



Frontend Framework

Vue.js

비즈니스서비스개발팀

김설한

• 목차

- ✓ Vue.js 소개
- ✓ 뷰 컴포넌트 (Vue Component)
- ✓ 뷰 라이브러리 (Vue Library)
- ✓ 간단 예제 : TO-DO 앱

01

Vue.js 소개

Vue.js 소개

1) Vue.js란?



- 사용자 인터페이스를 만들기 위한 프레임워크
- **컴포넌트 기반** 프레임워크로 협업에 유리
- 화면 요소가 변경되거나 조작이 일어날 때,
즉각 반응하여 데이터를 갱신(**reactivity**)
- **양방향 데이터 바인딩**과 **단방향 데이터 흐름**의
장점을 모두 결합한 프레임워크
- 기능을 **쉽게 결합(progressive)**할 수 있도록 설계

Vue.js 소개

2) Vue 인스턴스

- Vue 인스턴스 속성 : data, methods, computed, watched, 라이프사이클 혹 등
- Data 속성 : Vue의 반응성에 추가되어, 변경 시 감지되 재 랜더링 된다.
- Computed 속성 : Data 속성값을 이용해 연산하거나, 처리할 때 사용.

지정한 Data값이 변경될 때만 종속되어 실행된다.

```
data: {  
  message: '안녕하세요'  
},  
computed: {  
  reversedMessage: function () {  
    return this.message.split('').reverse().join('')  
  }  
}
```

- Watched 속성 : Data 속성값이 변경될 때를 감지할 때 사용.

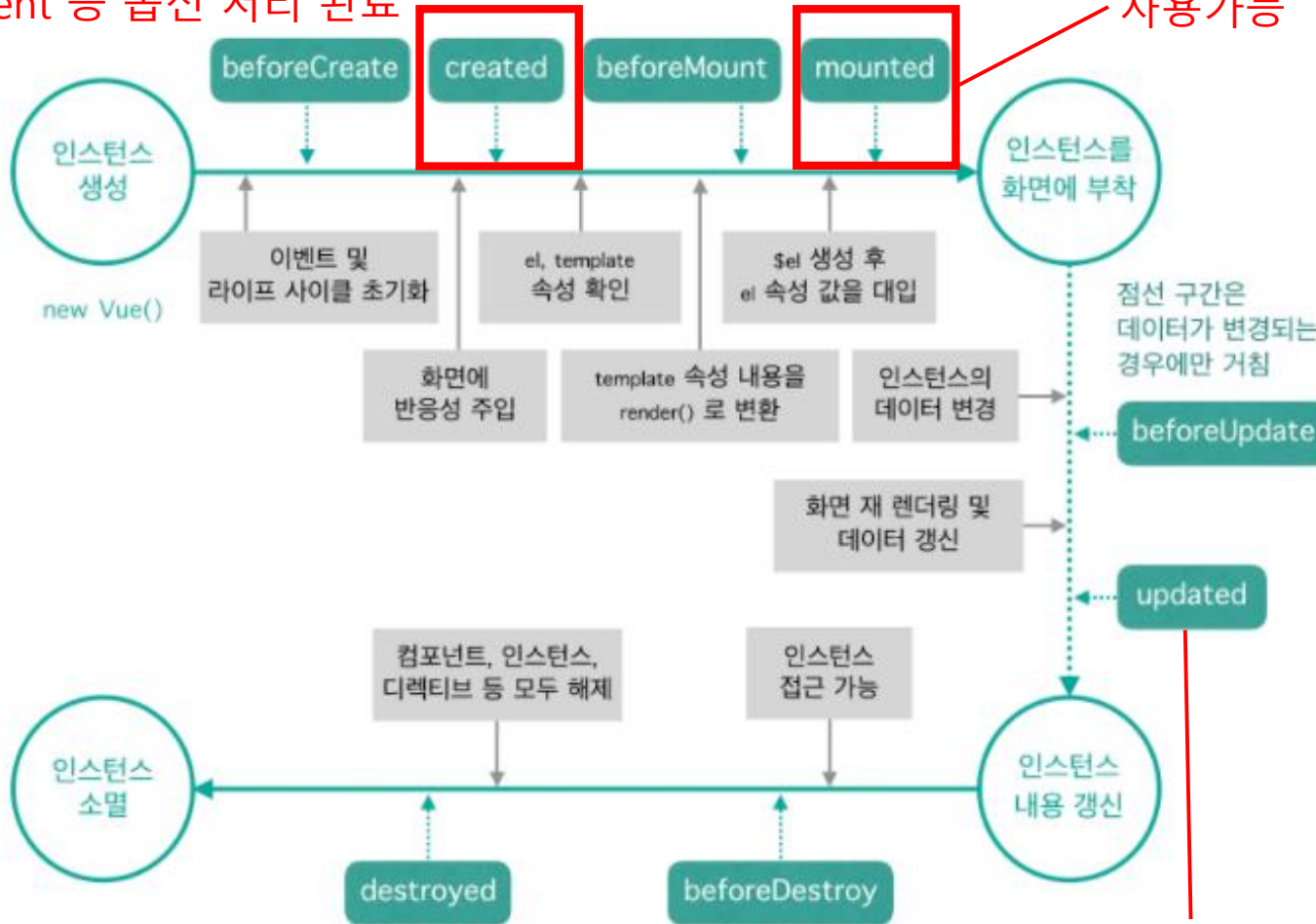
Data 속성값을 처리하지않고, 변화만 감지 할 때 주로 사용. (ex: API 호출 감지)

