

# 원쌤의 Vue.js 퀵스타트

## 6. 스타일 적용



# 1. HTML의 스타일 적용

## ❖ 웹 애플리케이션의 UI 디자인을 위해 다음을 주로 사용

- style attribute
- CSS Class

## ❖ CSS 스타일의 표기법

- kebob casing
  - 예) font-size
- 사용 이유 : HTML, CSS 스타일이 대소문자를 구별하지 않기 때문에

## ❖ 인라인 스타일

- 각 요소마다 style attribute를 이용해 스타일 지정하는 방법
- 권장하지 않음
  - 스타일을 변경하려면 모든 요소의 style attribute를 일일이 변경해야 함
  - 그렇기 때문에 CSS 클래스를 지정하고 HTML 요소에 바인딩하는 방법을 주로 사용함

# 1. HTML의 스타일 적용

## ❖ CSS 클래스를 이용한 스타일 지정 방법

- 동일한 디자인을 여러 요소에 적용
- 스타일 변경이 용이함 --> 유지보수 편리

---

```
<style>
    .test { background-color:aqua; color:brown; border:solid 1px black; }
</style>
<button class="test">버튼 1</button>
<button class="test">버튼 2</button>
<button class="test">버튼 3</button>
```

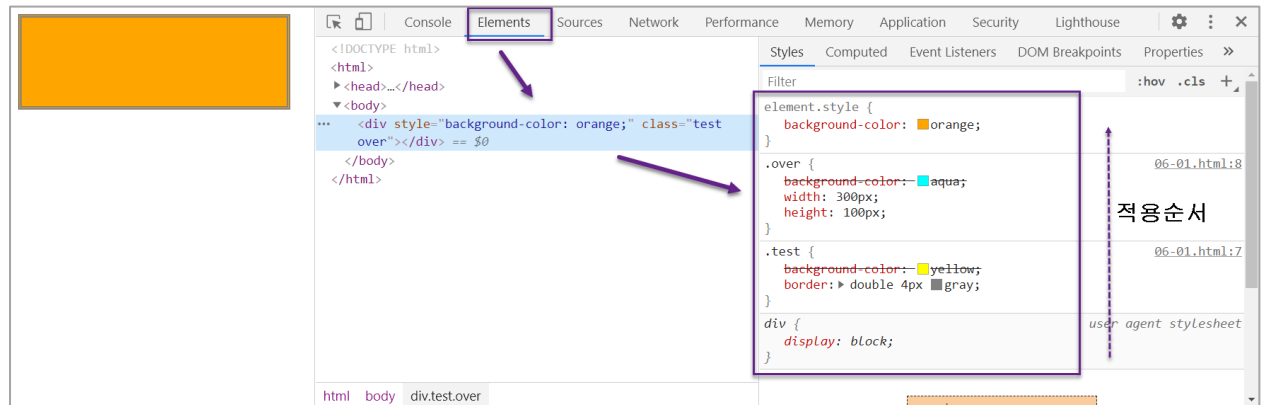
---

# 1. HTML의 스타일 적용

## ❖ 요소에 클래스, 스타일 적용 순서

- CSS 클래스 로딩 순서에 의해 결정
- 예제 06-01

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>06-01</title>
  <style>
    .test { background-color: yellow; border: double 4px gray; }
    .over { background-color: aqua; width: 300px; height: 100px; }
  </style>
</head>
<body>
  <div style="background-color: orange;"
    class="test over" ></div>
</script>
</body>
</html>
```



요소의 기본 스타일 --> .test 스타일 --> .over 스타일 --> 인라인 스타일

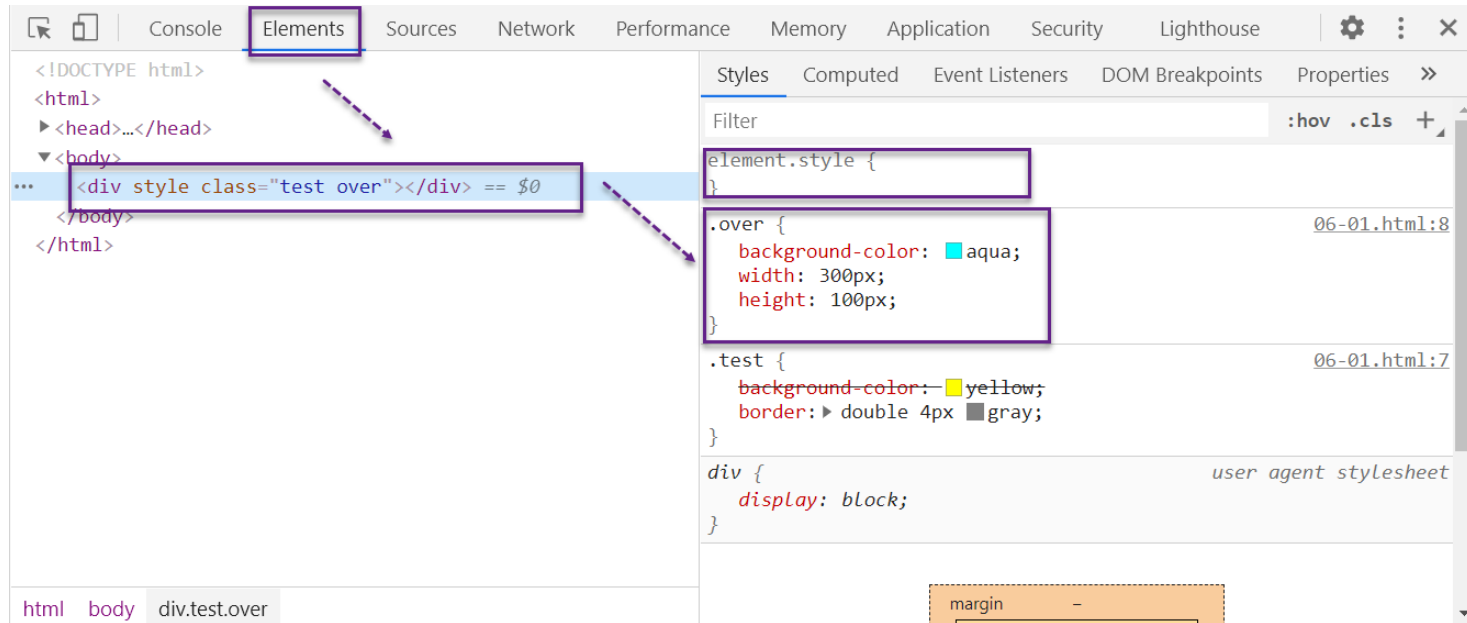
# 1. HTML의 스타일 적용

❖예제 06-01에서 인라인 스타일을 제거하면?

