는 브라우저에 정보가 계속해서 남아있는 반면, 세션 스토리지는 해당 세션이 끝나면, 즉 브라우저가 닫히면 데이터가 사라진다. 웹 스토리지는 데스크탑 기준 5~10MB의 저장 공간을 가지고 있어서 쿠키에 비해 훨씬 저장공간이 크다는 장점이 있다. 웹스토리지는 반면 HTML5부터 지원하기 때문에 이전 브라우저에서는 지원이 되지 않는다는 단점이 있다.

프로그레시브 렌더링(Progressive Rendering)이 무엇인가요?

프로그레시브 렌더링은 콘텐츠를 가능한 빨리 표시하기 위해 성능을 향상시키는 기술이다. 인터넷 속도가 느리거나 불안정한 모바일 환경이 아직 많이 남아있기 때문에 이럴 때 유용하게 사용한다.

대표적으로 레이저 로딩이 있다. 이미지를 한 번에 로드하는 것이 아니라, 자바스크립트를 통해 사용자가 표시하려는 부분만 스크롤 시에 이미지를 로드하는 것이다.

03 프론트엔드

2) HTML

.....

다국어 페이지는 어떤 방식으로 제공하나요?

SSR의 경우 HTTP 요청 시 클라이언트에서 Accept-Language 헤더와 같이 기본 언어 설정에 대한 정보를 보낸다. 서버는 이 정보를 사용해 해당 언어로 된 문서 버전을 반환한다. 반환된 문서는 lang 속성을 선언한다. CSR 시 클라이언트 사이드에서 해당 언어 팩(JSON 등)을 가져와 출력한다.

SEO는 무엇인가요?

SEO(search engine optimization, 검색 엔진 최적화)란 웹 페이지 검색엔진이 자료를 수집하고 순위를 매기는 방식에 맞게 웹 페이지를 구성해서 검색 결과의 상위에 나올 수 있게 하는 작업을 말한다. SPA를 개발하는 경우 여러 가지 이점이 있음에도 불구하고 SEO가 잘 되지 않는다는 약점이 있다. 따라서 정보 제공을 목적으로 하는 웹 페이지는 SPA 방식이 불리할 수 있으며, React나 Angular 같은 프레임워크는 서버 렌더링을 통해 SEO에 대응할 수 있는 기술을 지원하므로 선별적으로 사용하면 된다.

<section>과 <article>의 차이는 무엇인가요?

section은 보통 비슷한 특성의 콘텐츠를 담는 구역을 설정할 때 사용한다. 예를 들어, header, footer 사이에 sidebar나 content를 담는 식이다. 반면 article은 관련성이 없고 독립적인 내용들을 담을 때 사용한다. 예를 들어, section 안에서 서로 다른 기사들을 나열해야 할 때 각각의 기사를 article로 담는 식이다.

03 프론트엔드

2) HTML

.....

HTML5 tag를 설명해 주세요. (D사 화상면접)

모든 HTML 문서는 <!DOCTYPE> 선언으로 시작한다. HTML5의 경우 <!DOCTYPE html> 이런 식으로 말이다. 이 선언은 태그는 아니지만 브라우저가 어떤 타입을 받아들여야 할지를 알려주는 정보이다.

여러가지 태그가 있는데 주요한 것들 위주로 살펴보면, HTML5의 필수 태그는 html, head, body 등이 있다. html 태그는 HTML문서의 가장 최상단에 위치하는 태그이며, head 태그에는 style, script, title, link, meta 태그 등이 들어간다. body 태그는 HTML 문서의 내용이 들어간다.

meta 태그에 대해서 조금 더 살펴보면, meta 태그는 head 부분에서 다른 태그들(script, style, link, title 등)로 나타낼 수 없는 메타데이터를 나타내는 태그를 의미한다. <meta name="keywords" content="ABC"> 와 같이 검색 엔진을 위한 키워드나 <meta name="description" content="OWEN">과 같이 문서에 대한 설명 등에 사용된다. 화면에는 별다르게 표시되는 내용이 없지만, 검색 엔진이나 브라우저에서 읽힌다.

