

# Why We Learn about Vue ?

GreenLight

---

김수현 박지윤 임예지 문준용



Target

- Vue or React와 같은 개발 툴을 사용하는 이유가 궁금한 개발자
- SPA를 어디서 들어봤지만, 명확한 의미를 모르는 프론트 개발자
- 프론트엔드 프레임워크 원리가 궁금한 백엔드 개발자

# 목차

1. Vanilla JavaScript vs SPA Framework
2. 전통적인 웹 페이지 개발과 프론트 개발
3. Vue에 대하여

# 1

## Vanila JavaScript vs SPA Framework

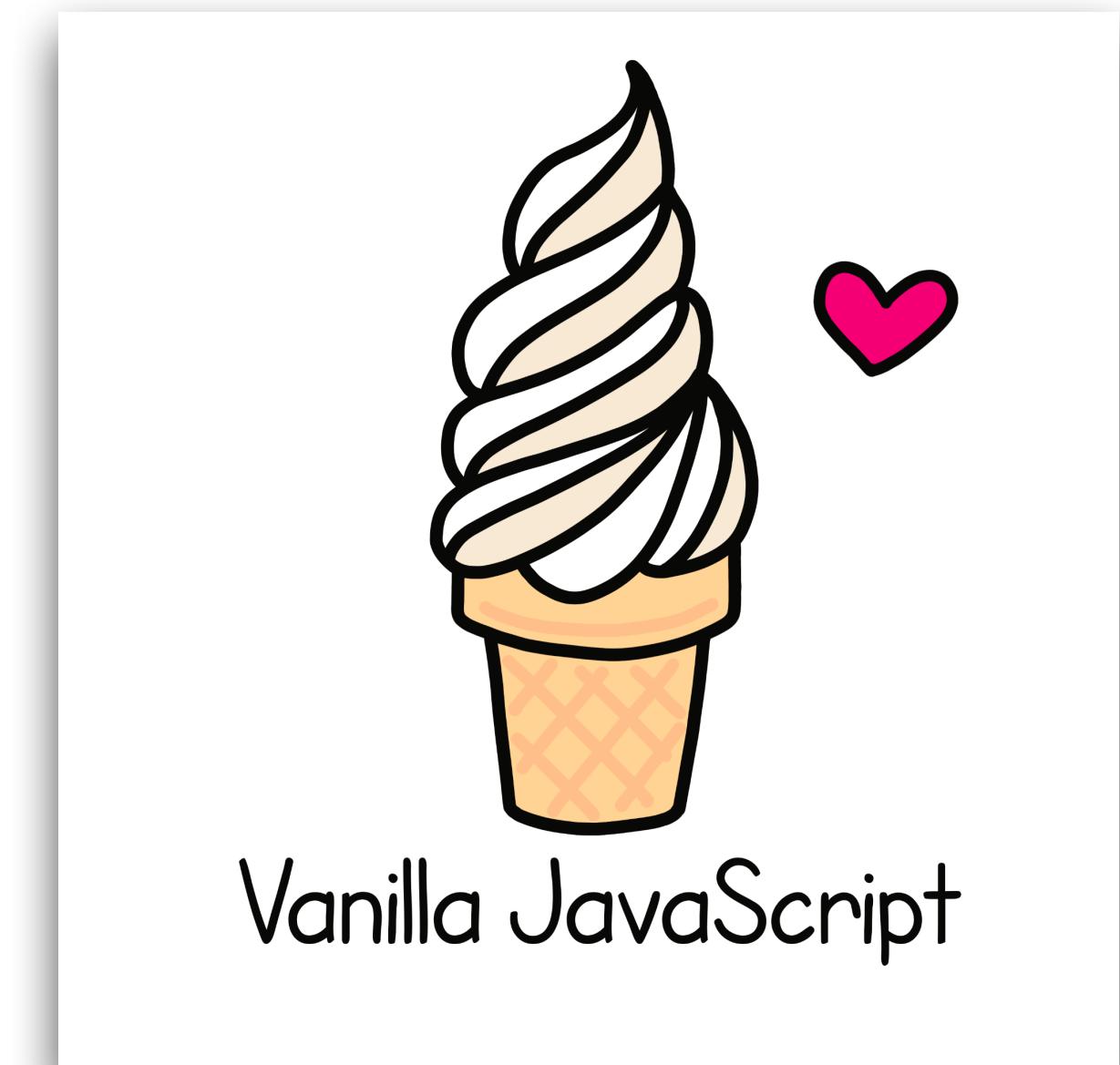
# 세부 목차

- 1 Vanilla JavaScript란?
- 2 Vue.js란?
- 3 AI가 알려준 Vue 키워드
- 4 JavaScript / Vue.js 프로젝트 비교