- 직전에 적용된 .over 클래스 스타일의 색상이 나타남

Console 에서 다음 명령 실행 : 인라인 스타일 삭제

```
document.getElementsByTagName('div')[0].style = "";
```



The screenshot shows the Chrome DevTools interface. The 'Elements' panel on the left displays the HTML structure: `<div style class="test over"></div> == $0`. A dashed arrow points from this line to the 'Styles' panel on the right. In the 'Styles' panel, the 'element.style' section is empty, indicating that the inline styles have been removed. Below it, the '.over' class rule is shown with the following properties: `background-color: aqua;`, `width: 300px;`, and `height: 100px;`. The '.test' class rule is also visible, with `background-color: yellow;` and `border: double 4px gray;`. The 'div' rule from the user agent stylesheet shows `display: block;`. At the bottom, the breadcrumb shows the path: `html > body > div.test.over`. A small orange box at the bottom right indicates the 'margin' property is set to '-'. The visual result of these changes is a cyan rectangle.

## 2. 인라인 스타일

### ❖ Vue에서는 인라인 스타일

- HTML에서와 마찬가지로 권장하지는 않지만 가끔 사용하기도 함

### ❖ 작성 방법

- v-bind:style + Javascript 객체
- CSS Style 속성이 아니라 Javascript 객체이므로 표기법이 다름
  - kebob-casing --> camelCasing

케밥 표기법(kebob casing)	카멜 표기법(camel casing)
font-size	fontSize
background-color	backgroundColor

- 자바스크립트가 camelCasing 사용

