Vue.js 기초 #2

**학습목표**

CSS 서식 변경 기능 구현

메소드 리턴값 출력

계산 속성 구현

**목차**

[1. CSS 클래스 2](#_Toc100683367)

[1) v-bind:class="javascript 객체" 2](#_Toc100683368)

[2) v-bind:class="javascript 문자열" 4](#_Toc100683369)

[2. 인라인 서식 5](#_Toc100683370)

[1) Naming Convention 5](#_Toc100683371)

[2) v-bind:style="javascript 객체" 6](#_Toc100683372)

[3. 계산 결과 출력 8](#_Toc100683373)

[1) 계산식 출력 8](#_Toc100683374)

[2) 메소드 리턴값 출력 10](#_Toc100683375)

[3) 계산 속성 getter 구현 11](#_Toc100683376)

[4) 계산 속성 getter setter 구현 15](#_Toc100683377)

[4. 연습문제 17](#_Toc100683378)

# CSS 클래스

## v-bind:class="javascript 객체"

### checkbox5.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      div#box { 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;  }      .green { background-color: #afa; }    </style>  </head>  <body>    <div id="app">  <div id="box" v-bind:class="{ green: value }">        <h1>checkbox</h1>        <p>{{ value }}</p>        <hr />        <input type="checkbox" v-model="value" />  </div>    </div>    <script type="text/javascript">      let app = {        data() {          return {          value: false          };        }      };      Vue.createApp(app).mount("#app");     </script>  </body>  </html> |

v-bind:class="{ green: value }"

value 속성값이 true 이면, 위 소스코드의 실행 결과는 다음과 같다.

class="green"

즉 green CSS 클래스 서식이 적용된다.

value 속성값이 false 이면, 위 소스코드의 실행 결과는 다음과 같다.

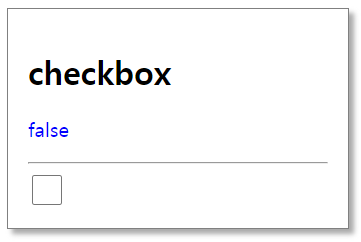
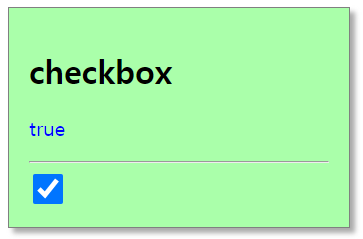
class=""

즉 CSS 클래스 서식이 적용되지 않는다.

class 속성에 v-bind: 접두어가 붙었으므로, { green: value } 부분은 javascript 객체이다.

속성명 green 부분은 CSS 클래스 이름이어야 한다.

속성값은 true/false 조건식이어야 한다.

### checkbox6.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      div#box { 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;  }      .green { background-color: #afa; }      .red { background-color: #faa; }    </style>  </head>  <body>    <div id="app">    <div id="box" v-bind:class="{ green: value == true, red: value == false }">        <h1>checkbox</h1>        <p>{{ value }}</p>        <hr />        <input type="checkbox" v-model="value" />    </div>    </div>    <script type="text/javascript">      let app = {        data() {          return {          value: false          };        }      };      Vue.createApp(app).mount("#app");    </script>  </body>  </html> |

v-bind:class="{ green: value == true, red: value == false }"

value == true 조건식이 true 이면, 위 소스코드의 실행 결과는 다음과 같다.

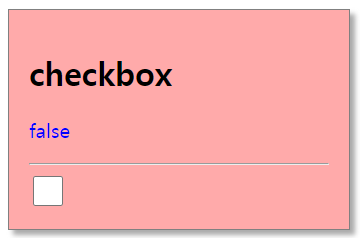
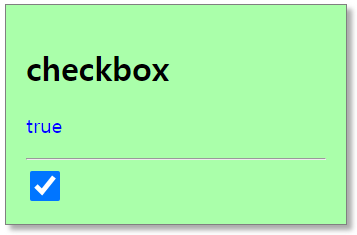
class="green"

즉 green CSS 클래스 서식이 적용된다.

value == false 조건식이 true 이면, 위 소스코드의 실행 결과는 다음과 같다.

class="red"

즉 red CSS 클래스 서식이 적용된다.

