

왜 우리는 Vue를 사용하는가?

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 Vanilla JavaScript vs SPA Framework

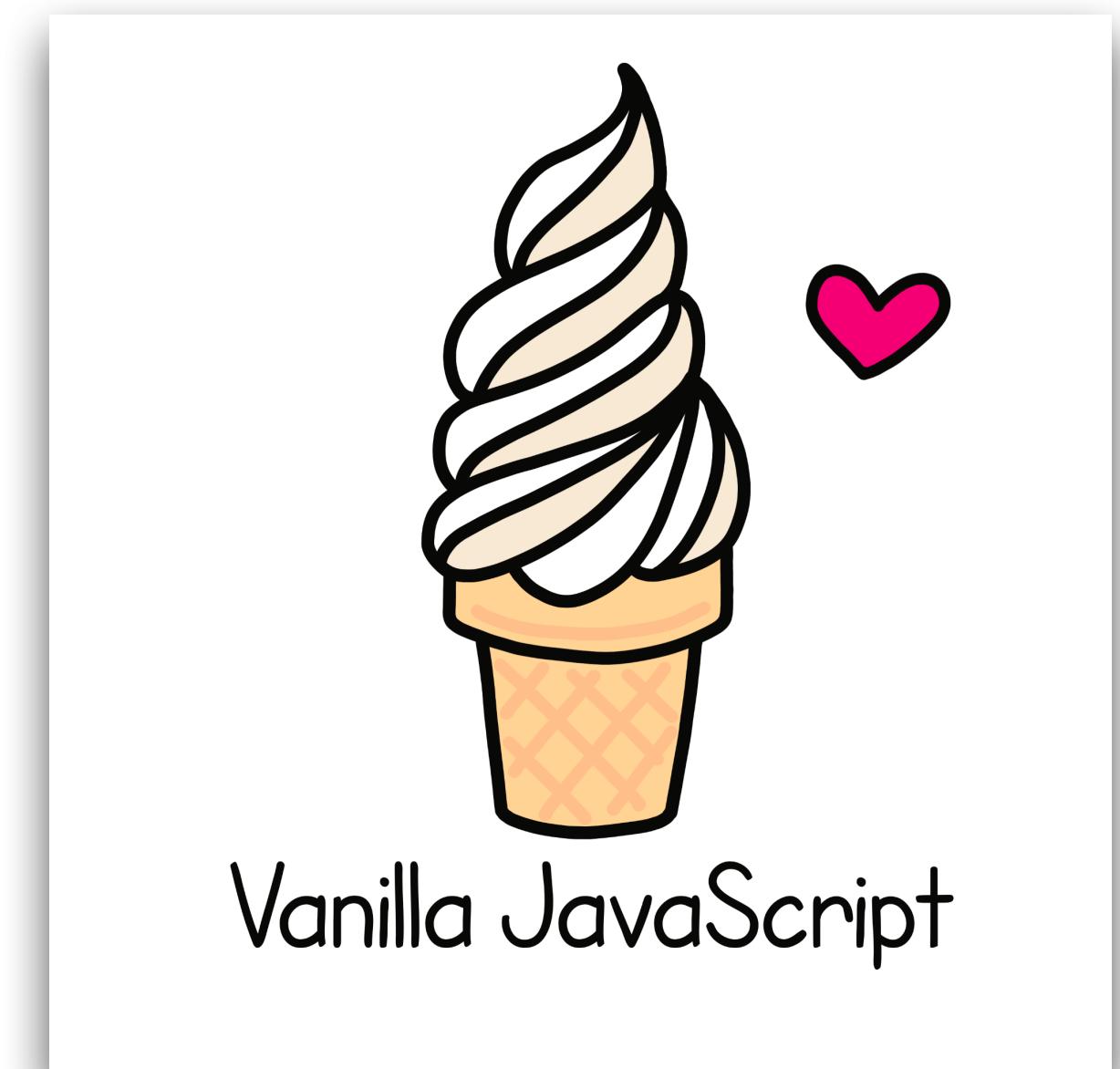
세부 목차

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

Vanilla JavaScript란?

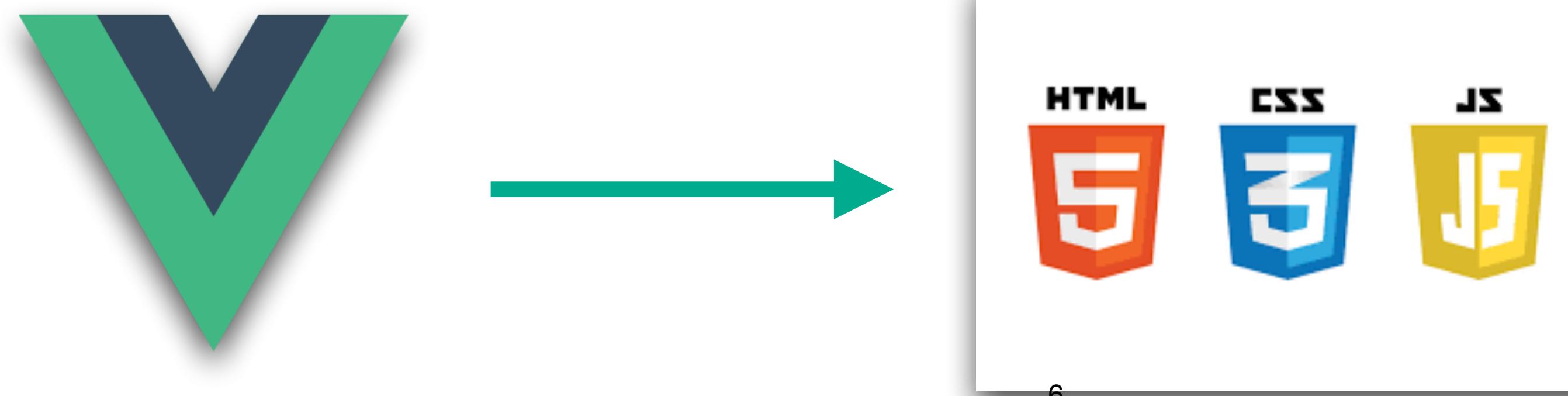
소프트웨어 세계에서는 Vanilla = Plain(기본)을 뜻한다고 합니다.
따라서 Vanilia JavaScript는 '순수 자바스크립트'를 의미합니다.

