React 개요

JS

▶ JS (JavaScript)

과거 : 간단한 연산을 하거나 시작적 효과를 주는 단순한 스크립트 언어

현재 : 웹 애플리케이션에서 가장 핵심적인 역할

서버 사이드는 물론 모바일 ,데스크톱 애플리케이션에서도 핵심역할

▶ 프레임워크

기존의 수많은 프레임워크의 구조 :

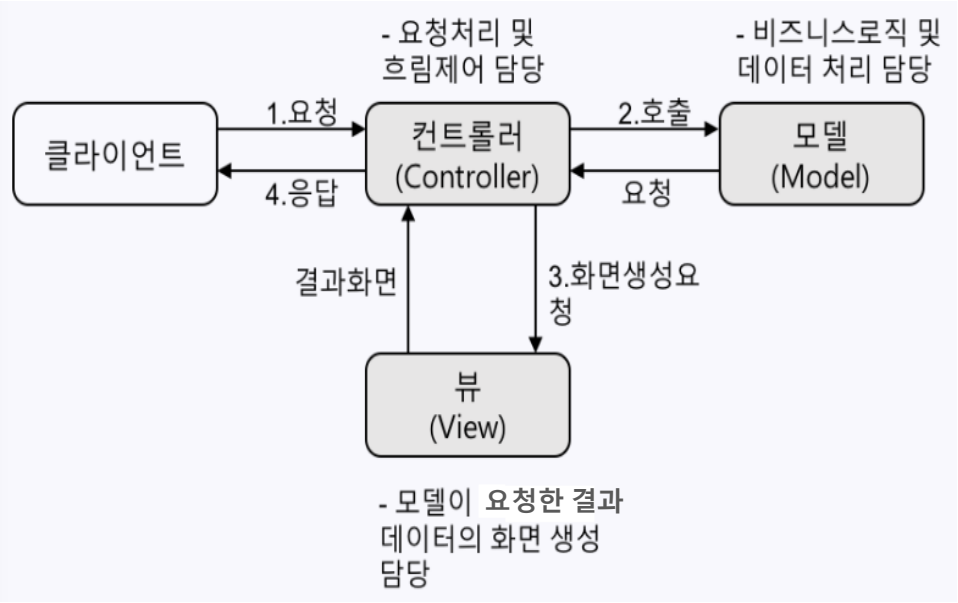
MVC(Model-View-Controller) 아키텍처,

MVW(Model-View-Whatever) 아키텍처

- 모델(Model)과 뷰(View)가 있다

  * 모델 : 애플리케이션에서 사용하는 데이터를 관리하는 영역

  * 뷰 : 사용자에게 보이는 부분



[그림] MVC 아키텍처

▶ DOM (Document Object Model)

하나의 웹페이지의 정보를 모두 담고 있는 자바스크립트 객체

화면 구조의 변경이 있을 경우 DOM으로부터 수정할 부분을 찾아 일일이 다 수정된 후 전체 구조 생성

▶ 보다 효율적인 방법

데이터가 변할 때마다 어떤 변화를 줄지 고민하는게 아니라 기존 뷰를 다 버리고 처음부터 새로 뷰를 생성하여 보여주는 방식

- 장점
  - 어떻게 변화를 줄지 신경 쓸 필요 없음
  - 변화가 있으면 기존것은 버리고 정해진 규칙에 따라 새로 뷰를 생성하여 보여줌
- 단점
  - CPU 점유율 증가, 메모리 사용 증가, 새로 랜더링시 끊김 현상 발생

