KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

컴포넌트 심화

[KB] IT's Your Life

KB 국민은행

1 단일 파일 컴포넌트에서의 스타일

전역 css

- src/main.js에서 임포트한 CSS 스타일
- 페이지 전체에 적용

- 다른 컴포넌트의 스타일과 충돌 피하기
  - 특정 컴포넌트만의 스타일 지정
  - 범위 CSS(Scoped CSS)
  - CSS 모듈

1 단일 파일 컴포넌트에서의 스타일

범위 CSS

- 컴포넌트가 렌더링하는 요소에 특성 기반의 추가적인 식별자를 부여

→ 충돌 회피

프로젝트 만들기

```
npm init vue style-test
cd style-test
npm install
```

- src/components 내 기존 파일 삭제
- 다음 컴포넌트 3개 추가
  - Child1.vue
  - Child2.vue
  - Child3.vue

src/components/Child1.vue ~ Child3.vue

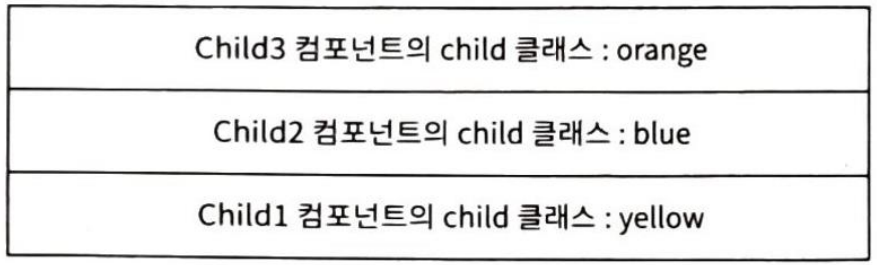
```
<template>
  <div class="child">
    <h2>Child1</h2>
  </div>
</template>

<script>
export default {
  name: 'Child1',
};
</script>

<style>
.child {
  background-color: yellow;   /* Child2는 blue, Child3는 orange */
  border: solid 1px black;
  margin: 1.5em;
  padding: 1em;
}
</style>
```

src/App.vue

```
<template>
  <div>
    <Child1 />
    <Child2 />
    <Child3 />
  </div>
</template>

<script>
import Child1 from './components/Child1.vue';
import Child2 from './components/Child2.vue';
import Child3 from './components/Child3.vue';

export default {
  name: 'App',
  components: { Child1, Child2, Child3 },
};
</script>
```

```
//yellow
import Child1 from './components/Child1';
//blue
import Child2 from './components/Child2';
//orange
import Child3 from './components/Child3';
```

import 순서



스타일 적용 순서

1 단일 파일 컴포넌트에서의 스타일

Child1

Child2

Child3

1 단일 파일 컴포넌트에서의 스타일

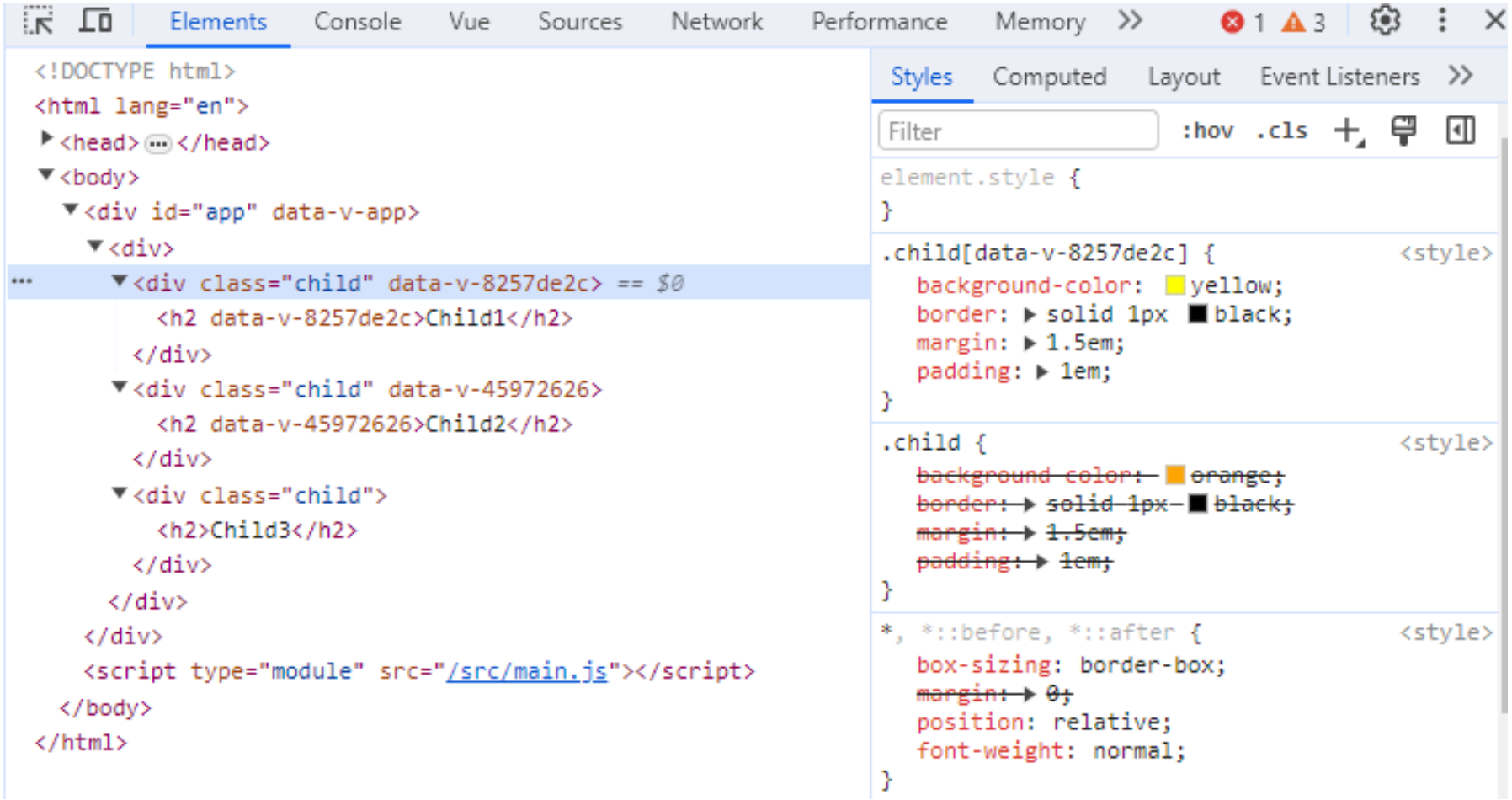
<style scoped>

- Child1, Child2의 <style>을 <style scoped>로 변경

Child1

Child2

Child3



1 단일 파일 컴포넌트에서의 스타일

CSS 모듈

- CSS 스타일을 마치 객체처럼 처리
- <style module>로 지정

src/components/Child3.vue

```
<template>
  <div :class="$style.child">
    <h2>Child3</h2>
  </div>
</template>

<script>
export default {
  name: 'Child3',
  created() {
    console.log(this.$style);
  },
};
</script>

<style module>
.child {
  background-color: orange;
  border: solid 1px black;
  margin: 1.5em;
  padding: 1em;
}
</style>
```

슬롯(slot)