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

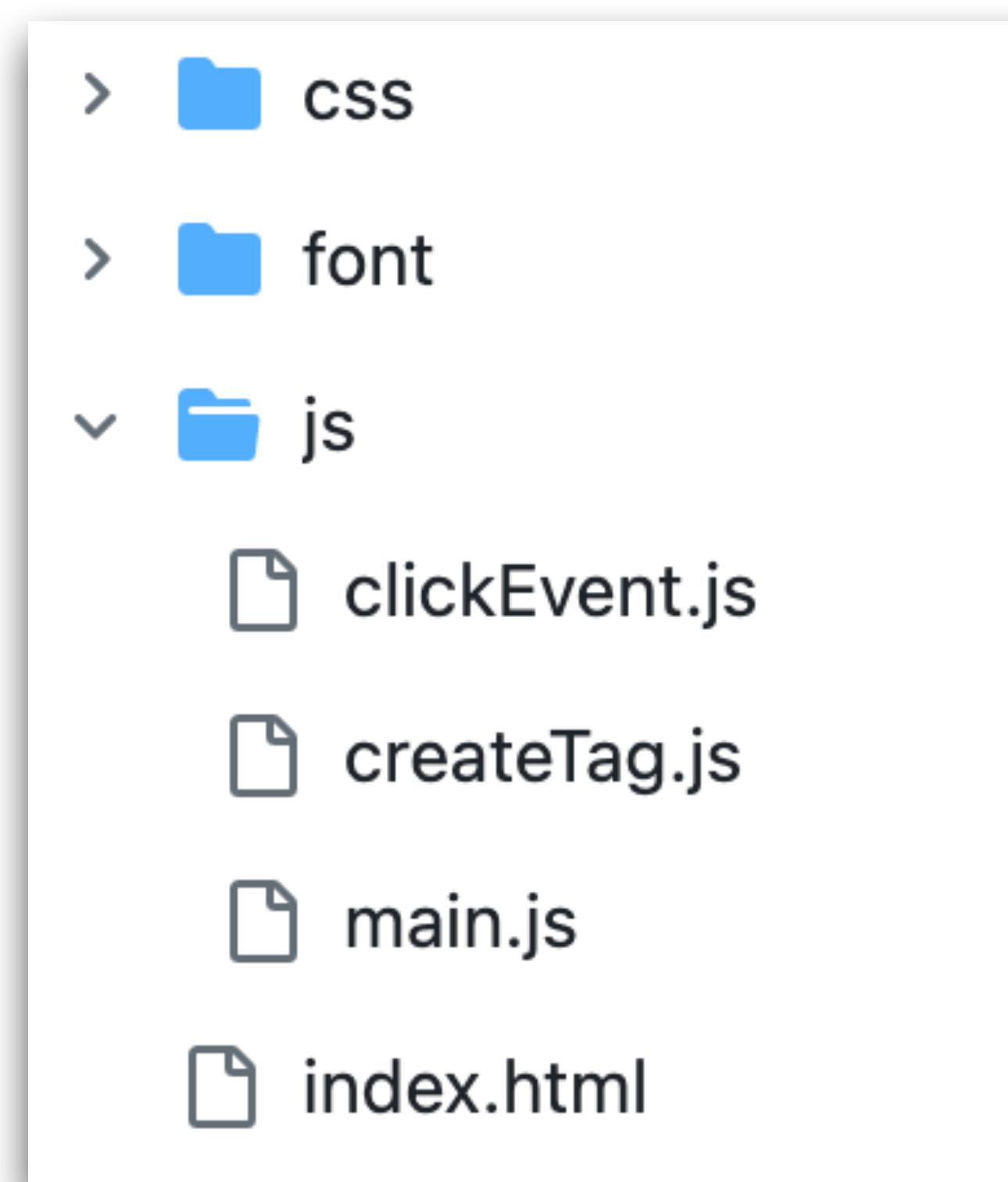
쉽다

강력하다

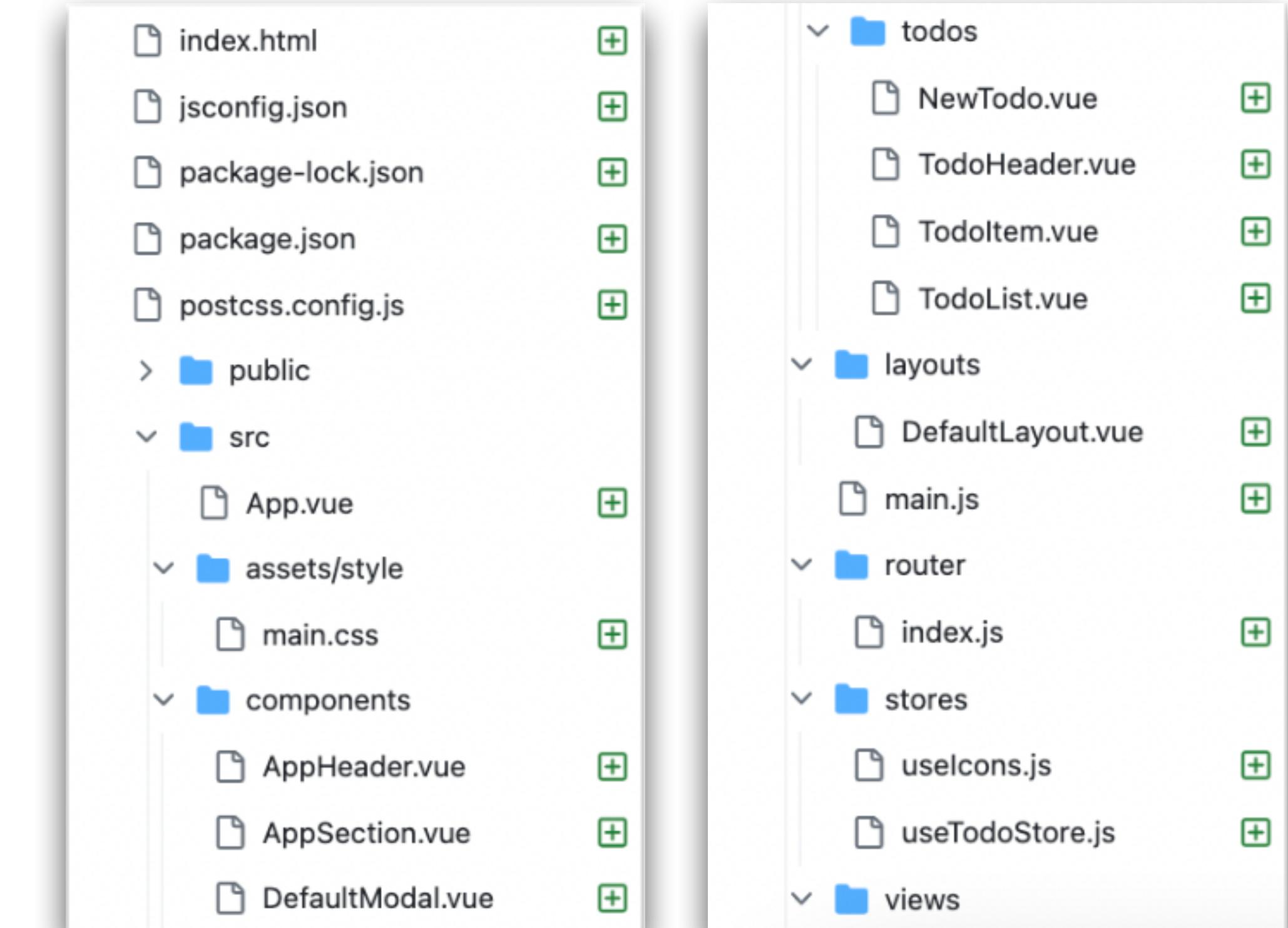
유연하다

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

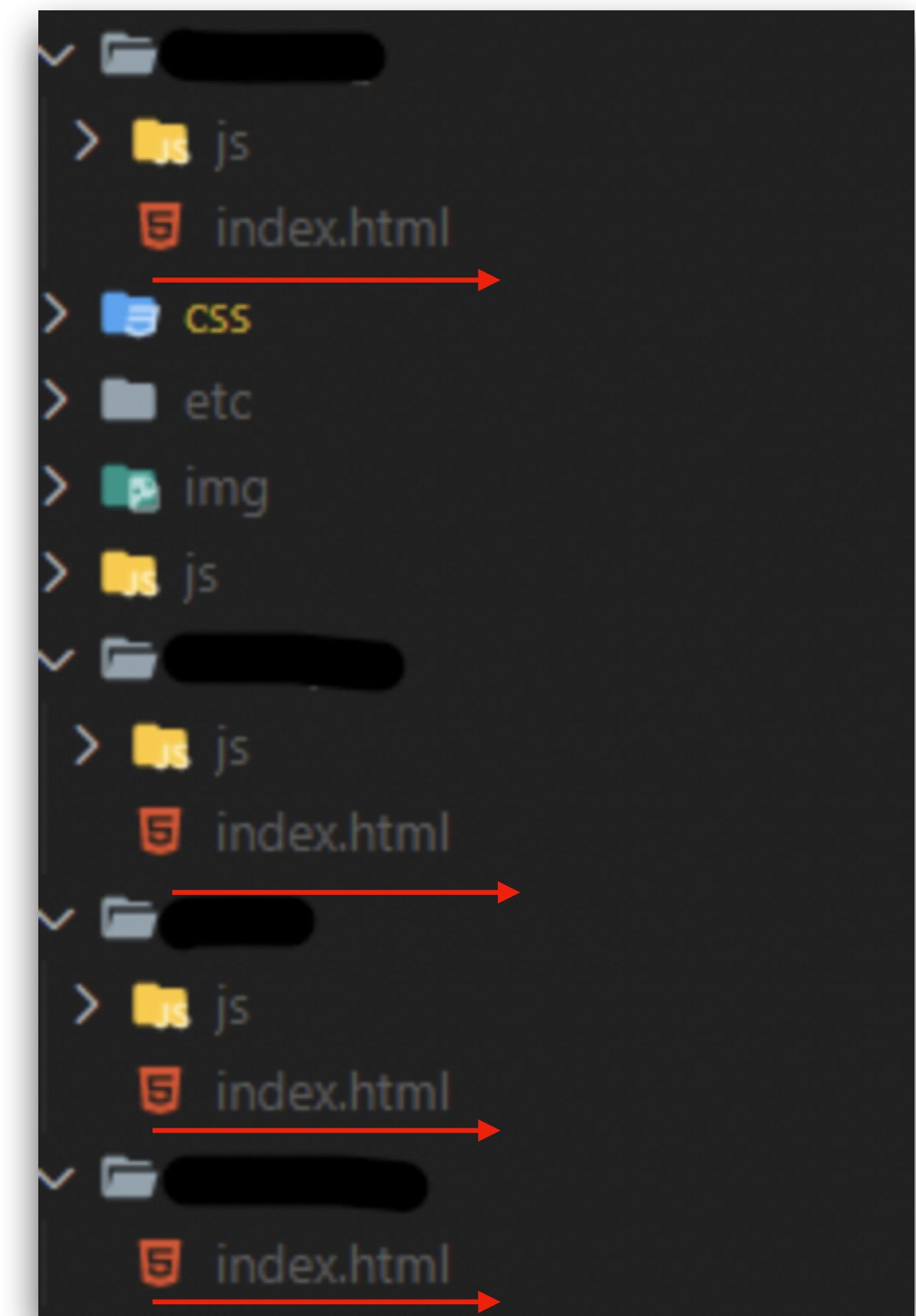
퀴즈

MPA와 SPA의 약어(**주관식**)
- 2개 모두 말하면 정답!