Sementic tag에 대해서 설명해 주세요. (D사 화상면접)

시멘틱 태그는 HTML5에 도입이 되었는데, 개발자와 브라우저에게 의미있는 태그를 제공하는 것을 의미한다. 예를 들어 <div> 태그는 non-sementic 태그이고, <table>, <article>은 sementic 태그에 속한다. 시멘틱 태그는 태그만 보고 대략적으로 들어갈 내용을 유추할 수 있다는 장점이 있다. 헤더와 푸터를 설정할 때에도 과거에는 <div id="header"></div> 와 같이 했던 것을 이제는 <header> 하나로 깔끔하게 정리할 수 있다.

프론트엔
|니

03

Web Browser

HTML

CSS

03 프론트엔드

3) CSS

.....

class와 id의 차이점을 설명해 주세요.

id와 class의 차이는 id는 유일한 요소에 적용할 때, 그리고 css는 복수의 요소에 적용할 때 사용한다는 점이다. 하나의 id는 한 문서에서 한 번만 사용이 가능하지만, 하나의 class는 여러 번 사용이 가능하다. 우선순위는 id가 class보다 높다.

float가 어떻게 동작하나요?

Float는 CSS 위치지정 속성입니다. Float된 요소는 페이지의 흐름의 일부가 되며, 페이지의 흐름에서 제거되는 position: absolute 요소와 달리 다른 요소(예: 플로팅 요소 주위로 흐르는 텍스트)의 위치에 영향을 줍니다.

CSS clear 속성은 float 요소에 left/right/both에 위치하도록 사용될 수 있습니다.

부모 요소에 float 요소만 있으면, 그 높이는 무효가 됩니다. 컨테이너의 float 요소 다음에 있지만 컨테이너가 닫히기 전에 float를 clear하면 해결할 수 있습니다.

클리어링(Clearing)에는 어떤 것들이 있으며, 각각은 어떤 때 사용하나요?

float 속성의 영향에서 벗어나기 위해 사용하는 clear 속성은 float의 특성을 지워주는 역할을 한다. 총 4가지 값이 있는데 both는 양쪽의 float 속성을 지워주며, left와 right는 각각 왼쪽, 오른쪽 속성을 지워주고 none은 기본 값으로 아무 것도 지워주지 않는다. 보통은 float 속성을 감싸고 있는 요소들의 height를 조정하기 위해 사용된다.

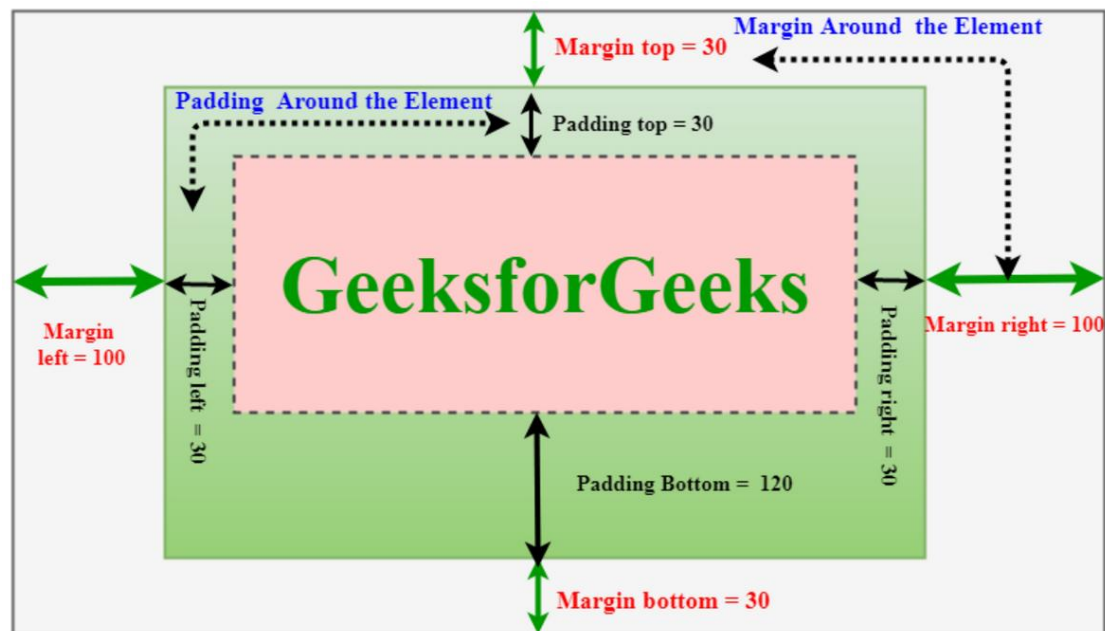
03 프론트엔드

3) CSS

.....

padding과 margin의 차이가 무엇인가요?

margin은 대상의 외부 여백을 의미하고, padding은 대상의 내부 여백을 의미한다.