- 부모 컴포넌트와 자식 컴포넌트 사이의 정보 교환 방법

→ props와 이벤트

- 부모 컴포넌트에서 자식 컴포넌트로 템플릿 정보를 전달하는 방법

→ 슬롯

슬롯 사용 전의 컴포넌트

프로젝트 생성

npm init vue slot-test
cd slot-test
npm install

컴포넌트

- CheckBox1.vue
- NoSlotTest.vue

src/components/CheckBox1.vue

```
<template>
    <div>
        <input type="checkbox"
           :value="id"
           :checked="checked"
           @change="$emit('check-changed', {id, checked: $event.target.checked })" />
        <span v-if="checked === true" style="color:blue; text-decoration:underline;">
            <i>{{label}}</i>
        </span>
        <span v-else style="color:gray">{{label}}</span>
    </div>
</template>

<script>
export default {
    name : "CheckBox1",
    props : ["id", "checked", "label"]
}
</script>
```

src/components/NoSlotTest.vue

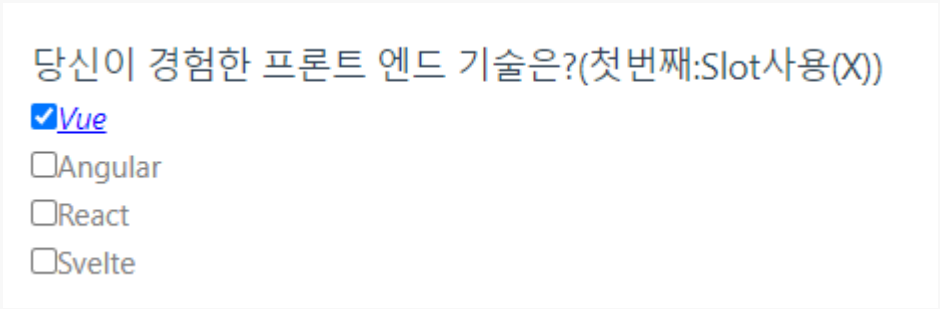
```
<template>
    <div>
      <h3>당신이 경험한 프론트 엔드 기술은?(첫번째:Slot사용(X))</h3>
      <CheckBox1 v-for="item in items" :key="item.id"
         :id="item.id"
         :label="item.label"
         :checked="item.checked"
         @check-changed="CheckBoxChanged">
      </CheckBox1>
    </div>
</template>
```

src/components/NoSlotTest.vue

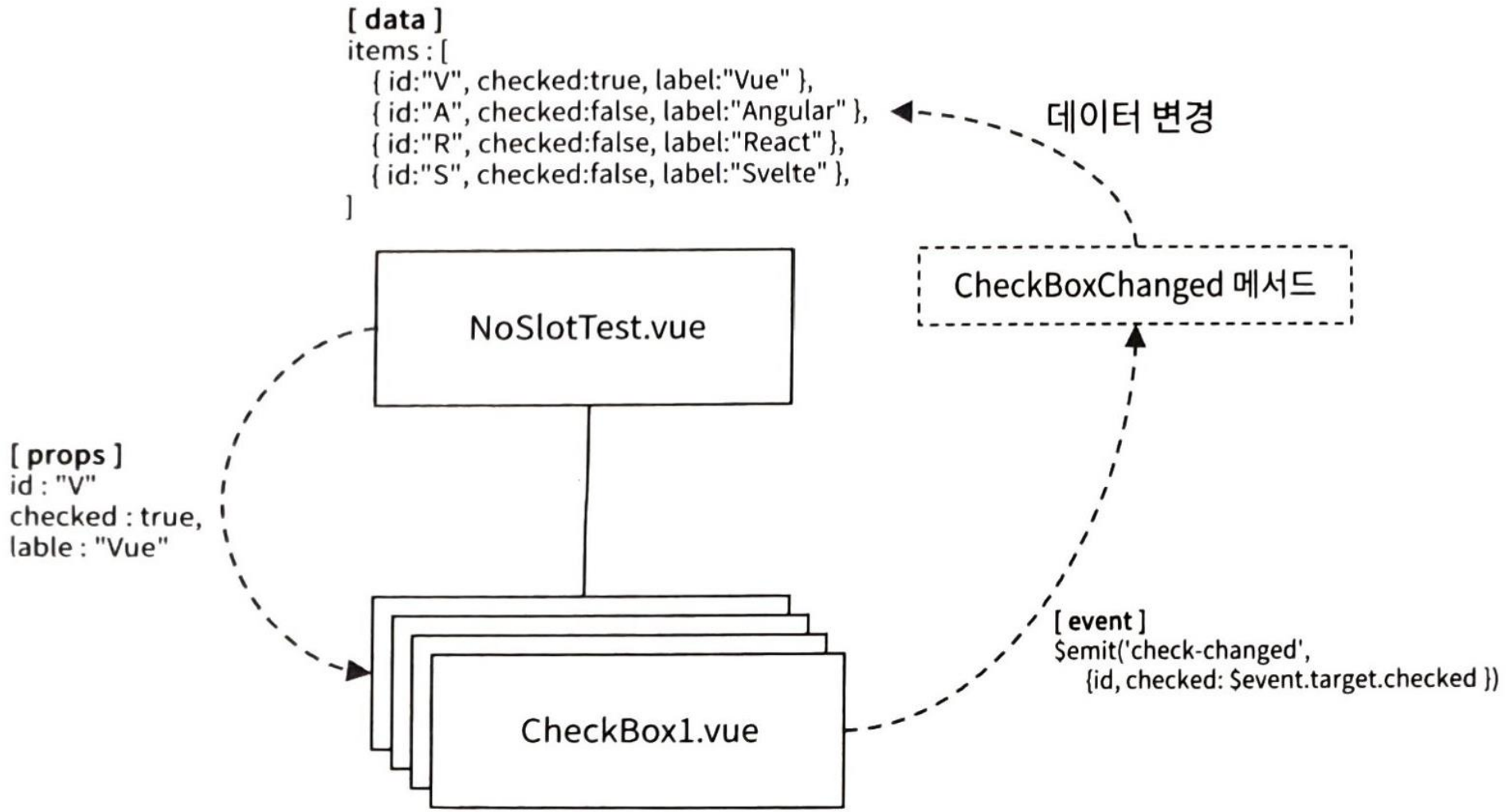
```
<script>
import CheckBox1 from './CheckBox1.vue';

export default {
    name : "NoSlotTest",
    components : { CheckBox1 },
    data() {
        return {
            items : [
                { id:"V", checked:true, label:"Vue" },
                { id:"A", checked:false, label:"Angular" },
                { id:"R", checked:false, label:"React" },
                { id:"S", checked:false, label:"Svelte" },
            ]
        }
    },
    methods : {
        CheckBoxChanged(e) {
            let item = this.items.find((item)=> item.id === e.id);
            item.checked = e.checked;
        }
    }
}
</script>
```

src/components/App.vue

```
<template>
  <div>
    <NoSlotTest />
  </div>
</template>

<script>
import NoSlotTest from './components/NoSlotTest.vue'

export default {
  name: 'App',
  components: { NoSlotTest }
}
</script>
```



컴포넌트의 구조


[data]
items : [
{ id:"V", checked:true, label:"Vue" },
{ id:"A", checked:false, label:"Angular" },
{ id:"R", checked:false, label:"React" },
{ id:"S", checked:false, label:"Svelte" },
]
데이터 변경
CheckBoxChanged 메서드
NoSlotTest.vue
[props]
id : "V"
checked : true,
lable : "Vue"
CheckBox1.vue
[event]
$emit('check-changed',
{id, checked: $event.target.checked })