## v-bind:class="javascript 문자열"

### select3.html

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

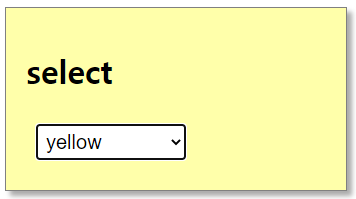
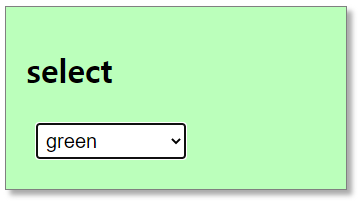
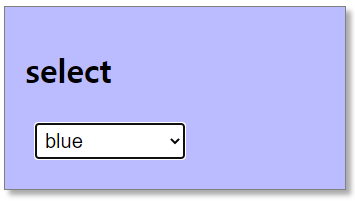
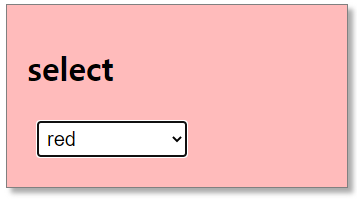
v-bind:class="color"

color 속성의 값은 문자열이다.

이 값은 CSS 클래스 이름과 일치해야 한다.

color 속성의 값이 "red" 이면 위 소스코드의 실행 결과는 다음과 같다.

class="red"

# 인라인 서식

## 이름 표기법

여러 단어로 구성된 변수 이름이나 메소드 이름을 표기하는 방식

- camel case

- snake case

- kebab case

- Pascal case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 예 | camel case | snake case | kebab case | Pascal case |
| font size | fontSize | font\_size | font-size | FontSize |
| left margin | leftMargin | left\_maring | left-margin | LeftMargin |
| background color | backgroundColor | background\_color | background-color | BackgroundColor |

Java 변수명, 속성명, 메소드 이름은 camel case 방식이 관례이다.

Java 클래스 이름은 Pascal case 방식이 관례이다.

C 변수 함수 이름은 snake case 방식이 관례이다.

CSS 서식 이름은 kebab case 이다.

예: font-size, margin-left, background-color

## v-bind:style="javascript 객체"

### select4.html

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

style 속성에 v-bind: 접두어가 붙었으므로, { backgroundColor: color } 부분은 javascript 객체이다.

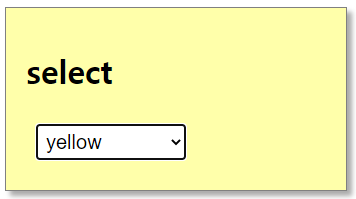
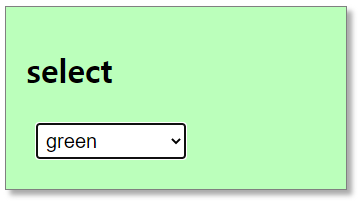
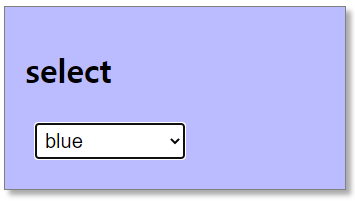
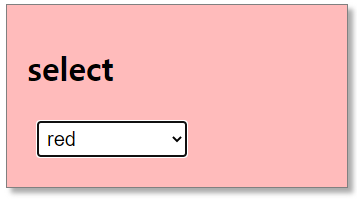
속성명 backgroundColor 부분은 CSS 서식 이름이어야 한다.

속성값은 CSS 서식 값이어야 한다.

### Naming Convention 변환

CSS 속성명은 kebab case 이지만, Java 객체 속성명은 camel case 이어야 한다.

예: background-color => backgroundColor

예를 들어서

(1) select 태그에서 yellow 항목을 선택하면

(2) 앱의 color 속성에 "#ffa" 값이 대입되고

(3) v-bind:style="{ backgroundColor: color }"

위 소스코드는 다음과 같이 된다

style="background-color: #ffa;"

style은 표준 HTML 속성이므로 "background-color: #ffa;" 부분도 평범한 HTML 속성값이다.

