

# mve 从一个简单的todo开始——使用vite

前端打包工具目前了解的有gulp\webpack\vite。gulp用得少，vite据宣传比webpack快不少，又是最新的技术，应该有不少优势。

## 一、准备vite环境

1. 首先打开vite的官网，根据文档，创建一个vanilla的项目，下一步选择ts。
2. 在vscode里打开该项目，使用npm i或类似命令加载依赖包。
3. 通过查看package.json里的scripts下允许的脚本，先尝试使用

```
npm run dev
```

启动项目。在浏览器打开控制台的url。

4. 尝试修改main.ts的内容，将Hello Vite改成Hello Vite with mve。效果很好，浏览器里的文字立即变成了后者。

## 二、准备mve环境

1. 加载mve模块包。

```
npm i mve-dom --save
```

2. 对main.ts中的代码作相应的调整：

```
import { dom } from 'mve-dom'
import './style.css'

const app = document.querySelector<HTMLDivElement>('#app')!

// app.innerHTML = `
//   <h1>Hello Vite with mve!</h1>
//   <a href="https://vitejs.dev/guide/features.html"
//     target="_blank">Documentation</a>
// `

const root=dom.root(function(me) {
  return {
    type:"div",
    text:"Hello Vite With mve"
  }
})
app.append(root.element)
if(root.init){
  root.init()
}
```

```
if(root.destroy){
  window.addEventListener("close",root.destroy)
}
```

即注释掉原来的app.innerHTML..., 加上后续mve与原生dom的桥梁逻辑。需要引入 mve-dom中定义的dom。init与destroy是mve的原生生命周期钩子函数。

3. 可能需要刷新界面, 可以看到浏览器已经是调整后的样式了。

## 三、制作一个简单的todo

1. 定义todo的记录模型。这里, 我们在src目录下新建一个模块, 叫todo.ts

```
import { mve } from "mve-core/util"

interface Todo{
  content:mve.Value<string>
  finish:mve.Value<boolean>
}

const todos=mve.arrayModelOf<Todo>([ ])
```

这里我们简单地定义两个字段, content为内容, finish为是否完成。导入mve模块, 模块里使用了mve的核心模型, 一个是mve.Value, 对应可修改的单值, 一个是mve.ArrayModel<T>, 对应可修改的列表。

2. 实现界面。先想一想, 一个todo界面, 有一个增加输入框, 有记录展示列表, 可以对展示记录修改、标记完成、删除。这里展示输入列表我们用table来简单实现, 第一列checkbox标记完成, 第二列展示内容input可以修改, 第三列删除按钮。先看代码:

```
import { mve } from "mve-core/util"
import { dom } from "mve-dom"
import {modelChildren} from "mve-core/modelChildren"

interface Todo{
  content:mve.Value<string>
  finish:mve.Value<boolean>
}

const todos=mve.arrayModelOf<Todo>([ ])

let addInput:HTMLInputElement
export const todoView=dom( {
  type:"div",
  children:[
    dom( {
      type:"input",
      init(e){
        addInput=e
      }
    })
  ]
})
```

```

    }
  )),
  dom({
    type: "button",
    text: "添加",
    event: {
      click() {
        const v = addInput.value.trim()
        if (v) {
          todos.unshift({
            content: mve.valueOf(v),
            finish: mve.valueOf(false)
          })
          addInput.value = ""
        } else {
          alert("请输入内容")
        }
      }
    }
  )),
  dom({
    type: "table",
    children: modelChildren(todos, function(me, todo, i) {
      return dom({
        type: "tr",
        children: [
          dom({
            type: "td",
            children: [
              dom({
                type: "input",
                attr: {
                  type: "checkbox"
                },
                prop: {
                  checked: todo.finish
                },
                event: {
                  change(e) {
                    const checkbox = e.target as HTMLInputElement
                    todo.finish(checkbox.checked)
                  }
                }
              })
            ]
          })
        ]
      })
    })
  )),
  dom({
    type: "td",
    children: [

```

```

        dom({
          type: "input",
          value: todo.content,
          event: {
            change(e) {
              const input = e.target as HTMLInputElement
              const v = input.value.trim()
              if (v) {
                todo.content(v)
              } else {
                input.value = todo.content()
              }
            }
          }
        })
      ]
    })),
    dom({
      type: "td",
      children: [
        dom({
          type: "a",
          text: "删除",
          attr: { href: "javascript:void(0)" },
          event: {
            click() {
              todos.remove(i())
            }
          }
        })
      ]
    })
  ]
})
})
})
})
})

```

代码可能比较长，因为mve的特点是使用纯ts而不使用xml，以实现最大化的代码复用。代码的逻辑很简单，可以看到dom函数就是类似于组装一个html元素如div、input，它有attr、prop、event属性，和html元素对应得上。可以跳转到mve-dom/index里查看DOMNode的定义。这样规划的原因是精简地使用dom的API。废话少说，到下一步。