슬롯의 기본 사용법

- 부모 컴포넌트에서 자식 컴포넌트로 템플릿을 전달하는 방법
- CheckBox2.vue, SlotTest.vue 컴포넌트 추가

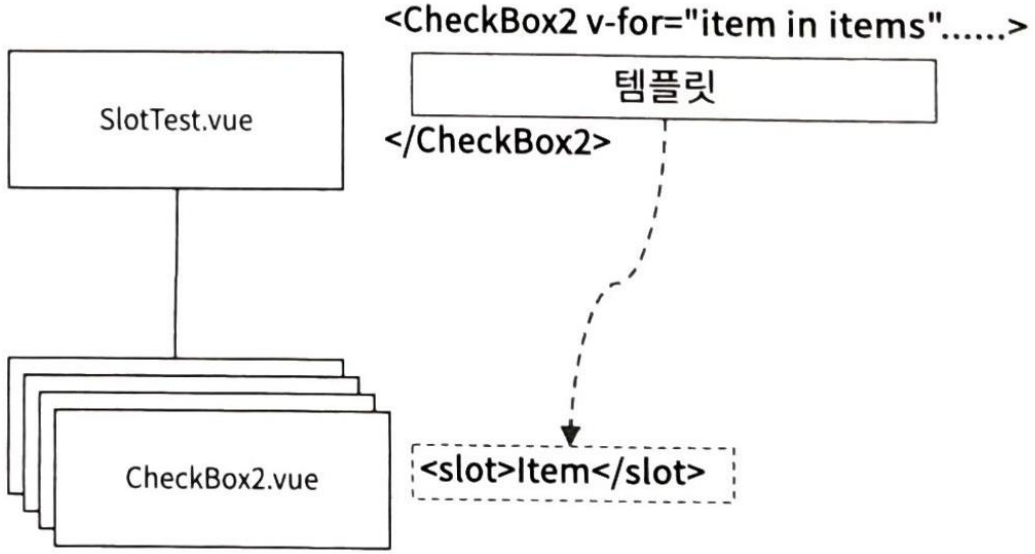
src/components/CheckBox2.vue

```
<template>
  <div>
    <input
      type="checkbox"
      :value="id"
      :checked="checked"
      @change="$emit('check-changed', { id, checked: $event.target.checked })"
    />
    <slot>Item</slot>
  </div>
</template>

<script>
export default {
  name: 'CheckBox2',
  props: ['id', 'checked'],
};
</script>
```

부모가 전달한 템플릿을 여기에 배치

src/components/SlotTest.vue

```
<template>
  <div>
    <h3>
      당신이 경험한 프론트 엔드 기술은?(두번째:slot기본)
    </h3>
    <CheckBox2
      v-for="item in items"
      :key="item.id"
      :id="item.id"
      :checked="item.checked"
      @check-changed="CheckBoxChanged">
      <span v-if="item.checked"
        style="color: blue; text-decoration: underline">
        <i>{{ item.label }}</i>
      </span>
      <span v-else style="color: gray">{{ item.label }}</span>
    </CheckBox2>
  </div>
</template>
```



전달되는 템플릿이 없는 경우 보여줄 fallback UI를 <slot></slot> 사이에 작성함

src/components/SlotTest.vue

```
<script>
import CheckBox2 from './CheckBox2.vue';

export default {
  name: 'SlotTest',
  components: { CheckBox2 },
  data() {
    return {
      items: [
        { id: 'V', checked: true, label: 'Vue' },
        { id: 'A', checked: false, label: 'Angular' },
        { id: 'R', checked: false, label: 'React' },
        { id: 'S', checked: false, label: 'Svelte' },
      ],
    };
  },
  methods: {
    CheckBoxChanged(e) {
      let item = this.items.find((item) => item.id === e.id);
      item.checked = e.checked;
    },
  },
};
</script>
```

src/components/App.vue

```
<template>
  <div>
    <NoSlotTest />
    <SlotTest />
  </div>
</template>

<script>
import NoSlotTest from './components/NoSlotTest.vue'
import SlotTest from './components/SlotTest.vue'

export default {
  name: 'App',
  components: { NoSlotTest, SlotTest }
}
</script>
```

명명된 슬롯

- 자식 컴포넌트에 slot 영역을 여러 개 지정한 경우
- 각 영역에 이름을 배정
  - <slot name="슬롯명"></slot>

- 부모 컴포넌트는 v-slot 디렉티브로 어떤 슬롯에대한 템플릿인지 이름으로 지정
  - <... v-slot:슬롯명> </...>
    - 슬롯명에 따움표 붙이지 않음

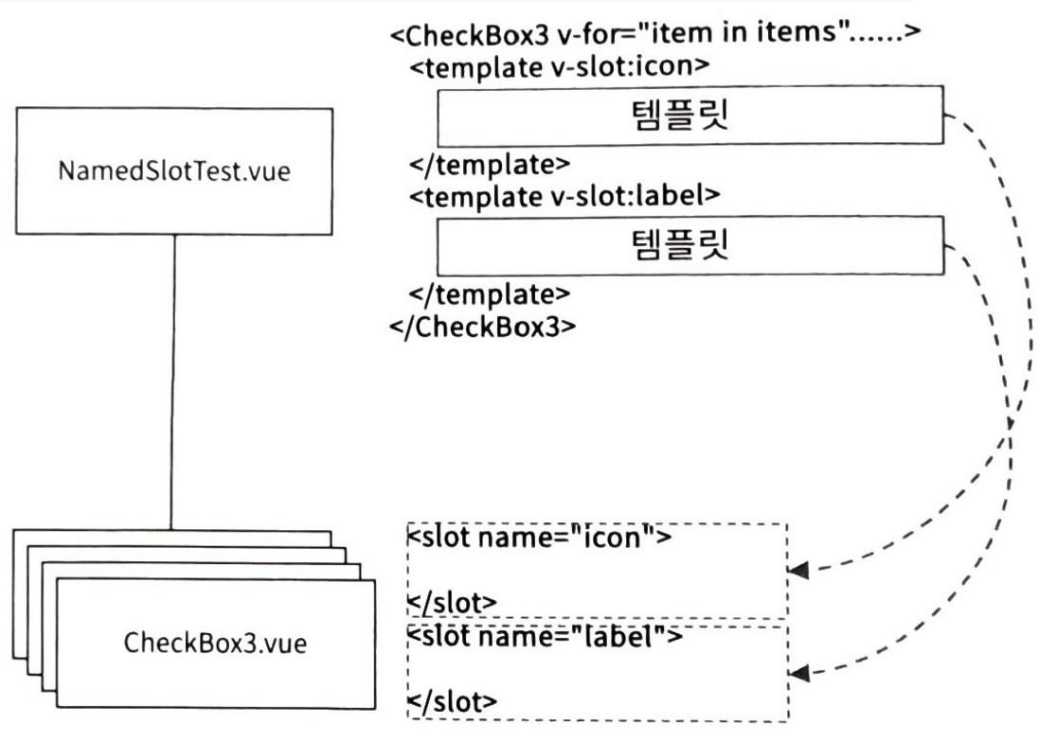
- CheckBox3.vue, NamedSlotTest.vue 컴포넌트 추가

src/components/CheckBox3.vue

```
<template>
  <div>
    <slot name="icon"></slot>
    <input
      type="checkbox"
      :value="id"
      :checked="checked"
      @change="$emit('check-changed', { id, checked: $event.target.checked })"
    />
    <slot name="label">Item</slot>
  </div>
</template>

<script>
export default {
  name: 'CheckBox3',
  props: ['id', 'checked'],
};
</script>
```

src/components/NamedSlotTest.vue

```
<template>
  <div>
    <h3>당신이 경험한 프론트 엔드 기술은?(세번째: named slot)</h3>
    <CheckBox3
      v-for="item in items"
      :key="item.id"
      :id="item.id"
      :checked="item.checked"
      @check-changed="CheckBoxChanged">
      <template v-slot:icon>
        <i v-if="item.checked" class="far fa-grin-beam"></i>
        <i v-else class="far fa-frown"></i>
      </template>
      <template v-slot:label>
        <span
          v-if="item.checked"
          style="color: blue; text-decoration: underline">
          <i>{{ item.label }}</i>
        </span>
        <span v-else style="color: gray">{{ item.label }}</span>
      </template>
    </CheckBox3>
  </div>
</template>
```



src/components/NamedSlotTest.vue

```
<script>
import CheckBox3 from './CheckBox3.vue';
export default {
  name: 'SlotTest',
  components: { CheckBox3 },
  data() {
    return {
      items: [
        { id: 'V', checked: true, label: 'Vue' },
        { id: 'A', checked: false, label: 'Angular' },
        { id: 'R', checked: false, label: 'React' },
        { id: 'S', checked: false, label: 'Svelte' },
      ],
    };
  },
  methods: {
    CheckBoxChanged(e) {
      let item = this.items.find((item) => item.id === e.id);
      item.checked = e.checked;
    },
  },
};
</script>
<style>
@import url('https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.14.0/css/all.min.css');
</style>
```

src/App.vue

```
<template>
  <div>
    <NoSlotTest />
    <SlotTest />
    <NamedSlotTest />
  </div>
</template>

<script>
import NoSlotTest from './components/NoSlotTest.vue'
import SlotTest from './components/SlotTest.vue'
import NamedSlotTest from './components/NamedSlotTest.vue'

export default {
  name: 'App',
  components: { NoSlotTest, SlotTest, NamedSlotTest }
}
</script>
```