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 장/단점

[장점]

- SEO 관점에서 유리하다.
- 초기 로드 속도가 빠르다.

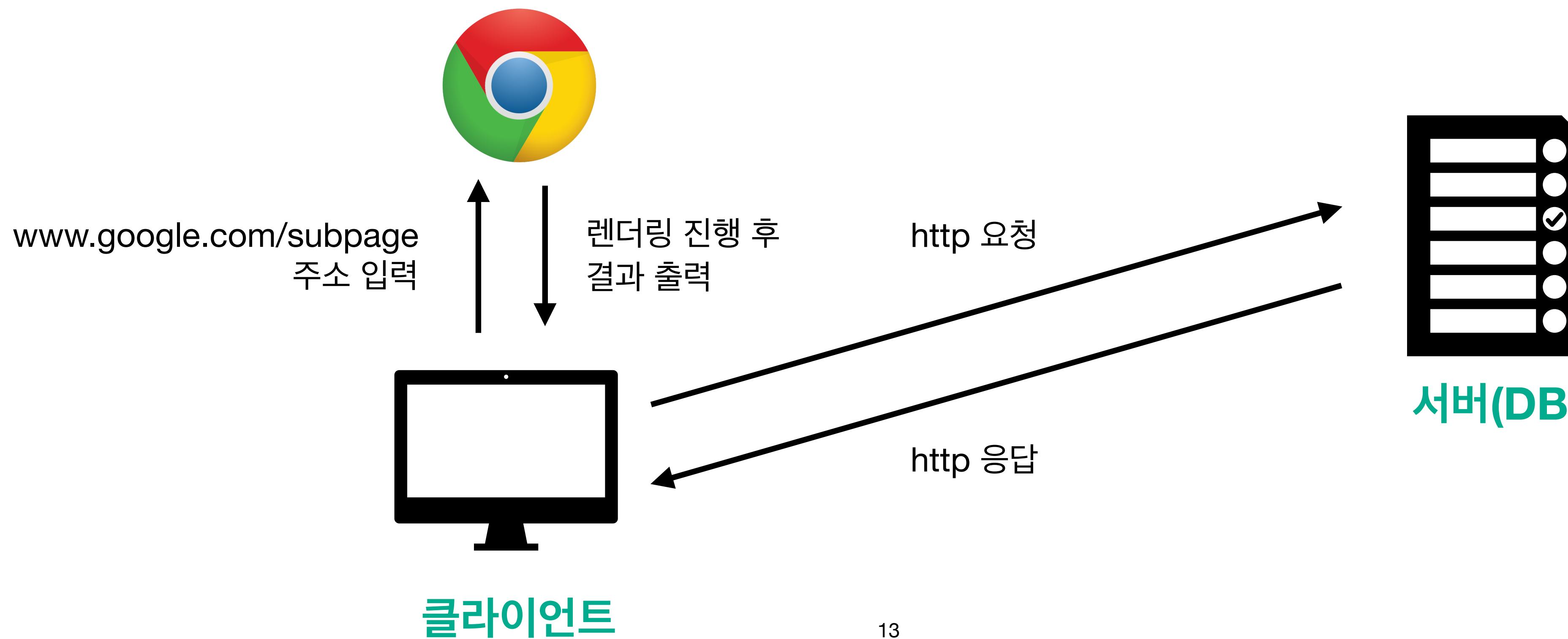
[단점]

- 새로운 페이지로 이동하면, 화면이 깜빡거린다.
- 불필요한 템플릿도 중복되어 로딩된다.

MPA 개념, 장/단점, 렌더링 방식

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

브라우저 렌더링 엔진



SPA 등장 배경, 개념

SPA 등장 배경: MPA의 문제점 해소

퀴즈

- MPA의 단점을 보완하기 위해 등장
- SPA는 수정된 일부분만 하여 화면의 새로고침이 발생 X

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

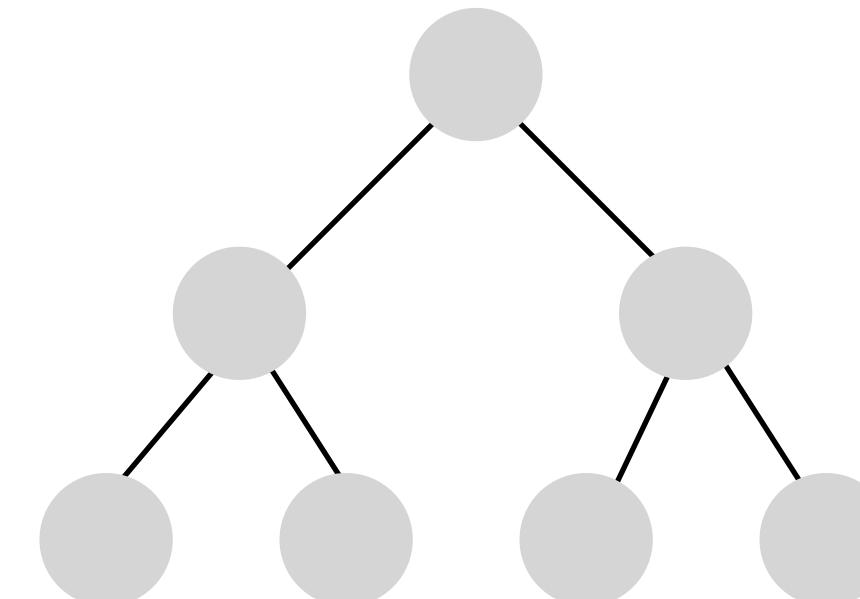
vite v4.3.5 building for production...
✓ 43 modules transformed.

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
```

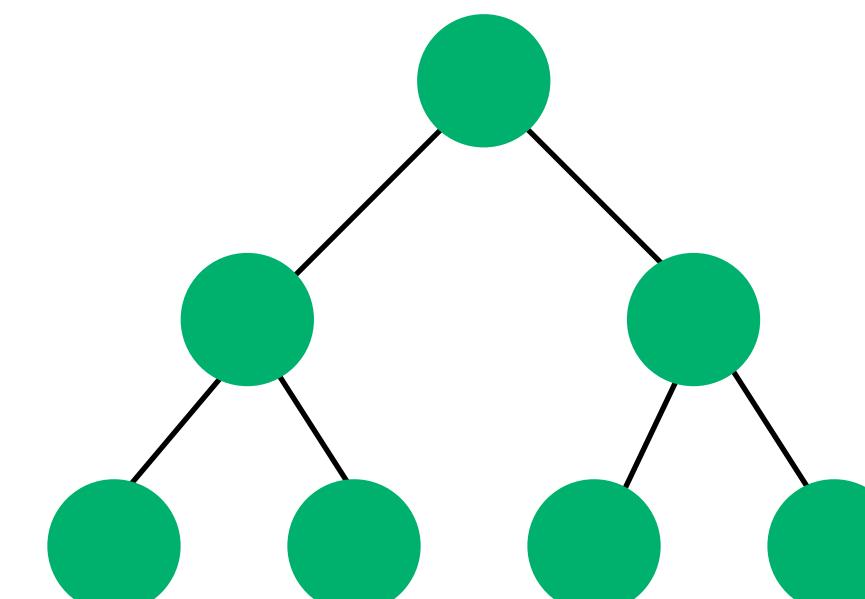
SPA 동작원리

SPA 동작 원리 - 가상 DOM 적용 전

실제 DOM



실제 DOM



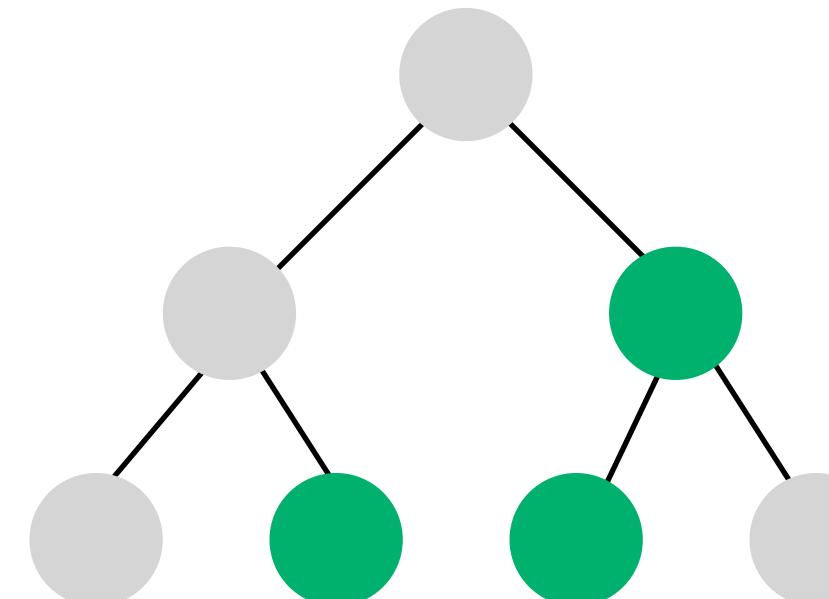
교체 전

교체 후

SPA 동작원리

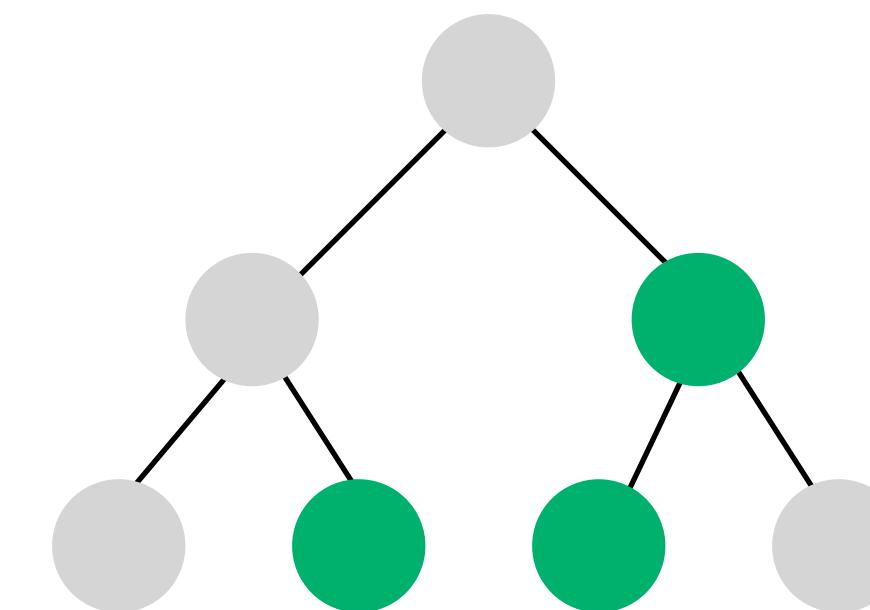
SPA 동작 원리 - 가상 DOM 적용 후

가상 DOM



업데이트할 데이터를
Virtual Dom에 리렌더링 함

실제 DOM



바뀐 부분만
실제 DOM에 적용이 됨



SPA 동작원리

SPA 동작 원리 - 컴포넌트

: 화면을 구성할 수 있는 블록

컴포넌트는 **객체가 한 번에 편리하게 교체될 수 있도록, 하나의 태그 (Vue.js의 경우 <template></template>)**로 감싸져 있음.