# Vanilla JavaScript란?

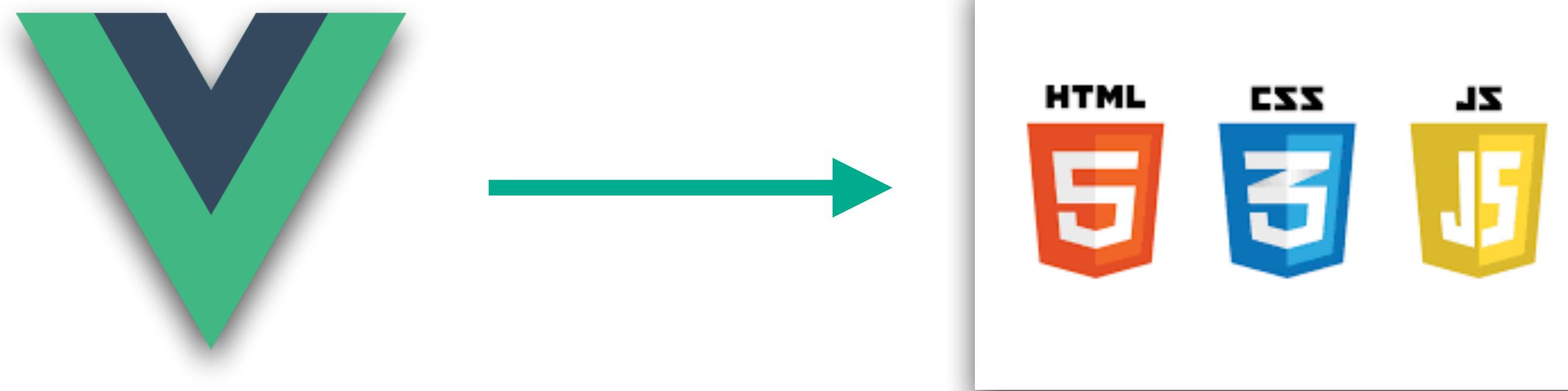
소프트웨어 세계에서는 Vanilla = Plain(기본)을 뜻한다고 합니다.  
따라서 Vanilia JavaScript는 '순수 자바스크립트'를 의미합니다.

추가적인 JavaScript 프레임워크나 라이브러리를 사용하지 않는  
것이 Vanilia JavaScript로 개발한 것입니다.



# Vue.js란?

- 사용자 인터페이스를 구축하기 위한 JavaScript 프레임워크
- 표준 HTML, CSS 및 JavaScript를 기반으로 구축되어 사용자 인터페이스를 효율적으로 개발할 수 있는 컴포넌트 기반 프로그래밍 모델을 제공