03 프론트엔드

3) CSS

.....

CSS 전처리기의 장점과 단점은 무엇인가요? (D사 화상면접)

CSS 전처리기를 사용하게 되면 selector를 nesting으로 관리할 수 있고, 조건문이나 반복문, 간단한 연산 등을 할 수 있어서 CSS를 마치 프로그래밍 하듯이 코딩할 수 있다는 장점이 있다. 단점은 웹에서는 CSS만 동작하기 때문에 전처리기는 직접 동작시킬 수가 없다.

따라서 CSS로 컴파일 후 동작시켜야 한다.

페이지에서 표준 폰트가 아닌 폰트 디자인을 사용할 때 어떤 방식으로 처리하나요?

웹 서버에 폰트 파일(.eot, .woff 등)을 올려놓고 사용한다.

CSS selector가 어떠한 원리로 동작하나요?

선택자는 크게 네 가지가 있다. id, class, tag, *(universal). 스타일 엔진은 키 선택터로부터 시작하여 왼쪽으로 이동하면서 엘리먼트가 규칙에 적합한지 확인한다. 만약 엘리먼트가 이 규칙에 적합하거나 적합하지 않다는게 확인되면 멈춘다.

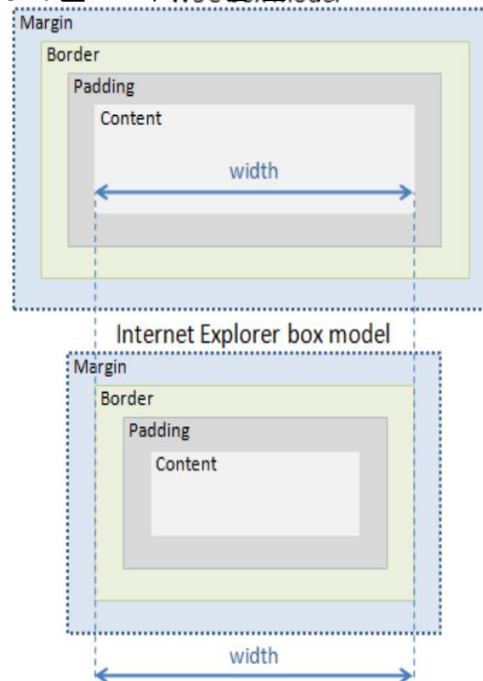
03 프론트엔드

3) CSS

.....

box model이 무엇이며, 브라우저에서 어떻게 동작하는지 설명해 주세요

box model은 각각의 Object를 박스 형태로 나타내어 브라우저에 배치하기 위한 규칙이다. W3C 박스 모델과 IE 박스 모델이 있는데 두 가지 박스 모델은 차이가 있다. W3C 박스 모델은 content-box로 width가 content만 포함하는 반면, IE 박스 모델은 border-box로 width에 content, padding, border를 모두 포함한다.



03 프론트엔드

3) CSS

.....

display 속성에 어떤 것들이 있는지 설명해 주세요.

display 속성에는 block, inline, inline-block, none이 있다

- block : 항상 새로운 라인에 요소가 시작되고 화면 크기의 전체 가로폭을 영역으로 차지한다. width 속성 값을 부여해주면 그 너비 만큼 영역을 차지한다.
- inline : 새로운 라인에서 시작되지 않으며 다른 요소들과 같은 줄에 배치될 수 있고 content 너비만큼의 영역을 차지한다. 그리고 width, height, margin-top, margin-bottom 속성이 적용이 되지 않는다.
- inline-block: block 레벨 요소와 inline 레벨 요소의 특징을 모두 가지고 있다. 한 줄에서 inline 레벨 요소들과 같이 배치될 수 있으며 width와 height 속성으로 영역의 크기를 지정할 수 있다.
- none: 선택한 요소들을 화면에 나타나지 않게 한다. visibility: hidden과의 차이점은 영역이 남아있는지 여부가 다르다는 점이다.

