【前端面试】Vue 每个生命周期都做了什么?

前言

这是一道非常常问的面试题，一般如果面试官准备问 Vue 相关的问题，那么往往都会拿 Vue 的生命周期作为开头。很多小伙伴可能觉得这道题非常简单，所以往往就不去准备，结果导致每次回答得都不够全面，而面试官就会觉得你这个人态度不端正。

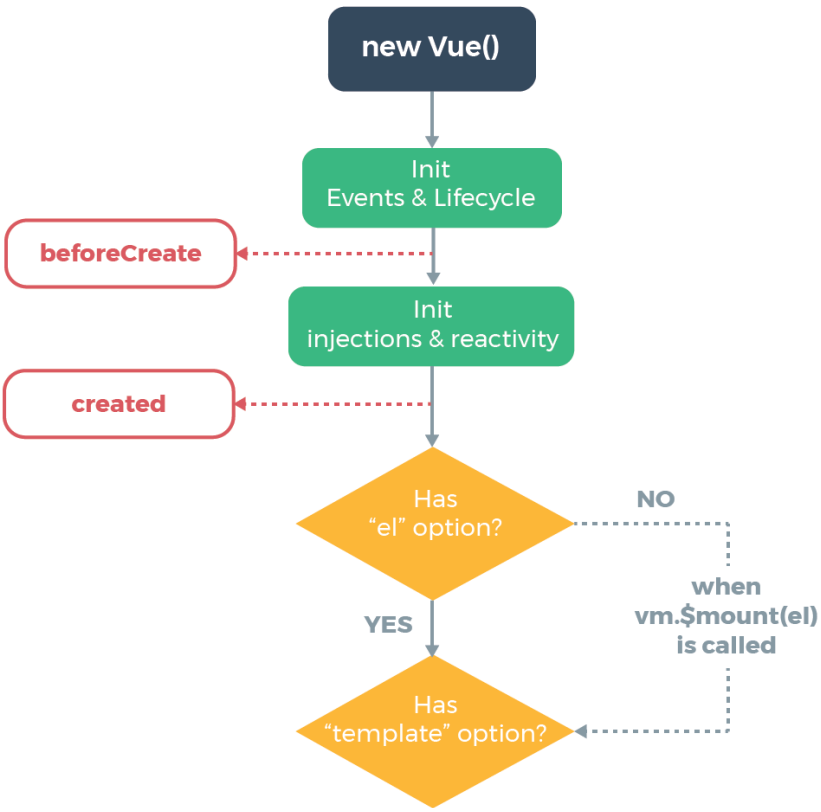
所以，我们还是好好学一学基础的东西吧！至少态度端正！

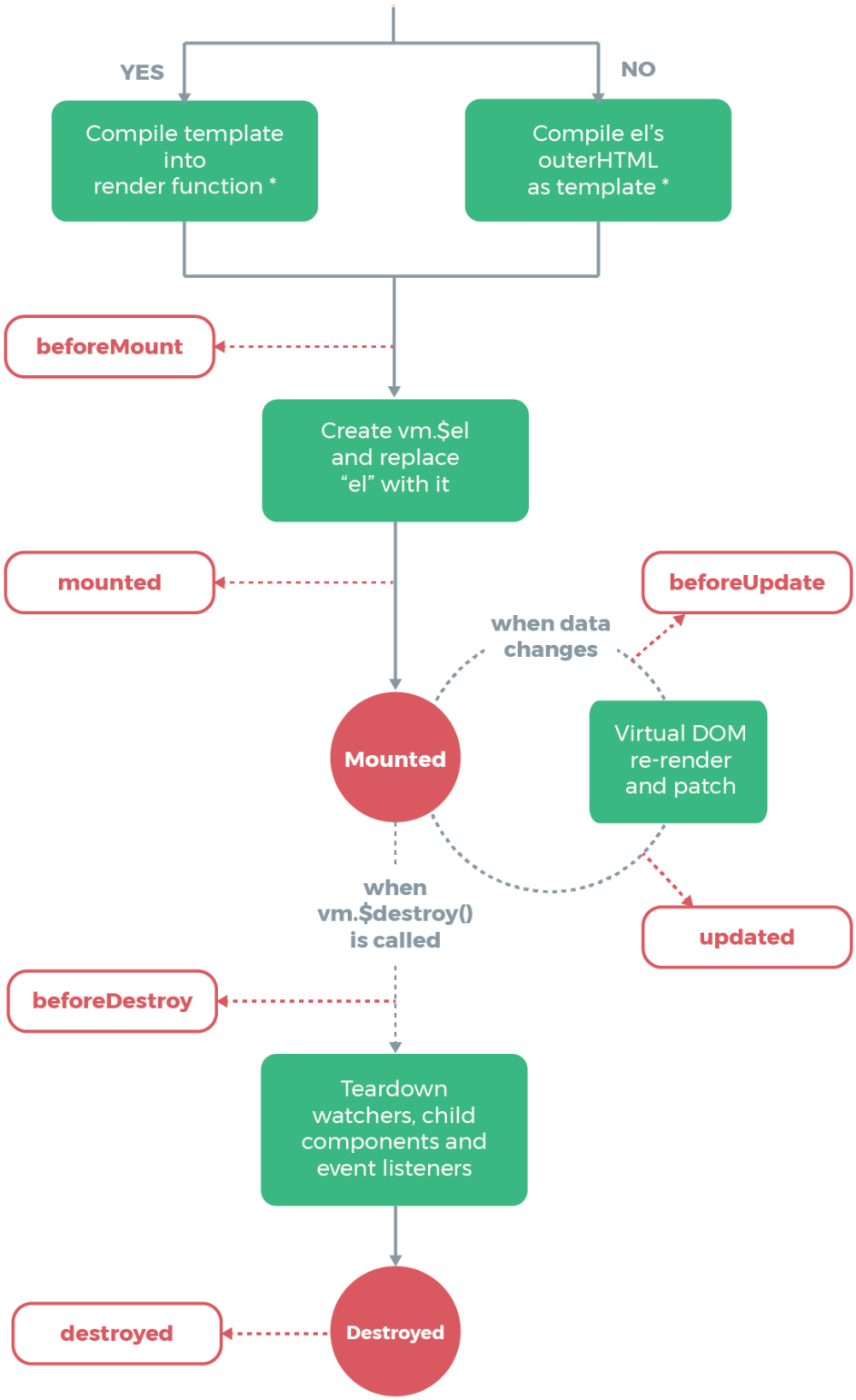
1.什么是 Vue 的生命周期?

