컴포넌트 스타일 적용 방법
1. 전역 CSS
   CSS 충돌
2. 범위 CSS(scoped CSS)
   <style scoped></style>
   느리다
   하위컴포넌트에 적용된다.
3. 모듈 CSS(module CSS)
   <style module></style>
   $style 로 사용
   v-bind:class="$style.hand"
   v-bind:class="[$style.hand,$style.border]"

9.2 컴포넌트에서의 스타일

컴포넌트의 스타일은 〈style〉 태그 내에 작성한다는 것은 이전 절에서 이미 다루었습니다만 이것은 전역 스타일이므로 페이지 전체에서 사용됩니다. 그러므로 다른 컴포넌트에서도 동일한 CSS 클래스명을 사용한다면 충돌이 발생합니다. 특정 컴포넌트만의 스타일을 지정하려면 범위 CSS(Scoped CSS)와 CSS 모듈(CSS Module)의 두 가지 방법을 사용할 수 있습니다.

9.2.1 범위 CSS(Scoped CSS)

우선 범위 CSS를 확인하기 위해 간단한 예제를 작성해보도록 합니다. 9장 예제를 실행해 디렉터리 안에서 Vue CLI의 기본 프리셋(default preset)을 이용해 styletest 프로젝트를 생성하겠습니다.

```
vue create styletest
cd styletest
```

프로젝트가 생성되면 src/components/HelloWorld.vue 파일을 삭제합니다. Child1.vue, Child2.vue라는 스타일 기능만을 테스트할 간단한 컴포넌트를 작성합니다. VSCode를 실행한 후 styletest 디렉터리를 열고 src/components 디렉터리 아래에 두개의 컴포넌트 파일을 작성하면 됩니다. Child2.vue 컴포넌트는 Child1.vue와 볼드체로 표현된 부분만 다릅니다.

[예제 09-09 : src/components/Child1.vue]

```
<template>
  <div class="main">{{msg}}</div>
</template>
<script>
export default {
  name: 'child1',
  data () {
    return {
      msg: 'Child1'
    }
  }
}
</script>
<style>
.main { border:solid 1px black; background-color:yellow; }
</style>
```

[예제 09-10 : src/components/Child2.vue]

```
<template>
  <div class="main">{{msg}}</div>
</template>
<script>
export default {
  name: 'child2',
  data () {
    return {
      msg: 'Child2'
    }
  }
}
</script>
<style>
.main { border:solid 1px black; background-color:aqua; }
</style>
```

두 컴포넌트의 스타일에는 서로 다른 배경색(background-color) 속성을 가진 main 클래스가 정의되어 있습니다. 하지만 이 상태로는 둘다 전역 클래스이기 때문에 충돌합니다. 충돌 여부를 확인하기 위해 App.vue를 수정합니다. App.vue의 〈template /〉 내부와 〈script /〉 내부의 코드를 예제 09-11의 볼드체로 표시된 부분으로 변경합니다.

[예제 09-11 : src/App.vue]

```
01: <template>
02:   <div id="app">
03:     <h2>{{msg}}</h2>
04:     <child1 />
05:     <child2 />
06:   </div>
07: </template>
08:
09: <script>
10: import Child1 from './components/Child1.vue'
11: import Child2 from './components/Child2.vue'
12:
13: export default {
14:   name: 'app',
15:   components : { Child1, Child2 },
16:   data () {
17:     return {
18:       msg: 'Welcome to Your Vue.js App'
19:     }
20:   }
21: }
22: </script>
23:
24: <style>
25: ......
26: </style>
```

yarn serve 명령어를 이용해 작성된 예제를 실행한 후 그림 09-03과 같이 브라우저의 개발자 도구를 열어서 결과를 확인해보세요.