(display: none은 영역도 없음)

03 프론트엔드

3) CSS

.....

요소를 배치하는 방법(static, relative, fixed, absolute) 간의 차이는 무엇인가요?

- static : 기본값으로 요소들이 겹치지 않고 상→하로 배치된다.
- relative : 원래 배치되어야 할 위치에서 지정한 값 만큼 떨어진 곳에 요소를 배치한다.
- fixed : 웹 브라우저 화면 전체를 기준으로 배치한다. 스크롤을 하더라도 위치가 고정된다.
- absolute : 가장 가까운 상위 요소의 위치를 기준으로 지정한 값 만큼 떨어진 곳에 요소를 배치한다.

CSS 애니메이션과 JS 애니메이션의 차이에 대해서 설명해 주세요.

- CSS

UI 요소에 대해 더 작은 자체 포함 상태가 있는 경우 사용하는 것이 좋다.

낮은 버전의 브라우저에서 지원을 안 하는 경우가 있다.

- JS

애니메이션을 세밀하게 제어해야 하는 경우 JS를 사용한다.

크로스 브라우징 측면에서 JS 애니메이션을 사용하는 것이 유리하다.

velocity.js와 같은 라이브러리를 사용하면 CSS보다 성능이 좋다.

03 프론트엔드

3) CSS

.....

flex를 사용하는 이유가 무엇인가요? (D사 화상면접)

flex는 레이아웃을 좀 더 편하게 잡기 위해서 만들어진 css 속성이다. flex를 사용하면 요소들의 크기나 위치를 쉽게 잡을 수 있다. 기존에 수평 구조를 만들 때 사용하는 속성이 float나 inline-block 등이 있었는데 여러가지 문제를 가지고 있었고 flex를 사용하면 이러한 속성의 한계를 보완할 수 있다. 물론 수평 뿐만 아니라 수직도 가능하다.

flex는 컨테이너와 아이템 개념을 사용하여 요소의 크기가 불분명하거나 동적인 경우에도 요소를 효율적으로 정렬할 수 있게 해준다.

CSS-in-JS에 대해서 설명해 주세요.

CSS-in-JS는 CSS보다 더 강력한 추상화이다. JS를 사용하여 스타일을 선언적이고, 유지보수 가능한 방식으로 설명한다. JS를 CSS로 전환하는 고성능 컴파일러로, 런타임 및 서버 사이드에서 작동한다. 인라인 스타일과는 다른 개념이다.

03 프론트엔드

3) CSS

.....

인라인 스타일

한 줄짜리 짤막한 스타일, 태그 안에 직접 지정하여 사용. HTML과 섞어서 사용

```
const textStyles = {  
  color: white,  
  backgroundColor: black  
}  
  
<p style={textStyles}>inline style!</p>
```

브라우저에서 DOM 노드를 다음과 같이 연결합니다

```
<p style="color: white; background-color: black;">inline style!</p>
```

03 프론트엔드

3) CSS

.....

CSS-in-JS

- 자바스크립트 파일 안에서 CSS를 작성할 수 있는 방법이다.
- 자바스크립트의 상태 값을 공유하여 동적으로 스타일링을 하기 위해 등장한 패러다임이다.

```
import styled from 'styled-components';
```

```
const Text = styled.div`  
  color: white,  
  background: black  
`
```

```
<Text>Hello CSS-in-JS</Text>
```

브라우저에서 DOM 노드를 다음과 같이 연결합니다

```
<style>  
.hash136s21 {  
  background-color: black;  
  color: white;  
}  
</style>
```

```
<p class="hash136s21">Hello CSS-in-JS</p>
```

03 프론트엔드

3) CSS

.....