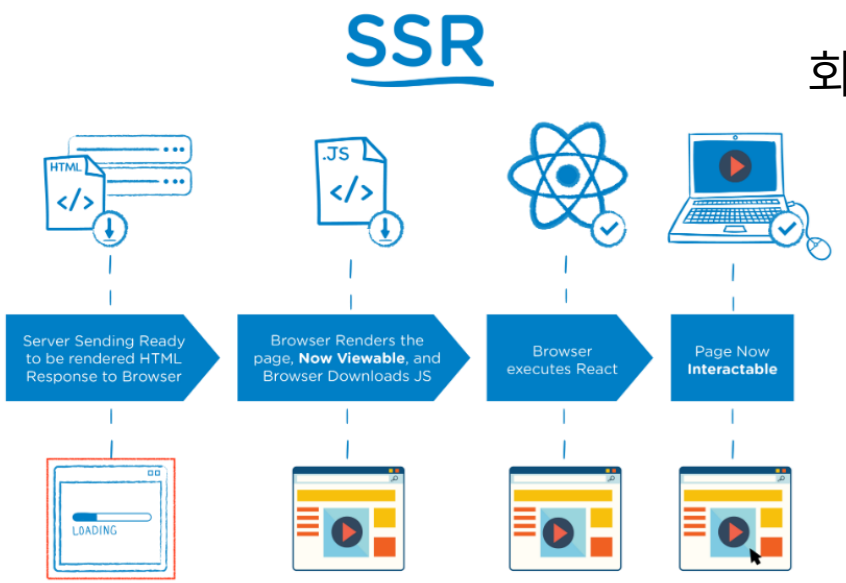
React

▶ 리액트(React)

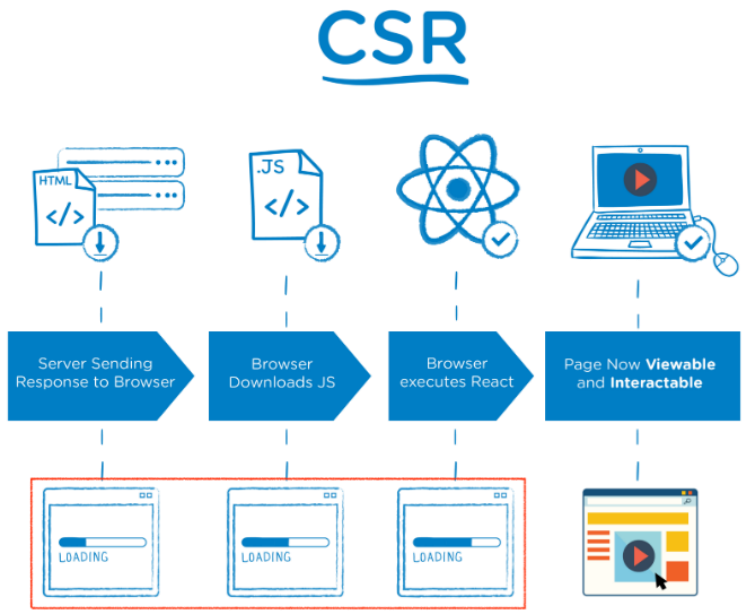
facebook에서 개발한 유저인터페이스 라이브러리

- JSX(JavaScript XML) 사용
- 데이터 바인딩 용이

어디에서
화면구성(Rendering)을
하는가?



Server Side Rendering



Client Side Rendering

출처: The Benefits of Server Side Rendering Over Client Side Rendering

▶ 리액트(React)

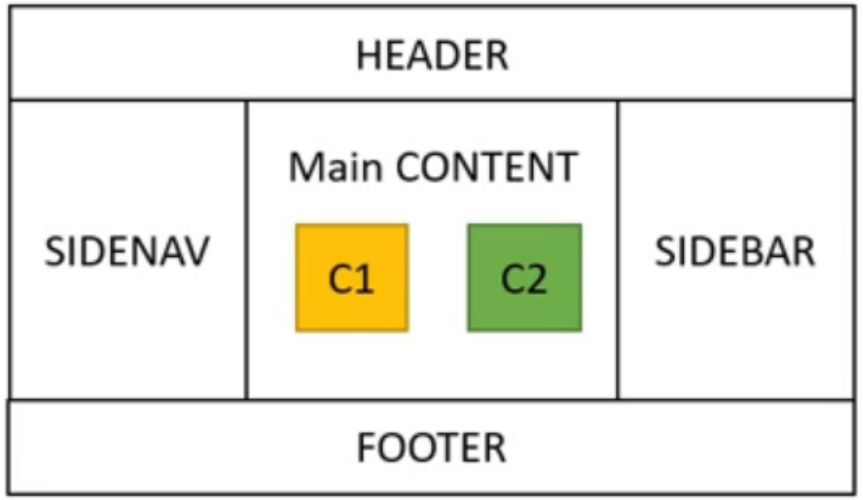
CSR 방식으로 작동

SSR	CSR
최초 화면 로딩이 빠름	최초 화면 로딩이 느림
화면 갱신이 느림	화면 갱신이 빠름
부분 갱신을 위한 전체페이지 reolad	필요한 부분만 부분 갱신

▶ SPA (Single Page Application)

- 하나의 페이지로 구성된 어플리케이션
- 새로운 페이지를 로딩하지 않음 → 페이지 refresh가 일어나지 않음(Seamless한 서비스의 핵심)
- 웹이 마치 앱처럼 동작
- 초기 로딩 시간이 소요되지만 그 이후 페이지 탐색에 대한 UX가 높아짐
- 다양한 JavaScript프레임워크들이 사용됨

