

클래스 바인딩

클래스 바인딩

- 클래스 바인딩

HTML 요소의 클래스를 동적으로 바인딩하기 위한 방법으로 크게 3가지로 나뉨

1. 객체구문
2. 배열구문
3. 컴포넌트 사용

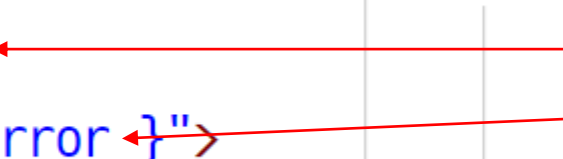
클래스 바인딩

• 객체 구문

- isActive가 true일 때 : 클래스명에 active 추가
- hasError가 true일 때 : 클래스명에 text-danger 추가

```
<div class="static"  
  v-bind:class="{ 'active': isActive,  
                  'text-danger': hasError }">  
</div>
```

```
var vm = new Vue({  
  el: '.static',  
  data: {  
    isActive: true,  
    hasError: false  
  }  
})
```

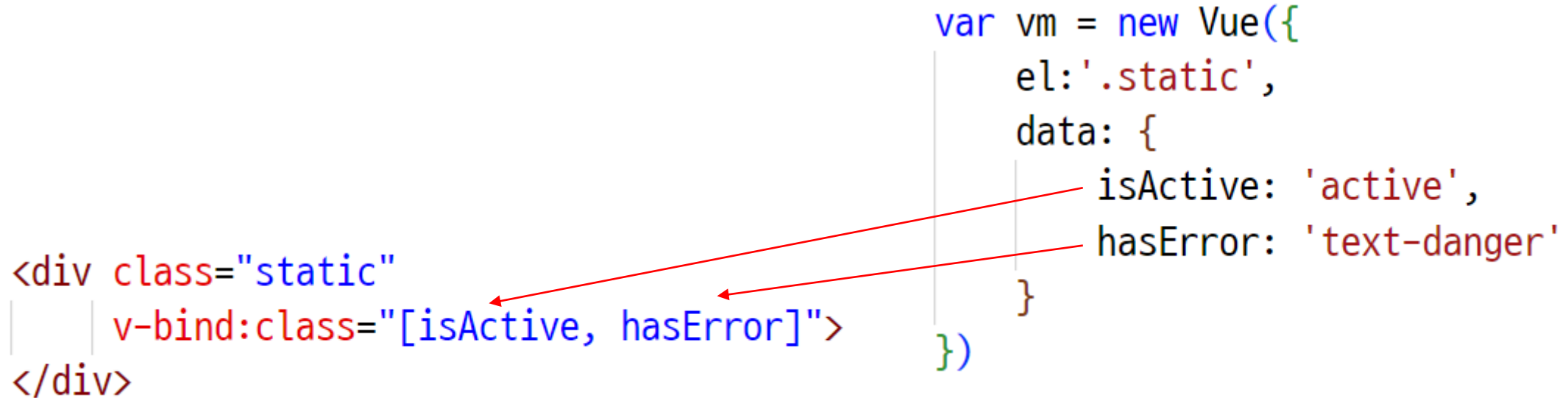


클래스 바인딩

• 배열 구문

vue 인스턴스의 data에 클래스로 사용할 값을 저장하고 배열 형태로 출력

```
<div class="static"  
  v-bind:class="[isActive, hasError]">  
</div>  
  
var vm = new Vue({  
  el: '.static',  
  data: {  
    isActive: 'active',  
    hasError: 'text-danger'  
  }  
})
```



클래스 바인딩

• 컴포넌트 사용

컴포넌트를 따로 사용하여 템플릿의 내용을 HTML태그 안에 출력

* 기존에 있던 클래스(Ex. static)는 유지됨

```
<my-component class="static"></my-component>
```

```
Vue.component('my-component', {
  template: '<p class="foo bar"> Hi </p>'
})
```

```
var vm = new Vue({
  el: '.static'
})
```

인라인 스타일 바인딩

인라인 스타일 바인딩

- 인라인 스타일 바인딩

v-bind:style을 사용해 인라인 방식으로 스타일을 변경할 수 있음

객체 구문과 배열 구문으로 사용 가능

인라인 스타일 바인딩

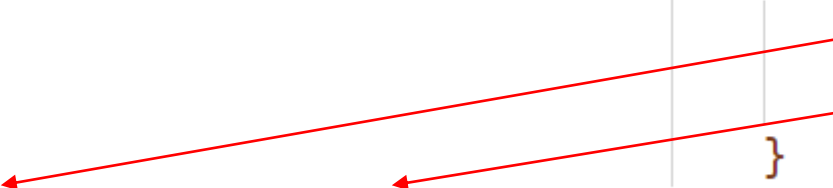
• 객체 구문

static 클래스를 찾아가서 activeColor와 fontSize에 데이터를 넣음

* 폰트사이즈의 경우 아래와 같이 v-bind:style 내에 + 'px'로 붙여줘도 되고
data안의 fontSize에 '50px'과 같이 사용해도 가능함

```
<div class="static"  
  v-bind:style="{color: activeColor, fontSize: fontSize + 'px'}">  
  aaa  
</div>
```

```
var vm = new Vue({  
  el: '.static',  
  data: {  
    activeColor: 'red',  
    fontSize: 50  
  }  
})
```



