

Vue.js

Programming Language

Index

1 Front-end Dev.

2 SPA (Single Page Application)

3 Vue.js ?

4 Vue App Instance

5 Appendix: Reactivity



Front-end Dev.



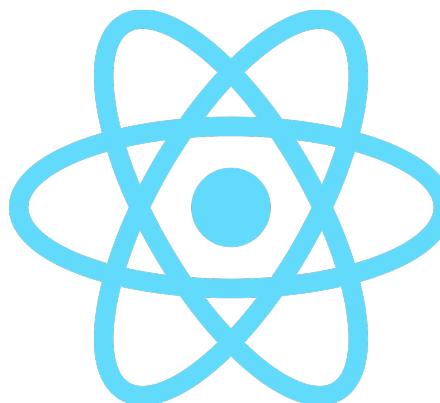
Front-end Development

Front-end Development

- ▶ 웹사이트와 웹 애플리케이션의 사용자 인터페이스(UI)와 사용자 경험(UX)을 만들고 디자인하는 것
- ▶ HTML, CSS, JavaScript 등을 활용하여 사용자가 직접 상호작용하는 부분을 개발

Client-side Framework

- ▶ 클라이언트 측에서 UI와 상호작용을 개발하기 위해 사용되는 JavaScript 기반 프레임워크



Client-side Framework

⚡ Client-side Framework 필요성

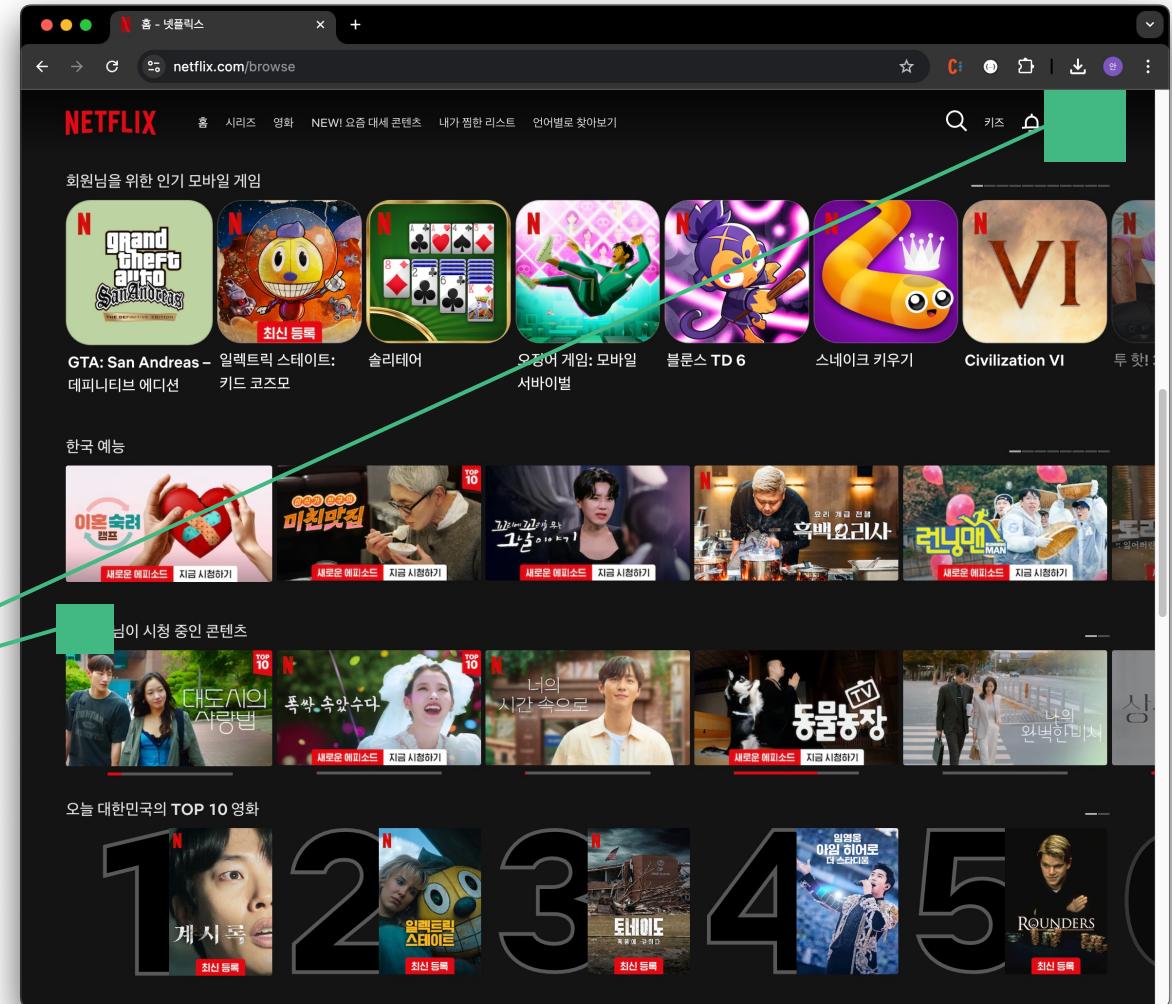
- ▶ 사용자는 웹에서 문서만을 읽는 것이 아닌 음악을 스트리밍하고 영화를 보고, 원거리에 있는 사람들과 텍스트 및 영상 채팅을 통해 즉시 통신하고 있음
- ▶ 즉 웹의 개념이 단순히 **무언가를 읽는 곳**에서 **무언가를 하는 곳**으로 변경되었음
- ▶ 이처럼 현대적이고 복잡한 대화형 웹 사이트를 “**웹 애플리케이션(web applications)**”이라 부름
- ▶ JavaScript 기반의 Client-side frameworks의 출현으로 매우 동적인 대화형 애플리케이션을 훨씬 더 쉽게 구축할 수 있게 됨
- ▶ 또한 화면에서 처리해야 하는 데이터의 양이 많아짐

Client-side Framework

Client-side Framework 필요성

- ▶ 웹 어플리케이션에서 사용하는 데이터가 많아지고 하나의 화면에서 반복적으로 사용하는 데이터가 늘어남
∴ 하나의 데이터가 바뀌면 여러 영역의 데이터가 동시에 업데이트가 적용되어야 함
- ▶ 애플리케이션의 기본 데이터를 안정적으로 추적하고 업데이트 (렌더링, 추가, 삭제 등)하는 도구가 필요
- ▶ 애플리케이션의 상태를 변경할 때마다 일치하도록 UI를 업데이트해야 한다는 것

계정의 닉네임을 변경할 경우 동시에
변경되어야 함



Client-side Framework

Client-side Framework 필요성

- ▶ 닉네임 변경 시 여러 영역을 수정해야 함
- ▶ Vanilla JS만 이용하여 해결한다면?