## AI가 알려준 Vue 키워드

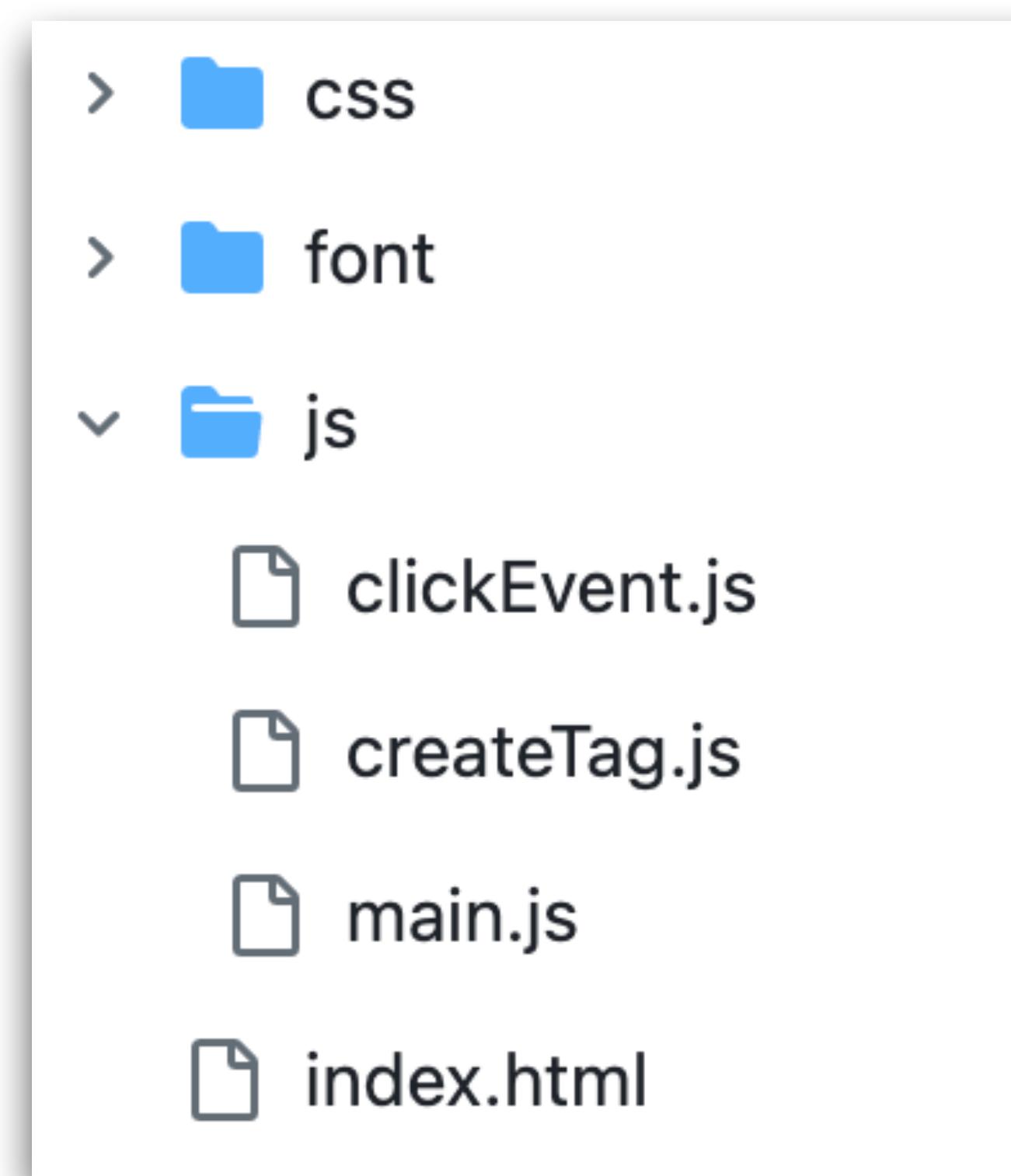
쉽다

강력하다

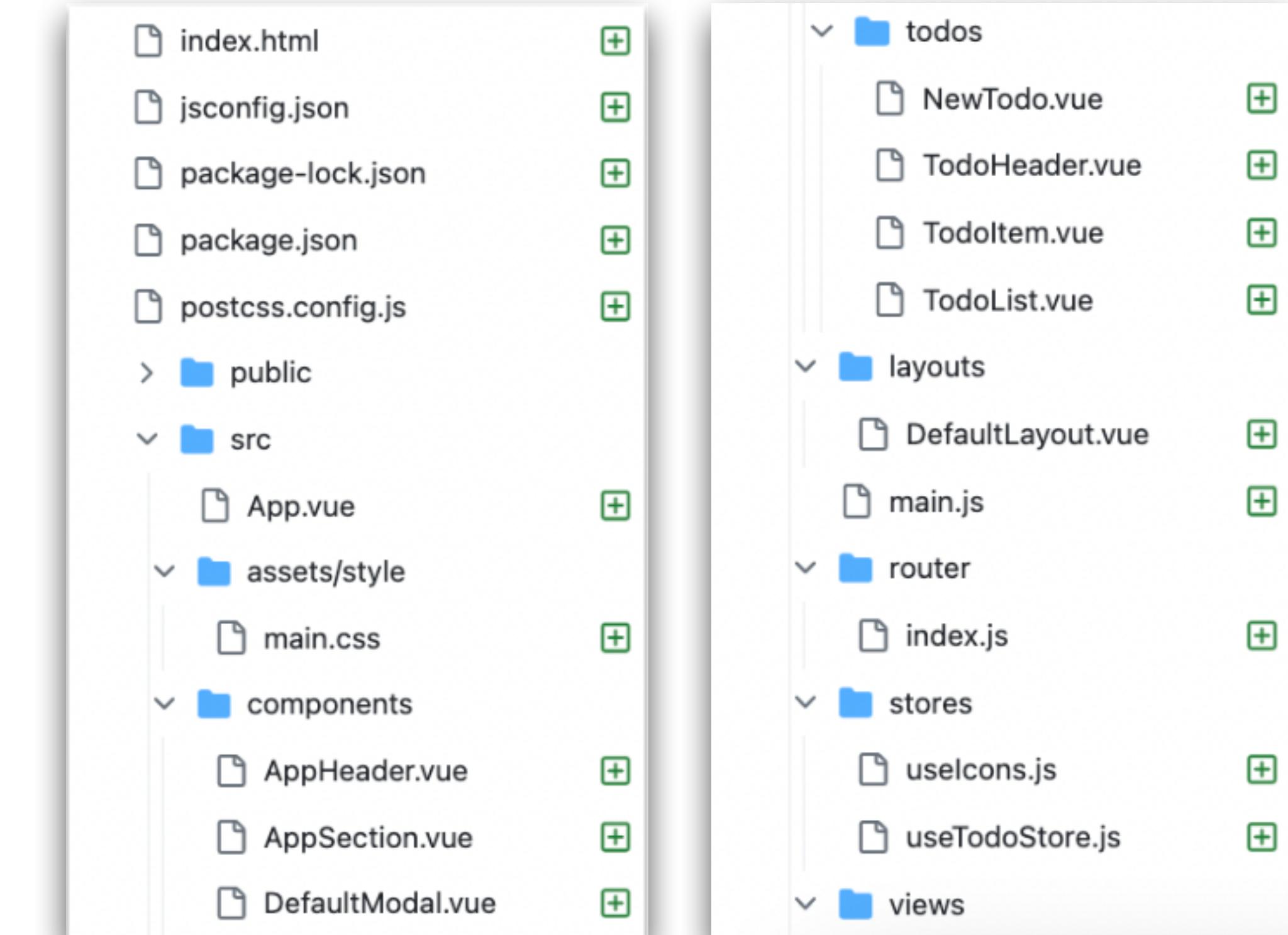
유연하다

# JavaScript / Vue.js 프로젝트 비교

## Vanila JavaScript



## Vue.js



# 2

전통적인 웹 페이지 개발과  
프론트 개발

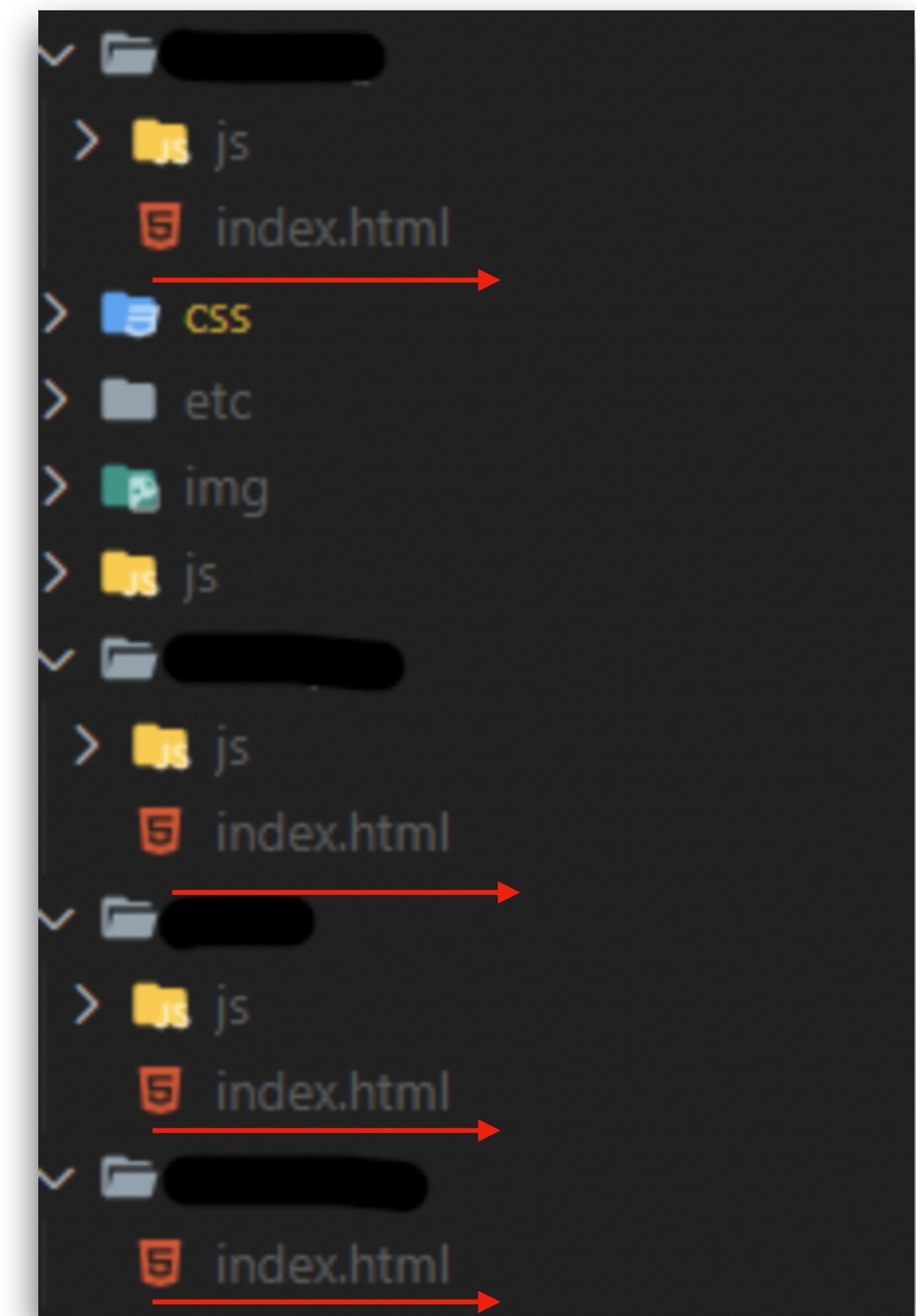
## 세부 목차

- 1 MPA 개념, 장/단점, 렌더링 방식
- 2 SPA 등장 배경, 개념
- 3 SPA 동작원리
- 4 SPA 라이브러리와 프레임워크

## MPA 개념, 장/단점, 렌더링 방식

### MPA (= Multiple Page Application)

- 전통적인 프론트 개발 방식
- 페이지 수 만큼 html 파일을 생성

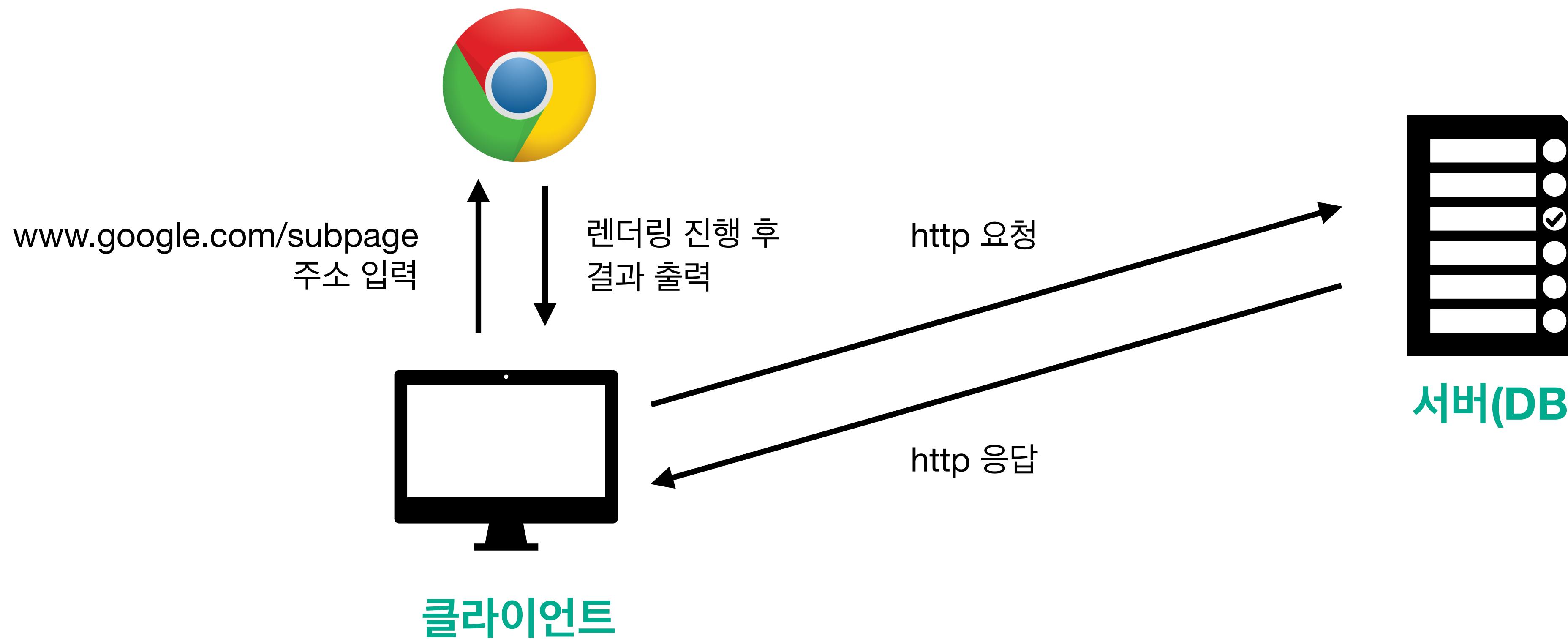


폴더마다 **index.html**이 존재함

# MPA 개념, 장/단점, 렌더링 방식

MPA 렌더링 방식 : **SSR** (= Server Side Rendering)

브라우저 렌더링 엔진



