

Vue 정리

사용법

- CDN방식으로 Vue를 사용하기 위한 script 태그 작성
- (Vue CLI 방식은 내일)

```
<html>
  <head>
    ...
    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
  </head>
  ...
```

- Vue에서 제공하는 여러가지 함수를 **구조 분해 할당** 방식으로 가져온다.

```
const { createApp, ref, computed, watch } = Vue;
```

createApp

| Vue App을 만들기 위한 함수, 인자로 객체를 받는다

- DOM 요소에 mount되는 객체
- Vue App이 요소를 동적으로 제어할 수 있도록, app의 최상위 DOM element와 **mount** 해야한다.

```
const app = createApp(
{
  setup: function setup() {
  }
}
);

app.mount("#app"); // app 이라는 id를 가진 DOM element(HTML 요소)와 Vue app을 mount
```

인자로 들어가는 객체에 syntactic sugar를 적용하여 아래와 같이 작성할 수 있다.

```
const app = createApp({
  setup() {
  }
});

app.mount("#app"); // app 이라는 id를 가진 DOM element(HTML 요소)와 Vue app을 mount
```

setup

composition API를 작성하기 위한 진입점 역할을 하는 함수

- setup() 함수 내에서 여러 변수를 선언하고 객체를 할당하거나 함수를 정의할 수 있다.
 - DOM element 와 동적으로 **bind** 되어야 하는 변수는 setup함수의 return 객체 안에 담아준다.

```
const app = createApp({
  setup() {
    const message = ref("안녕하세요");

    function myFunction {
      message.value = "Hello";
      console.log(message.value);
    }

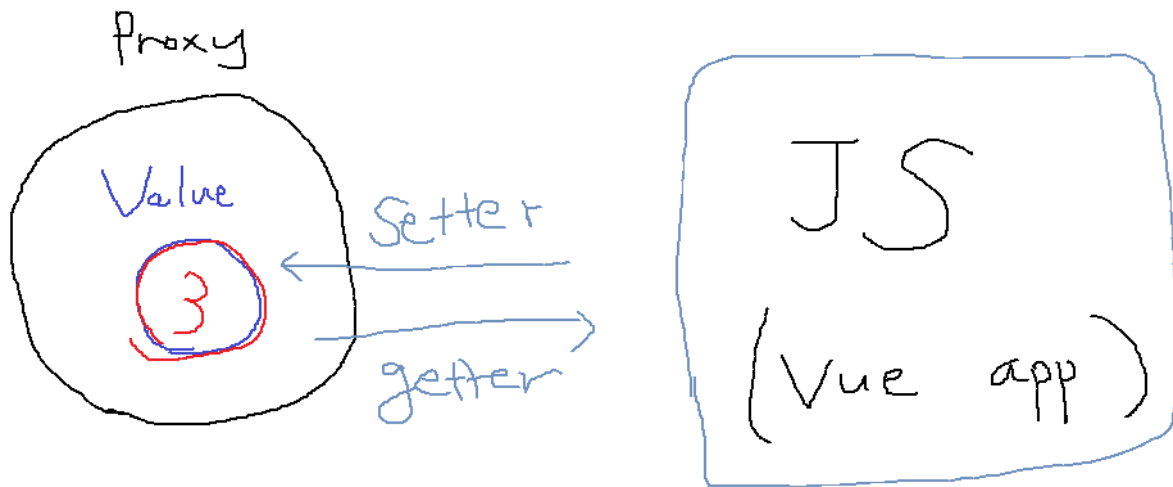
    return {
      message,
      myFunction,
    }
  }
}).mount("#app");
```