중복 코드가 과도하게 발생

```
<label for="nick">닉네임: </label>
<input type="text" id="nick" />
<hr />
<h1>안녕하세요 <span id="nick1"></span>님</h1>
<div>
  <span id="nick2"></span>님이 시청 중인 콘텐츠
</div>
<div>
  <span id="nick3"></span>님 취향 저격!!
</div>
<div>
  <span id="nick4"></span>님만을 위한 추천 목록
</div>
```

```
const nick = document.querySelector("#nick");
let initNick = "guest";

const nick1 = document.querySelector("#nick1");
const nick2 = document.querySelector("#nick2");
const nick3 = document.querySelector("#nick3");
const nick4 = document.querySelector("#nick4");

nick1.textContent = initNick;
nick2.textContent = initNick;
nick3.textContent = initNick;
nick4.textContent = initNick;

nick.addEventListener("change", (event) => {
  const newNick = event.target.value;

  nick1.textContent = newNick;
  nick2.textContent = newNick;
  nick3.textContent = newNick;
  nick4.textContent = newNick;
});
```

SPA
(Single Page Application)



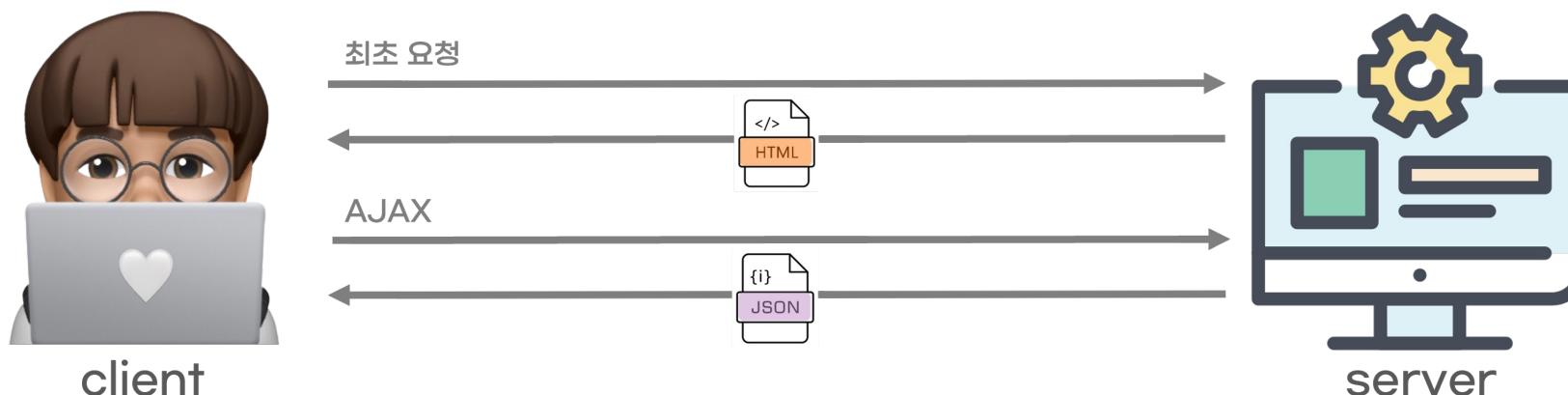
SPA (Single Page Application)

SPA

- ▶ 페이지 한 개로 구성된 웹 어플리케이션
- ▶ 웹 애플리케이션의 초기 로딩 후 새로운 페이지 요청 없이 동적으로 화면을 갱신하여 사용자와 상호작용하는 웹 애플리케이션
::::> CSR (Client Side Rendering)

