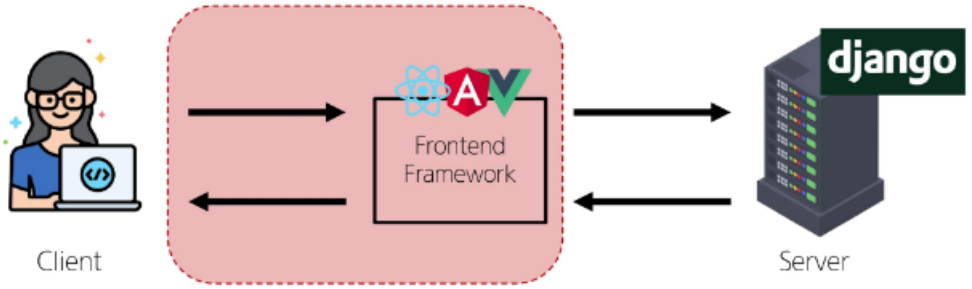
Vue

Introduction of Vue

Frontend 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 frameworks

클라이언트 측에서 UI와 상호작용을 개발하기 위해 사용되는 JavaScript 기반 프레임워크

Client-side frameworks가 필요한 이유

"웹에서 하는 일이 많아졌다." => "다루는 데이터가 많아졌다."

- 단순히 무언가를 읽는 곳에서 무언가를 하는 곳으로
- 사용자는 이제 웹에서 문서만을 읽는 것이 아닌 음악을 스트리밍하고, 영화를 보고, 지구 반대편 사람들과 텍스트 및 영상 채팅을 통해 즉시 통신하고 있음
- 이처럼 현대적이고 복잡한 대화형 웹 사이트를 "웹 애플리케이션(Web applications)"이라 부름
- JavaScript 기반의 Client-side frameworks가 등장하면서 매우 동적인 대화형 애플리케이션을 훨씬 더 쉽게 구축할 수 있게 됨
- 애플리케이션의 기본 데이터를 안정적으로 추적하고 업데이트(렌더링, 추가, 삭제 등)하는 도구가 필요
  ➤ 애플리케이션의 상태를 변경할 때마다 일치하도록 UI를 업데이트해야 함

필요성

1. 동적이고 반응적인 웹 애플리케이션 개발
   - 실시간 데이터 업데이트
2. 코드 재사용성 증가
   - 컴포넌트 기반 아키텍처
   - 모듈화된 코드 구조
3. 개발 생산성 향상
   - 강력한 개발 도구 지원

SPA

Single Page Application (SPA)

단일 페이지에서 동작하는 웹 애플리케이션

SPA 작동 원리

- 최초 로드 시 필요한 모든 리소스 다운로드
- 이후 페이지 갱신에 대해 필요한 데이터만을 비동기적으로 전달받아 회면의 필요한 부분만 동적으로 갱신
  - AJAX와 같은 기술을 사용하여 필요한 데이터만 비동기적으로 로드
  - 페이지 전체를 다시 로드할 필요 없이 필요한 데이터만 서버로부터 가져와서 화면에 표시
- JavaScript를 사용하여 클라이언트 측에서 동적으로 콘텐츠를 생성하고 업데이트

  ➤ CSR 방식

CSR

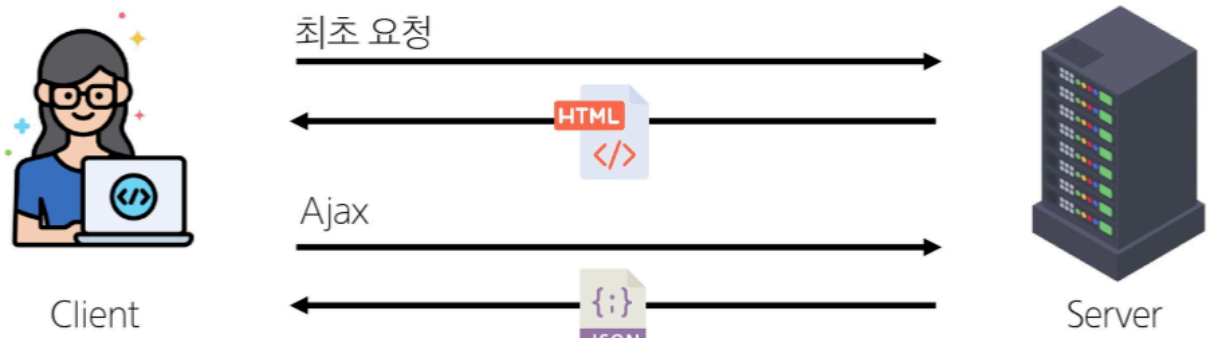
Client-side Rendering (CSR)

