Lifecycle Hooks

Each Vue component instance goes through a series of initialization steps when it's created - for example, it needs to set up data observation, compile the template, mount the instance to the DOM, and update the DOM when data changes. Along the way, it also runs functions called lifecycle hooks, giving users the opportunity to add their own code at specific stages.

Registering Lifecycle Hooks

For example, the `onMounted` hook can be used to run code after the component has finished the initial rendering and created the DOM nodes:

```vue
<script setup>
import { onMounted } from 'vue'

onMounted(() => {
  console.log(`the component is now mounted.`)
})
</script>
```

There are also other hooks which will be called at different stages of the instance's lifecycle, with the most commonly used being `onMounted`, `onUpdated`, and `onUnmounted`.

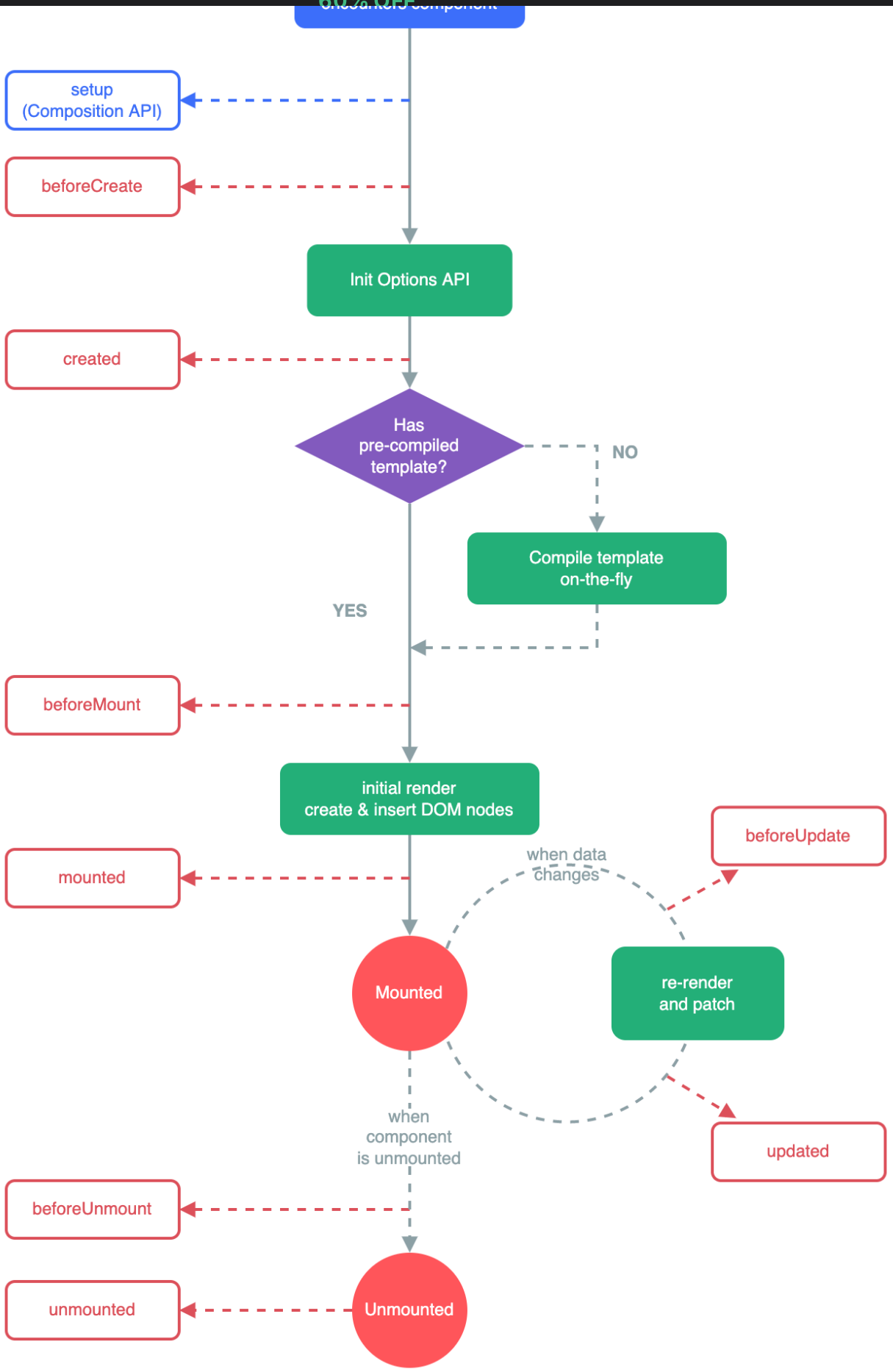
When calling `onMounted`, Vue automatically associates the registered callback function with the current active component instance. This requires these hooks to be registered **synchronously** during component setup. For example, do not do this:

```js
setTimeout(() => {
  onMounted(() => {
    // this won't work.
  })
}, 100)
```

stack is synchronous and originates from within `setup()`.

Lifecycle Diagram

Below is a diagram for the instance lifecycle. You don't need to fully understand everything going on right now, but as you learn and build more, it will be a useful reference.



Consult the Lifecycle Hooks API reference for details on all lifecycle hooks and their respective use cases.

Previous
Components Basics

Next
Registration