

# Vue3相关

## vue3的优点?

从使用语法上，比较大的区别就是多了组合式API，可以更好的组织逻辑。新增了 Suspense Teleport 组件，v-model 语法糖更新，去除 filter .sync 功能。

从框架本身看，首次渲染更快（proxy代理），diff 算法更快（静态标记），打包体积更小（tree-shaking），更好的支持TS，放弃IE浏览器。

## 响应式系统?


提供 reactive 和 ref 实现响应式数据。

reactive 是使用 proxy 实现数据劫持，不用遍历属性，支持删除属性和添加属性。vue2是单独额外处理的，vue2是使用 Object.defineProperty。

ref 如果是对象使用 proxy 实现数据劫持，如果是简单数据使用 obj 的 getter setter 代理value属性的写法实现的。

其他发布订阅，观察者模式基本一样。

## Vue2的数据劫持?



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>数据劫持-vue2</title>
  </head>
  <body>
    <div id="app"></div>

    <script>
      const target = {
        name: "jack",
        age: 18,
      };

      const vm = {};
      Object.keys(target).forEach((key) => {
        // 代理对象，原对象的key，get和set函数，这是ES5提供的函数
        Object.defineProperty(vm, key, {
          get() {
            return target[key];
          },
          set(v) {
            target[key] = v;
          }
        });
      });
    </script>
  </body>
</html>
```

```

        // 模拟更新效果
        render()
      },
    });
  });

  const render = () => {
    document.querySelector(
      "#app"
    ).innerHTML = `我是 ${vm.name} 今年 ${vm.age} 岁`;
  };
  render();
</script>
</body>
</html>

```

## Vue3的数据劫持?

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>数据劫持-vue3</title>
  </head>
  <body>
    <div id="app"></div>

    <script>
      const data = {
        name: "jack",
        age: 18,
        get fullName () {
          return this.name + 'Ma'
        }
      };

      const proxy = new Proxy(data, {
        get(target, key, receiver) {
          console.log('get')
          // return target[key]
          return Reflect.get(target, key, receiver)
        },
        set(target, key, value, receiver) {
          // target[key] = value
          Reflect.set(target, key, value, receiver)
          render()
        }
      });
    </script>
  </body>
</html>

```

```

        return true
      },
    });

    console.log(proxy)

    const render = () => {
      document.querySelector(
        "#app"
      ).innerHTML = `我是 ${proxy.name} 今年 ${proxy.age} 岁`;
    };
    render();
  </script>
</body>
</html>

```

## vite 构建工具优势?

先启动开发服务器（启动快），使用 ES Module 加载资源（热更新快），开发体验好。由于只支持ES6模块化，commonJs的代码需要改造，从 webpack（vue-cli）迁移到 vite（create-vue）还是有一些困难，建议新项目采用 vite。

## 组件如何才能支持v-model指令?

Vue2 `:value @input` === v-model `:show update:show` === .sync

Vue3 `:show @update:show` === v-model:show `:modelValue @update:modelValue` ===v-model

## 路由传参?

- 路由?
  - 路径 ==> 组件 `{path: '/user', component: User}`
  - 只有路径是 `/user` 才能来到 `User` 组件
  - 如果要传参?
    - `/user?id=100` 或者 `{path: '/user', query: {id: 100}}`
  - 如果取参数? `route.query`
- 动态路由?
  - 当你不同的路径，想指向同一个组件的时候使用: `/user/100` `/user/200`
  - 路径 ==> 组件 `{path: '/user/:id', component: User, name: 'user'}`
  - 如果要传参?
    - `/user/100` 或者 `{name: 'user', params: {id: 100}}`
  - 如果取参数? `route.params`

上面两种即是标准的路由写法和传参写法，其他的写法都不规范。如：

- 如何实现地址栏看不见的传参？
- vue2 (vue-router@3) 写法
  - `{path: '/user', component: User, name: 'user'}` 路由规则
  - `{name: 'user', params: {id: 100}}` 跳转路由传参ID
  - 这种写法地址栏不出现id，地址是 `/user` params上的id值没有在地址上，刷新后消失。

3 lines (583 sloc) | 56.4 KB

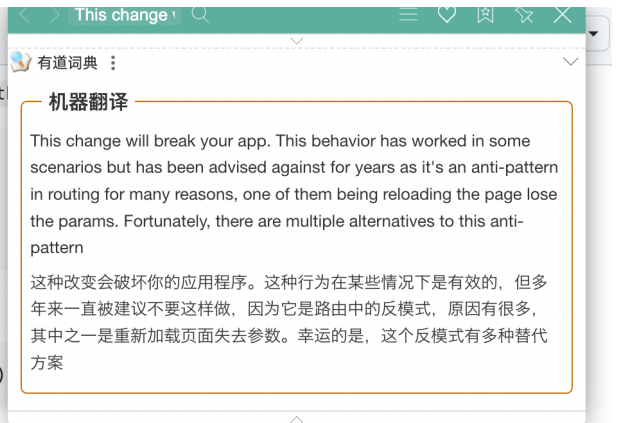
If you were relying on passing `params` that were not defined as part of the `path`:

```
{
  path: '/somewhere',
  name: 'somewhere'
}
```

And pushing with an *artificial* param:

```
router.push({ name: 'somewhere', params: { oops: 'gets removed' } })
```

