SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

Vue.js

<알림>

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로! Enabling People

1장. Vue.js 개요

챕터의 포인트

- 서버와 클라이언트
- Vue.js 특징
- MVVM 패턴
- Hello Vue.js

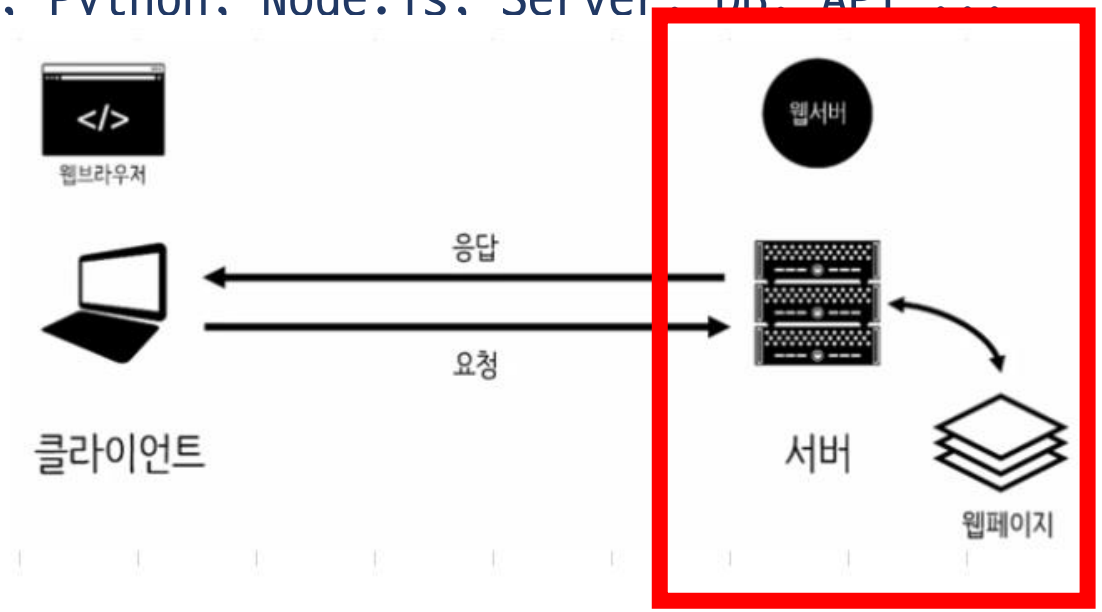
SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

서버와 클라이언트

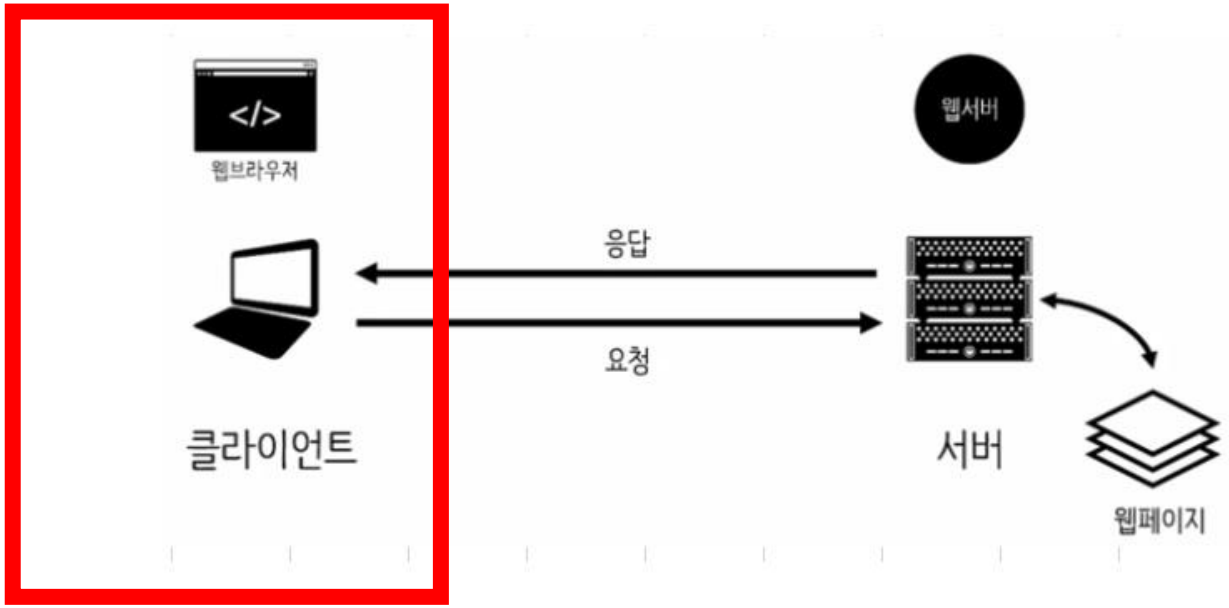
다른 말로 백엔드

- 사용자 눈에 보이지 않는 곳에서 작동하는 부분
- 클라이언트에게 정보나 서비스를 제공하는 시스템
  - ex) 웹서버, 데이터베이스 서버
- 클라이언트의 요청 (request) 을 받으면, 적절한 처리 후 클라이언트에게 응답 (response)
- 언어 및 기술: Java, Python, Node.js, Server, DB, API ...



프론트엔드

- 웹에서는 클라이언트가 프론트엔드
- 사용자 눈에 보이고 상호작용하는 부분
- 서버를 통해 접속할 수 있는 애플리케이션이나 서비스
  - ex) 브라우저, 웹사이트
- 서버에게 요청 (request) 하면, 서버로부터 응답 (response) 을 받는다.
- 언어 및 기술: HTML, CSS, JavaScript, React.js, Vue.js ...



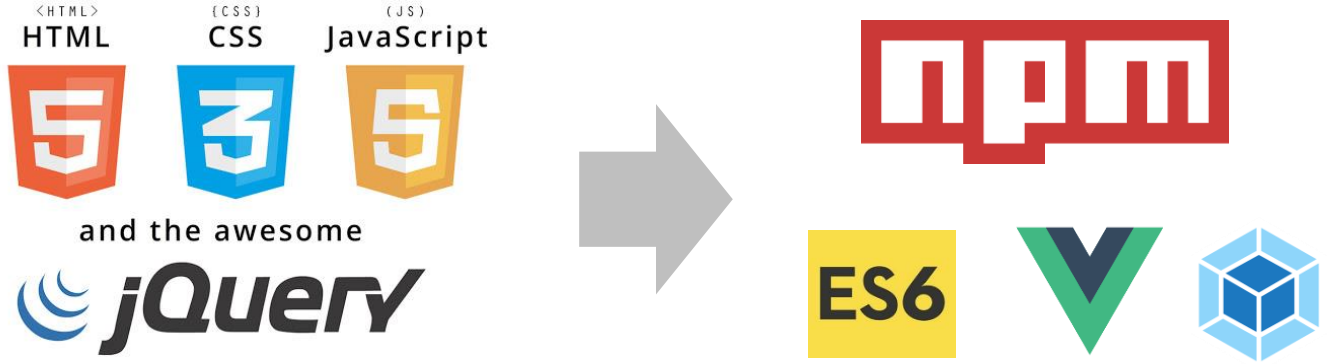
프론트엔드 개발의 과거와 현재

과거 Front-end 개발자

- 클라이언트의 브라우저에서 동작하는 코드를 구현하는 개발자
- 서버에 대한 지식 없이 개발 가능

현대 Front-end 개발자

- 웹 페이지 기능이 많아짐 (성능 중요)
- 코드량과 개발 범위 늘어남: Front-end 서버까지 다뤄야



SSR (Server Side Rendering)

- 전통적인 렌더링방식
- HTML 문서를 서버에서 생성해 클라이언트로 전송 => 첫 화면 생성 빠름
- 검색엔진 최적화 (SEO) 쉬움
- 화면 변경사항 발생 시
  - 서버로 다시 렌더링 요청 => 서버 부담 증가
  - 화면 전환 시 새로고침 발생

CSR (Client Side Rendering)

- 비교적 최근에 등장한 렌더링방식
- HTML 문서를 클라이언트에서 생성 => 첫 화면 생성 느림
- 검색엔진 최적화 (SEO) 어려움
- 화면 변경사항 발생 시
  - 서버에 요청하지 않고 변화가 존재하는 부분만 변경 => 서버 부담 감소
  - 새로고침 발생 안함: 부드러운 화면 전환

- Vue.js 에서 쓰이는 방식

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Vue.js 특징

Front-end Framework 필요성

DOM 제어의 변경과 Component 기반 웹 개발을 위함

- Component 단위로 조립식으로 웹페이지 UI 개발 (생산성 UP / 가독성 UP)
- Virtual DOM : jQuery 의 비효율적 DOM 제어 개선 (웹페이지 성능 UP)

조립식 성 만들기
= Component 기반 웹 사이트 개발

현대 Front-end 개발자의
필수 지식이 됨

Browser 가 HTML을 출력하는 원리

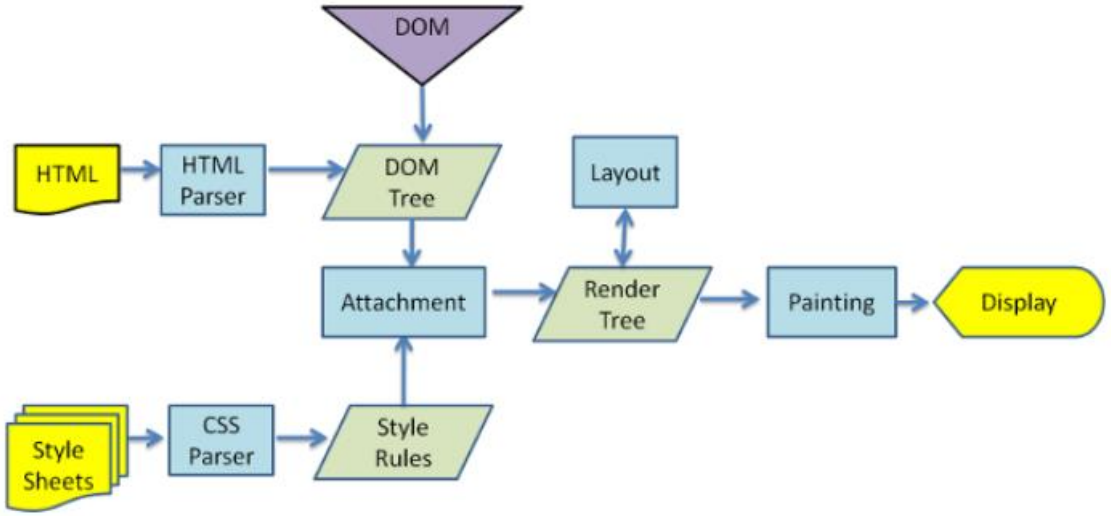
Browser가 HTML / CSS 을 전달 받은 후 Display 까지 과정

1. HTML Parsing 하여 DOM Tree 생성

2. Render Tree 생성
   - 각 DOM Tree의 Node에 Style 정보를 입힌다.

3. Layout 과정
   - 각 Render Tree Node들의
     출력 할 좌표가 계산되어 정해진다.

4. Painting
   - 이미지 / Color를 입혀 출력을 준비하는 과

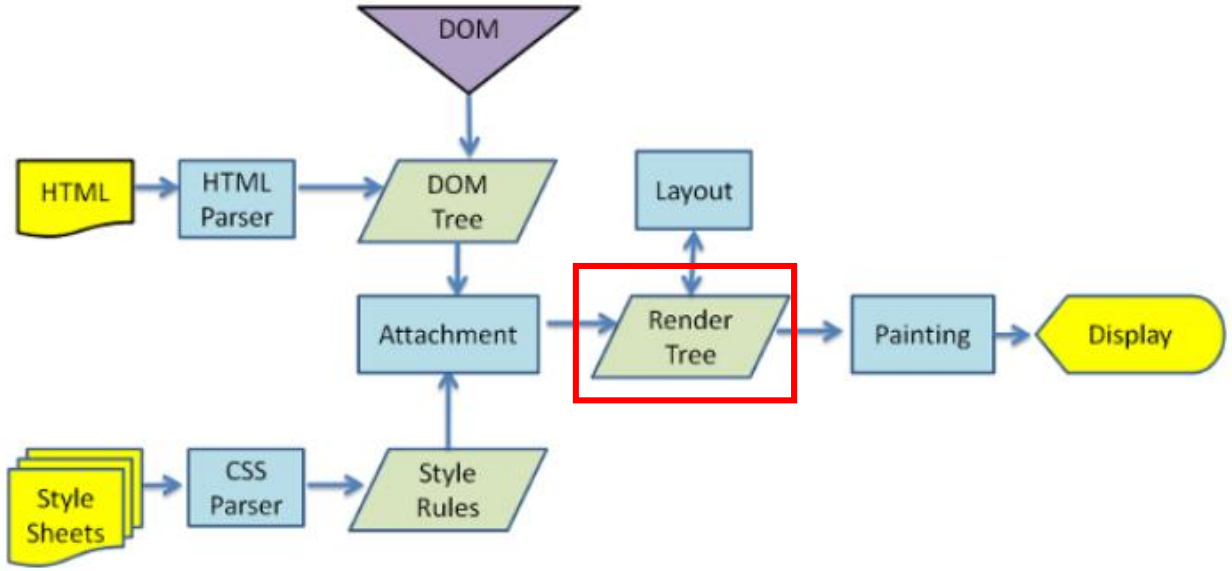


Virtual DOM의 필요성

일반적인 DOM 변경 발생시 Render Tree 부터 재 시작이 이루어진다.