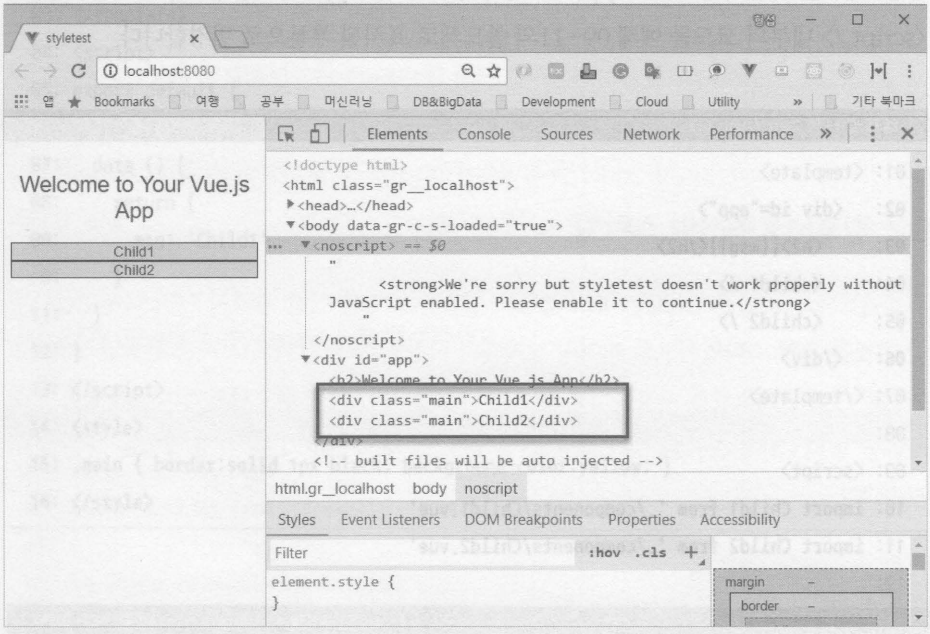

그림 09-03 예제 09-11 실행 결과

두 컴포넌트에 동일한 클래스가 적용되었음을 알 수 있습니다. 두 스타일이 모두 전역으로 정의되었기 때문입니다. 동일한 이름의 클래스로 두 개가 등록된 경우 등록된 순서로 마지막에 등록된 스타일만이 적용됩니다. 그 결과 Child1, Child2에 동일한 색상이 적용되었을 것입니다. 자 이제 범위 CSS로 변경해봅시다. 의외로 단순합니다. Child1.vue와 Child2.vue 안의 〈style〉 태그를 〈style scoped〉로 변경하기만 하면 됩니다. 변경한 후의 실행 결과는 다음과 같습니다.

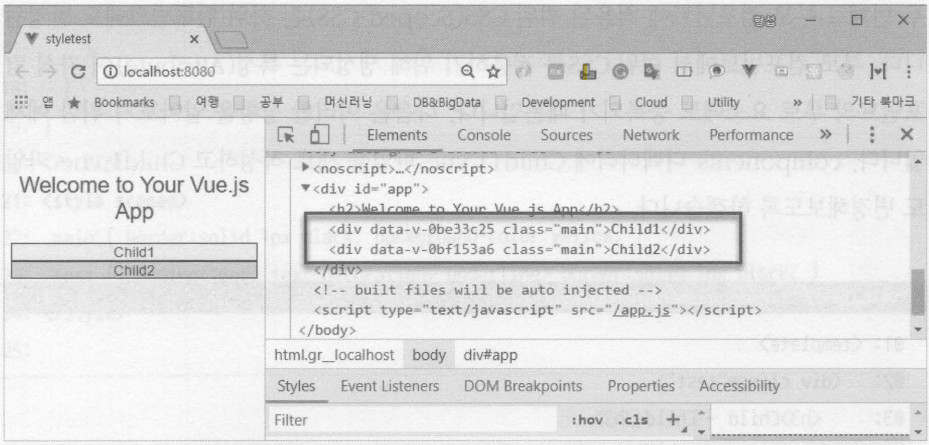

그림 09-04 범위 CSS

범위 CSS를 적용한 결과를 살펴보면 컴포넌트 요소에 data-v-xxxxx 형태의 특성(Attribute)이 부여된 것을 볼 수 있습니다. 스타일 클래스에도 이와 관련한 내용이 설정되어 있습니다.

```
<style type="text/css">
.main[data-v-0be33c25] { border:solid 1px black; background-color:yellow;
}
</style>
```

그림 09-05

하나의 컴포넌트에는 여러 개의 〈style〉 태그를 작성할 수 있습니다. 그러므로 전역 CSS와 범위 CSS를 구분해서 작성할 수 있습니다.

