

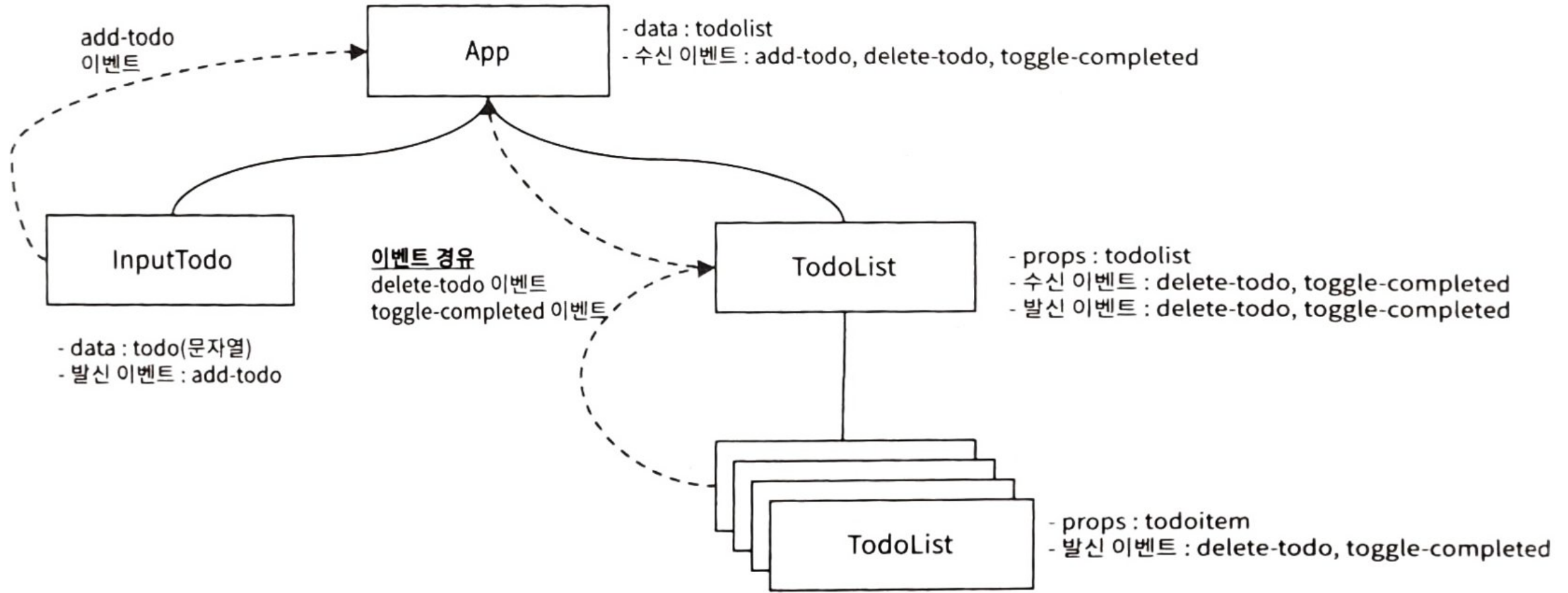
2025년 상반기 K-디지털 트레이닝

TodoList App 리팩토링(Composition API)

[KB] IT's Your Life

1 TodoList 앱 리팩토링

✓ TodoList 앱의 계층 구조(기존)



```
graph TD; App[App] -- "add-todo 이벤트" -.-> InputTodo[InputTodo]; App -- "이벤트 경유" -.-> TodoList[TodoList]; App -- "이벤트 경유" -.-> TodoListItems[... TodoListItem]; InputTodo -- "add-todo" --> App; TodoList -- "delete-todo 이벤트" -.-> App; TodoList -- "toggle-completed 이벤트" -.-> App; TodoListItems -- "delete-todo 이벤트" -.-> App; TodoListItems -- "toggle-completed 이벤트" -.-> App;
```