- Render Tree 과정 : 모든 DOM에 Style 정보를 다시 입히는 과정
- DOM 제어를 할 때마다, Client Browser의 속도 저하가 발생한다.

- 만약, 10개의 D e를 추가 재생성 한다.



Virtual DOM 동작 방식

속도 개선을 위한 Virtual DOM Tree 동작 과정

1. 진짜 DOM Tree 기반으로, Virtual DOM Tree를 생성한다.
2. DOM 제어를 Virtual DOM Tree에 모두 적용한다.
3. 모든 DOM 제어를 끝났을 때, Virtual DOM Tree를 진짜 DOM Tree에 적용한다
4. 렌더링을 시작한다.

모든 DOM 제어 이후, 최종적으로 한 번 렌더링을 수행하는 방법

Vue / React 등을 사용하게 되면,
Virtual DOM 방식으로 DOM 제어를 하게 된다.

Virtual DOM 특징

Virtual DOM 사용 전

- DOM 내 요소들이 많아질수록 DOM 요소를 핸들링하는 게 어려워짐
- 화면 변경 요청이 있을 때마다 DOM 요소 변경 => 성능 하락

Virtual DOM 특징

- 모든 DOM 제어가 끝나면 최종적으로 단 한번만 렌더링 => 성능 향상
- Vue.js, React.js 에서 사용
- Virtual DOM 채택 시, Real DOM 에 직접 접근 금지
  - querySelector, createElement, innerHTML 등 사용 안함

Progressive JavaScript Framwork

- JavaScript 프로젝트에서, 점진적으로 수정해 나가면서 Vue.js를 적용시킬 수 있다. (Vue.js를 적용하기 위해 완전히 뒤엎는 것이 아닌, 수정하는 방식으로 적용하는 프레임워크)

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

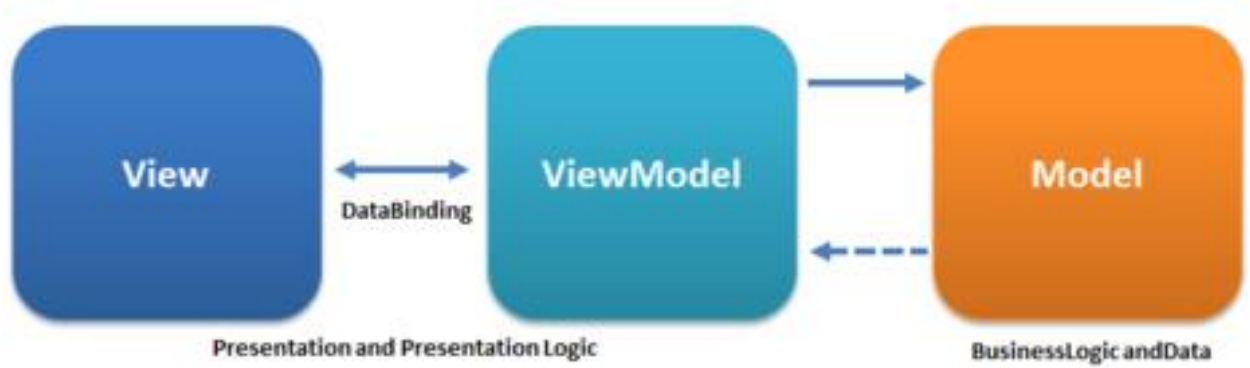
MVVM 패턴

디자인 패턴

- 프로그램 개발시 자주 나타나는 문제를 해결하기 위한 방법 / 규약을 묶어, 이름을 붙인 것
- 특정 디자인패턴의 규칙대로 웹 개발하면, 협업하기 수월하며, 유지보수가 좋다. (타인의 아주 창의적인 소스코드 방지)
- Vue.js로 웹 개발하는 방식
  - MVVM 패턴

MVVM 패턴으로 웹 개발을 한다는 의미

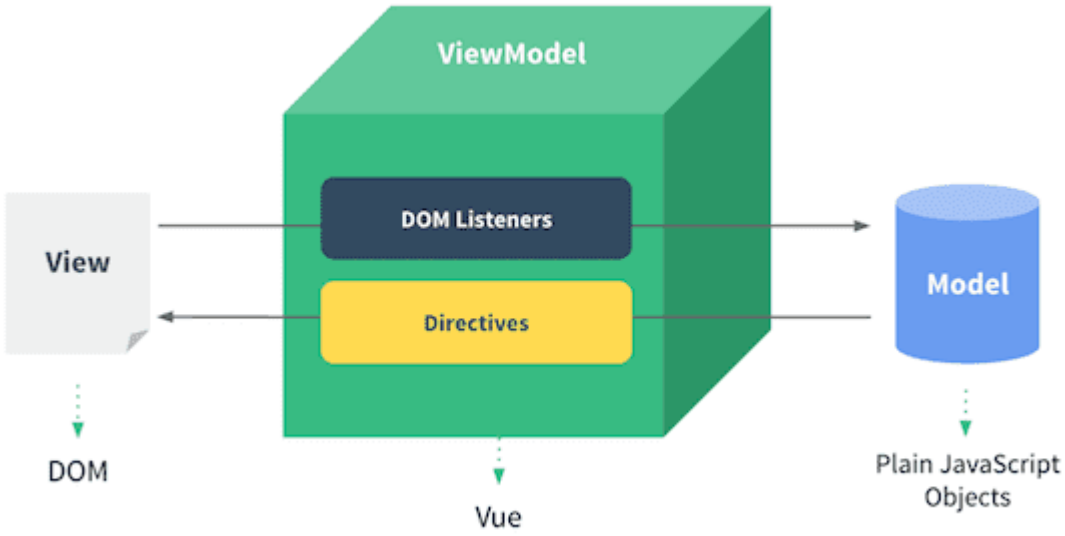
- 세 가지 역할을 분명히 구분하여, 웹 개발을 수행
  - Model : 실제 데이터를 처리하는 소스코드
  - View : UI에 해당하는 소스코드
  - ViewModel : View를 표현하기 위해 만들어진 코드



MVVM 패턴에서 Vue.js의 역할

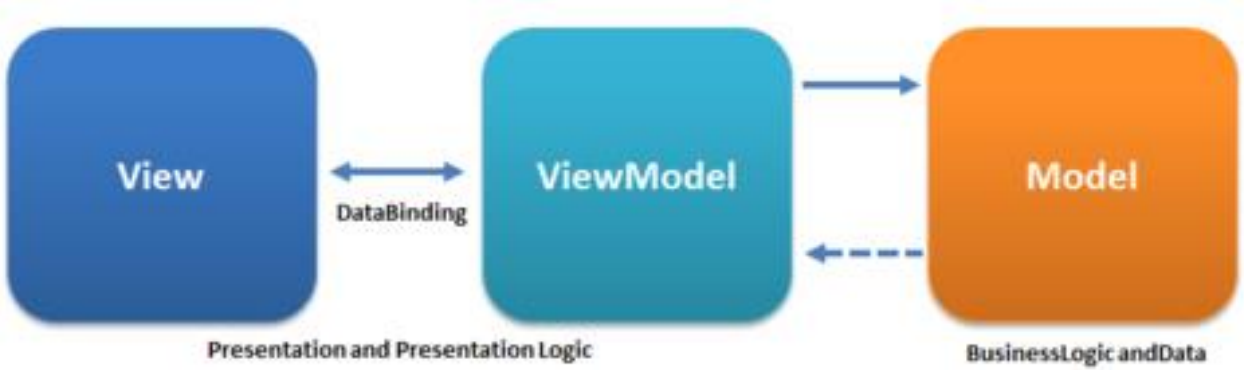
ViewModel 부분을 Vue.js가 담당한다.

- 기존 수업 : ViewModel 없이, Model (JavaScript) 로 View 를(DOM) 직접 제어
- 현재 수업 : Vue.js을 통하여, View(DOM)을 제어



Vue.js에서 MVVM패턴 동작과정

- 사용자는 View 를 통해 입력을 한다.
- View 에서 이벤트 발생시, View Model의 콜백 함수를 호출한다.
- ViewModel은 Model에게 필요한 데이터를 요청하고, 전달 받는다.
- ViewModel은 받은 데이터를 가공하여 저장한다.
- ViewModel에서 저장하면, Binding으로 인해 View의 값이 자동 갱신된다.



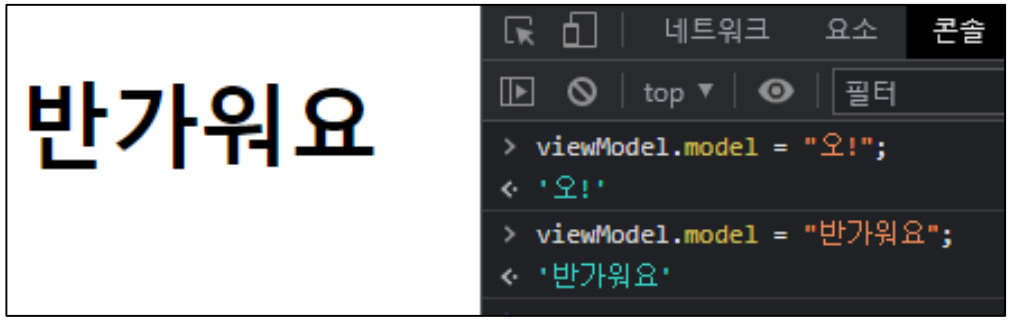
ViewModel 만들어보기

- 핵심은 data와 html의 일치화
  - 기존 html은 데이터를 받아와서 데이터가 변경이 되면 html도 같이 변경해줘야 했다.
  - 이로 인해 데이터와 html의 일치화 부분에 이슈가 자주 발생
  - 개발은 불편함 -> 개선의 형태로 발전한다.

ViewModel 만들어보기

- Object.defineProperty
  - viewModel에 model이라는 값의 속성을 정의하는 메서드
    - viewModel의 model을 바꾸면 html 또한 같이 바뀐다.

```
<script>
    const h1 = document.createElement('h1');
    document.body.append(h1);

    const viewModel = {};

    let model = "";

    // viewModel 객체에 model이라는 속성을 정의
    Object.defineProperty(viewModel, 'model', {
        get(){
            return model
        },
        set(value){
            model = value;
            h1.innerHTML = model;
        }
    });

    viewModel.model = "Vue HELLO"
</script>
```



MVVM Pattern 정리

- Vue.js 의 디자인 패턴
- Model
  - 실제 데이터 처리
- View
  - 사용자 UI
- ViewModel
  - View 와 Model 사이에 위치해 의존성 제거
  - View 에서 이벤트 발생 시, ViewModel 의 콜백함수 호출
  - Vue.js 는 ViewModel 담당

Vue.js 특징

- 쉬운 DOM 제어
  - 순수 JavaScript 에 비해 조건, 반복, 데이터 바인딩, 이벤트처리 쉬움
- 컴포넌트 기반의 조립식 웹 UI 개발
  - 코드의 가독성 및 생산성 높음
- Vue.js 에 쓰인 JavaScript 를 점진적으로 수정
  - Progressive JavaScript Framework
- CSR 방식
  - 서버 부담 덜어줌
- Virtual DOM 방식 사용해 웹페이지 성능 향상

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Hello Vue.js

[실습] Vue.js 시작하기

Vue 설치

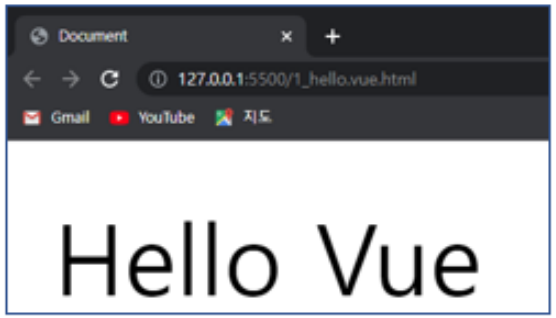
- CDN 방식으로 간단하게 Script 를 추가하자. - jsdelivre
- <script src="https://cdn.jsdelivr.net/npm/vue@2.7.8/dist/vue.min.js"></script>
- 2.7.8 버전 사용

```
<body>
  <!-- 이곳에 HTML 코드 넣을 예정 -->
  <script src="https://cdn.jsdelivr.net/npm/vue@2.7.8/dist/vue.min.js"></script>
  <script>

    // 이곳에 vue 코드 넣을 예정

  </script>
</body>
```

Hello Vue.js

- new Vue
  - el
    - #app으로 지정된 div를 Vue 형식으로 만든다.
  - data(){}
    - Vue Instance 에서 변수를 생성할 때 사용하는 객체, 함수안에 변수를 정의한다

```
<body>
    <div id="app">
        {{ message }}
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.8/dist/vue.min.js"></script>
    <script>
        const app = new Vue({
            el: "#app",
            data() {
                return {
                    message: "Hello Vue",
                }
            },
        });
    </script>
</body>
```



SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

2장. Vue.js 지시자

챕터의 포인트

- v-on
- 데이터 바인딩
- v-if
- v-for
- Vue.js 연습

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

v-on

v-on

- 이벤트핸들러 지정할 때 사용
- 클릭했을 때 bbq 메서드를 실행

```
v-on:click="bbq"
```

- 축약형 사용 가능

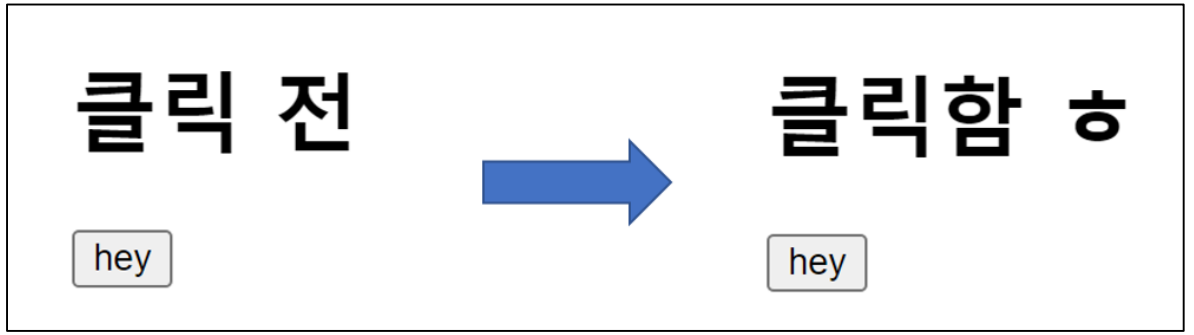
```
@click="bbq"
```

클릭 이벤트와 전역 변수

클릭 이벤트

- v-on:click 이벤트 등록
  - bbq 콜백함수 등록
- data 에 있는 변수들은 전역변수
  - this를 사용하여 접근 가능(this.message)
- methods
  - Vue Instance 에서
    메서드를 생성할 때 사용되는 객체
- message에 값을 넣으면, 즉시 반영 됨
  - viewModel

```
<body>
    <div id="app">
        <h1>{{ message }}</h1>
        <button v-on:click="bbq">hey</button>

    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.8/dist/vue.min.js"></script>
    <script>
        const app = new Vue({
            el: "#app",
            data() {
                return{
                    message: "클릭 전",
                    nextMessage: "클릭함 ㅎ",
                }
            },
            methods: {
                bbq() {
                    this.message = this.nextMessage;
                },
            },
        });
    </script>
</body>
```



순수 JavaScript 코드와 비교

순수 JavaScript 로 작성할 때와 과정을 비교해보자.