```
document.getElementById("a").style.fontSize='20pt';
```

## 2. 인라인 스타일

### ❖예제 06-02

```
<div id="app">
  <button id="a" :style="style1" @mouseover.stop="overEvent" @mouseout.stop="outEvent">테스트</button>
</div>
<script src="https://unpkg.com/vue"></script>
<script type="text/javascript">
  var vm = Vue.createApp({
    name: "App",
    data() {
      return {
        style1: { backgroundColor: "aqua", color: "black" },
      };
    },
    methods: {
      overEvent() {
        this.style1.backgroundColor = "purple";
        this.style1.color = "yellow";
      },
      outEvent() {
        this.style1.backgroundColor = "aqua";
        this.style1.color = "black";
      },
    },
  }).mount("#app");
</script>
```



## 2. 인라인 스타일

### ❖ 예제 06-03

- 속성을 하나씩 지정

[ 예제 06-02 ]

```
- 스타일 지정    --> :style="style1"
- 데이터       --> { style1 : { backgroundColor:"aqua", color:"black" } }
```

[ 예제 06-03 ]

```
- 스타일 지정    --> :style="{ backgroundColor:bgColor, color }"
- 데이터       --> { bgColor:"aqua", color:"black" }
```



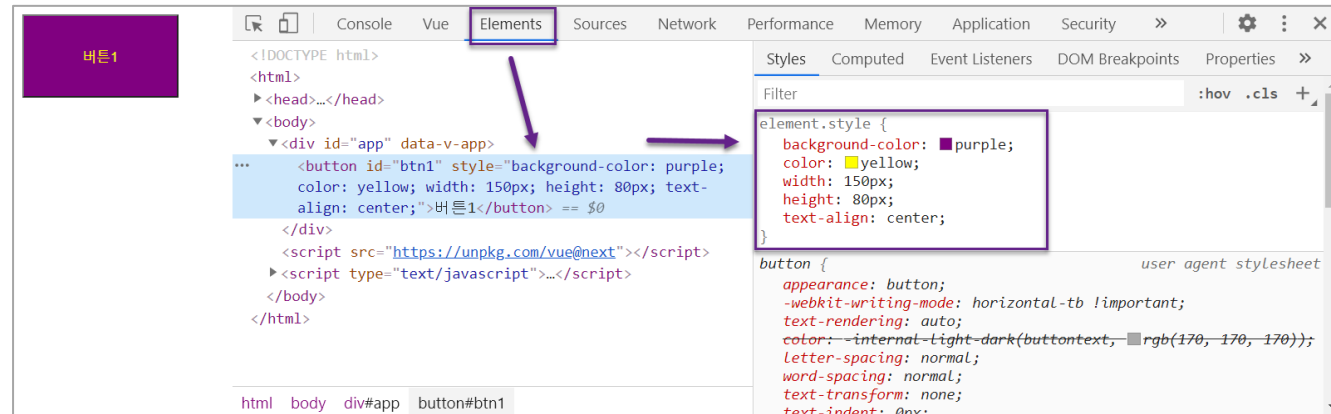
## 2. 인라인 스타일

### ❖ 예제 06-04

- 여러 개의 스타일 객체를 지정함

예제 06-04

```
01: <body>
02:   <div id="app">
03:     <button id="btn1" :style="[ myColor, myLayout ]">버튼1</button>
04:   </div>
05:   <script src="https://unpkg.com/vue"></script>
06:   <script type="text/javascript">
07:     var vm = Vue.createApp({
08:       name : "App",
09:       data() {
10:         return {
11:           myColor : { backgroundColor:'purple', color:'yellow' },
12:           myLayout : { width:'150px', height:'80px', textAlign:'center' }
13:         }
14:       }
15:     }).mount("#app")
16:   </script>
17: </body>
```



### 3. CSS 클래스 바인딩

#### ❖ CSS 클래스 지정 방법

- CSS 클래스 문자열을 바인딩하는 방법
- true/false 값을 가진 객체를 바인딩하는 방법

#### ❖ CSS 클래스 문자열 바인딩 : 예제 06-05

