

Vue 강의

이병일

Vue.js('뷰')란?

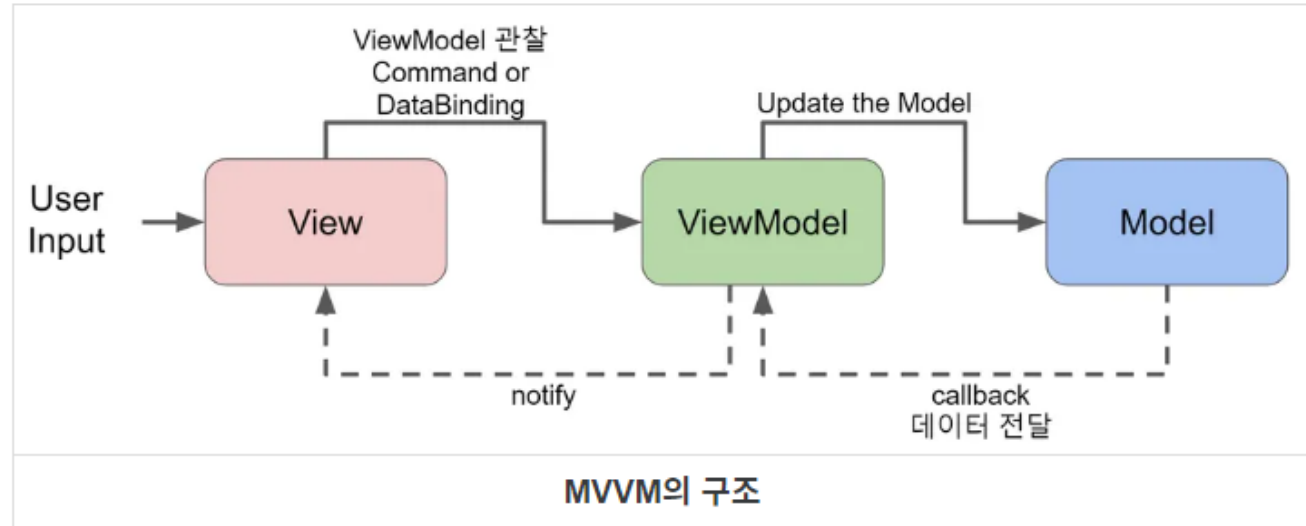
- ▶ 웹 페이지 화면을 개발하기 위한 프론트엔드 프레임워크
- ▶ 뷰는 화면단 라이브러리이자 프레임워크
- ▶ 뷰 코어 라이브러리는 화면단 데이터 표현에 관한 기능들을 중점적으로 지원하지만 프레임워크의 기능인 라우터, 상태관리, 테스트 등을 쉽게 결합할 수 있는 형태로도 제공됨
- ▶ 즉, 라이브러리 역할뿐만 아니라 프레임워크 역할도 할 수 있다는 의미
- ▶ 그래서 공식 사이트(www.vuejs.org)에서도 뷰를 점진적인 프레임워크라고 부름
- ▶ 뷰는 2014년 2월 처음으로 공식 배포되었음
- ▶ 구글에서 일하던 직원(에반 유)이 앵귤러를 더 가볍게 쓰고 싶어서 만든 프레임워크

Vue의 장점

- ▶ 배우기 용이함
- ▶ 리액트와 앵귤러에 비해 성능이 우수하고 빠름
- ▶ 리액트의 장점과 앵귤러의 장점을 갖고 있음

Vue의 특징

- ▶ UI 화면단 라이브러리
UI 화면 개발 방법 중 하나인 MVVM (Model-View-View Model)패턴의 뷰 모델에 해당하는 화면단 라이브러리
- ▶ 구글 검색처럼 화면의 요소가 변경되거나 조작이 일어날 때 즉각적으로 반응하여 화면의 데이터를 갱신하여 보여주는 역할



Model-View-View Model.

아키텍처 패턴 중 하나로, 애플리케이션을 데이터를 처리하는 모델(Model), 사용자에게 보여지는 UI인 뷰(View), 뷰에 바인딩되어 모델과 뷰 사이를 이어주는 뷰 모델(View Model)로 분리하는 방식이다.

Vue의 특징

▶ 또 하나의 가장 큰 특징은 컴포넌트 기반 프레임워크라는 점

- ▶ 컴포넌트는 마치 레고 블록과 같음
- ▶ 뷰의 컴포넌트를 조합하여 화면을 구성할 수 있음
- ▶ 컴포넌트 기반 방식으로 개발하면 재사용하기 쉽고 **HTML** 코드에서 화면의 구조를 직관적으로 파악할 수 있어 정해진 방식대로 컴포넌트를 활용하여 빠르게 구현할 수 있을뿐 아니라 남이 작성한 코드를 볼때도 수월함

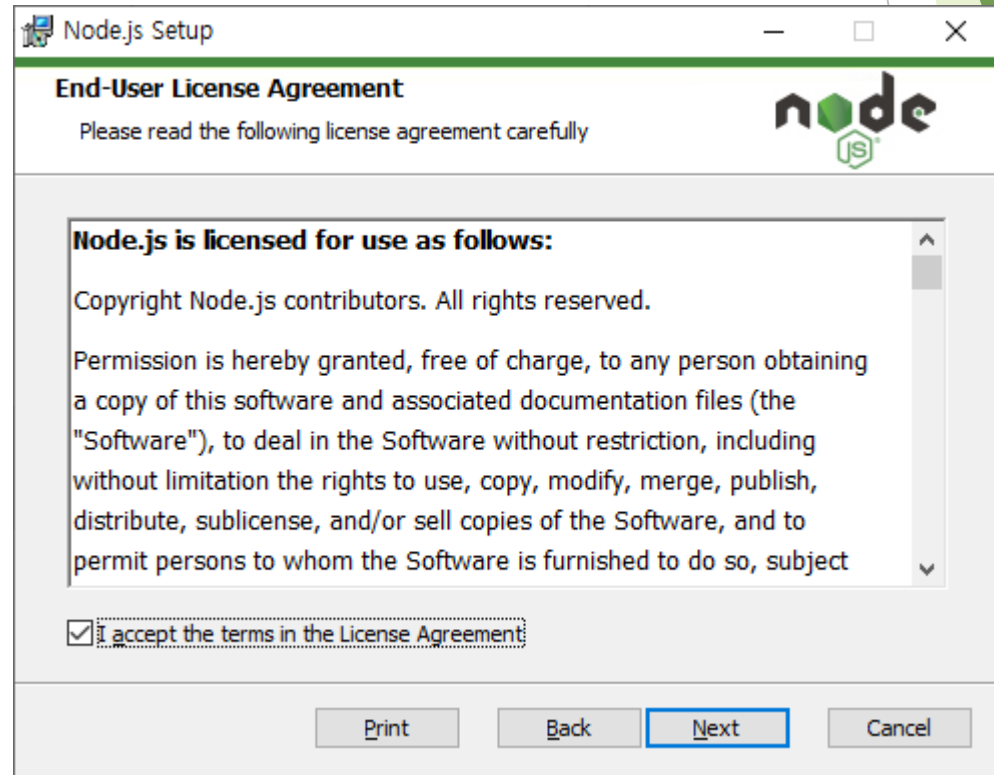
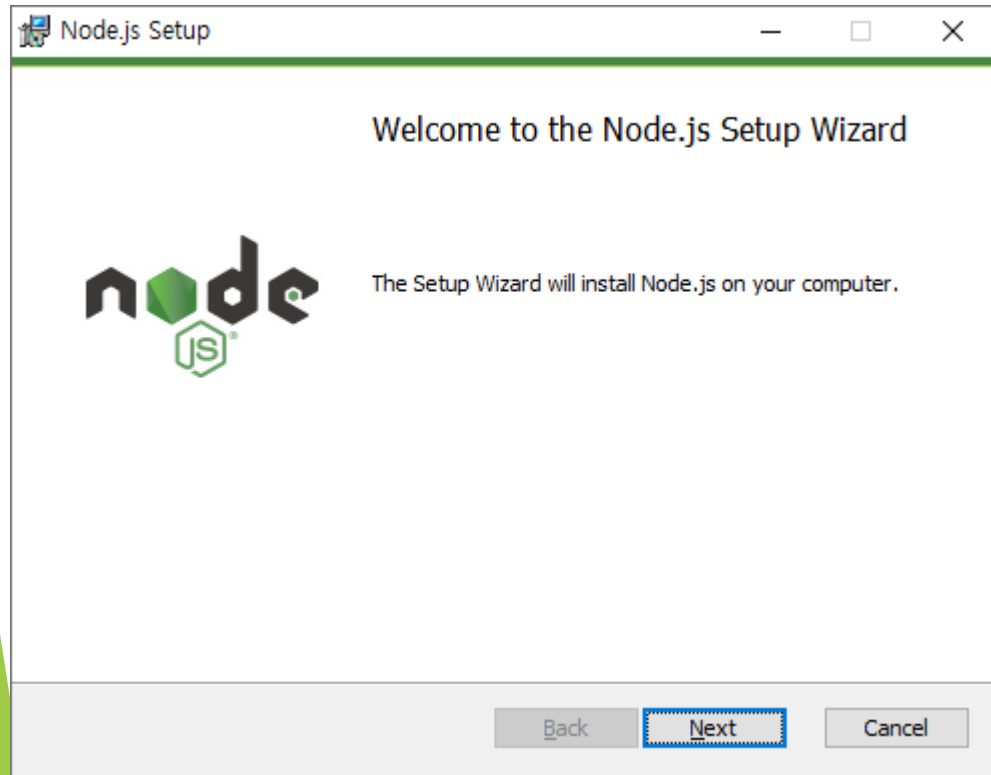
▶ 리액트와 앵귤러의 장점을 가짐