3. 在main.ts里面关联起来。这里我们使用简单的组合方式。删除之前的text属性，加上children数组，数组加上todoView这个元素。

```

import { dom } from 'mve-dom'
import './style.css'

```

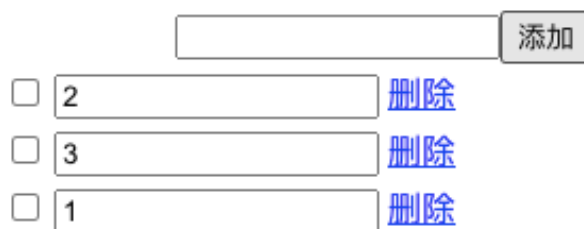
```
import { todoView } from './todo'

const app = document.querySelector<HTMLDivElement>('#app')!

// app.innerHTML = `
//   <h1>Hello Vite with mve!</h1>
//   <a href="https://vitejs.dev/guide/features.html"
//     target="_blank">Documentation</a>
// `

const root=dom.root(function(me) {
  return {
    type:"div",
    children:[
      todoView
    ]
  }
})
app.append(root.element)
if(root.init){
  root.init()
}
if(root.destroy){
  window.addEventListener("close",root.destroy)
}
```

保存，在浏览器中查看。



添加

☐ 2 删除

☐ 3 删除

☐ 1 删除

好像蛮丑。不过自己也没特别的见解。今天的主角是mve的用法。可以用F12查看我们创建的dom结构。

## 四、回顾我们做了什么

1. 我们创建了一个列表，可以对列表增删改查。
2. 可以浏览mve.Value的类型定义，发现它是一个函数，具有两种用法：不给值取、给值修改。我把它叫存储器。

3. 可以查看dom函数，在相同文件下可以看到还有一个类似svg函数。它们的参数里可以找到attr/text/children/style/event等与dom元素对应的属性。期中event对应响应事件，attr/style属性的map，其值可为具体值，也可为函数，为函数的时候，便类似于vue的属性绑定。其它类似的属性，不一一赘述。prop的map对应dom元素的js对象属性。源码不多，相信读源码比糟心写出来的文档更简单。
4. children属性的类型是 `EOChildren<EO>`，跳转定义可以看到其类型定义为：

```
/**重复的函数节点/组件封装成mve*/
export interface EOChildFun<EO>{
  (parent:VirtualChild<EO>,me:mve.LifeModel):BuildResult
}

/**存放空的生命周期 */
export class ChildLife{
  constructor(public readonly result:BuildResult){}
  static of(result:BuildResult){
    return new ChildLife(result)
  }
}

export type EOChildren<EO>= EO | EOChildFun<EO> | EOChildren<EO>[] | ChildLife
```

这里实现了函数子节点的嵌套。因此你可以灵活组合。

5. modelChildren实现了mve.ArrayModel到视图模型的绑定。在自己的mve实践中，90%是在使用这样的方式，或能用这样的方式替代。它类似于其它模板中的for语句。视图片段与模型片段一一对应，这是与vue、react不同的地方。mve不使用组件系统，代码基本上是通过这样的函数定义组合来的。其它可能会用到的是ifChildren，在mve-core/ifChildren下。
6. 如何实现filter功能？在Array中，filter返回了一个新的Array。但在mve中，虽然有类似的filter(mve-core/filterChildren)，但主推modelChildren这样的mvvm模式，可能需要在模型上加 `hide:mve.Value<boolean>` 字段，以与模型的display绑定。

## 五、后记

mve这个库我已经自迭代3-4年了，依赖统计是受vue启发，后期陆陆续续加入了很多惊喜的功能。当初主要是觉得js本身的DSL功能很强大不需要一套XML，而且不满新框架不支持工作中必须支持的IE8。当初我还宣传过，不过自己只是自嗨居多，并不是一个很好的讲述者，所幸没及早让人陪着折腾（我还自迭代了这么多）。

现在想做前端的工作基本上都是vue-react那一套，并不是很喜欢。觉得自己的mve很不错，是更纯正的mvvm，性能也能更高。至少也希望能启发改变前端的生态吧。

自己很熟悉的东西，总是不知道别人想了解什么，而基于假想的文章，会不会嫌啰嗦？更多想了解的，欢迎加群咨询。(qq:925964319)

本文的库

[mve-vite-demo: 使用vite来构建mve \(gitee.com\)](https://gitee.com/wangyang2010344/mve-vite-demo)

[wangyang2010344/mve-vite-demo \(github.com\)](https://github.com/wangyang2010344/mve-vite-demo)