```
07:     .buttonColor { background-color:aqua; color:black; }
08:     .buttonLayout { text-align:center; width:120px; }
09:     .staticBorder { border: khaki dashed 1px; }

13:     <div id="app">
14:         <button class="staticBorder" :class="myColor">테스트 버튼</button>
15:     </div>

20:     data() {
21:         return {
22:             myColor : "buttonColor buttonLayout",
23:         }
24:     }
```

### 3. CSS 클래스 바인딩

#### ❖ 여러 개의 CSS클래스명 바인딩 : 예제 06-06

##### ■ 배열의 이용함

```
01: <body>
02:   <div id="app">
03:     <button class="staticBorder" :class="[myColor, myLayout]">
04:       테스트 버튼</button>
05:   </div>
06:   <script src="https://unpkg.com/vue"></script>
07:   <script type="text/javascript">
08:     var vm = Vue.createApp({
09:       name : "App",
10:       data() {
11:         return {
12:           myColor : "buttonColor",
13:           myLayout : "buttonLayout",
14:         }
15:       }
16:     }).mount("#app")
17:   </script>
18: </body>
```

### 3. CSS 클래스 바인딩

❖ 예제 06-06에서 조건에 따라 CSS 클래스를 바인딩하려면?

- 예제 06-07 : 삼항연산식을 이용한 방법

```
<div id="app">
  <input type="checkbox" v-model="isMyLayout" />레이아웃 적용 여부<br />
  <button class="staticBorder" :class="[myColor, isMyLayout ? myLayout : '']">테스트 버튼</button>
</div>
<script src="https://unpkg.com/vue"></script>
<script type="text/javascript">
  var vm = Vue.createApp({
    name: "App",
    data() {
      return {
        myColor: "buttonColor",
        myLayout: "buttonLayout",
        isMyLayout: false,
      };
    },
  }).mount("#app");
</script>
```

3항 연산식은  $a ? b : c$ 와 같이 표현되며,  $a$ 가 true이면  $b$ 를 리턴하고 false이면  $c$ 를 리턴하는 구조입니다. `isMyLayout`이 true이면 `myLayout`에 초기화된 클래스 문자열이 리턴되고, false이면 빈 문자열('')이 리턴되도록 합니다.

### 3. CSS 클래스 바인딩

#### ❖ true/false 값을 가진 객체를 바인딩하는 방법

- 조건에 따라 바인딩 여부를 결정할 CSS 클래스가 여러개라면?
  - 삼항연산식 사용 : 복잡하고 작성도 쉽지 않음