명명된 슬롯의 활용

- 화면 레이아웃 관리 목적으로 많이 사용

- Layout.vue 컴포넌트와 App2.vue 추가

src/components/Layout.vue

```
<template>
  <div class="wrapper">
    <header class="box header">
      <slot name="header"></slot>
    </header>
    <aside class="box sidebar">
      <slot name="sidebar"></slot>
    </aside>
    <section class="box content">
      <slot></slot>
    </section>
    <footer class="box footer">
      <slot name="footer"></slot>
    </footer>
  </div>
</template>

<script>
export default {
  name: 'Layout',
};
</script>
```

v-slot:default에 대응

src/components/Layout.vue

```
<style scoped>
body {
  margin: 40px;
}
.sidebar {
  grid-area: sidebar;
}
.content {
  grid-area: content;
  position: relative;
}
.header {
  grid-area: header;
}
.footer {
  grid-area: footer;
}
.wrapper {
  display: grid;
  grid-gap: 5px;
  grid-template-rows: 100px 1fr 100px;
  grid-template-columns: 1fr 200px;
  grid-template-areas:
    'header  header'
    'content sidebar'
    'footer  footer';
  background-color: #fff;
  color: #444;
}
.box {
  background-color: #ddd;
  color: #444;
  border-radius: 5px;
  padding: 10px;
  font-size: 100%;
}
</style>
```

src/App2.vue

```
<template>
  <Layout>
    <template v-slot:header>
      <h2>헤더 영역</h2>
    </template>

    <template v-slot:sidebar>
      <h3>사이드</h3>
      <h3>사이드</h3>
      <h3>사이드</h3>
    </template>

    <template v-slot:default>
      <h1>컨텐트 영역</h1>
      <h1>컨텐트 영역</h1>
      <h1>컨텐트 영역</h1>
      <h1>컨텐트 영역</h1>
      <h1>컨텐트 영역</h1>
      <h1>컨텐트 영역</h1>
    </template>

    <template v-slot:footer>
      <h2>Footer text</h2>
    </template>
  </Layout>
</template>
```

```
<script>
import Layout from './components/Layout.vue';
export default {
  name: 'App2',
  components: { Layout },
};
</script>
```

헤더 영역

컨텐트 영역
컨텐트 영역
컨텐트 영역
컨텐트 영역
컨텐트 영역
컨텐트 영역

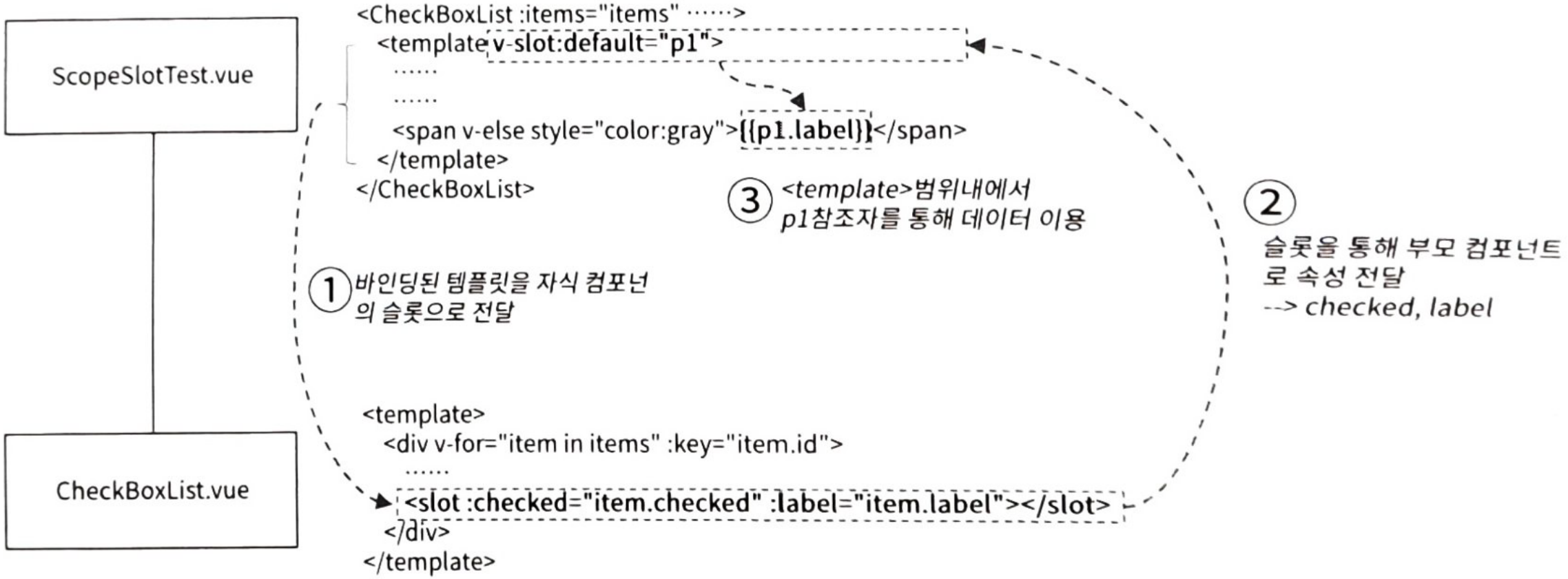
사이드
사이드
사이드

Footer text

범위 슬롯(Scoped Slot)

부모 컴포넌트에서 템플릿에 데이터를 바인딩할 때

- 자식 컴포넌트의 데이터를 이용해 바인딩하는 경우
  → 범위 슬롯 사용
- 전달된 데이터는 슬롯 템플릿 내부 범위에서만 사용가능



src/components/CheckBoxList.vue

```
<template>
  <div v-for="item in items" :key="item.id">
    <slot name="icon" :checked="item.checked"></slot>
    <input
      type="checkbox"
      :value="item.id"
      :checked="item.checked"
      @change="$emit('check-changed', { id: item.id, checked: $event.target.checked })"
    />
    <slot :checked="item.checked" :label="item.label"></slot>
  </div>
</template>

<script>
export default {
  name: 'CheckBoxList',
  props: ['items'],
};
</script>
```

src/components/ScopedSlotTest.vue

```
<template>
  <div>
    <h3>당신이 경험한 프론트 엔드 기술은?(네번째:ScopedSlot)</h3>

    <CheckBoxList :items="items" @check-changed="CheckBoxChanged">
      <template v-slot:icon="p1">
        <i v-if="p1.checked" class="far fa-grin-beam"></i>
        <i v-else class="far fa-frown"></i>
      </template>

      <template v-slot:default="p2">
        <span v-if="p2.checked" style="color: blue; text-decoration: underline">
          <i>{{ p2.label }}</i>
        </span>
        <span v-else style="color: gray">{{ p2.label }}</span>
      </template>
    </CheckBoxList>
  </div>
</template>
```

src/components/ScopedSlotTest.vue

```
<script>
import CheckBoxList from './CheckBoxList.vue';
export default {
  name: 'ScopedSlotTest',
  components: { CheckBoxList },
  data() {
    return {
      items: [
        { id: 'V', checked: true, label: 'Vue' },
        { id: 'A', checked: false, label: 'Angular' },
        { id: 'R', checked: false, label: 'React' },
        { id: 'S', checked: false, label: 'Svelte' },
      ],
    };
  },
  methods: {
    CheckBoxChanged(e) {
      let item = this.items.find((item) => item.id === e.id);
      item.checked = e.checked;
    },
  },
};
</script>
<style>
@import url('https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.14.0/css/all.min.css');
</style>
```

src/App.vue

```
<template>
  <div>
    <NoSlotTest />
    <SlotTest />
    <NamedSlotTest />
    <ScopedSlotTest />
  </div>
</template>

<script>
import NoSlotTest from './components/NoSlotTest.vue'
import SlotTest from './components/SlotTest.vue'
import NamedSlotTest from './components/NamedSlotTest.vue'
import ScopedSlotTest from './components/ScopedSlotTest.vue'

export default {
  name: 'App',
  components: { NoSlotTest, SlotTest, NamedSlotTest, ScopedSlotTest }
}
</script>
```

