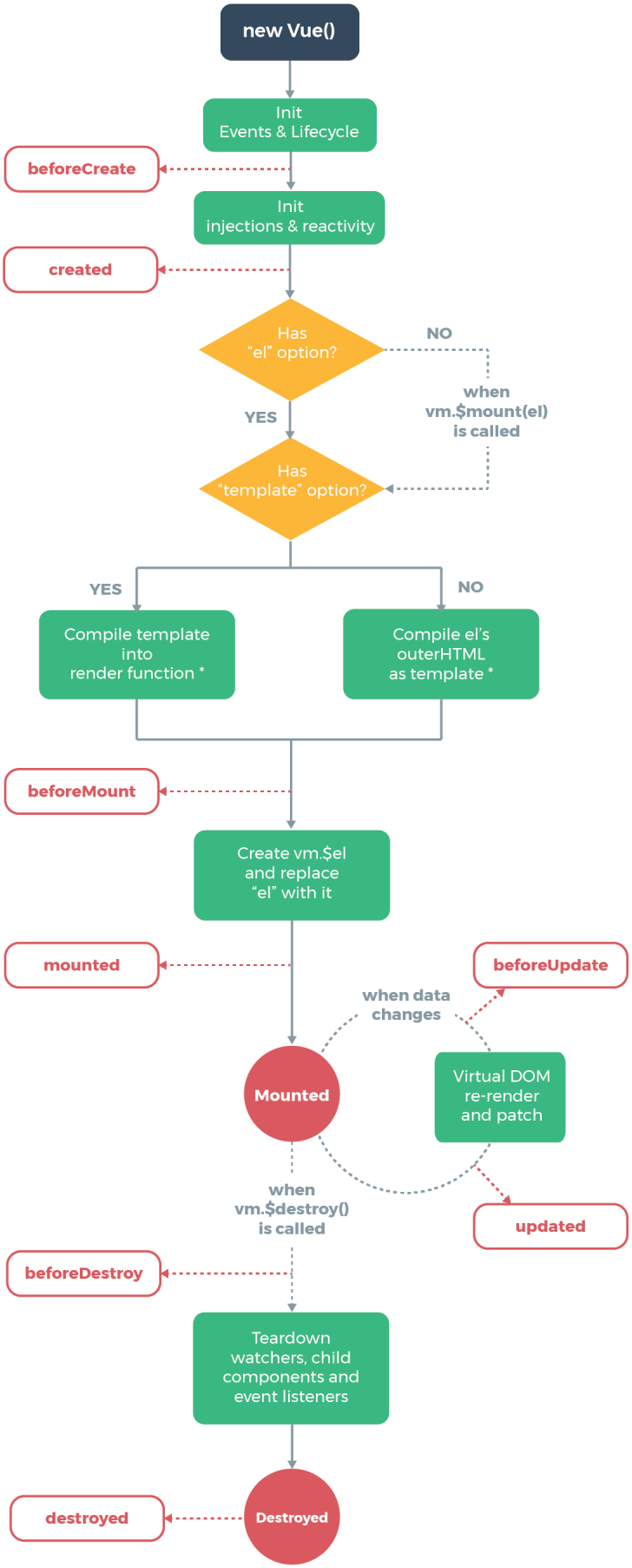
[FE] Vue.js 라이프 사이클

1. Vue.js 라이프 사이클

Vue.js 라이프사이클은 Vue 인스턴스의 생성부터 소멸까지의 과정입니다.

Vue 라이프 사이클이 실행되는 경우는 다음과 같습니다.

- Vue 인스턴스가 생성돼서 DOM에 마운트되는 경우
- 데이터가 변경되서 DOM을 업데이트하는 경우


new Vue()
Init
Events & Lifecycle
beforeCreate
Init
injections & reactivity
created
Has
"el" option?
NO
when
vm.$mount(el)
is called
YES
Has
"template" option?
YES
NO
Compile template
into
render function *
Compile el's
outerHTML
as template *
beforeMount
Create vm.$el
and replace
"el" with it
mounted
beforeUpdate
when data
changes
Mounted
Virtual DOM
re-render
and patch
when
vm.$destroy()
is called
updated
beforeDestroy
Teardown
watchers, child
components and
event listeners
destroyed
Destroyed

* template compilation is performed ahead-of-time if using
a build step, e.g. single-file components

Vue.js 라이프 사이클은 여러 단계가 정의되어 있습니다. 그리고 각 단계마다 사용자가 정의한 로직을 실행하는 주체가 있습니다. 이를 '훅'이라고 합니다.

Vue.js 공식문서에서는 '인스턴스 라이프사이클 훅'이라고 정의하고 있습니다.

2.인스턴스 라이프사이클 훅

Vue.js 라이프 사이클은 여러 단계를 순서대로 실행합니다. 각 단계마다 호출되는 훅이 있습니다.