```
:class="[ isColor ? myColor : '', isLayout ? myLayout : '', isFont ? myFont : '' ]"
```

- 이럴 때 객체를 바인딩하는 방법
- 바인딩할 객체
  - CSS 클래스명을 속성명으로 사용
  - true/false 를 속성의 값으로 가짐

### 3. CSS 클래스 바인딩

#### ❖예제 06-08

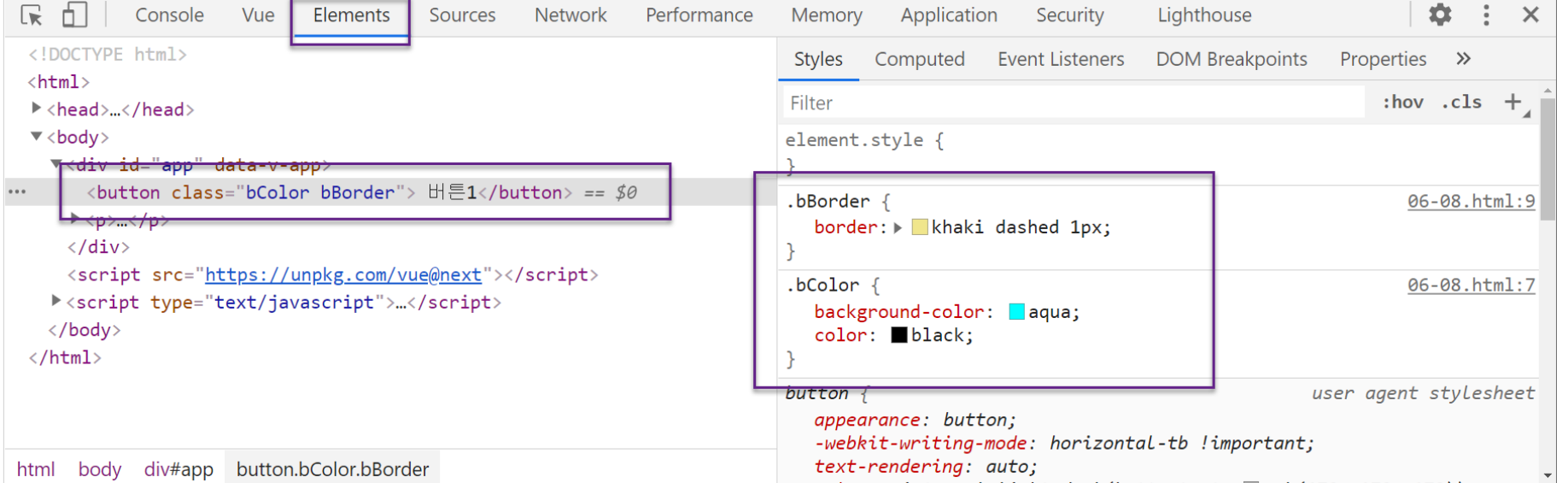
```
<div id="app">
  <button :class="{ bColor:setColor, bLayout:setAlign, bBorder:setBorder }">버튼1</button>
  <p>
    <input type="checkbox" v-model="setColor" value="true" />색상<br />
    <input type="checkbox" v-model="setAlign" value="true" />정렬,크기<br />
    <input type="checkbox" v-model="setBorder" value="true" />테두리선<br />
  </p>
</div>
<script src="https://unpkg.com/vue"></script>
<script type="text/javascript">
  var vm = Vue.createApp({
    name: "App",
    data() {
      return { setColor: false, setAlign: false, setBorder: false };
    },
  }).mount("#app");
</script>
```

### 3. CSS 클래스 바인딩

#### ❖ 예제 06-08 실행 결과

버튼1

- ☒ 색상
- ☐ 정렬 및 크기
- ☒ 테두리선



```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <div id="app" data-v-app>
      <button class="bColor bBorder"> 버튼1</button> == $0
    </div>
    <script src="https://unpkg.com/vue@next"></script>
    <script type="text/javascript">...</script>
  </body>
</html>
```

Styles

Filter :hov .cls +

element.style {

.bBorder {  
border: 1px dashed khaki;

.bColor {  
background-color: aqua;  
color: black;

button {  
appearance: button;  
-webkit-writing-mode: horizontal-tb !important;  
text-rendering: auto;



### 3. CSS 클래스 바인딩

#### ❖예제 06-09

- data 를 정의할 때 속성명을 일치시켜서 미리 객체를 작성하는 것이 바람직