- 두 개의 DOM 요소를 querySelector() 로 각각 가져옴
- <button> 에, addEventListener() 를 사용해 이벤트 등록
- <h1> 의 textContent 에 접근해 값 변경

단점

- 이벤트와 DOM 에 대한 전문성 필요
  - 이 경우, textContent 속성을 변경해야 한다는 것을 정확히 알고 있어야 함
  - 만약 다른 속성을 변경해야 한다면 추가적인 학습 필요

- Vue.js 코드에 비해 가독성 떨어짐
  - 화면에서 변경될 데이터, 클릭 시 작동할 이벤트핸들러 한눈에 파악 어려움
  - 변수와 함수가 많아질수록 생산성 떨어짐

- 매번 DOM 제어가 이루어지기에 성능 하락
  - Vue.js 는 Virtual DOM 사용하고, 모든 DOM 제어 이후 단 한 번만 렌더링

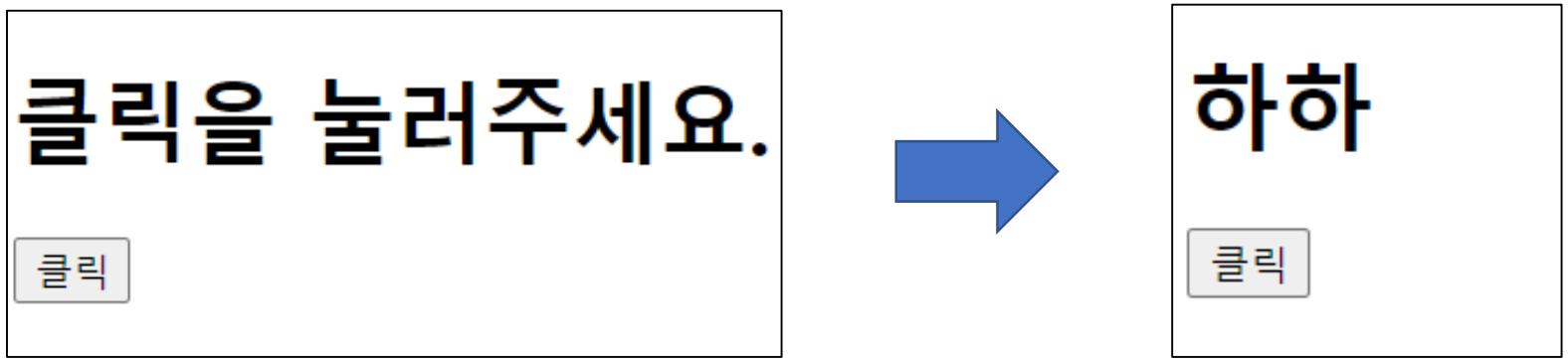
```
<h1>클릭 전</h1>
<button>hey</button>

const h1 = document.querySelector("h1");
const btn = document.querySelector("button");
btn.addEventListener("click", () => {
  h1.textContent = "클릭함 ㅎ";
});
```

[도전] 버튼 클릭시 글씨 변경

Vue.js 클릭 이벤트와 Binding 사용하기

- HTML <h1> , <button> 추가
- <h1> 초기상태 : “클릭을 눌러주세요.”

- 버튼 클릭시 <h1> : “하하 " 로 변경



SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

데이터 바인딩

데이터 바인딩

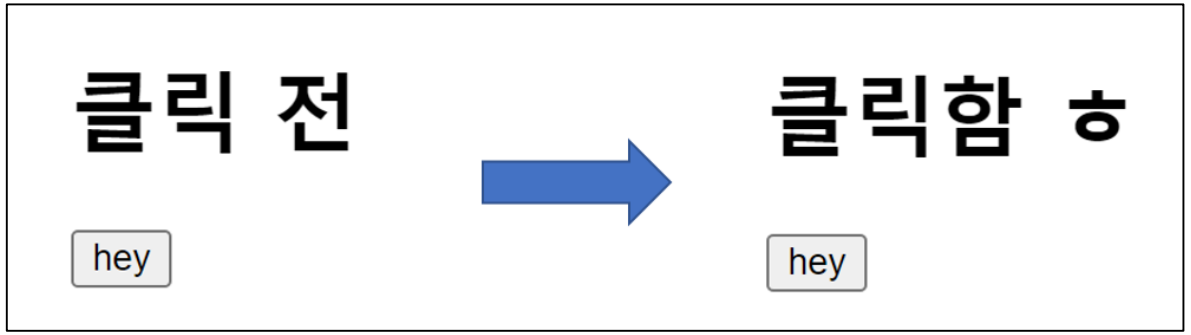
- bind: 묶다
- data 객체 내의 변수와, 화면의 데이터를 연동하는 것을 의미
- Vue Instance의 data 객체 멤버 변경 시, 화면에 바인딩된 변수가 즉시 수정되어 화면 변경
- 바인딩의 종류
  - 단방향 바인딩
    - v-bind, {{ }}
  - 양방향 바인딩
    - v-model

{{ 변수명 }}

- Vue Instance의 값을 수정하면,
  즉시 {{ message }} 에 값이 반영된다.

- 단방향 Binding : data 객체의
  해당되는 변수와 연결됨
  - 예제의 경우엔 message

- 단방향이란, 데이터가 한쪽 방향으로만 묶여있다는 뜻
  data 객체 안에 message 가 바뀔 수는 있지만,
  사용자가 화면에서 message 자체를 변경할 순 없다.



Binding

```html
<!-- HTML -->
<div id="app">
  <h1>{{ message }}</h1>
  <button v-on:click="bbq">hey</button>
</div>
```

```js
// vue instance
const app = new Vue({
  el: "#app",
  data: {
    message: "클릭 전",
    nextMessage: "클릭함 ㅎ",
  },
  methods: {
    bbq() {
      this.message = this.nextMessage;
    },
  },
});
```

v-bind

- 단방향 Binding
- 축약형
  - : 로도 활용이 가능하다.
    - ex) :href 는 v-bind:href와 같다.
- 두 가지 목적으로 사용
  - 태그의 속성을 동적으로 변경
  - 하위 컴포넌트로 props 전달
    - 추후 학습 예정

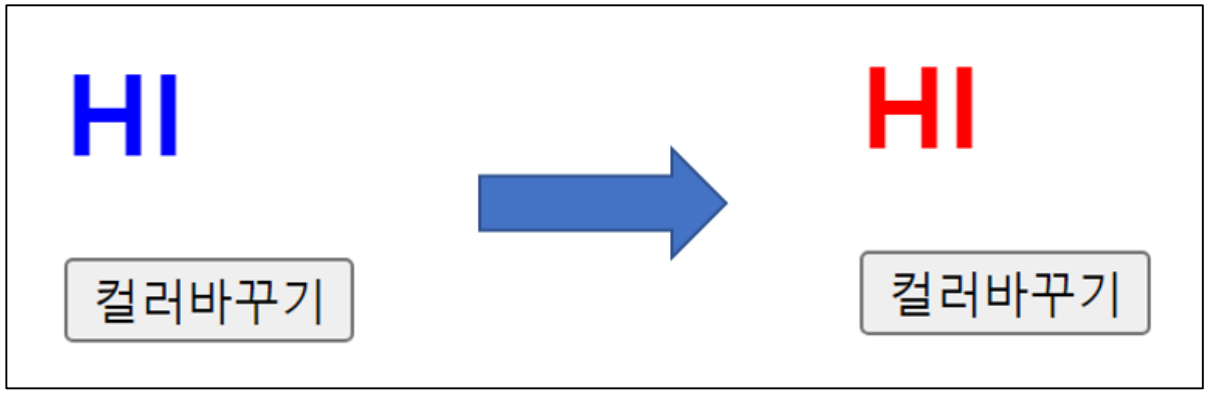
```
<body>
    <div id="app">
        <a v-bind:href="url1">구글</a>
        <a :href="url2">네이버</a>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.8/dist/vue.min.js"></script>
    <script>
        const app = new Vue({
            el: "#app",
            data() {
                return{
                    url1: "https://google.com",
                    url2: "https://naver.com",
                }
            },
        });
    </script>
</body>
```

[도전] class 속성 바꾸기

버튼을 클릭하면, “HI” 글씨의 컬러가 바뀌는 코드 작성하기

- 초기 상태 : color-blue class
  - 클릭 후 : colorRed class 로 변경
- 힌트
  - v-bind:class 활용

```
.colorBlue {
  color: blue;
}
.colorRed {
  color: red;
}
```



순수 JavaScript 코드와 비교

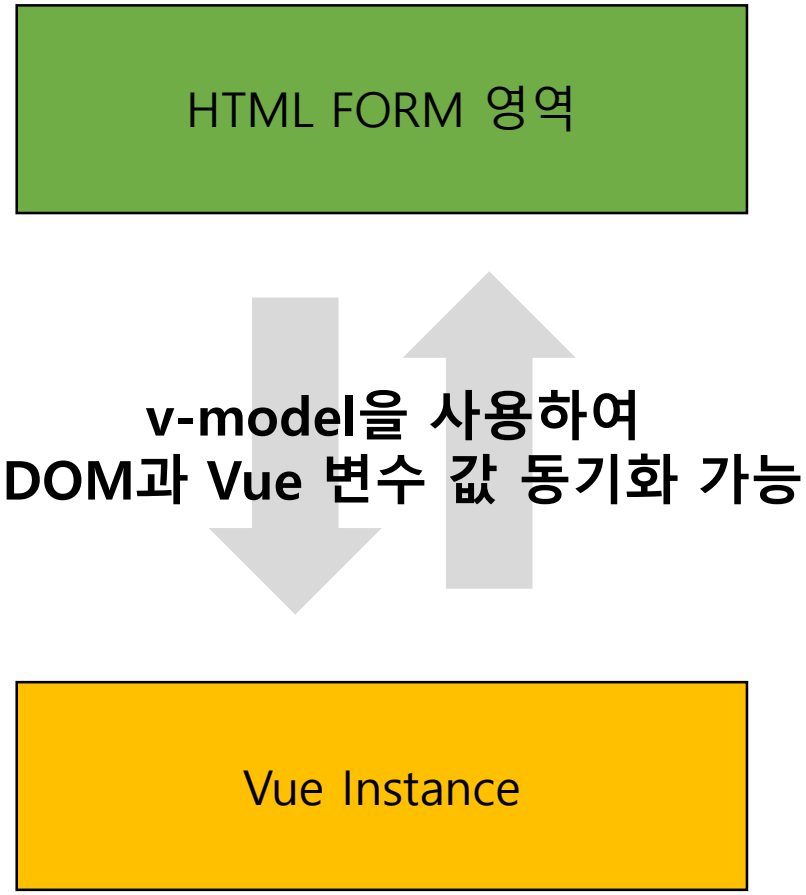
순수 Javascript로 만들어보기

- 방금 전 도전과제를 순수 JavaScript 로 구현
  - 다음 질문에 답해보자.
    - Vue.js v-bind 를 사용해 구현한 것과 비교해볼 때 어떤 단점이 확인되는가?
    - 주석 처리된 구문은 동작하지 않는다. 이유는 무엇

```
    <style>
        .colorBlue {
            color: blue;
        }
        .colorRed {
            color: red;
        }
    </style>
    <title>Document</title>
</head>
<body>
    <h1>HI</h1>
    <button>컬러바꾸기</button>
    <script>
        let colorOfHi = "colorBlue";
        const h1 = document.querySelector("h1");
        const btn = document.querySelector("button");
        h1.classList.add(colorOfHi);
        btn.addEventListener("click", () => {
    // 아래 코드는 작동하지 않는다.
    // colorOfHi = "colorRed";

    // 아래 코드는 작동한다.
            let tempColorClass = colorOfHi;
            colorOfHi = "colorRed";
            h1.classList.replace(tempColorClass, colorOfHi);
        });
    </script>
</body>
```

