

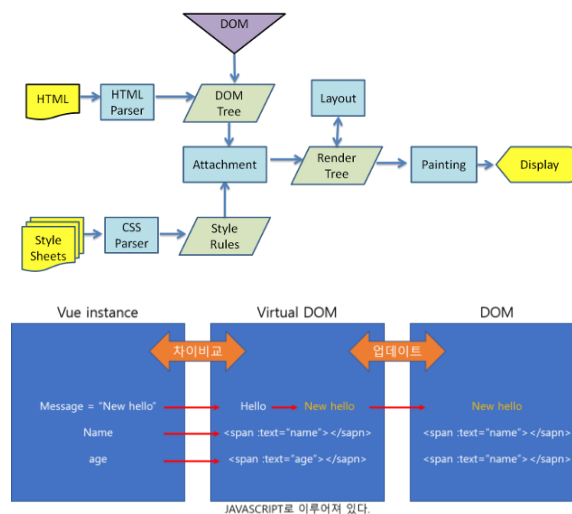


Vue 인스턴스

발표자 : 배소희

Vue 사전지식

- 라이프 사이클(Life Cycle) : 생명주기. 생성 후 소멸되기까지의 과정
- 훅(Hook) : 소프트웨어 구성 요소 간에 발생하는 함수 호출, 이벤트 등을 중간에서 바꾸거나 가로채는 기술을 처리하는 코드
- DOM(Document Object Model) : XML이나 HTML 문서에 접근하기 위한 인터페이스
- 가상DOM(Virtual DOM) : 불필요한 렌더링 횟수를 줄이기 위해 만든 가상의 DOM
- 렌더링(Rendering) : 서버로부터 HTML 파일을 받아 브라우저에 뿌려주는 과정
- 컴파일(Compile) : 사람이 이해하는 언어를 컴퓨터가 이해할 수 있는 언어로 변환
- 디렉티브(Directive) : 특정한 행위의 기능을 가진 DOM 엘리먼트
- 바인딩(Binding) : 구체적인 값을 할당하는 각각의 과정
- 컴포넌트(Component) : 재사용이 가능한 각각의 독립된 모듈
- 템플릿(Template) : html, css 등의 마크업 속성과 데이터 및 로직들을 연결하여 브라우저에서 볼 수 있는 html로 변환해주는 속성
- 서버사이드(Server-side) : 클라이언트-서버 구조의 서버쪽에서 행해지는 처리
- 서버사이드렌더링(Server-side Rendering) : 서버에서 페이지를 그려 클라이언트로 보낸 후 화면에 표시