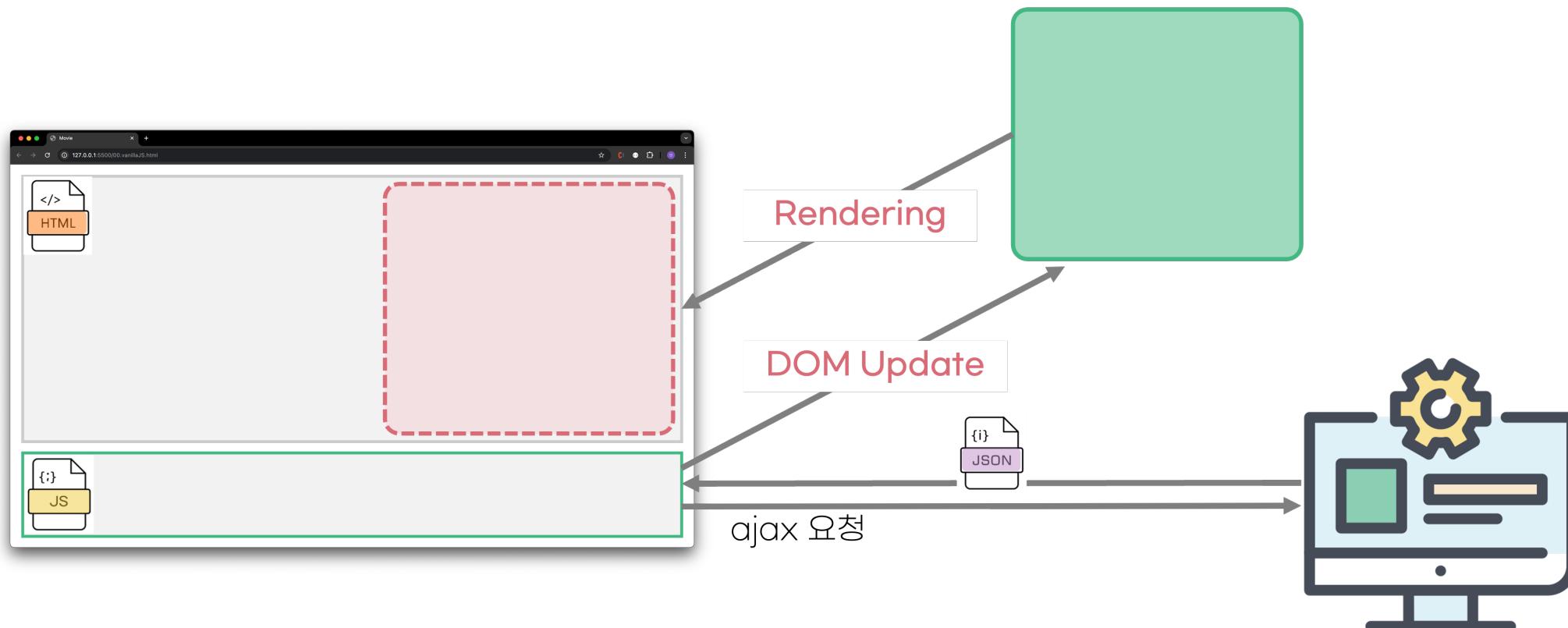
SPA 동작

- ▶ 서버로부터 필요한 모든 정적 HTML을 처음에 한번 가져옴
- ▶ 브라우저가 페이지를 로드하면 Vue 프레임워크는 각 HTML 요소에 적절한 JavaScript 코드를 실행
(이벤트에 응답, 데이터 요청 후 UI 업데이트 등)
 - 페이지 간 이동 시, 페이지 갱신에 필요한 데이터만을 JSON으로 전달받아 페이지 일부 갱신
 - Google Maps, 인스타그램 등의 서비스에서 갱신 시 새로 고침이 없는 이유



CSR

- ▶ 최초에 브라우저는 페이지에 필요한 최소한의 HTML 페이지와 JavaScript를 다운로드
- ▶ 클라이언트(브라우저)가 요청 (ajax)에 따라 필요한 데이터만 응답 받아 렌더링하는 방식
- ▶ 클라이언트의 요청 시 서버는 data (JSON, XML)만 제공
- ▶ 클라이언트 (브라우저)가 화면을 그리는 주체가 됨



CSR (Client Side Rendering)

CSR 장점과 단점

장점	단점
처음 로딩 후 동적으로 빠르게 렌더링이 되기 때문에 사용자 입장에서 UX가 좋음	초기 페이지 로딩이 전체 페이지에 대한 모든 문서 파일을 받다 보니 SSR에 비해 로딩 속도가 느림
처음에 스크립트 파일을 로드하기 때문에 서버에게 요청하는 횟수가 적어 서버의 부담이 적어지고 새로 고침이 발생하지 않아 클라이언트는 네이티브 앱과 비슷한 경험을 할 수 있음	포털사이트의 검색엔진 크롤러가 사이트에 대한 데이터를 정확하게 수집 못하는 경우가 발생 할 수 있음 ...> 검색엔진최적화(SEO)에 대한 추가 보완 작업이 필요
Front-end와 Back-end의 분리가 명확해짐 Front-end는 UI 렌더링 및 사용자 상호 작용 처리를 담당 & Back-end는 데이터 및 API 제공을 담당 대규모 애플리케이션을 더 쉽게 개발하고 유지 관리 가능	SEO(검색 엔진 최적화) 문제 페이지를 나중에 그려 나가는 것이기 때문에 검색에 잘 노출되지 않을 수 있음

SPA (Single Page Application)에서 사용하는 방식

Vue.js?



<https://vuejs.org>

Vue.js 공식 사이트

The screenshot shows the official Vue.js website (vuejs.org) displayed in a dark-themed browser window. The main title "The Progressive JavaScript Framework" is prominently featured in large, bold, blue and green letters. Below it is a subtitle: "An approachable, performant and versatile framework for building web user interfaces." A navigation bar at the top includes links for "Docs", "API", "Playground", "Ecosystem", "About", "Sponsor", "Experts", and "Register". There is also a promotional banner for "Vueconf.US" with the text "Use VUEJSDOCS \$200 off". At the bottom, there are sections for "Approachable", "Performant", and "Versatile" features, each with a brief description. A "Special Sponsor" section features the logo for "monterail".

The Progressive
JavaScript Framework

An approachable, performant and versatile framework for building web user interfaces.

Special Sponsor  monterail Official Vue & Nuxt Partner

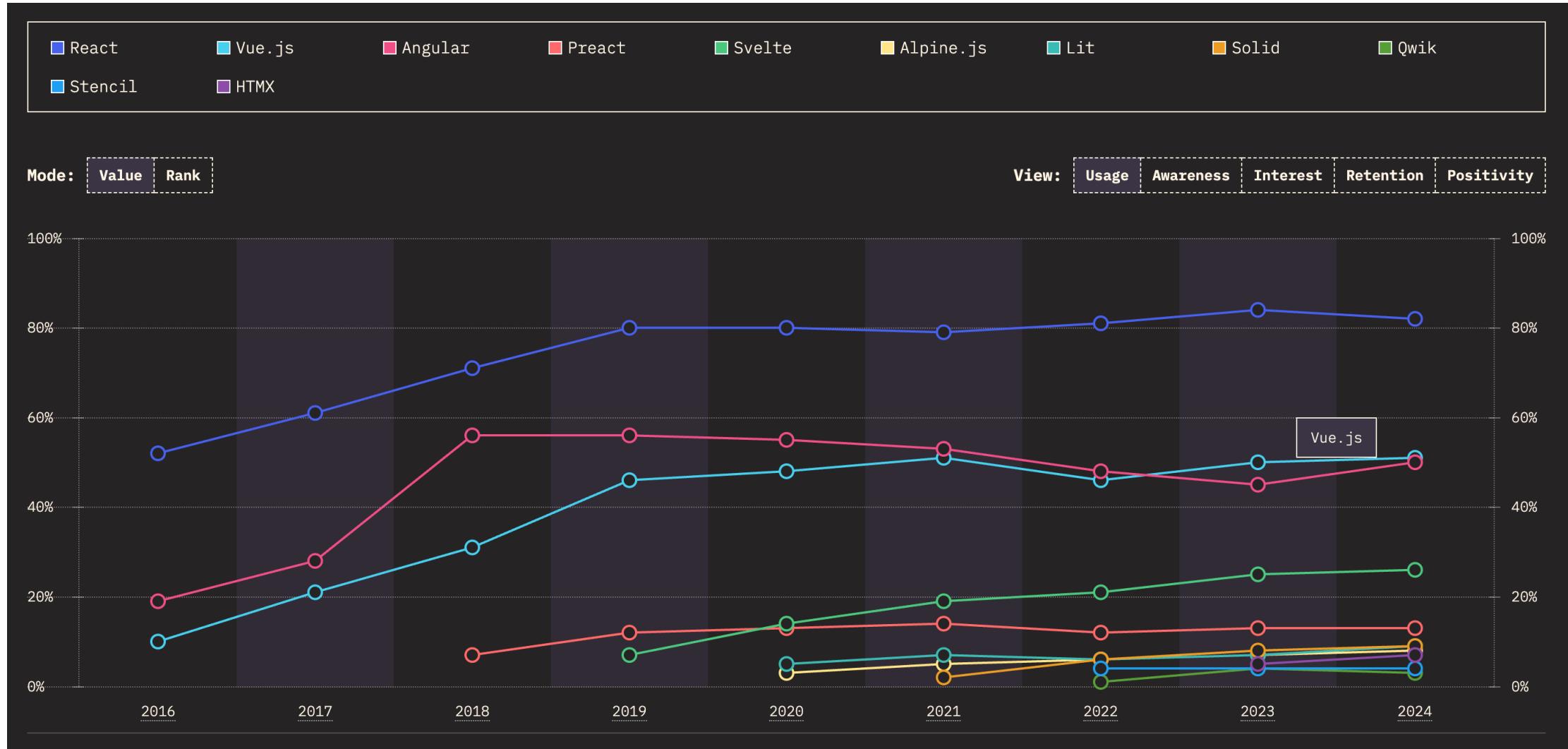
Approachable
Builds on top of standard HTML, CSS and JavaScript with intuitive API and world-class documentation.

