双11返场进行中！优惠Go on，快来捡漏！ 02 天 14 时 48 分 39 秒 Go >>>

Sunday • 更新于 2022-10-19

上一节 15：框架实现：... 17：深入事件更... 下一节

16：源码阅读：ELEMENT 节点下，事件的挂载和更新

在处理完成 `style` 的挂载和更新之后，接下来我们来看 `event` **事件** 的挂载和更新操作。

和之前一样，我们首先先创建对应的测试实例 `packages/vue/examples/imooc/runtime/render-element-event.html`：

代码块

```
<script>
  const { h, render } = Vue

  const vnode = h('button', {
    // 注意：不可以使用 onclick。因为 onclick 无法满足 /^on[^a-z]/ 的判断条件，这会导致 event
    onClick() {
      alert('点击')
    },
  }, '点击')
  // 挂载
  render(vnode, document.querySelector('#app'))

  setTimeout(() => {
    const vnode2 = h('button', {
      onDblclick() {
        alert('双击')
      },
    }, '双击')
    // 挂载
    render(vnode2, document.querySelector('#app'))
  }, 2000);
</script>
```

> **注意：**不可以使用 `onclick`。因为 `onclick` 无法满足 `/^on[^a-z]/` 的判断条件，这会导致 `event` 通过：`el[key] = value` 的方式绑定（虽然这样也可以绑定 `event`），从而无法进入 `patchEvent`。
> 在项目中，当我们通过 @click 绑定属性时，会得到 `onClick` 选项

在 `packages/runtime-dom/src/patchProp.ts` 中进入 `debugger`：

1. 第一次进入 `debugger`，执行 **挂载** 操作：
    1. 此时各参数为：

```
export const patchProp: DOMRendererOptions['patchProp'] = (
  el,  el = button {disabled: false, form: null, formAction: 'http://127.0.0.
  key,  key = "onClick"
  prevValue,  prevValue = null
  nextValue,  nextValue = f onClick()
  isSVG = false,  isSVG = false
  prevChildren,  prevChildren = "点击"
  parentComponent,  parentComponent = null
  parentSuspense,  parentSuspense = null
  unmountChildren  unmountChildren = (children, parentComponent, parentSusper
) => {
```

意见反馈 收藏教程 标记书签

1. 进入 `isOn` 方法：

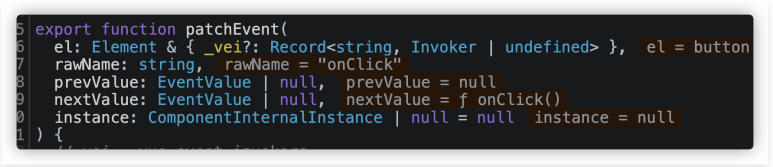
```
代码块
1    const onRE = /^on[^a-z]/
2    export const isOn = (key: string) => onRE.test(key)
```

2. 整体的代码比较简单，就是筛选出：** `on` 开头，不接 `a-z` ** 的字符串。

3. 即： `/^on[^a-z]/.test('onClick')`

3. 当前 满足条件，触发 `patchEvent(el, key, prevValue, nextValue, parentComponent)` 方法：

1. 进入 `patchEvent` 方法，此时各参数为：



2. 执行 `const invokers = el._vei || (el._vei = {})`：

1. 这里涉及到了一个 `_vei` 对象
2. `vue` 对其进行了注释： `vei = vue event invokers` ，即：**VUE事件调用者**
3. 那么这个事件调用者是什么意思呢?
4. 不要着急，我们继续往下看，大家现在需要记住，我们当前得到了一个 `invokers` 对象： `invokers = {}`

3. 执行 `const existingInvoker = invokers[rawName]`：

1. 因为当前 `invokers` 为 `{}`
2. 所以得到的 `existingInvoker = undefined`

4. 执行 `else` 判断条件：

1. 执行 `const [name, options] = parseName(rawName)`：

1. 进入 `parseName`
2. 该函数的作用就是拆解除 **事件名** `name` 和 addEventListener 的 `options`
3. 我们这了可以直接忽略掉 `options`
4. 得到 `name` 即可

2. 此时 `name = click`
3. 执行 `if (nextValue)`，当前 `nextValue: ƒ onClick()` 函数

1. 进入 `if` 判断
2. 执行 `const invoker = (invokers[rawName] = createInvoker(nextValue, instance))`

1. 进入 `createInvoker` 方法：
2. 这里面的代码做了两件事情：

1. `invoker.value = initialValue`：

1. 这个是需要我们 **重点关注** 的
2. 当前的 `initialValue` 即为 `nextValue`
3. 即： `invoker` 对象的 `value` 属性即为 `onClick` 函数