동적 컴포넌트 dynamic component

- 화면의 동일한 위치에 여러 컴포넌트를 표현해야 하는 경우 사용
- <component> 요소를 템플릿에 작성
- v-bind 디렉티브를 이용해 is 특성 값으로 어떤 컴포넌트를 그 위치에 나타낼지 결정
  - 컴포넌트 명을 문자열로 대입 – 대소문자 구분함
- 컴포넌트가 변경될 때
  - 컴포넌트가 다시 생성됨
  - 정적 컴포넌트인 경우 include 속성으로 캐싱

프로젝트 생성

```
npm init vue dynamic-component-test
cd dynamic-component-test
npm install
npm install bootstrap@5
```

- src/components, src/assests 내용 모두 삭제
- CoralSeaTab.vue, LeyteFulfTab.vue, CoralSeaTab.vue 컴포넌트 파일 생성

src/components/CoralSeaTab.vue

```
<template>
    <div class="tab">
        <h1>산호해 해전에 대해</h1>
        <div>{{ (new Date()).toTimeString() }}</div>
    </div>
</template>
<script>
    export default {
        name: "CoralSeaTab"
    }
</script>
```

src/components/LeyteGulfTab.vue

```
<template>
    <div class="tab">
        <h1>레이테만 해전에 대해</h1>
        <div>{{ (new Date()).toTimeString() }}</div>
    </div>
</template>
<script>
    export default {
        name: "LeyteGulfTab"
    }
</script>
```

src/components/MidwayTab.vue

```
<template>
    <div class="tab">
        <h1>미드웨이 해전에 대해</h1>
        <div>{{ (new Date()).toTimeString() }}</div>
    </div>
</template>
<script>
    export default {
        name: "MidwayTab"
    }
</script>
```

src/App.vue

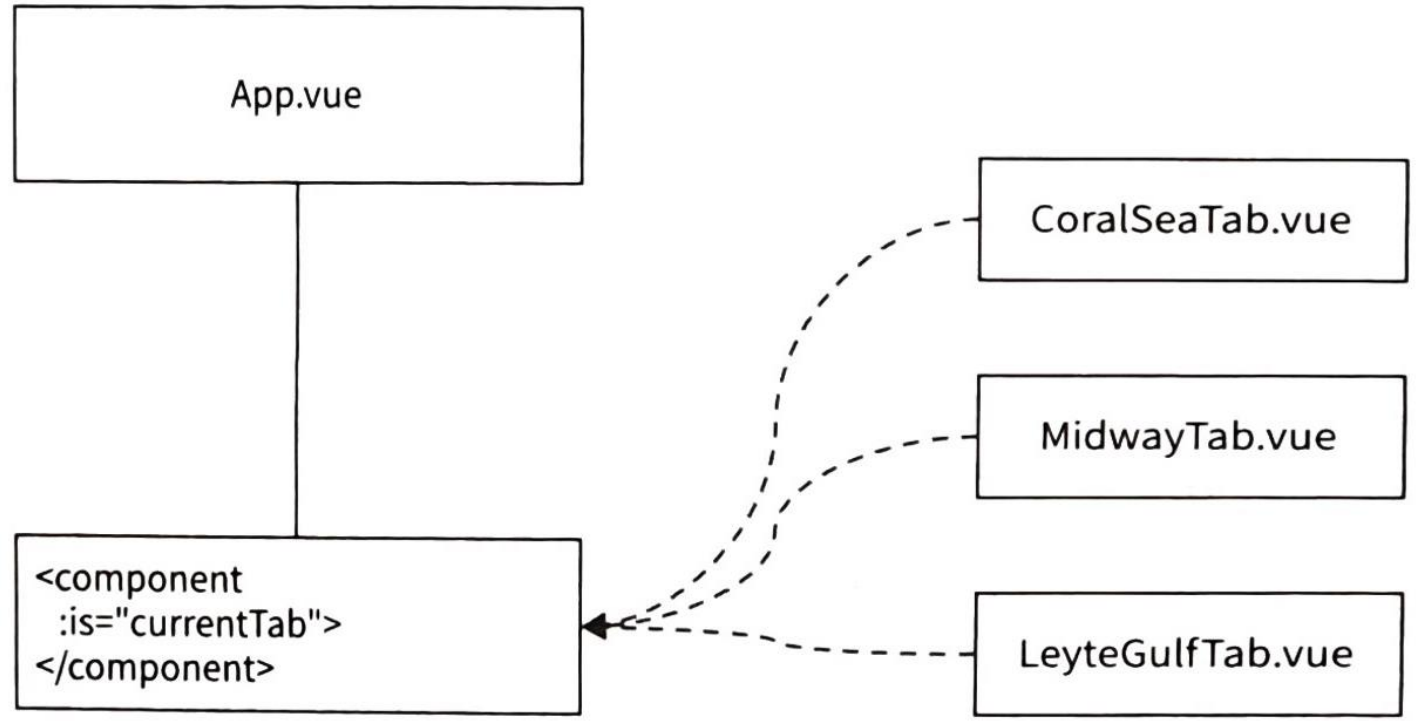
```
<template>
  <div class="header">
    <h1 class="headerText">태평양 전쟁의 해전</h1>
    <nav>
      <ul class="nav nav-tabs nav-fill">
        <li v-for="tab in tabs" :key="tab.id" class="nav-item">
          <a style="cursor:pointer;" class="nav-link"
            :class="{ active : tab.id === currentTab }"
            @click="changeTab(tab.id)">{{tab.label}}</a>
        </li>
      </ul>
    </nav>
  </div>
  <div class="container">
    <keep-alive include="MidwayTab,CoralSeaTab">
      <component :is="currentTab"></component>
    </keep-alive>
  </div>
</template>
```

src/App.vue

```
<script>
import CoralSeaTab from './components/CoralSeaTab.vue'
import LeyteGulfTab from './components/LeyteGulfTab.vue'
import MidwayTab from './components/MidwayTab.vue'

export default {
  name: 'App',
  components : { CoralSeaTab, LeyteGulfTab, MidwayTab },
  data() {
    return {
      currentTab : 'CoralSeaTab',
      tabs : [
        { id:"CoralSeaTab", label:"산호해 해전" },
        { id:"MidwayTab", label:"미드웨이 해전" },
        { id:"LeyteGulfTab", label:"레이테만 해전" }
      ]
    }
  },
```

src/App.vue

```
  methods : {
    changeTab(tab) {
      this.currentTab = tab;
    }
  }
}
</script>

<style>
.header { padding: 20px 0px 0px 20px; }
.headerText { padding: 0px 20px 40px 20px; }
.tab { padding: 30px }
</style>
```



* is 특성에 바인한 currentTab 값이 등록한 컴포넌트 이름과 같을 때 <component />를 대신해 해당 컴포넌트가 렌더링됨.
* 등록한 컴포넌트 이름은 components 옵션으로 등록한 것을 의미함.