주요 훅은 created, mounted, updated, destroyed가 있고, 이를 포함해서 총 8개의 인스턴스 라이프사이클 훅이 있습니다.

2-1. created

- 컴포넌트의 Data와 Method를 사용할 수 있습니다.

- DOM에 접근하는 로직을 사용할 수 없습니다.

저희 edgemaster에서 가장 많이 사용되는 훅입니다. 컴포넌트의 Vue 인스턴스가 만들어지는 시점에 실행이 되는 훅입니다.

쉽게 말해서, 라이프 사이클 훅 중에서 가장 먼저 사용자 정의 로직을 실행해 주는 훅입니다.

created 훅 예시 코드

```
export default {
/*
…
*/
created() {
  console.log('created');
}
}
```

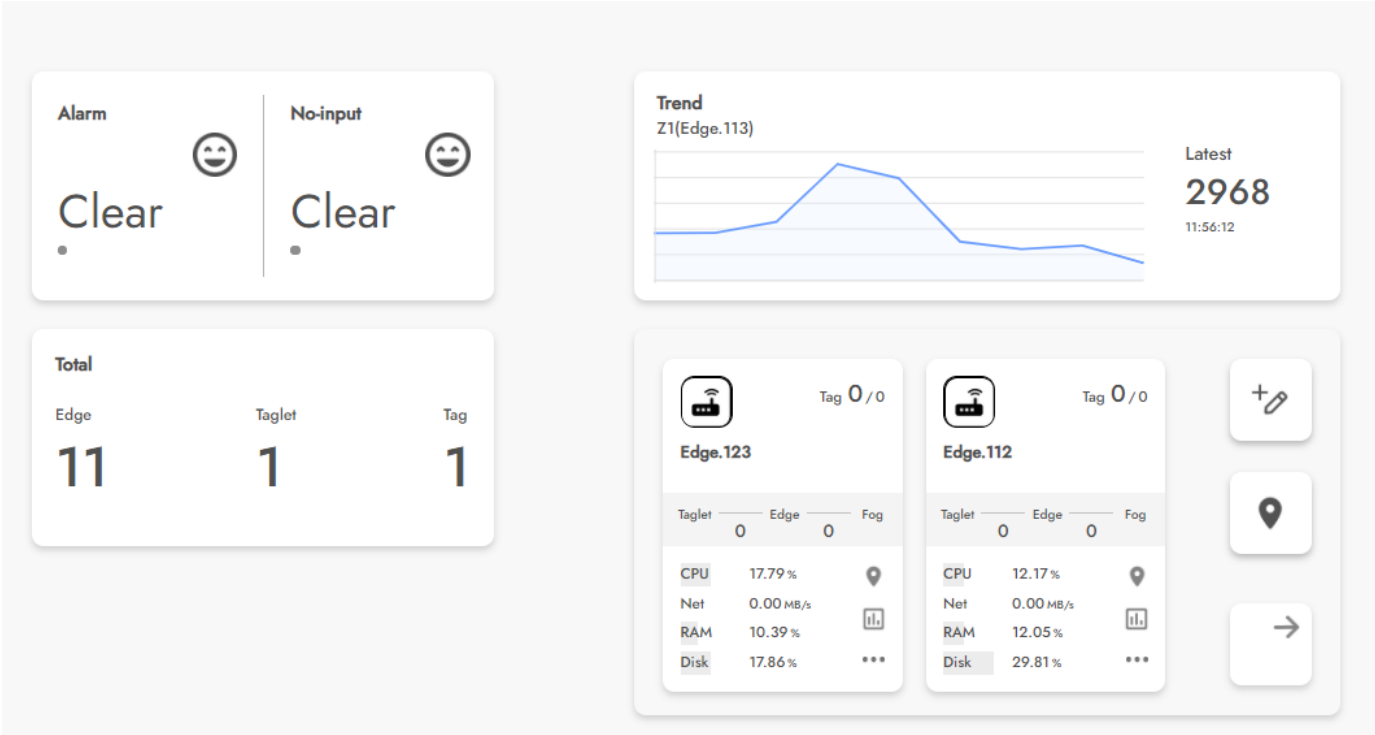
created 훅에서는 HTTP GET method를 사용해서 컴포넌트가 생성되기 전에 미리 서버로부터 데이터를 받아오는 기능을 주로 작성합니다.

edgemaster에서 created에 가장 많은 로직을 사용하는 컴포넌트는 Main.vue입니다. edgemaster에 Dashboard, map, group 페이지를 보여주는 컴포넌트입니다.

dash board를 보면, 처음 화면이 랜더링될 때, 데이터들이 들어가 있는 상태로 화면에 보여지게 됩니다.

Main컴포넌트가 생성되면서 서버로부터 데이터를 받아와서 virtual DOM에 마운트하기 때문입니다.