# SPA 등장 배경, 개념

## SPA 개념

- 한 개의 페이지로 구성된 Application
- 최초 실행시, 필요한 모든 정적 리소스를 한 번에 다운로드한다.

```
C:\우리_Worksace\WooriFISA\5.vue\vue-todo>npm run build

> vue-todo@0.0.0 build
> vite build

vite v4.3.5 building for production...
✓ 43 modules transformed.

dist/index.html          0.42 kB  gzip:  0.29 kB
dist/assets/AboutView-4d995ba2.css  0.09 kB  gzip:  0.10 kB
dist/assets/index-c51a37a5.css      9.17 kB  gzip:  2.50 kB
dist/assets/AboutView-c73e0bd4.js    0.23 kB  gzip:  0.20 kB
dist/assets/index-1444b337.js       99.11 kB  gzip: 38.57 kB
✓ built in 1.67s
```

## SPA 동작원리

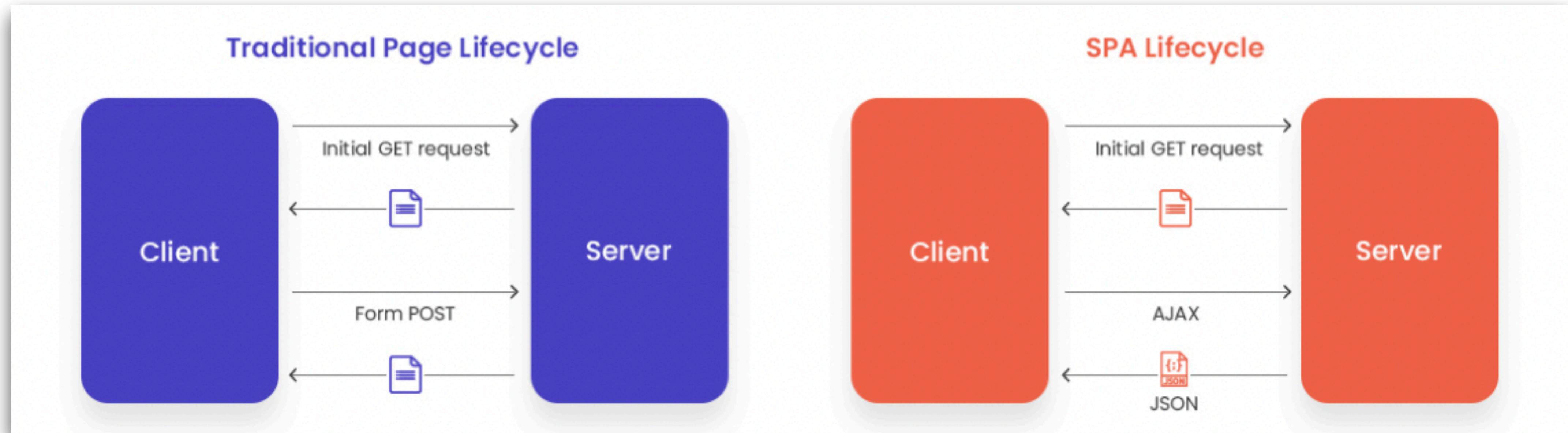
