7/13 ~ 7/17 2주차 - 4

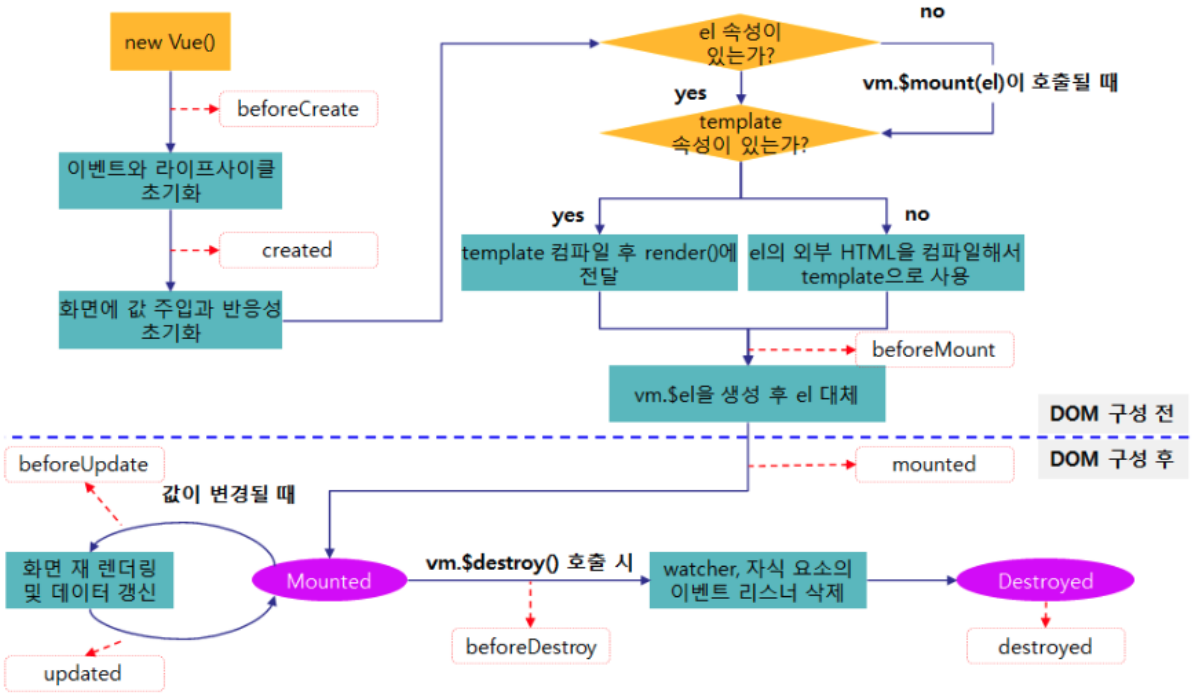
🤯 객체에 대해서 2

- Vue 객체의 라이프사이클과 관련 콜백 메서드에 대해 알아보기.

Vue 객체의 라이프사이클

- Vue 객체는 생성에서부터 화면 연결, 값 변경에 대한 처리 등 다양한 일들에 관여하다가 소멸된다.

아래 그림의 붉은 파산 사각형??? 각각의 상황에서 콜백되는 라이프 사이클 훅 메서드(hook method)이다. → 훅 채가는 느낌?? 갈고리에 걸려서



그림은 복잡하지만 자주 보고 익히고 중요한거 먼저 찾기

DOM 구성 전후로 나뉘는데

나중에 DOM에 대해서 정리하겠지만 간단하게

DOM 요약정리

- DOM은 HTML문서에 대한 인터페이스이다
- 뷰 포트에 무엇을! 렌더링 할지 결정하기 위해 사용되며,
- 페이지의 콘텐츠 및 구조, 그리고 스타일이 자바스크립트 프로그램에 의해 수정되기 위해 사용된다.
- DOM은 원본 HTML 문서 형태와 비슷하지만 몇 가지 차이점이 있다.
  - 항상 유효한 HTML 형식이다.
  - 자바스크립트에 수정될 수 있는 동적 모델이라는 점
  - 가상 요소를 포함하지 않는다 예) ::after
  - 보이지 않는 요소를 포함한다. 예) display: none

라이프사이클 훅

- 라이프 사이클과 연동돼서 호출되는 메서드들이다.