사용자 정의 v-model 만들기

부모 컴포넌트

```
<child-component v-model:message="parentMessage" />
```

자식 컴포넌트

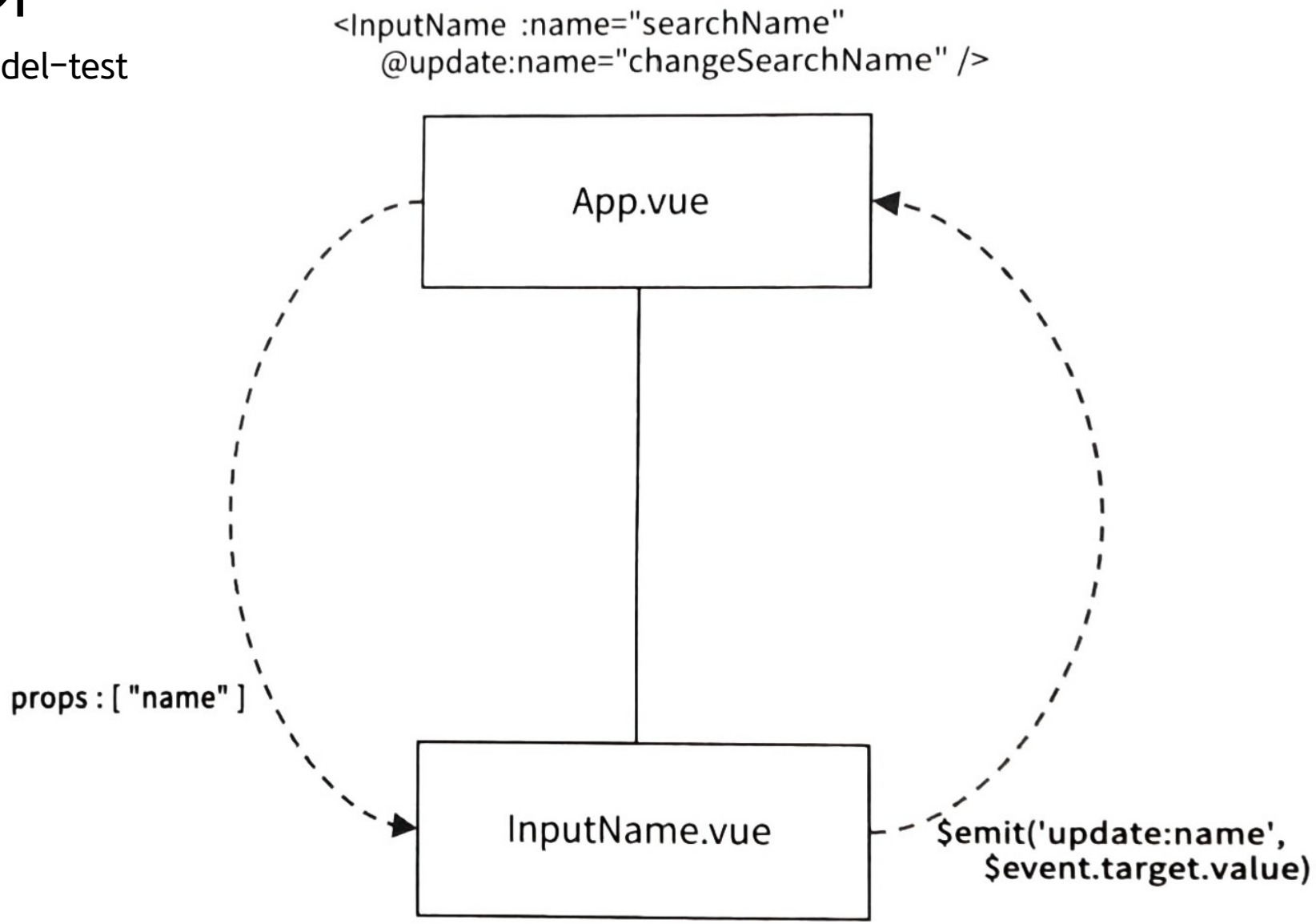
- 모델에 해당하는 속성을 반드시 정의

```
{
  name: "ChildComponent",
  props: {message: String},
  template: `
    <input type="text" :value="message"
        @input="$emit('update:message', $event.target.value)"/>
  `
}
```

- 이벤트명은 반드시 'update:속성명' 형식을 사용

실습 프로젝트 만들기

npm init vue v-model-test
cd v-model-test
npm install



src/components/InputName.vue

```
<template>
  <input
    type="text"
    :value="name"
    @input="$emit('update:name', $event.target.value)"
  />
</template>

<script>
export default {
  name: 'InputName',
  props: ['name'],
};
</script>
```

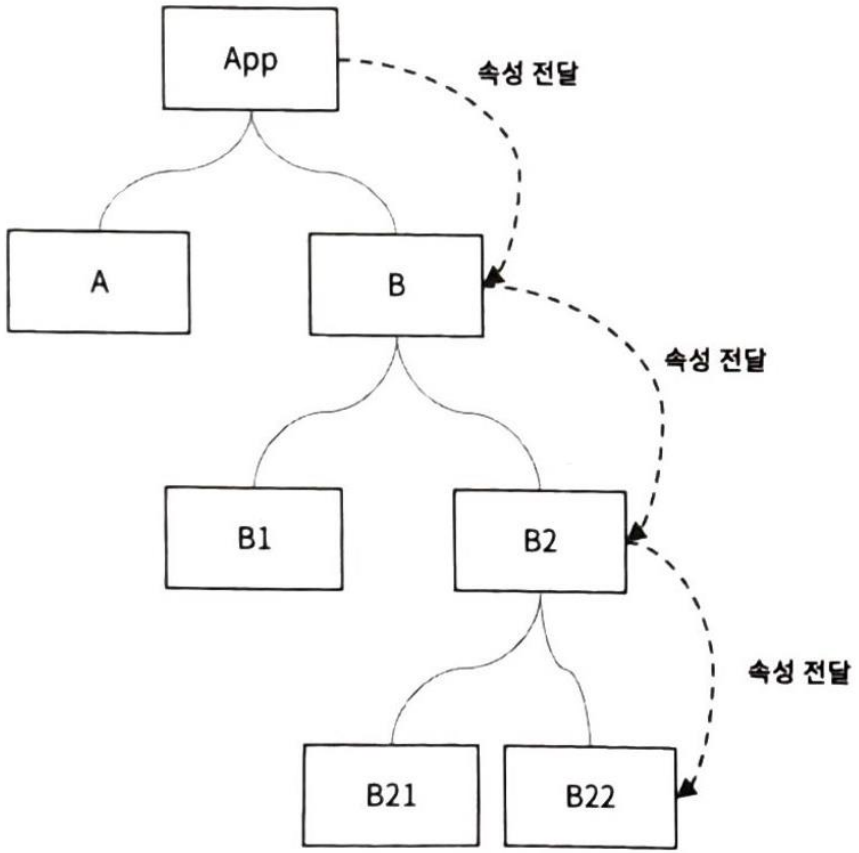
src/App.vue

```
<template>
  <div>
    <InputName v-model:name="searchName"}  />
    <h3>검색어 : {{ searchName }}</h3>
  </div>
</template>

<script>
import InputName from './components/InputName.vue';

export default {
  name: 'App',
  components: { InputName },
  data() {
    return { searchName: 'John' };
  },
};
</script>
```