```
<template v-for="todo in todos">
  <li v-if="!todo.isComplete">
    {{ todo.name }}
  </li>
</template>

<script setup>
import { ref } from 'vue';
const todos = ref([
  {
    id : 1,
    name: "공부하기",
    category: 'todo',
  },
]);
</script>
```

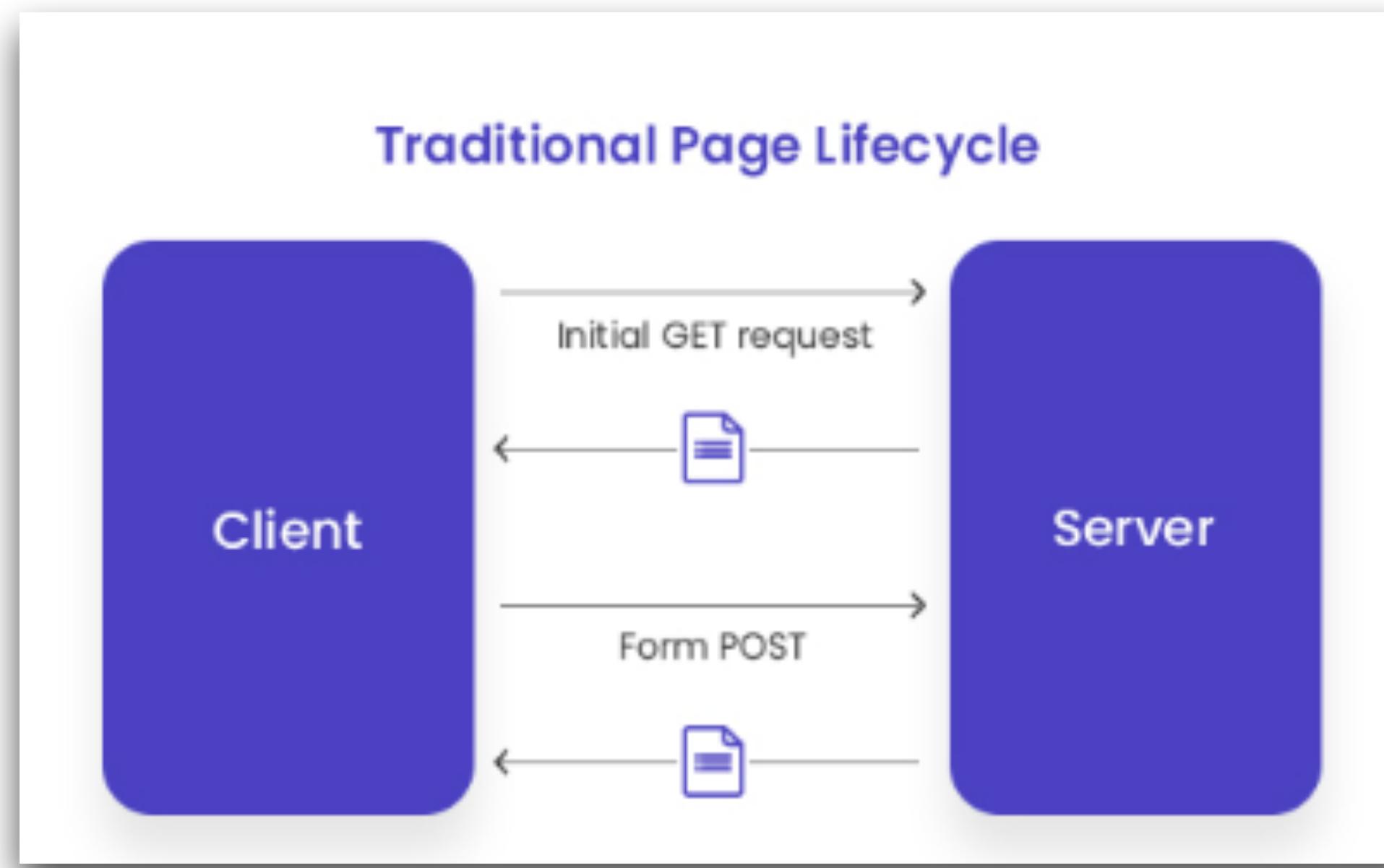
SPA 동작원리

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

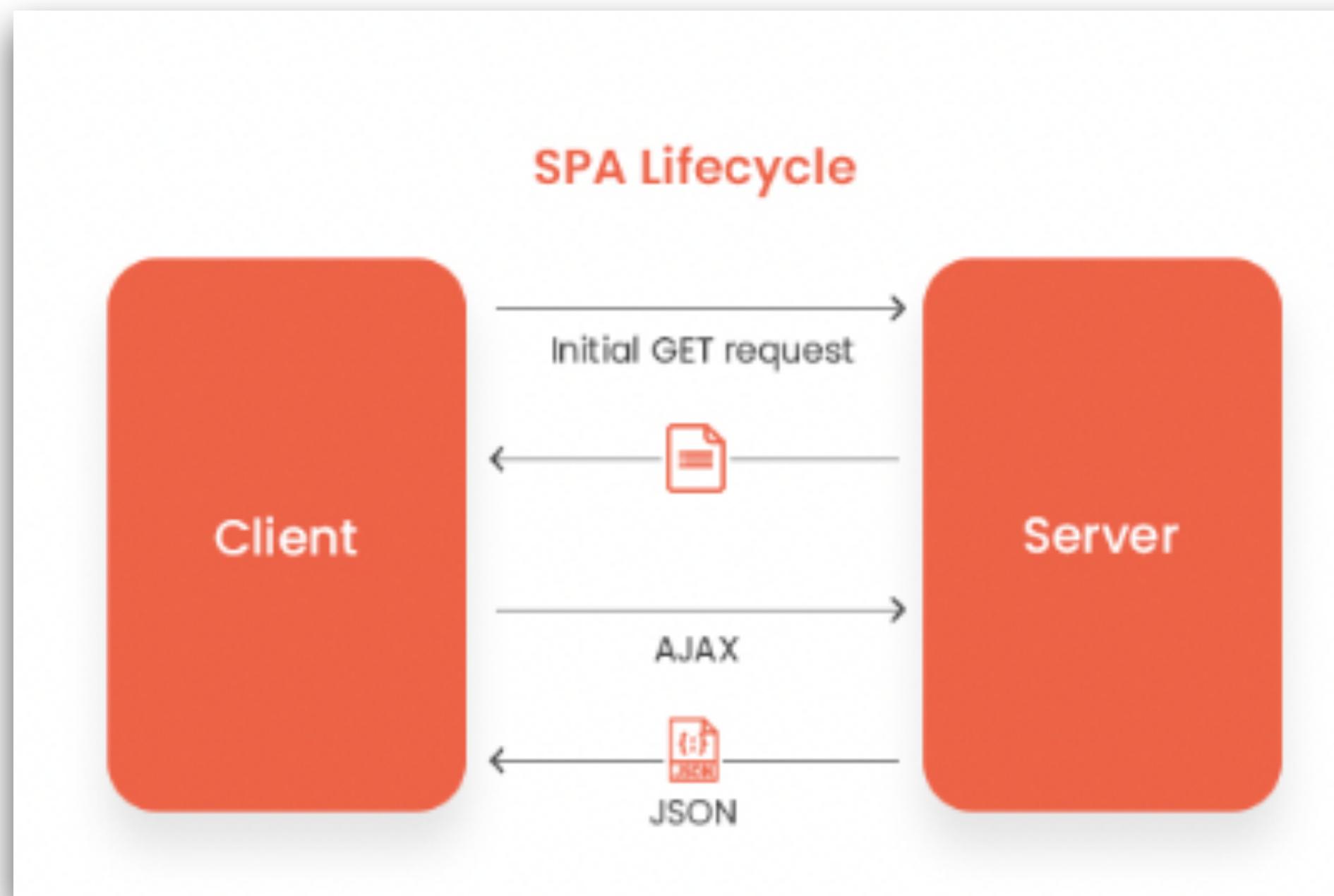
SSR



- 1 Client Server야. 데이터 변경 요청할게!
- 2 Server 그래! 변경 적용해서 정적파일 보내줄게!
- 3 Browser Rendering Engine 서버에서 온 새로운 파일이네? 렌더링 시작!