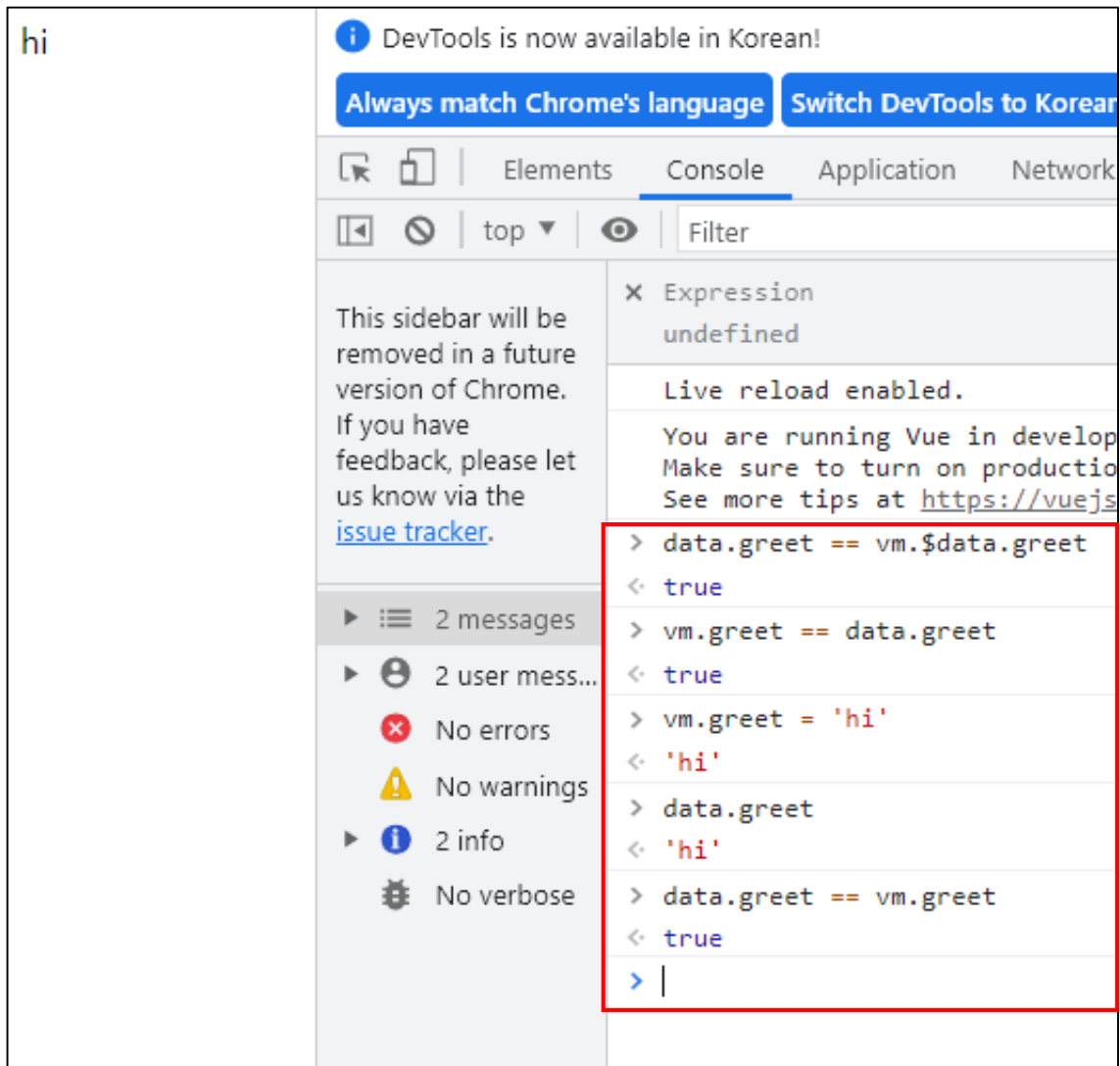
Vue 인스턴스

1. 인스턴스의 생성방법

```
var vm = new Vue({  
  // 옵션  
})
```

2. 데이터와 메소드

```
// 데이터 객체  
var data = { a: 1 }  
  
// Vue인스턴스에 데이터 객체를 추가합니다.  
var vm = new Vue({  
  data: data  
})  
  
// 인스턴스에 있는 속성은  
// 원본 데이터에 있는 값을 반환합니다.  
vm.a === data.a // => true  
  
// 인스턴스에 있는 속성값을 변경하면  
// 원본 데이터에도 영향을 미칩니다.  
vm.a = 2  
data.a // => 2  
  
// 반대로 마찬가지입니다.  
data.a = 3  
vm.a // => 3
```