CSS-in-JS 장점

- 컴포넌트로 생각하기—더이상 스타일시트의 묶음을 유지보수 할 필요가 없습니다. CSS-in-JS는 CSS 모델을 문서 레벨이 아니라 컴포넌트 레벨로 추상화합니다(모듈성).
- CSS-in-JS는 JavaScript 환경을 최대한 활용하여 CSS를 향상시킵니다.
- "진정한 분리 법칙"—스코프가 있는 선택자로는 충분하지 않습니다. CSS에는 명시적으로 정의 하지 않은 경우, 부모 요소에서 자동으로 상속되는 속성이 있습니다. [jss-isolate](#) 플러그인 덕분에 JSS 규칙은 부모 요소의 속성을 상속하지 않습니다.
- 스코프가 있는 선택자—CSS는 하나의 전역 네임스페이스만 있습니다. 복잡한 애플리케이션 내에서 선택자 충돌을 피할 수 없습니다. BEM과 같은 네이밍 컨벤션은 한 프로젝트 내에서는 도움이 되지만, 서드파티 코드를 통합할 때는 도움이 되지 않습니다. JSS는 JSON으로 표현된 것을 CSS로 컴파일 할 때, 기본적으로 고유한 이름을 생성합니다.
- 벤더 프리픽스—생성된 CSS 규칙은 자동적으로 벤더 프리픽스가 붙어있으므로 생각할 필요가 없습니다.
- 코드 공유—JavaScript와 CSS사이에 상수와 함수를 쉽게 공유할 수 있습니다.
- 현재 화면에 사용중인 스타일만 DOM에 있습니다([react-jss](#)).
- [작은 코드 제거](#)
- CSS 유닛 테스트!



기타 질문 04

인성질문

손코딩 테스트

04 기타질문

1) 인성질문

.....

1. 왜 전 회사 그만두고 개발하러 오셨나요?
2. 퇴사한 것을 후회하지 않으시나요?
3. 지원한 이유는 무엇인지?
4. 개발이 왜 좋은지?
5. 개발을 왜 시작했는지?
6. 어떤 점에서 개발자가 되고 싶은지?
7. 미래에 어떤 개발자가 되고 싶은지?
8. 전공 과목중에 좋아하는 과목이랑 싫어하는 과목 각각
9. 어떤 서비스를 하고 싶은지?
10. 만약 떨어진다면 어떤 기업에 또 지원할 것인지?
11. 5년, 10년 뒤에는 어떤 엔지니어가 되어있을지?
12. 주변 친구들 중 한명 칭찬해주세요
 - 1) 칭찬 한개만 더해주세요
13. 지원하는 회사의 장점은?
14. 회사에 궁금하신점 있으신가요?
15. 본인의 강점, 단점은?
16. 기술적인 이슈를 해결했던 경험을 이야기해주세요.
17. 입사한 팀이 코드리뷰, 클린코드, 테스트 아무것도 안하면 어떻게 하실 건가요?
18. 열심히 하지만 본인이 원하는 만큼 따라오지 못하는 팀 동료들이 있을 수 있는데, 이럴 때는 어떻게 하는 지?

04 기타질문

1) 인성질문

.....

19. 본인이 팀 프로젝트에서 협업 측면에서 맡았던 역할을 이야기 해주세요.
20. 팀 프로젝트에서 아쉬웠던 점 이야기 해주세요.
21. 전공을 하셨는데, 전공 관련해서 최근에 부족하다고 느껴서 공부하고 있는 분야가 있는지?
22. 최근에 재미있어 하는 일은?
23. 시간 남을 때 하는 일은? (개발 말고..)
24. 토이프로젝트 하고 있다고 하신 거 어떤 거 하시는 지 말씀해주세요.
25. 입사해서 원하는 선배의 모습이 무엇인가요?
26. 본인이 속한 그룹에서 몇 등 정도 하는 것 같은지?
 - 1) 그러면 본인보다 잘하는 동료들은 어떤 점에서 더 잘하는 것 같은지?
 - 2) 더 못한다고 한 사람들은 어떤 점에서 더 못하는지?
27. 사내 스터디에서 어떤 것을 공부하고 싶은지?
28. 지금 제일 가고 싶은 회사는?
29. 특별히 가고 싶은 회사는? 특정 도메인인지? 회사의 분위기인지? 어떤 점을 중점적으로 보는지?
30. 좋아하는 앱이 있는지?
31. 그 앱에 추가하고 싶은 기능이 있는지?
32. 본인이 생각하기에 좋은 개발 문화라는 것은 어떤 것인지?
33. 팀 프로젝트에서 힘들었던 팀원이 있다면? 어떤 점에서 힘들었는지?
 - 1) 어떻게 해결했는지?
34. 페어 프로그래밍은 어떤 식으로 진행했는지?