### select5.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      div#box { margin: 20px auto; padding: 20px; width: 300px;            border: 1px solid gray; box-shadow: 5px 5px 5px #aaa;            transition: background-color 1s, height 1s; }      select { padding: 5px; font-size: 15pt; width: 150px; margin: 10px; }    </style>  </head>  <body>    <div id="app">    <div id="box" v-bind:style="{ backgroundColor: color, height: h }">        <h1>select</h1>        <select v-model="color">          <option value="#ffa">yellow</option>          <option value="#bfb">green</option>          <option value="#bbf">blue</option>          <option value="#fbb">red</option>        </select>        <select v-model="h">          <option>200</option>          <option>300</option>          <option>400</option>        </select>     </div>   </div>    <script type="text/javascript">      let app = {        data() {          return {           color: "#ffa",          h: 200         };        }      };      Vue.createApp(app).mount("#app");    </script>  </body>  </html> |

Diagram

Description automatically generated with low confidence Graphical user interface

Description automatically generated

transition: background-color 1s, height 1s;

background-color 서식의 값이 변경될 때, 1초 간격으로 천천히 변경된다.

height 서식의 값이 변경될 때, 1초 간격으로 천천히 변경된다.

v-bind:style="{ backgroundColor: color, height: h }"

예를 들어 color 속성값이 "#ffa" 이고, h 속성값이 200 이면

위 소스코드의 실행 결과는 다음과 같다.

style="backgroundColor: #ffa; height: 200px;"

# 계산 결과 출력

## 계산식 출력

### compute1.html

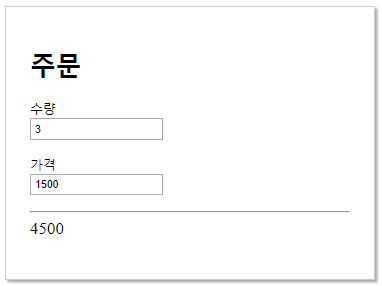
|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      div#app { padding: 30px; margin: 30px auto; width: 400px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa; }      div#app div { margin-bottom: 20px; }      label { display: block; }      input[type=text] { padding: 4px; }      div.total { font-size: 16pt; }    </style>  </head>  <body>    <div id="app">      <h1>주문</h1>      <div>        <label>수량</label>        <input type="text" v-model="quantity" />      </div>      <div>        <label>가격</label>        <input type="text" v-model="price" />      </div>      <hr />      <div class="total">        {{ quantity \* price }}      </div>    </div>    <script type="text/javascript">      let app = {        data() {          return {          quantity: 1,           price: 0         };        }      };      Vue.createApp(app).mount("#app");    </script>  </body>  </html> |

{{ quantity \* price }}

quantity 속성값과 price 속성값이 곱셈값이 출력된다.

두 속성 중에 하나가 변경되면, 곱셈값도 다시 계산되어 출력된다.

계산식이 간단하다면 위와 같이 구현할 수 있다.



## 메소드 리턴값 출력

### compute2.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      div#app { padding: 30px; margin: 30px auto; width: 400px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa; }      div#app div { margin-bottom: 20px; }      label { display: block; }      input[type=text] { padding: 4px; }      div.total { font-size: 16pt; }    </style>  </head>  <body>    <div id="app">      <h1>주문</h1>      <div>        <label>수량</label>        <input type="text" v-model="quantity" />      </div>      <div>        <label>가격</label>        <input type="text" v-model="price" />      </div>      <hr />      <div class="total">        {{ total() }}      </div>    </div>    <script type="text/javascript">      let app = {        data() {          return {            quantity: 1,            price: 0         };        },        methods: {          total() {            return this.quantity \* this.price;          }        }      };      Vue.createApp(app).mount("#app");    </script>  </body>  </html> |

{{ total() }}

total() 메소드의 리턴값이 출력된다.

