Vue.js 시작하기

2022-03-31

이승진

**학습목표**

Vue.js 시작하기

v-model 지시자

텍스트 입력 기능

목록 선택 기능

체크박스

라디오 버튼

**목차**

[1. Vue.js 소개 2](#_Toc68801678)

[1) Frontend Framework 2](#_Toc68801679)

[2) React.js vs Vue.js 2](#_Toc68801680)

[2. hello 프로젝트 생성 3](#_Toc68801681)

[1) 프로젝트 디렉토리 생성 3](#_Toc68801682)

[2) 프로젝트 설정 파일(package.json) 생성 3](#_Toc68801683)

[3) eslint 설정 파일 생성 4](#_Toc68801684)

[3. hello world 6](#_Toc68801685)

[1) hello.html 생성 6](#_Toc68801686)

[2) 실행 7](#_Toc68801687)

[3) hello.html 수정 10](#_Toc68801688)

[4. v-model 지시자 11](#_Toc68801689)

[1) hello1.html 11](#_Toc68801690)

[2) hello2.html 13](#_Toc68801691)

[3) checkbox1.html 15](#_Toc68801692)

[4) checkbox2.html 17](#_Toc68801693)

[5) select1.html 18](#_Toc68801694)

[6) select2.html 19](#_Toc68801695)

[7) radio1.html 20](#_Toc68801696)

[5. 요약 22](#_Toc68801697)

[6. 연습 문제 23](#_Toc68801698)

[1) form1.html 23](#_Toc68801699)

# Vue.js 소개

## Frontend Framework

### Application Framework Class Library

앱 개발에 필요한 골격에 해당하는 클래스들을 모아 놓은 라이브러리.

줄여서 Application Framework 이라고 부른다.

### Javascript Frontend Application Framework

javascript 언어로 frontend app을 개발하기 위한 라이브러리.

대표적인 프레임웍은 다음과 같다.

React.js, Vue.js

### Backend Framework

정보시스템의 경우 Java 언어 + Spring Framework 기술이 대세.

## React.js vs Vue.js

### React.js

가장 널리 사용되는 프론트엔드 개발 기술.

프론트엔드 개발자라면 React.js를 공부해야 한다.

React.js는 본격적인 프론트엔드 프로젝트를 개발하는 것만 가능하고,

웹서버 프로그래밍을 보조하는 Javascript 구현 기술로 간단히 사용하기 어렵다.

Spring Web MVC 프로젝트에 간단히 프론트엔드 기술을 적용할 때 React.js는 적절하지 않다.

### Vue.js

Vue.js는 본격적인 프론트엔드 프로젝트를 개발하는 것도 가능하지만,

웹서버 프로그래밍을 보조하는 Javascript 구현 기술로 간단히 사용할 수도 있다.

Spring Web MVC 프로젝트에 간단히 프론트엔드 기술을 적용할 때 사용할 수도 있다.

따라서 백엔드 개발자가 프론트엔드 개발 기술도 활용하고 싶을 때, Vue.js가 좋다.

|  |
| --- |
| **주의**  git bash 쉘에서 작업할 것이라면 꼭 Visual Studio Code 터미널에서 git bash 쉘을 열어야 한다.  Visual Studio Code 터미널이 아닌, git bash 창에서 명령을 실행하면,  화살표 키가 잘 작동하지 않는다. |

# hello 프로젝트 생성

Vue.js 예제들을 구현하고 실행해 보기 위한 간단한 프로젝트를 생성하자.

|  |
| --- |
| 참고:  **'02 Vue 기초'**에서 설명하고 있는 간단한 프로젝트 생성 및 구현 방법은  본격적인 프론트엔드 앱 개발을 위한 것이 아니다.  프론트엔드 앱 개발을 위한 프로젝트 생성 및 구현 방법은 **'03 Vue 중급'**에서 설명한다. |

## 프로젝트 디렉토리 생성

프로젝트를 만들기에 적당한 디렉토리로 이동하자.

|  |
| --- |
| cd /c/PJ/frontend |

적당한 이름의 프로젝트 디렉토리를 만들자.

|  |
| --- |
| mkdir hello |

|  |
| --- |
|  |

## 프로젝트 설정 파일(package.json) 생성

프로젝트 디렉토리로 이동

|  |
| --- |
| cd hello |

javascript 프로젝트(node.js 프로젝트)의 설정 파일은 package.json 파일이다.