클라이언트에서 콘텐츠를 렌더링하는 방식

CSR 작동 원리

1. 사용자가 웹사이트에 요청 보냄
2. 서버는 최소한의 HTML과 JavaScript 파일을 클라이언트로 전송
3. 클라이언트는 HTML과 JavaScript를 다운로드 받음
4. 브라우저가 JavaScript를 실행하여 동적으로 페이지 콘텐츠를 생성
5. 필요한 데이터는 API를 통해 서버로부터 비동기적으로 가져옴

CSR 작동 예시



1. 클라이언트는 서버로부터 최소한의 HTML 페이지와 해당 페이지에 필요한 JavaScript 응답 받음
2. 그런 다음 클라이언트 측에서 JavaScript를 사용하여 DOM을 업데이트하고 페이지를 렌더링
3. 이후 서버는 더 이상 HTML을 제공하지 않고 요청에 필요한 데이터만 응답

➤ Google Maps, Facebook, Instagram 등의 서비스에서 페이지 갱신 시 새로고침이 없는 이유

SPA와 CSR의 장점

1. 빠른 페이지 전환
   - 페이지가 처음 로드된 후에는 필요한 데이터만 가져오면 되고 JavaScript는 전체 페이지를 새로 고칠 필요 없이 페이지의 일부를 다시 렌더링할 수 있기 때문
   - 서버로 전송되는 데이터의 양을 최소화 (서버 부하 방지)
2. 사용자 경험
   - 새로고침이 발생하지 않아 네이티브 앱과 유사한 사용자 경험을 제공
3. Frontend와 Backend의 명확한 분리
   - Frontend는 UI 렌더링 및 사용자 상호 작용 처리를 담당 & Backend는 데이터 및 API 제공을 담당
   - 대규모 애플리케이션을 더 쉽게 개발하고 유지 관리 가능

SPA와 CSR의 장점

1. 느린 초기 로드 속도
   - 전체 페이지를 보기 전에 약간의 지연을 느낄 수 있음
   - JavaScript가 다운로드, 구문 분석 및 실행될 때까지 페이지가 완전히 렌더링 되지 않기 때문
2. SEO(검색 엔진 최적화 문제)

- 페이지를 나중에 그려 나가는 것이기 때문에 검색에 잘 노출되지 않을 수 있음
- 검색엔진 입장에서 HTML을 읽어서 분석해야 하는데 아직 콘텐츠가 모두 존재하지 않기 때문

SPA vs. MPA / CSR vs. SSR

- Multi Page Application (MPA)
  - 여러 개의 HTML 파일이 서버로부터 각각 로드
  - 사용자가 다른 페이지로 이동할 때마다 새로운 HTML 파일이 로드됨
- Server-side Rendering (SSR
  - 서버에서 화면을 렌더링 하는 방식
  - 모든 데이터가 담긴 HTML을 서버에서 완성 후 클라이언트에게 전달

Vue

What is Vue

사용자 인터페이스를 구축하기 위한 JavaScript 프레임워크

- Evan You에 의해 발표 (2014)
  - 학사 - 미술, 미술사 / 석사 - 디자인 & 테크놀로지 전공
  - Angular 개발팀 출신
- 최신 버전은 "Vue 3" (2024)