04 기타질문

1) 인성질문

.....

35. 사람들과 협업하면서 가장 어려웠던 점이 있다면?
36. 전에 했던 프로젝트가 어떤 프로젝트였고, 초기에 어떤 계획을 했고 어느정도 달성했는지, 맡은 역할은 무엇이었는지?
37. 프로젝트에서 기간이 얼마 남지 않았을 때 무엇에 집중하는지?
38. 스터디 하고 계시는 것 있는지?
39. 문서화는 왜 해야한다고 생각하시나요?
40. 다른 사람이 생각하기에 본인의 장, 단점은?
 - 1) 그 단점을 극복하기 위해 노력했던 것은?
41. 지금 읽고 계신 책은 무엇인가요?
42. 지금 공부하고 계신 분야는 무엇인가요?
43. 좋은 개발문화는 무엇인가요?
44. 그러면 좋은 사람, 좋은 개발자는 어떤 사람인가요?
45. 좋은 개발자의 요소를 세 가지 정도만 말씀 해주세요.
46. 회사는 교육기관이 아닙니다. 당신이 회사에서 실력을 키우고 배울 수 있고 함께 성장할 동료들을 얻을 수 있다면, 당신이 회사에 기여할 수 있는 것은 어떤 것이 있죠?
47. 페어 프로그래밍의 장점은?
48. 회사의 입장에서 페어 프로그래밍의 장점은?
49. 못하는 개발자 두명이 페어 프로그래밍을 한다면 오히려 능률이 떨어지는 것이 아닌지?
50. 협업을 하면서 가장 중요하게 생각하는 것은?

04 기타질문

1) 인성질문

.....

51. 소통 능력을 키우기 위해서 어떤 일을 할 수 있을까?
52. 실무에서 개발자 외의 사람들과 소통할 경우에는 어떻게 해야할까?
53. 새로운 언어를 배울 때 어느정도 시간이 걸리시나요?
54. 개인적으로 공부를 어떻게 하시는지 말씀해주세요. (강의, 책 등)
55. 리팩토링에 관한 책 읽어보셨나요?
56. 회사와 본인을 어떻게 연관짓나요?
57. 우리 회사에 기대하고 있는 것이 있나요?
58. 그러면 우리가 지원자분께 기대할 수 있는 것은 무엇이 있나요?
59. 활동하고 있는 IT 커뮤니티 있나요?
60. 최근 관심을 가지고 있는 IT 분야는?
61. 본인은 팀장 성향인지, 팀원 성향인지?
 - 1) 그럼 예를 들어 친구들끼리 여행가자! 해서 각자 의견을 물을 때 꼭 필요하지 않다면 굳이 의견을 내지 않는 스타일 이신가요?
62. 팀원과 갈등이 발생했을 때 어떻게 해결하는지?
 - 1) 설득을 하려고 해도 끝까지 일치하지 않을 수 있는데 이럴 땐 어떻게 하는지?
 - 2) 프로젝트의 기한도 길지 않아서 두 방식을 모두 해볼 수 없다고 가정하면 어떻게 해야할 지?
63. Frontend Framework를 사용하지 않고 바닐라 자바스크립트를 사용한 이유는 무엇인가요?
64. (어떤 프로젝트에 대한 설명을 하고 나서) 그렇게하기로 결정한 이유는 무엇인가요?
65. (스프링 시큐리티에 관심이 있다고 하고 난 후) 스프링 시큐리티에 가장 최근에 올라온 이슈는 무엇인지?

04 기타질문

1) 인성질문

.....

