

# Vue.js

## Day 4



발표자 고은경

# Contents

## ❖ Todo 앱 계속

기존 어플리케이션 구조의 문제점 해결

뷰 애니메이션

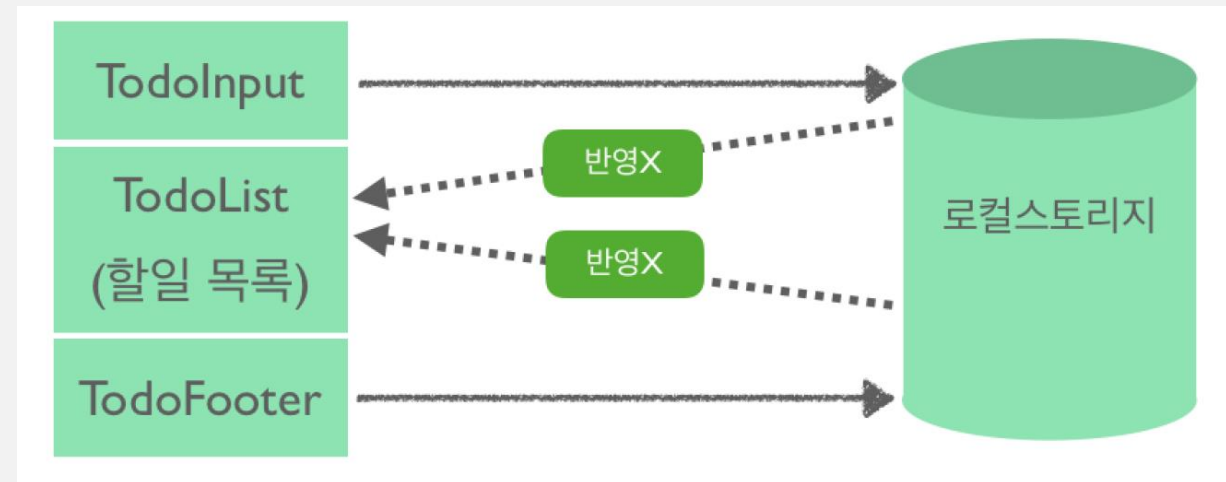
뷰 모달



# I 기존 어플리케이션 구조의 문제점 해결하기



- 문제: 할 일 추가와 모두 삭제가 바로 리스트에 반영이 안됨
- 해결
  - 데이터 조회, 추가, 삭제를 모두 App 컴포넌트에서 하게 만든다.
  - 하위 컴포넌트들은 데이터를 표현
  - 같은 데이터 속성을 사용하기 위해 최상위(루트) 컴포넌트인 App 컴포넌트에  
    `todoItems(데이터)` 정의
  - 하위 컴포넌트 `TodoList`에 props로 전달!!



# | 기존 어플리케이션 구조의 문제점 해결하기



## 1. 최상위 컴포넌트인 App 컴포넌트에 데이터 속성 todoItems 선언

TodoList.vue

```
export default {  
  data() {  
    return {  
      todoItems: []  
    }  
  },  
  created() {  
    if (localStorage.length > 0) {  
      for (var i=0; i<localStorage.length; i++) {  
        var key = localStorage.key(i)  
        if (key !== 'loglevel:webpack-dev-server') {  
          this.todoItems.push(key)  
        }  
      }  
    }  
  }  
},
```

App.vue

# 기존 어플리케이션 구조의 문제점 해결하기

## 1. 최상위 컴포넌트인 App 컴포넌트에 데이터 속성 todoItems 선언

App.vue

```
<template>
  <div id="app">
    <TodoHeader></TodoHeader>
    <TodoInput @add-todo="addTodo"></TodoInput>
    <TodoList @remove-todo="removeTodo" :propsItems="todoItems"></TodoList>
    <TodoFooter @clear-todo="clearTodo"></TodoFooter>
  </div>
</template>
```

```
<script>
import TodoHeader from "../components/ToDoHeader.vue"
import TodoInput from "../components/ToDoInput.vue"
import TodoList from "../components/ToDoList.vue"
import TodoFooter from "../components/ToDoFooter.vue"
```

```
export default {
  data() {
    return {
      todoItems: []
    }
  },
}
```

TodoList.vue

```
<template>
  <section>
    <transition-group name='list' tag="ul">
      <li v-for="(todoItem, index) in propsItems" :key="index" class="shadow">
        <i class="checkBtn fa fa-check" aria-hidden="true"></i>
        {{ todoItem }}
        <span class="removeBtn" @click="removeTodo(todoItem, index)">
          <i class="fa fa-trash-alt" aria-hidden="true"></i>
        </span>
      </li>
    </transition-group>
  </section>
</template>
```

```
<script>
export default {
  props: ['propsItems'],
}
```