HTML Form 또는 Vue 영역 값 동기화

- 양방향 바인딩
  - HTML Form의 값을 바꾸면, Vue 변수에 값이 변경됨
  - Vue 변수의 값을 변경하면, HTML Form의 값이 변경됨
  - <input> 태그에 사용됨
    - 사용자 입력이 data 객체의 속성을 변하게 함

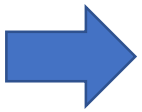


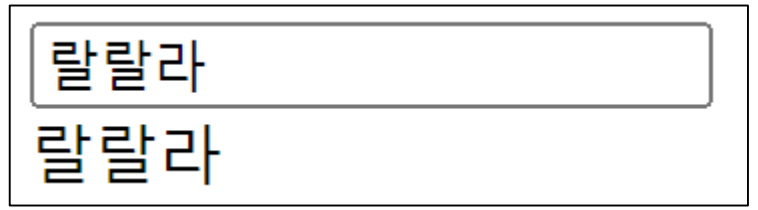
[실습] 양방향 Data Binding

v-model

- input의 값을 수정하면, HTML 출력 값도 바뀐다.

```
<body>

    <div id="app">
        <input type="text" v-model="message" />
        <div>{{ message }}</div>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.8/dist/vue.min.js"></script>
    <script>
        // vue instance
        const app = new Vue({
            el: "#app",
            data() {
                return{
                    message: "",
                }
            },
        });
    </script>
</body>
```





v-bind 와 v-model 비교하기

- 단방향 바인딩 (v-bind)
  - 데이터를 바꾸면 html도 바뀐다.
- 양방향 바인딩 (v-model)
  - 데이터를 바꾸면 html도 바뀐다.
  - html을 바꾸면 데이터도 바뀐다.

v-bind와 v-model 비교

- v-bind 와 v-model 을 <input> 에 사용했
- 직접 구현해보고 차이점을 확인해보자.

```
<body>
    <div id="app">
        <div>
            <input v-model="message" />
        </div>
        <div>
            <input v-bind:value="message" />
        </div>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.8/dist/vue.min.js"></script>
    <script>
        const app = new Vue({
            el: "#app",
            data() {
                return{
                    message: "글자를 지워보자",
                }
            },
        });
    </script>
</body>
```

글자를 지워보자

글자를 지워보자

v-if

v-if

- DOM 의 출력 여부 결정
- v-if="Boolean 값"
  - true인 경우 출력
  - false 인 경우 출력 안함
- v-else, v-else-if 와 함께 사용 가능
- v 로시작하는 경우""안에 자바스크립트가 들어간

```
<body>
    <!-- HTML -->
    <div id="app">
        <p v-if="seen">이제 나를 볼 수 있어요</p>
        <button v-on:click="openYourEyes">open your eyes</button>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.8/dist/vue.min.js"></script>
    <script>
        // vue instance
        const app = new Vue({
            el: "#app",
            data() {
                return{
                    seen: false,
                }
            },
            methods: {
                openYourEyes() {
                    this.seen = true;
                },
            },
        });
    </script>
</body>
```

v-else, v-else-if 와 함께 사용 가능

- 제시된 코드를 확인하고,
- 무엇이 출력될지 예상해보자.

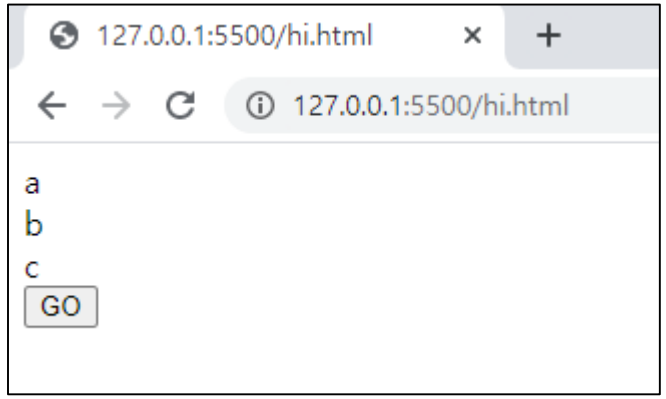
```
<body>
  <!-- HTML -->
  <div id="app">
    <p v-if="renderA">A가 보입니다.</p>
    <p v-else-if="renderB">B가 보입니다.</p>
    <p v-else>둘 다 안보입니다.</p>
  </div>
  <script src="https://cdn.jsdelivr.net/npm/vue@2.7.8/dist/vue.min.js"></script>
  <script>
    // vue instance
    const app = new Vue({
      el: "#app",
      data() {
        return{
          renderA: true,
          renderB: false,
        }
      },
    });
  </script>
</body>
```

[도전] 출력 전환

GO를 누를 때 마다, div 출력 결과 전환

- Div 7개 추가 (a ~ g)
- 버튼을 누를 때 마다 div 출력이 달라진다.
  - 처음 a, b, c 가 출력되어 있다.
  - 버튼 클릭시 d, e 가 출력된다.
  - 한번 더 클릭시 f, g 가 출력된다.
  - 한번 더 클릭시 다시 a, b, c 가 출력된다.

```html
<!-- HTML -->
<div id="app">
  <div>a</div>
  <div>b</div>
  <div>c</div>
  <div>d</div>
  <div>e</div>
  <div>f</div>
  <div>g</div>
  <button>go</button>
</div>
```



v-for

v-for

- 화면에 DOM 요소를 반복적으로 출력할 때 사용
- 배열 각각의 요소 순회, 인덱스도 받을 수 있음
- 키를 지정하지 않으면 warning 발생:
  - 키는 반복적으로 생성된 각각의 DOM 요소를 식별할 수 있어야

```
<body>
    <div id="app">

        <li v-for="(todo, index) in todos"  v-bind:key="index">
            {{ todo }}
        </li>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.8/dist/vue.min.js"></script>
    <script>
        const app = new Vue({
            el: "#app",
            data() {
                return {
                    todos: ["밥먹기", "잠자기", "코딩하기"],
                }
            },
        });
    </script>
</body>
```

- 밥먹기
- 잠자기
- 코딩하기

[도전] bucket List 만들기

v-for을 활용해서 bucketList 작성후 출력[제한시간 : 15분]

- 다음과 같이 객체를 다섯 개 이상 생성 한 후 v-for로 출력하기
  - 단 done이 true인경우에만 출력하기

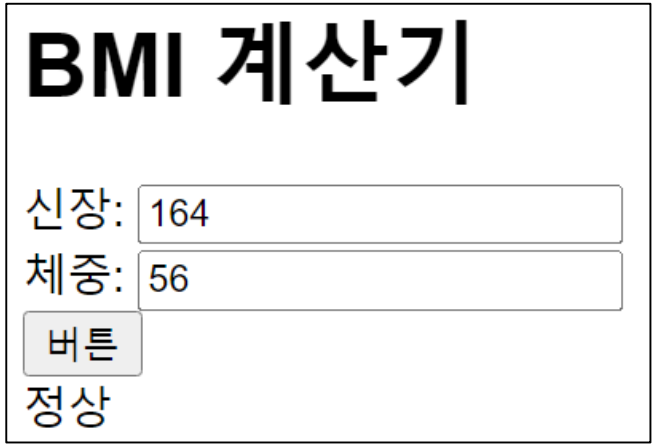
```
buckets: [
  { id: 1, text: "치킨먹기", done: true },
  { id: 2, text: "마라탕먹기", done: true },
  { id: 3, text: "놀러가기", done: false },
  { id: 4, text: "코딩하기", done: false },
  { id: 5, text: "티비보기", done: true },
],
```

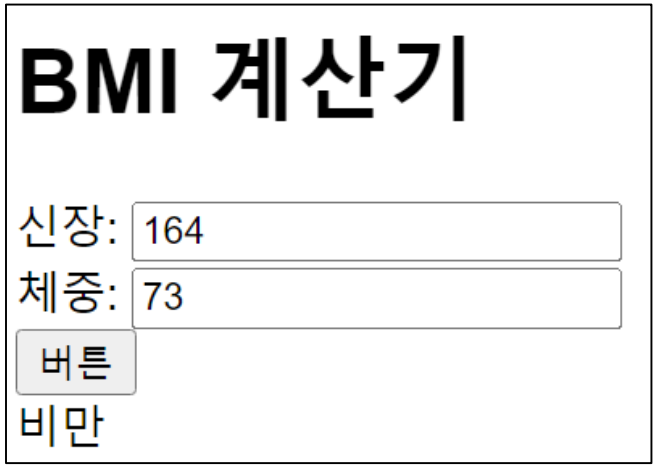
- 치킨먹기
- 마라탕먹기
- 티비보기

[도전] bmi 계산기 만들기

bmi 계산기

- input 에 담겨오는 값은 text의 형식이다. 형변환을 한 후 비교하자
- bmi 계산하는 법
  - 무게 / (키 * 키)
    - 조건
      - 0 (저체중)
      - 18.5 (정상)
      - 23 (과체중)
      - 25 (비만)





SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

3장. 라이프사이클

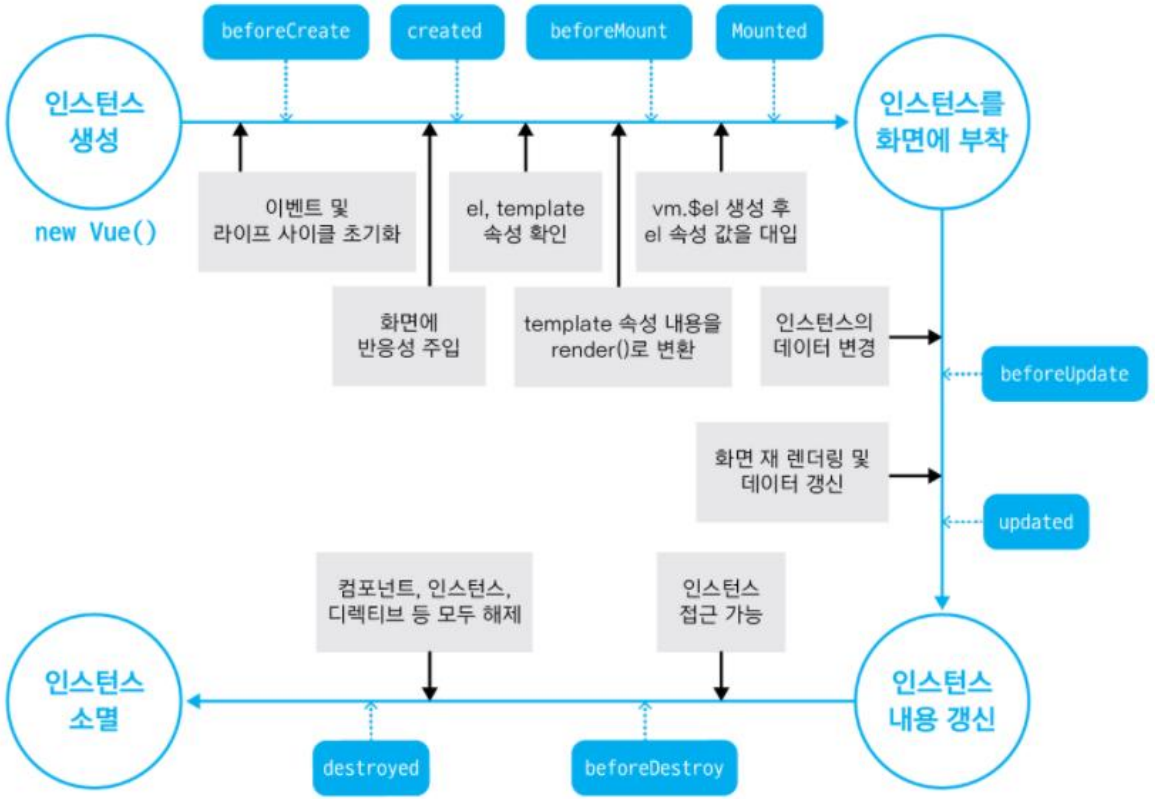
SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

라이프 사이클

라이프 사이클

컴포넌트 생성부터 소멸까지의 주기



Do it! vue js 입문

자주 쓰는 life cycle

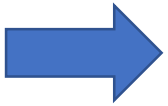
- created()
  - 인스턴스가 생성되었지만 아직 화면에 붙여지지 않은 단계에 자동 실행
  - 데이터 갱신에 적합
  - 비동기 통신 호출 가능 시점
- mounted()
  - 인스턴스가 화면에 부착된 이후 자동 실행
  - DOM에 접근이 가능
    - this.$refs 사용

[실습] created()

데이터 가져오기

- axios 사용
  - axios.get("https://jsonplaceholder.typicode.com/todos/1")
  - 라이프 사이클을 활용해 axios를 호출하기

```
<body>

    <div id="app">
        {{todo}}
        <div>{{todo.id}}</div>
        <div>{{todo.title}}</div>
        <div>{{todo.completed}}</div>
    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.8/dist/vue.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
    <script>
        const app = new Vue({
            el: "#app",
            data() {
                return {
                    todo: {}
                }
            },
            methods: {

            },
            async created() {
                const result = await axios.get("https://jsonplaceholder.typicode.com/todos/1");
                console.log(result);
                this.todo = result.data;
            }
        });
    </script>
</body>
```