Performant
Truly reactive, compiler-optimized rendering system that rarely requires manual optimization.

Versatile
A rich, incrementally adoptable ecosystem that scales between a library and a full-featured framework.

<https://2024.stateofjs.com/ko-KR/libraries/front-end-frameworks>

Front-end Framework Trend



Vue.js ?

▶ 사용자 인터페이스 개발을 위한 Progressive Framework

- **Progressive**: 웹과 네이티브 앱의 이점을 모두 수용하고 표준 패턴을 사용해 개발

▶ SPA(Single Page Application) 개발을 위한 Frontend Framework

- 최초 웹사이트에 접속했을 때, 모든 페이지에서 필요한 자원 (html, javascript, css, img, ..)을 로딩
- 페이지 이동시 전체 페이지가 바뀌는 방식이 아닌 변경이 필요한 부분만 렌더링

▶ Angular의 양방향 데이터 통신과 React의 Virtual DOM의 장점을 수용

▶ Vue의 두 가지 핵심 기능

- **선언적 렌더링 (Declarative Rendering)**: Vue는 표준 HTML을 템플릿 문법으로 확장하여 JavaScript 상태(State)를 기반으로 화면에 출력될 HTML을 선언적(declaratively)으로 작성 할 수 있음
- **반응성 (Reactivity)**: Vue는 JavaScript 상태 (State) 변경을 추적하고, 변경이 발생하면 DOM을 효율적으로 업데이트하는 것을 자동으로 수행



- APPENDIX 참고 (30page)

Vue.js 특징

▶ Approachable (접근성)

- 직관적이고 배우기 쉬움

▶ Versatile (유연성)

- Component를 통한 재사용성으로 개발 시간의 단축과 더 좋은 코드를 작성

▶ Performant (고성능)

- Angular (데이터 바인딩)와 React(Virtual DOM)의 장점을 모두 수용

MVVM Pattern

- ▶ Model + View + ViewModel

- ▶ Model: 순수 자바스크립트 객체

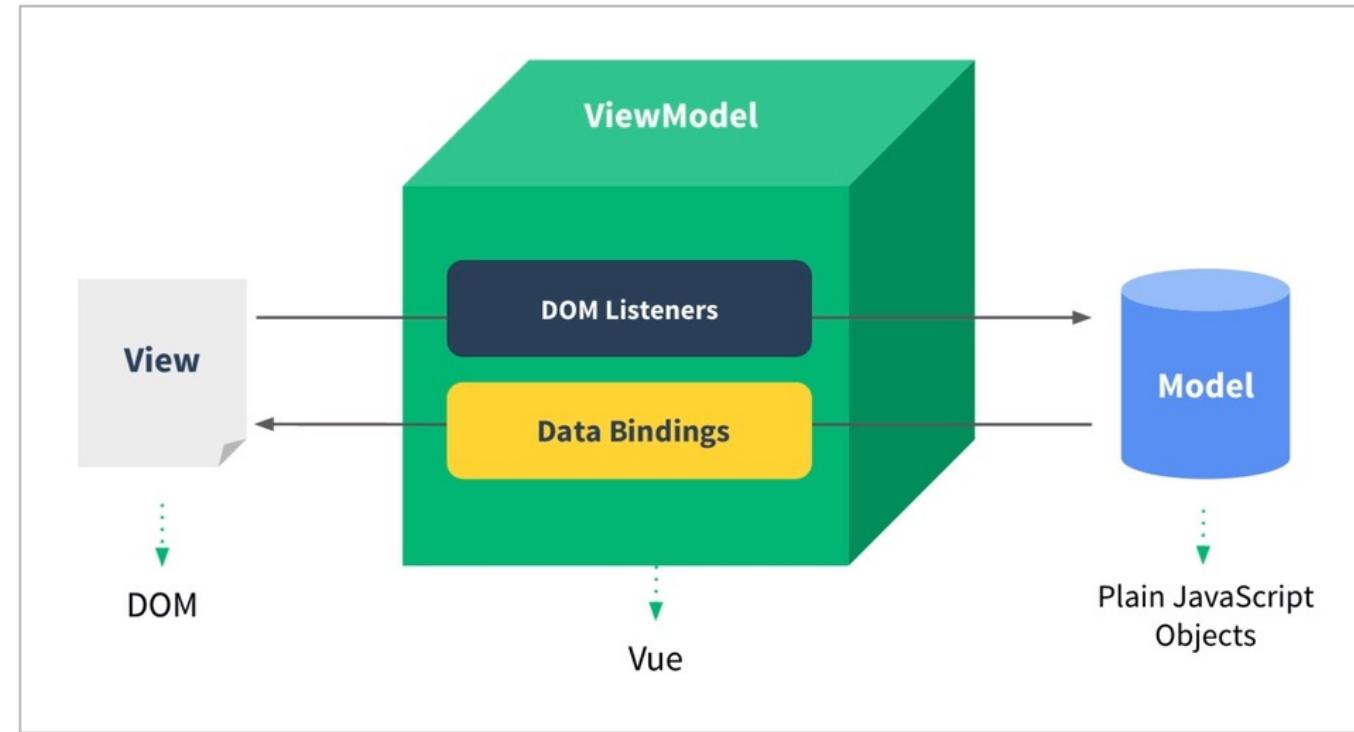
- ▶ View: 웹페이지의 DOM

- ▶ ViewModel: Vue의 역할

- 기존 자바스크립트에서 view에 해당하는 DOM으로 접근하거나 수정하기 위해서는 jQuery와 같은 외부 library를 이용
- Vue는 View와 Model을 연결하고 자동으로 바인딩하므로 양방향 통신을 가능하게 함