SPA 동작원리

CSR

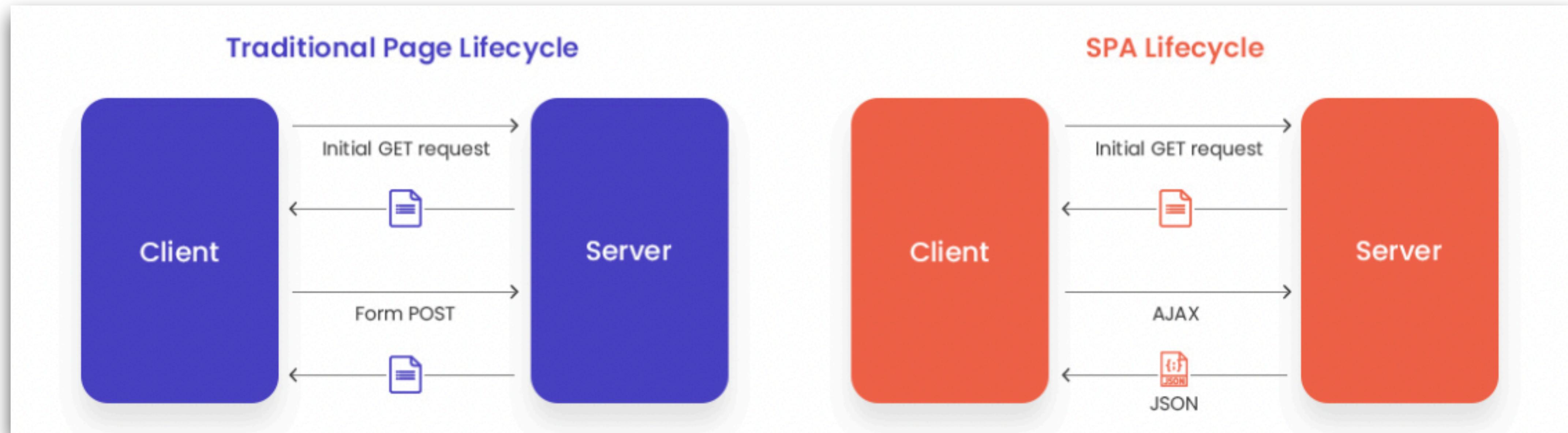


- 1 Client Server야. 데이터 변경 요청할게!
- 2 Server 그래! 여기 요청한 데이터를 JSON으로 보내줄게!
- 3 Client 변경된 데이터를 JS로 바로 적용할게~

SPA 동작원리

SSR

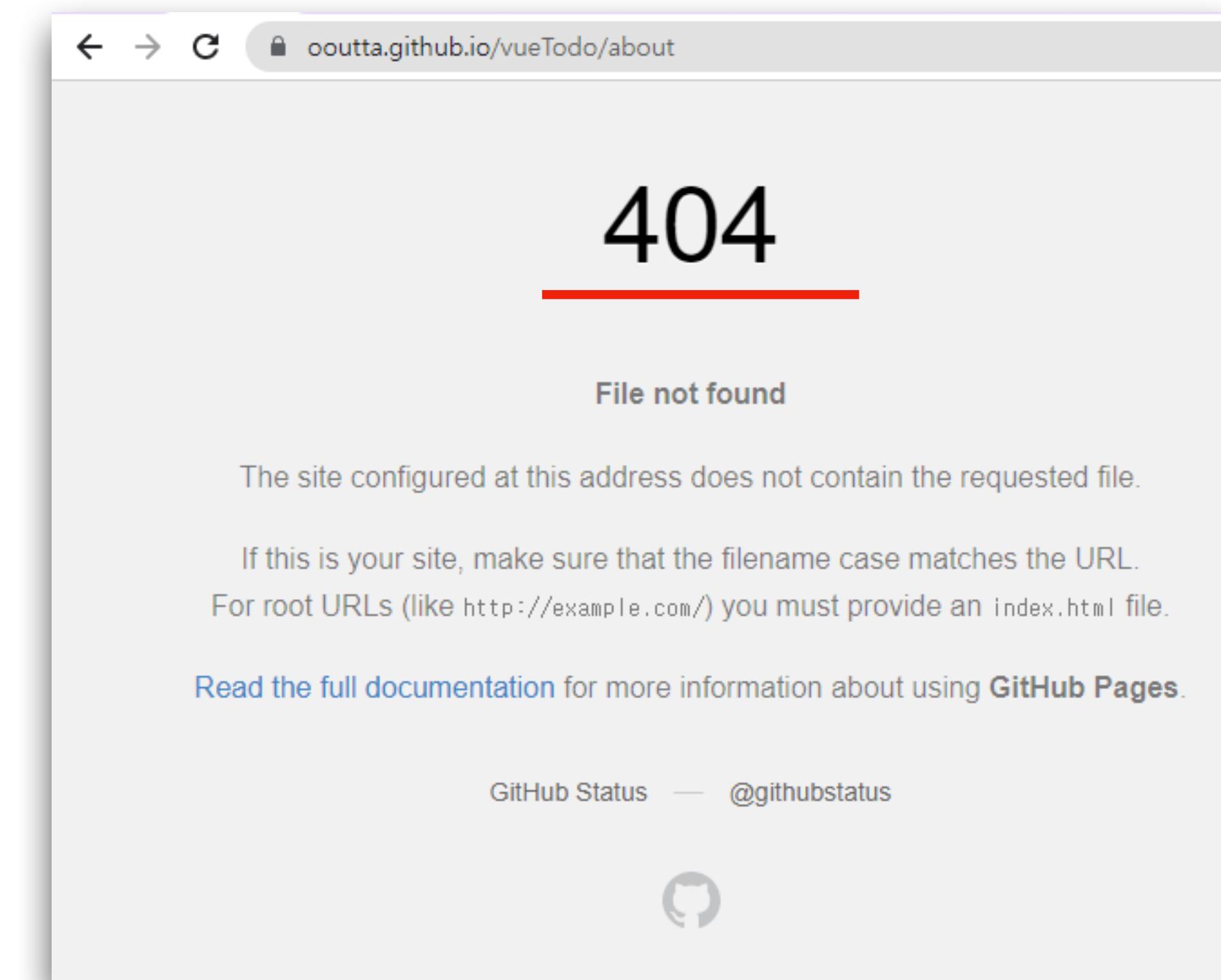
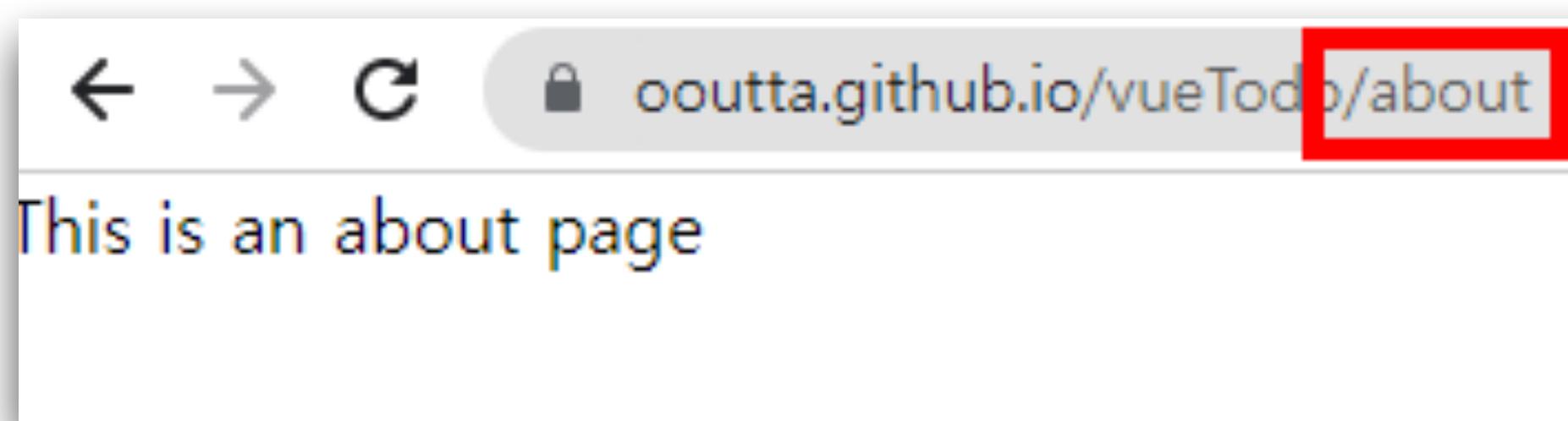
CSR



SPA 동작원리

SPA의 내부 URL에서 새로 고침을 한다면?

-> 404 ERROR 발생! (시연)



/about 경로에서 새로고침 할 시

SPA 동작원리

404 Not Found 해결 방안

- Vue의 라우팅 방식을 해시 모드로 이용
- 서버 측에서 해당 URL 요청이 왔을 경우 접속할 대체 경로 추가

Node.js: `connect-history-api-fallback` API 사용