반응형 객체

- 값이 바뀌면 DOM element도 같이 바뀌는 객체
- DOM element와 Javascript를 바인딩 한다고 표현한다

ref

DOM element와 동적으로 바인드되는 반응형 객체를 만들기 위해서는 값이나 객체를 감싸주는 Proxy가 필요한데, 이를 반환해주는 역할



- 실제 값은 Proxy객체의 value 프로퍼티에 담겨있음
- value에 접근(get, set)할 때마다 DOM 영역도 같이 바뀜

computed

- Javascript의 연산 결과를 반환하는 반응형 객체를 만들 수 있다
- 일반 함수와 역할은 동일하지만 값의 변동이 없다면 함수가 재실행되지 않고 캐싱된 값을 이용한다
- computed 함수는 콜백함수를 인자로 받으며 해당 콜백함수가 return하는 값을 바인딩한다.
- template 보간법으로는 단순한 계산만 가능하므로 복잡한 계산이 필요하다면 computed를 이용하면 된다.

```
<p>{{ myObj1 + myObj2 }}</p>
```

```
const app = createApp({
  setup() {
    const obj1 = ref("asd");
    const obj2 = ref("asd");

    const computeObj = computed(() => {
```

```

        return obj1.value + obj2.value;
    });
    return {
        message,
        myFunction,
    }
}
}).mount("#app");

```

watch

반응형 객체의 값이 바뀔 때 실행될 하는 함수를 정의

- 반응형 객체를 감시하고 있다가 값이 바뀌면 수행

```
watch(반응형 객체, 반응형 객체가 바뀔 때 실행될 함수);
```

```


const message = ref("asdf");

watch(message, (newValue, oldValue) => {
    console.log("message의 값이", oldValue, "에서", newValue, "로 변경되었습니다.");
});



```

각종 v-directive(지시자)

v-bind

-  로 간략화
- DOM element와 동적으로 바인딩될 객체를 지정하는 지시자

v-on

-  로 간략화 ()
- addEventListener의 역할을 대체하는 지시자
- event가 발생할 조건과 콜백함수 지정

```
<button @click="myFunction">함수실행</button>
```

v-model

- 양방향 바인딩을 위한 지시자
- 반응형 객체의 값이 DOM element에 반영될 뿐만 아니라, DOM element에 사용자가 입력한 값이 반응형 객체에 반영됨
- 사용자가 입력이 가능한 요소에만 사용 가능

v-if

- DOM element의 렌더링 여부를 결정하는 지시자
- boolean 값을 지정해줌

v-show

- DOM element의 가시성 여부를 결정하는 지시자
- boolean 값을 지정해줌

💡 v-if와 v-show의 차이

v-if가 false인 경우 DOM element 자체가 없는 것으로 취급

v-show가 false인 경우 DOM element는 존재하지만 사용자에게 보이지 않음

v-for

- 반복되는 형태의 DOM element를 동적으로 만들 수 있는 지시자

```
<div v-for="(student, index) in students">
  <p>{{index + 1}} : {{student}}</p>
</div>
```

- 각각의 요소를 식별할 수 있는 key를 사용하는 것이 권장된다
 - list가 업데이트 되었을 때, 기존 element 위치를 바탕으로 효율적인 rendering이 가능하다