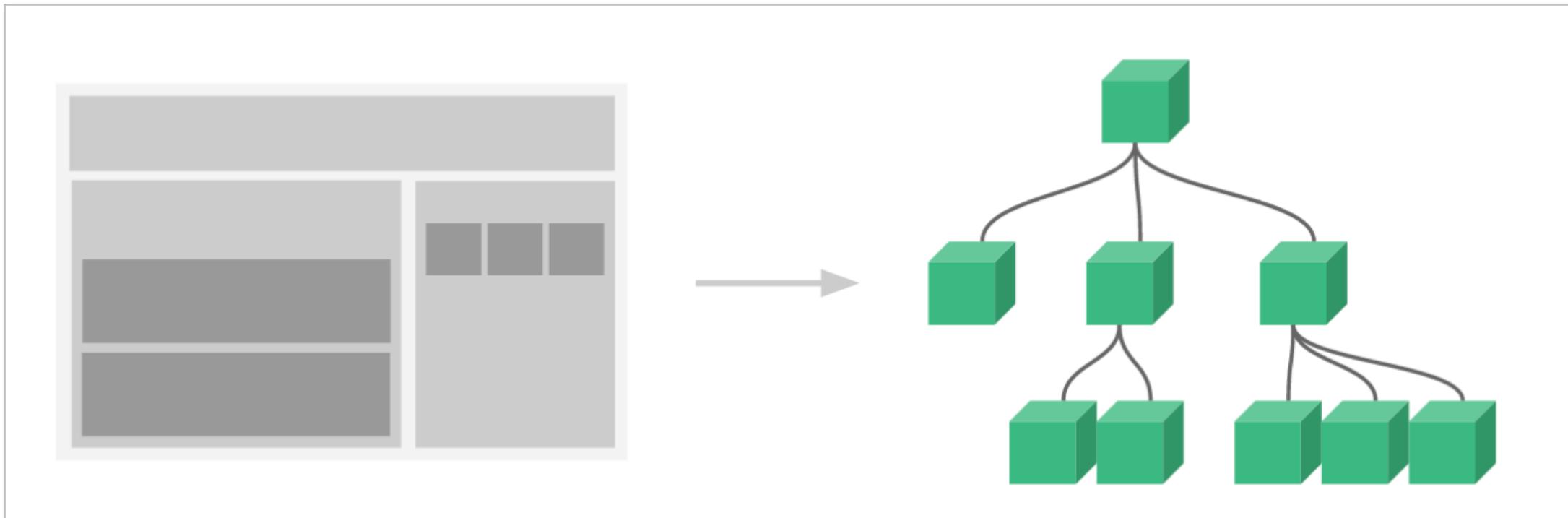


- 참고: Vue 3부터는 공식 문서와 소개에서 MVVM이라는 용어를 적극적으로 사용하지 않게 되었음
 - 컴포지션 API 도입으로 상태 관리와 로직 구성 방식이 더 유연해짐
 - 함수형 프로그래밍 패러다임 요소 강화
 - 단순한 MVVM보다 더 넓은 범위의 애플리케이션 아키텍처 지원



Component

- ▶ 웹 화면 구성
 - html element (button, input, image, ..)와 이 element들의 결합으로 만들어진 **기능을 가진 UI**
- ▶ component : 웹 화면 구성 중 다른 화면에서도 **재사용**할 수 있는 구조로 개발
- ▶ 다른 component에서 import해서 사용 가능



<https://ko.vuejs.org/guide/introduction.html#single-file-components>

⚡ SFC (Single File Component)

- ▶ Vue에서는 ComponentName.vue 파일로 생성
- ▶ html(template) + javascript(script) + css(style)를 하나의 파일에서 관리

```
1 <template>
2   | <div>{{ message }}</div>
3 </template>
4
5 <script>
6 export default {
7   components: {},
8   data() {
9     return {
10       message: 'Component 방식 Vue.js 입니다.',
11     }
12   },
13   setup() {},
14   created() {},
15   methods: {},
16 }
17 </script>
18
19 <style scoped>
20 div {
21   font-weight: bold;
22 }
23 </style>
```

vue의 기본 구조

<https://ko.vuejs.org/guide/introduction.html#api-styles>

Vue3?

- ▶ Vue component는 Options API와 Composition API 두 가지 스타일로 작성 가능

```

1  <template>
2  | <button @click="increment">숫자 세기: {{ count }}</button>
3  </template>
4
5  <script>
6  export default {
7  // data()에서 반환된 속성들은 반응적인 상태가 되어 `this`에 노출됩니다.
8  data() {
9  return {
10    count: 0,
11  }
12},
13
14 // methods는 속성 값을 변경하고 업데이트 할 수 있는 함수.
15 // 템플릿 내에서 이벤트 리스너로 바인딩 될 수 있음.
16 methods: {
17   increment() {
18     this.count++
19   },
20 },
21
22 // 생명주기 흑(Lifecycle hooks)은 컴포넌트 생명주기의 여러 단계에서 호출됩니다.
23 // 이 함수는 컴포넌트가 마운트 된 후 호출됩니다.
24 mounted() {
25   console.log(`숫자 세기의 초기값은 ${this.count} 입니다.`)
26 },
27 }
28 </script>

```

Options API

```

1  <template>
2  | <button @click="increment">숫자 세기: {{ count }}</button>
3  </template>
4
5  <script setup>
6  import { ref, onMounted } from 'vue'
7
8 // 반응적인 상태의 속성
9 const count = ref(0)
10
11 // 속성 값을 변경하고 업데이트 할 수 있는 함수.
12 function increment() {
13   count.value++
14 }
15
16 // 생명 주기 흑
17 onMounted(() => {
18   console.log(`숫자 세기의 초기값은 ${count.value} 입니다.`)
19 })
20 </script>

```

Composition API

<https://chromewebstore.google.com/detail/vuejs-devtools/nhdogjmejjiglipccpnnnanhbledajbpd?hl=ko>