Vue.js 소개

2) 라이프 사이클 (Life Cycle)

데이터, computed, method, watcher, event 등 옵션 처리 완료

\$el 속성(DOM요소) 사용가능



단순 변경 감지는 computed/watched 사용

- 인스턴스 상태에 따라 호출할 수 있는 속성들
- 라이프사이클 훅(hook) :
각 라이프사이클에서 실행되는 커스텀 로직

example)

```
mounted() {  
  this.searchIntegrationInfo();  
},
```

Vue.js 소개

3) 뷰의 반응성 (Vue Reactivity)

- 뷰가 데이터 변화를 감지했을 때, 자동으로 화면을 다시 갱신하는 특성
- .NET 프레임워크를 사용해 데이터를 Getter/Setter로 변환하여, 내부적으로 속성 변경을 감지
- 데이터가 변경된 부분을 자동으로 감지해, 재 렌더링 하기에 유용하지만 주의사항이 있다.

1) Object의 속성 추가/삭제 감지 불가

X `ObjectA.name = 'hello'`

O `Vue.set(ObjectA, 'name', 'hello')`

2) Array 인덱스 항목 직접 수정 시, 감지 불가

X `this.items[indexOfItem] = value`

O `this.items.splice(indexOfItem, 1, value)`

3) 배열의 길이를 수정하는 경우

X `this.items.length = newLength`

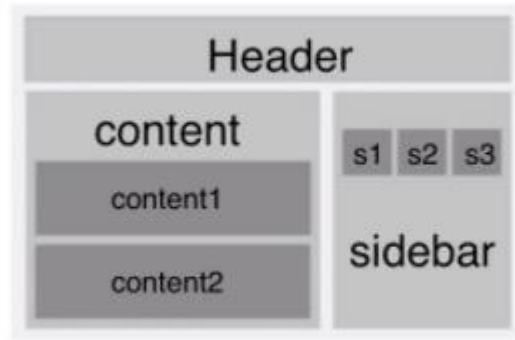
O `this.items.splice(newLength`

02

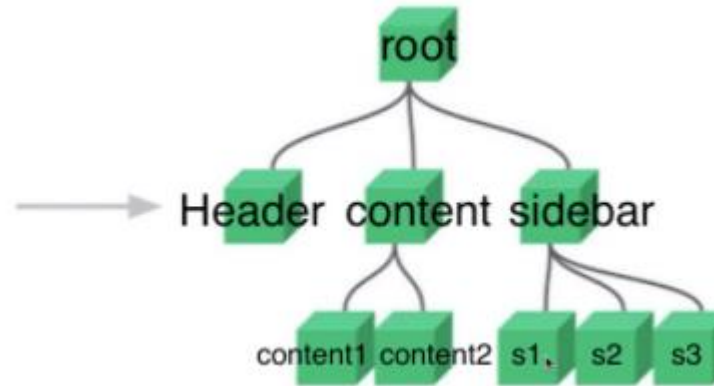
뷰 컴포넌트(Component)

