全部开发者教程

04：源码阅读：h 函数，跟踪 ELEMENT + ARRAY_CHILDREN 场景下的源码实现

05：框架实现：构建 h 函数，处理 ELEMENT + ARRAY_CHILDREN 场景

06：源码阅读：h 函数，组件的本质与对应的 VNode

07：框架实现：处理组件的 VNode

08：源码阅读：h 函数，跟踪 Text 、 Comment、 Fragment 场景

09：框架实现：实现剩余场景 Text09：框架实现：实现剩余场景 Text 、 Comment、 Fragment

10：源码阅读：对 class 和 style 的增强处理

Sunday • 更新于 2022-10-19

上一节 05：框架实现：... 07：框架实现：... 下一节

06：源码阅读：h 函数，组件的本质与对应的 VNode

组件是 vue 中非常重要的一个概念，这一小节我们就来看一下 **组件** 生成 VNode 的情况。

在 vue 中，组件本质上是 **一个对象或一个函数（** Function Component **）**

> 我们这里 **不考虑** 组件是函数的情况，因为这个比较少见。

我们可以直接利用 h 函数 + render 函数渲染出一个基本的组件：

1. 创建 packages/vue/examples/imooc/runtime/h-component.html

代码块

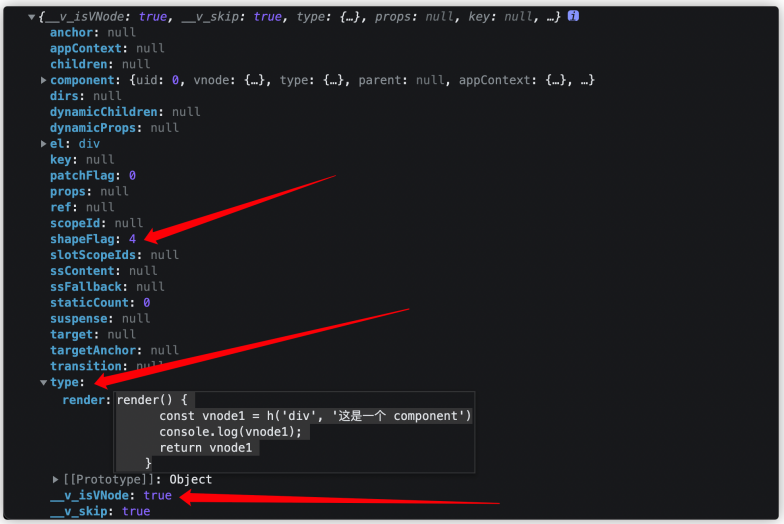
```
<script>
  const { h, render } = Vue

  const component = {
    render() {
      const vnode1 = h('div', '这是一个 component')
      console.log(vnode1);
      return vnode1
    }
  }
  const vnode2 = h(component)
  console.log(vnode2);
  render(vnode2, document.querySelector('#app'))

</script>
```

2. 在当前代码中共触发了两次 h 函数，我们来查看两次打印的结果：

    1. vnode 2 ：



索引目录

06：源码阅读：h

1. `shapeFlag`：这个是当前的类型表示，`4` 表示为一个 组件
2. `type`：是一个 **对象**，它的值包含了一个 `render` 函数，这个就是 `component` 的 **真实渲染** 内容
3. `__v_isVNode`： `VNode` 标记

2. `vnode1`：与 `ELEMENT + TEXT_CHILDREN` 相同

代码块

```
{
  __v_isVNode: true,
  type: "div",
  children: "这是一个 component",
  shapeFlag: 9
}
```

那么由此可知，对于 **组件** 而言，它的一个渲染，与之前不同的地方主要有两个：

1. `shapeFlag === 4`
2. `type`：是一个 **对象（组件实例）**，并且包含 `render` 函数

仅此而已，那么依据这样的概念，我们可以通过如下代码，完成同样的渲染：

代码块

```
const component = {
  render() {
    return {
      "__v_isVNode": true,
      "type": "div",
      "children": "这是一个 component",
      "shapeFlag": 9
    }
  }
}

render({
  "__v_isVNode": true,
  "type": component,
  "shapeFlag": 4
}, document.querySelector('#app'))
```

05：框架实现：构建 h 函数，处理 ELEMEN... ◂ 上一节　　下一节 ▸ 07：框架实现：处理组件的 VNode

我要提出意见反馈