프론트엔드 개발자를 위한 Vue.js 시작하기

Vue.js 시작하기

Component Lifecycle과 Props Data

한국기술교육대학교
온라인평생교육원

학습내용

- Component Lifecycle
- Props Data

학습목표

- Component Lifecycle에 대해 설명할 수 있다.
- Props Data를 이용해 상위 Component 데이터 활용을 할 수 있다.

Component Lifecycle

Component Lifecycle

1 Component Lifecycle

- Component는 생성부터 소멸까지 일련의 Lifecycle을 거침
- Create, Mount, Update, Destroy 단계로 나누어짐

1. Create 단계
2. Mount 단계
3. Update 단계
4. Destroy 단계

Component Lifecycle

1 Component Lifecycle

- 각 단계에서의 필요한 작업을 위해 지정된 함수가 호출됨

```
Vue.createApp({
  data() {
    return { count: 1 }
  },
  created() {
    console.log('count is: ' + this.count)
  }
})
```

Component Lifecycle

Component Lifecycle

2 Create 단계

- Component가 생성되는 단계이며 DOM에 추가되지 않은 단계임
- beforeCreate() 함수와 created() 함수를 선언할 수 있음

Component Lifecycle

2 Create 단계

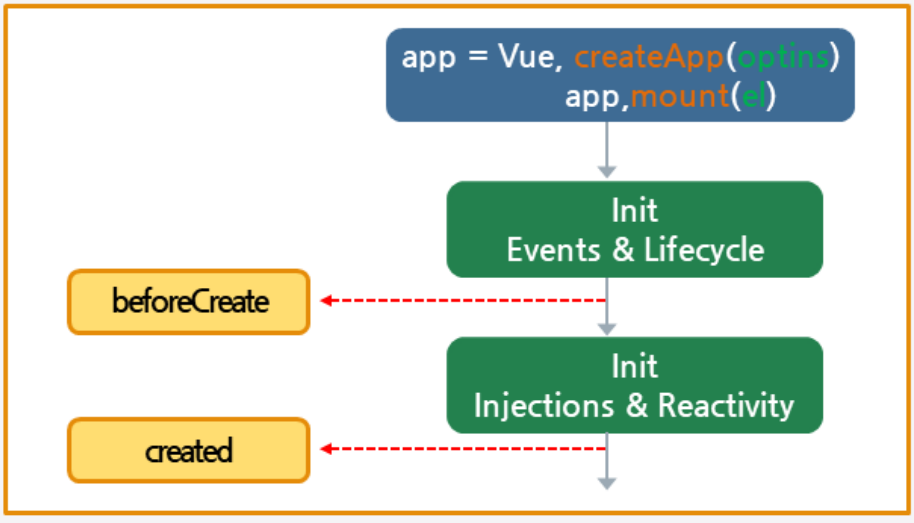
beforeCreate()	data와 events가 준비되지 않은 상태
created()	data, computed, methods, watch 등이 준비된 상태