```
<div id="app">
  <button :class="myStyle">버튼1</button>
  <p>
    <input type="checkbox" v-model="myStyle.bColor" value="true" />색상<br />
    <input type="checkbox" v-model="myStyle.bLayout" value="true" />정렬,크기<br />
    <input type="checkbox" v-model="myStyle.bBorder" value="true" />테두리선<br />
  </p>
</div>
<script src="https://unpkg.com/vue"></script>
<script type="text/javascript">
  var vm = Vue.createApp({
    name: "App",
    data() {
      return {
        myStyle: { bColor: false, bLayout: false, bBorder: false },
      };
    },
  }).mount("#app");
</script>
```



## 4. 동적 스타일 바인딩

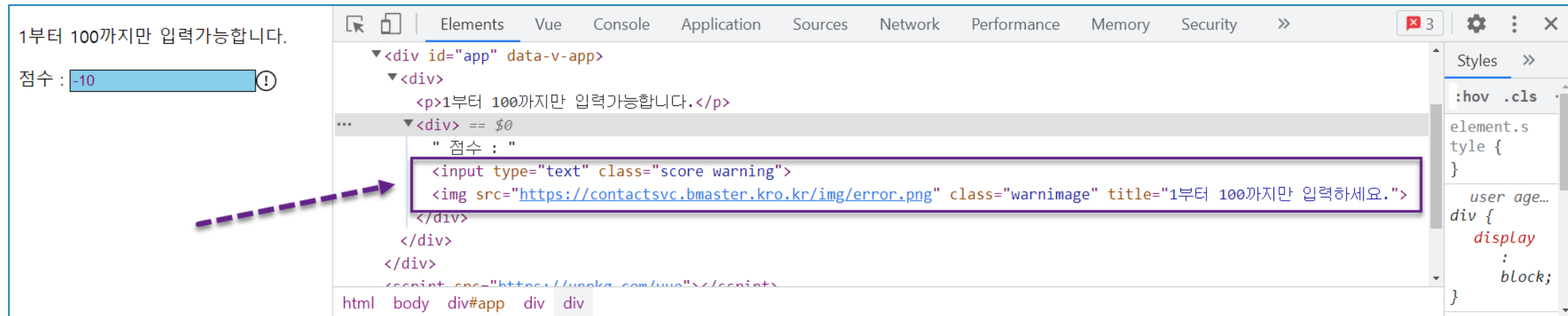
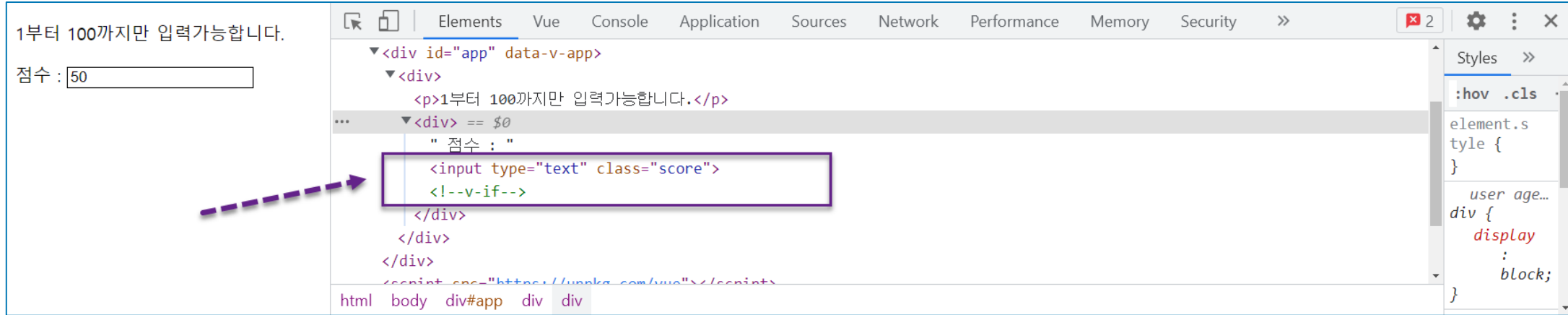
### ❖예제 06-10

- 계산된 속성과 메서드의 리턴값으로 CSS 클래스 지정
- 입력값이 올바른 범위에 있지 않을때 계산된 속성을 이용해 스타일 적용

```
<div id="app">
  <div>
    <p>1부터 100까지만 입력가능합니다.</p>
    <div>
      점수 : <input type="text" class="score" v-model.number="score" :class="info"/>
      