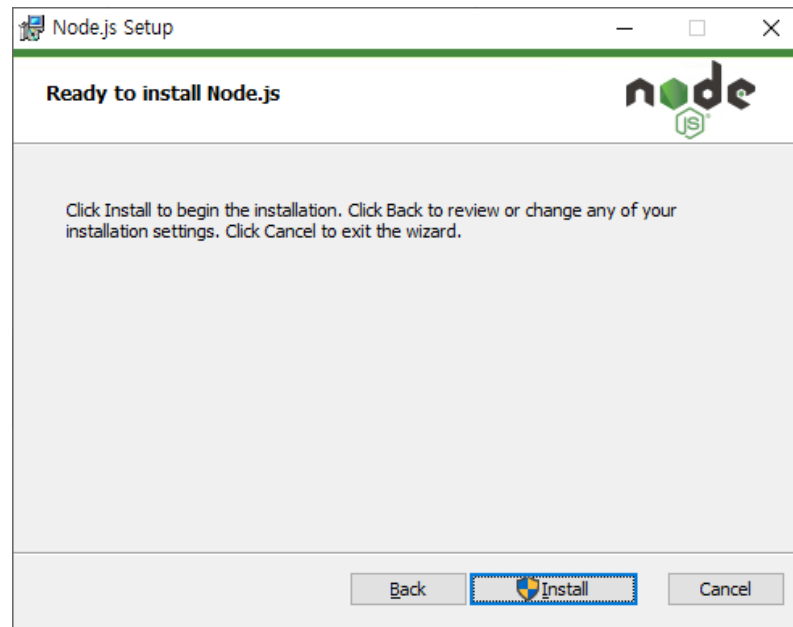
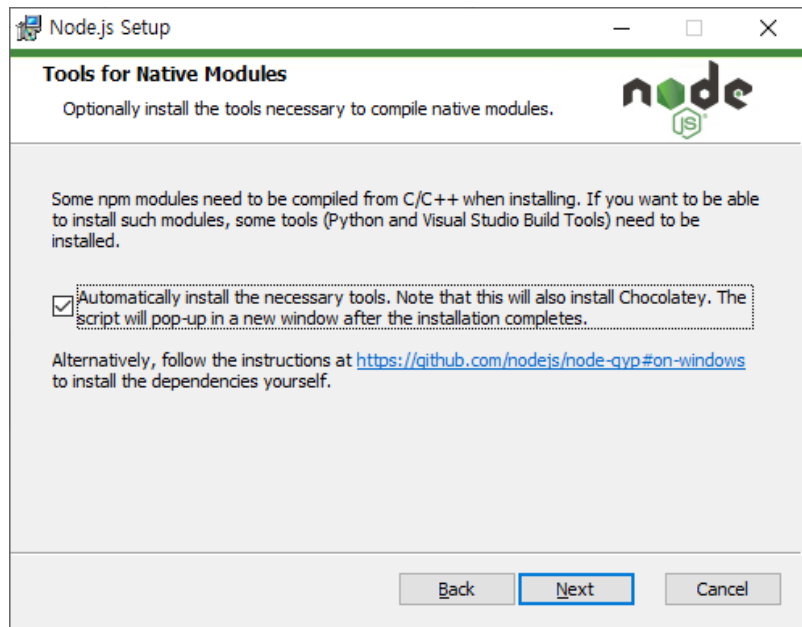
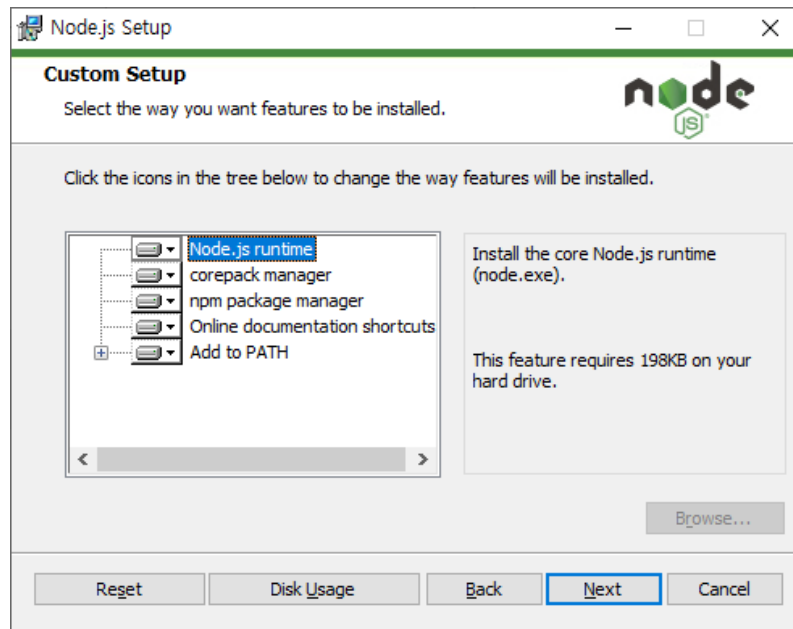
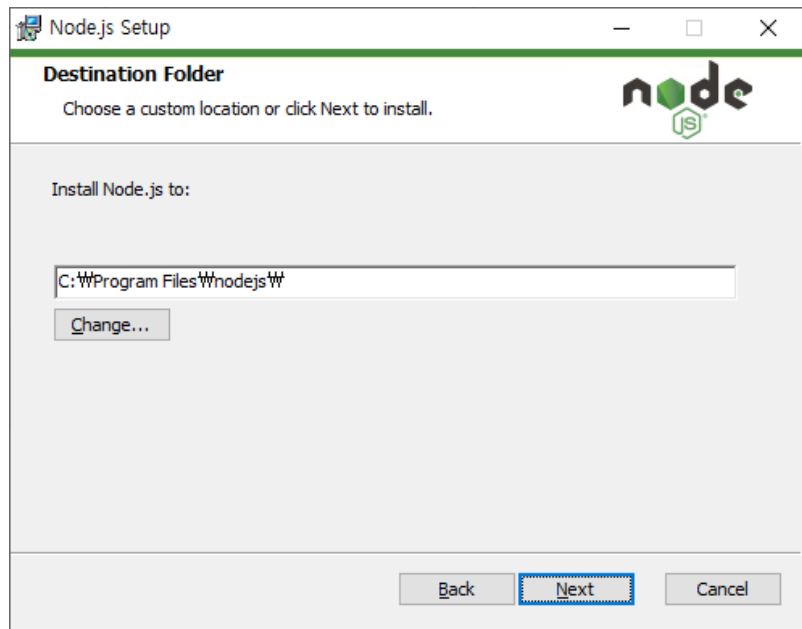
- ▶ 앵귤러의 양방향 데이터 바인딩과 리액트의 단방향 데이터 흐름의 장점을 모두 결합한 프레임워크임
- ▶ 양방향 데이터 바인딩이란 화면에 표시되는 값과 프레임워크의 모델 데이터 값이 동기화 되어 한쪽이 변경되면 다른 한쪽도 자동으로 변경되는 것을 말함
- ▶ 단방향 데이터 흐름은 컴포넌트의 단방향 통신을 의미하며 컴포넌트간에 데이터를 전달할 때 항상 상위 컴포넌트에서 하위 컴포넌트 한 방향으로만 전달하게끔 프레임워크가 구조화되어 있는게 바로 단방향 데이터 흐름

개발 환경 설정

- ▶ 크롬 브라우저 설치
- ▶ Visual Studio Code 설치(<https://code.visualstudio.com/download>)
- ▶ Node.js 설치 (<https://nodejs.org/en>)

Node.js는 JavaScript로 작성된 프로그램을 운영체제 상에서 일반 애플리케이션처럼 실행시켜주는 런타임 엔진으로 서버프로그램을 작성하는데 많이 사용됨





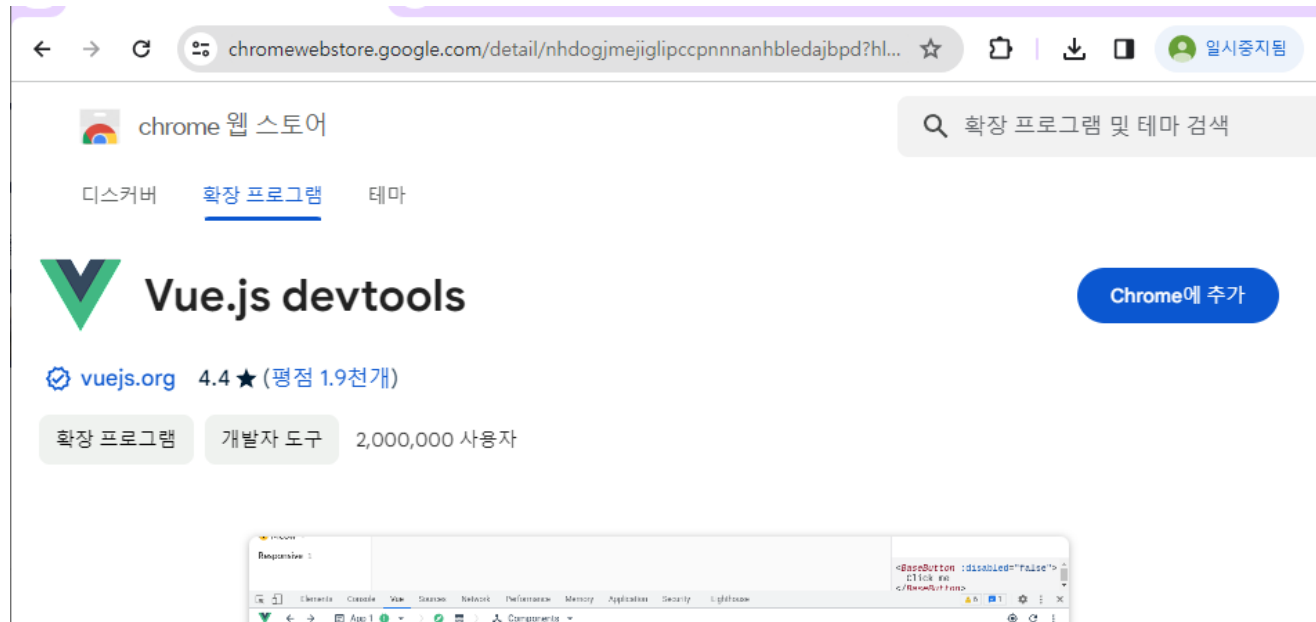
- ▶ 설치 확인을 위해 cmd 창을 열고 “**node -v**” 명령으로 버전 정보를 확인해 봄

```
cmd 명령 프롬프트
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

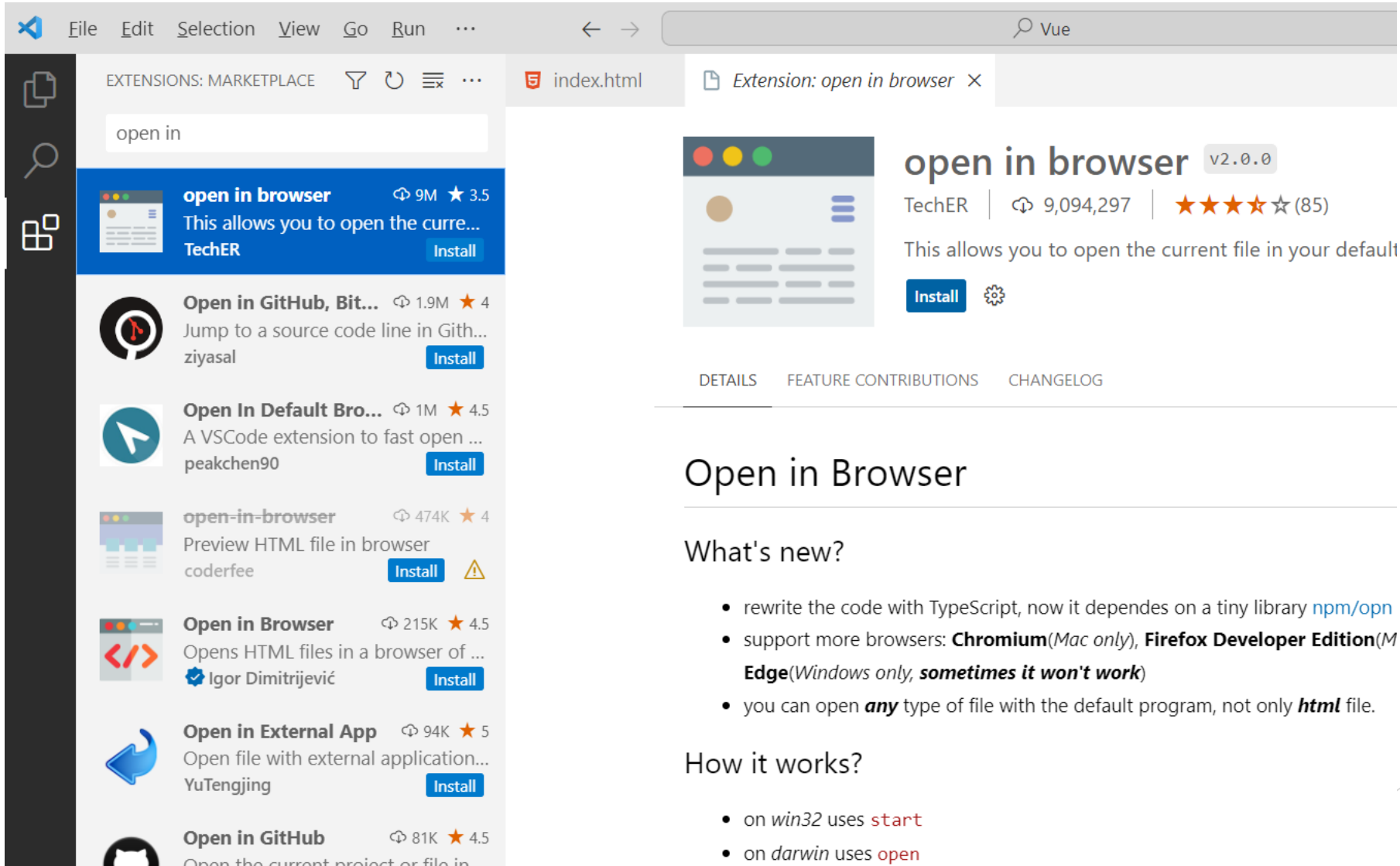
C:\Users\Wadmin>node -v
v20.10.0

C:\Users\Wadmin>
```

- ▶ Vue.js devtools 브라우저 확장 프로그램 설치
(<https://chromewebstore.google.com/detail/vuejs-devtools/nhdogjmejiglipccpnnanhbledajbpd?hl=ko>)



▶ VSCode에 open in browser 설치



The image shows the VS Code interface with the Extensions Marketplace open. The search bar contains 'open in'. The extension 'open in browser' by TechER is selected, showing its details. The extension has 9,094,297 downloads and a 4.5-star rating. The description states: 'This allows you to open the current file in your default browser.' The 'Install' button is visible. Below the extension details, there are tabs for 'DETAILS', 'FEATURE CONTRIBUTIONS', and 'CHANGELOG'. The 'DETAILS' tab is active, showing the title 'Open in Browser' and a section 'What's new?' with the following updates:

- rewrite the code with TypeScript, now it depends on a tiny library [npm/opn](#)
- support more browsers: **Chromium**(Mac only), **Firefox Developer Edition**(Mac only), **Edge**(Windows only, *sometimes it won't work*)
- you can open **any** type of file with the default program, not only **html** file.

Below this, there is a section 'How it works?' with the following information:

- on **win32** uses **start**
- on **darwin** uses **open**

The left sidebar shows the 'EXTENSIONS: MARKETPLACE' view with a search bar containing 'open in'. Below the search bar, a list of extensions is displayed, including 'open in browser', 'Open in GitHub, Bit...', 'Open In Default Bro...', 'open-in-browser', 'Open in Browser', 'Open in External App', and 'Open in GitHub'.