Component Lifecycle

Component Lifecycle

2 Create 단계

- Created 단계에서는 Mount가 되지 않은 상태이므로 $el을 사용할 수 없음



Component Lifecycle

3 Mount 단계

- Component의 DOM을 준비하여 화면에 출력되는 단계
- beforeMount(), mounted() 함수가 호출됨

Component Lifecycle

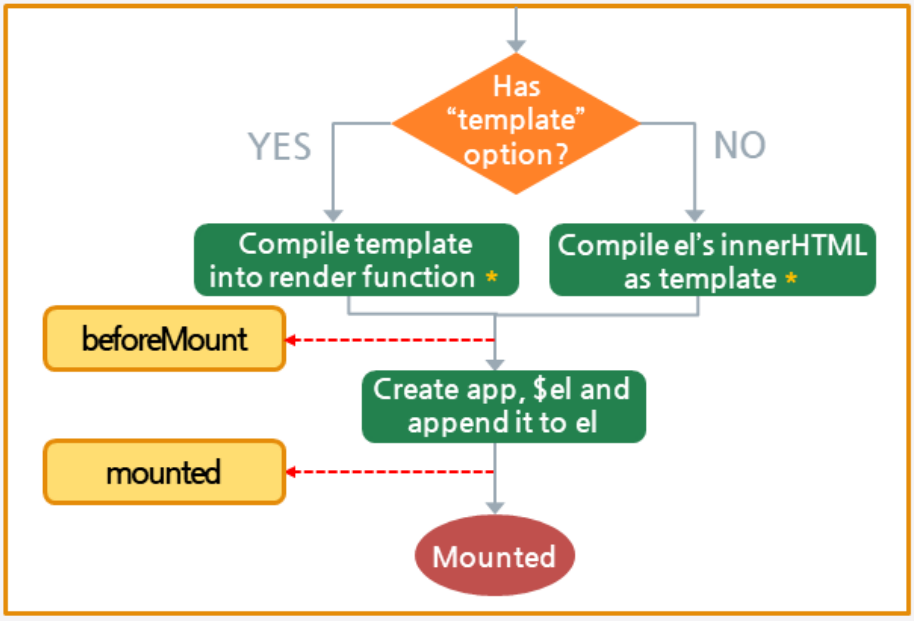
Component Lifecycle

3 Mount 단계

beforeMount()	Mount가 시작되기 직전에 호출됨
mounted()	• Mount가 완료된 후 호출됨 • $el로 Component의 DOM에 접근할 수 있음