※ Vue 2 문서에 접속하지 않도록 주의

Vue를 학습하는 이유

1. 낮은 학습 곡선
   - 간결하고 직관적인 문법을 가지고 있어 빠르게 익힐 수 있음
   - 잘 정리된 문서를 기반으로 어렵지 않게 학습할 수 있음
2. 확장성과 생태계
   - 다양한 플러그인과 라이브러리를 제공하는 높은 확장성
   - 전세계적으로 활성화된 커뮤니티를 기반으로 많은 개발자들이 새로운 기능을 개발하고 공유하고 있음
3. 유연성 및 성능
   - 작은 규모의 프로젝트부터 대규모의 애플리케이션까지 다양한 프로젝트에 적합
4. 가장 주목받는 Client-side Framework

SSAFY에서의 Vue

- Vue는 React나 Angular 대비 간결하고 직관적인 문법을 가지고 있어 학습이 상대적으로 원할
  - 짧은 시간내에 효율적으로 결과물을 만들어 낼 수 있음
- 거대하고 활발한 커뮤니티를 가지고 있어 풍부한 문서, 튜토리얼, 예제 및 다양한 리소스를 공유 받을 수 있음
  - 최신 업데이트 및 트렌드를 공유함으로써 지속적인 학습을 촉진

Vue 체험하기

- Vue CDN 사용

```
<!-- first-vue.html -->

<div id="app">
    <h1>{{ message }}</h1>
    <button v-on:click="count++">
        Count is: {{ count }}
    </button>
</div>

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
    const { createApp, ref } = Vue
    const app = createApp({
        setup() {
            const message = ref('Hello vue!')
            const count = ref(0)

            return {
                message,
                count
            }
        }
    })
    app.mount('#app')
</script>
```

Vue의 2가지 핵심 기능

1. 선언적 렌더링 (Declarative Rendering)
   - 표준 HTML을 확장하는 Vue "템플릿 구문"을 사용하여 JavaScript 상태(데이터)를 기반으로 화면에 출력될 HTML을 선언적으로 작성
2. 반응성 (Reactivity)
   - JavaScript 상태 변경을 추적하고, 변경사항이 발생하면 자동으로 DOM을 업데이트

Vue의 주요 특징 정리

1. 반응형 데이터 바인딩
   - 데이터 변경 시 자동 UI 업데이트
2. 컴포넌트 기반 아키텍처
   - 재사용 가능한 UI 조각
3. 간결한 문법과 직관적인 API
   - 낮은 학습 곡선
   - 높은 가독성
4. 유연한 스케일링
   - 작은 프로젝트부터 대규모 애플리케이션까지 적합

Component

재사용 가능한 코드 블록

특징

- UI를 독립적이고 재사용 가능한 일부분으로 분할하고 각 부분을 개별적으로 다룰 수 있음

➤ 자연스럽게 애플리케이션은 중첩된 Component의 트리 형태로 구성됨

Vue Application

1. CDN 방식
2. npm 방식

Vue Application 생성

1. CDN 작성

```
<!-- vue-instance.html -->

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

2. Application instance
   - CDN에서 Vue를 사용하는 경우 전역 Vue 객체를 불러오게 됨
   - 구조분해할당 문법으로 Vue 객체의 createApp 함수를 할당
   - 모든 Vue 애플리케이션은 createApp 함수로 새 Application instance를 생성하는 것으로 시작함

```
<!-- vue-instance.html -->
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
    const { createApp } = Vue
    const app = createApp({})
</script>
```

   - Root Component
     - createApp 함수에는 객체(컴포넌트)가 전달됨
     - 모든 App에는 다른 컴포넌트들을 하위 컴포넌트로 포함할 수 있는 Root(최상위) 컴포넌트가 필요(현재는 단일 컴포넌트)

3. Mounting the App (앱 연결)
   - HTML 요소에 Vue Application instance를 탑재(연결)
   - 각 앱 인스턴스에 대해 mount()는 한 번만 호출할 수 있음

```
<!-- vue-instance.html -->

<div id="app"></div>

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
    const { createApp } = Vue
    const app = createApp({})
    app.mount('#app')
</script>
```

반응형 상태

`ref()`

반응형 상태(데이터)를 선언하는 함수 (Declaring Reactive State)

➤ 반응형을 가지는 참조 변수를 만드는 것 (ref === reactive reference)

- value 속성이 있는 ref 객체로 래핑(wrapping)하여 반환하는 함수
- ref로 선언된 변수의 값이 변경되면, 해당 값을 사용하는 템플릿에서 자동으로 업데이트
- 인자는 어떠한 타입도 가능