▶ Visual Studio Code Extensions 설치

- **vue**의 확장자 파일을 다룰 수 있고, **Vuter**를 비롯하여 문법 하이라이팅, 인텔리센스, 포매팅과 같은 다양한 기능을 지원하는 **Vue**를 위한 확장 플러그인

The screenshot shows the Visual Studio Code interface with the 'vue vs code extension pack' search results. The left sidebar shows the search bar and a list of extensions. The main panel displays the details for the 'Vue VS Code Extension Pack' by sarah.drasner, including its version (v0.2.0), download count (315,332), and a list of included extensions like Prettier and JavaScript snippets.

확장: 마켓플레이스

vue vs code extension pack

Vue VS Code Extension Pack v0.2.0
sarah.drasner | 315,332 | ★★★★★ (12)
A collection of extensions for working with Vue Applications in VS Code

설치

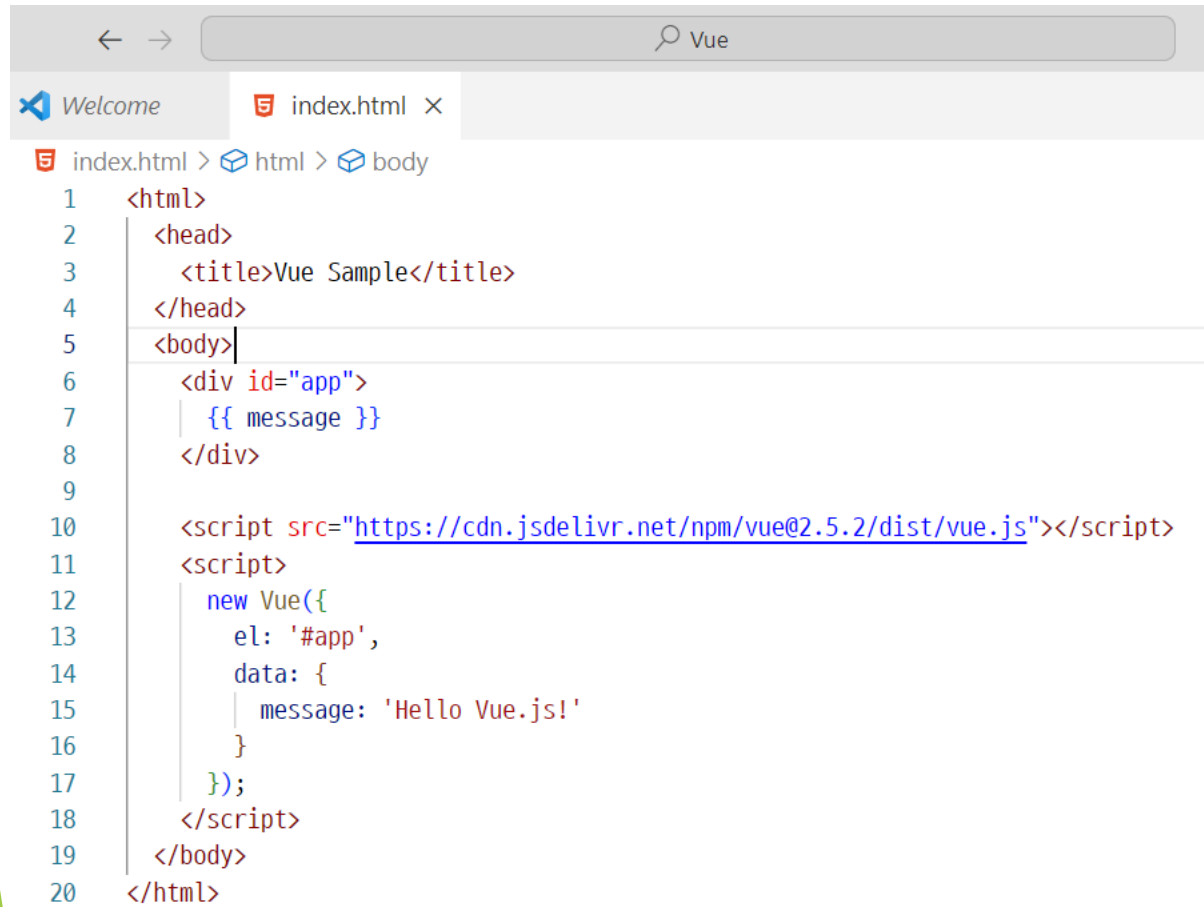
세부 정보

확장 팩(10)

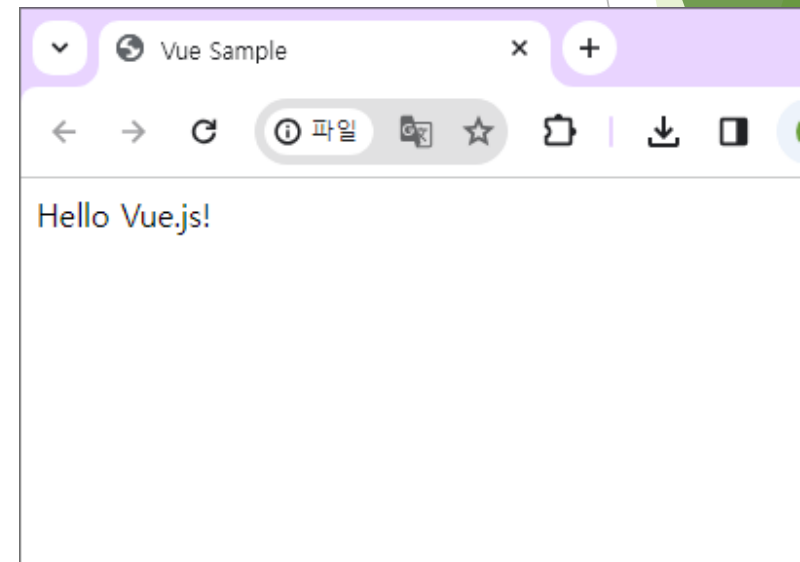
- Prettier - Code formatter**
Code formatter using prettier
Prettier
- JavaScript (ES6) code snippets**
Code snippets for JavaScript in ES6 syntax
charalampos karypidis

Hello Vue.js! 프로젝트 만들기

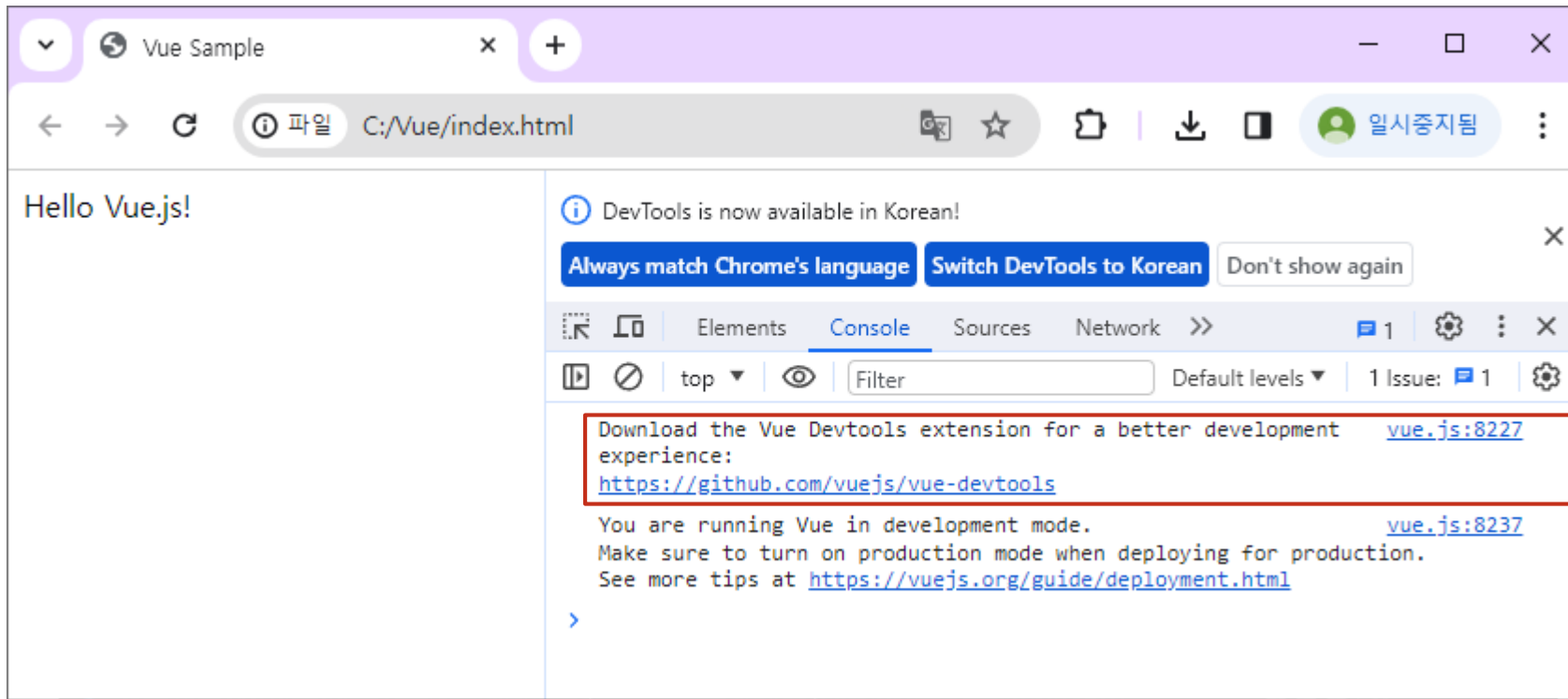
- ▶ Html 파일 생성 -> 뷰 소스 코드 추가 -> 브라우저로 실행
- VS 에서 index.html 파일 생성 (doc 입력후 Enter)



```
1 <html>
2   <head>
3     <title>Vue Sample</title>
4   </head>
5   <body>
6     <div id="app">
7       {{ message }}
8     </div>
9
10    <script src="https://cdn.jsdelivr.net/npm/vue@2.5.2/dist/vue.js"></script>
11    <script>
12      new Vue({
13        el: '#app',
14        data: {
15          message: 'Hello Vue.js!'
16        }
17      });
18    </script>
19  </body>
20 </html>
```



▶ 크롬 개발자 도구로 코드 확인하기 (F12 키)



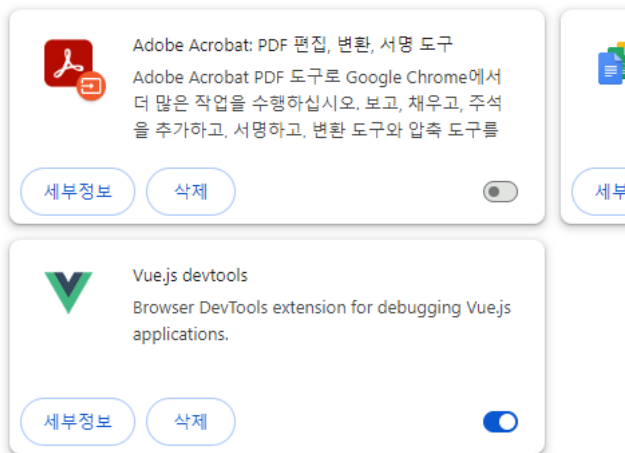
▶ 뷰 개발자 도구로 코드 확인하기

- 뷰 개발자 도구는 뷰 애플리케이션에서만 활용할 수 있으며, 컴포넌트로 구성된 애플리케이션의 구조를 한 눈에 확인할 수 있음
- 각 컴포넌트별로 정의된 속성의 변화를 실시간으로 확인할 수 있어 뷰로 제작한 웹 앱을 분석하거나 디버깅 할 때 유용하게 사용할 수 있음