두 속성 중에 하나가 변경되면, 메소드가 다시 호출되어 그 리턴값이 출력된다.

계산식이 복잡하다면, 위와 같이 메소드의 형태로 구현하는 것이 바람직할 것이다.

## 계산 속성 getter 구현

### compute3.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      div#app { padding: 30px; margin: 30px auto; width: 400px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa; }      div#app div { margin-bottom: 20px; }      label { display: block; }      input[type=text] { padding: 4px; }      div.total { font-size: 16pt; }    </style>  </head>  <body>    <div id="app">      <h1>주문</h1>      <div>        <label>수량</label>        <input type="text" v-model="quantity" />      </div>      <div>        <label>가격</label>        <input type="text" v-model="price" />      </div>      <hr />      <div class="total">        {{ total }}      </div>    </div>    <script type="text/javascript">      let app = {        data() {          return {            quantity: 1,            price: 0         };        },        computed:  {          total() {            return this.quantity \* this.price;          }        }      };      Vue.createApp(app).mount("#app");    </script>  </body>  </html> |

### 계산 속성 (computed property)

computed:  {

  total() {

    return this.quantity \* this.price;

  }

}

위 코드의 구조는 아래의 코드와 동일하다.

computed:  {

  total: function() {

    return this.quantity \* this.price;

  }

}

Vue 앱에 total 계산 속성을 정의한다. (computed property)

계산 속성이란, 구현 형태는 메소드인데, 사용법은 속성과 같은 것을 말한다.

계산 값을 리턴하는 getter 메소드와, 값을 대입하는 setter 메소드로 계산 속성을 구현한다.

getter 메소드만 구현하면, 읽기 전용 계산 속성이 된다.

위의 total 계산 속성은 getter 메소드만 구현했다.

{{ total }}

total 계산 속성의 값이 출력된다.

### 메소드 vs 계산 속성

메소드의 리턴값을 출력하는 형태로 구현하면, 화면을 다시 그릴 때 마다 메소드가 다시 호출된다.

계산 속성의 값을 출력하는 형태로 구현하면,

getter 메소드의 리턴값이 캐시에 저장된 후 화면에 출력된다.

그리고 다시 화면을 그려야 할 때, 계산 속성의 값을 다시 계산할 필요가 없다면, 캐시에 저장된 값을 사용하여 화면을 그린다.

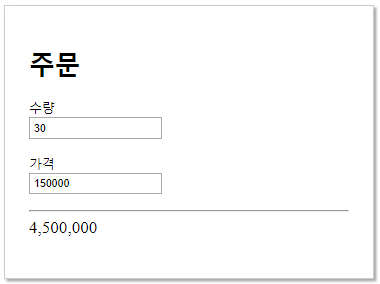
따라서 계산식의 값을 화면에 출력해야 하는 경우에, 그 계산식을 메소드로 구현하는 것보다,

계산 속성으로 구현하는 것이 바람직하다.

물론 계산식이 매우 간단하다면, {{ quantity \* price }} 형태로 구현하는 것도 좋다.

### compute4.html (부록: 시험 범위 아님)

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      div#app { padding: 30px; margin: 30px auto; width: 400px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa; }      div#app div { margin-bottom: 20px; }      label { display: block; }      input[type=text] { padding: 4px; }      div.total { font-size: 16pt; }    </style>  </head>  <body>    <div id="app">      <h1>주문</h1>      <div>        <label>수량</label>        <input type="text" v-model="quantity" />      </div>      <div>        <label>가격</label>        <input type="text" v-model="price" />      </div>      <hr />      <div class="total">        {{ total }}      </div>    </div>    <script type="text/javascript">      let app = {        data() {          return {           quantity: 30,          price: 150000         };        },        computed:  {          total() {            let s = String(this.quantity \* this.price);            for (let i = 3; s.length > i; i += 4)              s = s.slice(0, -i) + "," + s.slice(-i);            return s;          }  }      };      Vue.createApp(app).mount("#app");    </script>  </body>  </html> |



4,500,000 형태로 total 계산 속성값을 출력하기 위한 코드를 추가하였다.