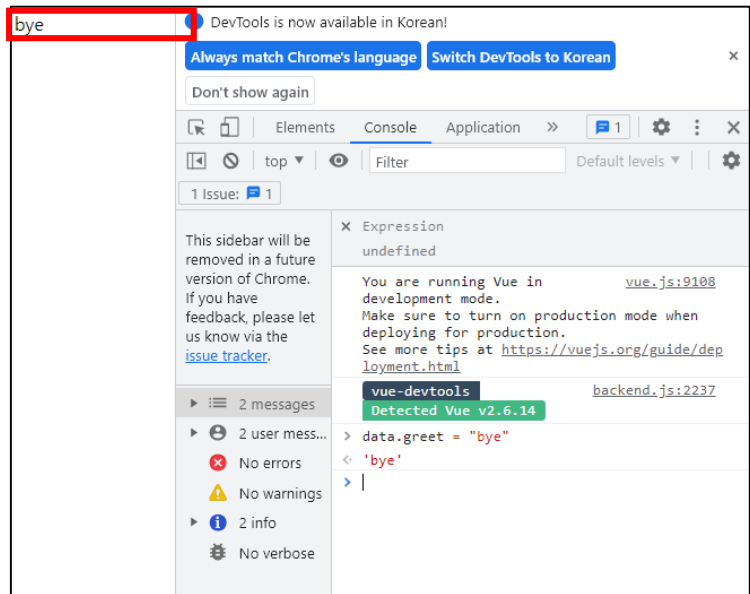
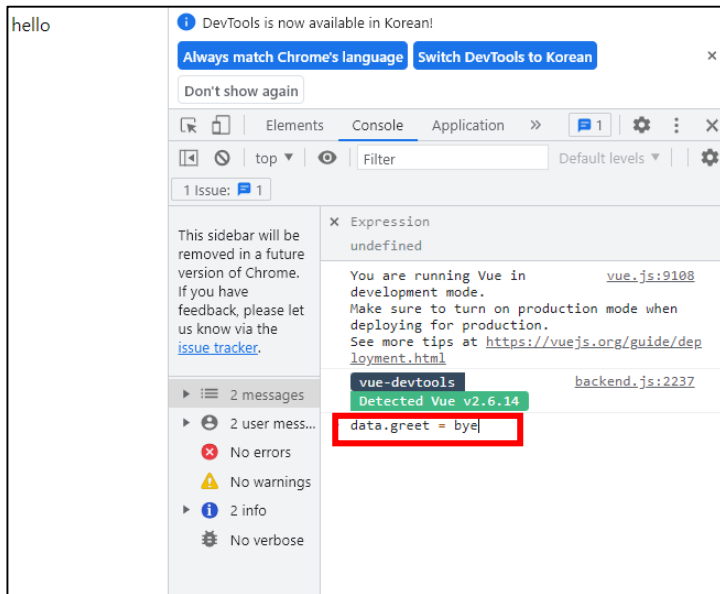
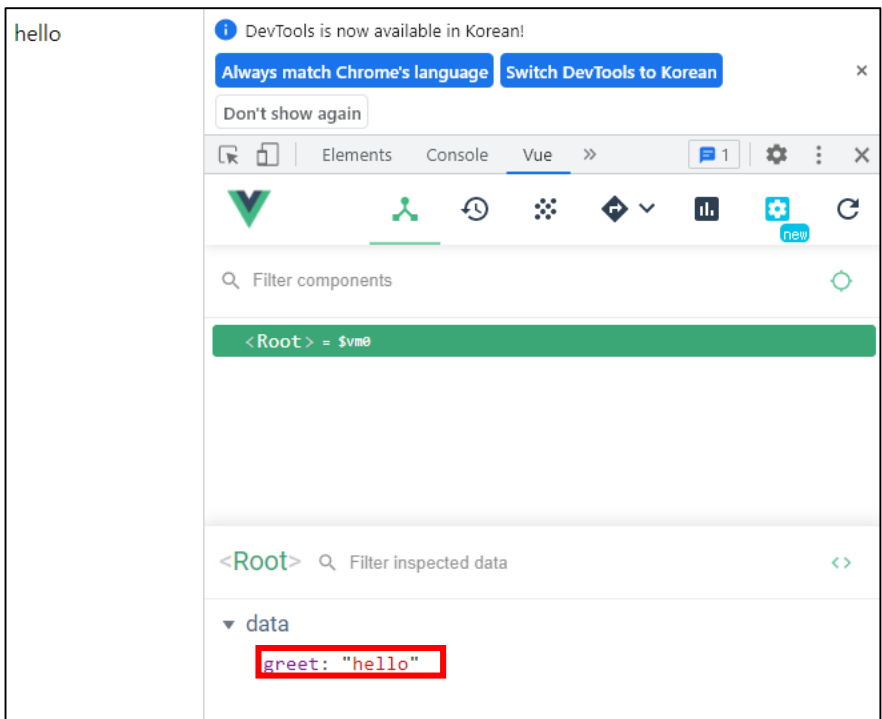
The screenshot shows a web browser with the text "hi" displayed. The DevTools console is open, showing several messages. A red box highlights a series of commands and their outputs in the console:

```
> data.greet == vm.$data.greet  
< true  
> vm.greet == data.greet  
< true  
> vm.greet = 'hi'  
< 'hi'  
> data.greet  
< 'hi'  
> data.greet == vm.greet  
< true  
> |
```