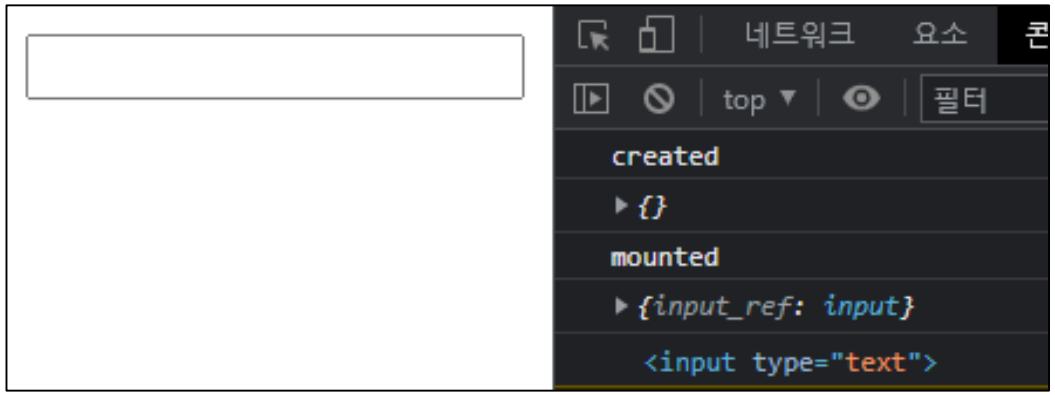
{ "userId": 1, "id": 1, "title": "delectus aut autem", "completed": false }
1
delectus aut autem
false

[실습] mounted

mounted

- DOM에 접근이 가능한 시점
  - Vue.js 에서는 Virual DOM 을 활용하기 때문에 예외 상황을 제외하고 DOM에 접근을 권장하지는 않는다.
    - 예외 상황) input에 foucs
  - DOM에 접근하는 경우 $refs를 활용한다

```
<body>
    <div id="app">
        <input ref="input_ref" type="text">
    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.8/dist/vue.min.js"></script>
    <script>
        const app = new Vue({
            el: "#app",
            data() {
                return {
                    todo: {}
                }
            },
            methods: {

            },
            created(){
                console.log("created");
                console.log(this.$refs);
            },
            mounted() {
                console.log("mounted");
                console.log(this.$refs);
                console.log(this.$refs.input_ref);
                this.$refs.input_ref.focus();
            }
        });
    </script>
</body>
```

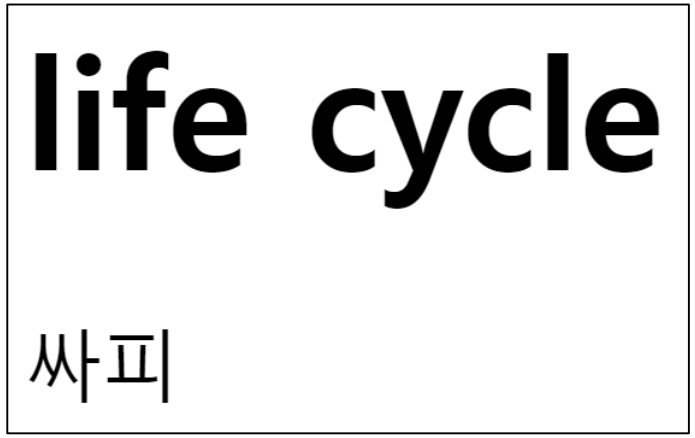



[도전] mounted()

글자 색 변경

- mounted 및 $refs 활용
  - $ref를 활용해서 SSAFY의 글자를 파란색으로 변경시키기
    - el.style.color = “blue”

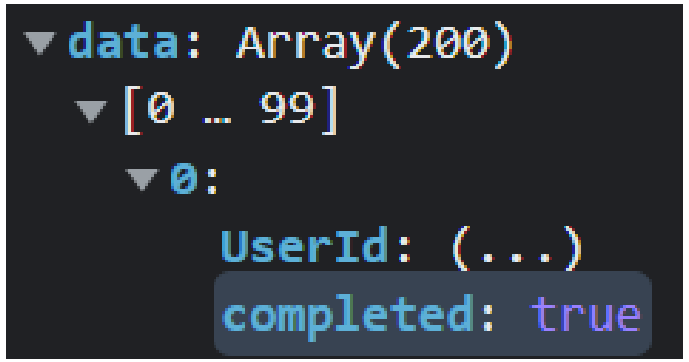
```
<div id="app">
    <h1>life cycle</h1>
    <p>싸피</p>
</div>
```



[도전] created()

데이터 가져오기

- axios 사용
  - axios.get("https://koreanjson.com/todos")
  - v-for 을 활용해서 나타내기
    - 단 가져온 data의 completed가 false인 경우에만 가져오기



대한민국의 영토는 한반도와 그 부속도서로 한다.
모든 국민은 양심의 자유를 가진다.
모든 국민은 인간다운 생활을 할 권리를 가진다.
모든 국민은 통신의 비밀을 침해받지 아니한다.
여자의 근로는 특별한 보호를 받으며, 고용·임금 및
대한민국은 민주공화국이다.
대한민국의 영토는 한반도와 그 부속도서로 한다.
모든 국민은 거주·이전의 자유를 가진다.
연소자의 근로는 특별한 보호를 받는다.
국가는 사회보장·사회복지의 증진에 노력할 의무를
정당의 목적이나 활동이 민주적 기본질서에 위배될
누구든지 체포 또는 구속을 당한 때에는 적부의 심
국가는 대외무역을 육성하며, 이를 규제·조정할 수
형사피고인은 상당한 이유가 없는 한 지체없이 공
모든 국민은 행위시의 법률에 의하여 범죄를 구성하
모든 국민은 학문과 예술의 자유를 가진다.
신체장애자 및 질병·노령 기타의 사유로 생활능력이
모든 국민은 법률이 정하는 바에 의하여 국가기관이
체포·구속·압수 또는 수색을 할 때에는 적법한 절차
에 해당하는 죄를 범하고 도피 또는 증거인멸의 염
모든 국민은 법 앞에 평등하다.
대한민국은 통일을 지향하며, 자유민주적 기본질서
형사피고인은 유죄의 판결이 확정될 때까지는 무죄
모든 국민은 고문을 받지 아니하며, 형사상 자기에
모든 국민은 인간으로서의 존엄과 가치를 가지며,
누구든지 체포 또는 구속을 당한 때에는 즉시 변호
대한민국은 국제평화의 유지에 노력하고 침략적 전
모든 국민은 신체의 자유를 가진다.
모든 국민은 행위시의 법률에 의하여 범죄를 구성하
국가유공자·상이군경 및 전몰군경의 유가족은 법률
국가는 평생교육을 진흥하여야 한다.
국가는 평생교육을 진흥하여야 한다.

[결론] Vue 를 사용해서 얻는 이점

사용시 직접적으로 느끼는 Vue 를 사용해서 얻는 장점

- 데이터와 HTML 부분을 일치화
  - 기존 data를 받아오면 해당 부분을 DOM에 넣어서 변경시키는 작업이 필요 없어짐
  - 데이터 관리가 굉장히 편해진다.

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

4장. Vue CLI

챕터의 포인트

- Vue CLI 개요
- 컴포넌트

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Vue CLI 개요

vue-cli

- Vue.js 개발환경을 설정해주는 도구
- 기본적인 프로젝트 세팅 제공
- 각각의 컴포넌트를 .vue 파일로 분리해 여러 개의 컴포넌트 사용 가능
- 클라이언트에서 node.js 테스트 서버 사용

vue-cli global 설치

```
$ npm i @vue/cli -g
```

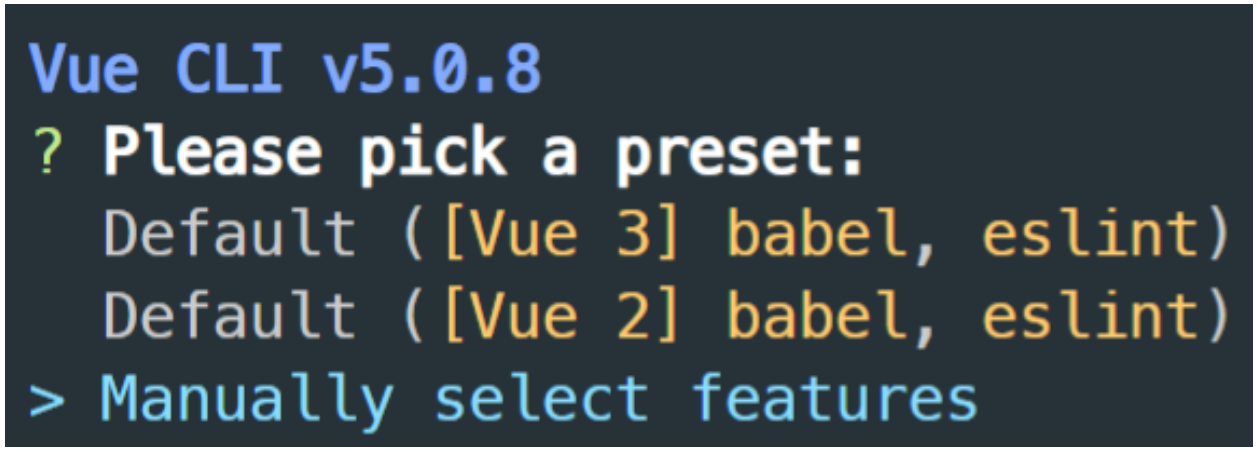
i 는 install 의 약자이며, init 과는 다르다.

```
$ npm init      package.json 생성
$ npm install   npm 패키지 설치
```

vue 프로젝트 생성

$ vue create 프로젝트이름

- 터미널의 현재 경로에, 입력한 프로젝트이름을 가진 Vue.js 템플릿 생성
  - vue create .
    - create시 . 를 입력하면 현재 폴더이름 그대로 프로젝트만 설치한다.
  - vue create main
    - main 폴더를 가진 vue-cli를 설치한다.
- Manually Select features 선택



프로젝트에 필요한 설정 선택

- Space Bar 로 선택
- Babel 은 필수
  - ES6 이상의 JavaScript 를 사용하는
    다른 브라우저와의 호환성을 위한 패키지
- Linter
  - 코딩 스타일 가이드
  - 관습적으로 지키는 규칙을 어길 시, warning 일으킴
  - 기본적으로 선택되어 있는데,
    원활한 실습을 위해 체크 해제
- Router, Vuex 선택

```
? Check the features needed for your project:
invert selection, and <enter> to proceed)
>(*) Babel
 ( ) TypeScript
 ( ) Progressive Web App (PWA) Support
 (*) Router
 (*) Vuex
 ( ) CSS Pre-processors
 ( ) Linter / Formatter
 ( ) Unit Testing
 ( ) E2E Testing
```

vue 프로젝트 생성

사용할 Vue.js 버전 선택

- 우리 수업에선 Vue 2 사용

- router 히스토리모드
  - Y 입력

- 설정 파일 저장 위치
  - In package.json 선택

- 현재 설정 저장 여부
  - N 입력

```
? Choose a version of Vue.js that you want to
 start the project with
  3.x
> 2.x
```

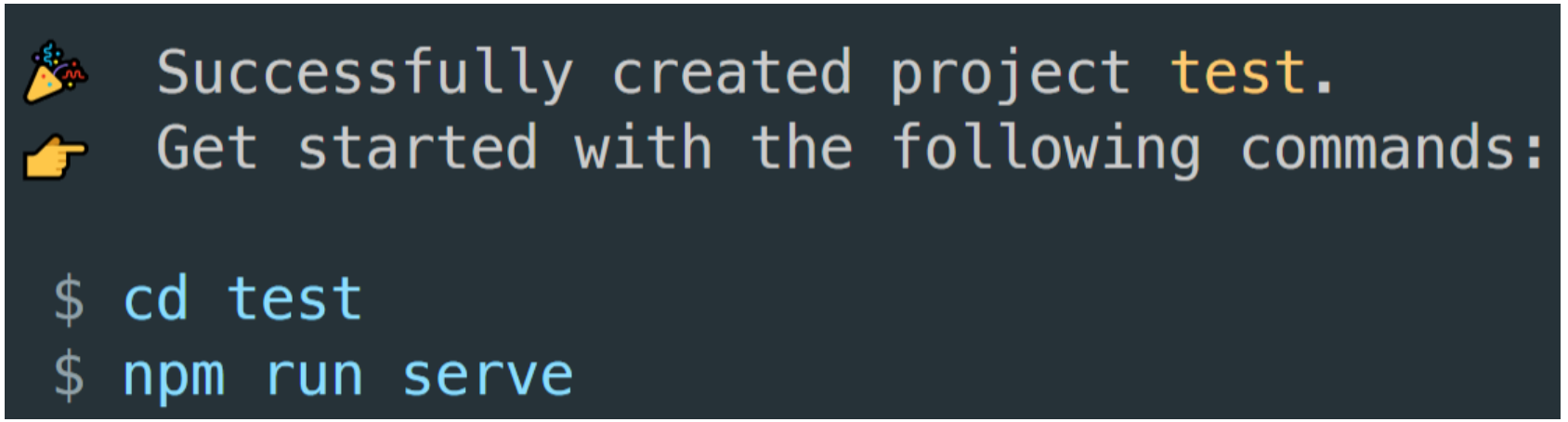
```
? Use history mode for router? (Requires proper server
setup for index fallback in production) (Y/n) Y
```

```
? Where do you prefer placing config for
Babel, ESLint, etc.?
  In dedicated config files
> In package.json
```

```
? Save this as a preset for future projects?
(y/N) N
```

다음 창이 보이면 성공

- 프로젝트 이름은 test 로 지정함
- cd 프로젝트이름
  - 해당 프로젝트 디렉터리로 이동
- npm run serve
  - 개발 테스트서버 작동