# I 기존 어플리케이션 구조의 문제점 해결하기




## 2. 할 일 추가: 하위 -> 상위 이벤트 전달

TodoInput.vue

`localStorage.setItem(value, value)`

App.vue

```
addTodo() {  
  var value = this.newTodoItem && this.newTodoItem.trim()  
  if (this.newTodoItem !== '') {  
    localStorage.setItem(value, value)  
    this.clearInput()  
  }  
}
```



```
addTodo() {  
  var value = this.newTodoItem && this.newTodoItem.trim()  
  if (this.newTodoItem !== '') {  
    this.$emit('add-todo', value)  
    this.clearInput()  
  }  
}
```

```
addTodo(value) {  
  // 중복값이 입력되는 것을 방지  
  if (!this.todoItems.includes(value)) {  
    localStorage.setItem(value, value)  
    // key-value 형식이므로 하나는 key, 하나는 value  
    // Local Storage는 크롬개발자도구의 Application에서 확인  
    this.todoItems.push(value)  
  }  
}
```

## TodoInput.vue

## App.vue

```
<template>
  <div class="inputBox shadow">
    <input type="text" v-model="newTodoItem" v-on:keypress.enter
      placeholder="할 일을 입력하세요">
    <span class="addContainer" v-on:click="addTodo">
      <i class="addBtn fa fa-plus" aria-hidden="true"></i>
    </span>
  </div>
</template>

<script>
export default {
  methods: {
    addTodo() {
      var value = this.newTodoItem && this.newTodoItem.trim()
      if (this.newTodoItem !== '') {
        this.$emit('add-todo', value)
        this.clearInput()
      } else {
        this.showModal = true
      }
    },
    clearInput() {
      this.newTodoItem = ''
    },
  },
}
```

이벤트발생

```
<template>
  <div id="app">
    <TodoHeader></TodoHeader>
    <TodoInput @add-todo="addTodo"></TodoInput>
    <TodoList @remove-todo="removeTodo" :propsItems="todoItems">
    <TodoFooter @clear-todo="clearTodo"></TodoFooter>
  </div>
</template>

<script>
import TodoHeader from "../components/ToDoHeader.vue"
import TodoInput from "../components/ToDoInput.vue"
import TodoList from "../components/ToDoList.vue"
import TodoFooter from "../components/ToDoFooter.vue"

export default {
  methods: {
    addTodo(value) {
      // 중복값이 입력되는 것을 방지
      if (!this.todoItems.includes(value)) {
        localStorage.setItem(value,value)
        // key-value 형식이므로 하나는 key, 하나는 value
        // Local Storage는 크롬개발자도구의 Application에서
        this.todoItems.push(value)
      }
    },
  },
}
```

# | 기존 어플리케이션 구조의 문제점 해결하기



## 3. 할 일 삭제: 하위 -> 상위 이벤트 전달

TodoList.vue

```
removeTodo(key, index) {  
  localStorage.removeItem(key)  
  this.todoItems.splice(index, 1)  
  // 한 개 지울 것  
},
```

App.vue



# 기존 어플리케이션 구조의 문제점 해결하기

## 3. 할 일 삭제: 하위 -> 상위 이벤트 전달

### TodoList.vue

```
<script>
export default {
  props: ['propsItems'],
  methods: {
    removeTodo(key, index) {
      this.$emit('remove-todo', key, index)
    }
  }
}
</script>
```

이벤트발생

#### \$emit 형식

- \$emit( '이벤트 이름' )
- \$emit( '이벤트 이름' , 인자 1, 인자 2)

### App.vue

```
<template>
  <div id="app">
    <TodoHeader></TodoHeader>
    <TodoInput @add-todo="addTodo"></TodoInput>
    <TodoList @remove-todo="removeTodo" :propsItems="todoItems"></TodoList>
    <TodoFooter @clear-todo="clearTodo"></TodoFooter>
  </div>
</template>

<script>
import TodoHeader from "../components/ToDoHeader.vue"
import TodoInput from "../components/ToDoInput.vue"
import TodoList from "../components/ToDoList.vue"
import TodoFooter from "../components/ToDoFooter.vue"

export default {
  removeTodo(key, index) {
    localStorage.removeItem(key)
    this.todoItems.splice(index,1)
    // 한 개 지울 것
  },
}
```

# | 기존 어플리케이션 구조의 문제점 해결하기



## 3. 할 일 목록 모두 삭제: 하위 -> 상위 이벤트 전달

TodoFooter.vue

```
clearTodo() {  
  |   localStorage.clear()  
  |   this.todoItems = []  
  |  
}
```