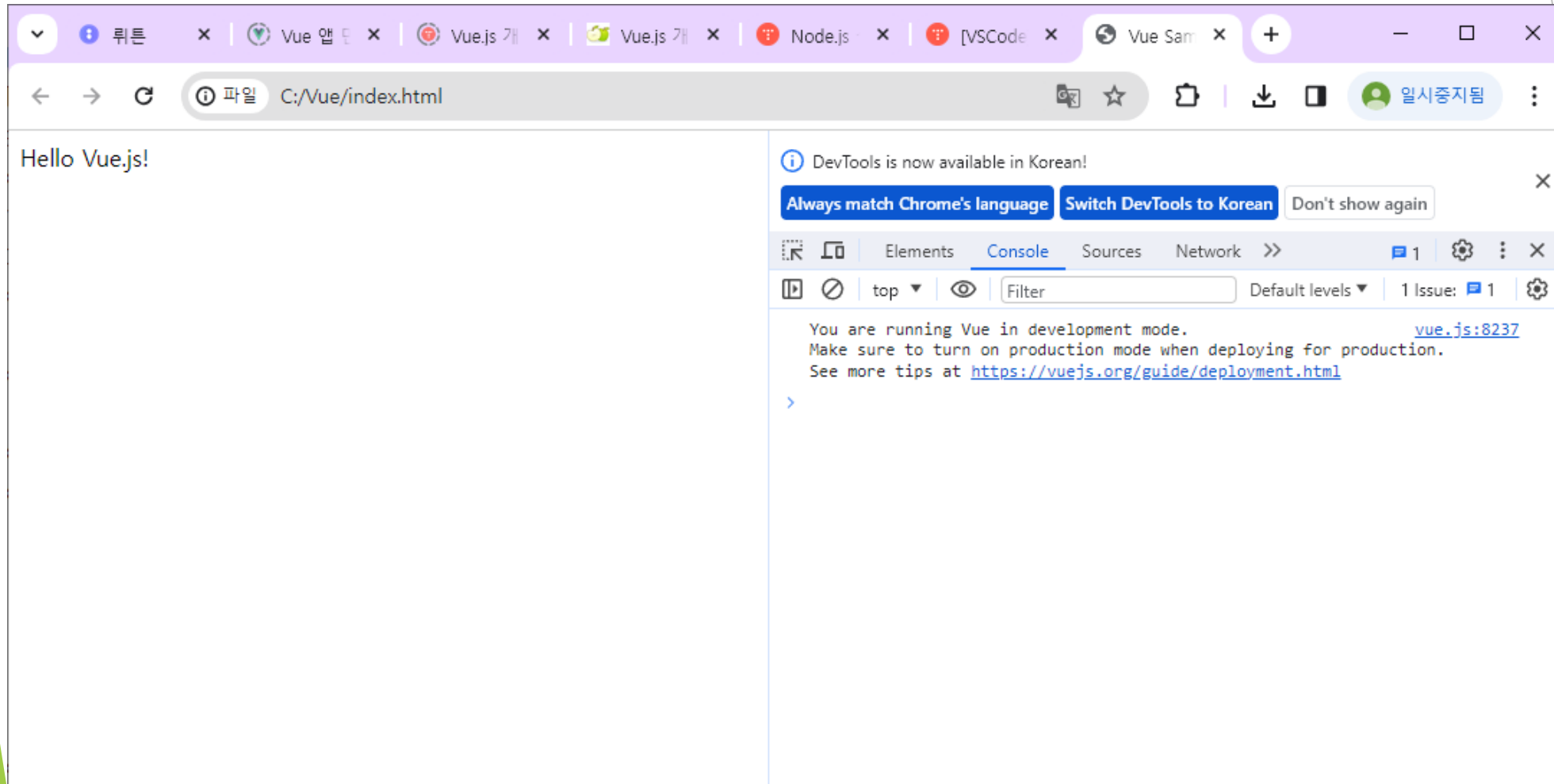
▶ 첫 번째 로그 해결 방법

- 현재 예제를 서버에서 띄운 것이 아니라 파일 시스템에서 접근하여 브라우저에 실행했기 때문에 경고 로그가 발생함
- 이 문제를 해결하기 위해서 크롬 확장 플러그인 설정을 변경해야 함

전체 확장 프로그램

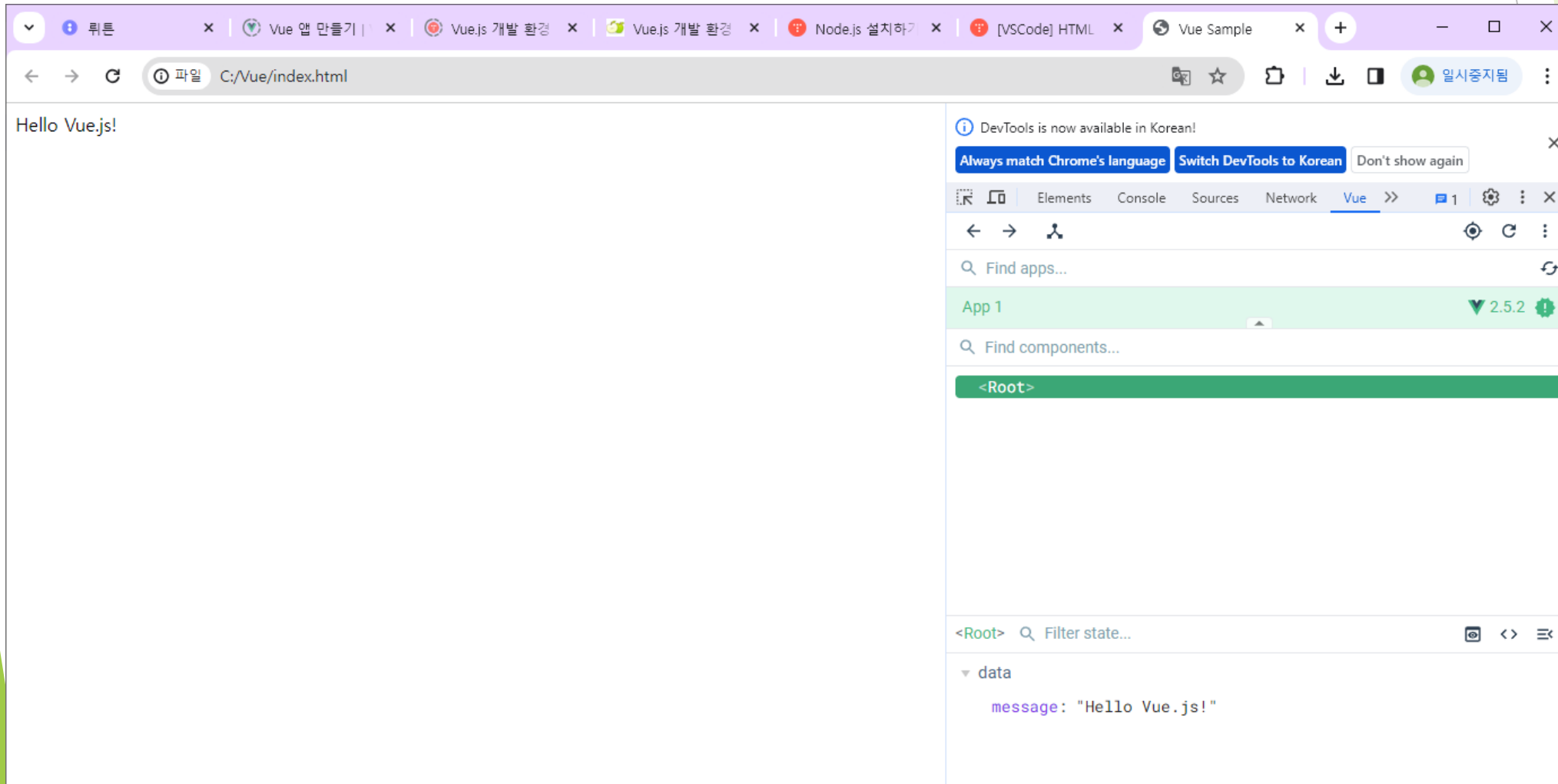


- ▶ 위 설정을 한 후 다시 페이지를 로딩하면 첫 번째 로그가 사라짐



뷰 개발자 도구 사용 방법

- ▶ 크롬 개발자 도구를 열고 'Vue' 탭을 선택
탭이 숨겨져 있을 경우 >>를 눌러 'Vue' 탭을 엽니다.

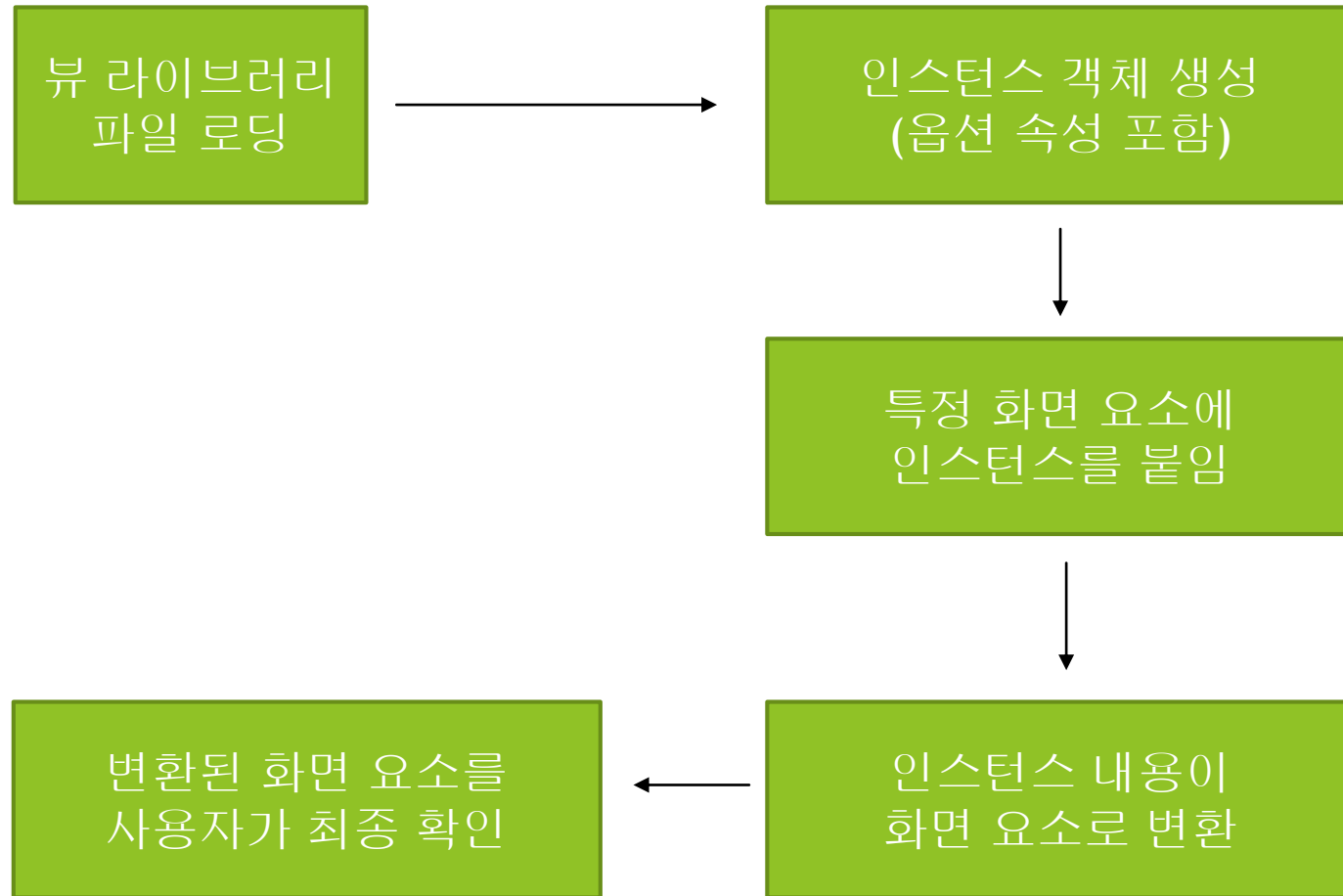


뷰 인스턴스

- ▶ 뷰 인스턴스는 뷰로 화면을 개발하기 위해 필수적으로 생성해야 하는 기본 단위
- ▶ 인스턴스는 뷰로 화면을 개발하기 위해 빠트릴 수 없는 필수 조건임
- ▶ 뷰 인스턴스 형식
new Vue({
 ...
})
- ▶ ‘Hello Vue.js!’ 텍스트를 화면에 표시하기 위해 new Vue()로 뷰 인스턴스를 생성
- ▶ 인스턴스 안에 el 속성으로 뷰 인스턴스가 그려질 지점을 지정
- ▶ data 속성에 message 값을 정의하여 화면의 {{message}}에 연결