Vue.js 개발 환경

- ▶ Vue.js devtools 설치

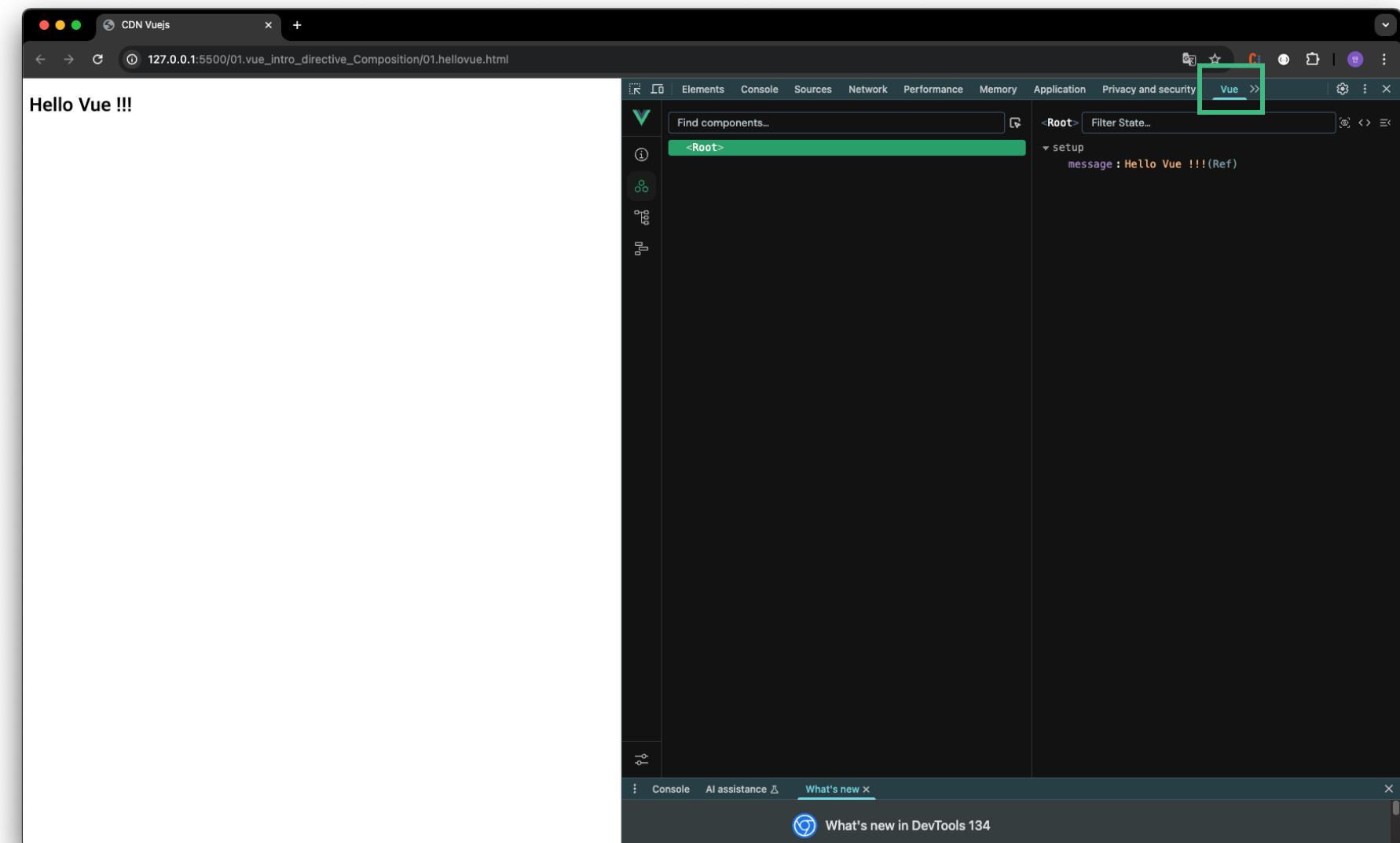
The screenshot shows the Chrome Web Store page for the 'Vue.js devtools' extension. At the top, it says 'chrome 웹 스토어'. On the right, there's a gear icon and an account dropdown for '@gmail.com'. Below the header, the navigation path is '홈 > 확장 프로그램 > Vue.js devtools'. The main content area features the Vue.js logo (a green triangle), the text 'Vue.js devtools', the URL 'vuejs.org', and a rating of '★★★★★ 1,875'. To the right is a blue button labeled 'Chrome에서 삭제'. Below this, there's a link to '개발자 도구' and a note that it's used by '2,000,000+명'.



- 참고: Vite project 생성시 포함되어 있음

Vue.js 개발 환경

▶ 개발자 도구에서 설치 확인



<https://ko.vuejs.org/guide/quick-start>

Vue.js installation

- ▶ CDN

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

- ▶ Vite

```
$ npm create vue@latest
```

```
✓ Project name: ... <your-project-name>
✓ Add TypeScript? ... No / Yes
✓ Add JSX Support? ... No / Yes
✓ Add Vue Router for Single Page Application development? ... No / Yes
✓ Add Pinia for state management? ... No / Yes
✓ Add Vitest for Unit testing? ... No / Yes
✓ Add an End-to-End Testing Solution? ... No / Cypress / Nightwatch / Playwrig
✓ Add ESLint for code quality? ... No / Yes
✓ Add Prettier for code formatting? ... No / Yes
✓ Add Vue DevTools 7 extension for debugging? (experimental) ... No / Yes
```

```
Scaffolding project in ./<your-project-name>...
Done.
```

👋 Hello Vue.js

Vue.js입니다.

Hello Vue !!!

```

1  <!DOCTYPE html>
2  <html lang="ko">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>CDN Vuejs</title>
7      <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
8    </head>
9    <body>
10   <div id="app">
11     <h1>CDN 방식 Vue.js입니다.</h1>
12     <h2>{{ message }}</h2>
13   </div>/#app

14
15   <script>
16     const { createApp, ref } = Vue;
17
18     let model = { message: "Hello Vue !!!" };

19
20     const app = createApp({
21       setup() {
22         const message = ref(model.message);
23         return {
24           message,
25         };
26       },
27     });

28
29     app.mount("#app");
30   </script>
31 </body>
32 </html>

```

View : 유저인터페이스

Model : 도메인 데이터

ViewModel : 상태와 연산

<https://ko.vuejs.org/style-guide>

Style Guide

- ▶ 우선 순위 A: 필수 (오류 방지)
- ▶ 우선 순위 B: 강력히 권장
- ▶ 우선 순위 C: 권장
- ▶ 우선 순위 D: 주의해서 사용하기