### SPA 렌더링 방식 - CSR(Client Side Rendering)

- 클라이언트는 서버에게 **최초 URL** 요청 시 한 번만 정적 파일을 가져오고 웹 브라우저에 렌더링함.
  - 이후의 페이지 내 렌더링은 서버의 요청 없이 **클라이언트**에서만 일어난다.
- > 화면 깜빡임 X

# SPA 동작원리

## SSR

## CSR



# SPA 라이브러리와 프레임워크

## MPA vs SPA

항목	MPA(SSR)	SPA(CSR)
초기 로딩 속도	당장 필요한 소스만 받아서 빠름	한 번에 프로젝트의 모든 소스 코드를 다운받아서 오래 걸림.
페이지 전환	새롭게 페이지를 서버에 요청하여 브라우저 렌더링 과정을 거친다. 서버에서 잖은 요청을 하게 됨.	API 통신(데이터 요청)을 할 때에만 서버에서 요청하므로, 요청 횟수가 잖고 뷰의 전환 속도가 빠름.
SEO	HTML에 정적 소스가 처음부터 포함되어 있기 때문에, 크롤러 봇의 수집이 잘됨.	처음부터 HTML의 정적 소스가 모두 비어있고, JavaScript로만 되어 있음. 크롤러 봇은 JavaScript를 읽어내지 못하기 때문에 데이터 수집을 할 수 없다.