> 엣지마스터 Dashboard 화면



> Main 컴포넌트 created훅 코드

```
async created() {
    this.setCurrentView('dashboard');
    this.setEdgeInfoList(await getEdgeInfoList());
    this.setFogInfo(await getFogInfo());
    this.setGroupList(await getGroupList());
    this.setRiskList(await getRiskList());
    this.setMapList(await getMapList());
    this.handleInitialize(this.gCurrentViewMode);
    this.handleStartMainInterval();
    this.setTrendDevice();
},
```

또 한 가지, edgemaster에서 created 훅 사용 예를 들면,

edgemaster가 실행되면 가장 먼저 최상위 컴포넌트의 created훅이 실행되는 컴포넌트는 App.vue입니다.

```
async created() {
    if (sessionStorage.id === 'admin') {
        this.sIsAdmin = true;
    }

    sessionStorage.getItem('id') || this.$router.push('/login');

    this.setMemberList(await getMemberList());
    this.setVersion(await getVersion());
    this.setBroadCast(await getPluginList());
}
```

HTTP GET요청 함수들을 통해서 '멤버 목록'과 '버전 정보' 그리고 'plugin 정보'를 우선적으로 받아오도록 정의되어 있습니다.

2-2. mounted

- 컴포넌트의 Data와 Method를 사용할 수 있습니다.
- DOM에 접근하는 로직을 사용할 수 있습니다.
- 모든 컴포넌트가 마운트된 상태를 보장하지는 않습니다.

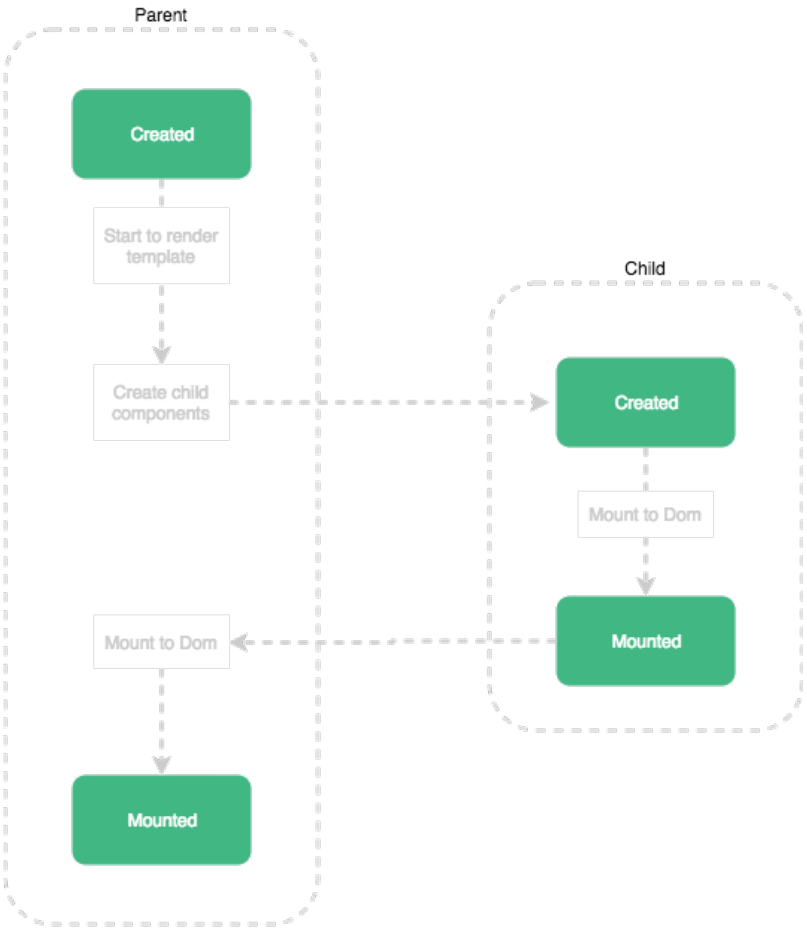
mounted훅은 새로운 컴포넌트가 DOM에 추가가 되고 난 후에 실행되는 훅입니다.

mounted 훅 예시 코드

```
export default {
/*
…
*/
mounted() {
  // $el    .
  console.log('mounted', this.$el);
  this.$nextTick(() => {
  //      .
  });
}
}
```

mounted 특징 중 하나가 아래 그림에 나와 있습니다. 아래 그림은 부모 컴포넌트와 자식 컴포넌트 간에 created 훅과 mounted 훅의 실행 시점을 표시한 그림입니다.

컴포넌트 간에 계층 관계가 있을 경우 created와 반대로, mounted 훅은 자식 컴포넌트가 부모 컴포넌트보다 먼저 실행됩니다.