1. beforeCreate
   - Vue 객체가 생성되고 데이터에 대한 관찰 기능 및 이벤트 감시자 설정 전에 호출된다. → 거의 다룰 일이 없다고 한다!
2. created
   - Vue 객체가 생성된 후 데이터에 대한 관찰이 가능하다.
   - computed, methods, watch 설정이 완료된 후이다.
3. beforeMount
   - 마운트가 시작되기전 호출

- 따라서 DOM 구성 이기 때문에 화면을 건드리기에는 부적절!

4. mounted
    - DOM 요소가 el에 의한 가상 DOM으로 대체된 후 호출됨.
    - 화면에 대한 완벽한 제어가 가능해지는 시점이다.
5. beforeUpdate
    - 데이터가 변경되서 가상 DOM이 다시 렌더링 되기 전에 호출됨.
6. update
    - 데이터 변경으로 가상 DOM이 다시 렌더링 되고 패치된 후에 호출됨.
    - 이 훅이 호출된 시점은 이미 가상 DOM이 렌더링 된 후이므로 DOM에 대한 추가 작업이 가능하다.
7. beforeDestroy
    - Vue 객체가 제거되기 전에 호출된다.
8. destroyed
    - Vue 객체가 제거된 후에 호출된다.
    - 이 훅이 호출 될 때 Vue 객체의 모든 디렉디브 바인딩이 해제되고 이벤트 연결도 제거 됨!

결론적으로 신경써야할 훅은 멀까요?

화면과 연결이 가능한 mounted, updated가 가장 활용도가 높다는 것 기억하기.

라이프사이클 훅 호출의 예를 보면.

훅 들은 Vue 객체를 만들면서 해당 객체의 속성으로 지정이 가능하다.

```
<div id="first">
  <h3 v-html="num"></h3>
  <button @click="plus">+</button>
</div>
```

- id가 first인 가 el의 대상이다.

```
let vi = new Vue({
    el: "#first",
    data: model,
    methods: {
        plus() {
           this.num++;
        },
    },
    beforeCreate: function () {
      // 아직 el이나 data가 연동되지 않았다.
      console.log("beforeCreate ", this.title, this.$el);
    },
    created: function () {
      // data , computed, methods, watch 설정이 완료된다.
      console.log("created ", this.title, this.$el);
    },
    beforeMount: function () {
      // data와 el이 준비 되었지만 아직 화면이 대체되지는 않았다.
      console.log("beforeMount ", this.title, this.$el.innerHTML);
      console.log("beforeMount ", document.querySelector("#first").innerHTML);
    },
    mounted: function () {
      // data와 el이 준비 되었으며 화면이 가상 DOM으로 대체되었다.
      console.log("mounted ", this.title, this.$el.innerHTML);
      console.log("mounted ", document.querySelector("#first").innerHTML);
    },
    beforeUpdate: function () {
      // +를 누르면 값이 업데이트 되려고 한다. 값은 변경되었지만 아직 화면은 갱신 전이다.
      console.log("beforeUpdate ", this.num, document.querySelector("#first").innerHTML);
    },
    updated: function () {
      // 값이 업데이트 되었고 화면도 갱신 되었다.
      console.log("updated ", this.num, document.querySelector("#first").innerHTML);
    },
    beforeDestory: function () {
      // 객체가 삭제되기 직전이다.
      console.log("beforeDestory ", this.title, this.$el);
    },
    destory: function () {
      // 객체가 삭제된 이후이다.
      console.log("destory ", this.title, this.$el);
    },
});
```

각각 모델, DOM에 접근하는 것 보기.