각 프로젝트 별 명령어 확인

package.json 살펴보기

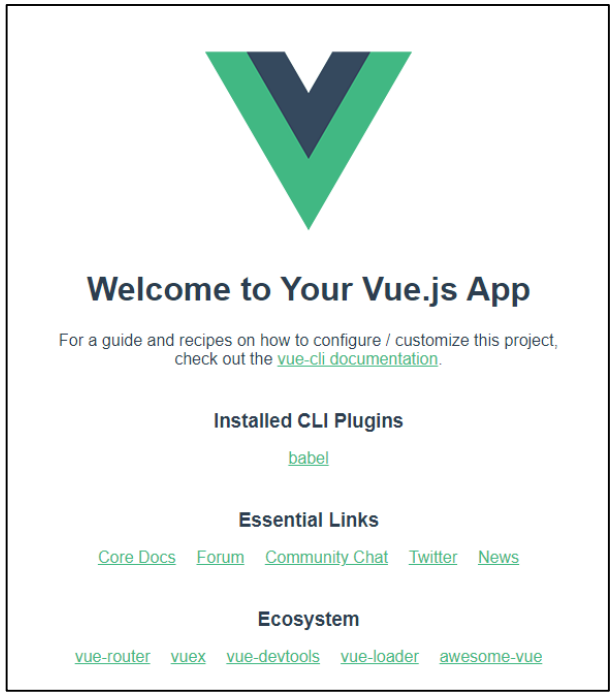
- scripts
  - 해당 부분에 명령어 script가 기재되어있다.
  - npm run +script 방식
    - npm run serve
    - npm run build

```
{
  "name": "1_cli",
  "version": "0.1.0",
  "private": true,
  ▷ 디버그
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build"
  },
  "dependencies": {
    "core-js": "^3.8.3",
    "vue": "^2.6.14",
    "vue-router": "^3.5.1",
    "vuex": "^3.6.2"
  },
  "devDependencies": {
    "@vue/cli-plugin-babel": "~5.0.0",
    "@vue/cli-plugin-router": "~5.0.0",
    "@vue/cli-plugin-vuex": "~5.0.0",
    "@vue/cli-service": "~5.0.0",
    "vue-template-compiler": "^2.6.14"
  },
  "browserslist": [
    "> 1%",
    "last 2 versions",
    "not dead"
  ]
}
```

테스트 서버 작동

- npm run serve
  - 다음 메세지가 출력되면 성공
  - 연결된 주소로 접속
    - http://localhost:8080
      - 8080 포트를 local에서 사용중이면 8081.. 식으로 포트가 변경되어 출력된다

```
DONE  Compiled successfully in 2992ms


App running at:
- Local:   http://localhost:8080/
- Network: http://192.168.0.94:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.
```



만약 다음과 같은 에러가 발생한다면

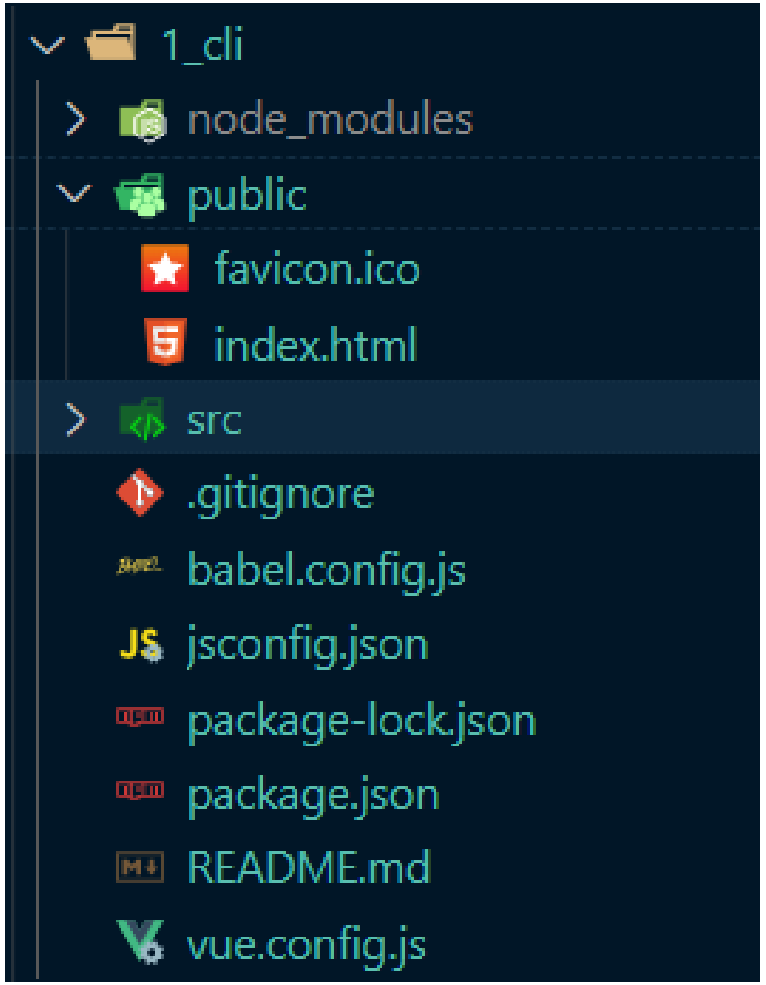
- 프로젝트 디렉터리 전체 경로 중 특수문자가 있다면 삭제
  - c:/Users/ssafy/Documents/my-vue-projects(test)/1_(실습)template/
    - 예시의 경우, 소괄호가 경로에 입력되어 있음
    - "실습" 처럼, 한글이 경로에 포함되어 있더라도 에러 발생 안함
    - 경로에 특수문자만 조심

```
ERROR  Failed to compile with 1 error                                    오후 2:32:02

error

Conflict: Multiple assets emit different content to the same filename index.html

ERROR in Conflict: Multiple assets emit different content to the same filename index.html
```

프로젝트 구조 파악

프로젝트 구조

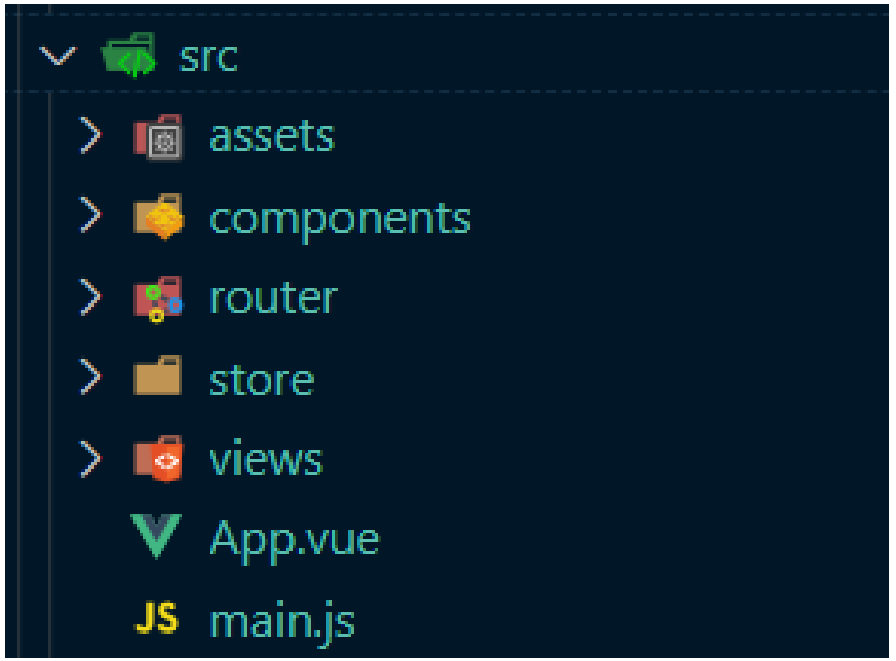
- 알아 둘 사항
  - npm i로 vue-cli를 설치한 만큼 node 상에서 돌아간다.
    - 그로 인해 npm run serve 등 서버 기동도 가능
    - npm run build 시 정의 된 내용들을 모아서 하나의 index.html 및 자바스크립트 형태로 변형시킨다.
- node_modules
  - npm install 시 생성되는 폴더
  - package.json 을 기반으로 필요 module을 설치하고 모아둔다.
- public
  - Vue는 SPA(single Page Application)
    - 단 하나의 index.html만 로딩 그 외 내용들은 javascript를 활용하는 방식
- src
  - vue application의 핵심 디렉토리
  - 전반적인 코드 작성을 이곳에서 한다.



프로젝트 구조 파악

프로젝트 구조(src)

- assets
  - css나 정적 파일(image)등을 저장하는 디렉토리
- components
  - 컴포넌트식 개발에 필요한 컴포넌트들을 저장
- router
  - router 관련 디렉토리
- store
  - Vuex 관련 디렉토리
- views
  - router 마다 보여질 컴포넌트를 모아둔 디렉토리
- App.vue
  - 어플리케이션의 최상위 컴포넌트
    - vue-cli 에서는 .vue 파일이 기본 형태로 되어있다.
- main.js
  - 어플리케이션의 진입점, 기본 셋팅 및 라이브러리 셋팅을 진행하는곳



main.js

- 어플리케이션의 진입점 이자 기초 셋팅이 진행
  - public/index.html 의 id app 을 읽어서 해당 부분을 Vue로 적용시킨다
  - router 와 store에 대한 설정이 되어있다.
  - vue-boostrap 및 외부 라이브러리도 해당 main.js 에서 진행 예정

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'

Vue.config.productionTip = false

new Vue({
  router,
  store,
  render: h => h(App)
}).$mount('#app')
```

main.js

```
<!DOCTYPE html>
<html lang="">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <link rel="icon" href="<%= BASE_URL %>favicon.ico">
    <title><%= htmlWebpackPlugin.options.title %></title>
  </head>
  <body>
    <noscript>
      <strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn
    </noscript>
    <div id="app"></div>
    <!-- built files will be auto injected -->
  </body>
</html>
```

public/index.html

App.vue 분석

App.vue

- 모든 컴포넌트의 최상위 컴포넌트
  - 기본적으로 모든 요청이 App.vue를 거쳐간다.
- template
  - html 부분
- style
  - css 부분
- script
  - App.vue에는 정의되어있지 않지만
    vue 관련 script 부분은 해당부분에 정의
  - template과 style 사이에 정의한다.

```vue
<template>
  <div id="app">
    <nav>
      <router-link to="/">Home</router-link> |
      <router-link to="/about">About</router-link>
    </nav>
    <router-view/>
  </div>
</template>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
}

nav {
  padding: 30px;
}

nav a {
  font-weight: bold;
  color: #2c3e50;
}

nav a.router-link-exact-active {
  color: #42b983;
}
</style>
```

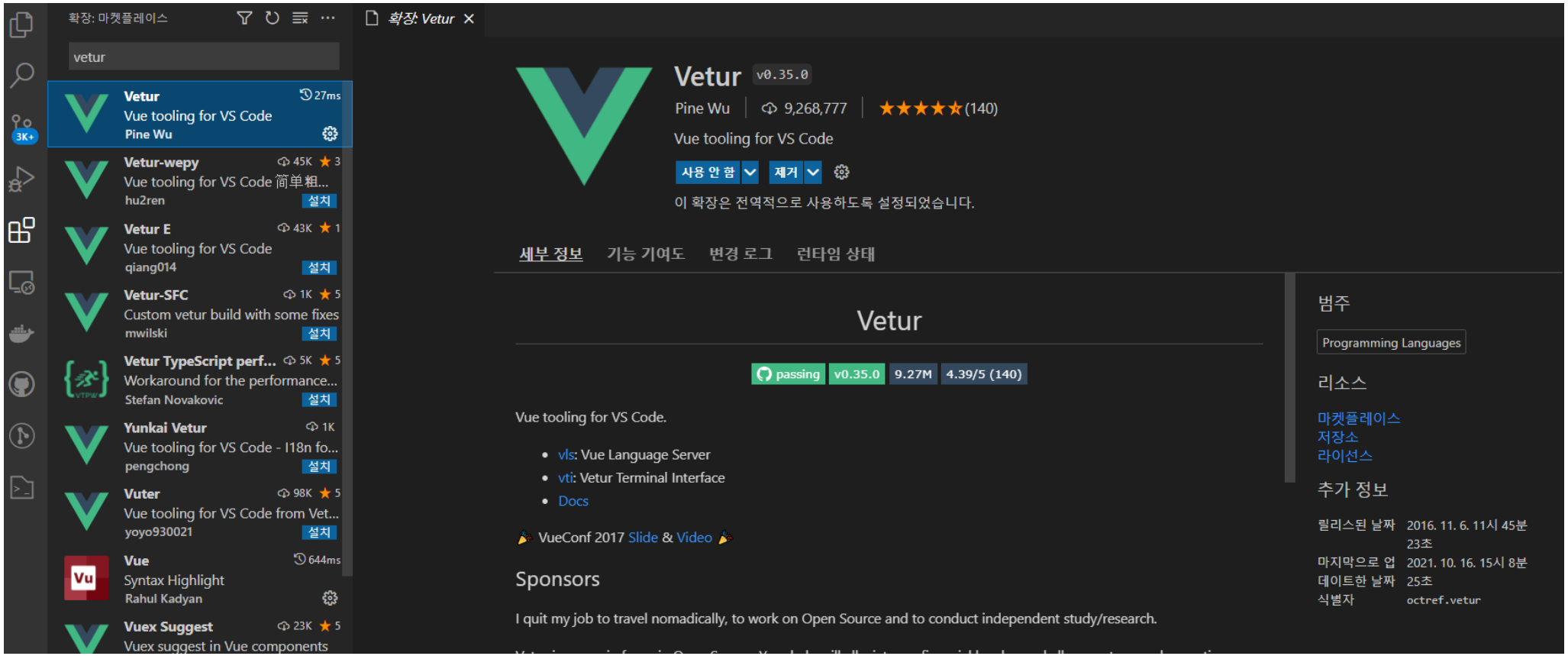
```vue
</template>

