
VueJS

1. 웹 프론트엔드 개념 웹 애플리케이션 및 웹사이트의 사용자 인터페이스를 개발하는 분야

역할과 책임

- **웹 디자인**
  - 사용자 경험(UX) 및 사용자 인터페이스(UI) 디자인 원칙을 적용하여 웹 페이지를 시각적으로 표현
- **웹페이지 개발**
  - HTML, CSS, JavaScript를 사용하여 디자인과 기능을 웹 페이지로 구현
- **브라우저 호환성**
  - 다양한 브라우저 및 기기에서 웹 페이지가 제대로 작동하도록 보장

웹 프론트엔드 개발자

사용자와 웹 페이지가 상호작용할 수 있도록
해당 기술과 개념을 활용하여 웹 페이지 개발

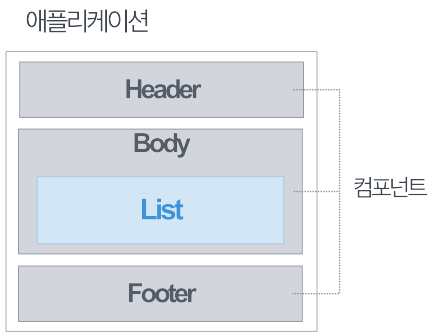
기술과 언어

- **HTML**
  - 웹 페이지의 구조와 내용(텍스트, 이미지, 링크 등)을 정의하는 마크업 언어
- **CSS**
  - 웹 페이지의 레이아웃과 디자인(색상, 폰트, 크기 등)을 정의하는 스타일시트 언어
- **JavaScript**
  - 웹 페이지의 이벤트 및 데이터 처리 등 웹 애플리케이션 개발에 사용되는 스트립트 언어
- **프레임워크 및 라이브러리**
  - 개발 생산성을 높이기 위해 React, Angular, Vue.js 등 프론트엔드 프레임워크 및 라이브러리 사용

2. Vue.js 특징 프론트엔드 웹 개발을 위한 오픈 소스 자바스크립트 프레임워크

01 컴포넌트 기반 아키텍처

- 애플리케이션을 작은 조각으로 분할
- 컴포넌트는 독립된 모듈로, 재사용이 가능



02 Vue CLI

- Vue.js 프로젝트를 생성, 빌드 등 간편하게 관리할 수 있게 해주는 도구

Command Prompt

```
# 프로젝트 생성
> vue create <project-name>

> cd <project-path>
# 프로젝트 실행
> npm run serve
# 프로젝트 빌드
> npm run build
```

03 싱글 파일 컴포넌트(SFC)

- HTML, CSS, JavaScript 코드를 한 파일에서 관리

App.vue File

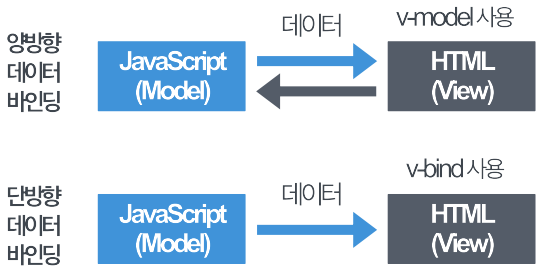
```
<script setup>
 // Write JavaScript Code
</script>

<template>
 // Write HTML Code
</template>

<style scoped>
 // Write CSS Code
</style>
```

04 반응성 데이터 바인딩

- 양방향 데이터 바인딩 제공
- 데이터 변경시 자동으로 UI를 업데이트하고 반대로 UI 변경시 데이터에 반영이 됨



05 Directives(지시자)

- HTML 요소에 추가 기능 및 동작 부여

지시자 종류
• **v-html :** 태그를 포함해서 렌더링 가능
• **v-once :** 컴포넌트를 초기에 한번만 렌더링(static)
• **v-text :** {{}} 대신 사용
• **v-bind :** 데이터를 바인딩할 때 사용
• **v-if :** 조건문 사용시
• **v-show :** 태그의 보여지는 여부를 설정
• **v-for :** 반복문 사용시

