FrontEnd Develpoment Trends

with FrontEnd Frameworks

Framework? Framework!

Framework 용어의 정리

- 라이브러리: 프로그램에서 필요로 하는 기능을 재활용하기 위해 모아둔 것
- Framework
  - 응용프로그램의 전체 혹은 일부 형태를 규정 혹은 방침화한 것
  - 뼈대 혹은 반제품으로 프레임워크를 이용 전체적인 틀을 구성한 후
  - 필요한 부분만 채워 완제품을 만들어낸다.

Framework? Framework!

JavaScript와 Framework

- 초기의 JavaScript
  - DOM의 요소에 접근하여 약간의 행동을 추가하는 제한적 용도로만 활용
  - 표준이 통일되어 있지 않아 브라우저마다 서로 다르게 동작
- JavaScript 표준화
  - 브라우저마다 제각각이던 JavaScript의 표준 규약을 제정
  - ECMAScript
- XMLHttpRequest와 비동기 통신의 등장
  - 표준사양으로 채택된 XMLHttpRequest
  - 비동기 통신인 Ajax의 핵심 기반 기술
  - 웹어플리케이션의 현대적 아키텍처로 널리 쓰이게 됨

Framework? Framework!

JavaScript와 Framework

- 웹브라우저를 탈출한 ECMAScript
  - 2009년 Node.js의 발표
  - 서버 사이드에서 자바스크립트를 실행하는 환경이었지만
    - 함께 탑재된 npm이 널리 활용되면서 JavaScript의 대표적 빌드 환경으로 쓰이게 됨

SPA의 도래

- SPA(Single Page Application)
  - Ajax가 페이지의 일부만을 Rich HTML로 만들 수 있었던 것에 비해
  - 페이지가 동적으로 변화하고 서버의 응답과 상관 없이 화면 전환이 가능한 애플리케이션

Framework? Framework!

주요 프레임워크: React

요약

- A JavaScript library for building user interface
- 사용자의 조작에 따라 사용자 인터페이스가 동적으로 변화하는 웹 애플리케이션을 개발할 수 있게 해 주는 프론트엔드 라이브러리

특징

- JSX: XML 포맷 템플릿을 자바스크립트에 직접 내장
- Virtual DOM: HTML DOM을 직접 제어하지 않고 메모리에 DOM Tree를 구축해 두고 이 트리와 실제 HTML에서 차이가 나는 부분만 수정

Framework? Framework!

주요 프레임워크: Angular

요약

- 강력한 명령행 도구와 필요한 기능을 모두 내장한 **풀스택 프레임워크**
- 버전 1.x와 버전 2.x가 프레임워크 구성이 크게 차이가 난다.

특징

- Angular CLI라는 강력한 명령행 도구를 가지고 있다.
- TypeScript를 기본으로 사용한다.
- RxJS를 이용 비동기적 이벤트를 처리할 수 있다.
- TypeScript의 문법인 **데코레이터**를 활용한 선언적 코딩 스타일을 사용한다.

Framework? Framework!

주요 프레임워크: Vue.js

요약

- 라이브러리적 측면과 프레임워크적 측면을 동시에 갖는 프레임워크
- **Progressive Framework**

특징

- React처럼 가상 DOM을 가진다
- **단일 파일 컴포넌트(Single File Component)** 스타일을 이용, 이해하기 쉽고 개발 난이도가 낮다.

Framework? Framework!