뷰 컴포넌트

1) 컴포넌트(Component)



[그림1]



[그림2]

- 컴포넌트(component) : 조합하여 화면을 구성하는 블록
- [그림1]처럼 컴포넌트로 화면 영역이 구조화 되고, [그림2]와 같은 관계가 형성된다.
- 뷰는 컴포넌트로 HTML 화면을 직관적으로 파악할 수 있게 한다.

이로 인해 재 사용성이 높아지며, 다른 사람의 코드를 이해하기도 수월해 진다.

뷰 컴포넌트

2) 싱글 파일 컴포넌트(Single File Components)

- .vue 파일로 프로젝트 구조를 구성하는 방식
- .vue 파일은 1개의 뷰 애플리케이션을 구성하는 1개의 컴포넌트와 동일하다.
- Vue CLI(command line interface) : 싱글 파일 컴포넌트 프로젝트 구성 편리하게 하는 도구

```
1  <!-- 화면에 표시할 요소들을 정의 -->
2  <template>
3    <!-- HTML 태그 내용 -->
4    <div> hello </div>
5  </template>
6  <!-- 뷰 컴포넌트 내용을 정의하는 영역 -->
7  <script>
8    export default {
9      // 자바 스크립트 내용-
10    }
11  </script>
12  <!-- template에 추가한 HTML태그의 CSS 스타일을 정의하는 영역 -->
13  <style>
14    /* CSS 스타일 내용 */
15  </style>
```

[*.vue]

03

뷰 라이브러리 (Library)

뷰 라이브러리

1) 뷰 라우터(Vue Router)

- 라우팅 : 웹 페이지 간 이동 방법. SPA에서 주로 사용.
- 뷰 라우터 : 뷰에서 라우팅 기능을 구현할 수 있도록 하는 공식 라이브러리

```
var UserProfile = { template: '<p>User Profile Component</p>' };  
var UserPost = { template: '<p>User Post Component</p>' };
```

```
var routes=[  
  {  
    path: '/user',  
    component: User,  
    children: [  
      {  
        path: 'posts',  
        component: UserPost  
      },  
      {  
        path: 'profile',  
        component: UserProfile  
      }  
    ]  
  }  
];  
var router = new VueRouter({  
  routes  
});
```

path : 변경될 url

component : path url에 뿌려질 컴포넌트

- 설치는 npm 명령어로 하고,
Vue.use()로 명시적으로 라우터를 추가해 사용한다.

```
npm install vue-router
```

```
import Vue from 'vue'  
import VueRouter from 'vue-router'  
  
Vue.use(VueRouter)
```

뷰 라이브러리

2) 엑시오스(Axios)

- 엑시오스 : 뷰 커뮤니티에서 가장 많이 사용하는 HTTP 통신 라이브러리.
- HTTP 요청에 대한 상세 속성들(URL, 요청방식, 데이터유형 등등)을 정의해 보낼 수 있다.

```
axios.get('url').then().catch();
```

```
axios.post('url').then().catch();
```

- then(), catch()로 API통신 성공/실패에 따른 로직 처리가 가능하다.

```
axios.get(`https://raw.githubusercontent.com/joshua1988/dojo-vuejs/master/data/demo.json`)  
  .then((response) => console.log(response))  
  .catch((error) => console.log(error));
```

- (+) javascript와 동일하게, 비동기 순차 처리는 async-await로 순서를 보장할 수 있다.

```
async testFunction() {  
  const response = await axios.get('api/test')  
  console.log(response)  
  const response2 = await axios.get('api/test')  
  console.log(response2)  
}
```

뷰 라이브러리

3) 뷰엑스(Vuex)