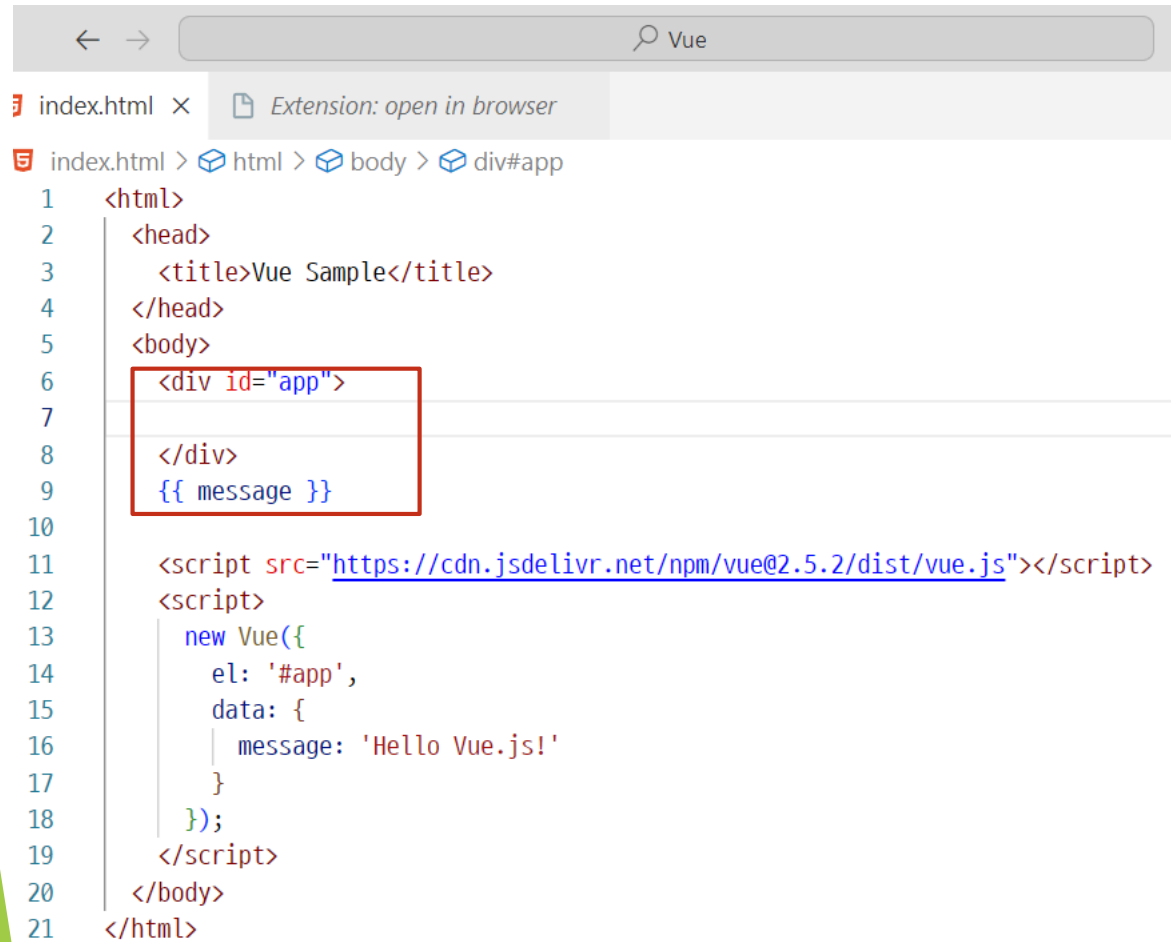
```
<html>  
  <head>  
    <title>Vue Sample</title>  
  </head>  
  <body>  
    <div id="app">  
      {{ message }}  
    </div>  
  
    <script src="https://cdn.jsdelivr.net/npm/vue@2.5.2/dist/vue.js"></script>  
    <script>  
      new Vue({  
        el: '#app',  
        data: {  
          message: 'Hello Vue.js!'  
        }  
      });  
    </script>  
  </body>  
</html>
```

인스턴스가 화면에 적용되는 과정

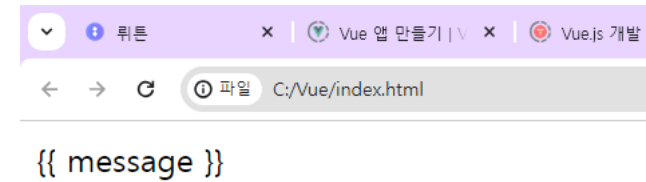


인스턴스의 유효 범위 확인

- ▶ 다음과 같이 코드를 변경해 보면



```
1 <html>
2   <head>
3     <title>Vue Sample</title>
4   </head>
5   <body>
6     <div id="app">
7
8     </div>
9     {{ message }}
10
11   <script src="https://cdn.jsdelivr.net/npm/vue@2.5.2/dist/vue.js"></script>
12   <script>
13     new Vue({
14       el: '#app',
15       data: {
16         message: 'Hello Vue.js!'
17       }
18     });
19   </script>
20 </body>
21 </html>
```



Vue 앱 만들기 | Vue.js 개발

파일 C:/Vue/index.html

{{ message }}

뷰 인스턴스 라이프 사이클

▶ beforeCreate

- 인스턴스가 생성되고 나서 가장 처음으로 실행되는 라이프 사이클
- 이 단계에서는 **data** 속성과 **methods** 속성이 아직 인스턴스에 정의되어 있지 않고, **DOM**과 같은 화면 요소에도 접근할 수 없음

▶ created

- **data** 속성과 **methods** 속성이 정의되기 때문에 **this.data** 또는 **this.fetchData()**와 같은 로직들을 이용하여 **data** 속성과 **methods** 속성에 정의된 값에 접근하여 로직을 실행할 수 있음
- 다만, 아직 인스턴스가 화면 요소에 부착되기 전이기 때문에 **template** 속성에 정의된 **DOM** 요소로 접근할 수 없음

▶ beforeMount

- **template** 속성에 지정한 마크업 속성을 **render()** 함수로 변환한 후 **el** 속성에 지정한 화면 요소에 인스턴스를 부착하기 전에 호출되는 단계
- **render()**는 자바스크립트로 화면의 **DOM**을 그리는 함수

▶ mounted

- **template** 속성에 정의한 화면 요소에 접근할 수 있어 화면 요소를 제어하는 로직을 수행하기 좋은 단계

▶ beforeUpdate

- **el** 속성에서 지정한 화면 요소에 인스턴스가 부착되고 나면 인스턴스에 정의한 속성들이 화면에 치환됨
- 치환된 값은 뷰의 반응성을 제공하기 위해 **\$watch** 속성으로 감시
- 이를 데이터 관찰이라고 함

뷰 인스턴스 라이프 사이클

▶ update

- 데이터가 변경되고 나서 가상 돔으로 다시 화면을 그리고 나면 실행되는 단계
- 이 단계에서 데이터 값을 변경하면 무한 루프에 빠질 수 있기 때문에 값을 변경하려면 **computed**, **watch**와 같은 속성을 사용해야 함

▶ beforeDestory

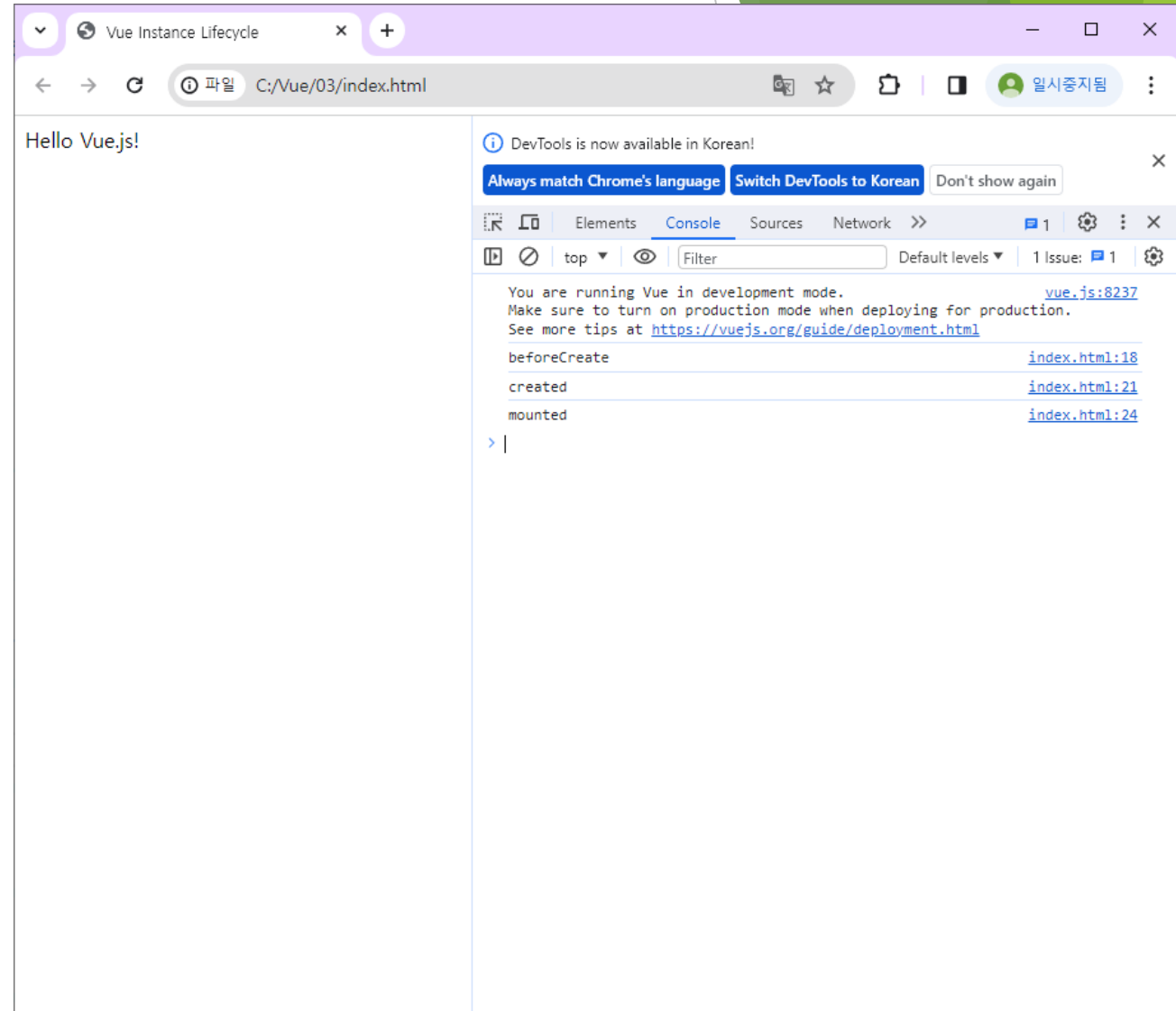
- 뷰 인스턴스가 파괴되기 직전에 호출되는 단계
- 뷰 인스턴스의 데이터를 삭제하기 좋은 단계

▶ destroyed

- 뷰 인스턴스가 파괴되고 나서 호출되는 단계
- 뷰 인스턴스에 정의한 모든 속성이 제거되고 하위에 선언한 인스턴스들 또한 모두 파괴됨

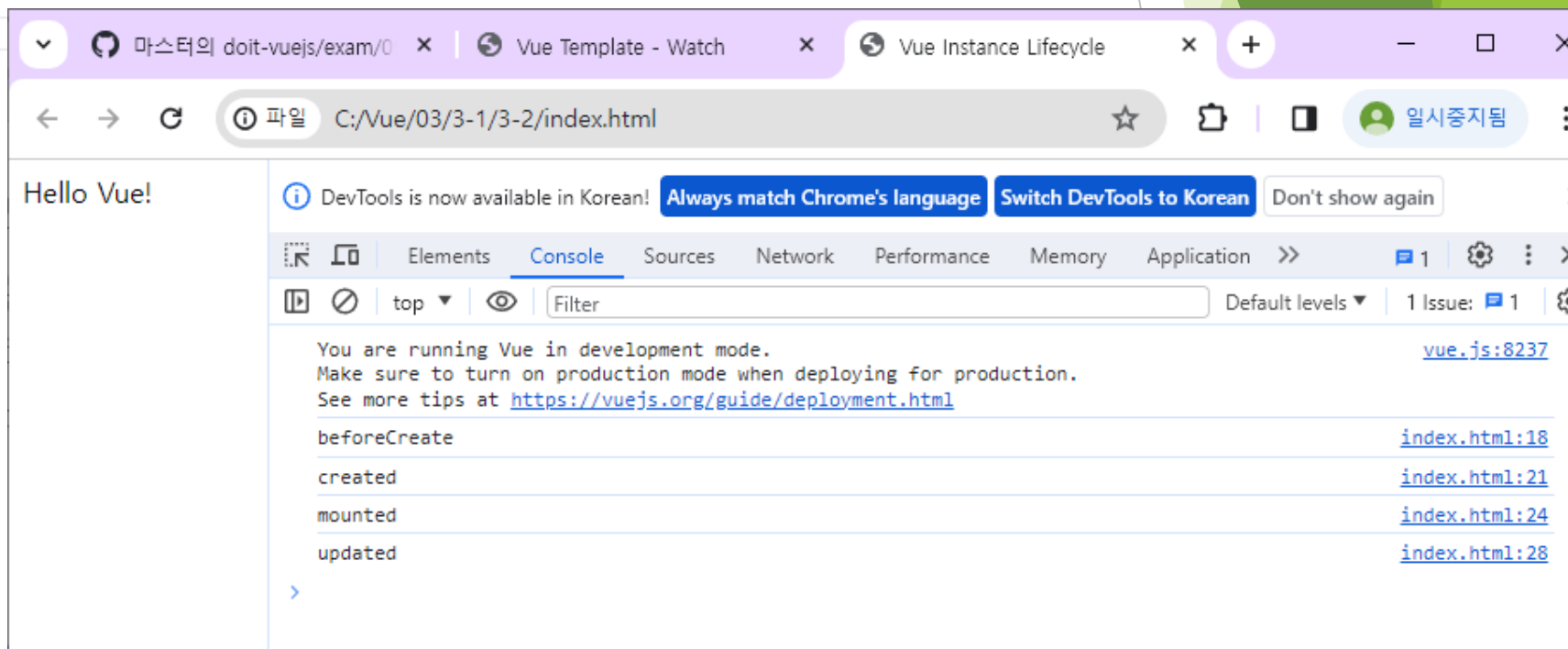
라이프 사이클 예제

```
1 <html>
2   <head>
3     <title>Vue Instance Lifecycle</title>
4   </head>
5   <body>
6     <div id="app">
7       {{ message }}
8     </div>
9
10    <script src="https://cdn.jsdelivr.net/npm/vue@2.5.2/dist/vue.js"></script>
11    <script>
12      new Vue({
13        el: '#app',
14        data: {
15          message: 'Hello Vue.js!'
16        },
17        beforeCreate: function() {
18          console.log("beforeCreate");
19        },
20        created: function() {
21          console.log("created");
22        },
23        mounted: function() {
24          console.log("mounted");
25        },
26        updated: function() {
27          console.log("updated");
28        }
29      });
30    </script>
31  </body>
32 </html>
```