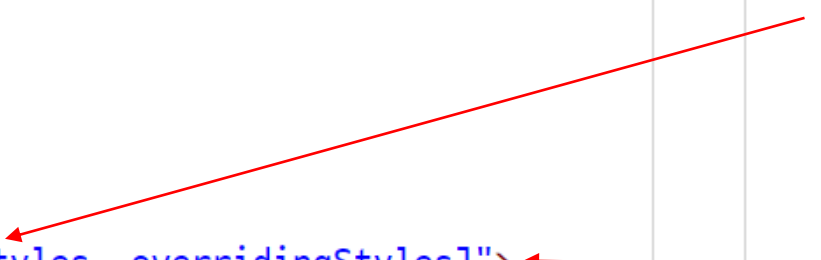
인라인 스타일 바인딩

• 배열 구문

vue 인스턴스의 data에 클래스로 사용할 값을 저장하고 배열 형태로 사용

```
<div id="app">
  <div v-bind:style="[baseStyles, overridingStyles]">
    Test
  </div>
</div>
```

```
var vm = new Vue({
  el: '#app',
  data: {
    baseStyles: {
      color: 'red',
      fontSize: '50px'
    },
    overridingStyles: {
      backgroundColor: 'blue',
      fontWeight: 'bold'
    }
  }
});
```



폼 입력 바인딩

폼 입력 바인딩

• 폼 입력 바인딩

v-model 디렉티브를 사용하여 input, textarea 등의 엘리먼트에 양방향 데이터 바인딩을 생성할 수 있음

• 양방향 데이터 바인딩

두 개의 데이터 정보를 일치 시키는 방법으로 여태 했던 방법들은 인스턴스에서 데이터를 넘겨주기만 했으므로 단방향 바인딩
단방향 바인딩 : Vue 인스턴스 → 템플릿

양방향 바인딩은 인스턴스와 템플릿의 데이터가 서로 접근이 가능한 상태
양방향 바인딩 : Vue 인스턴스 ↔ 템플릿

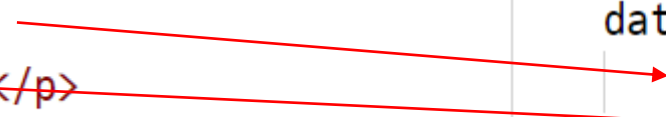
폼 입력 바인딩

- input

input 태그에 입력된 값을 Vue 인스턴스의 message에 데이터를 넣고
p태그에서 {{ }}를 사용하여 message에 있는 값을 가져옴

```
<div id="app">
  <input v-model="message">
  <p>메시지 : {{ message }}</p>
</div>
```

```
var vm = new Vue({
  el: '#app',
  data: {
    message: ''
  }
});
```



The diagram consists of two red arrows. The first arrow originates from the `message` property within the `data` object of the Vue instance and points to the `message` attribute in the `v-model` directive of the `<input>` tag. The second arrow originates from the `message` property access within the `{{ message }}` interpolation in the `<p>` tag and points to the same `message` property within the `data` object of the Vue instance.

폼 입력 바인딩

- checkbox

체크박스를 클릭하면 value값이 checkedNames의 배열에 들어감

```
<div id="app">
  <input type="checkbox" id="dog" value="dog" v-model="checkedNames">
  <label for="dog">강아지</label>
  <input type="checkbox" id="cat" value="cat" v-model="checkedNames">
  <label for="cat">고양이</label>
  <input type="checkbox" id="pig" value="pig" v-model="checkedNames">
  <label for="pig">돼지</label>
  <br>
  <span>체크한 동물 : {{ checkedNames }}</span>
</div>
```

```
var vm = new Vue({
  el: '#app',
  data: {
    checkedNames: []
  }
});
```

폼 입력 바인딩

- select

체크박스를 클릭하면 value값이 checkedNames의 배열에 들어감

```
<div id="app">
  <select v-model="selected">
    <option value="">동물을 선택 해주세요.</option>
    <option>강아지</option>
    <option>고양이</option>
    <option>돼지</option>
  </select>
  <br>
  <span>동물: {{ selected }}</span>
</div>
```

```
var vm = new Vue({
  el: '#app',
  data: {
    selected: ''
  }
});
```

폼 입력 바인딩

• checkbox 값 바인딩

true-value : 박스가 체크 되었을 때 들어갈 값

false-value : 박스가 체크되지 않았을 때 들어갈 값

isChecked : toggle의 값이 yes일 때 true 반환 / 아닐 경우 false

isChecked : toggle의 값이 no일 때 true 반환 / 아닐 경우 false

```
<div id="app">
  <input type="checkbox" v-model="toggle"
    true-value="yes" false-value="no">
  <p>체크된 경우: {{ isChecked }}</p>
  <p>체크되지 않은 경우: {{ isChecked }}</p>
</div>
```

```
var vm = new Vue({
  el: '#app',
  data: {
    toggle: 'no'
  },
  computed: {
    isChecked: function() {
      return this.toggle === 'yes';
    },
    isChecked: function() {
      return this.toggle === 'no';
    }
  }
});
```