```
var history = require('connect-history-api-fallback');
var express = require('express');

var app = express();
app.use(history());

history({
  rewrites: [
    { from: '/about/', to: '/index.html'}
  ]
});
```



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 잣은 요청을 하게 됨.	API 통신(데이터 요청)을 할 때에만 서버에서 요청하므로, 요청 횟수가 잦고 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.

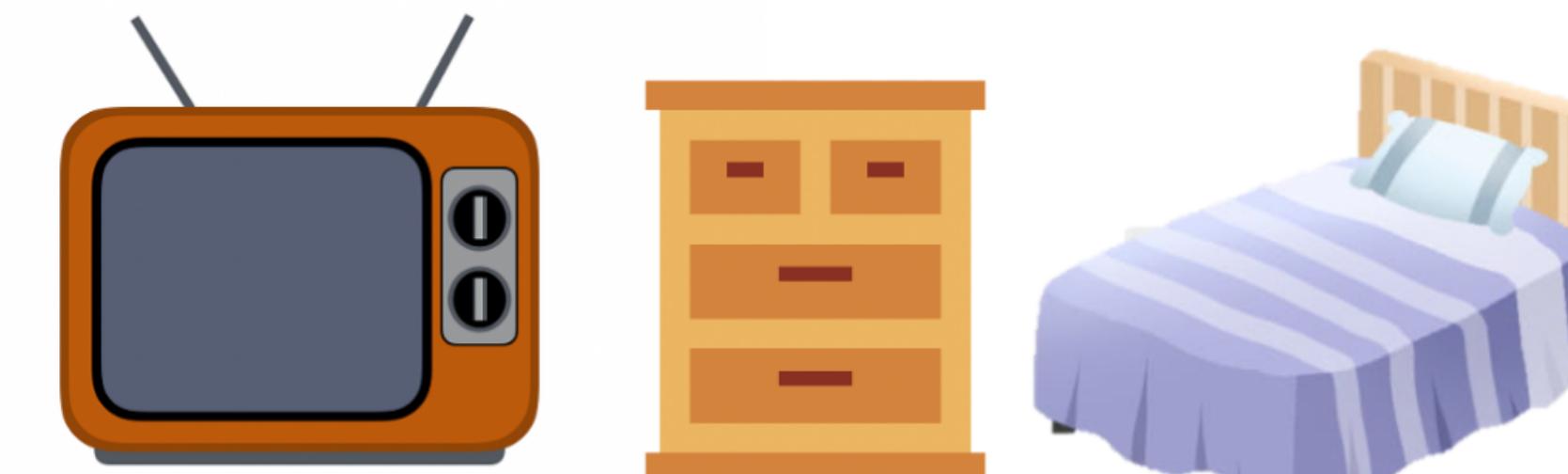
SPA 라이브러리와 프레임워크

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

**Vue & React
모두 공부해야 하나요?**

NO !

결론

1

프로젝트 규모가 작은 경우

Vanilla JS

2

취업하고자 하는 회사의 스택 or 취향

Vue / React

3 Vue

세부 목차

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

Vue.js 특징 3 가지

컴포넌트

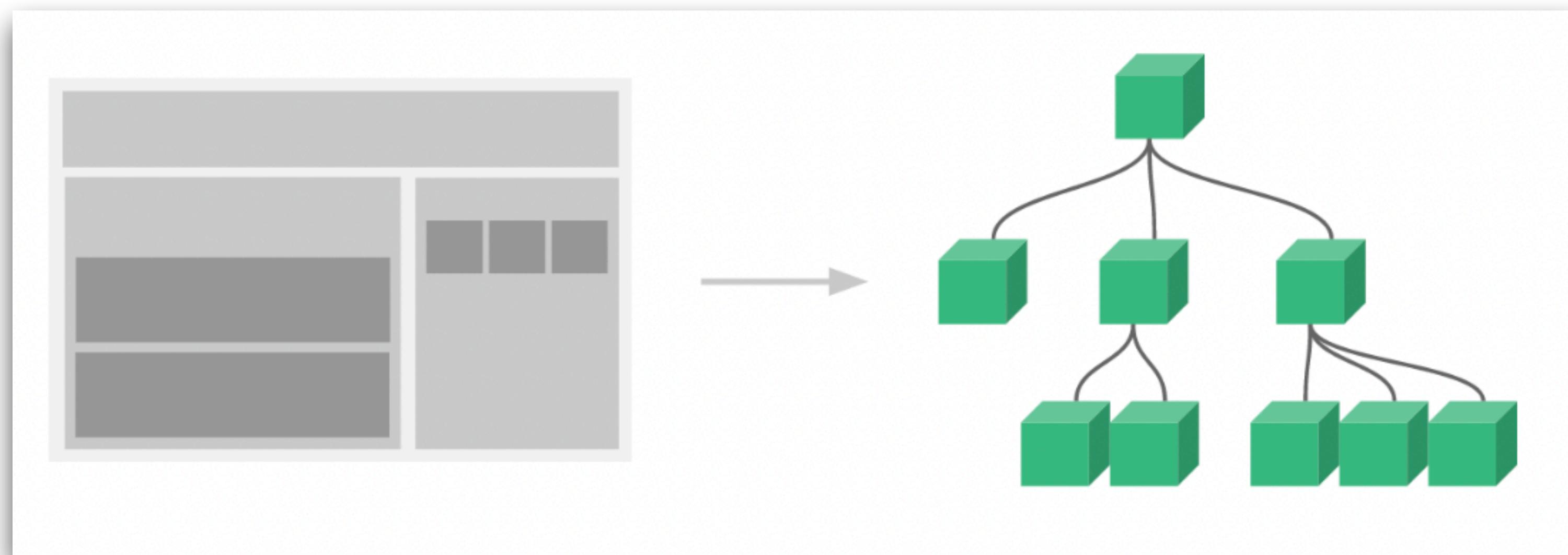
SFC

랜더
파이프 라인

Vue.js 특징 3 가지

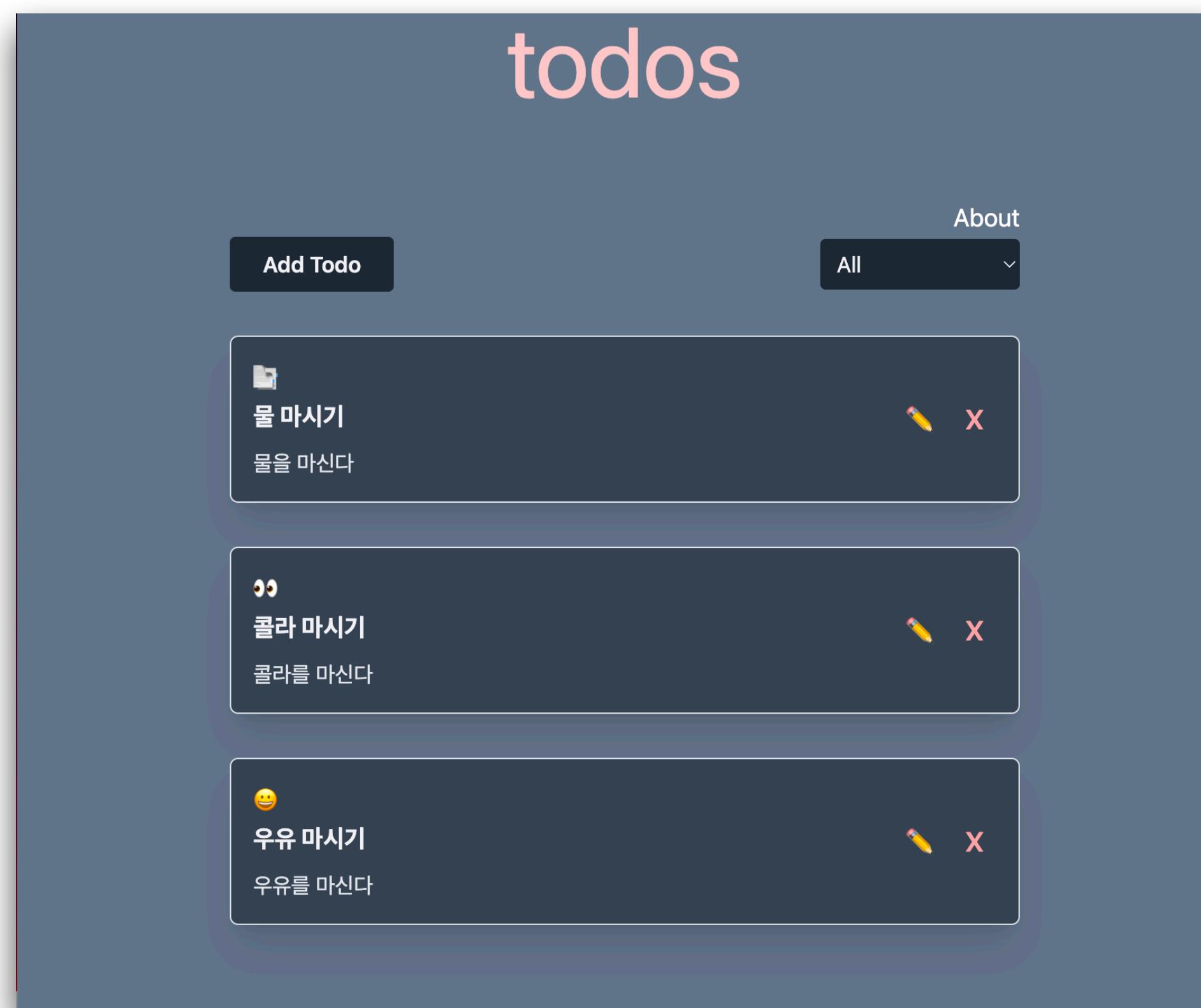
1. 컴포넌트 기반 아키텍처

- 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>
```