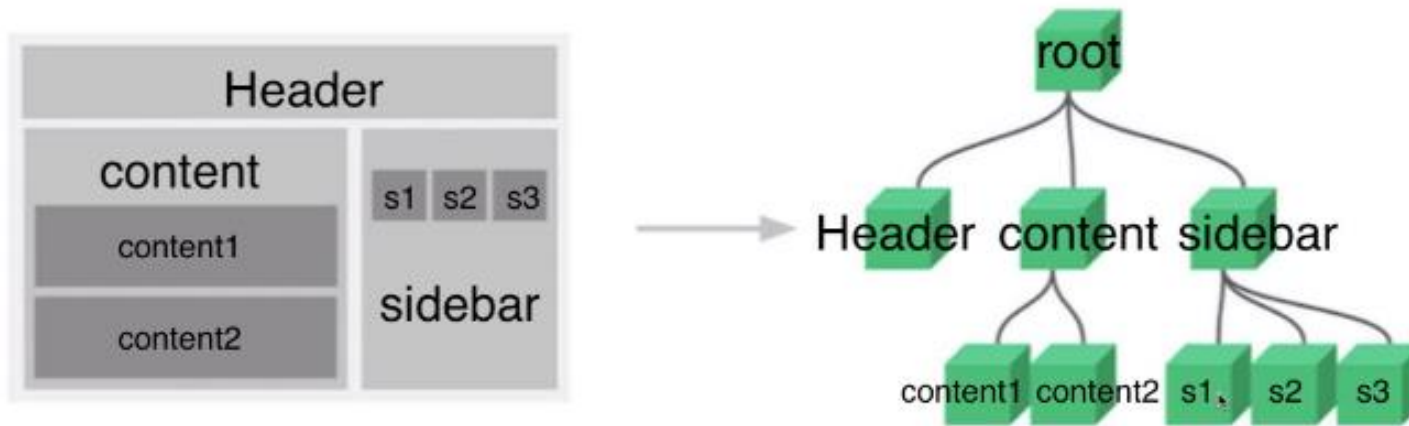
- ▶ 로그를 보면 **updated** 속성 함수는 호출 되지 않음
- ▶ 그 이유는 **updated** 라이프 사이클 혹은 뷰 인스턴스에서 데이터 변경이 일어나 화면이 다시 그려질 때 호출되는 로직이기 때문
- ▶ 다음과 같이 **mounted** 단계에서 기존에 정의된 **data** 속성의 **message** 값을 변경해 봄

```
10 <script src="https://cdn.jsdelivr.net/npm/vue@2.5.2/dist/vue.js"></script>
11 <script>
12   new Vue({
13     el: '#app',
14     data: {
15       message: 'Hello Vue.js!'
16     },
17     beforeCreate: function() {
18       console.log("beforeCreate");
19     },
20     created: function() {
21       console.log("created");
22     },
23     mounted: function() {
24       console.log("mounted");
25       this.message = 'Hello Vue!';
26     },
27     updated: function() {
28       console.log("updated");
29     }
30   });
31 </script>
32 </body>
```



뷰 컴포넌트

- ▶ 컴포넌트란?
조합하여 화면을 구성할 수 있는 블록(화면의 특정 영역)을 의미함
- ▶ 컴포넌트를 활용하면 화면을 빠르게 구조화 하여 일괄적인 패턴으로 개발할 수 있음
- ▶ 이렇게 화면의 영역을 컴포넌트로 쪼개서 재활용할 수 있는 형태로 관리하면 나중에 코드를 다시 사용하기가 훨씬 편리해 짐
- ▶ 또한 모든 사람들이 정해진 방식대로 컴포넌트를 등록하거나 사용하게 되므로 남이 작성한 코드를 직관적으로 이해할 수 있음
- ▶ 뷰에서는 웹 화면을 구성할 때 흔히 사용하는 네비게이션 바, 테이블, 리스트, 인풋 박스 등과 같은 화면 구성 요소들을 잘게 쪼개어 컴포넌트로 관리함



컴포넌트 등록하기

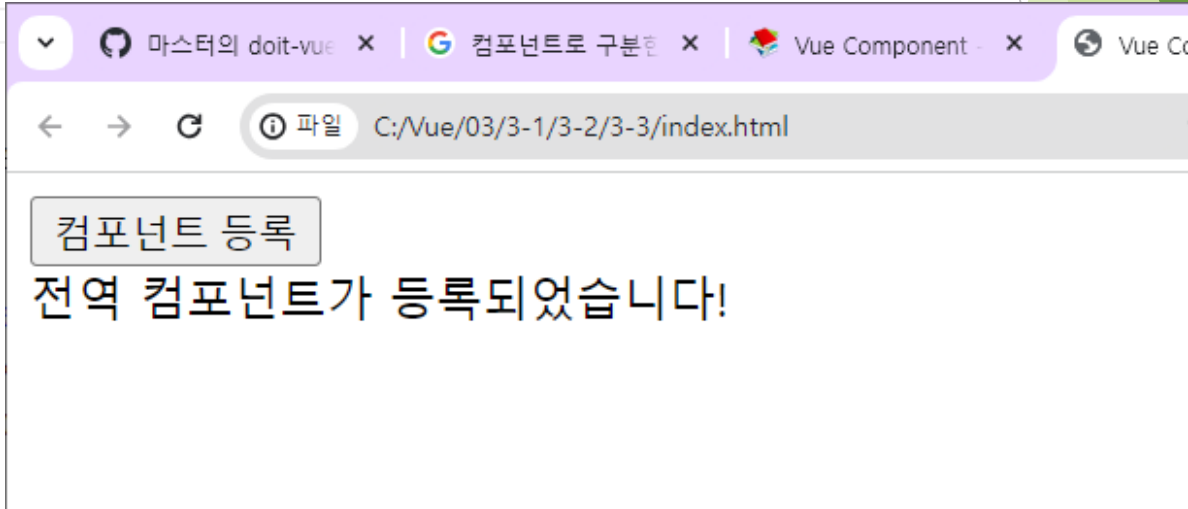
- ▶ 컴포넌트를 등록하는 방법은 전역과 지역 두 가지가 있음
- ▶ 지역(Local) 컴포넌트는 특정 인스턴스에서만 유효한 범위를 갖고, 전역(Global) 컴포넌트는 여러 인스턴스에서 공통으로 사용할 수 있음
- ▶ 즉, 지역은 특정 범위내에서만 사용할 수 있고, 전역은 뷰로 접근 가능한 모든 범위에서 사용 가능함
- ▶ 전역 컴포넌트 등록

```
Vue.component('컴포넌트 이름',{  
  // 컴포넌트 내용  
});
```

- ▶ 컴포넌트 내용에는 **template, data, methods** 등 인스턴스 옵션 속성을 정의할 수 있음

전역 컴포넌트 등록하기

```
3-2 > 3-3 > index.html > html > body > script
1 <html>
2 <head>
3 | <title>Vue Component Registration</title>
4 </head>
5 <body>
6 | <div id="app">
7 | | <button>컴포넌트 등록</button>
8 | | <my-component></my-component>
9 | </div>
10
11 <script src="https://cdn.jsdelivr.net/npm/vue@2.5.2/dist/vue.js"></script>
12 <script>
13 |   Vue (property) template: string
14 |   template: '<div>전역 컴포넌트가 등록되었습니다!</div>'
15 | ];
16
17   new Vue({
18 |     el: '#app'
19 |   });
20 </script>
21 </body>
22 </html>
```

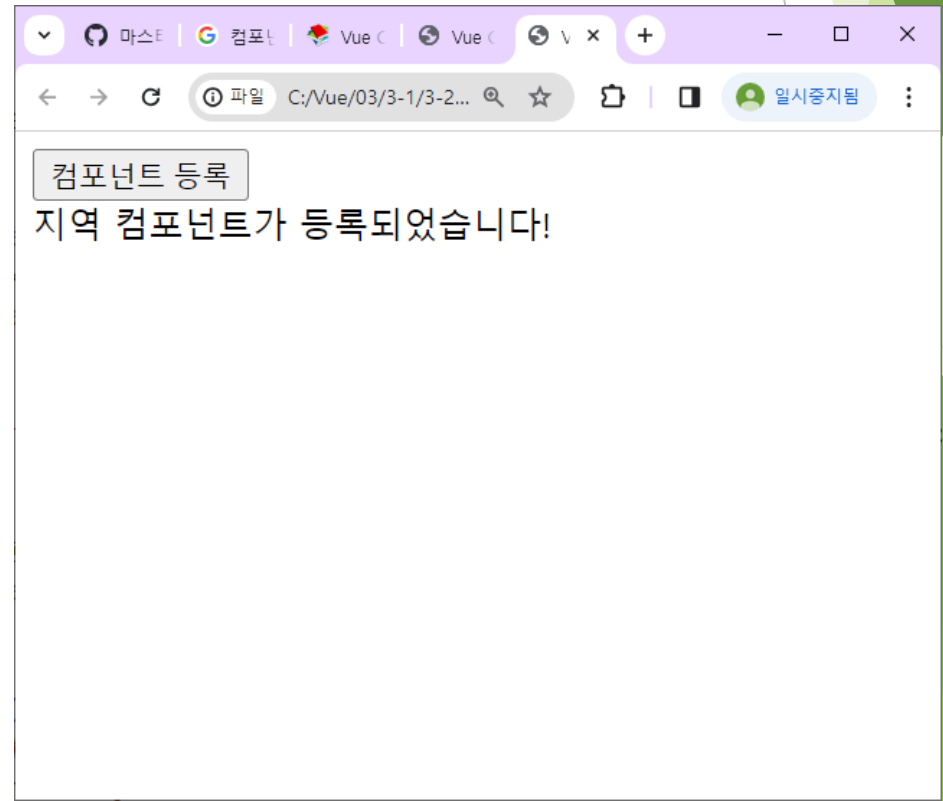


지역 컴포넌트 등록

- ▶ 지역 컴포넌트 등록은 전역 컴포넌트 등록과는 다르게 인스턴스에 **components** 속성을 추가하고 등록할 컴포넌트 이름과 내용을 정의하면 됨