  Angular, React, Vue etc

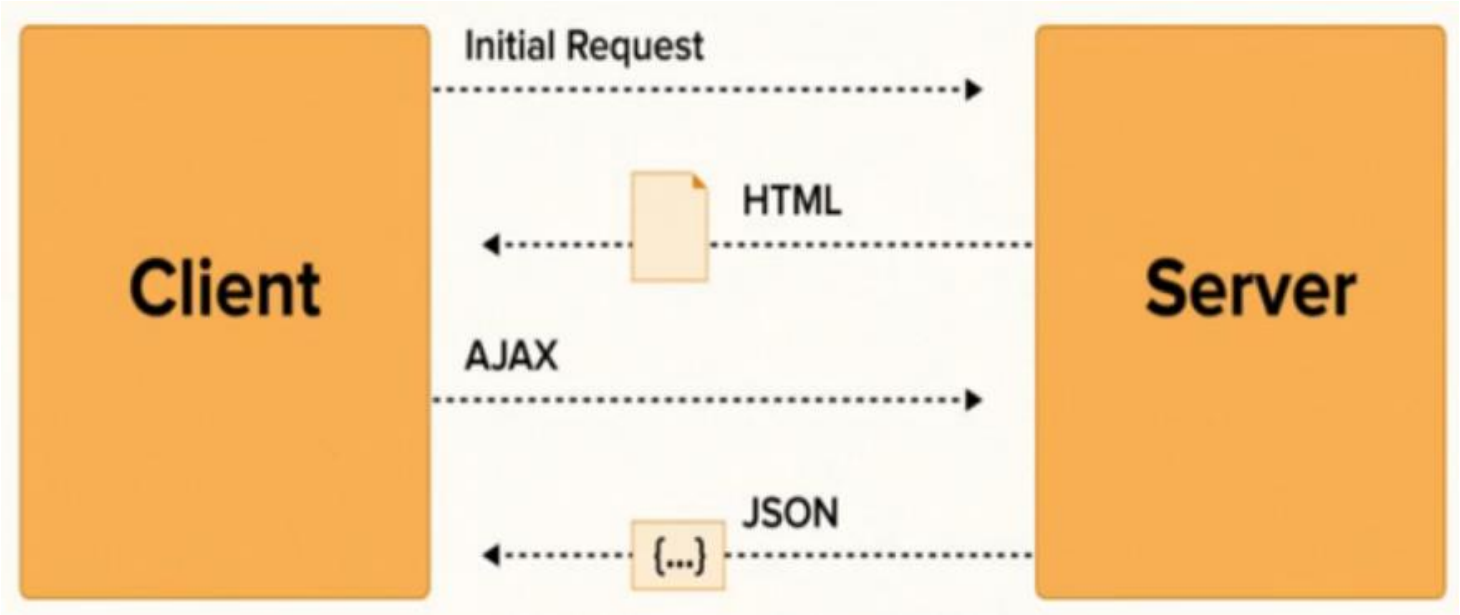


holy grail layout

▶ SPA (Single Page Application)

기존

- AJAX(Asynchronous JavaScript and XML)을 이용하여 페이지의 일부만 갱신할수 있다
  - 어플리케이션이 커질수록 작업이 복잡해짐 (DOM요소를 선택 → HTML조작 → 데이터 바인딩)
  - 느려질수 있다(DOM변화 → CSS연산 → 레이아웃 구성 및 repaint)



▶ 리액트(React)

자바스크립트 라이브러리의 한 종류로, 오직 **V(View)**만 신경쓰는 라이브러리.

- 초기 렌더링 : 최초 실행 렌더링 (사용자 화면에 뷰를 보여 주는것)
    - render() 함수 : 컴포넌트가 어떻게 생겼는지 정의하는 역할
      뷰가 어떻게 생겼고 어떻게 작동하는지에 대한 정보를 지닌 객체 반환
    - 렌더함수로 렌더링이 끝나면 실제 페이지의 DOM요소 안에 주입
  - 리렌더링 : 데이터 변경시 다시 실행되는 렌더링
    - 업데이트가 발생 되면 render() 함수를 다시 호출하여 뷰 생성
    - 기존에 만들었던 컴포넌트와 현재 만든 컴포넌트를 비교하여 바뀐 부분만 DOM트리를 업데이트

용어 설명

렌더링 : 사용자 화면에 뷰를 보여 주는것

컴포넌트 : 해당 컴포넌트의 생김새와 작동 방식을 정의, 재사용이 가능한 수많은 API 기능들을 내장하고 있음

▶ 리액트(React)의 특징

✓ Virtual DOM 사용

웹페이지와 DOM 사이의 중간 매개체의 역할

- 화면 구조의 변경이 있을 경우
- 업데이트 해야할 최소한의 부분만 빠르게 탐색하여 변경 후 변경된 부분만 실제 DOM 에 반영하는 구조

Virtual DOM 이 있기 때문에 빠른 렌더링 가능.