Vue App Instance



Vue App Instance

<https://ko.vuejs.org/guide/essentials/application.html>

Vue App Creating

- ▶ `createApp()` 함수를 사용하여 app instance를 생성

```
import { createApp } from 'Vue'

const app = createApp({
  /* 최상위 컴포넌트 옵션 */
})
```

- ▶ 최상위 컴포넌트

- SFC를 사용하는 경우 일반적으로 다른 파일에서 루트 컴포넌트를 가져옴

```
import { createApp } from 'Vue'
// 싱글 파일 컴포넌트(SFC)에서 최상위 컴포넌트 앱을 가져옵니다.
import App from './App.vue'

const app = createApp(App);
```

Vue App Instance

Vue App Mounting

- ▶ App Instance는 .mount() 메서드가 호출될 때까지 아무 것도 렌더링하지 않음
- ▶ .mount() 메서드는 반드시 앱의 환경설정 및 asset이 모두 등록 완료된 후에 호출되어야 함

```
<div id="app">
  <h2>{{ message }}</h2>
</div>/#app
```

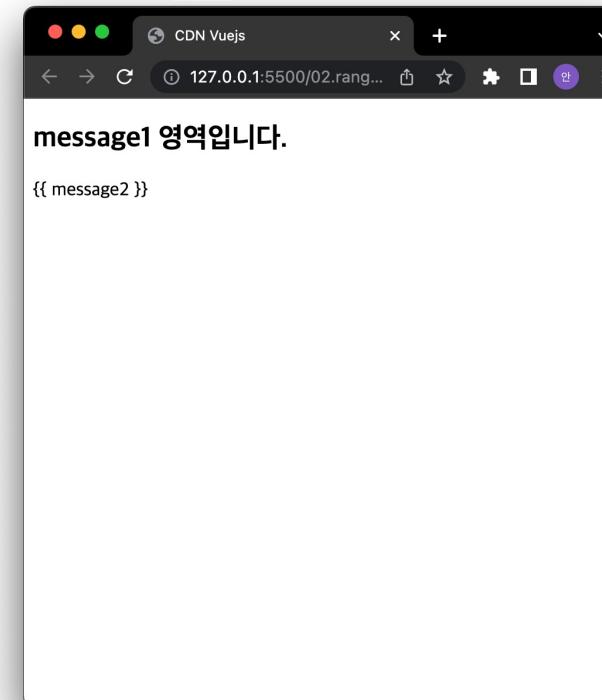
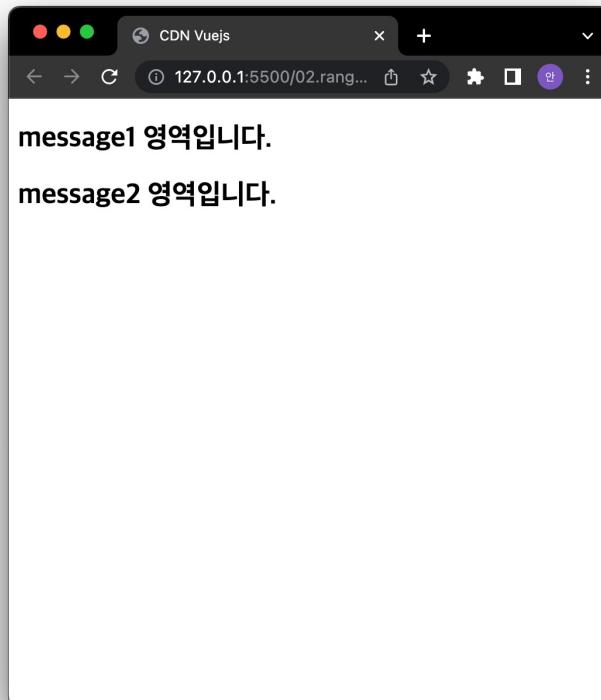
```
const { createApp, ref } = Vue;

const app = createApp({
  setup() {
    const message1 = ref("message 영역입니다.");
    return {
      message1,
    };
  },
});

app.mount("#app");
```

Multiple App Instance

- ▶ 동일 페이지 내 app instance는 여러 개 가능
- ▶ createApp API를 사용하면 여러 Vue App instance를 생성할 수 있으며, 각각은 구성 및 전역 데이터에 대한 고유 범위를 갖음



```
<div id="app1">
|   <h2>{{ message1 }}</h2>
</div>/#app1
<div id="app2">
|   <h2>{{ message2 }}</h2>
</div>/#app2
<script>
  const { createApp, ref } = Vue;

  createApp({
    setup() {
      const message1 = ref("message1 영역입니다.");
      return {
        message1,
      };
    },
  }).mount("#app1");

  createApp({
    setup() {
      const message2 = ref("message2 영역입니다.");
      return {
        message2,
      };
    },
  }).mount("#app2");
</script>
```

Appendix: Reactivity



<https://ko.vuejs.org/guide/essentials/reactivity-fundamentals>

💡 반응형

- ▶ 반응형이란 component의 상태(state)가 바뀌었을 때 자동으로 component의 DOM을 변경해 주는 것
- ▶ 즉, data가 바뀌면 개발자가 화면 update를 위한 코드를 직접 구현할 필요 없이 vue가 알아서 자동으로 업데이트 해 줌
- ▶ **선언적 렌더링** : view와 logic을 가진 각각의 component를 구현할 때 component의 상태와 어떻게 렌더링될지 선언 (`<template>`)하고 상태가 바뀌는 로직만 관리 (`<script>`)해주면 렌더링은 신경 쓸 필요 없음
- ▶ 반응형 상태로 만들 수 있는 함수 : **ref()**, **reactive()**

선언적 렌더링 (Declarative Rendering)

💡 선언적 렌더링

- ▶ 선언적 렌더링의 핵심은 “**상태 (데이터)가 UI를 결정한다**”는 점이다
- ▶ 전통적인 방식에서는 개발자가 직접 DOM을 조작해야 했지만,
- ▶ Vue.js에서는 데이터만 변경해도 자동으로 화면이 업데이트됨
- ▶ 정리: 개발자가 최종적으로 원하는 UI 상태를 선언하면, 프레임워크 (Vue.js)가 그 상태를 자동으로 DOM에 반영하는 접근 방식