2. 以 `invoker.attached = getNow()` 为主的 **时间** 处理

意见反馈 收藏教程 标记书签

2. 主要应用于 `onClick` 为 **数组[]** 时，多个回调方法的触发时机问题
3. 我们这里 **无需关注**

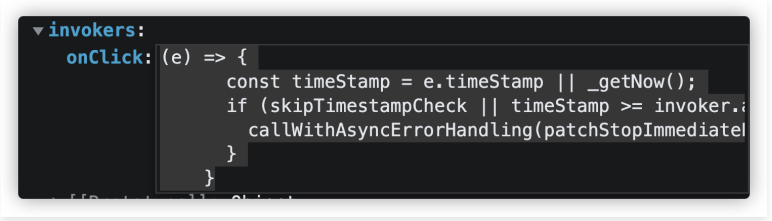
3. 得到 `invoker` **函数**，并为 `invokers[rawName]` 进行了 **缓存**
4. 执行 `addEventListener(el, name, invoker, options)`

1. 该方法的代码比较简单，就是执行了 `el.addEventListener(event, handler, options)`
2. 对比参数，即执行了：`el.addEventListener(name, invoker, options)`

2. 支持事件 **挂载** 完成
3. 等待两秒之后，第二次进行，执行 **更新** 操作：

1. 进入 `patchEvent` 方法
2. 因为在 **挂载** 时，我们已经进行了 `invoker` 的 **缓存**，所以再次进入时：

1. `invokers` 不在为 `null`，而是一个对象，里面包含了上一次的 `onClick` 方法



3. `existingInvoker` 依然为 `null`
4. 进入 `else`：

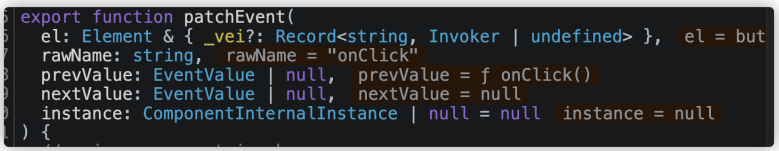
1. 生成 `invoker` 函数
2. 通过 `addEventListener` 进行挂载

4. 那么此时 **双击** 事件被 **挂载完成。**
5. 但是大家到此时可能会非常奇怪，目前为止我们分别完成了 **单击事件的挂载** 和 **双击事件的挂载**，但是有一个问题，那就是 **旧事件（单击事件）此时依然存在**，并没有被 **卸载。**这是为什么呢?
6. 我们知道 **属性的挂载** 其实是在 `packages/runtime-core/src/renderer.ts` 中的 `patchProps` 中进行的，观察内部方法，我们可以发现 **内部进行了两次** `for` **循环**：

1. 第一次是新增新属性
2. 第二次是卸载旧属性

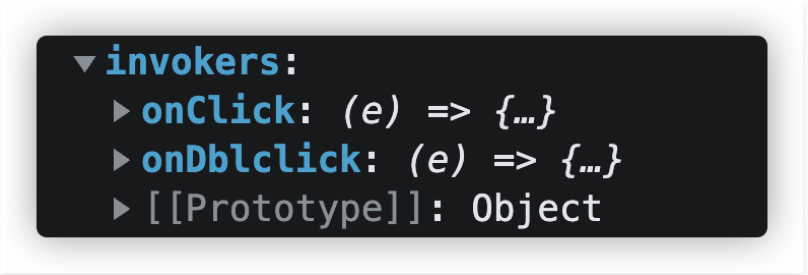
7. 所以说，此时，我们还会 **第三次** 进入 `patchProp` 方法，本次的目的是：**卸载** `onClick`

1. 忽略相同逻辑，同样进入 `patchEvent` 方法，此时的参数为：



2. 此时 `invokers` 的值为：

意见反馈 收藏教程 标记书签



3. 此时 `existingInvoker` 将存在值，值为 `onClick` **的回调方法**
4. 再次进入 `else`
    1. **注意：** 此时因为 `nextValue` 为 `null` ，而 `existingInvoker` 存在
    2. 所以会走：

```
removeEventListener(el, name, existingInvoker, options)
invokers[rawName] = undefined
```

8. 至此 **卸载旧事件** 完成

由以上代码可知：

1. 我们一共三次进入 `patchEvent` 方法
    1. 第一次进入为 **挂载** `onClick` 行为
    2. 第二次进入为 **挂载 onDblclick** 行为
    3. 第三次进入为 **卸载** `onClick` 行为
2. 挂载事件，通过 `el.addEventListener` 完成
3. 卸载事件，通过 `el.removeEventListener` 完成
4. 除此之外，还有一个 `_vei` 即 `invokers` 对象 和 `invoker` 函数，我们说两个东西需要重点关注，那么这两个对象有什么意义呢？

下一小节，我们将详细说明 `invokers` 对象 和 `invoker` 函数 在当前事件处理中存在的作用。

15：框架实现：ELEMENT 节点下， style 属... ‹ 上一节　　下一节 › 17：深入事件更新：vue event invokers

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

Copyright © 2022 imooc.com All Rights Reserved | 京ICP备 12003892号-11　京公网安备110108020301151号

意见反馈　收藏教程　标记书签