let s = String(this.quantity \* this.price);

this.quantity \* this.price 계산식의 값을 문자열로 변환하여, 지역 변수 s에 대입한다.

s.slice(0, -i)

s 문자열의 인덱스 0 위치부터, 오른쪽 끝에서 i 번째 위치까지의 부분 문자열을 리턴한다.

s.slice(-i)

s 문자열의 오른쪽 끝에서 i 번째 위치부터, 끝까지의 부분 문자열을 리턴한다.

s = s.slice(0, -i) + "," + s.slice(-i);

s 문자열의 오른쪽 끝에서 i 번째 위치에, "," 문자열이 삽입된 새 문자열을 생성하여, 변수 s에 대입한다.

for (let i = 3; s.length > i; i += 4)

   s = s.slice(0, -i) + "," + s.slice(-i);

s 문자열의 오른쪽 끝에서 3, 7, 11, 15,.. 번째 위치에 "," 문자열을 삽입한다.

12,000,000,000

**11** 10 9 8 **7** 6 5 4 **3** 2 1 0

## 계산 속성 getter setter 구현

### compute5.html

|  |
| --- |
| <html>  <head>    <meta charset="utf-8" />    <script src="https://unpkg.com/vue@next"></script>    <style>      div#app { padding: 30px; margin: 30px auto; width: 400px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa; }      div#app div { margin-bottom: 20px; }      label { display: block; }      input[type=text] { padding: 4px; }    </style>  </head>  <body>    <div id="app">      <h1>주문</h1>      <div>        <label>수량</label>        <input type="text" v-model="quantity" />      </div>      <div>        <label>가격</label>        <input type="text" v-model="price" />      </div>      <hr />      <div>        <label>합계</label>        <input type="text" v-model="total" />      </div>    </div>    <script type="text/javascript">      let app = {        data() {          return {           quantity: 1,          price: 150         };        },        computed:  {          total : {            get() {              return this.quantity \* this.price;            },            set(value) {              this.quantity = value / this.price;            }          }  }      };      Vue.createApp(app).mount("#app");    </script>  </body>  </html> |

### getter / setter 구현

total : {

  get() {

    return this.quantity \* this.price;

  },

  set(value) {

    this.quantity = value / this.price;

  }

}

위 코드의 구조는 아래 코드와 동일하다.

total : {

  get: function() {

    return this.quantity \* this.price;

  },

  set: function(value) {

    this.quantity = value / this.price;

  }

}

total 계산 속성에 getter 메소드 뿐만 아니라, setter 메소드도 구현하였다.

따라서 total 계산 속성의 값을 읽을 수 있을 뿐만 아니라, 이 속성에 값을 대입할 수도 있다.

<input type="text" v-model="total" />

이 input 태그에 값을 입력하면 total 계산 속성에 그 값이 실시간으로 대입된다.

즉 total 계산 속성의 setter 메소드가 즉시 호출된다.

set 메소드의 파라미터 값은, 계산 속성에 대입되는 값이다.



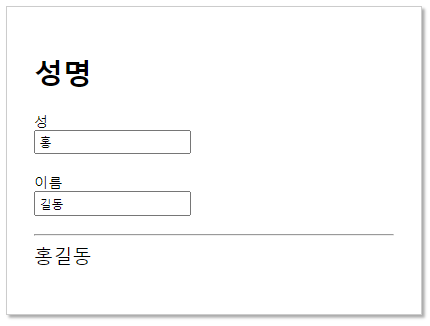
가격을 입력한 후 합계를 입력하면, 그 합계에 맞춰서 수량이 변경된다.

# 연습문제



합계가 10000원 이상일 때만, 위 화면과 같이 합계가 빨간색으로 보이도록 구현하라.

v-bind:class 와 v-bind:style 두 방식 다 구현해 보라.



성(last name)과 이름(first name)을 따로 입력 받아서,

이름 전체(full name)를 출력하는 기능을 computed property로 구현하라.



from 값 이상 to 값 이하의 모든 정수를 더한 합계를 출력하는 기능을

method로 구현하라.