Sunday • 更新于 2022-10-19

上一节 09：框架实现：... 11：框架实现：... 下一节

10：源码阅读：对 class 和 style 的增强处理

vue 对 class 和 style 做了专门的增强，使其可以支持 Object 和 Array 。

比如说，我们可以写如下测试案例 packages/vue/examples/imooc/runtime/h-element-class.html：

代码块

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

  const vnode = h('div', {
    class: {
      'red': true
    }
  }, '增强的 class')

  render(vnode, document.querySelector('#app'))
</script>
```

这样，我们可以得到一个 class: red 的 div。

这样的 h 函数，最终得到的 vnode 如下：

代码块

```
{
  __v_isVNode: true,
  type: "div",
  shapeFlag: 9,
  props: {class: 'red'},
  children: "增强的 class"
}
```

由以上的 VNode 可以发现，最终得出的 VNode 与

代码块

```
  const vnode = h('div', {
    class: 'red'
  }, 'hello render')
```

是完全相同的。

那么 vue 是如何来处理这种增强的呢？

我们下面就来一探究竟（style 的增强处理与 class 非常相似，所以我们只看 class 即可）：

1. 进入 _createVNode 的 debugger （**仅关注 class 的处理**）
2. 此时 props 为：

代码块

```
  props: {
      class: {
      'red': true
```

3. 执行 `if (props)`，存在。进入 判断：

    1. 执行 `let { class: klass, style } = props`，得到 `klass: {red: true}`
    2. 执行 `props.class = normalizeClass(klass)`，这里的 `normalizeClass` 方法就是处理 `class` **增强的关键**：

        1. 进入 `normalizeClass` 方法：

代码块

```
import { isArray, isObject, isString } from '.'

/**
 * 规范化 class 类，处理 class 的增强
 */
export function normalizeClass(value: unknown): string {
    let res = ''
    // 判断是否为 string，如果是 string 就不需要专门处理
    if (isString(value)) {
        res = value
    }
    // 额外的数组增强。官方案例：https://cn.vuejs.org/guide/essentials/class-an
    else if (isArray(value)) {
        // 循环得到数组中的每个元素，通过 normalizeClass 方法进行迭代处理
        for (let i = 0; i < value.length; i++) {
            const normalized = normalizeClass(value[i])
            if (normalized) {
                res += normalized + ' '
            }
        }
    }
    // 额外的对象增强。官方案例：https://cn.vuejs.org/guide/essentials/class-an
    else if (isObject(value)) {
        // for in 获取到所有的 key，这里的 key（name） 即为 类名。value 为 boole
        for (const name in value as object) {
            // 把 value 当做 boolean 来看，拼接 name
            if ((value as object)[name]) {
                res += name + ' '
            }
        }
    }
    // 去左右空格
    return res.trim()
}
```

        2. 此时 `props` 的 `class` 即为 `red`

由以上代码可知：

1. 对于 `class` 的增强其实还是比较简单的，只是额外对 `class` 和 `style` 进行了单独的处理。
2. 整体的处理方式也比较简单：
    1. 针对数组：进行迭代循环
    2. 针对对象：根据 `value` 拼接 `name`

09：框架实现：实现剩余场景 Text09：框架... ◂ 上一节　　下一节 ▸ 11：框架实现：完成虚拟节点下的 class 和 s...

我要提出意见反馈