```
const { createApp, ref } = Vue
const app = createApp({
    setup() {
        const message = ref('Hello vue!')
        console.log(message) // ref 객체
        console.log(message.value) // Hello vue!
    }
})
```

- 템플릿의 참조에 접근하려면 setup 함수에서 선언 및 반환 필요
- 편의상 템플릿에서 ref를 사용할 때는 .value를 작성할 필요 없음(automatically unwrapped)

```
const app = createApp({
    setup() {
        const message = ref('Hello vue!')
        return {
            message
        }
    }
})
```

```
<!-- vue-instance.html -->

<div id="app">
    <h1>{{ message }}</h1>
</div>
```

Vue 기본 구조

- createApp()에 전달되는 객체는 Vue 컴포넌트
- 컴포넌트의 상태는 setup() 함수 내에서 선언되어야 하며 객체를 반환해야 함

```
const app = createApp({
    setup() {
        const message = ref('Hello vue!')
        return {
            message
        }
    }
})
```

템플릿 렌더링

- 반환된 객체의 속성은 템플릿에서 사용할 수 있음
- Mustache syntax(콧수염 구문)를 사용하여 메세지 값을 기반으로 동적 텍스트를 렌더링

```html
<div id="app">
    <h1>{{ message }}</h1>
</div>
```

```
const app = createApp({
    setup() {
        const message = ref('Hello vue!')
        return {
            message
        }
    }
})
```

- 콘텐츠는 식별자나 경로에만 국한되지 않으며 유효한 JavaScript 표현식을 사용할 수 있음

```
<h1>{{ message.split('').reverse().join('') }}</h1>
```

Event Listeners in Vue

- `'v-on'` directive를 사용하여 DOM 이벤트를 수신할 수 있음
- 함수 내에서 반응형 변수를 변경하여 구성 요소 상태를 업데이트

```
<div id="app">
    <button v-on:click="increment">{{ count }}</button>
</div>
```

```
const { createApp, ref } = Vue

const app = createApp({
    setup() {
        const number = ref(0)
        const increment = function () {
            count.value++
        }
        return {
            count,
            increment
        }
    }
})
```

Template Syntax

DOM을 기본 구성 요소 인스턴스의 데이터에 *선언적으로 바인딩*할 수 있는 HTML 기반 *템플릿 구문*을 사용
(Vue Instance와 DOM을 연결) (확장된 문법 제공)

Template Syntax 종류

1. Tex Interpolation
2. Raw HTML
3. Attribute Bindings
4. JavaScript Expressions

1. Text Interpolation

```
<p>Message: {{ msg }}</p>
```

- 데이터 바인딩의 가장 기본적인 형태
- 이중 중괄호 구문 (콧수염 구문)을 사용
- 콧수염 구문은 해당 구성 요소 인스턴스의 msg 속성 값으로 대체
- msg 속성이 변경될 때마다 업데이트 됨

2. Raw HTML

```
<div v-html="rawHtml"></div>
```

```
const rawHtml = ref('<span style="color:red">This should be red.</span>')
```

- 콧수염 구문은 데이터를 일반 텍스트로 해결하기 때문에 실제 HTML을 출력하려면 **`v-html`**을 사용해야 함

3. Attribute Bindings

```
<div v-bind:id="dynamicId"></div>
```

```
const dynamicId = ref('my-id')
```

- 콧수염 구문은 HTML 속성 내에서 사용할 수 없기 때문에 v-bind를 사용
- HTML의 id 속성 값을 vue의 dynamicId 속성과 동기화되도록 함
- 바인딩 값이 null이나 undefined인 경우 렌더링 요소에서 제거됨

4. JavaScript Expressions