This change will break your app. This behavior has worked in some scenarios but has been **advised against** for years as it's an anti-pattern in routing for many reasons, one of them being reloading the page lose the params. Fortunately, there are multiple alternatives to this anti-pattern:



vue3(vue-router@4)默认不支持了，代替方案：pinia

## TS相关

### 枚举类型编译结果？

- 枚举是 TS 为数不多的非 JavaScript 类型级扩展(不仅仅是类型)的特性之一
- 因为：其他类型仅仅被当做类型，而枚举不仅用作类型，还提供值(枚举成员都是有值的)
- 也就是说，其他的类型会在编译为 JS 代码时自动移除。但是，枚举类型会被编译为 JS 代码

```
enum Direction {
  Up = 'UP',
  Down = 'DOWN',
  Left = 'LEFT',
  Right = 'RIGHT'
}

// 会被编译为以下 JS 代码：
var Direction;

(function (Direction) {
  Direction['Up'] = 'UP'
  Direction['Down'] = 'DOWN'
```

```
Direction['Left'] = 'LEFT'
Direction['Right'] = 'RIGHT'
})(Direction || Direction = {})
```

- 说明：枚举与前面讲到的字面量类型+联合类型组合的功能类似，都用来表示一组明确的可选值列表
- 一般情况下，推荐使用字面量类型+联合类型组合的方式，因为相比枚举，这种方式更加直观、简洁、高效

## type 和 interface 区别？

- 类型别名和接口非常相似，在许多情况下，可以在它们之间 **自由选择**。
- 接口的几乎所有特性都以类型的形式可用，关键的区别在于不能重新打开类型以添加新属性，而接口总是 **可扩展** 的。

### interface

不支持简单类型

复用：可以继承

### type

支持：对象类型，其他类型

不同的点：

- type 不可重复定义

typescript

```
type Person = {
  name: string;
};
// 标识符"Person"重复  Error
type Person = {
  age: number;
};
```

- interface 重复定义会合并

typescript

```
interface Person {
  name: string;
}
interface Person {
  age: number;
}
// 类型会合并，注意：属性类型和方法类型不能重复定义
const p: Person = {
  name: 'jack',
  age: 18,
};
```

使用场景：扩展第三方库的接口类型

```
import 'vue-router'

declare module 'vue-router' {
  // 扩展 元信息类型
  interface RouteMeta {
    // 标题
    title?: string
  }
}
```

```
import NavBar from './NavBar.vue'
declare module 'vue' {
  interface GlobalComponents {
    VanNavBar: typeof NavBar;
  }
}
```

## ts为你的项目能带来什么？

- 良好的拼写检查和提示
  - 不同的数据有不同的方法
  - 响应式数据有提示
- 方便快捷的代码重构
  - 修改数据的类型，名称

```
pnpm type-check
```

可检查出所有有问题的文件，会有类型报错。

## Vue3组合式API如何添加类型？

回顾：

- defineProps 建议开启 响应式语法糖 vite.config.ts

```
const { money, car = 'xxx' } = defineProps<{
  money: number
  car?: string
}>()
```

- defineEmit

```
const emit = defineEmit<{  
  (e: 'changeChannel', id: number): void  

```

- ref

```
const count = ref(0)  
const list = ref<TodoItem[]>([])
```

- reactive

```
const book = reactive({ title: 'xxx' })  
  
// 推导不正确，需要加类型，变量后类型注解  
const book: { title: string, year?: number } = reactive({ title: 'xxx' })
```

- computed

```
const double = computed(()=>count.value*2)  
  
const money = computed<string>(()=>(count.value*2).toFixed(2))
```

## 优医问诊项目

极速问诊+问诊室+问诊记录，三个模块的功能一定要自己描述出来。

- 开发团队7人，2人后台，2人前端，1个产品，1设计，1测试。
  - 一个前端负责医生端和后台管理，你负责患者端。
- 患者H5端开发3个月，开发和测试。
- 上线，后台配置好了gitlab流水线，推dev分支是测试，推master分支是上线，回退版本是重新执行流水线。

## pnpm是什么？

- 一个磁盘利用率更高的包管理器

## 你使用什么创建的项目？

- 使用基于vite的create-vue脚手架

## 你如何实现主题定制？

- css变量，修改vant的主题css变量

## 状态持久化怎么实现的？

- 使用pinia的持久化插件，原理就是：同步到本地的localStorage，也可以指定存储方式

## 移动端适配怎么实现？

- 使用postcss插件px-to-viewport实现，原理是把px单位转换成vw单位

## 请求工具函数是什么？

- 基于axios二次封装的请求函数，方便使用和复用。

## 你如何实现响应数据类型？

- 通过 `axios.request<type1,type2>()` type1没有改resd的时候时间 type2改了响应数据的时候使用

## 你项目中封装过哪些组件？

- 自定义的通用业务组件，单选按钮组，svg图标组件
- 业务组件封装，支付抽屉
- 基于vant二次封装的组件，更多操作组件，通用导航

## 如何给全局组件添加类型？

- `GlobalComponents` 给全局组件加类型

## 测试接口的工具使用什么？

- Apifox 提供测试接口功能，需要登录的接口可以配置token

## 如何真机调试？

- 在同一个局域网，通过IP访问，使用eruda插件可以再手上看控制台，审查元素。



# 其他

<https://zhoushugang.gitee.io/patient-h5-note/project/#what>