props와 event

**학습목표**

Vue 컴포넌트의 props와 event

**목차**

[1. vue\_app21 프로젝트 2](#_Toc151338701)

[1) Single Page App 2](#_Toc151338702)

[2) vue\_app21 프로젝트 생성 2](#_Toc151338703)

[3) 포트 번호 변경 2](#_Toc151338704)

[2. MyCounter1 컴포넌트 3](#_Toc151338705)

[1) MyCounter1.vue 생성 3](#_Toc151338706)

[2) Virtual DOM 4](#_Toc151338707)

[3) App.vue 수정 5](#_Toc151338708)

[4) 실행 6](#_Toc151338709)

[3. props 7](#_Toc151338710)

[1) props 선언 방법 - 배열 7](#_Toc151338711)

[2) props 선언 방법 - 객체 형태 7](#_Toc151338712)

[3) props 선언 방법 - 객체 형태 8](#_Toc151338713)

[4. MyCounter2 컴포넌트 9](#_Toc151338714)

[1) MyCounter2.vue 생성 9](#_Toc151338715)

[2) v-bind:style 10](#_Toc151338716)

[3) $emit 메소드 10](#_Toc151338717)

[4) App.vue 수정 11](#_Toc151338718)

[5) 실행 12](#_Toc151338719)

[5. 연습문제 13](#_Toc151338720)

# vue\_app21 프로젝트

## Single Page App

Vue 앱 프로젝트는 index.html 페이지 하나를 생성하는 프로젝트이다.

javascript 프론트앤드 개발에서는, html 페이지 하나에 앱의 기능을 전부 구현하는 형태의 개발이 흔하다.

html 페이지 하나에 앱의 기능을 전부 구현한 프론트엔드 앱을 Single Page App 이라고 한다. (SPA)

javascript로 구현한 기능은, A 페이지에서 B 페이지로 넘어갈 때,

A 페이지에서 실행 중이던 javascript 코드는 모두 종료되고,

B 페이지의 javascript 코드가 새로 시작하게 된다.

웹페이지 새로고침을 하면, 그 페이지에서 실행 중이던 javascript 코드는 모두 초기화 된다. (종료 후 재시작)

javascript 프론트엔드 앱은 웹 페이지 하나로 구현된다. (Single Page App)

## vue\_app21 프로젝트 생성

프로젝트를 생성할 적당한 폴더로 이동하자.

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

아래 명령을 실행해서 프로젝트를 생성하자.

|  |
| --- |
| vue create vue\_app21 |

|  |
| --- |
|  |

default 가 선택된 상태에서 엔터 키를 누르자.

default 설정에는 babel 도구와 eslint 도구 설정이 포함된다.

조금 기다리면 프로젝트 생성이 완료된다.

## 포트 번호 변경

컴퓨터에 오라클 DBMS나 톰캣이 설치되어 있다면, 8080 포트가 이미 사용중일 수 있다.

vue 프로젝트가 사용할 포트 번호를 8082로 변경하는 방법은 다음과 같다.

### package.json

|  |
| --- |
| ... 생략 ...    "scripts": {      "serve": "vue-cli-service serve --port 8082",      "build": "vue-cli-service build",      "lint": "vue-cli-service lint"    },  ... 생략 ... |

package.json 파일을 열고 위의 내용을 찾아서,

노란색으로 칠한 부분을 추가하자.

# MyCounter1 컴포넌트

## MyCounter1.vue 생성

src/MyCounter1.vue

|  |
| --- |
| <template>    <div>      <h2>{{ title }} - <small>{{ description }}</small></h2>      <input type="number" v-model="counter" />      <button type="button" @click="counter++"> ++ </button>      <button type="button" @click="counter--"> -- </button>    </div>  </template>  <script>  export default {    name: "MyCounter1",    props: [ "title", "description" ],    data() {      return { counter: 0 };    }  }  </script>  <style scoped>    div { border: 1px solid gray; padding: 0 20px 20px 20px; margin-top: 10px; }    input { padding: 5px; margin-right: 5px; }    button { padding: 0.4em 1em; margin-right: 5px; }  </style> |

### v-on: 접두어와 @ 문자

|  |
| --- |
| <button type="button" @click="counter++"> ++ </button> |

위 코드는 아래 코드와 동일하다.

|  |
| --- |
| <button type="button" v-on:click="counter++"> ++ </button> |

즉 v-on:click 에서 v-on: 대신 @ 문자를 사용할 수 있다.