Component Lifecycle

3 Mount 단계



Component Lifecycle

Component Lifecycle

4 Update 단계

- 화면 출력이 변경 될 때 호출됨
- beforeUpdate(), updated() 함수가 호출됨

Component Lifecycle

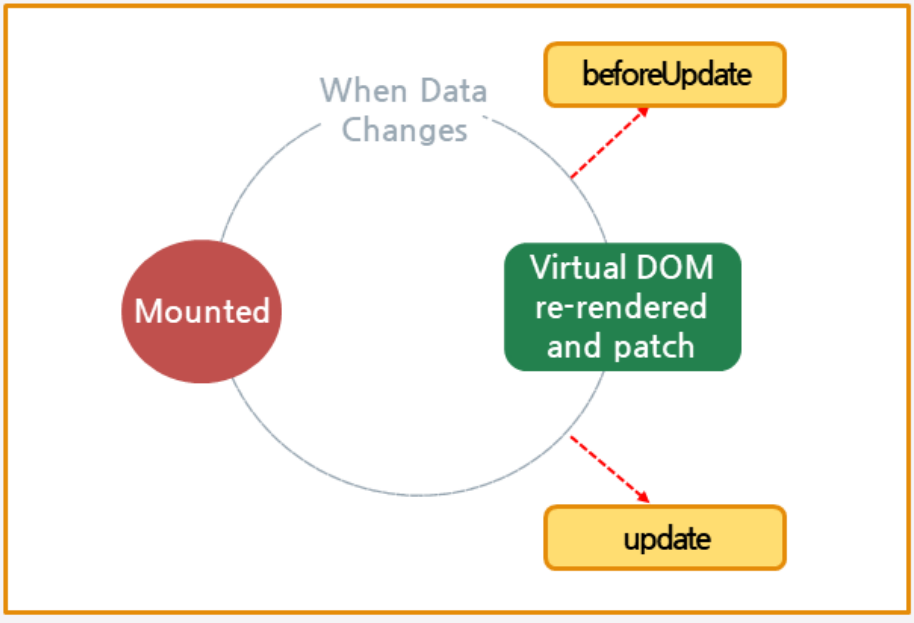
4 Update 단계

beforeUpdate()	데이터가 변경되기 전에 호출됨
updated()	데이터 Update가 완료된 후 호출됨

Component Lifecycle

Component Lifecycle

4 Update 단계



Component Lifecycle

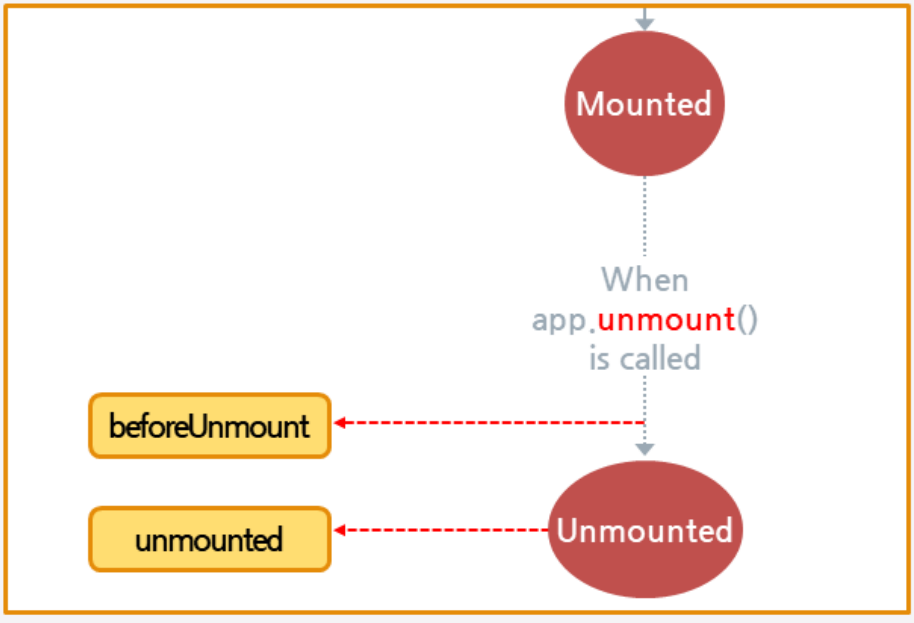
5 Destroy 단계

- Component의 소멸 단계임
- beforeUnmount()와 unmounted()함수가 호출됨

Component Lifecycle

Component Lifecycle

5 Destroy 단계

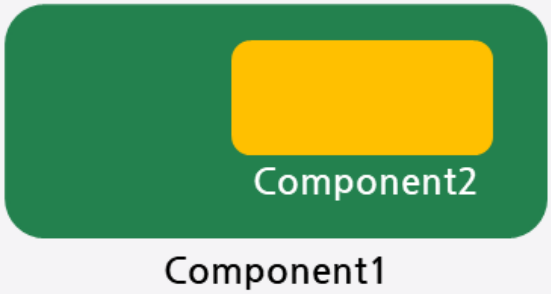


Props Data

Props Data

1 Props란?

- 상위 Component가 전달한 데이터임
- 상위 Component에서 하위 Component를 이용할 때 속성으로 추가한 데이터임



Props Data

2 props option

- 상위에서 속성으로 추가한 데이터를 받기 위한 option
- 상위에서 사용한 속성명을 props에 등록
- props내에 상위의 속성 값이 자동 대입되어 이용됨

Props Data

Props Data

2 props option

- 상위 Component

```
<Component2 attr1="hello" attr2="10"
attr3="true"></Component2>
```

- 하위 Component

```
export default {
    props: ['attr1', 'attr2', 'attr3'],
};
```

Props Data

2 props option

- props option에 대입된 값은 Component에서 this로 이용함

```
<template>
    <div>
        I am sub component!
        <br/>
        super data : {{message}}
        <br/>
        attr1 : {{attr1}}, attr2 : {{attr2}}, attr3 :
{{attr3}}
    </div>
</template>
```

Props Data

Props Data

2 props option

- props option에 대입된 값은 Component에서 this로 이용함

```
<script>
export default {
    props: ['attr1', 'attr2', 'attr3'],
    data(){
        return {
            message: this.attr1 + ':' + this.attr2 +
            ':' + this.attr3
        }
    }
};
</script>
```

Props Data

3 Type 지정

- props로 전달되는 데이터의 Type 지정
- Object Literal로 props 선언하면서 Value에 Type 지정

Props Data

Props Data

3 Type 지정

- String, Number, Boolean, Array, Object, Function 등 제공

```
props: {
        attr1: {
            type: String
        },
        attr2: {
            type: Number
        },
        attr3: {
            type: Boolean
        }
},
```

Props Data

4 상위 함수 호출

- 하위 Component에서 상위 Component의 함수 호출
- 상위 Component에서 속성으로 함수를 전달
- 하위 Component에서 $props를 이용해 상위 함수 호출

Props Data

Props Data

4 상위 함수 호출

- 상위 Component

```
<Sub v-
bind:attr4="superMethod"></Sub>
.....................
methods:{
    superMethod: function () {
      console.log("method function
call...");
    },
}
```

Props Data

4 상위 함수 호출

- 하위 Component

```
props: {
        attr4: {
            type: Function
        }
},
data(){
   this.$props.attr4()
   //.....................
}
```

