프론트엔드 면접문제\_강현주\_20200704

1. Http 통신과 소켓 통신은 어떻게 틀린가.

* **Http 통신** : Client의 요청(Request)이 있을 때만 서버가 응답(Response)하여 해당 정보를 전송하고 곧바로 연결을 종료하는 방식이다. 이러한 연결 방식은 Client가 요청을 보내는 경우에만 Server가 응답하는 단방향적 통신으로, Server가 Client로 요청을 보낼수는 없다. 예를들어 네이버 같은 사이트의 링크를 클릭한 순간에 Client(웹)는 Server(웹서버)로 네이버에 대한 내용을 보내달라고 요청을 한 것이다. 네이버에 대한 내용들을 받은 후에는 바로 연결이 종료된다. 그렇기 때문에 요청을 보낼 때에는 내용을 기다리는 시간과 함께 연결하는 시간이 들어가게 된다. 이러한 Http 통신은 실시간 연결이 아닌, 필요한 경우에만 Server로 접근하는 콘텐츠 위주의 데이터를 사용할 때 용이하며 일반적으로 모바일 어플리케이션은 필요한 경우에만 Server로 정보를 요청하는 경우가 많은데, 이러한 Web Server로 Http 통신을 주로 사용하고 비용 및 유지보수 등 대부분의 방면에서 좋다. 만약 게시물에 대한 내용을 요청하기 위해 실시간으로 연결을 유지하는 Socket 통신을 사용하게 되면, 게시물을 받은 후에도 계속 통신을 위한 연결이 성립되어 있어 부하가 걸리게 된다.
* Http 통신의 특징)

1. Client가 요청을 보내는 경우에만 Server가 응답하는 단방향 통신이다.
2. Server로부터 응답을 받은 후에는 연결이 바로 종료된다.
3. 실시간 연결이 아니고, 필요한 경우에만 Server로 요청을 보내는 상황에 유용하다.
4. 요청을 보내 Server의 응답을 기다리는 어플리케이션(Android or Ios)의 개발에 주로 사용된다

* **Socket 통신** : Server와 Client가 특정 Port를 통해 실시간으로 양방향 통신을 하는 방식이다. Http 통신과 달리 Server와 Client가 특정 Port를 통해 연결을 성립하고 있어 실시간으로 양방향 통신을 한다. Client만 필요한 경우에 요청을 보내는 Http 통신과 달리 Socket 통신은 Server 역시 Client로 요청을 보낼 수 있으며, 계속 연결을 유지하는 연결지향형 통신이기 때문에 실시간 통신이 필요한 경우에 자주 사용된다. 예를 들면, 실시간 Streaming 중계나 실시간 채팅과 같이 즉각적으로 정보를 주고받는 경우에 사용하는데, 실시간 동영상 Streaming 서비스를 Http 통신으로 구현하였다고 가정했을 때, 사용자가 서버로 동영상을 요청하기 위해서는 동영상이 종료되는 순간까지 계속해서 Http 통신을 보내야 하고 이러한 구조는 계속 연결을 요청하기 때문에 부하가 걸리게 된다. 그러므로 이러한 경우에는 Socket을 통해 구현하는 것이 적합하다.
  + 소켓 통신의 특징)

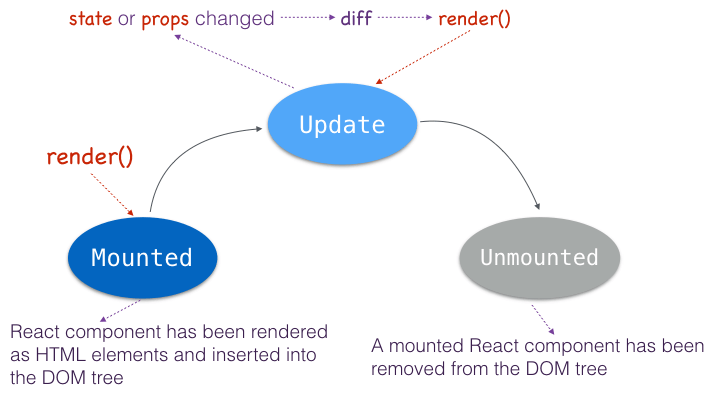
1. Server와 Client가 계속 연결을 유지하는 양방향 통신이다.
2. Server와 Client가 실시간으로 데이터를 주고받는 상황이 필요한 경우에 사용된다.
3. 실시간 동영상 Streaming이나 온라인 게임 등과 같은 경우에 자주 사용된다.