### name: "MyCounter1",

이 컴포넌트의 이름은 "MyCounter1" 이다.

### props: [ "title", "description" ]

부모 컴포넌트로부터 자식 컴포넌트로 전달되는 값을 props 이라고 한다.

<h2>{{ title }} - <small>{{ description }}</small></h2>

title props 값과 description props 값이 이 부분에 출력된다.

### 컴포넌트의 속성

data() {

  return { counter: 0 };

}

컴포넌트 속성을 정의하는 data 값은 함수이어야 한다.

컴포넌트의 속성은, 이 함수가 리턴하는 객체에 정의되어야 한다.

<button type="button" @click="counter++"> ++ </button>

<button type="button" @click="counter--"> -- </button>

++ 버튼을 클릭하면, counter++ 코드가 실행된다. 컴포넌트의 counter 속성값이 1 증가한다.

-- 버튼을 클릭하면, counter-- 코드가 실행된다. 컴포넌트의 counter 속성값이 1 감소한다.

<input type="number" v-model="counter" />

컴포넌트의 counter 속성값이 input 태그에 실시간으로 채워진다.

## Virtual DOM

<template> 부분은 표준 html 태그가 아니고, Virtual DOM 객체이다.

Virtual DOM 개념이 꽤 어렵기 때문에, 이것을 배우는 것은 뒤로 미루자.

일단 <template> 부분이 표준 html 태그가 아니라는 것만 기억하자.

### Virtual DOM 태그 사이 여백

표준 html 태그에서는, 태그와 태그 사이에 있는 공백이나 줄바꿈 문자가, 공백 하나로 웹브라우저 창에 그려진다. 이 공백이 태그들 사이의 작은 여백 역할을 한다.

<template> 부분에서는, 태그와 태그 사이의 공백이나 줄바꿈 문자가 전부 제거된다.

그래서 표준 html과 달리 태그들 사이에 작은 여백이 전혀 없이 웹브라우저 창에 그려진다.

그래서 <template> 부분에서는, 태그들 사이의 작은 여백을 주기 위한 margin 값이 필요하다.

예:

  input { padding: 5px; margin-right: 5px; }

  button { padding: 0.4em 1em; margin-right: 5px; }

이 margin-right 값을 주지 않으면, input 태그와 button 태그들이 딱 붙게 된다.

### DOM

Document Object Model

html 태그 객체

웹브라우저 내부에 구현된 태그 객체 (C++)

### Virtual DOM

javascript 라이브러리에 구현된 태그 객체 (javascript)

### 랜더링 과정

Virtual DOM을 이용하여 화면을 그린다.

화면을 그린 후 앱의 데이터가 변경되었다.

그래서 화면을 다시 그려야 한다.

(1) 변경된 데이터를 반영해서 Virtual DOM 웹페이지를 새로 만듦. (화면 전체를 새로 만든다)

(2) 직전에 화면을 그릴 때 만들었던 Virtual DOM 웹페이지와

방금 새로 만든 Virtual DOM 웹페이지를 비교함 (직전에 만들었던 화면과 방금 새로 만든 화면을 비교한다)

(3) 위의 비교를 통해 발견한 차이.

그 차이를 웹브라우저 DOM에 반영함.

## App.vue 수정

src/App.vue

|  |
| --- |
| <template>    <div id="app">      <h1>MyCounter1</h1>      <MyCounter1 title="카운터" description="첫째 컴포넌트" />      <MyCounter1 title="카운터" description="둘째 컴포넌트" />    </div>  </template>  <script>  import MyCounter1 from './MyCounter1.vue'  export default {    name: "App",    components: { MyCounter1  }  }  </script>  <style scoped>  div#app { padding: 0 30px 30px 30px; margin: 30px auto; max-width: 400px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa; }  </style> |

<MyCounter1 title="카운터" description="첫째 컴포넌트" />

Counter1 컴포넌트 객체를 생성한다.

title props 값으로 "카운터" 문자열을 전달한다.

description props 값으로 "첫째 컴포넌트" 문자열을 전달한다.

<MyCounter1 title="카운터" description="둘째 컴포넌트" />

Counter1 컴포넌트 객체를 한 개 더 생성한다.

import MyCounter1  from './MyCounter1.vue'

컴포넌트를 사용하려면, 먼저 import 해야 한다.