주요 프레임워크: React Native

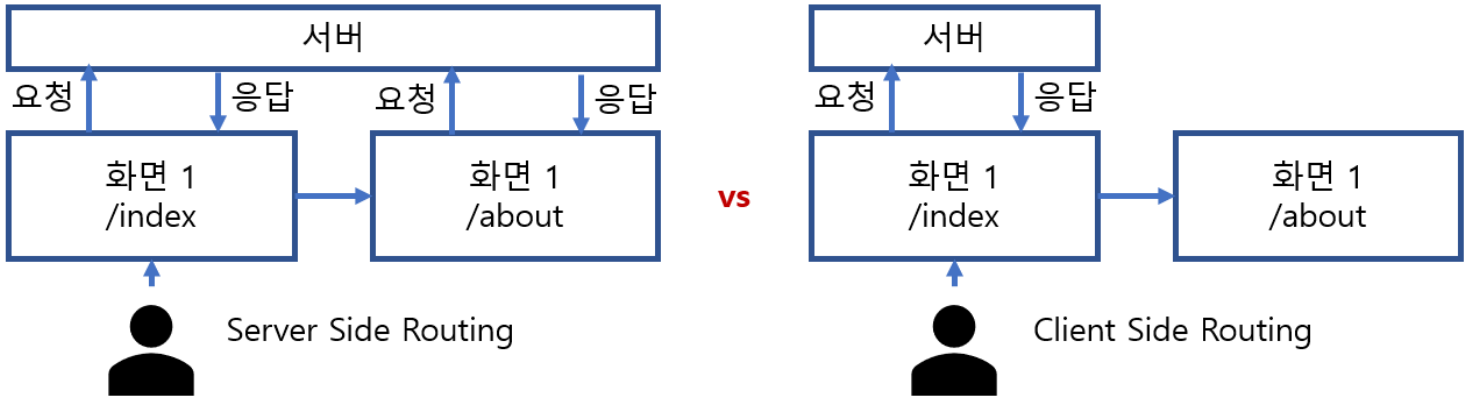
요약

- React의 기술을 기반으로 모바일용 네이티브 앱을 개발할 수 있는 프레임워크
- iOS와 안드로이드를 모두 지원하는 앱을 개발할 수 있다.

프론트엔드 구현 기술 최신 동향

클라이언트 사이드 라우팅

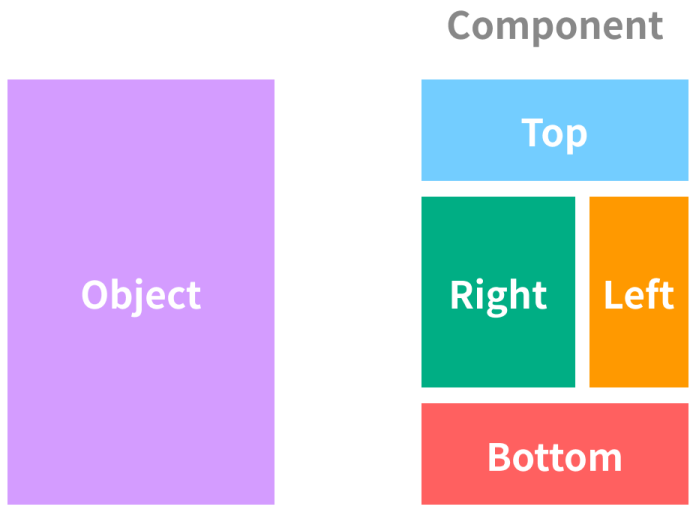
- 라우팅: 사용자의 요청이 들어온 URL에 대하여 적절한 응답을 돌려주기 위해 어떤 처리와 매칭시킬지를 결정하는 과정
- 서버 사이드 라우팅: Server에 HTTP 요청을 보내 직접적으로 엔드포인트를 바꾸는 방식 = 페이지 직접 접근
- 클라이언트 사이드 라우팅: 클라이언트 상에서 엔드포인트를 바꾸는 방식 = 가상 엔드포인트(서버에 실재하지 않음)



프론트엔드 구현 기술 최신 동향

컴포넌트 지향

- 컴포넌트와 컴포넌트 간의 **상호작용** 형태로 프로그램을 작성하는 방식
- 애플리케이션의 기본 정보를 포함하는 페이지를 통째로 하나의 루트 컴포넌트로 정의
- 루트 컴포넌트 안에 다른 컴포넌트를 포함시키는 형태로 전체 프로그램을 개발한다.
- 각 컴포넌트가 단독으로 필요한 기능을 수행할 수 있는 스타일과 스크립트를 포함하는 독립된 존재여야 한다.