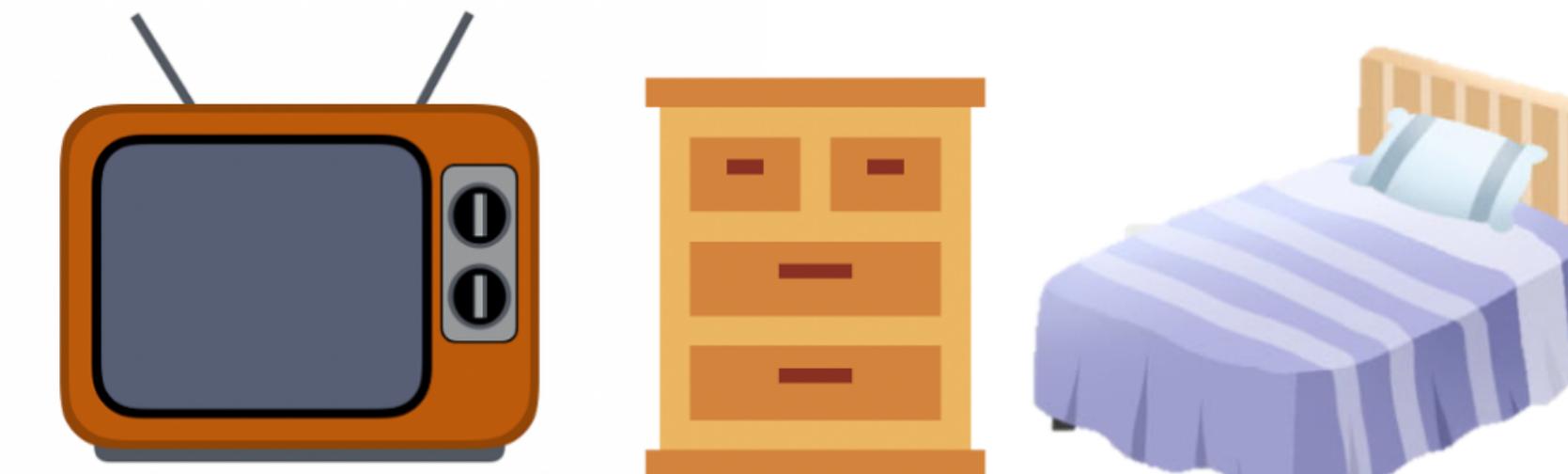
# SPA 라이브러리와 프레임워크

차이 : 통제권이 어디에 있는가

프레임워크와 라이브러리의 차이



FrameWork(프레임워크)



Library(라이브러리)

# SPA 라이브러리와 프레임워크

## 라이브러리

- 미리 작성된 도구(코드, 변수, 함수, 클래스)를 이용해 사용자가 **능동적**으로 코드 작성
- 통제권: **개발자** > 라이브러리
- Ex) React

## 프레임워크

- 미리 정의된 메뉴얼 안에서 사용자가 **수동적**으로 코드 작성
- 통제권: **개발자** < **프레임워크**
- Ex) Vue.js, Next.js, Nuxt.js, Angular

