Sunday • 更新于 2022-10-19

上一节 03：编程范式之… 05：企业应用的… 下一节

04：命令式 VS 声明式

那么在我们讲解完成 **命令式** 和 **声明式** 之后，很多同学肯定会对这两种编程范式进行一个对比。

是命令式好呢？还是声明式好呢？

那么想要弄清楚这个问题，那么我们首先就需要先搞清楚，评价一种编程范式好还是不好的标准是什么？

通常情况下，我们评价一个编程范式通常会从两个方面入手：

1. 性能
2. 可维护性

那么接下来我们就通过这两个方面，来分析一下命令式和声明式。

性能

性能一直是我们在进行项目开发时特别关注的方向，那么我们通常如何来表述一个功能的性能好坏呢？

我们来看一个例子：

> 为指定 div 设置文本为 “hello world”

那么针对于这个需求而言，最简单的代码就是：

代码块

```
div.innerText = "hello world" // 耗时为：1
```

你应该找不到比这个更简单的代码实现了。

那么此时我们把这个操作的 **耗时** 比作： `1` 。（PS：**耗时越少，性能越强**）

然后我们来看声明式，声明式的代码为：

代码块

```
<div>{{ msg }}</div>  <!-- 耗时为：1 + n -->
<!-- 将 msg 修改为 hello world -->
```

那么：**已知修改 `text` 最简单的方式是 `innerText` **，所以说无论声明式的代码是如何实现的文本切换，那么它的耗时一定是 `> 1` 的，我们把它比作 `1 + n`（对比的性能消耗） 。

所以，由以上举例可知：**命令式的性能 > 声明式的性能**

可维护性

可维护性代表的维度非常多，但是通常情况下，所谓的可维护性指的是：对代码可以方便的 **阅读、修改、删除、增加** 。

那么想要达到这个目的，说白了就是：**代码的逻辑要足够简单**，让人一看就懂。

那么明确了这个概念，我们来看下命令式和声明式在同一段业务下的代码逻辑：

代码块

```
// 命令式
// 1. 获取到第一层的 div
const divEle = document.querySelector('#app')
// 2. 获取到它的子 div
const subDivEle = divEle.querySelector('div')
// 3. 获取第三层的 p
const subPEle = subDivEle.querySelector('p')
// 4. 定义变量 msg
const msg = 'hello world'
// 5. 为该 p 元素设置 innerHTML 为 hello world
subPEle.innerHTML = msg
```

代码块

```
// 声明式
<div id="app">
  <div>
    <p>{{ msg }}</p>
  </div>
</div>
```

对于以上代码而言，**声明式** 的代码明显更加利于阅读，所以也更加利于维护。

所以，由以上举例可知：**命令式的可维护性 < 声明式的可维护性 **

总结

由以上分析可知两点内容：

1. **命令式的性能 > 声明式的性能**
2. **命令式的可维护性 < 声明式的可维护性**

那么双方各有优劣，我们在日常开发中应该使用哪种范式呢？

想要搞明白这点，那么我们还需要搞明白更多的知识。

请看下章：**企业应用 && 框架 开发与设计原则**

03：编程范式之声明式编程 ◂ 上一节　　下一节 ▸ 05：企业应用的开发与设计原则

我要提出意见反馈

企业服务　网站地图　网站首页　关于我们　联系我们　讲师招募　帮助中心　意见反馈　代码托管

意见反馈　收藏教程　标记书签