아래 명령을 실행하여 이 파일을 생성하자.

|  |
| --- |
| npm init -yes |

주의 사항: 프로젝트 폴더에서 이 명령을 실행해야 한다.

|  |
| --- |
|  |

## eslint 설정 파일 생성

프로젝트 폴더 에 eslint 설정 파일을 추가하자.

eslint 설정 파일은 .eslintrc 이다.

### 프로젝트에 eslint 설정 파일(.eslintrc) 추가 명령

주의: 프로젝트 폴더에서 아래 명령을 실행해야 한다.

|  |
| --- |
| npx eslint --init |

|  |
| --- |
|  |

위 아래 화살표 키를 눌러서 'To check syntax and find problems' 항목을 선택하고

엔터키 입력

check syntax : 문법 오류를 검사한다

find problems : 흔한 실수를 검사한다

enforce code style: 코드 스타일을 검사한다

|  |
| --- |
|  |

Javascript modules (import/export) 항목을 선택하고 엔터키 입력

|  |
| --- |
|  |

Vue.js 항목을 선택하고 엔터키 입력

|  |
| --- |
|  |

No 항목을 선택하고 엔터키 입력

|  |
| --- |
|  |

Browser 항목을 선택하고 엔터키 입력

Browser : 이 프로젝트의 코드가 웹브라우저에서 실행될 것이라면 선택

Node : 이 프로젝트의 코드가 node.js에서 실행될 것이라면 선택

|  |
| --- |
|  |

JSON 항목을 선택하고 엔터

(생성될 eslint 설정 파일의 포맷을 지정한다)

|  |
| --- |
|  |

Yes 항목을 선택하고 엔터키 입력

위 명령에 의해서, eslint의 설정 파일인 .eslintrc.json 파일 생성된다.

# hello world

## hello.html 생성

### hello.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>  </head>  <body>    <div id="app">      <h1>{{ title }}</h1>      <p>{{ body }}</p>    </div>    <script type="text/javascript">      let app = {        data() {          return {            title: 'hello world',            body: 'good morning'          };        }      };      Vue.createApp(app).mount("#app");     </script>  </body>  </html> |

### Vue CDN

<script src="https://unpkg.com/vue@next"></script>

vue.js javascript 라이브러리 파일에 대한 링크이다.

이 파일의 라이브러리 가 웹브라우저 창에서 실행된다.

즉 vue.js 가 이 창에서 실행된다.

위와 같이 javascript 라이브러리 파일들을 웹페이지에 링크해서 사용할 수 있도록

웹서버(https://unpkg.com)를 제공해 주는 것을

CDN (Contents Delivery Network) 서비스라고 한다.

### Vue 앱

let app = { 생략 }

Vue.createApp(app).mount("#app");

위 코드는 Vue 앱 객체를 생성하여 실행을 시작한다.

노란색으로 칠한 부분에, Vue 앱의 기능을 구현해야 한다.

노란색 부분에 Vue 앱 객체의 속성과 메소드를 구현한다.

.mount("#app")

Vue 앱의 기능이 실행될 태그의 id를 지정한다.

Vue 앱의 기능은 <div id="app"> 태그 내부에서만 실행된다.

data() {

return {

       title: 'hello world',

       body: 'good morning'

    };

}

앱 객체의 속성을 선언하는 코드이다. (앱 객체의 멤버 변수 선언)

<h1>{{ title }}</h1>

title 속성값이 여기에 출력된다.

<p>{{ body }}</p>

body 속성값이 여기에 출력된다.

## 실행

### live-server 설치

live-server는 개발 환경에서 프로젝트를 실행하기 위한 웹 서버이다.