프론트엔드 구현 기술 최신 동향

SSR과 프리 렌더링

클라이언트 사이드 렌더링과 한계

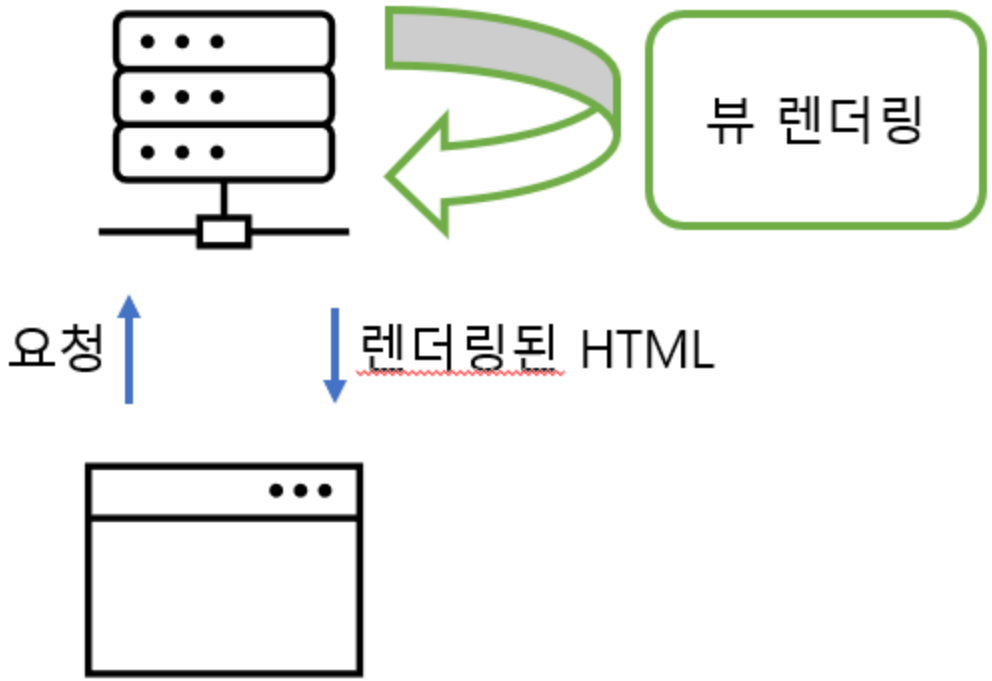
- SPA에서는 일반적인 웹 애플리케이션과 달리 내용이 없는 빈 HTML을 응답으로 받아온다.
- 클라이언트 사이드 렌더링의 문제점
  - 페이지 표시 속도
  - 크롤러가 대응하지 못함
  - **기계가독성**이 낮다.

프론트엔드 구현 기술 최신 동향

SSR과 프리 렌더링

서버 사이드 렌더링

- SSR을 적용하면 서버에서 전송하는 응답 결과에 렌더링을 마친 상태를 전달한다.
- 서버측에서 SPA에서 렌더링할 결과를 대신 완성해서 전달해야 한다.
- 서버사이드 렌더링 솔루션
  - React: ReactDOMServer
  - Vue.js: vue-server-renderer
  - Angular: Angular Universal

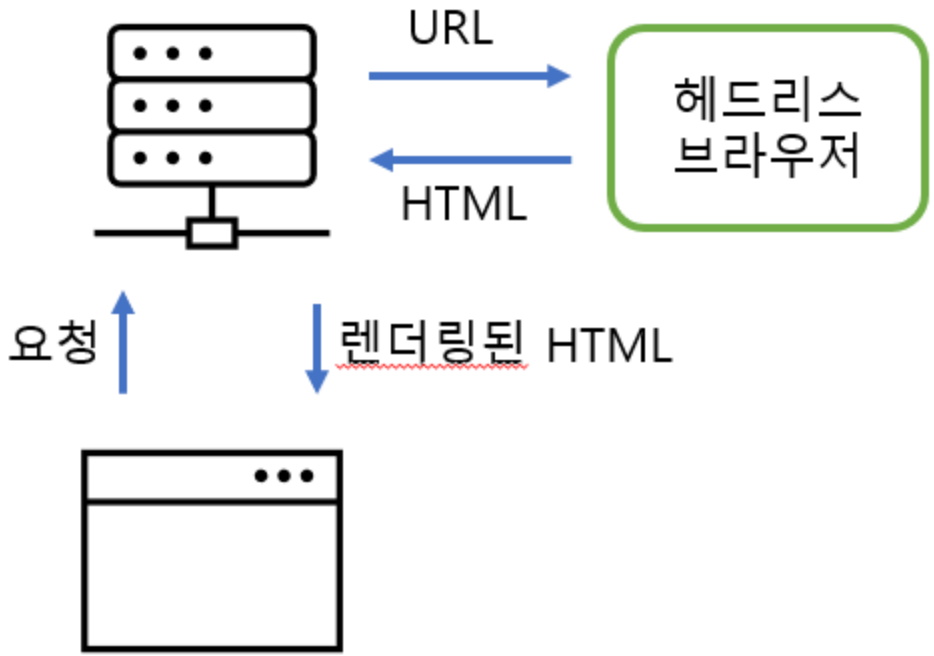


프론트엔드 구현 기술 최신 동향

SSR과 프리 렌더링

프리 렌더링(Pre-Rendering)