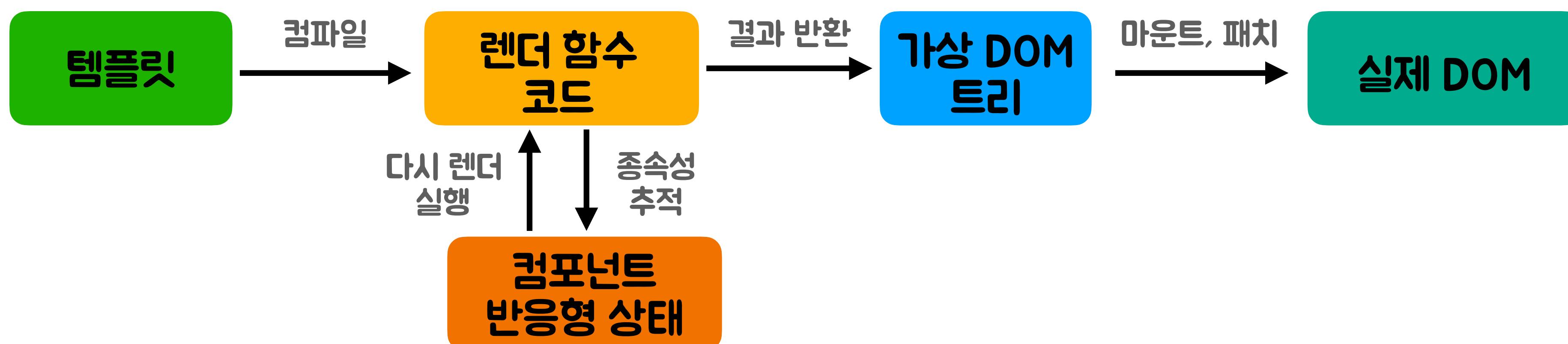
Vue.js 특징 3 가지

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 꺼낼 수 없음.

-> 불편한 부분을 찾고 하지 말아야겠다는
규칙을 만들자!

비슷한 것끼리 분류해서 정리함.

-> 지속적으로 관리가 잘되는 코드를
좋은 아키텍쳐가 필요해!

디자인 패턴과 MVVM 패턴

(소프트웨어) 디자인 패턴 → 효율적인 코드를 만들기 위한 방법론

: 소프트웨어 개발 방법으로 사용되는 패턴으로 과거의 소프트웨어 개발 과정에서 발견된 설계의 노하우를 축적하여 그 방법에 이름을 붙여서 이후에 재사용하기 좋은 형태로 특정 규약을 만들어서 정리한 것.

소프트웨어 아키텍처



소프트웨어 시스템의 기본 구조와 그러한 구조 및 시스템을 만드는 분야

- > 소프트웨어의 내부적인 질을 높이기 위해 고려되는 소프트웨어 구조
- > 소프트웨어의 뼈대 or 고수준의 기반을 담당

디자인 패턴

소프트웨어 디자인에서 공통적으로 발생하는 문제에 대해 재사용 가능한 해결책 -> 각각의 모듈들이 어떤 것을 하는지, 클래스의 범위, 함수의 목적 등 코드 수준의 디자인을 담당

디자인 패턴과 MVVM 패턴

Why Design Patterns(Eg)

- Reusable** → Easy reusable in different projects and applications
- Speed** → Speed up development process we use proven development paradigms
- Standard Solution** → Standard solution for common programming problems
- Maintenance** → Easy maintainable as the code is written in good structure by various patterns
- Loosely Coupled** → It helps to make the change without much affecting

디자인 패턴과 MVVM 패턴

Why Design Patterns(Kr)

재사용성

다양한 프로젝트와 애플리케이션에 쉽게 재사용을 높일 수 있음.

속도

이미 검증된 개발 패러다임을 사용해서 개발자간 커뮤니케이션이 수월함

표준 해결

흔한 프로그래밍 문제 해결에 표준적인 해결책이 될 수 있음.