The console also shows a message about DevTools being available in Korean, buttons to switch the language, and a sidebar with 2 messages, 2 user messages, 0 errors, 0 warnings, 2 info messages, and no verbose messages.

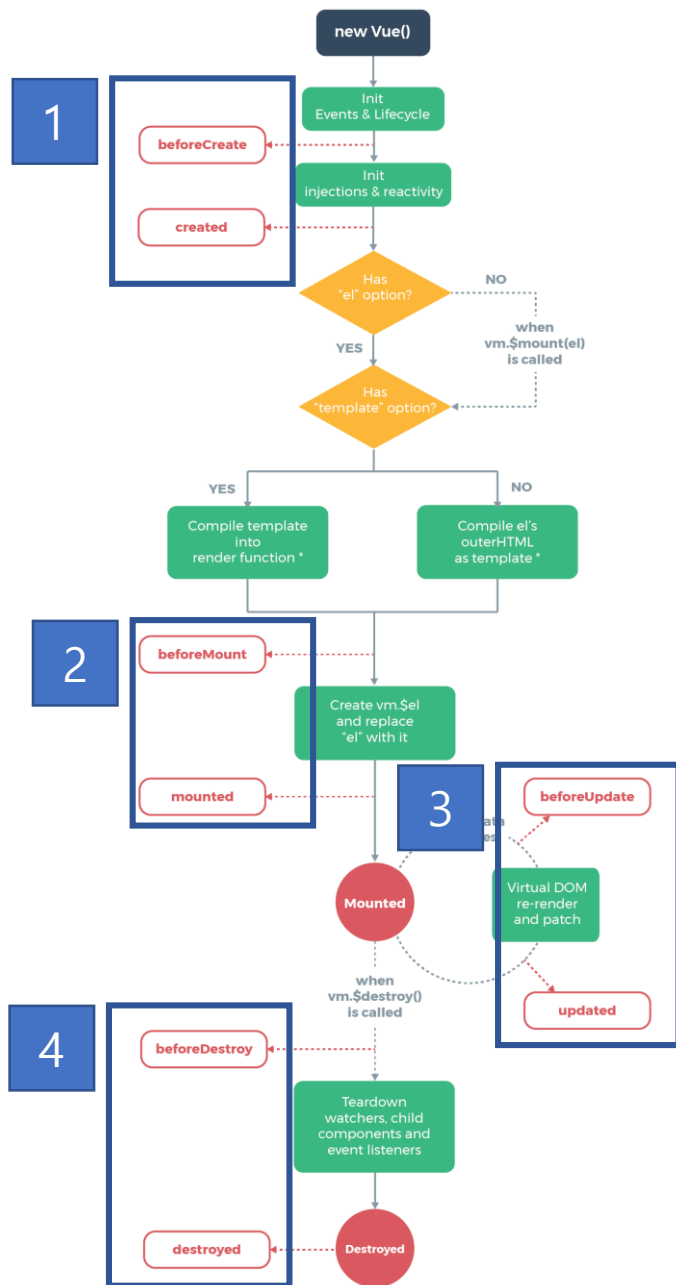