- 뷰엑스 : 애플리케이션 상태 관리 패턴 라이브러리.
- 모든 데이터를 중앙에서 관리하는 기능. (공통으로 사용하는 데이터, 메서드 등 정의)
- 뷰엑스는 4가지 속성(State, Getters, Mutations, Actions)을 가지고있다.

```
state: { /* this.$store.state.[변수명] */  
  name : 'suzy',  
  friendsList: [],  
},  
getters: { /* this.$store.getters.[함수명] */  
  getName(state){  
    return state.name;  
  }  
},  
mutations: { /* this.$store.commit.([함수명],[매개변수]) */  
  setName(state, value){  
    state.name = value;  
  }  
  setFriendsList(state, value){  
    state.friendsList = value;  
  },  
}  
actions: { /* this.$store.dispatch.([함수명],[매개변수]) */  
  async GET_FRIENDS_LIST({ commit }, params) {  
    const { data } = await getFriendsList(params);  
    commit('setFriendsList', data);  
  },  
}
```

1) state (== data)

2) getters (== data getter)

3) mutations (== data setter)

4) actions (== 비동기 method)

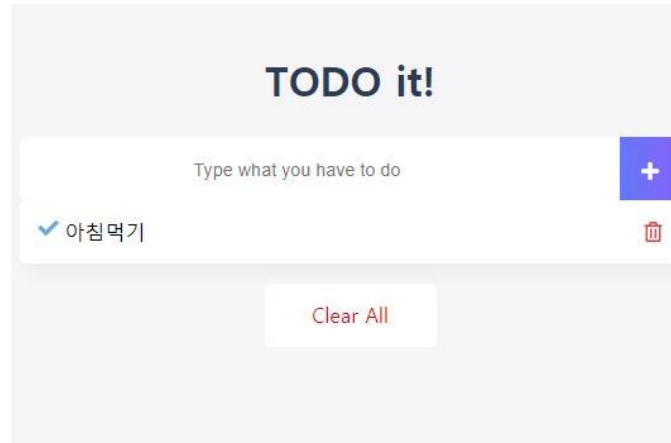
04

간단 예제 : TO-DO 앱

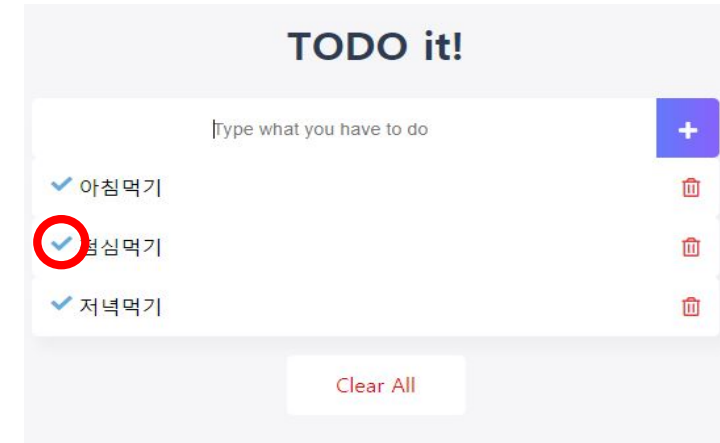
간단 예제: TO-DO앱

1) 소개

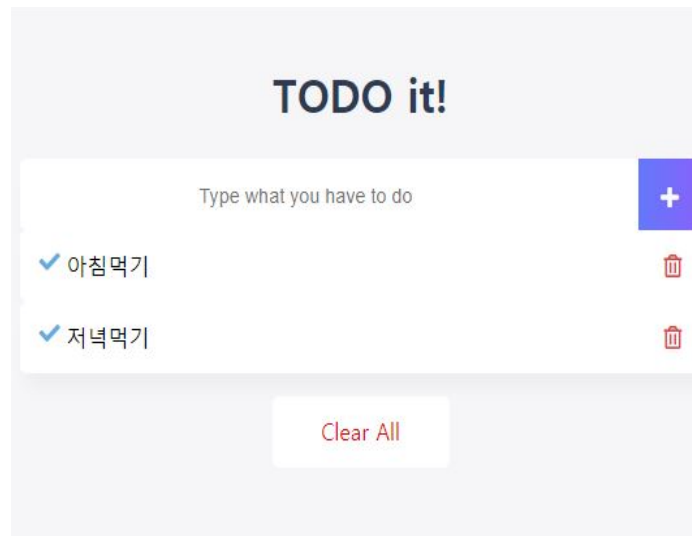
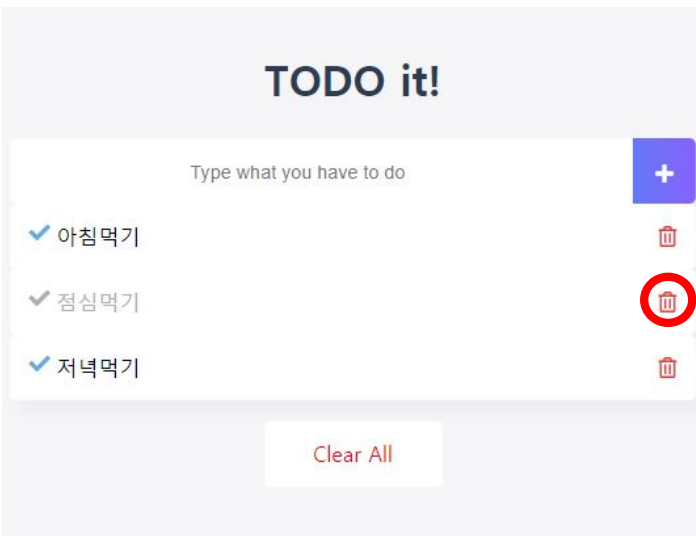
1) 할일 입력 & 추가



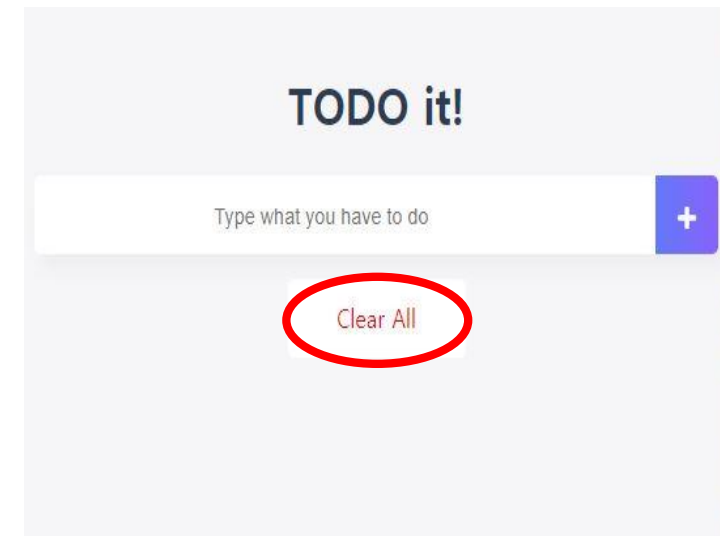
2) 할일 완료



3) 완료된 일 삭제



4) 전체 삭제



간단 예제: TO-DO앱



2) 화면 컴포넌트 구성

화면 구성
Component 분리

```
1 <template>
2   <div id="app">
3     <Todoheader ></Todoheader>