<script>
export default {
  name: 'App',

}
</script>
<style>
```

Vetur 설정

Confidential

Vetur

- 가장 인기있는 VSCode Vue.js 확장
  - 자동완성, 컬러 하이라이팅 지원



라우터 코드 수정

- src/router/index.js 수정
  - 에러 방지를 위해, 라우터 코드를 수정하자.
  - 이미 삭제한 파일을 찾을 수 없는 에러 방지

```
import Vue from 'vue'
import VueRouter from 'vue-router'

Vue.use(VueRouter)

const routes = []

const router = new VueRouter({
  mode: 'history',
  base: process.env.BASE_URL,
  routes
})

export default router
```

App.vue 수정

App.vue 파일을 다음과 같이 수정

- 하나의 .vue 파일은 다음 구조를 지닌다.
  - <template> - HTML
  - <script> - JavaScript
  - <style> - CSS

- 코드 작성 중 새로고침 불필요
  - VSCode 파일 저장을 기준으로,
  - 테스트서버는 재시작되고 브라우저도 자동 새로고침

```
<template>
  <div>
    <h1>Hello Vue.js</h1>
  </div>
</template>

<script>
export default {
  data(){
    return{

    }
  }

}
</script>
<style>

</style>
```

SAMSUNG SW ACADEMY FOR YOUTH

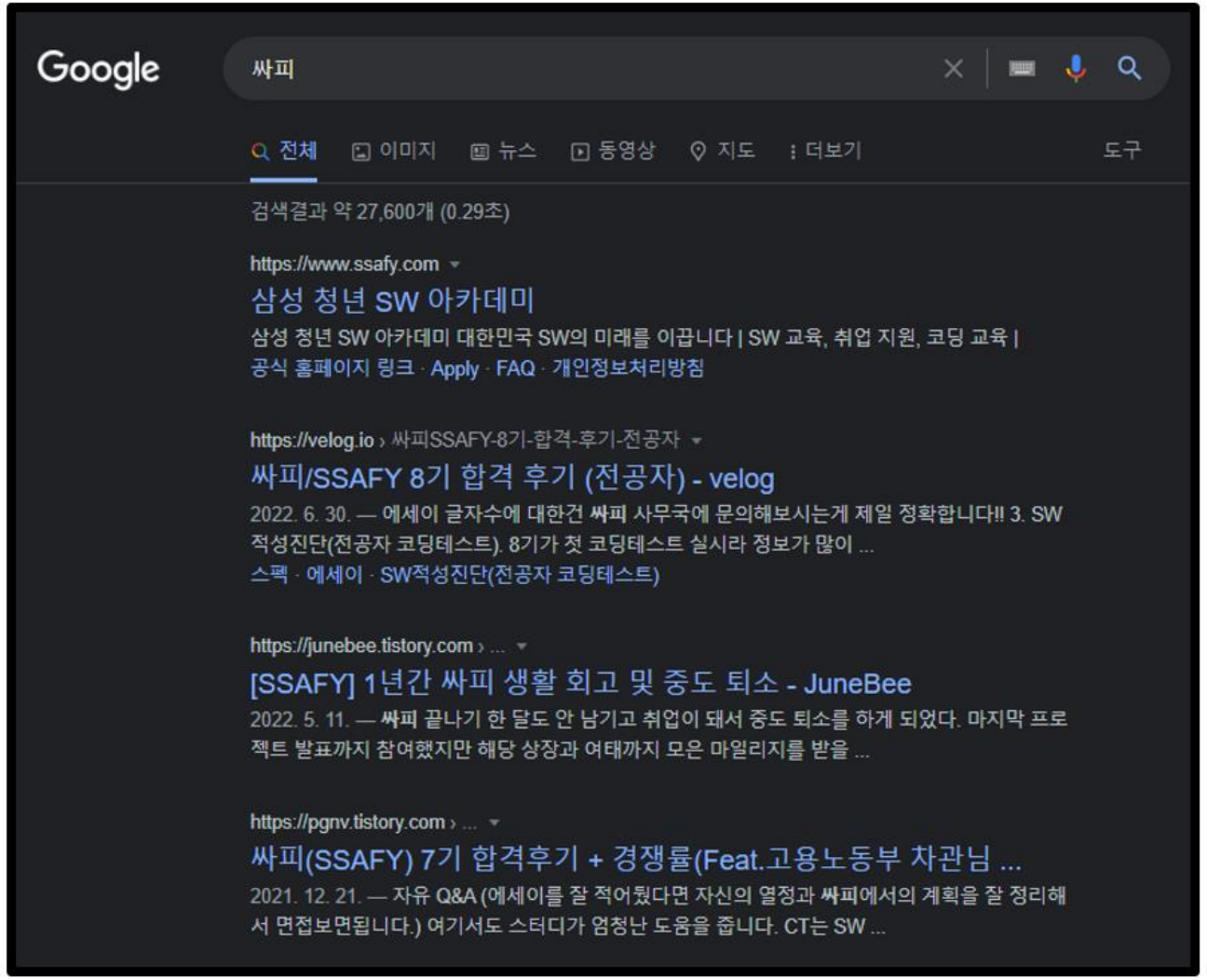
함께가요 미래로!
Enabling People

컴포넌트

일반적인 웹 개발 과정

구글 검색화면의 예

- 하나의 페이지 - 하나의 HTML 문서
- 만약, 하나의 HTML 을 나눌 수 있다면?
  - 가독성 및 생산성 향상
  - 유지보수 원활



컴포넌트 단위로 나눔

- component: 부품
- 다음과 같이 나눈다고 가정

- Header
  - Search - 검색 창
  - Navigation - 메뉴 바
- Main
  - EachResult - 검색 결과

- 컴포넌트 안에 컴포넌트가 있을 경우,
  부모 컴포넌트, 자식 컴포넌트로 구.
- 이 중, EachResult 는 v-for 를 사용하

컴포넌트 방식 개발

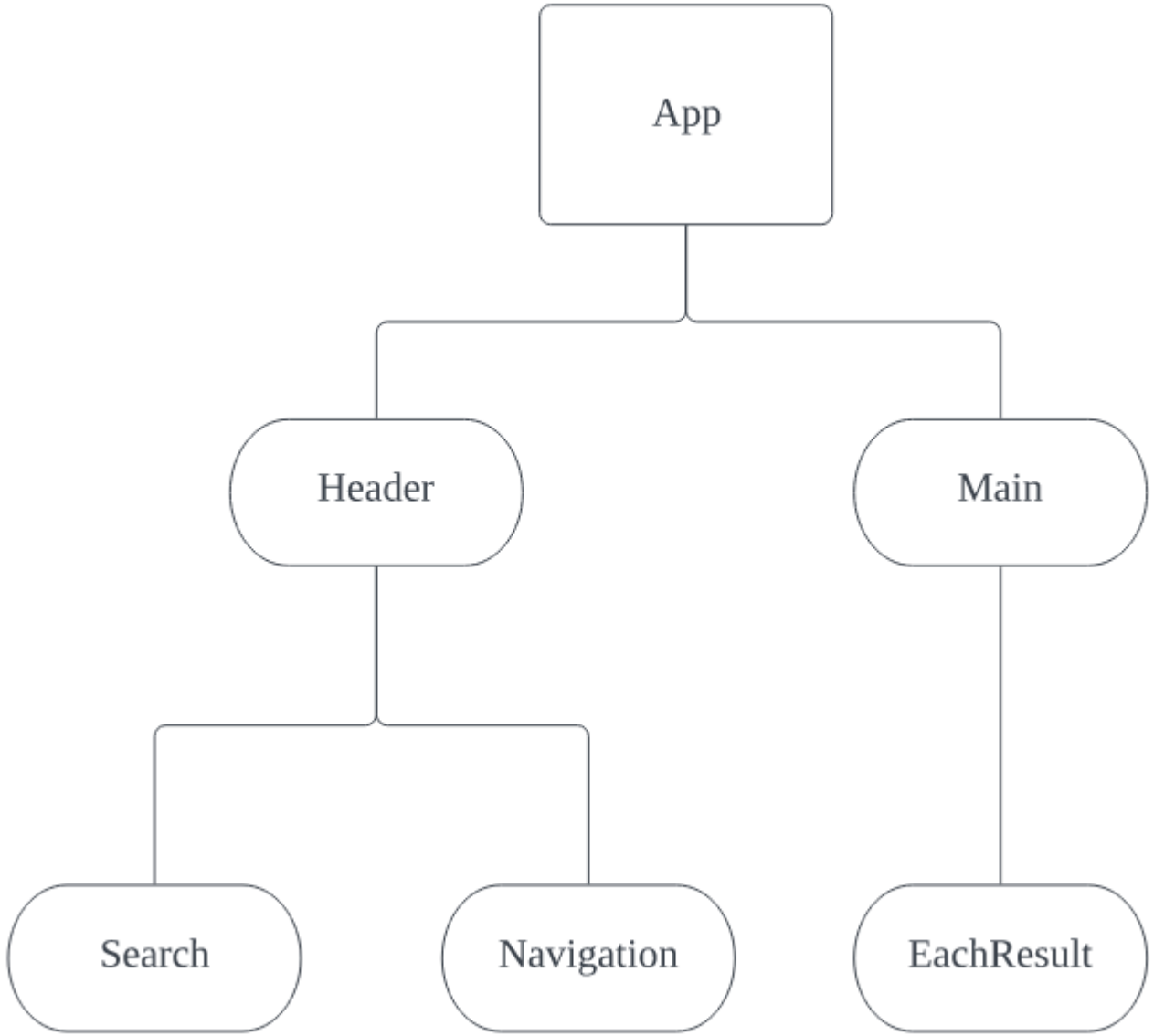
- 수십 개의 컴포넌트를 각각 개발
- 하나의 HTML 문서로 결합
- 각각의 컴포넌트는 필요에 따라 다른 컴포넌트에 재사용
- Vue CLI 사용해 개발시
  - 컴포넌트는 각각의 파일 단위 구분
  - .vue로 구분한다.

컴포넌트 트리

트리 형식으로 표현 가능

- App - 루트 컴포넌트
  - Header, Main : App 의 자식 컴포넌

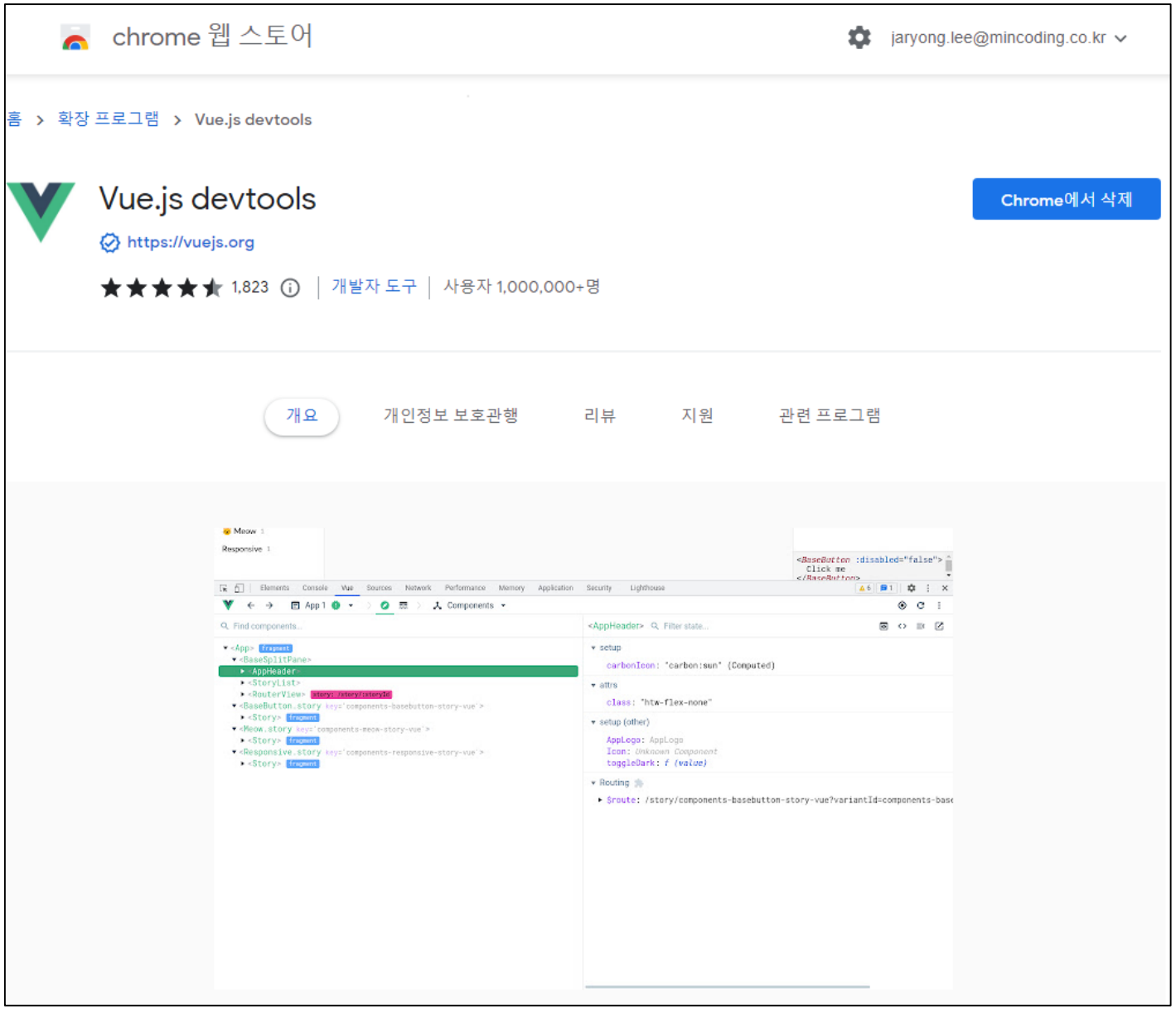
직접 구현해보자



Vue.js devtools 설치

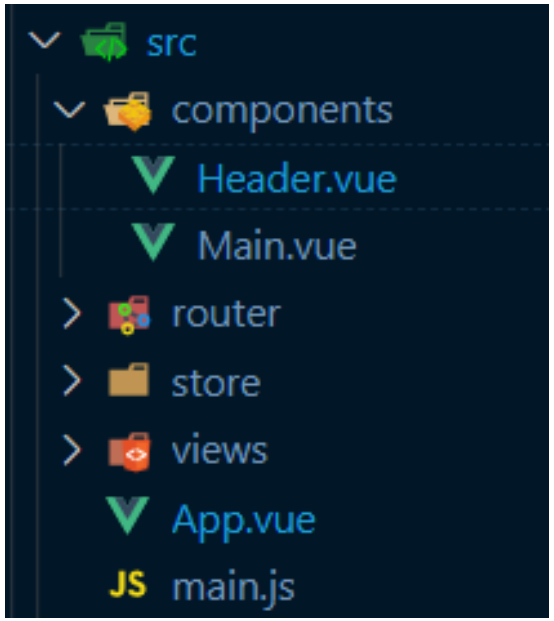
크롬 확장프로그램 설치

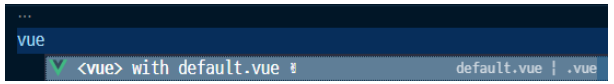
- 크롬 웹 스토어 접속
  - Vue.js devtools 설치

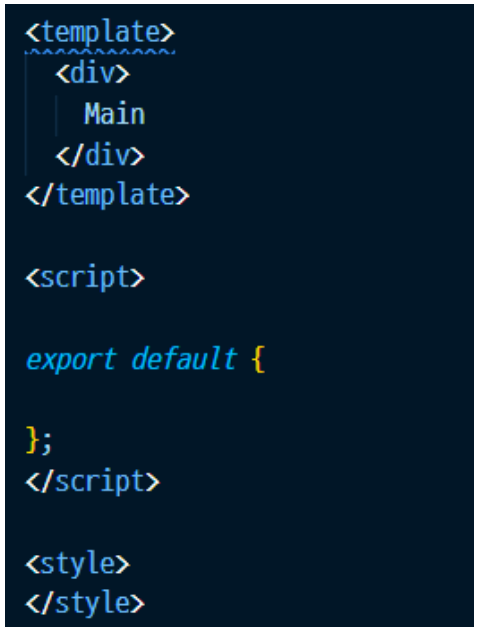


컴포넌트 구현

- components 폴더에 각 componet생성
- .vue 파일에서 vue 라고 입력 후
  Ctrl + Space(자동완성)
  실행시 기본 템플릿이 생성