# 3 Vue

## 세부 목차

- 1 Vue.js 특징 3 가지
- 2 디자인 패턴과 MVVM 패턴
- 3 Vue.js 핵심 기능 2 가지
- 4 결론

3-1

## Vue.js 특징 3 가지

#컴포넌트

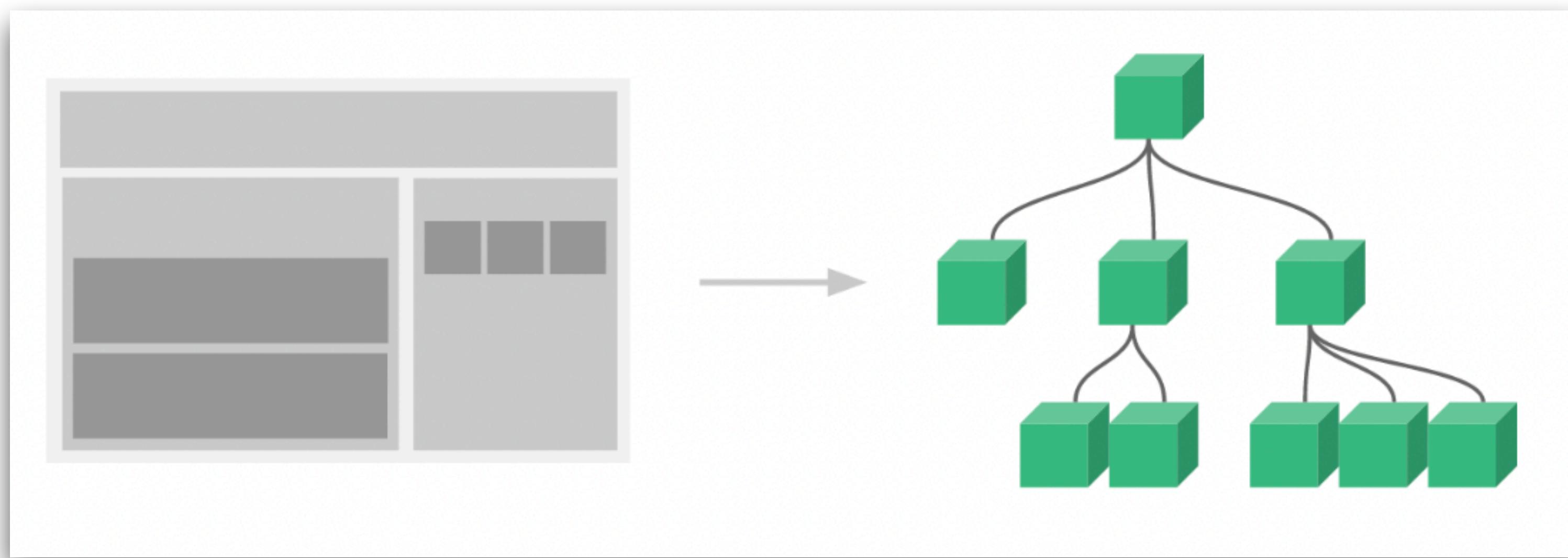
#SFC

#렌더  
파이프 라인

# Vue.js 특징 3 가지

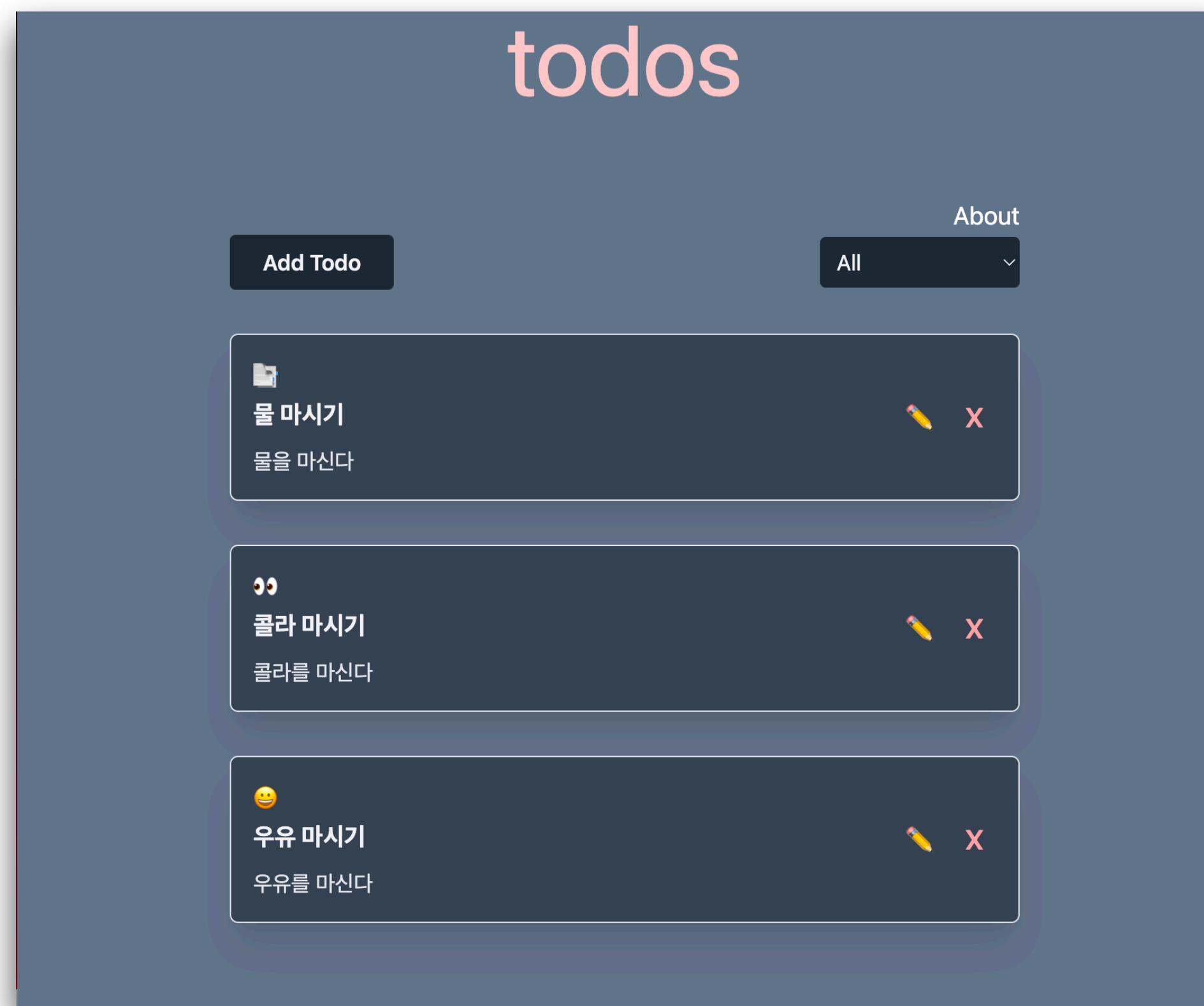
## 1. 컴포넌트 기반 아키텍처

- 자체 포함된 재사용 가능한 구성 요소에서 앱을 빌드하고 있음을 의미



# Vue.js 특징 3 가지

## 1. 컴포넌트 기반 아키텍처 - 예시 화면



## 컴포넌트 구조

```
components
├── __tests__
│   └── HelloWorld.spec.js
├── icons
│   ├── IconCommunity.vue
│   ├── IconDocumentation.vue
│   ├── IconEcosystem.vue
│   ├── IconSupport.vue
│   └── IconTooling.vue
└── todos
    ├── EditTodo.vue
    ├── NewTodo.vue
    ├── TodoFilter.vue
    ├── TodoHeader.vue
    ├── TodoItem.vue
    └── TodoList.vue
    ├── AppHeader.vue
    ├── AppSection.vue
    ├── DefaultModal.vue
    └── HelloWorld.vue
```