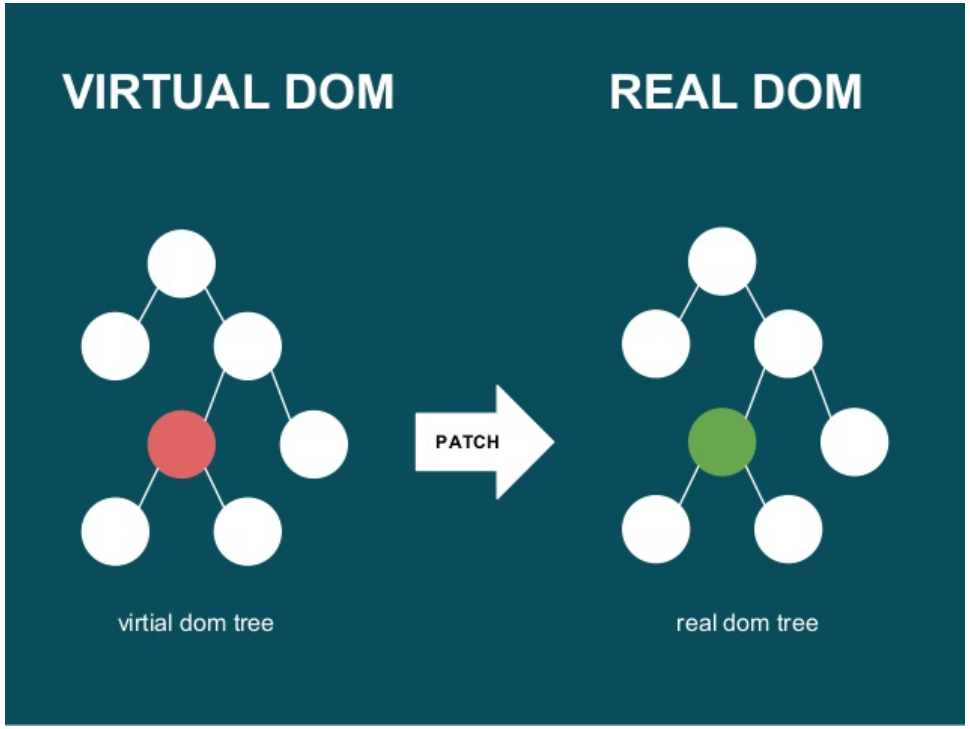
- 미리 렌더링 해 두기
- 서버 사이드 렌더링과는 달리 서버에 렌더링 로직이 존제하지 않음
- 요청이 들어왔을 때 대상이 크롤러인지를 판단, 가상 브라우저(headless browser)로 접근하는 방식으로 생성한 다음 저장해 둔 렌더링 결과를 제공하는 방식



프론트엔드 구현 기술 최신 동향

가상 DOM

- 자바스크립트가 HTML을 렌더링하는 방법 중 하나
  - 메모리상에 가상의 DOM을 만들어 두고
  - 가상 DOM 구조를 업데이트 한 다음
  - 가상 DOM의 현재 상태와 이전 상태의 차이를 구해서
  - 차이만을 실제 DOM에 반영

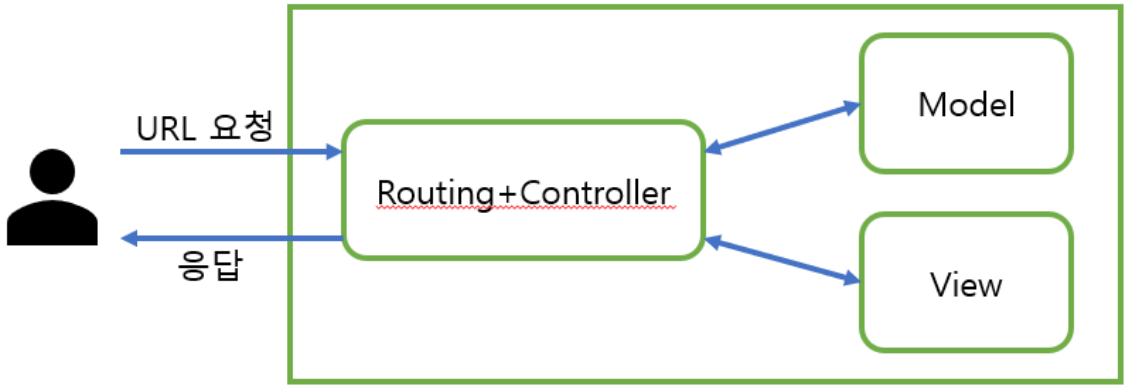


프론트엔드 구현 기술 최신 동향

MVC, MVP, MVVM

MVC

- Model-View-Controller
  - 모델: 데이터, 데이터에 접근하기 위한 기능, 다양한 로직
  - 뷰: 콘텐츠를 어떻게 출력할지를 정의하는 부분
  - 컨터롤러: 특정 URL을 요청받았을 때 (Routing), 컨트롤러가 모델과 정보를 주고 받은 다음 적절한 뷰를 골라 사용자에게 반환