아래 명령을 실행하여 live-server를 설치하자.

|  |
| --- |
| npm install -g live-server |

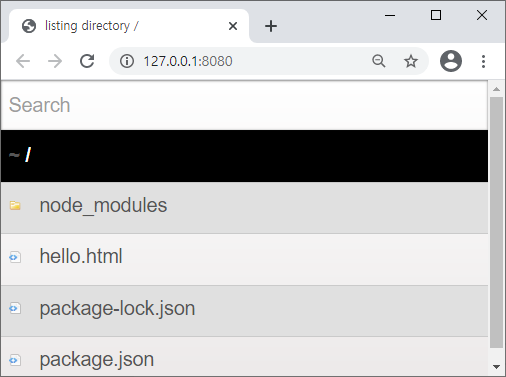
|  |
| --- |
|  |

### live-server 실행

아래 명령을 실행하면 live-server 가 실행되고, 웹브라우저도 자동으로 실행된다.

|  |
| --- |
| live-server |

|  |
| --- |
|  |



### live-server 포트 번호

live-server가 사용하는 디폴트 포트 번호는 8080 이다.

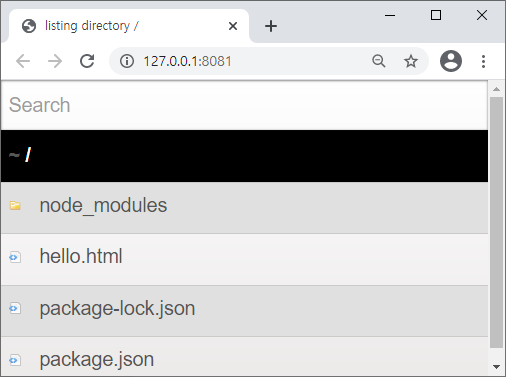
다른 SW가 이미 8080 포트를 사용 중이라면, 에러가 발생할 것이다.

8080 포트를 사용하는 SW들 많으니, 다른 포트 번호를 사용하는 것이 좋다.

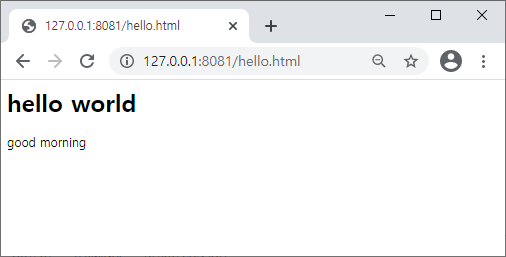
아래 명령을 실행하면, 8081 포트에서 live-server 가 실행되고, 웹브라우저도 자동으로 실행된다.

|  |
| --- |
| live-server --port=8081 |

|  |
| --- |
|  |



hello.html 을 클릭하자.



live-server는 웹브라우저의 http://127.0.0.1:8081/hello.html URL 요청을 받고

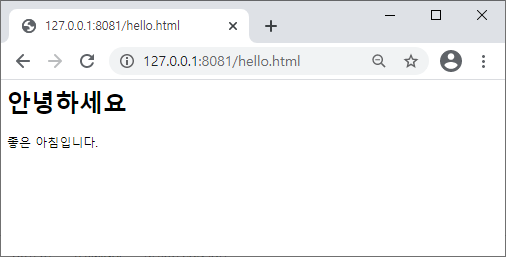
hello.html 파일을 웹브라우저에 전송하는 역할만 한다.

javascript 코드는 웹브라우저에서 실행된다.

## hello.html 수정

hello.html 파일을 수정하자.

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>  </head>  <body>    <div id="app">      <h1>{{ title }}</h1>      <p>{{ body }}</p>    </div>    <script type="text/javascript">      let app = {        data() {          return {            title: '안녕하세요',            body: '좋은 아침 입니다.'          };        }      };      Vue.createApp(app).mount("#app");     </script>  </body>  </html> |



소스코드를 수정하고 저장하면,

수정된 파일이 live-server를 거쳐서 웹브라우저에 즉시 다시 전송된다.

웹브라우저에서 새로고침 할 필요 없다.

