24.实例和生命周期

学习要点：

1.应用实例

2.生命周期

本节课我们来开始学习 Vue3.x 的应用实例和生命周期的问题。

一. 应用实例

1. 首先，回顾一下：我们创建一个应用实例，具体方案如下：

```
// 创建一个应用实例，{}里是选项对象
const app = Vue.createApp({})

// 这里的 app 是实例
console.log(app)

// app.mount()这个返回的是一个代理对象，并不返回应用本身
// 它返回的是这个组件本身，也就是根组件
const vm = app.mount('#app')
```

PS：.mount()方法作用是挂载 DOM 元素，是 Vue 的应用 API，并允许链式调用；

PS：在“API 参考”中找到“应用 API”目录，可以参考更多的应用 API；

PS：目前学习了两个，另一个是注册全局组件的.component()；

2. 由于，.mount()返回的是根组件实例对象，那么它可以直接调用 data()内的数据；

```
const app = Vue.createApp({
    data() {
        return {
            count : 100
        }
    }
})

const vm = app.mount('#app')
console.log(vm.count)
```

二. 生命周期

1. 首先，可以参考一下文档中生命周期的图示，看下它执行的流程；
2. 其次，我们要了解一下常用的声明周期的钩子，具体如下：

(1) .created 钩子：当实例被创建后会执行(在模板渲染前执行，一般用于初始化属性值)；

(2) .mounted 钩子：当实例被挂载后会执行(在模板渲染完成后执行，此时可以操作 DOM)；

(3) .updated 钩子：当虚拟 DOM 被修改后会执行；

```
<div id="app">
    <div id="abc">{{count}}</div>
</div>

// 创建一个应用实例，{}里是选项对象
const app = Vue.createApp({
    data() {
        return {
            count : 0
        }
    },

    // 初始化
    created() {
        this.count = 100
        console.log('初始化~')

        // 无法获取 DOM
        //console.log(document.getElementById('abc').innerText)
    },

    // 挂载后执行
    mounted() {
        // 获取 DOM
        console.log(document.getElementById('abc').innerText)
    },

    // 数据被修改后执行
    updated() {
        console.log('数据被修改了~')
    }

})
```