프론트엔드 구현 기술 최신 동향

MVC, MVP, MVVM

프론트엔드에서의 MV

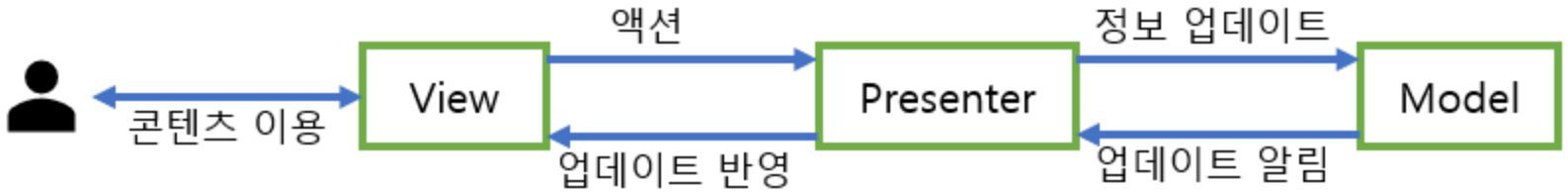
- 프론트엔드 애플리케이션에서 MV는 MVC 애플리케이션의 뷰에 해당
  - 본래 웹 애플리케이션은 콘텐츠가 HTML에 포함된 채로 전달되기 때문에 뷰 안에 MV와 같은 구성을 할 필요가 없음
  - SPA에서는 뷰 안에서도 Ajax 등을 통한 데이터 접근 계층이 추가되거나 필요에 따라 URL을 교체하는 처리 등이 개입하기 때문에 MV 구조하 필요하게 됨
- 주요 패턴
  - MVP(Model-View-Presenter) 패턴
  - MVVM(Model-View-ViewModel) 패턴

프론트엔드 구현 기술 최신 동향

MVC, MVP, MVVM

프론트엔드에서의 MVP

- 모델과 뷰 사이에 프리젠터가 위치
- 모델과 뷰 사이의 입출력 인터페이스 역할을 수행
- 사용자가 모델의 정보를 수정하거나 읽으려면 반드시 뷰를 거쳐 프리젠터가 제공하는 인터페이스를 통해야 한다.

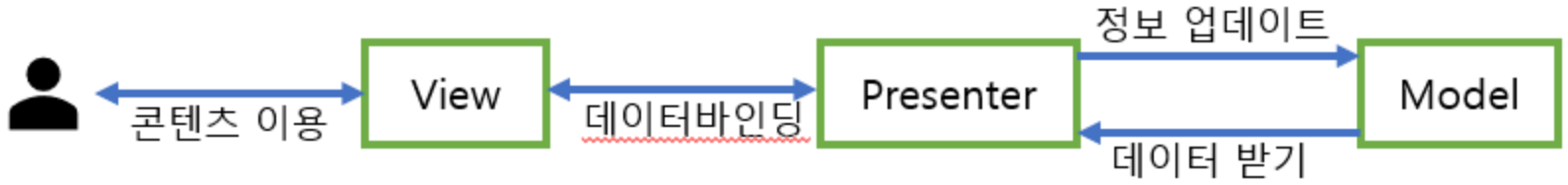


프론트엔드 구현 기술 최신 동향

MVC, MVP, MVVM

프론트엔드에서의 MVVM

- 모델과 뷰 사이에 뷰모델이 위치
- 뷰모델은 양방향 데이터 바인딩을 담당한다
- 뷰를 통해 변경하려는 모델의 값이 뷰모델을 거쳐 변경된 내용을 탐지하고 모델에 변경 내용을 반영한다.



프론트엔드 구현 기술 최신 동향

Flux

- 액션크리에이터(ActionCreator), 디스패처(Dispatcher), 스토어(Store), 뷰(View) 단계를 반복하는 형태로 구성