

全部开发者教程

ELEMENT 节点的卸载操作

11：源码阅读：class 属性和其他属性的区分挂载

12：深入属性挂载：HTML Attributes 和 DOM Properties

13：框架实现：区分处理 ELEMENT 节点的各种属性挂载

14：源码阅读：ELEMENT 节点下，style 属性的挂载和更新

15：框架实现：ELEMENT 节点下，style 属性的挂载和更新

16：源码阅读：ELEMENT 节点下，事件的挂载和更新

17：深入事件更新：vue event invokers

18：框架实现：ELEMENT 节点下，事件的挂载和更新



Sunday • 更新于 2022-10-19

上一节 16：源码阅读：... 18：框架实现：... 下一节

17：深入事件更新：vue event invokers

那么这一小节我们来看下 `invokers` 对象 和 `invoker` 函数（即：vue event invokers。简称：`vei`）** 在事件处理中存在的作用。

在上一小节中，我们进行了两次挂载和一次卸载操作，本质上并没有对事件进行 **更新** 操作。而 `vei` 的作用是在更新中，才可以体现的。

我们知道，`vue` 对事件的处理是通过：

- `el.addEventListener`
- `el.removeEventListener`

来完成的。

那么现在我们来思考一下：

如果一个 `button` 最初的 `click` 事件，点击之后打印 `hello`
两秒之后，更新打印 `你好`

那么这样的一个更新操作，如果让我们通过 `el.addEventListener` 和 `el.removeEventListener` 来实现，那么我们会怎么做？

可能有同学说，这还不简单吗？很轻松的写出如下代码：

<> 代码块

```
1 <script>
2   const btnEle = document.querySelector('button')
3   // 设置初始点击行为
4   const invoker = () => {
5     alert('hello')
6   }
7   btnEle.addEventListener('click', invoker)
8
9   // 两秒之后，更新点击事件
10  setTimeout(() => {
11    // 先删除
12    btnEle.removeEventListener('click', invoker)
13    // 再添加
14    btnEle.addEventListener('click', () => {
15      alert('你好')
16    })
17  }, 2000);
18
19 </script>
```

但是我们知道如果频繁的删除、新增事件是非常消耗性能的，那么有没有更好的方案呢？

肯定是有，这个方案就是 `vei`，我们来看下面这段代码：

<> 代码块

```
1 <script>
```

意见反馈

收藏教程

标记书签

索引目录

17：深入事件更新



```

4   const invoker = () => {
5     invoker.value()
6   }
7   // 为 invoker 指定了 value 属性，对应的值是《事件点击行为》
8   invoker.value = () => {
9     alert('hello')
10  }
11  // 把 invoker 作为回调函数，invoker 内部通过触发 value，来触发真正的点击行为
12  btnEle.addEventListener('click', invoker)
13
14  // 两秒之后更新
15  setTimeout(() => {
16    // 因为真正的事件点击行为其实是 invoker.value，所以我们想要更新事件，就不需要再次触发 addE
17    invoker.value = () => {
18      alert('你好')
19    }
20  }, 2000);
21 </script>

```

vue 就是通过这样一种方式，来完成的事件更新操作。具体更新代码比较简单，可以看 `packages/runtime-dom/src/modules/events.ts` 第 77 行：

<> 代码块

```

1   if (nextValue && existingInvoker) {
2     // patch
3     existingInvoker.value = nextValue
4   }

```

至于 `invokers` 则充当了一个事件缓存器，把所有的事件：**以事件名为 key，以事件行为为 value**。保存到 `el._vei` 中。

那么这样我们搞明白了 `vei` 它在事件处理中所存在的意义。

明白了这个之后，下面我们就可以实现 `event` 事件的挂载和更新操作了。

16: 源码阅读: ELEMENT 节点下, 事件的... ◀ 上一节 下一节 ▶ 18: 框架实现: ELEMENT 节点下, 事件的...

✎ 我要提出意见反馈