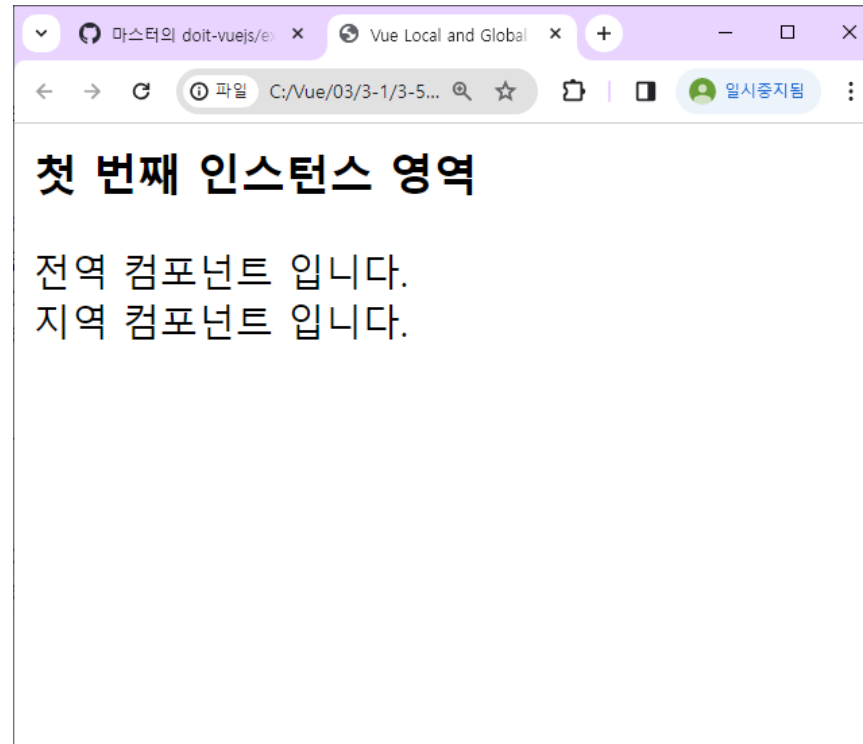
```
1 <html>
2   <head>
3     <title>Vue Component Registration</title>
4   </head>
5   <body>
6     <div id="app">
7       <button>컴포넌트 등록</button>
8       <my-local-component></my-local-component>
9     </div>
10
11     <script src="https://cdn.jsdelivr.net/npm/vue@2.5.2/dist/vue.js"></script>
12     <script>
13       var cmp = {
14         // 컴포넌트 내용
15         template: '<div>지역 컴포넌트가 등록되었습니다!</div>'
16       };
17
18       new Vue({
19         el: '#app',
20         components: {
21           'my-local-component': cmp
22         }
23       });
24     </script>
25   </body>
26 </html>
```

```
new Vue({
  components: {
    '컴포넌트 이름': '컴포넌트 내용'
  }
});
```



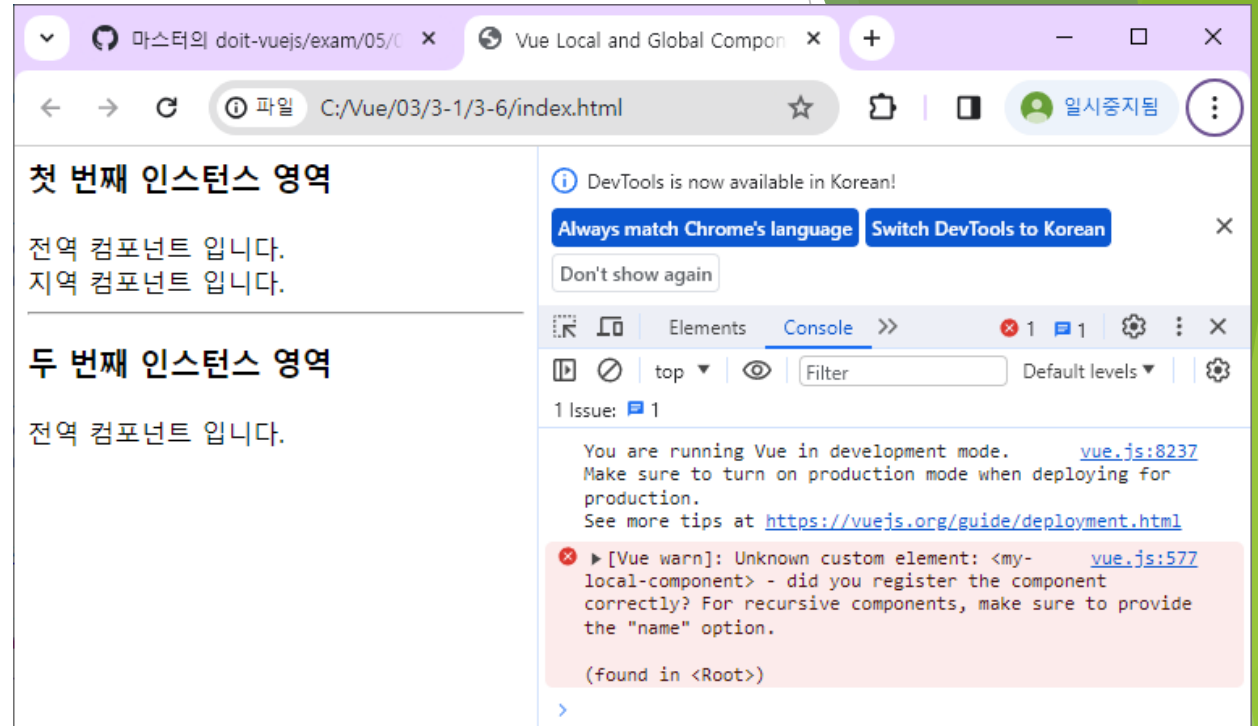
지역 컴포넌트와 전역 컴포넌트 차이점

```
1 <html>
2 <head>
3   <title>Vue Local and Global Components</title>
4 </head>
5 <body>
6   <div id="app">
7     <h3>첫 번째 인스턴스 영역</h3>
8     <my-global-component></my-global-component>
9     <my-local-component></my-local-component>
10  </div>
11
12  <script src="https://cdn.jsdelivr.net/npm/vue@2.5.2/dist/vue.js"></script>
13  <script>
14    // 전역 컴포넌트 등록
15    Vue.component('my-global-component', {
16      template: '<div>전역 컴포넌트 입니다.</div>'
17    });
18
19    // 지역 컴포넌트 내용
20    var cmp = {
21      template: '<div>지역 컴포넌트 입니다.</div>'
22    };
23
24    new Vue({
25      el: '#app',
26      // 지역 컴포넌트 등록
27      components: {
28        'my-local-component': cmp
29      }
30    });
31  </script>
32 </body>
```



인스턴스를 하나 더 생성하고 해당 인스턴스에 지역, 전역 컴포넌트를 모두 표시

```
1 <html>
2 <head>
3   <title>Vue Local and Global Components</title>
4 </head>
5 <body>
6   <div id="app">
7     <h3>첫 번째 인스턴스 영역</h3>
8     <my-global-component></my-global-component>
9     <my-local-component></my-local-component>
10  </div>
11  <hr>
12  <div id="app2">
13    <h3>두 번째 인스턴스 영역</h3>
14    <my-global-component></my-global-component>
15    <my-local-component></my-local-component>
16  </div>
17
18  <script src="https://cdn.jsdelivr.net/npm/vue@2.5.2/dist/vue.js"></script>
19  <script>
20    // 전역 컴포넌트 등록
21    Vue.component('my-global-component', {
22      template: '<div>전역 컴포넌트 입니다.</div>'
23    });
24
25    // 지역 컴포넌트 내용
26    var cmp = {
27      template: '<div>지역 컴포넌트 입니다.</div>'
28    };
29
30    new Vue({
31      el: '#app',
32      // 지역 컴포넌트 등록
33      components: {
34        'my-local-component': cmp
35      }
36    });
37
38    // 두 번째 인스턴스
39    new Vue({
40      el: '#app2'
41    });
42  </script>
43 </body>
44 </html>
```



전역 컴포넌트는 인스턴스를 생성할 때마다 인스턴스에 **components** 속성으로 등록할 필요없이 한 번 등록하면 어느 인스턴스에든지 사용할 수 있음

반대로 지역 컴포넌트는 새 인스턴스를 생성할 때마다 등록해 줘야 함

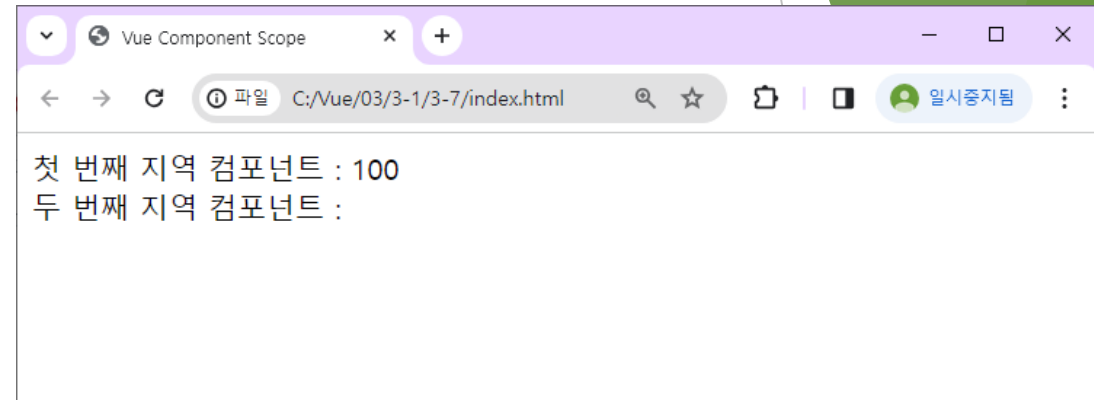
뷰 컴포넌트 통신

- ▶ 앵귤러1 이나 백본과 같은 초창기 자바스크립트 프레임워크에서는 한 화면을 1개의 뷰(View)로 간주했기 때문에 한 화면의 데이터를 해당 화면 영역 어디서든지 호출할 수 있었음
- ▶ 하지만 뷰(Vue.js)의 경우 컴포넌트로 화면을 구성하기 때문에 같은 웹 페이지라도 데이터를 공유할 수 없음
- ▶ 그 이유는 컴포넌트마다 자체적으로 고유한 유효 범위(Scope)를 갖기 때문
- ▶ 이는 뷰 프레임워크 내부적으로 정의된 특징
- ▶ 따라서 각 컴포넌트의 유효 범위가 독자적이기 때문에 다른 컴포넌트의 값을 직접적으로 참조할 수 없음

컴포넌트 유효 범위 증명 예제

3-7 > index.html > html > body > script > cmp1

```
1 <html>
2 <head>
3   <title>Vue Component Scope</title>
4 </head>
5 <body>
6   <div id="app">
7     <my-component1></my-component1>
8     <my-component2></my-component2>
9   </div>
10
11   <script src="https://cdn.jsdelivr.net/npm/vue@2.5.2/dist/vue.js"></script>
12   <script>
13     // 첫 번째 컴포넌트 내용
14     var cmp1 = {
15       template: '<div>첫 번째 지역 컴포넌트 : {{ cmp1Data }}</div>',
16       data: function() {
17         return {
18           cmp1Data : 100
19         }
20       }
21     };
22
23     // 두 번째 컴포넌트 내용
24     var cmp2 = {
25       template: '<div>두 번째 지역 컴포넌트 : {{ cmp2Data }}</div>',
26       data: function() {
27         return {
28           cmp2Data : cmp1.data.cmp1Data
29         }
30       }
31     };
32
33     new Vue({
34       el: '#app',
35       // 지역 컴포넌트 등록
36       components: {
37         'my-component1': cmp1,
38         'my-component2': cmp2
39       }
40     });
41   </script>
42 </body>
43 </html>
```



이렇게 다른 컴포넌트의 값을 참조하지 못하게 함으로써
뷰에서 미리 정의해 놓은 데이터 전달 방식에 따라 일관된
구조로 애플리케이션을 작성하게 되므로 개발자 개인의
스타일대로 구성되지 않고, 애플리케이션이 모두 동일한
데이터 흐름을 갖게 되므로 다른 사람의 코드를 빠르게
파악할 수 있어 협업하기에는 좋음

상/하 컴포넌트 관계

- ▶ 앞에서 살펴본 것처럼 컴포넌트는 각각 고유한 유효 범위를 갖고 있기 때문에 직접 다른 컴포넌트의 값을 참조할 수 없음
- ▶ 따라서 뷰 프레임워크 자체에서 정의한 컴포넌트 데이터 전달 방법을 따라야 함
- ▶ 가장 기본적인 데이터 전달 방법은 바로 상위(부모) -> 하위(자식) 컴포넌트 간의 데이터 전달 방식임
- ▶ 상위에서 하위로는 **props**라는 특별한 속성을 전달함
- ▶ 하위에서 상위로는 기본적으로 이벤트만 전달 함

상위에서 하위 컴포넌트로 데이터 전달하기

- ▶ **props**는 상위 컴포넌트에서 하위 컴포넌트로 데이터를 전달할 때 사용하는 속성

```
Vue.component('child-component', {  
  props: ['props 속성 이름'],  
});
```