# v-model 지시자

## hello1.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      body { padding: 20px; }      input { padding: 5px; margin: 5px; }    </style>  </head>  <body>    <div id="app">      <h1>{{ title }}</h1>      <p>{{ body }}</p>      <hr />      <input type="text" v-model="title" /> <br />      <input type="text" v-model="body" /> <br />    </div>    <script type="text/javascript">      let app = {        data() {          return {            title: '안녕하세요',            body: '좋은 아침 입니다.'          };        }      };      Vue.createApp(app).mount("#app");     </script>  </body>  </html> |

<input type="text" v-model="title" /> <br />

v-model 지시자의 값은 앱의 속성명이다.

(1) 앱의 title 속성값이 이 input 태그에 채워져서 출력된다.

(2) 사용자가 input 태그에 값을 입력하면, 입력된 값이 앱의 title 속성에 즉시 대입된다.

<h1>{{ title }}</h1>

(3) 앱의 title 속성값이 변경되면, 변경된 값이 {{ title }} 부분에 즉시 출력된다.

### v-model 지시자

v-model="title" 지시자는, input 태그와 앱의 title 속성을 양방향으로 연결한다.

<input type="text" v-model="title" />

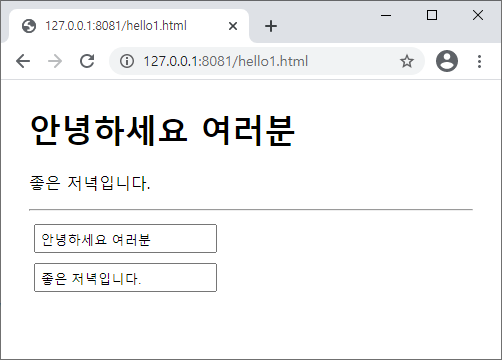
data: {

    title: '안녕하세요',

}

앱의 title 속성값이 변경되면, 그 변경된 내용이 v-model 지시자로 연결된 input 태그에 즉시 채워진다.

input 태그에 값을 입력하면, 입력된 값이 앱의 title 속성에 즉시 대입된다.



input 태그에 값을 입력하면, 입력된 내용이 화면에 즉시 반영된다.

### CSS 서식

body { padding: 20px; }

body 태그의 내부 여백 20px

상, 하, 좌, 우 여백이 모두 20px

input { padding: 5px; margin: 5px; }

input 태그의 내부 여백 5px

외부 여백 5px

### padding과 margin

태그의 경계선(border)을 기준으로, padding은 태그의 안쪽 여백이고, margin은 태그의 바깥쪽 여백이다.



## hello2.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      body { padding: 20px; }      input { padding: 5px; margin: 5px; }    </style>  </head>  <body>    <div id="app">      <h1>{{ title }}</h1>      <p>{{ body }}</p>      <hr />      <input type="text" v-model="title" /> <br />      <input type="text" v-model="body" /> <br />      <hr />      <input type="text" v-model="title" /> <br />      <input type="text" v-model="body" /> <br />    </div>    <script type="text/javascript">      let app = {        data() {          return {            title: '안녕하세요',            body: '좋은 아침 입니다.'          };        }      };      Vue.createApp(app).mount("#app");     </script>  </body>  </html> |

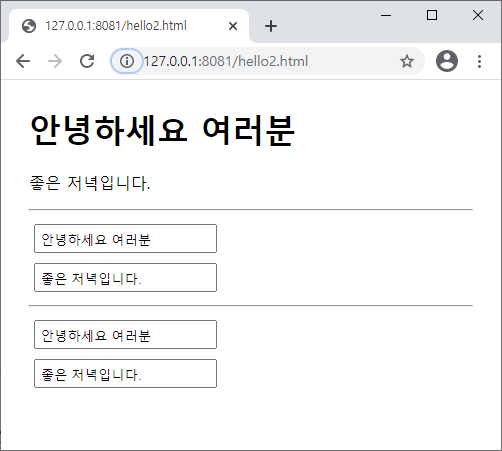
(1) 앱의 title 속성에 연결된 input 태그가 두 개이다.

두 input 태그 중 하나에 값을 입력하면, 입력된 값이 앱의 title 속성에 즉시 대입된다.

(2) 앱의 title 속성의 값이 변경되면,

변경된 값이, title 속성과 연결된 input 태그들에 즉시 채워지고,

{{ title }} 부분에도 즉시 출력된다.



input 태그에 값을 입력하면, 입력된 내용이 화면 전체에 즉시 반영된다.

