클래스와 스타일 바인딩

일반적으로 엘리먼트에 데이터를 바인딩하는 이유는 클래스 목록과 해당 인라인 스타일을 조작하기 위함입니다. `class` , `style` 둘 다 속성이므로 다른 속성과 마찬가지로 `v-bind` 를 사용하여 문자열 값을 동적으로 할당할 수 있습니다. 그러나 연결된 문자열을 사용하여 이러한 값을 생성하려고 하면 성가시고 오류가 발생하기 쉽습니다. 이러한 이유로 Vue는 `v-bind` 가 `class` 및 `style` 과 함께 사용될 때 특별한 향상을 제공합니다. 문자열 외에도 표현식은 객체 또는 배열로 평가될 수 있습니다.

HTML 클래스 바인딩

Vue School의 무료 동영상 강의 보기

객체로 바인딩 하기

클래스를 동적으로 토글하기 위해 객체를 `:class` (`v-bind:class` 의 줄임말)에 전달할 수 있습니다:

template

```
<div :class="{ active: isActive }"></div>
```

위의 구문은 `isActive` 데이터 속성의 **truthiness**에 의해 `active` 클래스의 존재 여부가 결정됨을 의미합니다.

객체에 더 많은 필드를 사용하여 여러 클래스를 토글할 수 있습니다. 또한 `:class` 디렉티브는 일반 `class` 속성과 공존할 수도 있습니다. 따라서 다음과 같은 상황이 주어지고:

js

```
const isActive = ref(true)
const hasError = ref(false)
```

아래와 같이 템플릿이 구성되 있다면:

template

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

다음과 같이 렌더링 됩니다:

template

```
<div class="static active"></div>
```

`isActive` 또는 `hasError` 가 변경되면 그에 따라 클래스 목록이 업데이트됩니다. 예를 들어 `hasError` 가 `true` 가 되면 클래스 목록은 `"static active text-danger"` 가 됩니다.

바인딩된 객체는 인라인일 필요가 없습니다:

js

```
const classObject = reactive({
  active: true,
  'text-danger': false
})
```

template

```
<div :class="classObject"></div>
```

그러면 다음이 렌더링됩니다:

template
```
<div class="active"></div>
```

객체를 반환하는 계산된 속성에 바인딩할 수도 있습니다. 이는 일반적이고 강력한 패턴입니다:

js
```
const isActive = ref(true)
const error = ref(null)

const classObject = computed(() => ({
  active: isActive.value && !error.value,
  'text-danger': error.value && error.value.type === 'fatal'
}))
```

template
```
<div :class="classObject"></div>
```

배열로 바인딩 하기

:class 를 배열로 바인딩하여 클래스 목록을 적용할 수 있습니다:

js
```
const activeClass = ref('active')
const errorClass = ref('text-danger')
```

template
```
<div :class="[activeClass, errorClass]"></div>
```

다음과 같이 렌더링 됩니다:

template
```
<div class="active text-danger"></div>
```

삼항 표현식을 사용하여 목록 내 클래스도 토글할 수 있습니다:

template
```
<div :class="[isActive ? activeClass : '', errorClass]"></div>
```

위 코드는 errorClass 를 항상 적용하지만, activeClass 는 isActive 가 truthy 일 때만 적용됩니다.

그러나 조건부 클래스가 여러 개인 경우 다소 장황할 수 있습니다. 이러한 이유로 배열 구문 내에서 객체 구문을 사용할 수도 있습니다:

template
```
<div :class="[{ activeClass: isActive }, errorClass]"></div>
```

컴포넌트에서 사용하기

이 섹션은 컴포넌트에 대한 지식이 있다고 가정하므로, 건너뛰고 나중에 읽어도 됩니다.

최상위(root) 엘리먼트가 하나로 구성된 컴포넌트에서 class 속성을 사용하면, 해당 클래스가 컴포넌트의 루트 엘리먼트에 이미 정의된 기존 클래스와 병합되어 추가됩니다.

MyComponent 라는 컴포넌트의 템플릿이 아래와 같이 구성되어 있다고 가정:

template
```
<!-- 자식 컴포넌트의 템플릿 -->
<p class="foo bar">안녕!</p>
```

그런 다음 사용할 때 몇 가지 클래스를 추가합니다:

```template
<!-- 컴포넌트가 사용될 때 -->
<MyComponent class="baz boo" />
```

다음과 같이 렌더링 됩니다:

```template
<p class="foo bar baz boo">안녕!</p>
```

클래스 바인딩도 마찬가지입니다:

```template
<MyComponent :class="{ active: isActive }" />
```

`isActive` 가 truthy이면 렌더링된 HTML은 다음과 같습니다:

```template
<p class="foo bar active">안녕!</p>
```

여러 개의 최상위 엘리먼트로 컴포넌트가 구성되어 있는 경우, 클래스를 적용할 엘리먼트를 정의해야 합니다. `$attrs` 컴포넌트 속성을 사용하여 이 작업을 수행할 수 있습니다.

```template
<!-- MyComponent 템플릿에서 $attrs 속성을 사용 -->
<p :class="$attrs.class">안녕!</p>
<span>반가워!</span>
```

```template
<MyComponent class="baz" />
```

다음과 같이 렌더링 됩니다:

```html
<p class="baz">Hi!</p>
<span>반가워!</span>
```

컴포넌트 속성 상속에 대한 자세한 내용은 폴스루 속성 섹션에서 확인할 수 있습니다.

인라인 스타일 바인딩

객체로 바인딩

`:style` 은 **HTML** 엘리먼트의 **`style`** 속성에 해당하는 JavaScript 객체에 대한 바인딩을 지원합니다:

```js
const activeColor = ref('red')
const fontSize = ref(30)
```

```template
<div :style="{ color: activeColor, fontSize: fontSize + 'px' }"></div>
```

`:style` 에 사용될 CSS 속성에 해당하는 키 문자열은 camelCase가 권장되지만, kebab-cased(실제 CSS에서 사용되는 방식)도 지원합니다. 예를 들면 다음과 같습니다:

template

```
<div :style="{ 'font-size': fontSize + 'px' }"></div>
```

템플릿이 더 깔끔해지도록 스타일 객체를 직접 바인딩하는 것이 좋은 방법입니다:

js

```
const styleObject = reactive({
  color: 'red',
  fontSize: '30px'
})
```

template

```
<div :style="styleObject"></div>
```

일반적으로 인라인 스타일에 바인딩 하는 경우, 객체를 반환하는 계산된 속성을 사용합니다.

배열로 바인딩 하기

스타일 객체 여러 개로 이루어진 배열을 :style 에 바인딩할 수 있습니다. 객체들은 병합되어 엘리먼트에 적용됩니다:

template

```
<div :style="[baseStyles, overridingStyles]"></div>
```

접두사 자동완성

Vue가 실행되고 있을 때, 해당 브라우저에서 지원되지 않는 CSS 속성이 :style 에 사용되면, 자동으로 해당 속성과 벤더 접두사가 조합된 여러 개의 특수한 속성을 테스트하고 지원되는 속성을 찾아서 추가합니다.

다중 값

스타일 속성에 다중 값을 배열로 제공할 수 있습니다. 예를 들면 다음과 같습니다:

template

```
<div :style="{ display: ['flex', '-webkit-box', '-ms-flexbox'] }"></div>
```

이 경우, 브라우저가 지원하는 배열 내 마지막 값을 렌더링합니다. 이 예제에서 브라우저가 flex 와 -webkit-box 속성만 지원한다면, flex 라는 표준 속성 값이 있음에도 display: -webkit-box 를 렌더링 합니다.