如果使用 Vue 框架，那么学习它的生命周期就不可避免的。我们人生老病死，尘归尘，土归土，这算是一个周期。一个 Vue 应用也是如此，从最初的创建，到最后的消亡，这也是一个周期。

人的一生分为了：幼年、少年、青年、中年、老年等阶段，我们需要在不同的阶段去做不同的事。一个 Vue 应用也是如此，只不过它将生命周期的各个阶段用钩子函数替代了，钩子函数内部就是我们需要做的事情。

官网的一张图就非常清晰的介绍了一个 Vue 应用或者组件的生命周期：





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

这是一张 Vue2.x 的生命周期图，从创建到销毁，分为了这几个阶段：beforeCreate、created、beforeMount、mounted、beforeUpdate、updated、beforeDestory、destoryed。这些就是每个阶段对应的钩子函数，好比我们人生每个阶段，那么接下来我们就讲一讲每个钩子函数在做什么事。

2.beforeCreate

第一个周期函数，Vue 实例创建之前会执行它，此时 data 和 methods 中的数据或方法还未初始化，无法调用，只能使用一些默认事件。

示例代码：

```
<script>
export default {
  data() {
    return {
      msg: "小猪课堂",
    };
  },
  beforeCreate() {
    console.info("-----beforeCreate-----");
    console.info("data", this.msg);
    console.info("methods", this.getMsg);
  },
  methods: {
    getMsg() {
      return
    },
  },
};
</script>
```

输出结果：

```
[HMR] Waiting for update signal from WDS...
-----beforeCreate-----
data undefined
methods undefined
```

上段代码中我们定义了一个 data 数据和一个方法，可以看到，我们想在 beforeCreate 钩子函数里面想要获取 data 和 methods 是获取不到的，因为这个时候它们还未初始化。

3.created

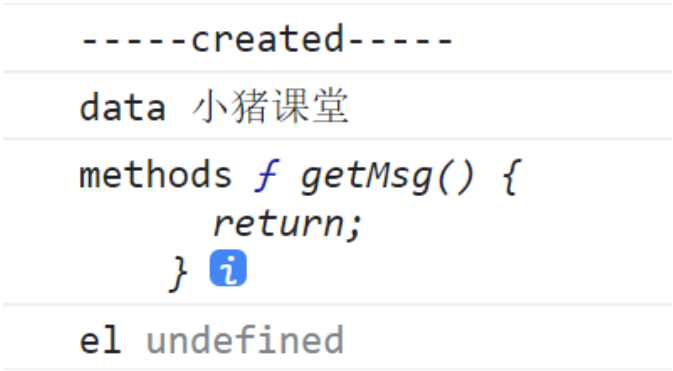
顾名思义，该钩子函数是在 Vue 实例化之后执行的，此时 data 和 methods 已经初始化完成了，可以供我们调用，但是模板还没有编译，也就是我们还不能获取到 DOM。