Props Data

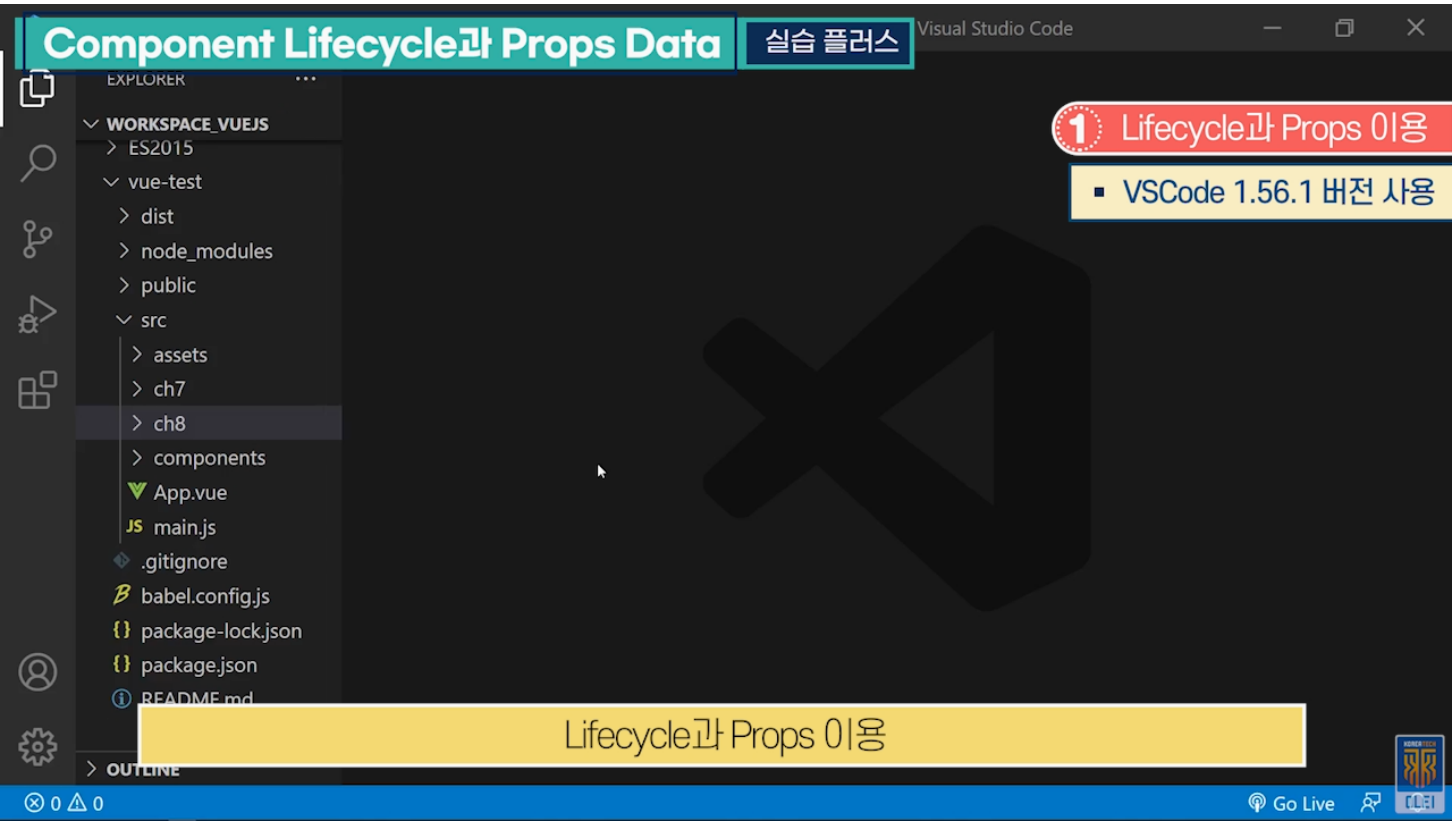
Props Data

5 required

- 필수 전달 데이터도 지정
- Props에 required: true 이용

```
props: {
        attr3: {
            type: Boolean,
            required: true
        }
},
```

실습 플러스



실습단계
src → ch9 서브 폴더 생성
2개의 Component 정의, 상하 관계로 엮어서 테스트 진행
Sub.vue 생성
다른 Component에 포함돼서 사용될 Component 정의
서브로 등록되어 사용될 Component
전달받을 속성 값 설정
상위 Component의 함수도 전달받을 수 있음
상위 속성 값을 받기 위한 Props 선언 후, 데이터 선언
전달받은 함수와 속성 확인
화면 template 부분 선언, 식별을 위해 'I am sub Component' 작성
super data 출력
상위에서 전달한 데이터를 하위에서 속성 명으로 바로 이용해도 됨
서브 Component 이용하는 Super Component 생성
ch9 → Super.vue 파일 생성

실습 플러스

실습단계
서브 Component를 이용하기 위해 해당 파일 import
자신의 Component 정의
Component 내의 Lifecycle 함수를 추가하여 template 정의한 곳에 있는 HTML 조정
Button Tag를 새로 생성
Button에 들어가는 문자열
사용될 서브 Component 등록
서브에 전달할 함수 선언
template 명시
서브 Component의 속성 값 부여
attr4는 함수 타입, 앞서 정의한 함수 명 superMethod 작성
이전에 생성한 ch8의 html, main.js, vue.config.js 파일 복사하여 이용
Vue의 CLI 명령어 실행
브라우저에 Component 출력
서브에서 Super의 함수 호출은 console로 확인