# Vue.js 특징 3 가지

## 2. 단일 파일 구성 요소

**SFC(=Single File Components)**

**모든 SFC에는 세 부분으로 나뉩니다.**

- template: HTML
- script: JavaScript
- style: CSS or SCSS

```
<template>
<div>
  <p class="hello-msg">{{ message }}</p>
  <button v-bind:click="sayBye">
    Thanks
  </button>
</div>
</template>

<script>
export default {
  data () {
    return {
      message: "Hello Everyone !"
    }
  },
  methods: {
    sayBye () {
      this.message = "Thanks a lot!"
    }
  }
}
</script>

<style>
.hello-msg {
  font-size: 30px;
  color: red;
}
</style>
```

# Vue.js 특징 3 가지 3. 렌더 파이프 라인

1. 컴파일 : Vue 템플릿은 렌더 함수로 컴파일 됨

2. 마운트 : 런타임 렌더러는 렌더 함수를 호출,  
변환된 가상 DOM 트리 탐색,  
이를 기반으로 실제 DOM 노드 생성

3. 패치 : 마운트 중에 사용된 의존성이 변경되면 이팩트가 다시 실행



## 디자인 패턴과 MVVM 패턴



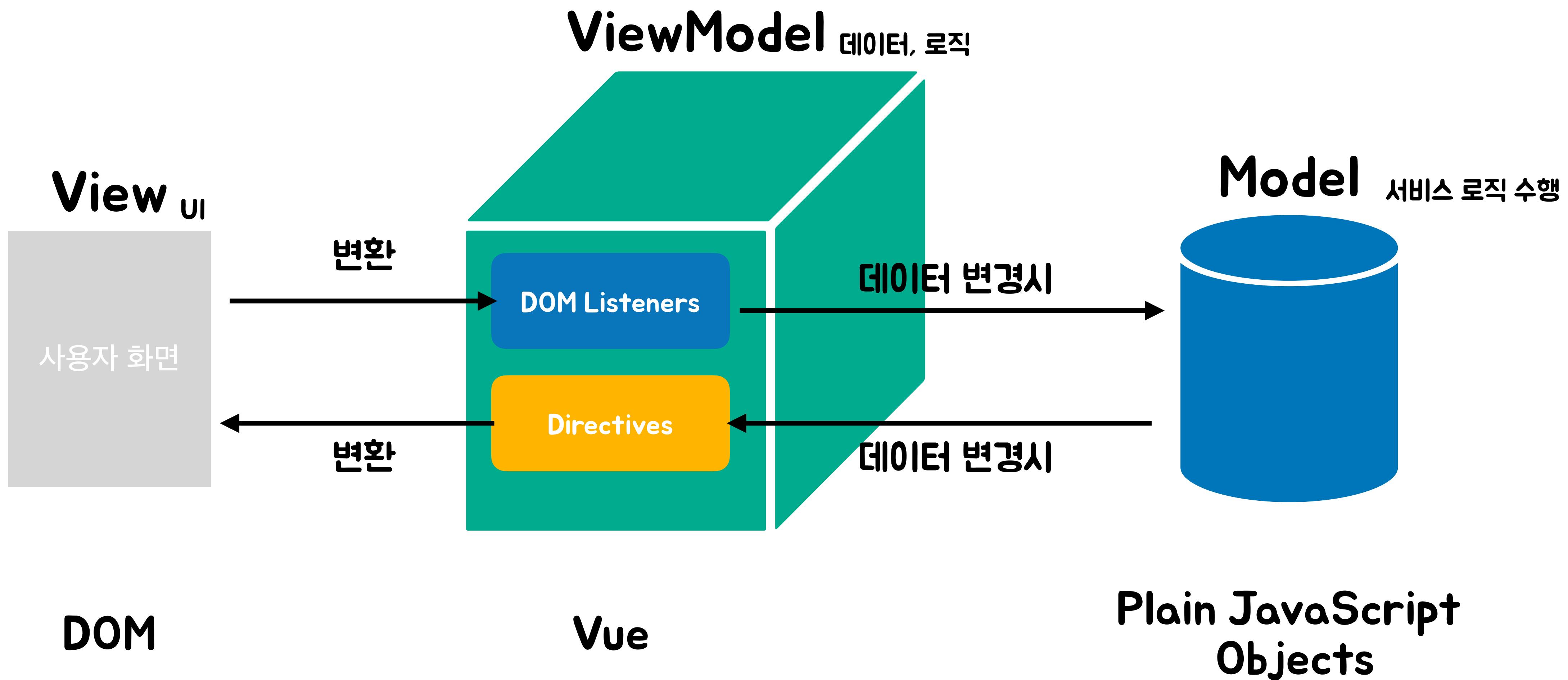
필요한 옷을 쉽게 꺼낸다는 목적성 X

-> 불편한 부분을 찾고 하지 말아야겠다는  
규칙을 만들자!

비슷한 것끼리 분류해서 모은다 -> 목적성 O

-> 지속적으로 관리가 잘되는 코드를  
좋은 아키텍쳐가 필요해!

# 디자인 패턴과 MVVM 패턴



# 참고문서

- [Vue.js 공식문서\(v3-docs, kr\)](#)
- [Vue.js란 무엇입니까?](#)
- [Vue.js 라우터 해시 모드, 히스토리 모드](#)
- [Node.js connect-history-fallback API](#)
- [라이브러리와 프레임워크 차이점](#)
- [Software architecture](#)
- [Software design pattern](#)
- [Software Design Patterns](#)
- [프론트엔드에서 MV\\* 아키텍처란 무엇인가요?](#)
- [프레임워크 없이 만드는 SSR](#)

git!