```
<template>
  <div>
    Main
  </div>
</template>

<script>

export default {

};
</script>

<style>
</style>
```

Navigation.vue

컴포넌트 생성시 주의사항

- 컴포넌트들은 모두 하나의 루트 엘리먼트를 가지는 형태가 되어야한다.
  - 그렇지 않을 경우 에러가 발생

```
<template>
  <div>
    TEST
  </div>
</template>

<script>
export default {

}
</script>

<style>

</style>
```

```
<template>
  <div>
    TEST
  </div>
  <div>
    ERROR
  </div>
</template>

<script>
export default {

}
</script>

<style>

</style>
```

컴포넌트 활용

- 가져오기
  - import 컴포넌트명 from "상대경로"
  - import 는 ES Module 문법으로서,
    node.js 의 require 와 같은 의미

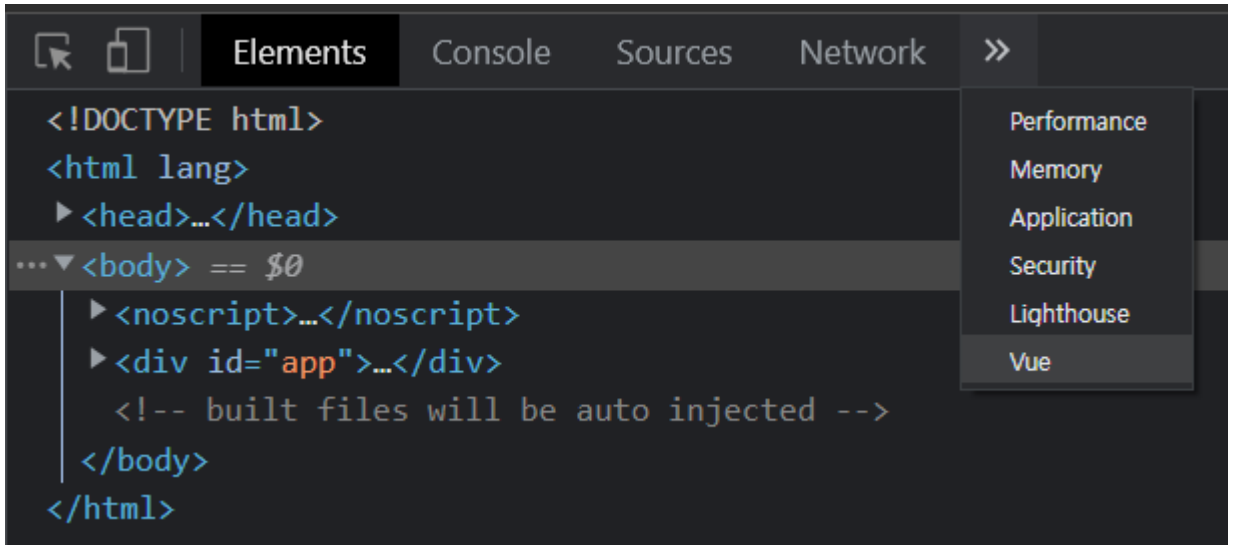
- 등록하기
  - 컴포넌트를 사용하기 위해,
    components 객체에 등록
  - import 를 제외한 모든 객체는
    export default 내부에 작성

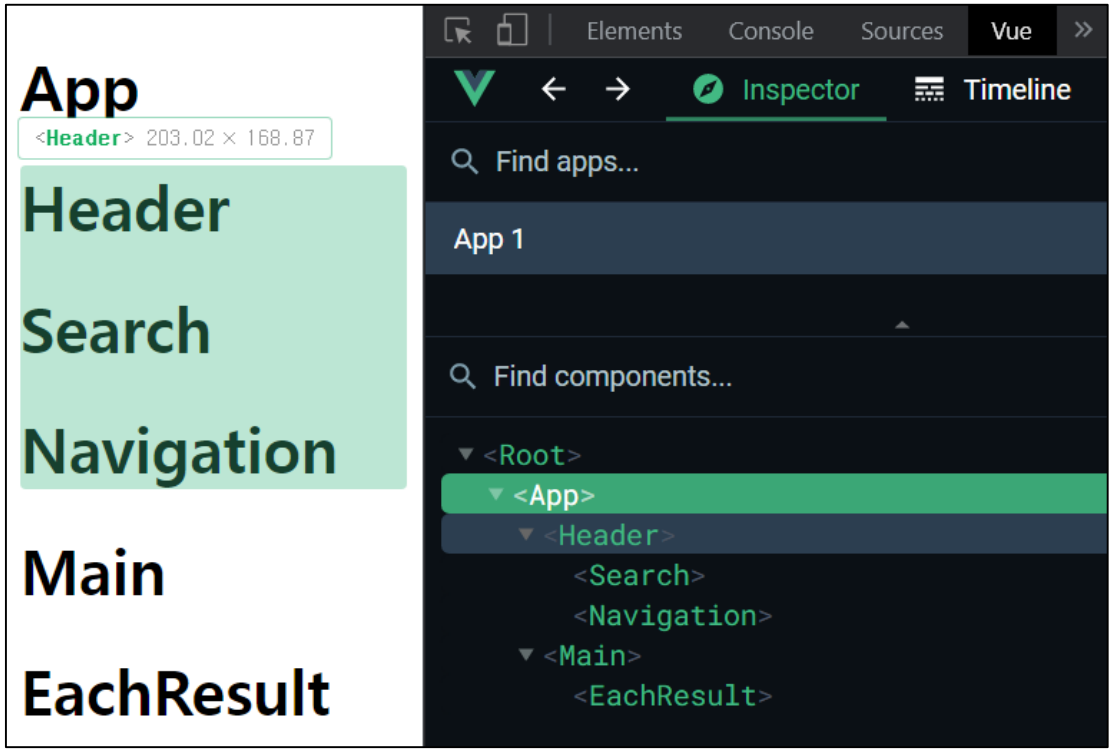
- 붙이기
  - 가져온 컴포넌트를 태그 사용법과 동일하게
  - 화면에 위치하고 싶은 곳에 기입
  - 이후 컴포넌트 간 부모 자식 관계가 성립

```vue
<template>
  <div>
    <h1>App</h1>
    <Header ></Header>
    <Main ></Main>
  </div>
</template>

<script>
import Header from "./components/Header";
import Main from "./components/Main";
export default {
  components: {
    Header,
    Main,
  },
};
</script>

<style>
</style>
```

Vue.js devtools 사용

- 개발자 도구에서 >> 누르고 Vue 클릭



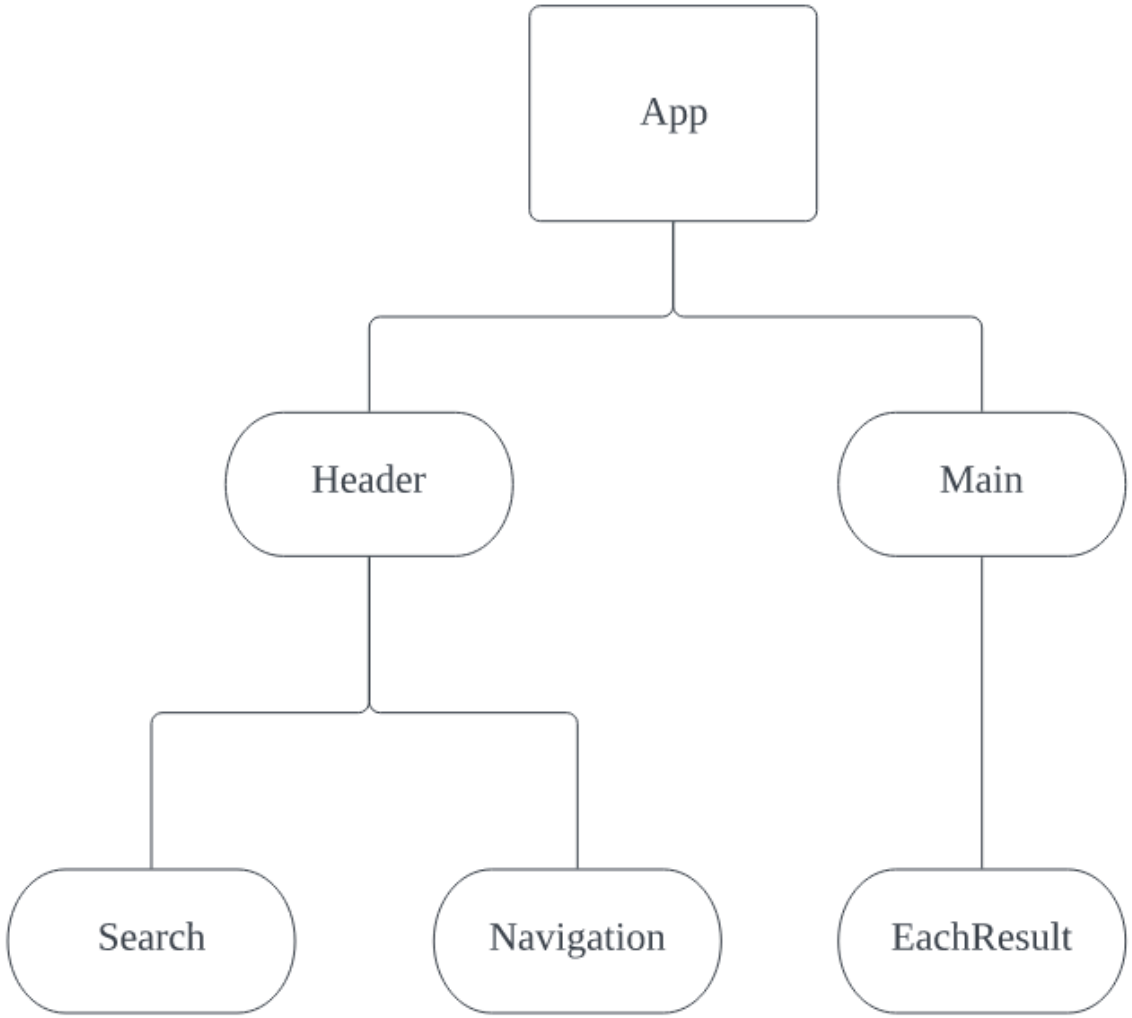


[도전] 컴포넌트 트리 구현

다음 컴포넌트 트리를 구현하

- Header 의 자식 컴포넌트
  - Search, Navigation
- Main 의 자식 컴포넌트
  - EachResult

- 구현 후 Vue.js devtools 로 결과 확인



SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

내일 방송에서 만나요!

삼성 청년 SW 아카데미