경로명에서 './' 부분을 생략할 수 없다.

'./MyCounter1.vue' 경로명은, 동일한 폴더에 있는 MyCounter1.vue 파일을 의미한다.

name: "App",

이 컴포넌트의 이름은 "App" 이다.

components: { MyCounter1 }

사용할 컴포넌트를 여기서 미리 선언해야 한다.

## 실행

쉘에서 다음 명령을 실행하자.

|  |
| --- |
| npm run serve |

Graphical user interface

Description automatically generated

# props

## props 선언 방법 - 배열

부모 컴포넌트로부터 자식 컴포넌트로 전달되는 값을 props 이라고 한다.

### 문자열 배열 형태

|  |
| --- |
| export default {    name: "Counter1",    props: [ "title", "description" ],  } |

컴포넌트의 props를 위와 같이 문자열 배열 행태로 선언할 수 있다.

### props 값 전달

|  |
| --- |
| <MyCounter1  title="카운터1" description="첫째 컴포넌트" /> |

위와 같이 props 값을 전달한다.

### props 값 전달 생략 가능

props 값을 전달하지 않아도 에러가 아니다.

그 props 값은 undefined 가 된다.

## props 선언 방법 - 객체 형태

|  |
| --- |
| props: {    title: String,    description: String  } |

위와 같이 객체 형태로 props를 선언할 수 있다.

props 타입은 String, Number, Boolean, Array, Object, Function 등이 가능하다.

### props 값 전달 v-bind

|  |
| --- |
| props: {    value: Number,    enabled: Boolean  } |

위와 같은 props 선언이 있을 때,

|  |
| --- |
| <MyCounter1 value="7" enabled="true" /> |

위와 같이 props 값을 전달하는 것은 오류이다.

태그의 애트리뷰트 값은 전부 string으로 해석되기 때문이다.

태그의 애트리뷰트 값을 javascript 표현식으로 해석하려면, v-bind: 접두어를 붙어야 한다.

|  |
| --- |
| <MyCounter1 v-bind:value="7" v-bind:enabled="true" /> |

위와 같이 props 값을 전달해야, 7 Number 값과 true Boolean 값이 전달된다.

props 값이 문자열이 아니라면, v-bind: 접두어가 필요하다.

## props 선언 방법 - 객체 형태

### MyCounter1.vue

|  |
| --- |
| props: {    title: { type: String, required: true },    description: { type: String, default: "안녕하세요" }  } |

required: true 속성이 선언된 props는 값이 전달이 필수이다.

값을 전달하지 않을 경우 에러가 발생한다.

default: 값을 전달하지 않은 경우 디폴트값

### App.vue

|  |
| --- |
| <MyCounter1 description="첫째 컴포넌트" />  <MyCounter1 title="카운터1" /> |

위와 같이 Counter1.vue 파일과 App.vue 파일을 수정하고 실행하자.

기능을 정상적으로 작동한다.

F12 키를 눌러서 웹브라우저 개발자 창을 열과 Console 탭을 보자.

Graphical user interface, text, application

Description automatically generated

경고 메시지가 출력되었다.

javascript 실행 오류 에러 메시지는 개발자창의 Console 탭에 출력된다.

에러 메시지가 혹시 있는지 종종 Console 탭을 확인해야 한다.

# MyCounter2 컴포넌트

## MyCounter2.vue 생성

src/MyCounter2.vue

|  |
| --- |
| <template>    <div :style="{ backgroundColor: color }">      <h2>{{ title }} - <small>{{ description }}</small></h2>      <input type="number" v-model="counter" />      <button type="button" @click="counter++"> ++ </button>      <button type="button" @click="counter--"> -- </button>      <button type="button" @click="$emit('send', counter)"> 전달 </button>    </div>  </template>  <script>  export default {    name: "MyCounter1",    props: {      title: { type: String, required: true },      description: { type: String, default: "안녕하세요" },      color: { type: String, default: "#fff" }    },    data: function() {      return { counter: 0 };    }  }  </script>  <style scoped>    div { border: 1px solid gray; padding: 0 20px 20px 20px; margin-top: 10px; }    input { padding: 5px; margin-right: 5px; }    button { padding: 0.4em 1em; margin-right: 5px; }  </style> |

<div :style="{ backgroundColor: color }">

이 태그에 background-color CSS 서식을 적용한다.