💡 선언적 렌더링 장점

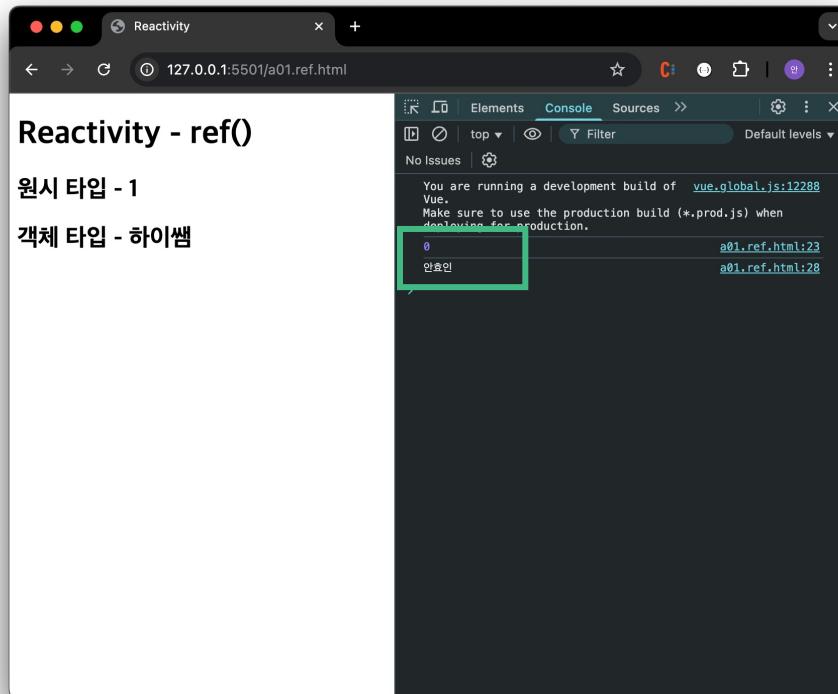
- ▶ 코드가 간결해짐
- ▶ UI 로직에 집중할 수 있음 (데이터가 변경되면 UI는 자동 업데이트)
- ▶ 상태 관리가 훨씬 예측 가능해짐



- 예시 : 데이터 목록 관리
 - 명령형: 데이터 추가/삭제 시 DOM을 직접 조작
 - 선언적: 배열만 변경하면 Vue가 자동으로 화면 업데이트

ref()

- ▶ 단일 값의 반응형 참조를 생성하는 메서드 (ref === reactive reference)
- ▶ 모든 원시 타입과 객체를 반응성을 가진 참조형 데이터로 생성
- ▶ ref() 함수 호출 시 .value를 가진 객체 리턴
- ▶ methods에서는 .value 프로퍼티로 접근,
template에서는 .value를 불일 필요 없음



```

<div id="app">
  <h1>Reactivity - ref()</h1>
  <h2>원시 타입 - {{ count }}</h2>
  <h2>객체 타입 - {{ user.name }}</h2>
</div>/#app

```

.value 사용 X

```

<script>
  const { createApp, ref } = Vue;

  const app = createApp({
    setup() {
      // 기본 타입
      const count = ref(0);
      console.log(count.value); // 0
      count.value++; // 값 변경

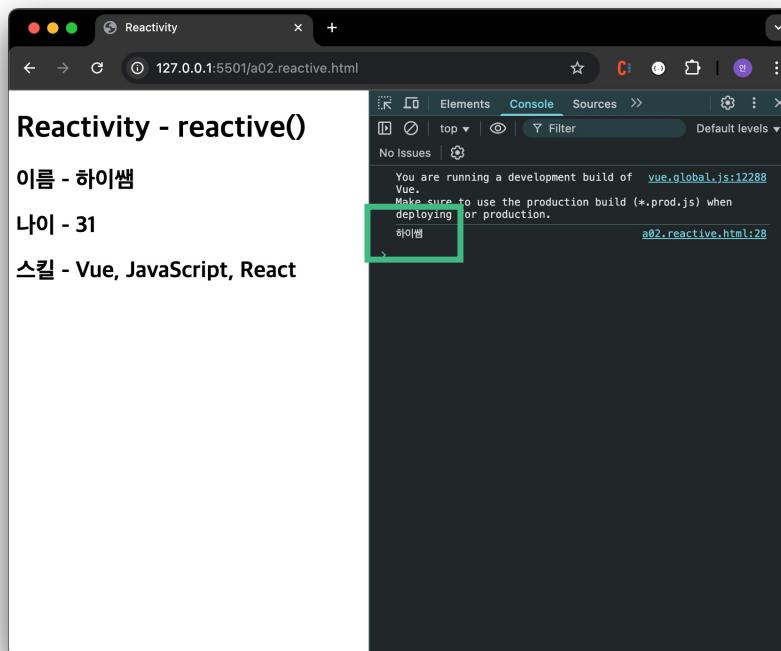
      // 객체
      const user = ref({ name: "안효인" });
      console.log(user.value.name); // '안효인'
      user.value.name = "하이쌤"; // 객체 변경
      return {
        count,
        user,
      };
    },
  }).mount("#app");
</script>

```

.value 사용

reactive()

- ▶ 객체의 반응형 상태를 생성하는 메서드
- ▶ 사용법은 ref() 함수와 같지만 인자의 type으로 기본타입(String, number, boolean)이 올 수 없음
- ▶ reactive의 인자 타입 : object, array, collection (Map, Set)
- ▶ 객체와 배열에 최적화
- ▶ 직접 프로퍼티 접근 가능 (.value 사용 없이 접근 가능)



a02.reactive.html

```

<div id="app">
  <h1>Reactivity - reactive()</h1>
  <h2>이름 - {{ state.name }}</h2>
  <h2>나이 - {{ state.age }}</h2>
  <h2>스킬 - {{ state.skills.join(', ') }}</h2>
</div>/#app

<script>
  const { createApp, reactive } = Vue;

  const app = createApp({
    setup() {
      const state = reactive({
        name: "하이쌤",
        age: 30,
        skills: ["Vue", "JavaScript"],
      });

      console.log(state.name); // '하이쌤'
      state.age++; // 직접 변경 가능
      state.skills.push("React"); // 배열 조작 가능
      return {
        state,
      };
    },
  }).mount("#app");
</script>

```

.value 사용 X

ref() VS reactive()

구분	ref()	reactive()
적합한 타입	원시 타입, 단일 객체	복잡한 객체, 배열
값 접근	.value 필요	직접 접근
템플릿 사용	자동 언래핑	그대로 사용
재할당	가능	제한적

▶ 선택 기준

- 간단한 값: ref()
- 복잡한 객체: reactive()

Thank you !!!

Programming Language