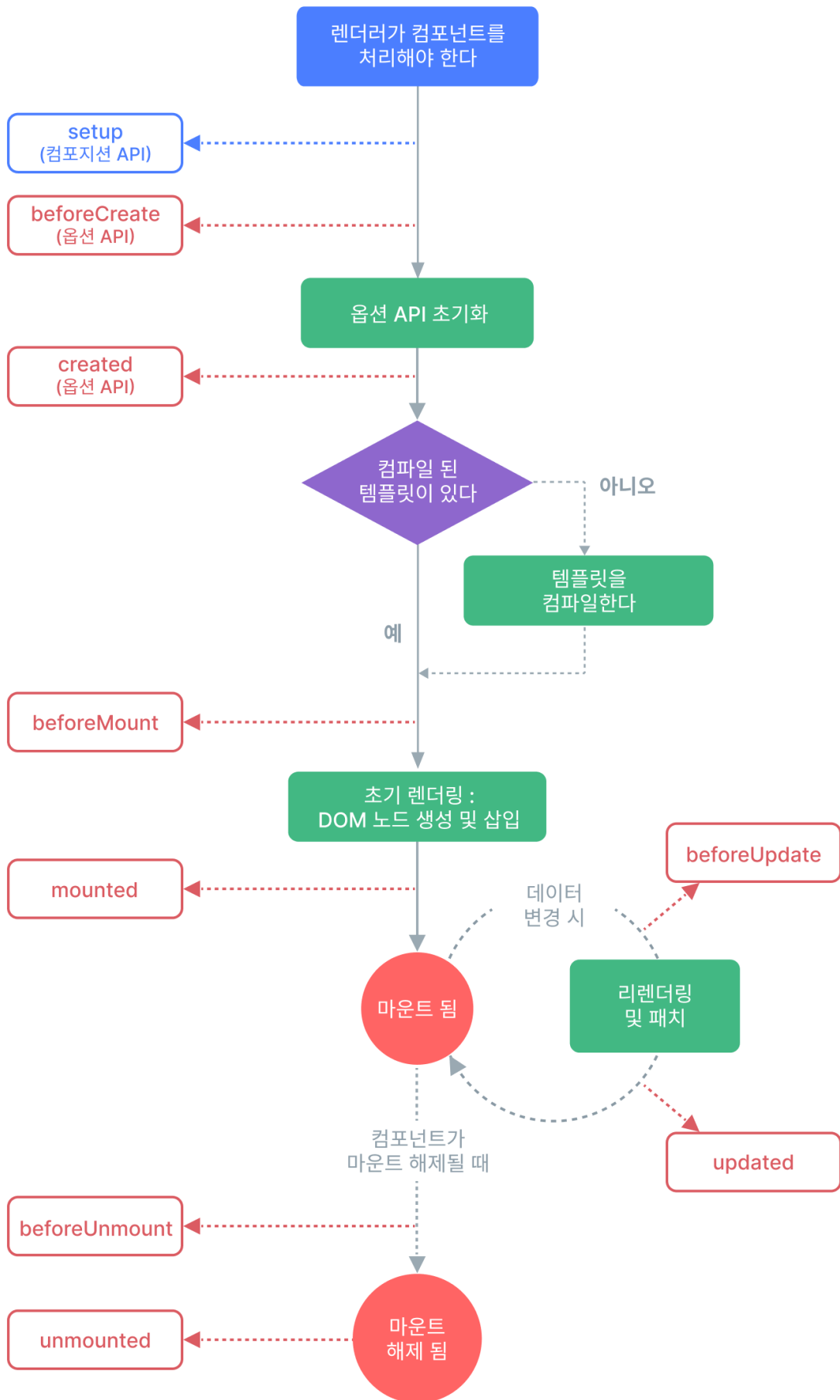
- index를 사용하거나, 객체별 고유한 값을 지정해준다

```
<div v-for="student in students" :key="student.id">  
  <p>{{student}}</p>  
</div>
```

```
<div v-for="(student, index) in students" :key="index">  
  <p>{{student}}</p>  
</div>
```

Life Cycle Hooks

Vue App은 하나 이상의 Vue Component로 이루어져 있으며 각각의 Vue Component는 DOM element와 mount되는 javascript 객체이며, 객체의 인스턴스는 생성부터 소멸까지의 생명 주기를 가진다. 이러한 생명 주기 중 특정 시점에 수행될 동작을 정의할 수 있다



생명주기 훅

- onMounted()
 - vue component가 생성되어 처음으로 DOM element에 mount되었을 때 수행될 함수
- onUpdated()
 - mount된 이후 반응형 객체의 값이 변경되어 DOM element에 변동이 생겼을 때 수행될 함수

이러한 생명주기 훅은 `setup()` 함수 내에서 일반 함수를 정의하는 것처럼 사용할 수 있다