App.vue

# 기존 어플리케이션 구조의 문제점 해결하기

## 3. 할 일 목록 모두 삭제: 하위 -> 상위 이벤트 전달

TodoFooter.vue

App.vue

```
<template>
  <div class="clearAllContainer">
    <span class="clearAllBtn" @click="clearTodo">
  </div>
</template>

<script>
export default {
  methods: {
    clearTodo() {
      // localStorage.clear()
      this.$emit("clear-todo")
    }
  }
}
</script>
```

이벤트발생

```
<template>
  <div id="app">
    <TodoHeader></TodoHeader>
    <TodoInput @add-todo="addTodo"></TodoInput>
    <TodoList @remove-todo="removeTodo" :propsItems="todoItems"></TodoList>
    <TodoFooter @clear-todo="clearTodo"></TodoFooter>
  </div>
</template>

<script>
import TodoHeader from "../components/ToDoHeader.vue"
import TodoInput from "../components/ToDoInput.vue"
import TodoList from "../components/ToDoList.vue"
import TodoFooter from "../components/ToDoFooter.vue"

export default {
  clearTodo() {
    localStorage.clear()
    this.todoItems = []
  }
}
```

# I 뷰 애니메이션



- 뷰 프레임워크 자체에서 지원하는 애니메이션 기능
- 데이터 추가, 변경, 삭제에 대해 fade in, fade out 등 여러 애니메이션 효과 지원

TodoList.vue 의 <template> 코드 변경

```
<transition-group name='list' tag="ul">  
  <li v-for="(todoItem, index) in propsItems" :key="index" class="shadow">  
    <i class="checkBtn fa fa-check" aria-hidden="true"></i>  
    {{ todoItem }}  
    <span class="removeBtn" @click="removeTodo(todoItem, index)">  
      <i class="fa fa-trash-alt" aria-hidden="true"></i>  
    </span>  
  </li>  
</transition-group>
```

# | 뷰 애니메이션



## TodoList.vue 의 <style> 코드 변경

동일한 속성값을 갖는 여러 클래스를  
한 번에 표시할 수 있음

입력 시에는 밑에서 올라오게,  
삭제 시에는 오른쪽으로 사라지게 설정

```
.list-item {  
  display: inline-block;  
  margin-right: 10px;  
}
```

```
.list-move {  
  transition: transform 1s;  
}
```

```
.list-enter-active, .list-leave-active {  
  transition: all 1s;  
}
```

```
.list-enter {  
  opacity: 0;  
  transform: translateY(30px);  
}
```

```
.list-leave-to {  
  opacity: 0;  
  transform: translateX(30px);  
}
```

# I 뷰 모달



- 텍스트 입력 값이 없을 때의 예외처리하기
- 뷰 모달: 뷰 공식사이트에서 제공하는 팝업 대화상자

TodoInput.vue의 <template> 태그

```
<modal v-if="showModal" @close="showModal = false">
  <h3 slot="header">경고</h3>
  <span slot="footer" @click="showModal = false">
    | 할 일을 입력하세요.
    | <i class="closeModalBtn fa fa-times" aria-hidden="true"></i>
  </span>
</modal>
```

# 뷰 모달



```
<script>
import Modal from './common/Modal.vue'

export default {
  data() {
    return {
      newTodoItem: '',
      showModal: false
    }
  },
  methods: {
    addTodo() {
      var value = this.newTodoItem && this.newTodoItem.trim()
      if (this.newTodoItem !== '') {
        this.$emit('add-todo', value)
        this.clearInput()
      } else {
        this.showModal = true
      }
    },
    clearInput() {
      this.newTodoItem = ''
    },
  },
  components: {
    modal: Modal
  }
}
```

Modal.vue 불러오기

모달 동작을 위한 플래그 값

텍스트 미입력 시 모달 동작

모달 컴포넌트 등록

TodoInput.vue의  
<script> 태그

# 뷰 모달



<style> 태그가 scoped 속성을 가지고있을 때, CSS는 현재 컴포넌트의 엘리먼트에만 적용된다.

## Modal.vue

```
<template>
  <transition name="modal">
    <div class="modal-mask" @keyup.esc="$emit('close')">
      <div class="modal-wrapper">
        <div class="modal-container">

          <div class="modal-header">
            <slot name="header">
            </slot>
          </div>

          ...

          <style scoped>
          ...
          .modal-header h3 {
            margin-top: 0;
            color: #42b983;
          }
          ...
        </div>
      </div>
    </div>
  </transition>
</template>
```

<style scoped>

TODO it!

할 일을 입력하세요



✓ b



✓ d



✓ c



경고

할 일을 입력하세요. ✕

<style>

TODO it!

할 일을 입력하세요



✓ b



✓ d



✓ c



경고

할 일을 입력하세요. ✕





**Thank You**