06 라우팅 및 상태 관리

- Vue Router와 Vuex 같은 공식 확장 라이브러리 제공
  : 애플리케이션의 내비게이션 및 데이터 상태 관리
- **Vue Router**(라우팅 라이브러리)
  : 다양한 기능으로 페이지를 새로고침 없이 전환
- **Vuex**(글로벌 상태관리 라이브러리)
  : 모든 컴포넌트에 대한 중앙 집중식 저장소 역할
  : 예측 가능한 방식으로 상태를 변경

3. Vue.js vs jQuery

Vue.js

소개
- 유연한 JavaScript 프레임워크
- 사용자 인터페이스 및 웹 애플리케이션을 개발하는 데 사용

특징
- 컴포넌트 기반 아키텍처로 재사용 가능한 UI 요소를 구축
- 반응성 데이터 바인딩으로 데이터와 UI 사이 동기화의 단순화
- Vue CLI를 사용하여 프로젝트 설정
- 라우팅 및 상태 관리를 위한 공식 라이브러리 지원

사용 사례
- 중간 규모 또는 단일 페이지 웹 애플리케이션을 개발하려는 경우
- 프로젝트를 컴포넌트로 나누어 개발 및 관리하려는 경우

jQuery

소개
- JavaScript 라이브러리
- 웹 개발을 단순화하고 브라우저 간 호환성을 관리하는데 중점을 둠

특징
- 간단하고 직관적인 문법 사용
- 브라우저 간 호환성 문제를 해결
- DOM 조작, 이벤트 처리 및 애니메이션을 쉽게 다루기 가능
- jQuery 플러그인을 통해 다양한 확장 기능을 추가

사용 사례
- 작은 웹 페이지 또는 프로젝트에서 간단한 DOM 조작이 필요한 경우
- 빠르게 웹 페이지에 애니메이션 또는 동적 효과를 추가하려는 경우

비교	Vue.js	jQuery
DOM 성능	가상 DOM 사용 : 실제 DOM 조작을 최소화하고 성능을 향상시킴	직접 DOM 조작 : 복잡한 변경사항 발생시 성능 문제가 발생할 수 있음
유지 관리	독립적인 컴포넌트를 구성 : 개발과 유지 관리를 단순화하고 재사용성을 높임	컴포넌트 구조를 기본적으로 제공하지 않음 : 복잡한 프로젝트에서는 코드의 가독성과 유지 관리가 어려움
데이터 바인딩	양방향 데이터 바인딩 지원 및 자동 동기화 처리 : UI 요소를 쉽게 업데이트하고 데이터 상태를 관리함	직접 기능 구현

Vue.js는 현대적인 웹 개발의 요구 사항을 충족시키고 개발자에게 높은 생산성과 코드의 가독성을 제공하는 데에 많은 이점을 가짐.
jQuery는 여전히 특정 상황에서 유용할 수 있지만, 더 복잡하고 대규모인 웹 애플리케이션을 개발하는 데는 Vue.js와 같은 프레임워크가 더 적합할 수 있음.

3. Vue.js vs jQuery

Vue.js 버튼 클릭 이벤트 처리

```
<script setup>
  import { ref } from ‘vue’;

  const message = ref('Hello, Vue.js!‘);

  function updateMessage() {
    message.value = 'Button clicked with Vue.js!';
  }
</script>

<template>
  <button @click="updateMessage">Click me</button>
  <p>{{ message }}</p>
</template>
```

- 데이터 바인딩을 통해 메시지가 동적으로 업데이트 됨
- 코드가 간결하고 구조화되어 있어, 상태 관리 및 이벤트 처리가 용이함

jQuery 버튼 클릭 이벤트 처리

```
<body>
  <div>
    <button id="myButton">Click me</button>
    <p id="message">Hello, jQuery!</p>
  </div>

  <script>
    $(document).ready(function() {
      $(‘#myButton’).click(function() {
        $(‘#message’).text(‘Button clicked with jQuery!’);
      });
    });
  </script>
</body>
```

- DOM 요소에 대한 직접적인 선택 및 조작을 수행해야 함

Vue.js가 코드를 더 간결하게 작성하고 유지 관리하기 쉽게 만들며
데이터와 UI 간의 바인딩을 쉽게 처리하는 데 도움이 됨