Virtual DOM을 사용한다고 모두 빠른 것은 아님

- **지속적으로 데이터가 변화하는 대규모 애플리케이션** 구축할 때 효과를 얻을 수 있다

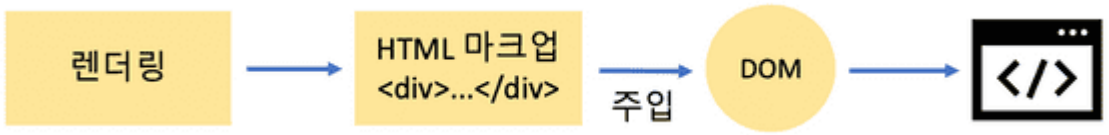
▶ DOM (Document Object Model)

하나의 웹페이지의 정보를 모두 담고 있는 자바스크립트 객체
화면 구조의 변경이 있을 경우
DOM으로부터 수정할 부분을 찾아 일일이 다 수정된 후 전체 구조 생성

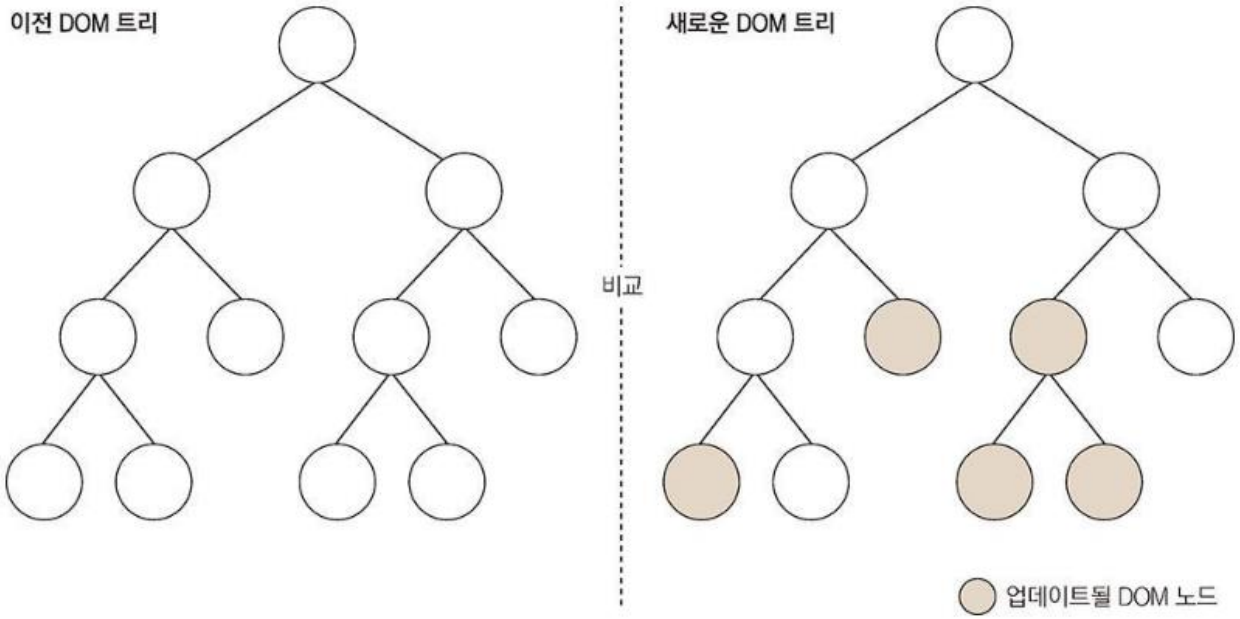
▶ Virtual DOM

웹페이지와 DOM 사이의 중간 매개체의 역할
화면 구조의 변경이 있을 경우
업데이트 해야할 최소한의 부분만 빠르게 탐색하여 변경 후 변경된 부분만
실제 DOM 에 반영하는 구조
Virtual DOM 이 있기 때문에 빠른 렌더링이 가능해진다.