범위 CSS를 사용할 때는 몇 가지 사항을 기억하고 있어야 합니다.

첫 번째로 범위 CSS는 특성 선택자(Attribute Selector)를 사용하기 때문에 브라우저에서 스타일을 적용하는 속도가 느립니다. 그렇기 때문에 반드시 속도가 빠른 ID 선택자, 클래스 선택자, 태그명 선택자로 요소를 선택해 스타일을 적용하도록 해야 합니다. 그래야만 스타일 적용으로 인한 브라우저의 실행 속도가 느려지는 문제를 보완할 수 있습니다.

두 번째로 부모 컴포넌트에 적용된 범위 CSS(Scoped CSS)는 하위 컴포넌트에도 반영됩니다. 부모 컴포넌트에서 범위 CSS를 적용하기 위해 생성되는 특성(Attribute)이 자식 컴포넌트의 루트 요소에도 등록되기 때문입니다. 다음은 이러한 상황을 알아보기 위한 예제입니다. components 디렉터리에 Child11.vue 파일을 새로 작성하고 Child1.vue 파일도 변경해보도록 하겠습니다.

[예제 09-12 : src/components/Child11.vue]

```
<template>
  <div class="test">
    <h3>Child - Child</h3>
  </div>
</template>

<style scoped>
  .test { font-style:italic; }
</style>
```

[예제 09-13 : src/components/Child1.vue 변경]

```
<template>
  <div class="main test">
    {{msg}}
    <child11 />
    {{msg}}
  </div>
</template>
<script>
import Child11 from './Child11.vue'

export default {
  name: 'child1',
  components : { Child11 },
  data () {
    return {
```

```
16:         msg: 'Child1'
17:       }
18:     }
19: }
20: </script>
21: <style scoped>
22: .main { border:solid 1px black; background-color:yellow; }
23: .test { padding:10px; text-decoration: underline; border:solid 1px black; }
24: </style>
25:
```

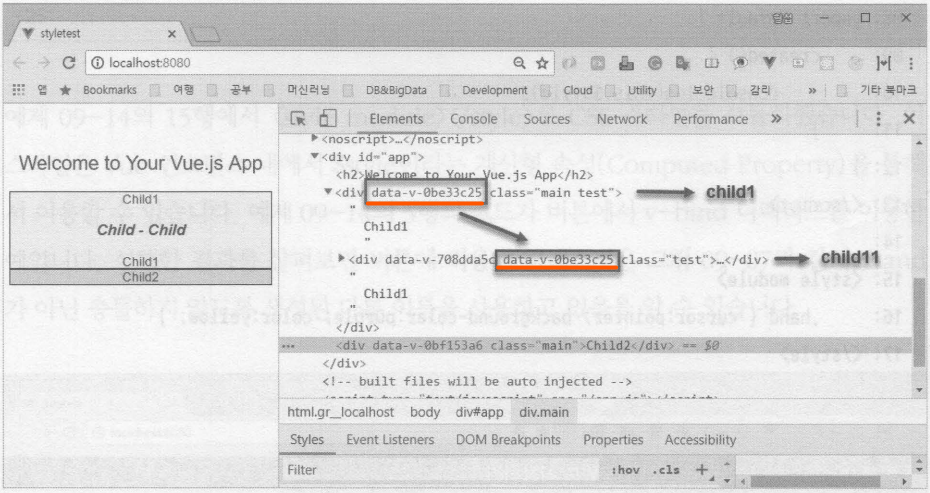

그림 09-06 예제 09-13 실행 결과

그림 09-06에서 확인할 수 있듯이 Child1 컴포넌트에 적용된 범위 CSS 스타일을 위한 특성(data-v-0be33c25)이 자식 컴포넌트인 Child11의 루트 요소에 반영된 것을 볼 수 있습니다.