<button type="button" @click="$emit('send', counter)"> 전달 </button>

이 버튼이 클릭되면 부모에게 'send' 이벤트를 전달한다.

이벤트의 파라미터로 counter 값을 전달한다.

## v-bind:style

<template> 영역의 태그에 CSS 서식을 적용하려면 v-bind:style 애트리뷰트를 사용해야 한다.

|  |
| --- |
| <div v-bind:style="{ backgroundColor: color }"> |

v-bind 접두어가 붙었으므로 "{ backgroundColor: color }" 이 부분은 javascript 표현식으로 해석된다.

javascript 표현식으로 해석하면 { backgroundColor: color } 이것은 객체이다.

backgroundColor 부분은 속성명이고, color 부분은 속성의 값이다.

속성명은 문자열로 해석되고, 속성값은 javascript 표현식으로 해석된다.

즉 속성명은 "backgroundColor" 이고, 속성값은 color props 값이다.

배경색 CSS 서식 속성의 이름은 background-color 이다.

그런데 javascript 속성명에 - 문자를 사용할 수 없기 때문에,

background-color 서식 속성명 대신, backgroundColor 속성명을 사용한다.

### v-bind: 접두어와 : 문자

v-bind: 접두어 대신 : 문자만 입력해도 된다.

v-on: 접두어 대신 @ 문자만 사용할 수 있는 것 처럼

v-bind: 접두어 대신 : 문자만 사용할 수 있다.

|  |
| --- |
| <div :style="{ backgroundColor: color }"> |

## $emit 메소드

|  |
| --- |
| <button type="button" @click="$emit('send', counter)"> 전달 </button> |

자식 컴포넌트로부터 부모 컴포넌트에게 이벤트를 전달한다.

이 이벤트의 이름은 'send' 이고, 이벤트 파라미터 값으로 counter 값을 전달한다.

$emit('이벤트\_이름', 이벤트\_파라미터1, 이벤트\_파라미터2,...)

이벤트 파라미터의 수는 0 개 이상이다. 즉 이벤트 파라미터가 없을 수도 있고, 한 개나 여러 개 있을 수도 있다.

이벤트를 전달하는 이유는, 자식 컴포넌트로부터 부모 컴포넌트에게 어떤 시점(event)을 알리거나, 어떤 값을 전달하기 위함이다.

$emit 메소드를 호출하는 곳이

method 본문에서는, this를 생략할 수 없으니, this.$emit(...) 형태이어야 한다.

태그 부분이면, this를 생략해야 하니, $emit(...) 형태이어야 한다.

### 이벤트 전달 받기

|  |
| --- |
| <MyCounter2 title="카운터2"  v-on:send="showValue" /> |

이벤트를 전달 받는 방법은, v-on:이벤트명="메소드" 이다.

MyCounter2 컴포넌트에서 send 이벤트를 전달하면, 부모 컴포넌트의 showValue 메소드가 호출된다.

MyCounter2 컴포넌트가 전달한 이벤트 파라미터 값은, showValue 메소드의 파라미터로 전달된다.

## App.vue 수정

src/App.vue

|  |
| --- |
| <template>    <div id="app">      <h1>MyCounter2</h1>      <MyCounter2 title="카운터2" description="첫째 컴포넌트" color="#ffb"  v-on:send="showValue" />      <MyCounter2 title="카운터2" color="#afa"  v-on:send="showValue" />    </div>  </template>  <script>  import MyCounter2 from './MyCounter2.vue'  export default {    name: "App",    methods: {      showValue(value) {        alert(value);      }    },    components: { MyCounter2 }  }  </script>  <style scoped>  div#app { padding: 0 30px 30px 30px; margin: 30px auto; max-width: 400px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa; }  </style> |

MyCounter2 컴포넌트에서 send 이벤트를 전달하면, showValue 메소드가 호출된다.

MyCounter2 컴포넌트가 전달한 이벤트 파라미터 값은, showValue 메소드의 파라미터로 전달된다.

### v-on: 접두어와 @ 문자

v-on:send 부분을 @send 으로 수정해도 된다.

## 실행

Graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generated

# 연습문제

MyText 컴포넌트

input 태그와 button 태그

input 태그에 문자열을 입력하고 button을 클릭하면 'text' 이벤트와 입력된 문자열이 부모 태그에 전달.

App 컴포넌트

text' 이벤트를 전달 받아서, 화면에 <p> 태그에 출력