## checkbox1.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      div#app { margin: 20px auto; padding: 20px; width: 300px;            border: 1px solid gray; box-shadow: 5px 5px 5px #aaa; }      p { font-size: 15pt; color: blue; }      input { width: 30px; height: 30px;  }    </style>  </head>  <body>    <div id="app">        <h1>checkbox</h1>        <p>{{ value }}</p>        <hr />        <input type="checkbox" v-model="value" />    </div>    <script type="text/javascript">      let app = {        data() {          return {            value: false          };        }      };      Vue.createApp(app).mount("#app");     </script>  </body>  </html> |

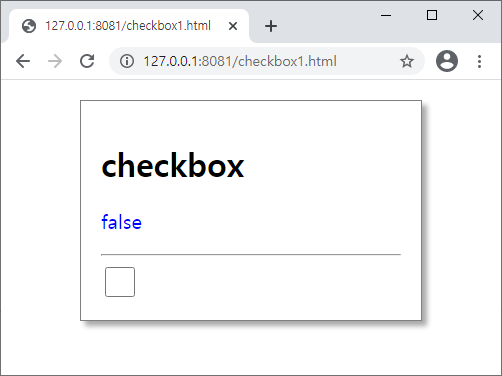
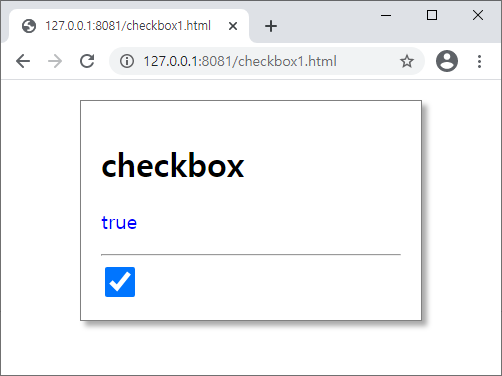
<input type="checkbox" v-model="value" />

이 체크박스는 앱의 value 속성에 양방향으로 연결된다.

이 체크박스의 체크 상태가 바뀌면, 체크 상태 true/false 값이 앱의 value 속성에 즉시 대입된다.

체크박스가 체크되면, 앱의 value 속성에 true 값이 대입되고,

체크가 꺼지면, 앱의 value 속성에 false 값이 대입된다.

### CSS 서식

div#app { margin: 20px auto; padding: 20px; width: 300px;

          border: 1px solid gray; box-shadow: 5px 5px 5px #aaa; }

<div id="app"> 태그의 서식을 지정함.

margin: 20px auto;

아래쪽 위쪽 외부 여백 20px

왼쪽 오른쪽 외부 여백 auto

왼쪽 오른쪽 외부 여백이 auto 이면, 이 태그는 가로 방향으로 가운데 위치하게 된다.

padding: 20px;

위 아래 좌 우 여백 20px

width: 300px;

폭 300px 크기

border: 1px solid gray;

1px 굵기의 회색 실선으로 경계선을 그린다.

box-shadow: 5px 5px 5px #aaa;

5px 크기의 밝은 회색 그림자를 그린다.

p { font-size: 15pt; color: blue; }

p 태그의 서식을 지정함

폰트 크기 15pt

글자색 blue

input { width: 30px; height: 30px;  }

input 태그의 서식을 지정함

width: 30px;

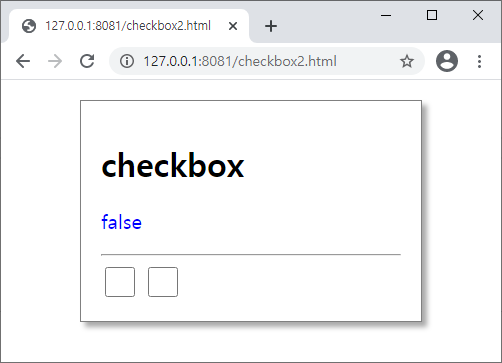
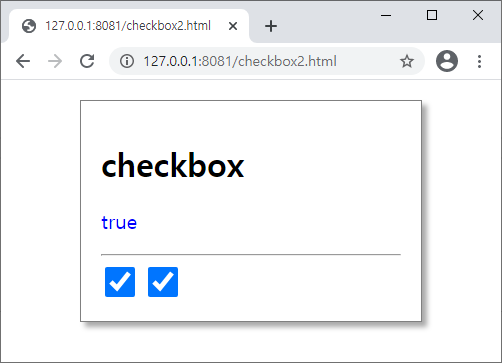
폭 30px 크기

height: 30px;