实例代码：

```
created() {
  console.info("-----created-----");
  console.info("data", this.msg);
  console.info("methods", this.getMsg);
  console.info("el", this.$el);
```

```
},
```

输出结果：



可以看到 data 和 methods 可以获取到了，但是 el 节点还不能获取，因为此时模板渲染还没有开始。

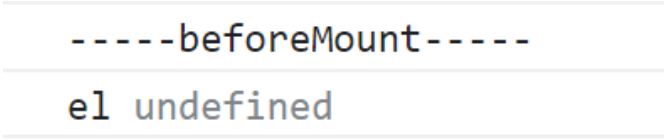
4.beforeMount

该钩子函数在模板渲染之前调用，也就是 DOM 节点挂载到真实 DOM 树之前调用。此模板进行编译，会调用 render 函数生成 vDom，也就是虚拟 DOM，此时我们同样无法获取 DOM 节点。

示例代码：

```
beforeMount(){
  console.info("-----beforeMount-----");
  console.info("el", this.$el);
},
```

输出结果：



可以看到此时我们同样是无法获取 DOM 节点的，因为此时只存在 VDOM，还在 JS 级别。

5.mounted

执行该钩子函数时，我们的模板编译好了，而且挂载到真实 DOM 树上面去了，也就是我们的页面可以显示了。

示例代码：

```
mounted(){
  console.info("-----mounted-----");
  console.info("el", this.$el);
},
```

输出结果：

```
-----mounted-----
el  ▶<div id="app">…</div>
```

此时我们可以获取到 DOM 节点了。

6.beforeUpdate

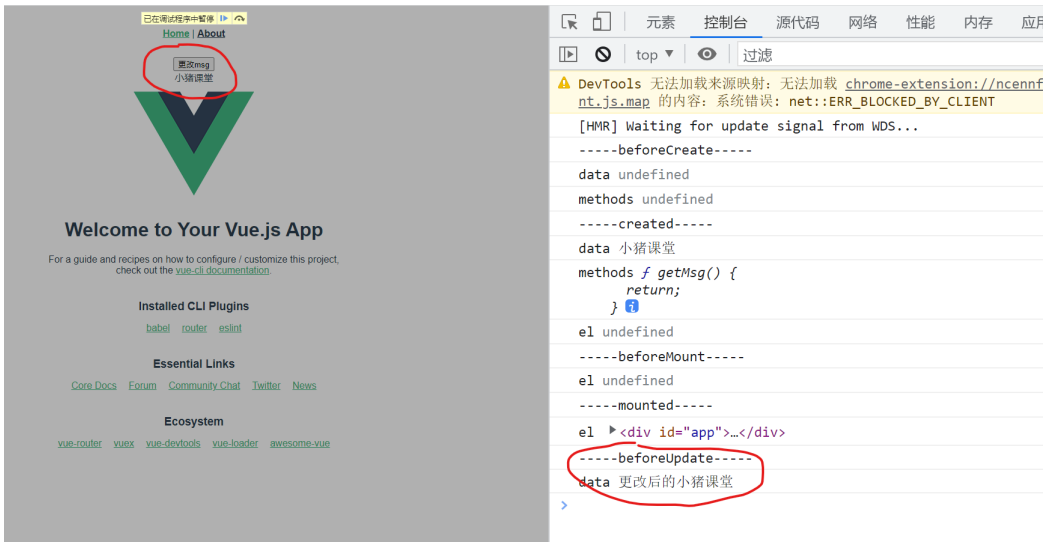
上面的生命周期函数其实都发生在初始化阶段，当我们的页面或者组件发生变化时，便会执行对应的更新阶段的钩子函数。

该钩子函数在 data 数据发生变化之后调用，此时 data 里面的数据已经是最新的了，但是页面上 DOM 还没有更新最新的数据。

示例代码：

```
beforeUpdate() {
  console.info("-----beforeUpdate-----");
  console.info("data", this.msg);
},
methods: {
  getMsg() {
    return;
  },
  changeMsg() {
    this.msg = "更改后的小猪课堂";
  },
},
```

输出结果：



上段代码中我们新增了一个改变 data 数据的方法，点击页面上的按钮便会更改数据，然后我们在 beforeUpdate 钩子函数后面打了一个断点。此时会发现控制台打印的 msg 和页面上的 msg 不一致，这也就印证 beforeUpdate 发生在 data 更新之后，DOM 渲染之前。