The diagram illustrates the component structure and data flow of a To-Do List application:

- App**:
 - state : `reactive({ todolist: [] })`
 - `onMounted()` : state의 todolist 초기화
 - 수신 이벤트 : add-todo, delete-todo, toggle-completed
- InputTodo**:
 - `todo : ref('')`
 - 발신 이벤트 : add-todo
- TodoList**:
 - props : todolist
 - 수신 이벤트 : delete-todo, toggle-completed
 - 발신 이벤트 : delete-todo, toggle-completed
- TodoListItem** (stack):
 - props : todoitem
 - 발신 이벤트 : delete-todo, toggle-completed

Event Flow:

- A dashed arrow labeled "add-todo 이벤트" points from **InputTodo** to **App**.
- Solid arrows point from **App** to both **InputTodo** and **TodoList**.
- Dashed arrows labeled "이벤트 경유" (event passing) point from **App** to both **TodoList** and the **TodoListItem** stack.
- Dashed arrows labeled "delete-todo 이벤트" and "toggle-completed 이벤트" point from both **TodoList** and the **TodoListItem** stack back to **App**.

1 TodoList 앱 리팩토링

src/App.vue

```
<template>
  <div class="container">
    <div class="card card-body bg-light">
      <div class="title">:: Todolist App</div>
    </div>
    <div class="card card-default card-borderless">
      <div class="card-body">
        <InputTodo @add-todo="addTodo"></InputTodo>
        <TodoList :todoList="state.todoList" @delete-todo="deleteTodo"
          @toggle-completed="toggleCompleted"></TodoList>
      </div>
    </div>
  </div>
</template>
```

1 TodoList 앱 리팩토링

src/App.vue

```
<script>
import { reactive, onMounted } from 'vue'
import InputTodo from './components/InputTodo.vue'
import TodoList from './components/TodoList.vue'

export default {
  name : "App",
  components : { InputTodo, TodoList },
  setup() {
    const ts = new Date().getTime();
    const state = reactive({ todoList : [] })

    onMounted(()=>{
      state.todoList.push({ id: ts, todo:"자전거 타기", completed: false })
      state.todoList.push({ id: ts+1, todo:"딸과 공원 산책", completed: true })
      state.todoList.push({ id: ts+2, todo:"일요일 애견 카페", completed: false })
      state.todoList.push({ id: ts+3, todo:"Vue 원고 집필", completed: false })
    })
  }
}
```

1 TodoList 앱 리팩토링

src/App.vue

```
const addTodo = (todo) => {  
  if (todo.length >= 2) {  
    state.todoList.push({ id: new Date().getTime(),  
      todo: todo, completed: false });  
  }  
}  
  
const deleteTodo = (id) => {  
  let index = state.todoList.findIndex((item) => id === item.id);  
  state.todoList.splice(index, 1);  
}  
  
const toggleCompleted = (id) => {  
  let index = state.todoList.findIndex((item) => id === item.id);  
  state.todoList[index].completed = !state.todoList[index].completed;  
}  
  
  return { state, addTodo, deleteTodo, toggleCompleted }  
}  
</script>
```

1 TodoList 앱 리팩토링

src/components/InputTodo.vue

```
<template>
...
</template>

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

export default {
  name: 'InputTodo',
  setup(props, context) {
    const todo = ref('');
    const addTodoHandler = () => {
      if (todo.value.length >= 3) {
        context.emit('add-todo', todo.value);
        todo.value = '';
      }
    };
    return { todo, addTodoHandler };
  },
};
</script>
```

2 <script setup> 사용하기

✓ Composition API 사용하기

○ 장점

- 적은 상용구 코드 사용으로 간결한 코드를 작성
- 순수 타입스크립트 언어를 사용해 props, 이벤트를 선언
- 런타임 성능이 더 좋음
- IDE에서의 타입 추론 성능이 더 뛰어남