높이 30px 크기

## checkbox2.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      div#app { margin: 20px auto; padding: 20px; width: 300px;            border: 1px solid gray; box-shadow: 5px 5px 5px #aaa; }      p { font-size: 15pt; color: blue; }      input { width: 30px; height: 30px;  }    </style>  </head>  <body>    <div id="app">        <h1>checkbox</h1>        <p>{{ value }}</p>        <hr />        <input type="checkbox" v-model="value" />        <input type="checkbox" v-model="value" />    </div>    <script type="text/javascript">      let app = {        data() {          return {            value: false          };        }      };      Vue.createApp(app).mount("#app");     </script>  </body>  </html> |

<input type="checkbox" v-model="value" />

v-model 지시자는 양방향 연결이다.

두 체크 박스 중의 하나의 체크 상태가 바뀌면,

체크 상태 true/false 값이 앱의 value 속성에 즉시 대입되고,

앱의 value 속성 값이 바뀌면, 나머지 체크 박스의 체크 상태도 따라서 바뀐다.

## select1.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      div#app { margin: 20px auto; padding: 20px; width: 300px;            border: 1px solid gray; box-shadow: 5px 5px 5px #aaa; }      p { font-size: 15pt; color: blue; }      select { padding: 5px; font-size: 15pt; width: 150px; }    </style>  </head>  <body>    <div id="app">        <h1>select</h1>        <p>{{ choice }}</p>        <hr />        <select v-model="choice">          <option>yellow</option>          <option>green</option>          <option>blue</option>          <option>red</option>        </select>    </div>    <script type="text/javascript">      let app = {        data() {          return {            choice: "yellow"          };        }      };      Vue.createApp(app).mount("#app");     </script>  </body>  </html> |

<select v-model="choice">

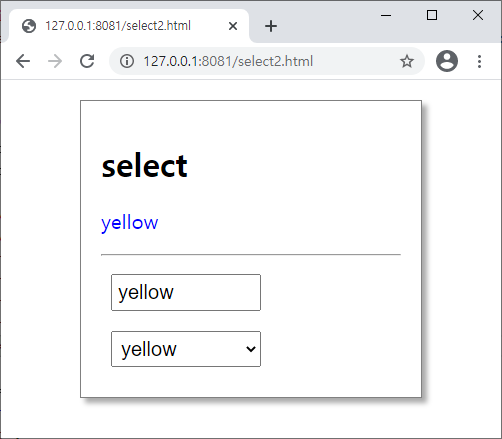
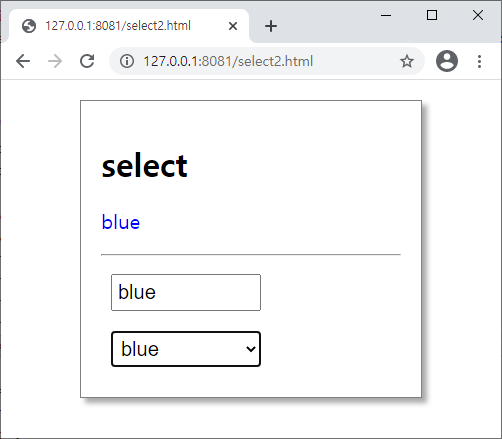
이 select 태그는 앱의 choice 속성에 양방향으로 연결된다.

이 select 태그에서 선택된 항목이 바뀌면, 선택된 항목이 앱의 choice 속성에 즉시 대입된다.

앱의 choice 속성값이 바뀌면, 그 값에 따라 select 태그에서 선택된 항목이 즉시 바뀐다.