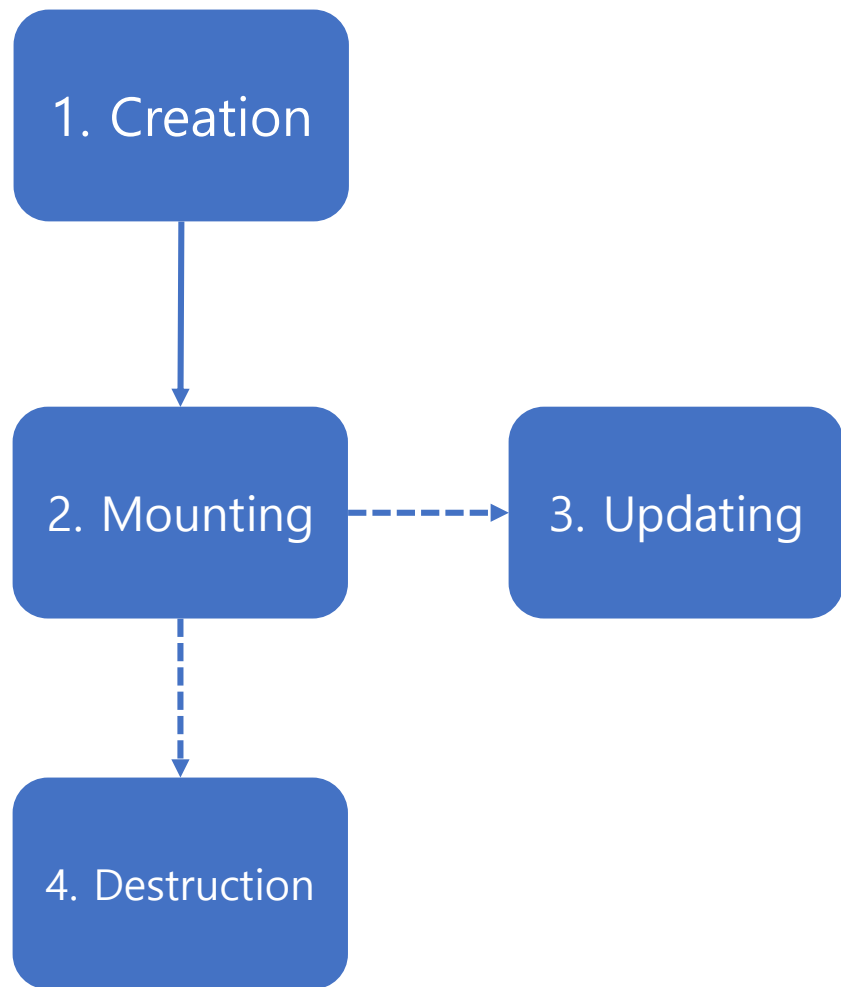
Vue 인스턴스

```
<body>
  <div id="app">
    {{greet}}
  </div>
</body>
<!-- 개발버전, 도움되는 콘솔 경고를 포함. -->
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
  var data = { greet : "hello" };
  var vm = new Vue({
    el : '#app',
    data : data
  });
```