- 66. 전 직장에서도 업무가 맞지 않아서 퇴사하셨다고 하셨는데, 그러면 여기서도 업무가 맞지 않으면 동일하게 하실 것인지?
- 67. 전 직장에서 퇴사하신 것을 후회하지 않으십니까?
- 68. OAuth에 대해서는 왜 공부하셨나요? (XX 에 대해서는 왜 공부하셨나요?)
- 69. 입사 후 쉬운 일, 어려운 일이 있을 때 이 일에 대한 분배를 어떻게 하실 건가요?
- 70. 만약 떨어진다면 어떤 점이 부족해서 떨어졌다고 생각할지?
- 71. 공백기간동안 뭘 하면서 보내셨나요?
- 72. 인생을 살면서 무엇인가 하나에 푹 빠져본 적이 있으신가요?
- 73. 인생에서 가장 후회되는 것이 있다면 어떤 것인가요?
- 74. 가장 자신있는 기술 분야는 어떤 분야입니까?
- 75. 퇴사 후에 친구들을 만나거나 하는 것이 부담스러워졌는지?
- 76. 협업하는 과제, 혼자하는 과제 중 어떤 것을 더 선호하는지? 그 이유는?
- 77. 본인의 의견을 주장할 때 강하게 주장하는 편인가요?
 - 1) 만약 본인의 주장이 본인과 협업하는 사람(예를 들어 프론트 개발자, 기획자 등)이 받아들이지 못한다면 어떻게 해결하실 건가요?
- 78. 입사 후에 저희 회사에서 어떤 일을 하고 싶으신가요? (무엇을 기대하고 계신가요?)
- 79. 저희 회사 서비스 중 어떤 서비스가 추가되었는데, 여기에 또 다른 기능을 추가한다면 어떤 기능이 있을까요?

기타 질문 04

인성질문

손코딩 테스트

04 기타질문

2) 손코딩 테스트

Stack에서 계속해서 push/pop 되는 상황에서
O(1)로 최소값을 찾는 스택 구현

```
class Stack:
    def __init__(self):
        self.container = list()

        # 최소값을 담아두기 위한 컨테이너
        self.min_container = list()

    def push(self, data):
        self.container.append(data)

        # 최소값 갱신
        if not self.min_container or data < self.min_container[-1]:
            self.min_container.append(data)

    def pop(self):
        pop_data = self.container.pop()

        # 삭제되는 데이터가 최소값이라면 최소값을 갱신
        if pop_data == self.min_container[-1]:
            self.min_container.pop()

        return pop_data

    def _min(self):
        if not self.min_container:
            return None

        return self.min_container[-1]
```

입력 문자열을 거꾸로 출력하는 코드를 짜주세요.

```
1 public class StringReverse {
2     public static void main(String[] args) {
3
4         // 문자열
5         String str = "ABCDE";
6
7         // 문자열 reverse
8         String reverse = "";
9         for (int i = str.length() - 1; i >= 0; i--) {
10             reverse = reverse + str.charAt(i);
11         }
12
13         // 결과 출력
14         System.out.println(reverse); // "EDCBA"
15
16     }
17 }
```

04 기타질문

2) 손코딩 테스트

.....

Array에서 index와 값이 일치하는 index 찾기(정렬된 배열)

1) 예시로 -10, -4, -3, -1, 0, 1, 4, 7, 10, 13, 15 이런식이면 7이 답

```
public class FindIndex {  
    public static void main(String[] args) {  
        int[] arr = { -10, -4, -3, -1, 0, 1, 4, 7, 10, 13, 15 };  
        for (int i = 0; i < arr.length; i++) {  
            if (i == arr[i]) { // 인덱스 값과 배열값이 같으면 그때의 인덱스를 출력  
                System.out.println(i);  
            }  
        }  
    }  
}
```


04 기타질문

2) 손코딩 테스트

큐로 스택 구현하기

```
class Stack
{
    queue<int> a, b;
    void push (int n)
    {
        if(a.empty())
            a.push(n);

        else {
            while(!a.empty())
            {
                b.push(a.front());
                a.pop();
            }
            a.push(n);
            while(!b.empty())
            {
                a.push(b.front());
                b.pop();
            }
        }
    }

    int pop()
    {
        int front = a.front();
        a.pop();
        return front;
    }
}
```

스택으로 큐 구현하기

```
class Queue
{
    stack<int> a, b;
    void push (int n)
    {
        a.push(n);
    }

    int pop()
    {
        if(b.empty())
            while(!a.empty())
            {
                b.push(a.top());
                a.pop();
            }

        int top = b.top();
        b.pop();
        return top;
    }
}
```

최대 공약수와 최소 공배수를 구하는 알고리즘

```
int gcd(int a, int b)
{
    if(b == 0) return a;
    return gcd(b, a % b);
}

int lcm(int a, int b)
{
    return a * b / gcd(a, b);
}
```