props를 이용한 정보 전달

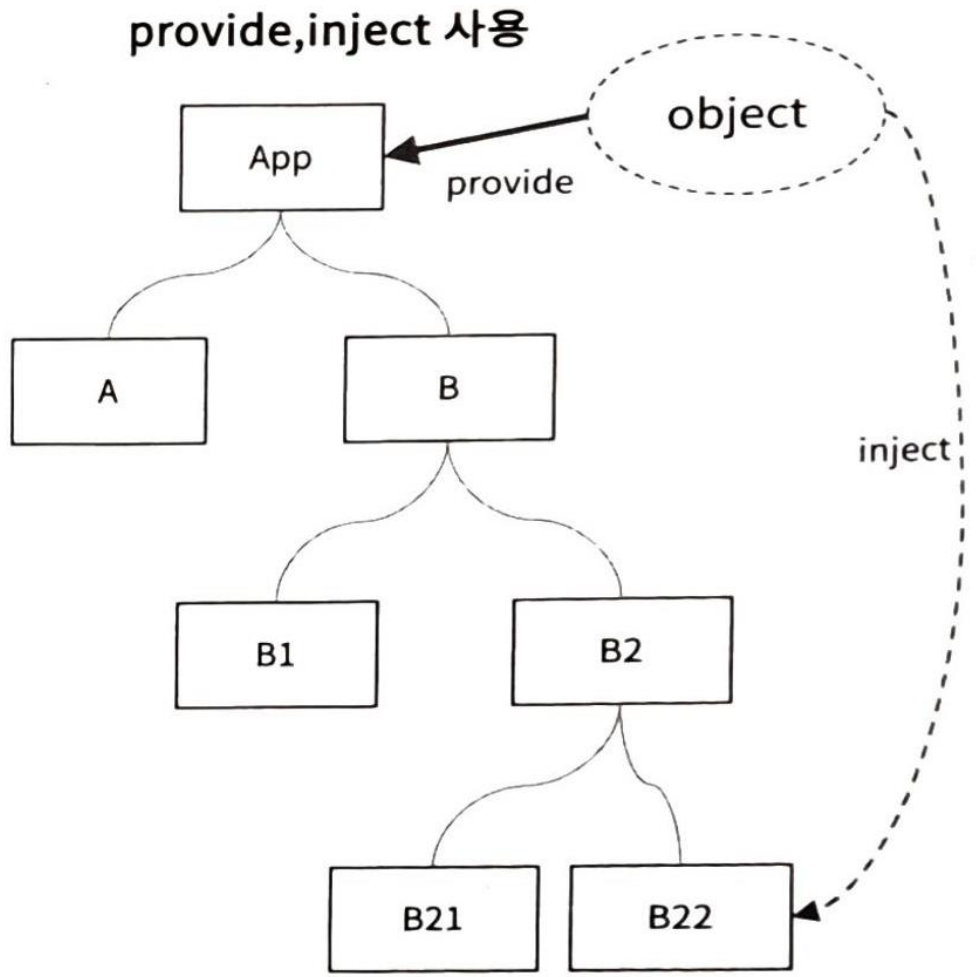
- 컴포넌트의 계층 구조가 복잡해지면
  → 계층 구조를 따라 연속적으로 속성을 전달해야 하는 문제 발생

연속적인 속성의 전달



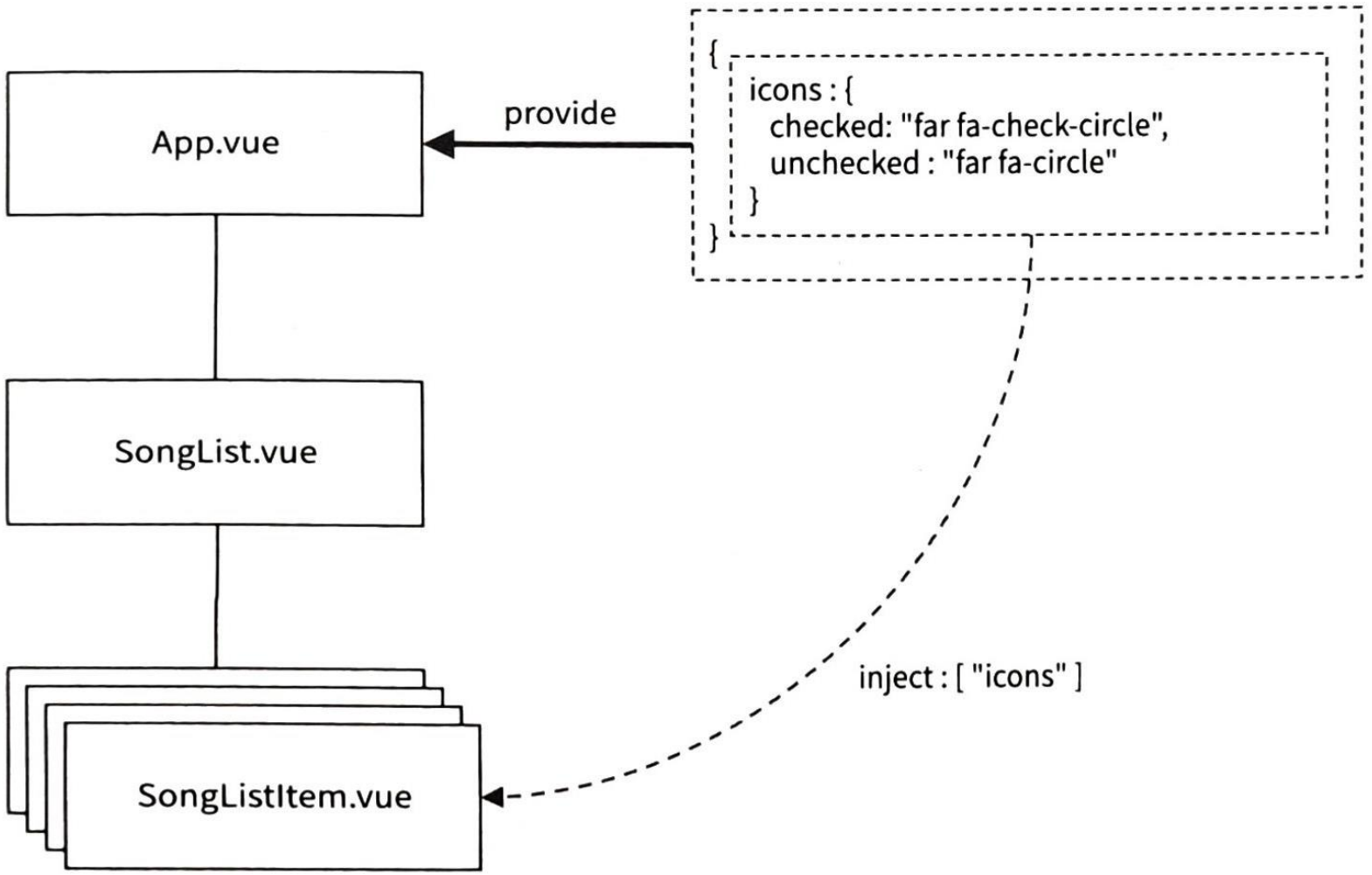
provide, inject

- 공용 데이터를 부모 컴포넌트(App)에서 제공(provide)
- 하위 컴포넌트 트리상의 어느 컴포넌트에서나 필요한 데이터를 주입(inject)하여 사용



실습 프로젝트 생성

npm init vue provide-inject-teleport-test
cd provide-inject-teleport-test
npm install



src/App.vue

```
<template>
  <div>
    <h2>최신 인기곡</h2>

  </div>
</template>

<script>
import { computed } from 'vue';

export default {
  name: 'App',
  data() {
    return {
      songs: [
      { id: 1, title: 'Blueming', done: true },
      { id: 2, title: 'Dynamite', done: true },
      { id: 3, title: 'Lovesick Girls', done: false },
      { id: 4, title: '마리아(Maria)', done: false },
     ],
    };
  },
  provide() {
    return {
      icons: {
        checked: 'far fa-check-circle',
        unchecked: 'far fa-circle',
      },
      doneCount: computed(() => {
        return this.songs.filter(
                  (s) => s.done === true).length;
      }),
    };
  },
};
</script>

<style>
@import
url('https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.14.0/css/all.min.css');
</style>
```

src/components/SongListItem.vue

```
<template>
  <li>
    <i :class="song.done ? icons.checked : icons.unchecked"></i>
    {{ song.title }}
  </li>
</template>

<script>
export default {
  name: 'SongListItem',
  inject: ['icons'],
  props: ['song'],
};
</script>
```

src/components/SongList.vue

```
<template>
  <ul>
    <SongListItem v-for="s in songs" :key="s.id" :song="s" />
  </ul>
  <div>체크된 곡 수 : {{ doneCount }}</div>
</template>

<script>
import SongListItem from './SongListItem.vue';
export default {
  name: 'SongList',
  components: {
    SongListItem,
  },
  props: ['songs'],
  inject: ['doneCount'],
};
</script>
```