console 창에서 data 의 값을 변경 해주면 화면이 다시 랜더링 됩니다.
여기서 유의해야할 점은 data에 있는 속성들은 인스턴스가 생성될 때 존재
한 것들만 반응형이라는 것입니다.
어떤 속성이 나중에 필요하다는 것을 알고 있으면 초기값을 지정해야합니다.

Vue 라이프사이클



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

Vue 라이프사이클

1. Creation

1) beforeCreate

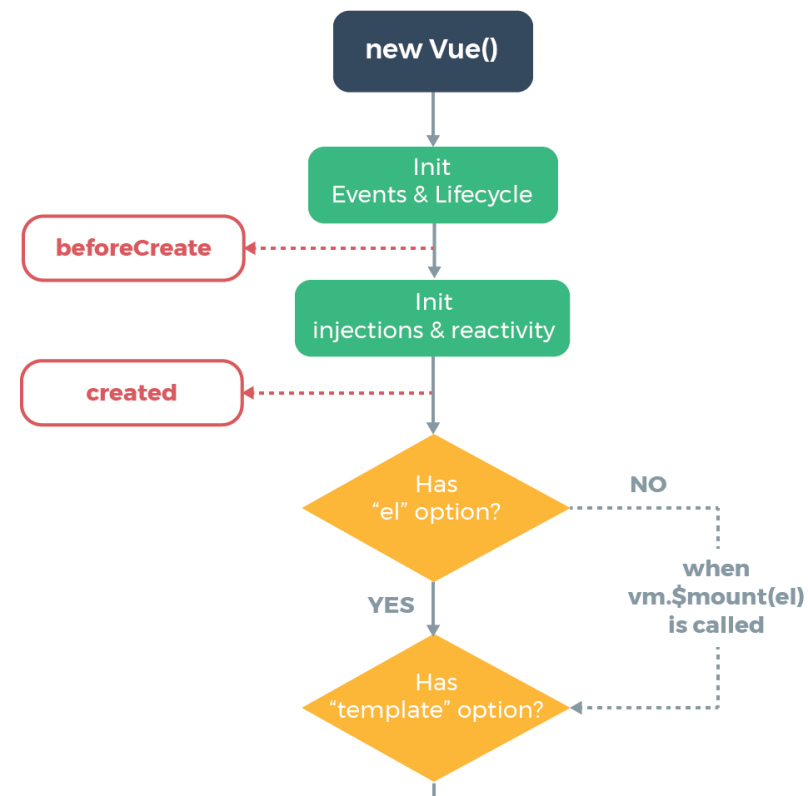
2) created

beforeCreate

- 모든 훅 중 가장 먼저 실행
- data와 events 가 세팅되지 않은 시점

created

- Data와 events가 활성화 되어 접근 가능
- 템플릿과 가상DOM 은 랜더링되지 않은 상태



* 인스턴스의 생성방법

```
var vm = new Vue({  
  // 옵션  
})
```

Vue 라이프사이클

2. Mounting

1) beforeMount

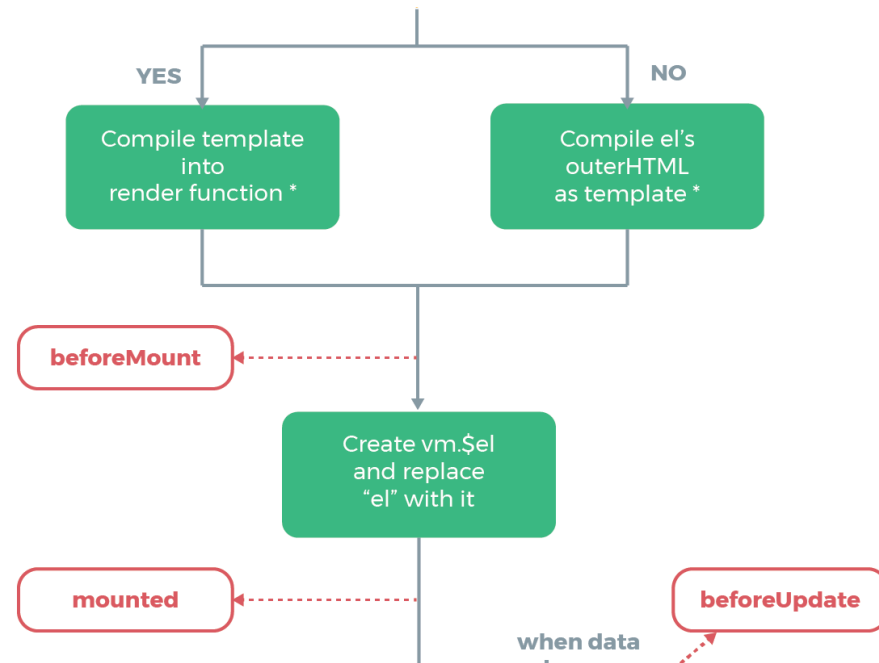
2) mounted

beforeMount

- 템플릿과 랜더 함수들이 컴파일 된 후에 첫 랜더링이 일어나기 직전에 실행
- 서버사이드 렌더링시에는 호출되지 않음

mounted

- 컴포넌트, 템플릿, 랜더링된 돔에 접근가능