    </div>
  </div>
</div>
<script src="https://unpkg.com/vue"></script>
<script type="text/javascript">
  var vm = Vue.createApp({
    name: "App",
    data() {
      return { score: 50 };
    },
    computed: {
      info() {
        return { warning: this.score < 1 || this.score > 100 };
      },
    },
  }).mount("#app");
</script>
```

## 4. 동적 스타일 바인딩

### ❖예제 06-10 실행 결과



## 5. TodoList 예제

### ❖ 화면 시안

- Bootstrap@5 활용
- 예제 06-11~12

:: Todolist App

할일을 여기에 입력!

추가

~~할일1 (완료)~~

삭제

할일2

삭제

할일2

삭제

## 5. TodoList 예제

### ❖ 데이터와 메서드 정의

[ 데이터 ]	
todo	텍스트 박스에 사용자가 입력하는 내용을 받아내기 위한 data입니다.
todolist	추가한 todo들의 목록. todo 한 건은 다음과 같습니다.
id	todo 한 건의 고유 키. 이 예제에서는 timestamp를 이용합니다.
todo	todo 내용
completed	완료 여부(true, false)
[ 메서드 ]	
addTodo	텍스트 박스에 할 일(todo)을 입력하고 엔터를 누르거나 추가 버튼을 클릭하면 todolist에 새로운 todo를 추가합니다.
deleteTodo	삭제 버튼을 클릭하면 id를 이용해 할 일(todo)을 찾아서 삭제합니다.
toggleCompleted	할 일(todo) 한 건을 클릭하면 id를 이용해 completed 값을 토글합니다.