▶ Virtual DOM



[그림] 초기 렌더링

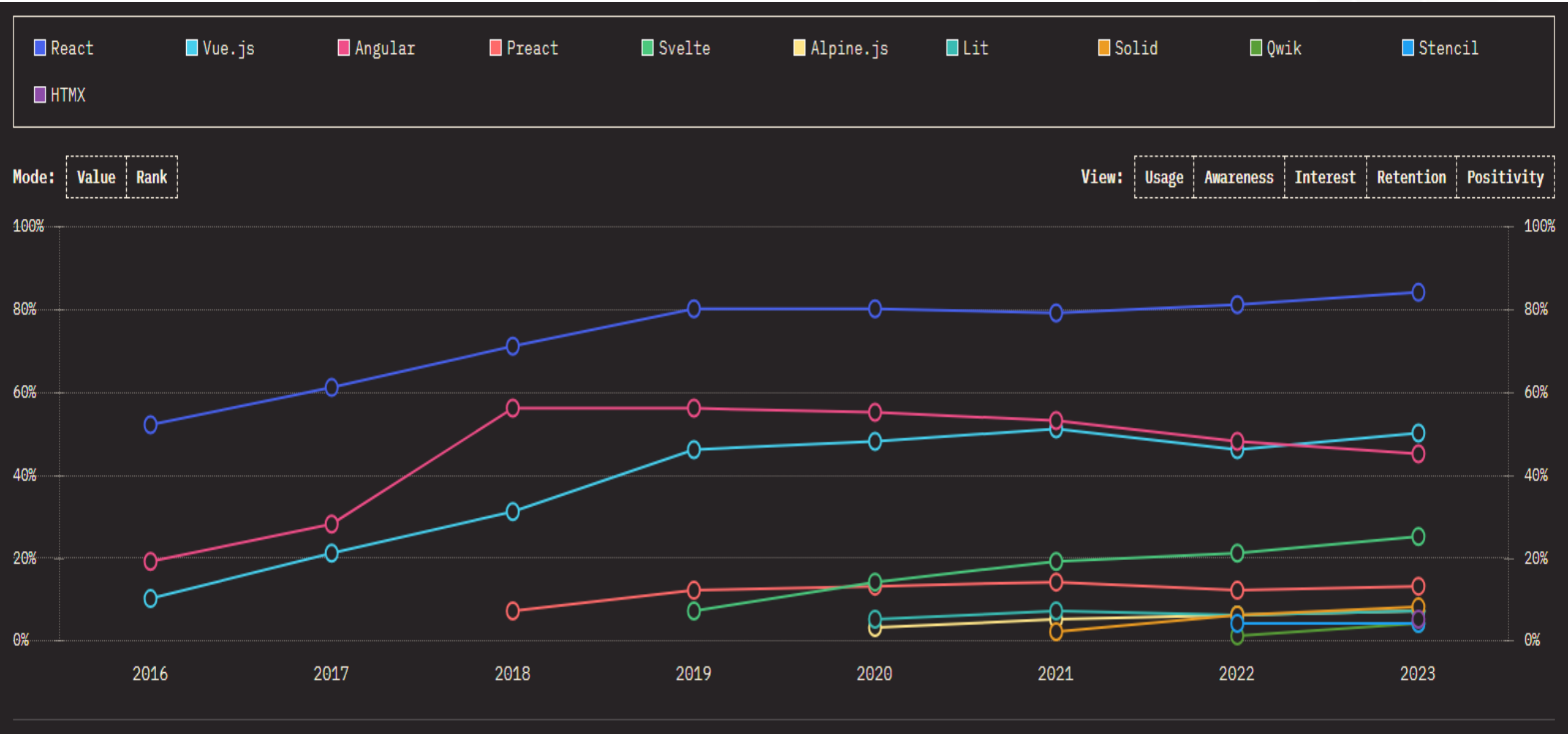


[그림] 리렌더링

▶ React 쓰는 이유

1. Component 기반의 UI 라이브러리
2. 선언형 프로그래밍
3. Virtual DOM 사용

Front-end-frameworks 동향



출처 : https://2024.stateofjs.com/en-US/libraries/front-end-frameworks/


68,149
83,860
8,974
952
211
115 Ember
87
TOP JS TRENDS
jQuery
NEXT.
Vue.js


RACE
Most
Popular
Programming
Languages
1965-2022