4     <Todoinput></Todoinput>
5     <Todolist></Todolist>
6     <Todofooter></Todofooter>
7   </div>
8 </template>
9
10 <script>
11   import Todoheader from './components/Todoheader.vue'
12   import Todoinput from './components/Todoinput.vue'
13   import Todolist from './components/Todolist.vue'
14   import Todofooter from './components/Todofooter.vue'
15
16   export default {
17     components: {
18       Todoheader,
19       Todoinput,
20       Todolist,
21       Todofooter
22     }
23   }
24 </script>
```

간단 예제: TO-DO앱

3) Vuex 데이터 구성

```
src > store > modules >  todoApp.js >  default
1  const state = {
2    |   TodoItems : []
3  | }
4  const getters = {
5    |   storedTodoItems(state){
6    |     |   return state.TodoItems;
7    |   }
8  | };
9  const mutations = {
10 |   addTodo(state, newTodo) {
11 |     |   state.TodoItems.push(newTodo);
12 |   },
13 |   removeTodo (state, removeIndex){
14 |     |   state.TodoItems.splice(removeIndex,1);
15 |   },
16 |   toggleTodo(state, toggleIndex){
17 |     |   state.TodoItems[toggleIndex].completed = !state.TodoItems[toggleIndex].completed;
18 |   },
19 |   clearTodo(state){
20 |     |   state.TodoItems = [];
21 |   }
22 | };
23 | };
24 | export default{
25 |   |   state,
26 |   |   getters,
27 |   |   mutations
28 | }
```

