컴포넌트 심화 학습

생성일	@2024년 11월 4일 오후 12:36
태그	

▼ 컴포넌트 안에 다른 컴포넌트 사용하기

- 부모 컴포넌트와 자식 컴포넌트
- Props
  - 부모 컴포넌트에서 자식 컴포넌트로 데이터 전달

  ▼ 기본 사용

  - 자식 컴포넌트

```
<template>
  <h2>{{ title }}</h2>
</template>

<script>
export default {
  props: {
    title: {
      type: String,
      default: "페이지 제목입니다.",
    },
  },
};
</script>
```

  - 부모 컴포넌트

```
<template>
  <div>
    <PageTitle :title="부모 컴포넌트에서 자식 컴포넌트로 데이터 전달" />
  </div>
</template>

<script>
import PageTitle from "../components/PageTitle.vue";

export default {
  components: { PageTitle },
};
</script>
```

▼ 데이터 타입별 전달 방법

종류	설명
정적 데이터	디렉티브 없이 props에 값을 전달 - 모든 값이 문자열로 인식
동적 데이터	v-bind 디렉티브 사용
숫자형(Number)	v-bind 디렉티브 사용
논리 자료형(Boolean)	v-bind 디렉티브 사용
배열(Array)	v-bind 디렉티브 사용

객체(Object)	v-bind 디렉티브 사용
객체(Object)의 속성	v-bind 디렉티브 사용

▼ Prop 유효성 검사

- props 옵션 정의 시 다음 유형들을 통해 유효성 검사 가능
  - 데이터 타입(type), 기본값(default), 필수여부(required), 유효성 검사(validator)
    - null, undefined는 모든 타입 유효성 검사 통과
    - 객체나 배열의 기본 값은 항상 팩토리 함수로부터 반환되어야 함

▼ 자식 컴포넌트에서 부모 컴포넌트로 이벤트/데이터 전달하기(커스텀 이벤트)

1. 자식 컴포넌트에서 $emit 이용

```
this.$emit('custom-event'[, data]);

// data : 커스텀 이벤트와 함께 전달할 데이터
```

2. 부모 컴포넌트에서 커스텀 이벤트를 정의

```
<childComponent @custon-event="EventHandler" />

methods : {
    EventHandler(data){
        // data : 자식 컴포넌트에서 $emit으로 보낸 데이터
    }
}
```

▼ ref 속성

- 부모 컴포넌트에서 자식 컴포넌트에 직접 접근하기
  - 부모 컴포넌트 내부 자식 컴포넌트에 ref=”id” 를 지정
    ⇒ 자식 컴포넌트 의 속성에 접근
  - 호출 코드

```
this.$refs.child_component
```

▼ 부모 컴포넌트에서 자식 컴포넌트의 이벤트 직접 발생시키기

- 자식 컴포넌트 내부 HTML 태그에 ref=”id” 를 지정
  ⇒ Vue 컴포넌트의 함수에서 this.$refs를 통해 접근 가능
  - id 값은 유일한 키 값을 사용
- 호출 코드

```
this.$refs.child_component.$refs.ref_id.event();
```

▼ 부모 컴포넌트에서 자식 컴포넌트의 함수 직접 호출하기

- 호출 코드

```
this.$refs.child_component.child_function();
```

▼ 부모 컴포넌트에서 자식 컴포넌트의 데이터 옵션 값 직접 변경하기

- 호출 코드

```
this.$refs.child_component.data;
```

▼ 부모 컴포넌트에서 자식 컴포넌트의 데이터 옵션 값 동기화하기

- 부모 컴포넌트의 computed 속성을 사용

```
computed : {
    property() {
        return this.$refs.child_component.data;
    }
}
```

▼ Slot

- 정의
  - 컴포넌트 내에서 다른 컴포넌트를 사용할 때
    쓰는 컴포넌트의 마크업을 재정의하거나 확장하는 기능
  - 컴포넌트의 재활용성을 높여줌
  - 코드 훨씬 간결, 직관적

▼ 사용

1. 자식 컴포넌트

```
<template>
  <div class="modal-container">
    <header>
      <slot name="header"></slot>
    </header>
    <main>
      <slot></slot>
    </main>
    <footer>
      <slot name="footer"></slot>
    </footer>
  </div>
</template>
```

2. 부모 컴포넌트

```
<modal-layout>
    <template v-slot:header>
        <h1>팝업 타이틀</h1>
    </template>
    <template v-slot:default>
        <p>팝업 컨텐츠 1</p>
        <p>팝업 컨텐츠 2</p>
    </template>
    <template v-slot:footer>
        <button>닫기</button>
    </template>
</modal-layout>
```

```
// v-slot 대신 단축어로 # 사용 가능
<modal-layout>
    <template #header>
        <h1>팝업 타이틀</h1>
    </template>
    <template #default>
        <p>팝업 컨텐츠 1</p>
        <p>팝업 컨텐츠 2</p>
```

```
    </template>
    <template #footer>
        <button>닫기</button>
    </template>
</modal-layout>
```

▼ Provide/Inject

- 컴포넌트의 계층 구조를 거치지 않고 한 번에 바로 전달
- 사용
  1. 부모 컴포넌트에서 Provide 옵션

```
provide(){
    return {
        data : this.property;
    }
```

  2. 자식 컴포넌트에서 inject 옵션

```
inject : ['data']
```

- 단점
  - 자식 컴포넌트에서 전달받는 데이터가 어떤 부모 컴포넌트에서 전달되는지 확인 불가

▼ Template refs

- JavaScript에서 HTML 객체에 바로 접근하기 위해 ref 속성 사용
  - Vue 개발 시 특별한 경우가 아니면 HTML 객체에 바로 접근 하지 않음
- 사용
  - this.$refs.ref_id을 통해 HTML 객체에 접근

```
<tag ref="id"/>

this.$refs.ref_id.event();
```