출처 :

1. <https://mangkyu.tistory.com/48>

2. 컴포넌트의 각 생명주기는 어떻게 되고 각 시점은 언제 호출되는가?

* 컴포넌트 생명주기는 3가지의 시점(마운트, 업데이트, 언마운트)으로 나뉜다.



1. **마운트** : 리액트 컴포넌트가 Instance로 생성되어 DOM tree에 삽입되어 브라우저 상에 나타나는것을 마운트라고 한다.

마운트시 호출하는 메서드는 컴포넌트를 새로 만들 때 마다 호출되는 클래스 생성자 메서드 constructor, props에 있는 값을 state에 동기화하는 static getDerivedStateFromProps 메서드, 클래스 컴포넌트에서 구현하는 메서드로 UI를 렌더링 하는 render, 컴포넌트가 DOM tree에 삽입된 직후에 호출되는 메서드인 componentDidMount 가 있다.

1. **업데이트** : 리액트 컴포넌트가 업데이트 되는 것을 업데이트 라고 한다. 컴포넌트 업데이트 경우는 다음의 4가지이다. 1. props가 바뀔 때 2. state가 바뀔 때 3. 부모 컴포넌트가 리렌더링 될 때 4. 강제로 트리거를 발생시킬 때 ( this.forceUpdate) 이다.

업데이트 시 호출하는 메서드로는 static getDerivedStateFromProps, 컴포넌트가 다시렌더링을 해야 할지 말아야 할지 결정하는 shouldComponentUpdate, render, render메서드 호출후 DOM 변화를 반영하기 직전에 호출되는 getSnapshotBeforeUpdate, 리렌더링을 완료한 후 실행되는 componentDidUpdate가 있다.

1. **언마운트** : 리액트 컴포넌트가 DOM상에서 제거 되는 것을 언마운트라고 한다.

언마운트시 호출하는 메서드로는 컴포넌트가 DOM에서 제거되기 직전에 호출되는 componentWillUnmount가 있다.

출처 :

1. <https://medium.com/humanscape-tech/react-%EC%BB%B4%ED%8F%AC%EB%84%8C%ED%8A%B8-%EC%83%9D%EB%AA%85%EC%A3%BC%EA%B8%B0-c7f45ef2d0be>

3. 자바스크립트에서 사용하는 자료구조에는 어떤 것들이 있는가?

* **자료구조**는 데이터의 표현 및 저장 방법을 의미한다. 알고리즘은 그 저장된 데이터를 처리하는 과정이다. 자료구조는 적은 수의 데이터를 관리하기 위해 고안된 것이 아니다. 가령 1개의 데이터를 처리하는데 1초가 걸리는 크기 1의 구조가 있다고 하면, 이 구조를 이용해 1000개의 데이터를 처리하려면 1000의 크기와 1000초가 필요하게 된다. 이 때, 자료구조는 1개의 데이터를 처리하는 시간과 크기를 줄여 데이터를 처리하는데 더 적은 크기와 더 적은 시간이 필요하게끔 하는 역할을 한다. 따라서 자료구조가 중요한 이유는 프로그램 내에서 자료를 가장 효율적으로 처리하기 위함이고, 그것을 통해 보다 성능 좋은 프로그램을 만들기 위함이다.
* **자료구조의 종류**

1. **배열 (Array) :** 배열은 여러 데이터를 그룹핑해서 관리하기 위한 자료구조로써 배열 안에 여러 정보를 담을 수 있고, 반복문과 결합하면 정보를 더 효율적으로 관리할 수 있다. 또 크기가 정해져 있으며 인덱스로 식별 /조회가 가능하다.
2. **리스트 (List)** : 리스트는 배열이 갖는 인덱스 구조의 장점을 버린 대신 각 데이터의 빈틈을 없애는 방법을 선택했다. 따라서 데이터의 삽입과 삭제에 대한 데이터 낭비 줄고, 검색 시간이 길어졌다는 특징이 있다. 리스트에는 Array List와 Linked List가 존재한다. 둘의 큰 차이는 '데이터 연결구조' 다. Array Lists는 배열을 이용해 Javascript의 배열 구조처럼 리스트를 구현한 것을 말한다.

* Array List
* 내부적으로 배열을 사용하기 때문에 인덱스를 이용해 데이터에 접근이 가능하다.
* 데이터 조회 속도 증가 / 데이터를 추가 하거나 삭제하게 되면 각 순서가 일정하게 변경 되어야하기 때문에 데이터의 삽입과 삭제에 상대적으로 오랜시간 소요된다.