- 모든 하위 컴포넌트가 마운트된 상태를 보장하지 않음
- 서버랜더링에서는 호출되지 않음

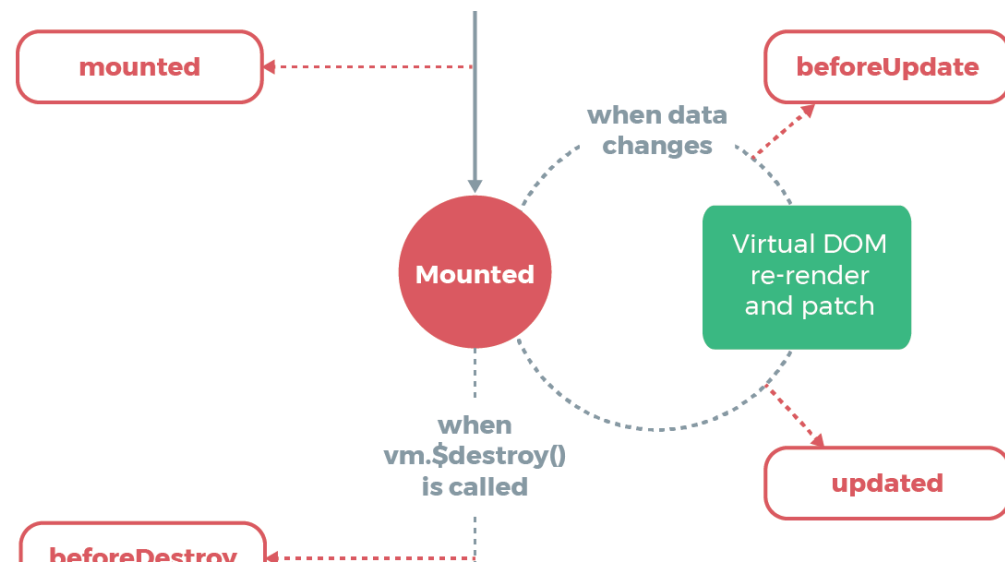


Vue 라이프사이클

3. Updating

1) beforeUpdate

2) updated



컴포넌트에서 사용되는 반응형 속성들이 변경되거나 재 렌더링이 발생되면 실행

beforeUpdate

- 돔이 재랜더링되고 패치되기 직전에 실행됨
- 이 변경으로 인한 재 렌더링은 트리거 되지 않음

updated

- 컴포넌트의 데이터가 변하여 재랜더링이 일어난 후에 실행
- 모든 자식 컴포넌트의 재랜더링 상태를 보장하지 않음

- updated 에서는 돔이 업데이트 완료된 상태이므로 돔 종속적인 연산이 가능. But 상태를 변경할 경우 무한루프에 빠질 수 있음.

Vue 라이프사이클

4. Destruction

1) beforeDestroy

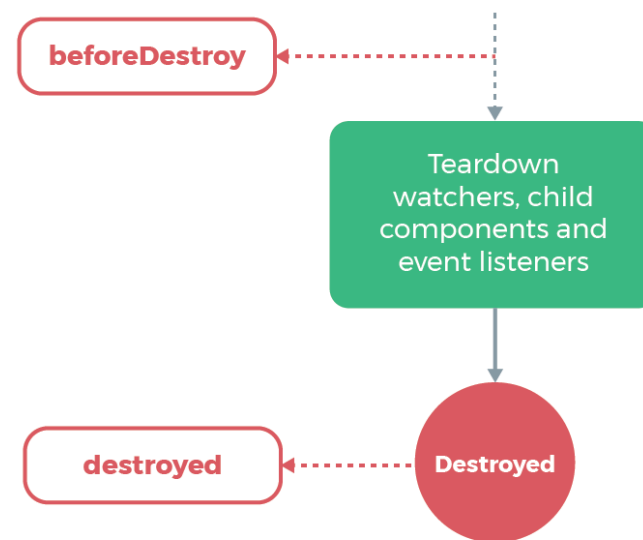
2) destroyed

beforeDestroy

- 해체(뷰 인스턴스 제거)되기 직전에 호출

destroyed

- 해체(뷰 인스턴스 제거) 된 후에 호출
- Vue 인스턴스의 모든 디렉티브가 바인딩 해제되고 모든 이벤트 리스너가 제거되며 하위 Vue 인스턴스도 삭제
- 서버 렌더링시 호출되지 않음



Q & A

감사합니다.