## select2.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      div#app { margin: 20px auto; padding: 20px; width: 300px;            border: 1px solid gray; box-shadow: 5px 5px 5px #aaa; }      p { font-size: 15pt; color: blue; }      select { padding: 5px; font-size: 15pt; width: 150px; }      input, select { padding: 5px; font-size: 15pt; width: 150px; margin: 10px; }    </style>  </head>  <body>    <div id="app">        <h1>select</h1>        <p>{{ choice }}</p>        <hr />        <input type="text" v-model="choice" />        <select v-model="choice">          <option>yellow</option>          <option>green</option>          <option>blue</option>          <option>red</option>        </select>    </div>    <script type="text/javascript">      let app = {        data() {          return {            choice: "yellow"          };        }      };      Vue.createApp(app).mount("#app");     </script>  </body>  </html> |

input 태그와 select 태그 둘다 choice 속성에 양방형으로 연결되어 있다.

select 태그에서 선택 항목을 변경하면, 그 값이 즉시 input 태그에도 채워진다.

input 태그에 값을 입력하면, 그 값이 앱의 choice 속성에 즉시 대입된다.

그리고 그 값과 일치하는 select 항목이 즉시 선택된다.

일치하는 select 항목이 없는 경우에는, select 태그는 아무것도 선택되지 않은 상태가 된다.

## radio1.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      div#app { margin: 20px auto; padding: 20px; width: 300px;            border: 1px solid gray; box-shadow: 5px 5px 5px #aaa; }      p { font-size: 15pt; color: blue; }      select { padding: 5px; font-size: 15pt; width: 150px; margin: 10px; }      label { font-size: 14pt; }    </style>  </head>  <body>    <div id="app">        <h1>select</h1>        <p>{{ choice }}</p>        <hr />        <select v-model="choice">          <option>yellow</option>          <option>green</option>          <option>blue</option>          <option>red</option>        </select>        <div>          <label>            <input type="radio" value="yellow" v-model="choice" />yellow          </label>          <label>            <input type="radio" value="green" v-model="choice" />green          </label>          <label>            <input type="radio" value="blue" v-model="choice" />blue          </label>          <label>            <input type="radio" value="red" v-model="choice" />red          </label>        </div>    </div>    <script type="text/javascript">      let app = {        data() {          return {            choice: "yellow"          };        }      };      Vue.createApp(app).mount("#app");     </script>  </body>  </html> |

select 태그와 input type="radio" 태그들이 모두 속성에 양방형으로 연결되어 있다.

select 태그에서 선택 항목을 변경하면, 그 값에 해당하는 radio 태그가 즉시 선택된 상태가 된다.

radio 태그들 중 하나를 선택하면, 그 radio 태그의 value 값이 앱의 choice 속성에 즉시 대입된다.

그리고 앱의 choice 속성와 일치하는 select 항목이 즉시 선택된다.

(1) <input type="radio" value="yellow" v-model="choice" />

이 라디오 버튼을 클릭하여 선택하면, 앱의 choice 속성에 "yellow" 문자열이 즉시 대입된다.

(2) 앱의 choice 속성에 "yellow" 문자열이 대입되면,

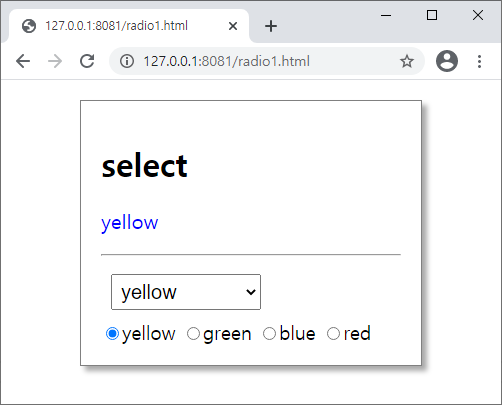
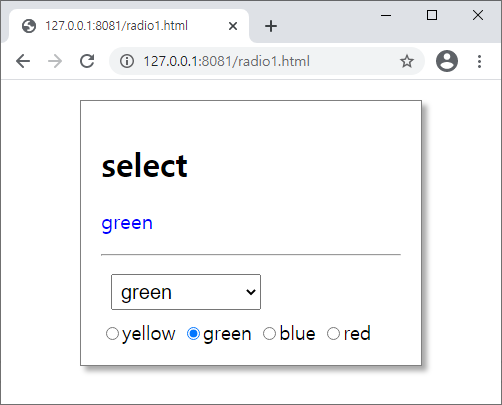
<select v-model="choice"> 태그에서 <option>yellow</option> 항목이 자동으로 선택된 상태가 된다.