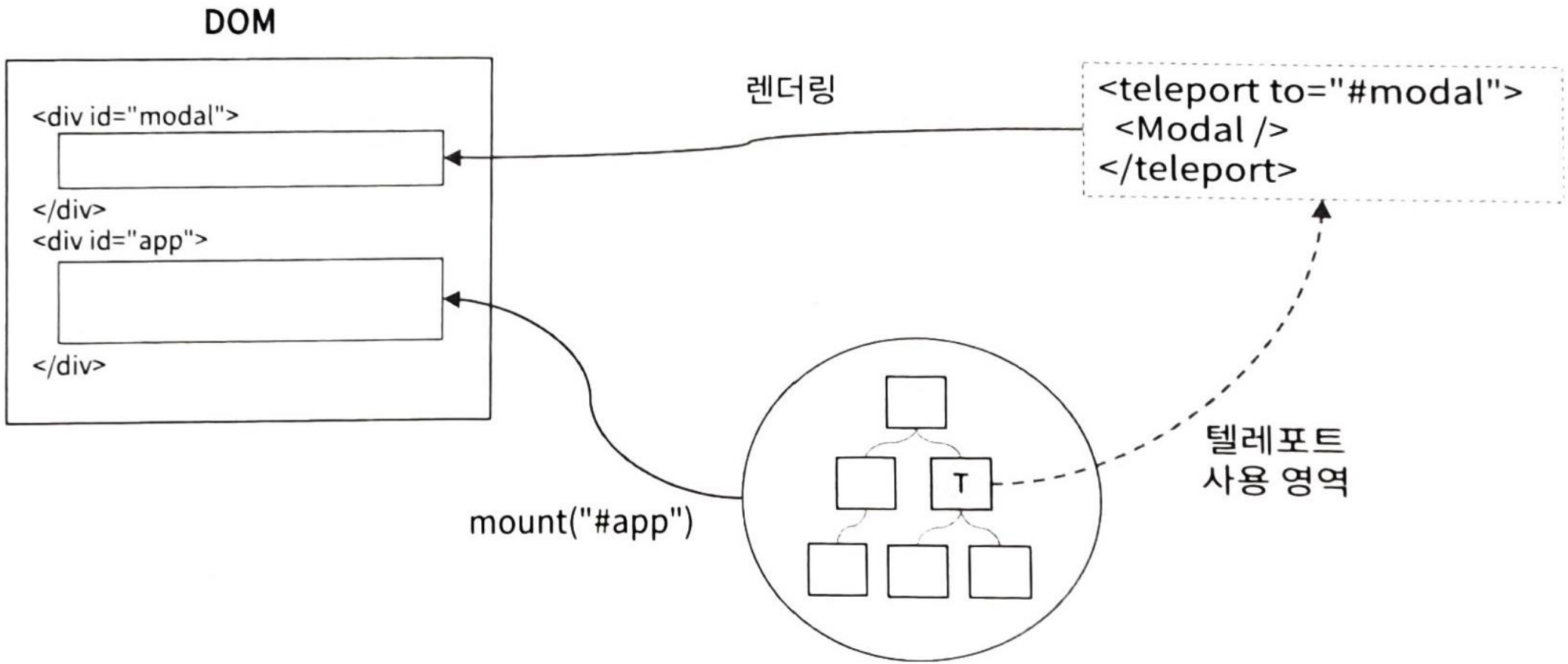
src/App.vue

```
<template>
  <div>
    <h2>최신 인기곡</h2>
    <SongList :songs="songs" />
  </div>
</template>

<script>
import { computed } from 'vue';
import SongList from './components/SongList.vue';

export default {
  name: 'App',
  components: { SongList },
  data() {
    return {
      songs: [
      { id: 1, title: 'Blueming', done: true },
      { id: 2, title: 'Dynamite', done: true },
      { id: 3, title: 'Lovesick Girls', done: false },
      { id: 4, title: '마리아(Maria)', done: false },
      ],
    };
  },
  provide() {
    return {
      icons: {
        checked: 'far fa-check-circle',
        unchecked: 'far fa-circle',
      },
      doneCount: computed(() => {
        return this.songs.filter(
                    (s) => s.done === true).length;
      }),
    };
  },
};
</script>

<style>
@import
url('https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.14.0/css/all.min.css');
</style>
```

텔레포트 Teleport

- 애플리케이션에서 모달, 툴팁과 같이 메인 화면과 독립적이면서 공유 UI를 제공하는 경우
- 컴포넌트 트리의 계층 구조와 관계없이 별도의 요소에 렌더링해야 함

→ 이런 경우 사용하는 기능이 텔레포트임



src/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <link rel="icon" href="/favicon.ico">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Vite App</title>
  </head>
  <body>
    <div id="modal"></div>
    <div id="app"></div>
    <script type="module" src="/src/main.js"></script>
  </body>
</html>
```

src/components/Modal.vue

```
<template>
    <div class="modal">
        <div class="box">
            처리중
        </div>
    </div>
</template>

<script>
export default {
    name : "Modal"
}
</script>
```

src/components/Modal.vue

```
<style scoped>
.modal {
    display: block; position: fixed; z-index: 1;
    left: 0; top: 0; width: 100%; height: 100%;
    overflow: auto;
    background-color: rgba(0,0,0,0.4);
}
.box {
    display: flex; flex-direction: column;
    align-items: center; justify-content: center;
    position: absolute; left: 50%; top: 50%; background-color:aqua;
    border: double 3px gray; width:100px; height: 80px;
    margin-top:-50px; margin-left:-50px;
}
</style>
```

src/App.vue

```
<template>
  <div>
    <h2>최신 인기곡</h2>
    <SongList :songs="songs" />
    <br /><br />
     <button @click="changeModal">Teleport를 이용한 Modal 기능</button>
     <teleport to="#modal">
       <Modal v-if="isModal" />
     </teleport>
  </div>
</template>

<script>
import { computed } from 'vue';
import SongList from './components/SongList.vue'
import Modal from './components/Modal.vue'

export default {
  name : "App",
  components : { SongList, Modal },
```

src/App.vue

```
data() {
  return {
      songs : [
          { id:1, title:"Blueming", done:true },
          { id:2, title:"Dynamite", done:true },
          { id:3, title:"Lovesick Girls", done:false },
          { id:4, title:"마리아(Maria)", done:false },
      ],
      isModal : false,
  }
},

methods : {
  changeModal() {
    this.isModal = true;
    setTimeout(()=>{ this.isModal = false }, 2000);
  }
},
```

src/App.vue

```
  provide() {
    return {
      icons : {
        checked : "far fa-check-circle",
        unchecked : "far fa-circle",
      },
      doneCount : computed(()=> {
        return this.songs.filter((s)=>s.done === true).length
      })
    }
  },
}
</script>

<style>
@import url("https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.14.0/css/all.min.css");
</style>
```

지금까지의 컴포넌트

- index.html을 로드하고 관련된 .js 파일을 로드

  → 화면에 당장 그 컴포넌트가 필요하지 않아도 미리 로딩함,
  초기 화면 로딩 시간이 지연되는 원인

비동기 컴포넌트

- 실제 해당 컴포넌트를 사용하는 시점에 관련된 .js 파일을 로딩
- 프로젝트를 빌드할 때 비동기 컴포넌트를 위한 별도의 .js 파일 분리
- 비동기 컴포넌트의 사용
  - defineAsyncComponent 메서드 사용

[일반적인 컴포넌트의 임포트]

```
import SyncComponent from './SyncComponent.vue';
```

[비동기 컴포넌트의 임포트]

```
import { defineAsyncComponent } from 'vue'

const AsyncComponent = defineAsyncComponent(
  () => import('./AsyncComponent.vue')
)
```