张轩 • 更新于 2023-07-24

上一节 Vitest 总结 下一节

Vnode 以及 Render Function

文档地址：https://cn.vuejs.org/guide/extras/rendering-mechanism.html

Virtual DOM：一种虚拟的，保存在内存中的数据结构，用来代表 UI 的表现，和真实 DOM 节点保持同步。Virtual DOM是由一系列的 Vnode 组成的。

代码块

```
// 模拟一个简单的 Vnode
const vnode = {
  type: 'div',
  props: {
    id: 'hello'
  },
  children: [
    /* more vnodes */
  ]
}
```

Render Pipeline

- **Compile**：Vue 组件的 Template 会被编译成 **render function**：一个可以返回 Virtual DOM 树的函

- **Mount**，执行 render function，遍历虚拟DOM 树，生成真正的 DOM 节点。
- **Patch**，当组件中任何响应式对象（依赖）发生变化的时候，执行更新操作 。生成新的虚拟节点树，Vue 内部会遍历新的虚拟节点树，和旧的树做对比，然后执行**必要**的更新。

虚拟DOM 的优点

- 可以使用一种更方便的方式，供开发者操控 UI 的状态和结构，不必和真实的DOM 节点打交道。
- 更新效率更高，计算需要的最小化操作，并完成更新。

看一下 Render Functions

```
// 在 main.ts 中
console.log(App2)
// 返回
{
  render: ƒ _sfc_render(_ctx, _cache, $props, $setup, $data, $options)
  setup: ƒ setup(__props, { expose })
  __file: "/Users/liusha/Public/v-element/src/App2.vue"
  __hmrId: "e16649ff"
  __name: "App2"
}
// 原始的 template
<template>
  <Button>button</Button>
</template>

// template 会被转换成这样的 function
function _sfc_render(_ctx, _cache, $props, $setup, $data, $options) {
  return _openBlock(), _createBlock($setup["Button"], null, {
    default:  withCtx(() => [
```

```
      ]),
      _: 1
      /* STABLE */
    });
  }
```

- Template 比 render function 更接近 html，更好懂，更容易修改。
- Template 更容易做静态优化，Vue 的 compiler 在编译过程中可以做很多自动的性能优化。

在实践中，templates适应大多数的情况，但是在少数情况下，还是需要学习使用 render function。因为它本身是 javascript 语法，要更灵活多变。Vue 提供对应的 API 可以不使用 templates，而是直接使用 render function。

创建 Vnode

文档地址：https://cn.vuejs.org/guide/extras/render-function.html

h 和 createVnode 都可以创建 vnode，h 是 hyperscript 的缩写，意思就是 “JavaScript that produces HTML (hypertext markup language)”，很多 virtualDOM 的实现都使用这个函数名称。还有一个函数称之为 createVnode，更形象，两个函数的用法几乎是一样的。

代码块

```
import { h, createVnode } from 'vue'

const vnode = h(
  'div', // type
  { id: 'foo', class: 'bar' }, // props
  [
    /* children */
  ]
)
```

声明 Render Function

代码块

```
import { ref, h } from 'vue'

export default {
  props: {
    /* ... */
  },
  setup(props) {
    const count = ref(1)

    // 返回渲染函数
    return () => h('div', props.msg + count.value)
  }
}
```

测试的一些生命周期函数

https://vitest.dev/api/#setup-and-teardown

Vitest 上一节 下一节 总结