```
{{ number + 1 }}
{{ ok ? 'YES' : 'NO' }}
{{ message.split('').reverse().join('') }}
<div v-bind:id="`list-${id}`"></div>
```

- Vue는 모든 데이터 바인딩 내에서 JavaScript 표현식의 모든 기능을 지원
- Vue 템플릿에서 JavaScript 표현식을 사용할 수 있는 위치
    1. 콧수염 구문 내부
    2. 모든 directive의 속성 값 (**"v-"**로 시작하는 특수 속성)

Expressions 주의사항

- 각 바인딩에는 하나의 단일 표현식만 포함될 수 있음
    - 표현식은 값으로 평가할 수 있는 코드 조각 (return 뒤에 사용할 수 있는 코드여야 함)
- 작동하지 않는 경우

```
<!-- 표현식이 아닌 선언식 -->
{{ const number = 1 }}

<!-- 제어문은 삼항 표현식을 사용해야 함 -->
{{ if (ok) { return message } }}
```

참고

ref 객체

ref 객체가 필요한 이유

- 일반적인 변수가 아닌 객체 데이터 타입으로 사용하는 이유는?
- Vue는 템플릿에서 ref를 사용하고 나중에 ref의 값을 변경하면 자동으로 변경 사항을 감지하고 그에 따라 DOM을 업데이트 함 (“의존성 추적 기반의 반응형 시스템")
- Vue는 렌더링 중에 사용된 모든 ref를 추적하며, 나중에 ref가 변경되면 이를 추적하는 구성 요소에 대해 다시 렌더링
- 이를 위해서 참조 자료형의 객체 타입으로 구현한 것

    ➤ JavaScript에서는 일반 변수의 접근 또는 변형을 감지할 방법이 없기 때문

    ➤ https://vuejs.org/guide/essentials/reactivity-fundamentals.html#why-refs

반응형 변수 vs. 일반 변수

```
<div id="app">
    <p>반응형 변수: {{ refValue }}</p>
    <p>일반 변수: {{ normalValue }}</p>
    <button v-on:click="updateValue">값 업데이트</button>
</div>
```

```
const app = createApp({
    setup() {
        const refValue = ref(0)
        let normalValue = 0
        const updateValue = function () {
            refValue.value++
            normalValue++
            console.log(refValue)
            console.log(normalValue)
        }
        return {
            refValue,
            normalValue,
            updateValue
        }
    }
})
```

Ref Unwrap 주의사항

- 템플릿에서의 unwrap은 ref가 최상위 속성인 경우에만 적용가능
- 다음 표현식은 어떻게 출력될까?

```
const object = {id: ref(0)}
```

```
{{ object.id + 1 }}
```

[object Object]1

➤ object는 최상위 속성이지만 object.id는 그렇지 않음

➤ 표현식을 평가할 때 object.id가 unwrap 되지 않고 ref 객체로 남아 있기 때문

- 이 문제를 해결하기 위해서는 "id를 최상위 속성으로 분해"해야 함

```
const object = { id: ref(0) }
const { id } = object
```

```
{{ id + 1 }}
```

- 단, ref가 "{{ }}"의 최종 평가 값인 경우는 unwrap 가능

```
{{ object.id }}
```

➤ `{{ object.id.value }}`와 동일

SEO

Search Engine Optimization (SEO)

- google, bing과 같은 검색 엔진 등에 내 서비스나 제품 등이 효율적으로 검색 엔진에 노출되도록 개선하는 과정을 일컫는 작업
- 정보의 대상은 주로 HTML에 작성된 내용
- 검색
  - 각 사이트가 운용하는 검색 엔진에 의해 이루어지는 작업
- 검색 엔진
  - 웹 상에 존재하는 가능한 모든 정보들을 긁어 모으는 방식으로 동작
- 최근에는 SPA, 즉 CSR로 구성된 서비스의 비중이 증가
- SPA 서비스도 검색 대상으로 넓히기 위해 JS를 지원하는 방식으로 발전하는 중

CSR과 SSR

- CSR과 SSR은 흑과 백이 아님
- 애플리케이션의 목적, 규모, 성능 및 SEO 요구 사항에 따라 달라질 수 있음

  ➤ 내 서비스에 적합한 렌더링 방식을 적절하게 활용할 수 있어야 함
- SPA 서비스에서도 SSR을 지원하는 Framework가 발전하고 있음
  - Vue의 Nuxt.js
  - React의 Next.js