(1) <select v-model="choice"> 태그에서 <option>green</option> 항목을 선택되면,

앱의 choice 속성에 "green" 문자열이 즉시 대입된다.

(2) 앱의 choice 속성에 "green" 문자열이 대입되면,

<input type="radio" value="green" v-model="choice" /> 이 라디오 버튼이 자동으로 선택된 상태가 된다.

<label>

  <input type="radio" value="yellow" v-model="choice" />yellow

</label>

input type="radio" 태그를 <label> 태그로 묶어주면,

<label> 태그 내부 아무 곳이나 클릭하면, radio 태그가 선택 상태가 되므로 편리하다.

# 요약

### Vue 3 CDN

  <script src="https://unpkg.com/vue@next"></script>

### {{ 속성명 }}

앱의 속성값이 출력된다.

앱의 속성값이 변경되면 즉시 다시 출력된다.

### v-model="속성명"

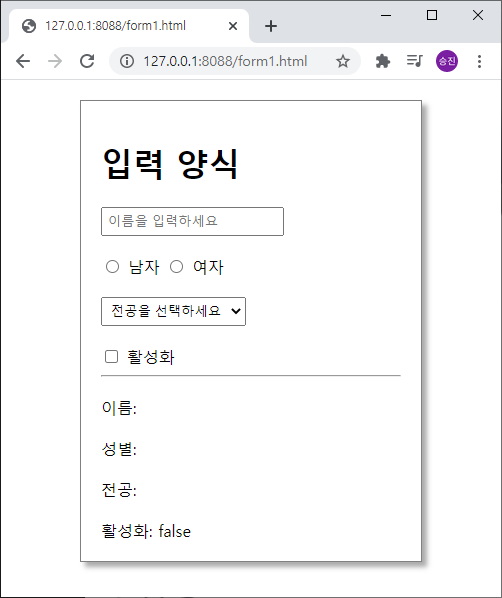
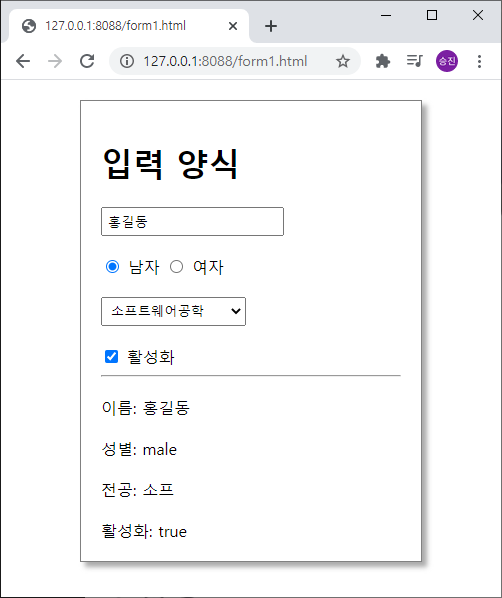
v-model 지시자는 입력 태그와 앱의 속성을 양방향으로 연결한다.

입력 태그에 입력된 값이 수정되면, 연결된 앱의 속성에 그 값이 즉시 대입된다.

연결된 앱의 속성값이 바뀌면, 입력 태그의 값도 즉시 바뀐다.

# 연습 문제

## form1.html

처음 실행한 화면 입력한 화면

위와 같은 화면을 구현하라.

### 힌트

<input type="text" placeholder="이름을 입력하세요"/>

데이터를 입력하기 전에만, '이름을 입력하세요' 안내 문구가 보인다.

데이터를 입력하면 안내 문구가 사라진다.

<option value="소프">소프트웨어공학</option>

이 항목을 선택했을 때의 값은, "소프트웨어공학" 이 아니고, value의 값 "소프" 이다.