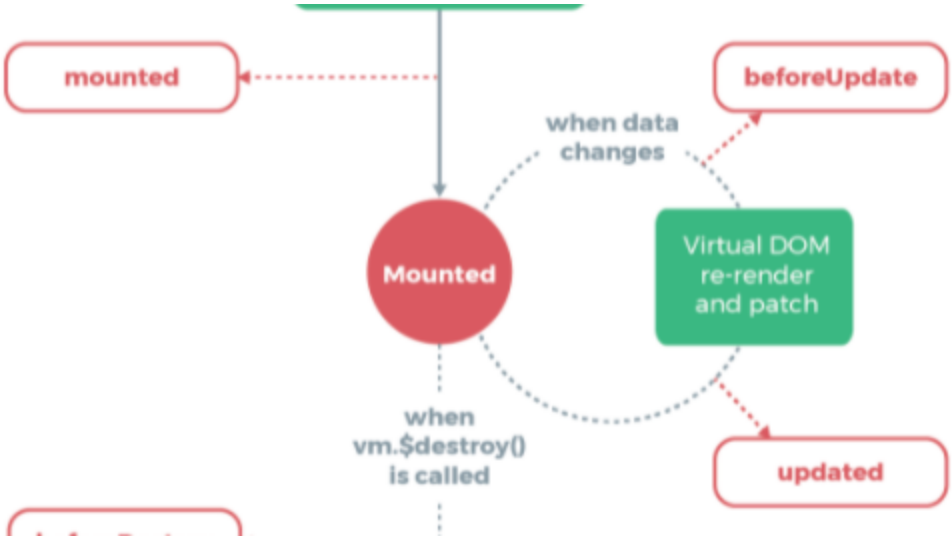
주의할 점은, 만약 mounted 훅에서 DOM을 수정하는 로직이 있는 경우에는 모든 컴포넌트가 DOM에 마운트된 후에 로직을 실행해 주는 함수인 nextTick()함수에 사용해야 합니다.

이 내용은 Vue.js의 공식문서에 잘 나와 있습니다.

Note that `mounted` does **not** guarantee that all child components have also been mounted. If you want to wait until the entire view has been rendered, you can use vm.$nextTick inside of `mounted`:

2-3. updated

mounted와 비슷한 시점에 실행되는 훅입니다.



컴포넌트의 신규 생성으로 DOM에 마운트 되고 난 후에 실행되는 mounted 훅과는 다르게, updated훅은 기존에 DOM에 있는 컴포넌트가 데이터의 변경으로 컴포넌트가 DOM에 다시 랜더링 된 후에 실행되는 훅입니다.

mounted와 마찬가지로 변경된 데이터가 다시 랜더링이 완료된 DOM을 조작하는 로직을 실행할 경우, nextTick()을 사용할 수 있습니다.

updated 훅 예시 코드

```
export default {
/*
…
*/
mounted() {
  // $el    .
  console.log('updated');
  this.$nextTick(() => {
  //      .
  });
}
}
```

저희 edgemaster에서는 사용하지 않는 훅입니다.

2-4. destroyed

컴포넌트가 DOM에서 소멸되고 난 후에 동작하는 훅입니다. 소멸되는 컴포넌트에 설정된 Event(@change, @click ...)나 디렉티브 바인딩(v-model, v-if, v-show...)도 함께 소멸됩니다.

당연히, 자식 컴포넌트를 가지는 부모 컴포넌트가 destroyed 될 경우에는 하위의 자식 컴포넌트도 소멸됩니다.

destroyed훅 예시 코드

```
export default {
/*
…
*/
destroyed() {
  console.log('destroyed');
}
}
```

edgemaster에서는 destroyed 훅을 두 가지 용도로 사용했습니다.

첫 번째로는 컴포넌트가 소멸될 때, Vue.js 프레임워크에서 중앙 저장소로 사용하는 Store에 저장된 페이지 상태 정보 데이터를 초기화하기 위해서 사용했습니다.

edgemaster에서 url 경로에 맞는 페이지 상태 데이터로 변경하기 위해서 소멸과 동시에 경로에 맞는 상태 정보 데이터로 변경해주기 위해서 destroyed 훅에 관련 로직 구현했습니다.

아래 코드가 전역 상태 정보를 변경해주는 코드입니다.

```
destroyed() {
    this.setDeviceEditMode(this.gDefaultStr);
},
```

두 번째로는 edgemaster 웹에서 ems나 emc로부터 주기적으로 edge정보를 받아오는 API를 호출하기 위해서 Interval을 사용합니다. 컴포넌트가 제거될 때, clearInterval하기 위한 용도로 사용하고 있습니다.

주요 훅의 특징을 알아보았습니다.

감사합니다.

참고 :

Vue.js 공식 사이트

지그재그 기술 블로그

https://beomy.tistory.com/47