TO-DO 앱

4) 컴포넌트 - Todoheader

```
src > components > ▼ Todoheader.vue > {} "Todoheader.vue"
1  <template>
2    <div>
3      <h1>TODO it!</h1>
4    </div>
5  </template>
6
7  <script>
8    export default {
9
10   }
11 </script>
12
13 <style scoped>
14   h1{
15     color: #2F3B52;
16     font-weight: 900;
17     margin: 2.5rem 0 1.5rem;
18   }
19 </style>
```

간단 예제: TO-DO앱

4) 컴포넌트 - Todoinput

src > components > Todoinput.vue > {} "Todoinput.vue" > template

```
1 <template>
2   <div class="inputBox shadow">
3     <input v-on:keyup.enter="addTodo" type="text" v-model="newTodoItem" placeholder="Type what you have to do">
4     <span class="addContainer" v-on:click="addTodo">
5       <i class="addBtn fas fa-plus" aria-hidden="true"></i>
6     </span>
7
8     <Modal v-if="showModal" v-on:close="showModal = false">
9       <h3 slot="header"> 경고 ! </h3>
10      <h3 slot="body">
11        <i class="closeModalBtn fas fa-times" v-on:click="closeModal"></i>
12      </h3>
13    </Modal>
14
15  </div>
16 </template>
17 <script>
18   import Modal from './common/Modal.vue'
19   export default {
20     data(){
21       return {
22         newTodoItem : '',
23         showModal: false
24       }
25     },
26     methods : {
27       addTodo(){
28         if(this.newTodoItem !== null && this.newTodoItem !== ''){
29           const newTodo = {completed: false, item: this.newTodoItem};
30           this.$store.commit('addTodo', newTodo);
31           this.clearInput();
32         } else {
33           this.showModal = !this.showModal;
34         }
35       },
36       clearInput(){
37         this.newTodoItem = '';
38       },
39       closeModal(){
40         this.showModal = false;
41       }
42     },
43     components: {
44       Modal
45     }
46   }
47 </script>
```

간단 예제: TO-DO앱

4) 컴포넌트 - Todolist

src > components > Todolist.vue > {} "Todolist.vue"

```
1  <template>
2    <ul>
3      <li v-for="(TodoItem,index) in storedTodoItems" :key="TodoItem.item" class="shadow">
4        <i class="checkBtn fas fa-check" v-bind:class = "{checkBtnCompleted: TodoItem.completed}" v-on:click="toggleTodo(index)"></i>
5        <span v-bind:class="{textCompleted: TodoItem.completed}">
6          {{ TodoItem.item }}</span>
7        <span class="removeBtn" type="button" v-on:click="removeTodo(index)">
8          <i class="far fa-trash-alt" aria-hidden="true"></i>
9        </span>
10     </li>
11   </ul>
12 </template>
13
14 <script>
15 import { mapGetters, mapMutations } from 'vuex'
16
17 export default {
18   methods : {
19     ...mapMutations(['removeTodo', 'toggleTodo'])
20   },
21   computed: {
22     ...mapGetters(['storedTodoItems'])
23   }
24 }
25 </script>
```

간단 예제: TO-DO앱

4) 컴포넌트 - Todofooter

```
src > components > ▼ Todofooter.vue > {} "Todofooter.vue" > template
1  <template>
2    <div class="clearAllContainer">
3      <span class="clearAllBtn" v-on:click="clearTodo">Clear All</span>
4    </div>
5  </template>
6
7  <script>
8    import { mapMutations } from 'vuex'
9
10   export default {
11     methods : {
12       ...mapMutations(['clearTodo'])
13     }
14   }
15 </script>
```

감사합니다 !