* Linked List
* 배열을 사용하지 않고, 하나의 데이터에 다음 엘리먼트의 위치정보를 포함한다.
* 특정 데이터를 조회하는 인덱스가 존재하지 않기 때문에 조회하는 속도가 느림.
* 데이터 추가 or 삭제시 다른 데이터에 영향을 주지 않기 때문에 상대적으로 빠름

두 List 구조는 장단점을 가지고 있다. 일반적으로 고정된 데이터의 검색이 필요하면 Array List를 사용하며, 검색이 필요없는 가변적인 데이터가 필요한 경우에는 Linked List를 사용한다.

1. **큐 (Queue) :** '줄', '줄을 서서 기다리다'라는 뜻을 가지고 있다. 일상생활에 비유해서 많이 설명되고 있다. '은행창구에서 번호표를 뽑아 기다리기', '신호를 기다리는 차들' 등등 이 예들은 일종의 큐 구조에 해당된다고 할 수 있다. 이러한 큐 구조에 공통적으로 적용되는 특징이 있는데 ‘FIFO & LILO’ 즉, 먼저 들어간 사람이 먼저 나오고 나중에 들어간 사람이 나중에 나온다. 이러한 큐 구조는 컴퓨터 과학 전반에 자주 쓰이는 자료구조이다. 가장 대표적인 예로 '버퍼(Buffer)'를 들 수 있다. 일반적인 큐는 선형이지만 크기가 제한 되어 있고 빈 공간을 사용하려면 모든 자료를 꺼내거나 자료를 한 칸씩 옮겨야 한다는 단점이 있기 때문에 순환큐를 구현해 선형 큐의 문제점을 보완할수도 있다.
2. **스택 (Stack)** : 함수의 호출을 기록하는 자료구조. 기본적으로 프로그램 안에서 위치한 곳이다. 만약 어떤 함수를 실행시킨다면, 우리는 스택 위에 무언가를 올리는(push) 행위를 한 것이고 우리가 함수로 부터 반환을 받을 때, 스택의 맨 위를 가져오는(pop) 것이다. 흔히 스택을 쌓는다고 이야기 하는 것처럼 스택은 하나의 바구니에 데이터들이 순차적으로 담겨져있는 형태를 가지며 LIFO (Last In First Out)으로 데이터가 쌓이게 된다. ex) 웹 브라우저의 '앞으로가기' '뒤로가기'
3. **그래프 (Graph)** : 그래프는 상,하위의 개념이 없이 각각의 node와 그 node들 간의 간선(edge)을 하나로 모아 놓은 자료구조이다. 꼭 모든 node들이 서로 관계를 갖고 있어야만 하지 않다. 따라서 그래프는 node간 연결이 없는 고립된 부분이 있을 수도 있고, 순환할 수도 있고, 안할 수도 있기 때문에 가장 형식에 얽매이지 않은 자료구조라고 볼 수 있다. 이러한 특성 때문에 그래프 자료구조를 네트워크 모델 이라고 부른다.

* node (노드) : 위치라는 개념. (vertex 라고도 부름)
* edge (간선) : 위치 간의 관계. 즉, 노드를 연결하는 선 (link, branch 라고도 부름)

1. **트리 (Tree)** : 트리란 쉽게 말해 가계도이다. 나무 뿌리를 생각하면 쉬운데, 트리의 최상위 노드인 root Node와 다수의 subtree Node 들의 집합이다. root와 subtree의 관계를 parent(부모노드)와 child(자식노드) 관계로 표현할 수 있고, 각각의 subtree들은 자신의 휘하에 또 다른 subtree를 두어 parent, child 관계를 가질 수 있다. 검색에 활용하기 위해 이진 탐색 트리(Binary Search Tree)를 많이 사용한다.

* 이진 탐색 트리 (Binary Search Tree) : Binary Tree는 한 개의 Parent node가 최대 2개의 Child node를 갖는 Tree를 말한다. B-tree의 구성은 root 와 left, right로 구성 되어있다. 찾고자 하는 값(value)을 입력하면 value와 root를 비교한 후, 작으면 left를 크면 right를 root로 변환 하고 변경된 root를 기준으로 다시 비교하여 left 혹은 right로 이동하는 방식의 Tree가 BST이다. 더 이상 비교할 Child Node(left, right)가 존재하지 않을 때 해당 위치에 data를 입력한다.

출처 :

1. <https://velog.io/@rockjeon/%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0>