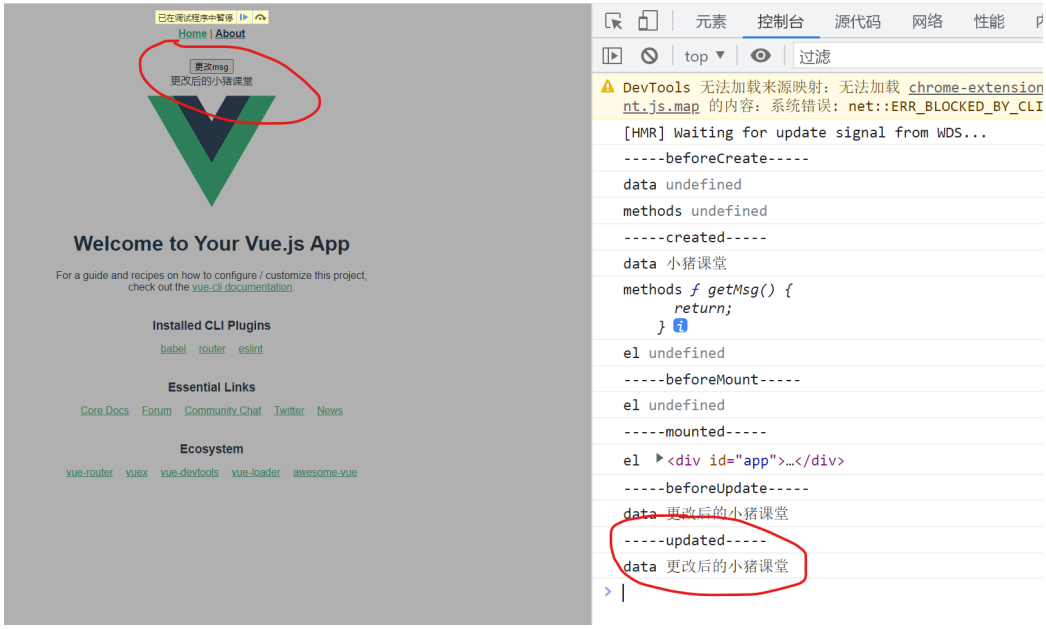
7.updated

该钩子函数会在 data 数据更新之后执行，而且此时页面也渲染更新完成了，显示的就是最新的数据。

示例代码：

```
updated(){
  console.info("-----updated-----");
  console.info("data", this.msg);
},
methods: {
  getMsg() {
    return;
  },
  changeMsg() {
    this.msg = "更改后的小猪课堂";
  },
},
```

输出结果：



上段代码基本和 beforeUpdate 代码一样，我们在 updated 钩子函数后面打了一个断点，当执行该钩子函数的时候，我们发现此时页面上的数据和我们 data 中的数据是一致的，说明此时 data 和页面都已经完成了更新。

注意：不要在 updated 中修改 data 数据，很容易造成死循环。

8.beforeDestroy

此钩子函数发生在 Vue 组件实例销毁之前，此时组件实际上还没有被销毁，还可以正常使用。我们通常会在这个钩子函数里面解除一些全局或者自定义事件。

示例代码：

```
beforeDestory(){
  console.info("-----beforeDestory-----");
  console.info("组件销毁之前");
},
```

9.destroyed

此钩子函数会在组件实例销毁之后执行，此时所有的组件包括子组件都被销毁了。

示例代码：

```
destroyed(){
  console.info("-----destroyed-----");
  console.info("组件被销毁");
},
```

通常这个时候只剩下了 DOM 空壳了。

10.补充

通常情况下我们切换路由等一些操作时，组件都会被销毁，切换回来的时候组件又重新渲染。但是有时候我们为了提高性能，我们可以在切换路由的时候不必销毁组件，这个时候我们就需要用到 Vue 的一个内置组件 keep-alive。

使用 keep-alive 组件包裹后的组件则不会被销毁，使用 keep-alive 这种方式之后，我们便会多出两个生命周期函数。

10.1 activated

页面渲染的时候执行。

实例代码：

```
<keep-alive>
  <router-view />
</keep-alive>
activated(){
  console.log("home 组件被渲染")
}
```

输出结果：

> Home页面的mounted执行
>
> home组件被渲染
>
> home组件被渲染

上段代码中我们将路由组件使用 keep-alive 包裹了起来，当我们第一次渲染组件的时候会执行组件的 mounted 钩子函数，但是当我们不停切换路由的时候，mounted 钩子函数没有被执行

了，只执行了 activated 钩子函数，说明组件并没有被销毁。

10.2 deactivated

页面被隐藏或者页面即将被替换成新的页面时被执行。

实例代码：

```
deactivated(){
  console.info("deactivated 被执行")
}
```

输出结果：

当我们的组件将要被切换走的时候，会执行此钩子函数，有点类似于 beforeDestroy 钩子函数，但是效果完全是不一样的。

总结

Vue 的生命周期函数是必须要掌握的，很多面试者常常觉得简单，不去认真学习，一心钻研项目，导致面试的时候说的不全面，最终被淘汰，大家一定要重视起来。

如果觉得文章不过瘾，可以去看我 B 站录制的视频： 小猪课堂