1. vue中数据监听的方法。

当data中某个属性的值发生变化需要执行一些其它操作时，可以使用watch去监听这个属性，当属性值发生变化时，执行回调函数，处理相应的操作。

当监听的属性不是object对象时。

data: function() {

return {

num: 0,

};

}

watch: {

num: function(newV, oldV) {

console.log(‘num由'+oldV+’变成了’+newV);

},

}

当监听的是一个object对象时，object中属性值发生变化时不会触发回调函数，因为object指向的地址没有发生变化，所以监听对象发生变化时有以下三种方法

当监听的是user对象时。

data: function() {

return {

user: {

name: ‘zhang san’,

age: 23,

},

};

}

方法1：给user对象重新赋值，可以直接监听user对象

watch: {

user: function(newV, oldV) {

console.log('counter change to %o from %o', newV, oldV);

},

}

方法2：使用deep参数

使用deep需要使用watch的完整形式：handler是监听回调函数，deep: true指定了不仅仅监听user的变化，也监听其内部属性的变化，所以当user.name或user.age变化时都能触发handler回调。

watch: {

counter: {

handler: function(newV, oldV) {

console.log(newV);

},

deep: true,

}

}

比较方法1和方法2的优缺点

使用deep参数会为数据每一层都添加监听，当层级较深时比较耗费性能。所以一般来说使用重新赋值的方式是较优的方案，但如果只是想监听内部嵌套的某个属性的数据的话，重新赋值就比较重了，下面介绍监听对象中某个属性的方法。

方法3：监听object对象中的某个属性

watch: {

user.name: function(newV, oldV) {

console.log('counter.up change to %d from %d', newV, oldV);

},

user.age: function(newV, oldV) {

console.log('counter.down change to %d from %d', newV, oldV);

},

}

1. angular中数据监听的方法。

方法一: angular中监听数据变化可以使用钩子函数ngDoCheck来监听。

export class TestComponent implements DoCheck {

public name = 'zhangsan';

public oldName: string;

constructor() {

this.oldName = this.name;

}

ngDoCheck() {

if (this.name !== this.oldName) {

console.log('执行操作');

}

}

}

此钩子函数在发生脏值检测时就会被调用，触发脏值检测的条件有Dom事件、异步请求、Location变更事件等等，因此会有大量情况触发此钩子函数，很少的调用是由于数据修改而触发，因此可能产生性能问题。

方法二：angular弃用了angular1中的数据监听$watch方法，但官方引入了rxJS来解决数据监听的需求。

场景：在commonService中定义初始化语言’en’，当切换语言时，在组件中监听并执行对应操作。

在commonService中

import { BehaviorSubject } from 'rxjs';

export class CommonService {

// 定义language，设置初始语言'en'。

public language = new BehaviorSubject<string>('en');

constructor() { }

changeLanguage(lang) {

// 点击切换语言时，调用language对象的next方法。

this.language.next(lang);

}

}

在其它组件中监听language的变化，来执行相应操作。

在TestComponent 中

export class TestComponent {

public lang: string;

constructor(public commonService: CommonService) {

// 调用next()方法后，此处订阅获得变化后的值。

this.commonService.language.subscribe((lang) => {

this.lang = lang;

});

}

}