유지보수성

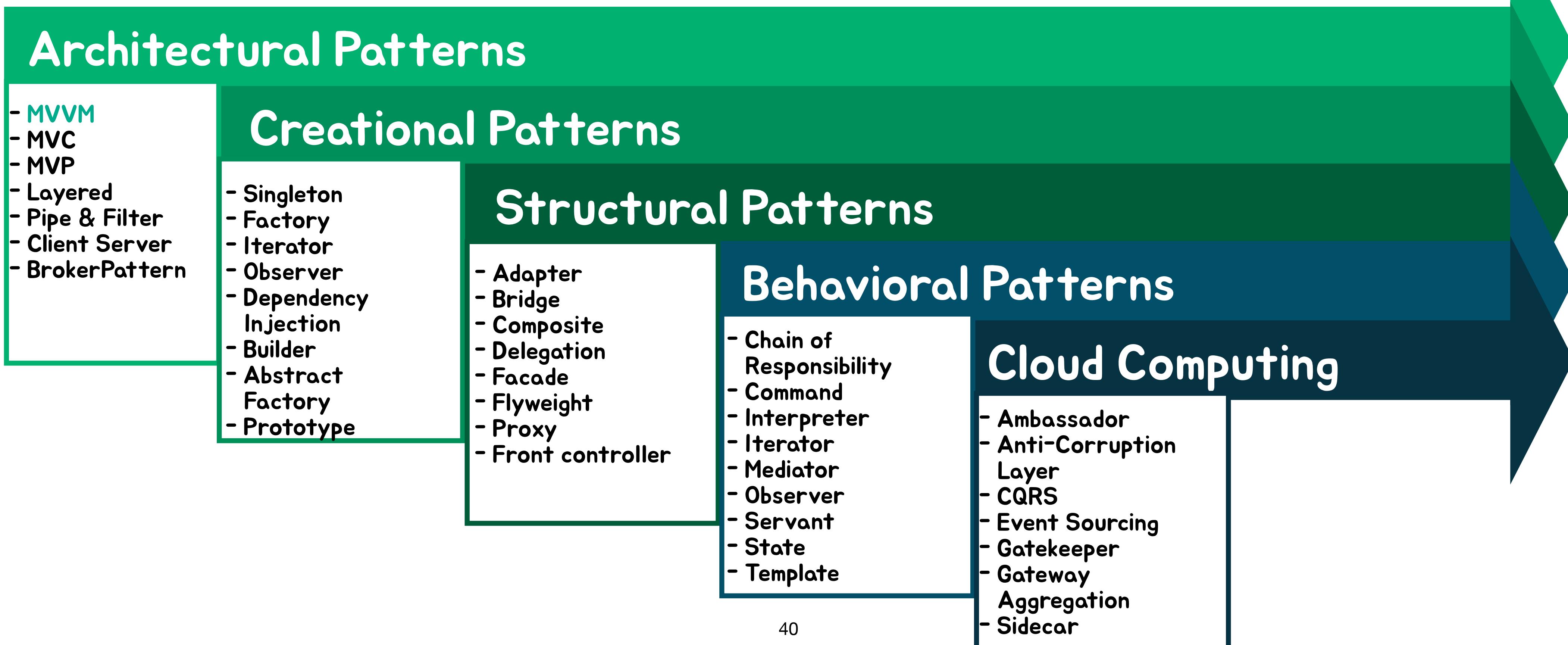
다양한 패턴으로 설계할 때 적용할 경우 코드에서 쉬운 유지보수성이 됨.

느슨한 결합

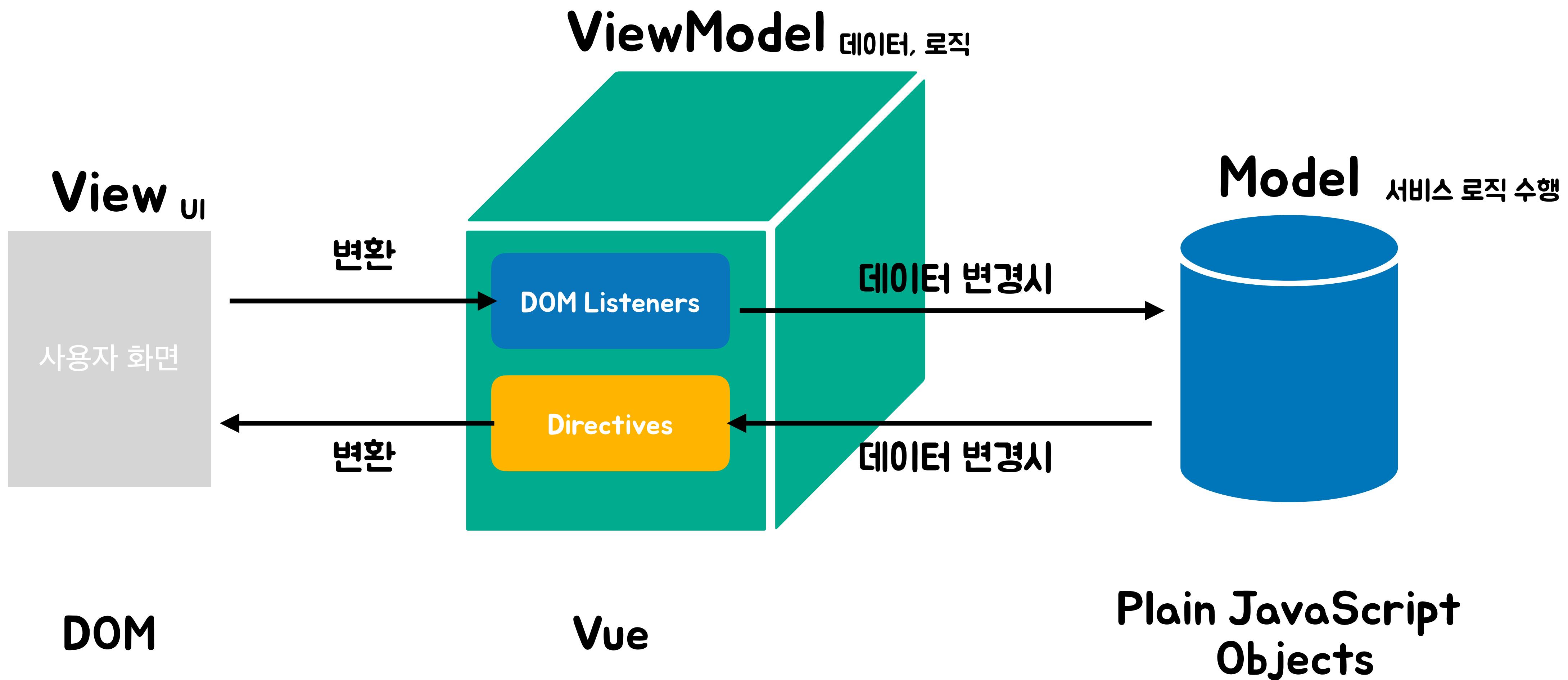
전체 프레임워크를 더욱 안정적으로 만들고 시스템의 유연성을 증가

디자인 패턴과 MVVM 패턴

Types of Design Patterns



디자인 패턴과 MVVM 패턴



Vue.js 핵심 기능 2 가지

1. 선언적 렌더링

간단한 템플릿 구문을 사용해
선언적으로 DOM에 데이터를 렌더링하는 것

<template> 태그 안에 '{{ }}' 와 같은 구문
을 선언해서 html DOM에 데이터를 전달

```
var app = new Vue ({  
  el: '#app',  
  data: {  
    message: '안녕하세요 Vue!'  
  }  
})
```

<app.js>

```
<div id="app">  
  {{ message }}  
</div>
```

<index.html>

Vue.js 핵심 기능 2 가지

2. 반응성

Vue에서는 Javascript의 **Proxy 객체**를 활용하여
반응성을 가진 데이터 바인딩 기능을 제공

반응형 데이터들은 할당과 동시에 **Observe** 기능이 활성화됨

데이터가 변경되었을 때 내부의 **Observe** 기능으로 감지한 변화를
해당 데이터를 사용하는 **VIEW**를 찾아 반영

Vue.js 핵심 기능 2 가지

2. 반응성 적용방법

Vue의 `ref`, `reactive API`를 활용하여 적용

`ref()` : 원시자료형(`number`, `string`), 객체, 배열
`reactive()`: 객체 또는 배열

- ✓ `ref()`가 모든 자료형이 가능하므로 `ref()`를 주로 활용
- ✓ `ref()`를 사용하면 데이터가 `Proxy` 객체로 감싸져서 `refData.value`로 사용

Vue.js 핵심 기능 2 가지

2. 반응성 예제

count에 반응성 데이터가 할당되면서
Observe 기능 활성화(Proxy 생성)

increment()함수를 매개로 count
Proxy의 setter 함수로 데이터 변경

해당 데이터를 사용하는 View(DOM)
을 찾아서 데이터를 반영

```
<script setup>
import { ref } from 'vue'

const count = ref(0);

function increment() {
    count.value++;
}

</script>

<template>
    <h2>Counter</h2>
    <p>{{ count }}</p>
    <button @click='increment'>
        increment
    </button>
</template>
```

App.vue

Counter

0

increment

Counter

2

increment

결론

우리가 Vue를 사용하는 이유

1. 처리해야할 데이터 양이 많아질 경우 SPA를 채택하면 렌더링 속도 향상
2. 컴포넌트 단위의 개발로 코드 유지보수 효율 향상
3. 쉽고 편한 데이터 바인딩으로 작성해야할 코드의 양 감소
4. 러닝커브가 낮아 급하게 레거시 프로젝트를 전환해야 할 때 개발속도 향상

참고문서

- '왜 우리는 Vue.js를 사용하는가?' - 기술 문서
- 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!

디자인 패턴과 MVVM 패턴

GOF 디자인 패턴 대표 3가지

- Creational Patterns(생성 패턴)** **객체 생성**에 관련된 패턴으로, 객체의 생성과 조합을 캡슐화해 특정 객체가 생성하거나 변경을 크게 받지 않도록 **유연성**을 제공함.
- Structural Patterns(구조 패턴)** 클래스나 객체를 **조합**해 더 큰 구조를 만드는 패턴. 서로 다른 인터페이스를 지닌 2개 이상의 객체를 묶음.
- Behavioral Patterns(행위 패턴)** 객체나 클래스 사이의 **알고리즘**이나 **책임 분배**에 관련된 패턴. 객체 사이의 **결합도를 최소화**하는 것에 중점을 둠.