2 <script setup> 사용하기

✓ 기존과 다른점

- 템플릿에서 사용하는 값
 - 최상위의 변수, 함수는 직접 템플릿에서 사용
- 컴포넌트 등록
 - import한 컴포넌트는 바로 템플릿에서 지역 컴포넌트로 사용
- 속성과 발신 이벤트 처리
 - defineProps, defineEmits 함수를 이용해 속성과 emit 함수를 생성

//기존 방식

```
setup(props, context) {  
  //이벤트를 발신할 때  
  context.emit('add-todo', todo)
```



// <script setup> 방식

```
const props = defineProps({  
  todoItem : { type : Object, required: true }  
})  
  
const emit = defineEmits(['delete-todo', 'toggle-completed'])  
//이벤트를 발신할 때는 다음과 같이  
emit('delete-todo', id)
```

<script setup> 사용하기

파일명

```
<template>
...
</template>
```

```
<script setup>
import { reactive, onMounted } from 'vue';
import InputTodo from './components/InputTodo.vue';
import TodoList from './components/TodoList.vue';

const ts = new Date().getTime();
const state = reactive({ todoList: [] });

onMounted(() => {
  state.todoList.push({ id: ts, todo: '자전거 타기', completed: false });
  state.todoList.push({ id: ts + 1, todo: '딸과 공원 산책', completed: true });
  state.todoList.push({ id: ts + 2, todo: '일요일 애견 카페', completed: false });
  state.todoList.push({ id: ts + 3, todo: 'Vue 원고 집필', completed: false });
});
```

<script setup> 사용하기

src/App.vue

```
const addTodo = (todo) => {  
  if (todo.length >= 2) {  
    state.todoList.push({  
      id: new Date().getTime(),  
      todo: todo,  
      completed: false,  
    });  
  }  
};  
  
const deleteTodo = (id) => {  
  let index = state.todoList.findIndex((item) => id === item.id);  
  state.todoList.splice(index, 1);  
};  
  
const toggleCompleted = (id) => {  
  let index = state.todoList.findIndex((item) => id === item.id);  
  state.todoList[index].completed = !state.todoList[index].completed;  
};  
</script>
```

<script setup> 사용하기

src/components/InputTodo.vue

```
<template>
...
</template>

<script setup>
import { ref } from 'vue';

const emit = defineEmits(['add-todo']);
const todo = ref('');

const addTodoHandler = () => {
  if (todo.value.length >= 3) {
    emit('add-todo', todo.value);
    todo.value = '';
  }
};
</script>
```

<script setup> 사용하기

src/components/ToDoList.vue

```
<template>
  <div class="row">
    <div class="col">
      <ul class="list-group">
        <ToDoListItem v-for="todoItem in todoList" :key="todoItem.id"
          :todoItem="todoItem"
          @delete-todo="emit('delete-todo', todoItem.id)"
          @toggle-completed="emit('toggle-completed', todoItem.id)"
        />
      </ul>
    </div>
  </div>
</template>

<script setup>
import ToDoListItem from './ToDoListItem.vue';

const props = defineProps({
  todoList: { type: Array, required: true },
});

const emit = defineEmits(['delete-todo', 'toggle-completed']);
</script>
```

<script setup> 사용하기

src/components/TodoListItem.vue

```
<template>
  <li
    class="list-group-item"
    :class="{ 'list-group-item-success': todoItem.completed }"
    @click="emit('toggle-completed', todoItem.id)" >
    <span class="pointer" :class="{ 'todo-done': todoItem.completed }">
      {{ todoItem.todo }} {{ todoItem.completed ? '(완료)' : '' }}
    </span>
    <span
      class="float-end badge bg-secondary pointer"
      @click.stop="emit('delete-todo', todoItem.id)"
      >삭제</span>
    </li>
</template>

<script setup>
const props = defineProps({
  todoItem: { type: Object, required: true },
});

const emit = defineEmits(['delete-todo', 'toggle-completed']);
</script>
```