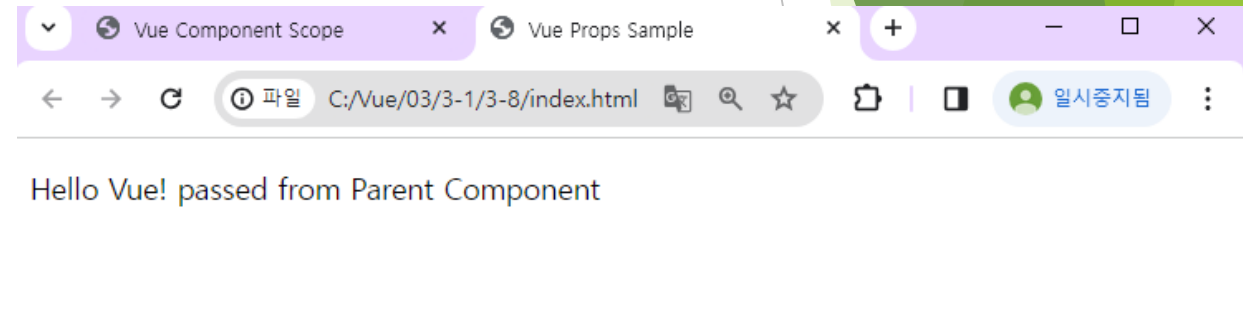
- ▶ 그런 다음 상위 컴포넌트의 **HTML** 코드에 등록된 **child-component** 태그에 **v-bind** 속성을 추가함

```
<child-component v-bind:props 속성이름="상위 컴포넌트의 data 속성"> </child-component>
```

- ▶ **v-bind** 속성의 왼쪽 값으로 하위 컴포넌트에서 정의한 **props** 속성을 넣고, 오른쪽 값으로 하위 컴포넌트에 전달할 상위 컴포넌트의 **data** 속성을 지정함

props 속성을 사용한 데이터 전달 예제

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Vue Props Sample</title>
7   </head>
8   <body>
9     <div id="app">
10      <!-- 팁 : 오른쪽에서 왼쪽으로 속성을 읽으면 더 수월합니다. -->
11      <child-component v-bind:propsdata="message"></child-component>
12    </div>
13
14    <script src="https://cdn.jsdelivr.net/npm/vue@2.5.2/dist/vue.js"></script>
15    <script>
16      Vue.component('child-component', {
17        props: ['propsdata'],
18        template: '<p>{{ propsdata }}</p>',
19      });
20
21      new Vue({
22        el: '#app',
23        data: {
24          message: 'Hello Vue! passed from Parent Component'
25        }
26      });
27    </script>
28  </body>
29 </html>
```



이렇게 인스턴스에 새로운 컴포넌트를 등록하면 컴포넌트는 상위 컴포넌트(부모)가 되고, 새로 등록된 컴포넌트는 하위(자식) 컴포넌트가 됨

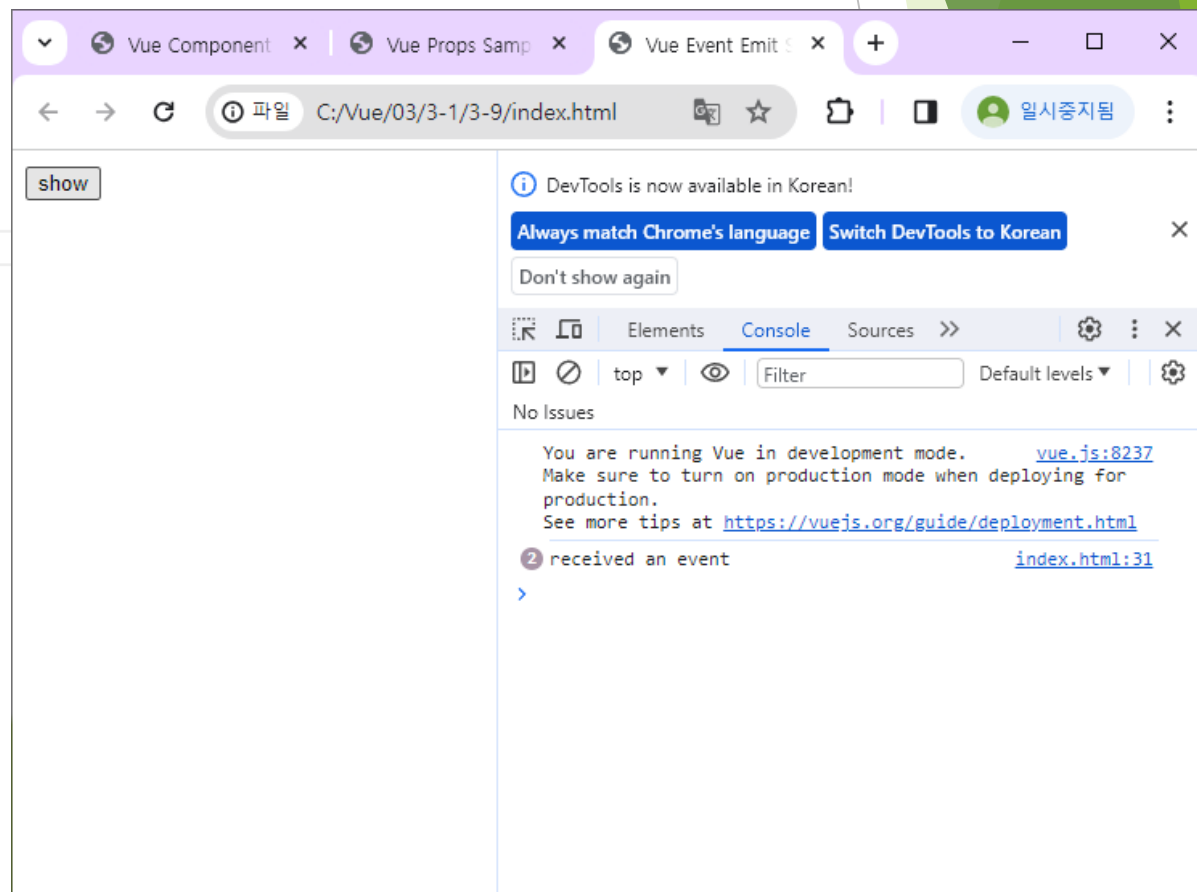
그리고 이렇게 새 컴포넌트를 등록한 인스턴스를 최상위 컴포넌트(Root Component)라고 부름

하위에서 상위 컴포넌트로 이벤트 전달하기

- ▶ 하위에서 이벤트를 발생시켜(event emit) 상위 컴포넌트에 신호를 보냄
- ▶ 상위 컴포넌트에서는 하위 컴포넌트의 특정 이벤트가 발생하기를 기다리고 있다가 하위 컴포넌트에서 특정 이벤트가 발생하면 상위 컴포넌트에서 해당 이벤트를 수신하여 상위 컴포넌트의 메서드를 호출함
- ▶ 이벤트 발생
`this.$emit('이벤트명');`
- ▶ 이벤트 수신
`<child-component v-on:이벤트명="상위 컴포넌트의 메서드 명"> </child-component>`

이벤트를 발생하고 수신하는 예제

```
8 <body>
9   <div id="app">
10     <child-component v-on:show-log="printText"></child-component>
11   </div>
12
13   <script src="https://cdn.jsdelivr.net/npm/vue@2.5.2/dist/vue.js"></script>
14   <script>
15     Vue.component('child-component', {
16       template: '<button v-on:click="showLog">show</button>',
17       methods: {
18         showLog: function() {
19           this.$emit('show-log');
20         }
21       }
22     });
23
24     new Vue({
25       el: '#app',
26       data: {
27         message: 'Hello Vue! passed from Parent Component'
28       },
29       methods: {
30         printText: function() {
31           console.log("received an event");
32         }
33       }
34     });
35   </script>
36 </body>
37 </html>
```



같은 레벨의 컴포넌트 간 통신

- ▶ 상-하위 관계가 아닌 같은 레벨에 있는 컴포넌트끼리는 이벤트 버스를 이용함
- ▶ 이벤트 버스(Event Bus)는 개발자가 지정한 2개의 컴포넌트 간에 데이터를 주고 받을 수 있는 방법
- ▶ 이벤트 버스를 이용하면 상위-하위 관계를 유지하고 있지 않아도 데이터를 한 컴포넌트에서 다른 컴포넌트로 전달할 수 있음
- ▶ 이벤트 버스 형식

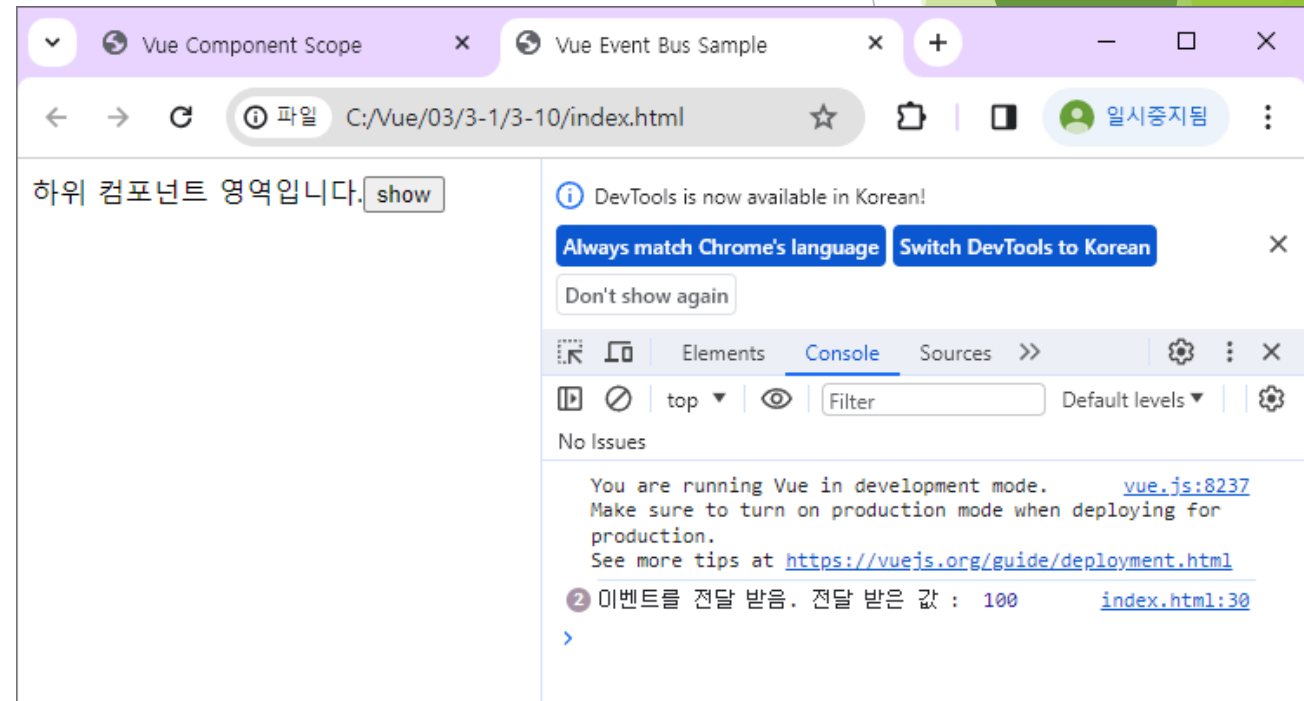
```
// 이벤트 버스를 위한 추가 인스턴스 1개 생성  
var eventBus = new Vue();
```

```
// 이벤트를 보내는 컴포넌트  
methods: {  
  메서드명: function() {  
    eventBus.$emit('이벤트명', 데이터);  
  }  
}
```

```
// 이벤트를 받는 컴포넌트  
methods: {  
  created: function() {  
    eventBus.$on('이벤트명', function(데이터) {  
      ...  
    });  
  }  
}
```

이벤트 버스 구현 예제

```
8 <body>
9   <div id="app">
10     <child-component></child-component>
11   </div>
12
13   <script src="https://cdn.jsdelivr.net/npm/vue@2.5.2/dist/vue.js"></script>
14   <script>
15     var eventBus = new Vue();
16
17     Vue.component('child-component', {
18       template: '<div>하위 컴포넌트 영역입니다.<button v-on:click="showLog">show</button></div>',
19       methods: {
20         showLog: function() {
21           eventBus.$emit('triggerEventBus', 100);
22         }
23       }
24     });
25
26     var app = new Vue({
27       el: '#app',
28       created: function() {
29         eventBus.$on('triggerEventBus', function(value){
30           console.log("이벤트를 전달 받음. 전달 받은 값 : ", value);
31         });
32       }
33     });
34   </script>
35 </body>
36 </html>
```



- ▶ 이벤트 버스를 활용하면 **props** 속성을 이용하지 않고도 원하는 컴포넌트 간에 직접적으로 데이터를 전달할 수 있어 편리하지만 컴포넌트가 많아지면 어디서 어디로 보냈는지 관리가 되지 않는 문제가 발생
- ▶ 이 문제를 해결하려면 뷰엑스(Vuex)라는 상태 관리 도구가 필요
- ▶ 뷰엑스는 중/대형 애플리케이션에서 컴포넌트 간의 데이터 관리를 효율적으로 하는 라이브 러