9.2.2 CSS 모듈(CSS Module)

CSS 모듈은 CSS 스타일을 마치 객체처럼 다룰 수 있게 합니다. 설정하는 방법은 간단합니다. 〈style module〉〈/style〉과 같이 사용하면 됩니다. 예제를 통해서 살펴보겠습니다.

styletest 프로젝트의 components 디렉터리 아래에 새로운 컴포넌트(Module1.vue)를 하나 추가하고 App.vue에서 사용하도록 설정합니다.

[예제 09-14 : src/components/Module1.vue]

```
<template>
  <div>
    <button :class="$style.hand"> CSS Module을 적용한 버튼 </button>
  </div>
</template>

<script>
export default {
    created() {
        console.log(this.$style);
    }
}
</script>

<style module>
    .hand { cursor:pointer; background-color:purple; color:yellow; }
</style>
```

[예제 09-15 : src/App.vue에서 컴포넌트 사용하기]

```
<template>
  <div id="app">
    <h2>{{msg}}</h2>
    <child1 />
    <child2 />
    <module1 />
  </div>
</template>

<script>
import Child1 from './components/Child1.vue'
import Child2 from './components/Child2.vue'
```

```
13: import Module1 from './components/Module1.vue'
14:
15: export default {
16:   name: 'app',
17:   components : { Child1, Child2, Module1 },
18:   data () {
19:     return {
20:       msg: 'Welcome to Your Vue.js App'
21:     }
22:   }
23: }
24: </script>
25: ......
```

예제 09-14의 15행에서 〈style module〉〈/style〉로 CSS 스타일을 모듈화했습니다. 이 스타일은 Vue 인스턴스 내에서 $style이라는 계산형 속성(Computed Property)을 통해서 이용할 수 있습니다. 예제 09-14의 3행의 코드가 버튼에서 v-bind 디렉티브를 이용한 예입니다. 실행한 결과를 살펴보면 버튼에 적용된 클래스명은 그림 09-07과 같이 .hand가 아닌 충돌하지 않도록 생성된 다른 이름을 사용하고 있음을 알 수 있습니다.

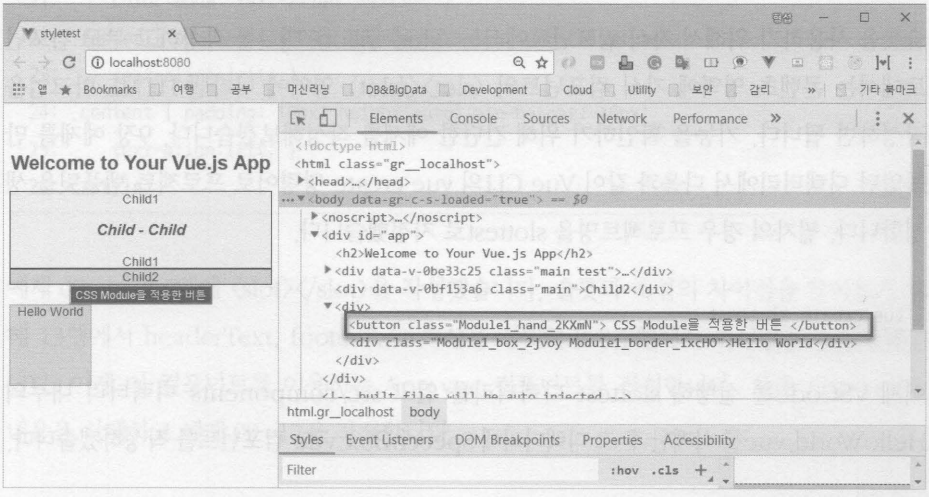

그림 09-07 예제 09-14, 15 실행 결과

적용해야 할 CSS 클래스가 여러 개라면 배열 문법을 이용해 한 번에 적용할 수 있습니다.

```
<div v-bind:class="[$style.box, $style.border]">Hello World</div>
```