## 5. TodoList 예제

### ❖ 예제 06-13

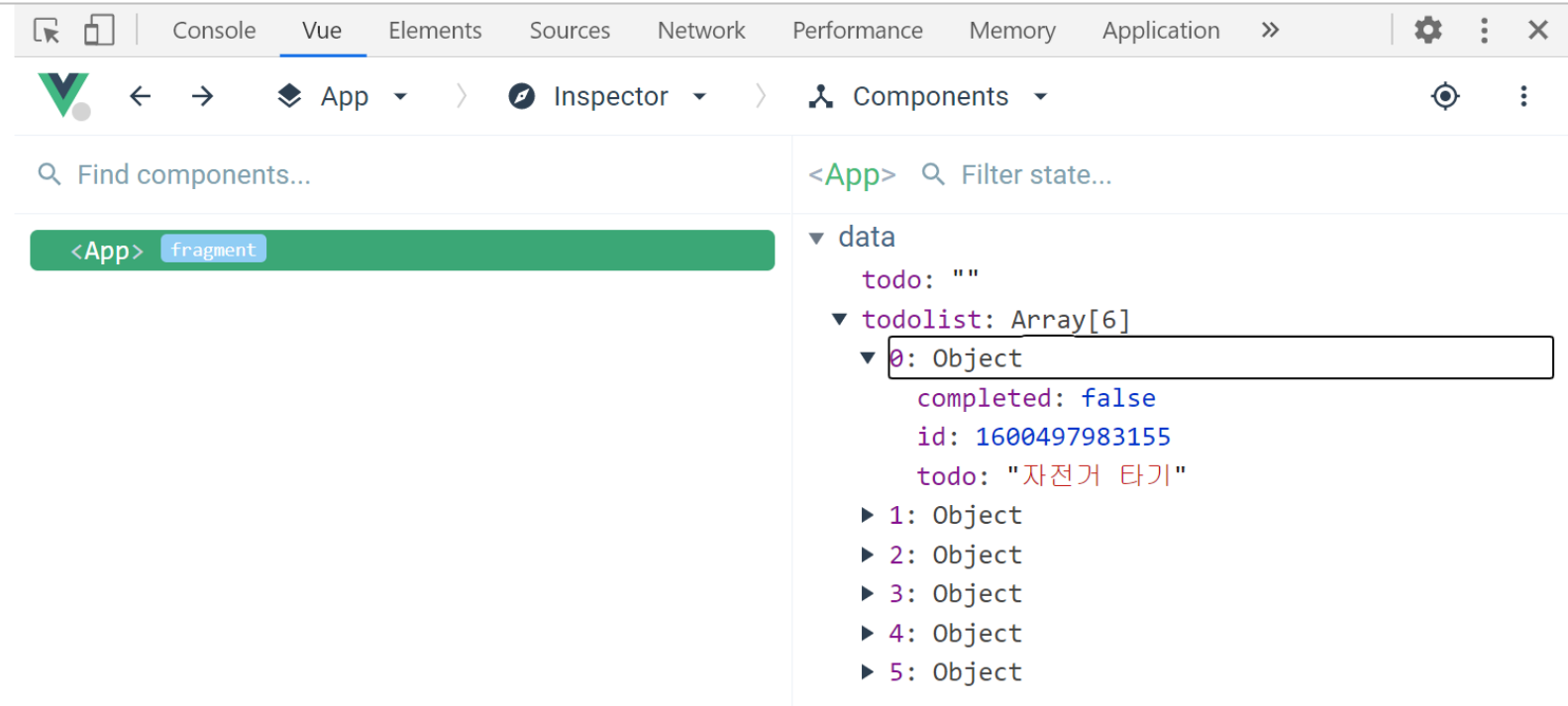
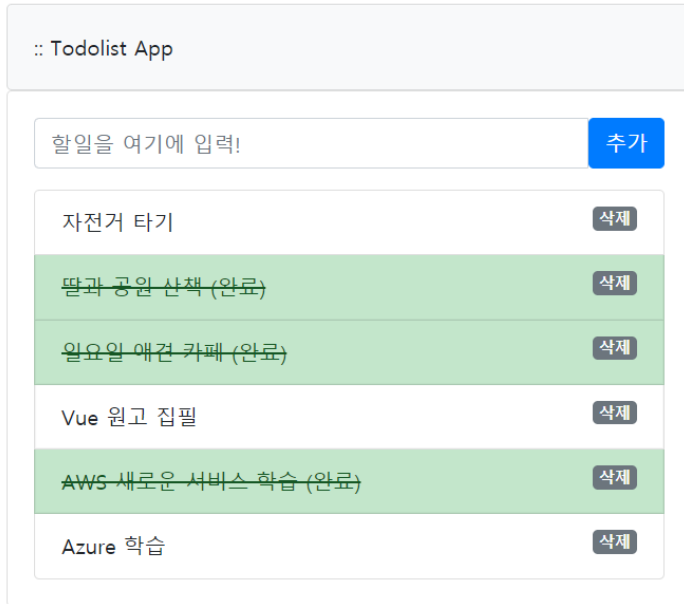
- Vue 인스턴스 생성
- Vue 인스턴스 내부에 data 옵션, methods 옵션 작성
- Javascript 배열의 메서드들 학습 필요
  - [https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global\\_Objects/Array](https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array)

### ❖ 예제 06-14

- 템플릿 작성
- {{ }}, v-bind, v-model, v-for 등의 디렉티브 활용
- CSS 클래스, 문자열을 동적으로 바인딩
  - 완료 여부 문자열
  - 완료된 할일의 스타일
- 삭제 이벤트에 .stop 수식어 지정
  - 지정하지 않으면 버블링이 일어나서 삭제후 토글을 시도함 --> 오류 발생

# 5. TodoList 예제

## ❖ 실행 결과



## 6. 마무리

지금까지 Vue 애플리케이션에서 스타일을 적용하는 방법들을 살펴보았습니다. 인라인 스타일 방법을 사용할 수도 있지만 유지 보수 측면이나 웹 퍼블리셔와의 협업 차원에서 CSS 클래스 바인딩 방법을 권장합니다.

또한 이제까지 3~6장의 학습한 내용을 이용하여 간단한 TodoList 앱 예제를 만들어 보았습니다. 예제 작성의 순서를 살펴보면 다음과 같습니다.

화면 시안 --> data 정의 --> 메서드 정의 --> 템플릿 작성

단순하게 템플릿, Vue 인스턴스, 이벤트 같은 개별적인 내용보다 data --> UI 라는 바인딩 개념을 이해하는 것이 더 중요하다고 필자는 생각하는데, 위의 예제 작성 순서는 data --> UI로의 진행 방향과 일치합니다. 템플릿 코드의 작성보다 화면 시안을 반영해 data와 data의 변경을 일으키는 메서드의 작성을 먼저 진행해야 합니다.

