关于Vue的计算属性的认识 computed

概述

计算属性其实它的意思是说：你在用的时候，把它当做属性使用。

- 定义是用对象定义 + 函数，也就computed:{}
- 使用就直接用函数名即可，不需要括号

属性 --- 它的使用：{{函数}} 和你在data定义一个属性其他使用方式来说没有任何差异。

> 一句话：在*computed*定义是函数，使用属性

存在的原因是什么？

对你的开发过程中：存在数据二次加工的时候，你可以考虑使用：computed和watch

开发过程中，我们可能会对data中的属性进行加工处理。比如：

- 日期格式化
- 金额签名加 ￥
- 对性别为0 显示成女 1 男 2 保密
- 购物车的时候，要对购物车的结算做一个总计
- 等等

除了上面的方式，

- 也可以通过函数处理
  - vue的methods定义的函数
- filters机制。
  - 2.x支持
  - 在 3.x 中，过滤器已移除，且不再支持。取而代之的是，我们建议用方法调用或计算属性来替换它们。
- 通过computed解决，

computed的初始

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
```

```
8       <title>20、Vue计算属性computed.html</title>
9
10  </head>
11  <body>
12
13
14  <div id="app">
15      <h1>{{reverseMessage()}}</h1>
16      <h1>{{reveseMessageComputed}}</h1>
17
18  </div>
19
20  <script src="js/vue.global.js"></script>
21  <script>
22
23      function changeMessage(msg){
24          return msg.split("").reverse().join(" ")
25      }
26
27      var vm = Vue.createApp({
28          data() {
29             return {
30                 message:"Hello Vue!!!"
31             }
32          },
33          computed:{
34              reveseMessageComputed:function(){
35                  return
    this.message.split("").reverse().join(" ")
36              }
37          },
```

```
38          methods: {
39              reverseMessage(){
40                  return
    this.message.split("").reverse().join(" ")
41              }
42          }
43      }).mount("#app");
44  </script>
45  </body>
46  </html>
```

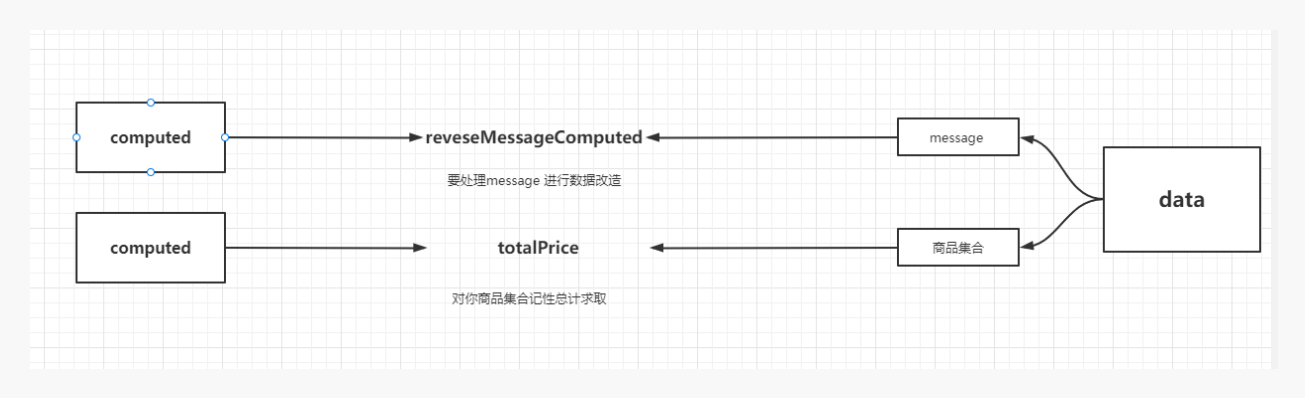
- 定义用函数，使用用属性

computed计算属性

前面已经对computed定义和使用部分已经明白，一句话：定义用函数，使用用属性。它的计算两字又如何理解呢？

所谓计算就指：如果computed的计算属性使用了data中的数据模型中的属性，默认中属性发生变化，computed监听计算属性会发生再次触发，再次渲染。

分析



- vue初始化以后，在vue整个执行之后，会自动触发一次computed，把里面定义的所有的计算属性会渲染和同步一次给视图。所有computed定义的计算属性，一定要有返回值return
- 如果任何data中的数据，发生变更的话，computed会再次触发，再次计算。

vue的methods定义的函数和computed的区别是什么？

- methods的函数的话是没有缓存，
- computed是有缓存的

```
<!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>Document</title>
</head>
<body>



<div id="app">
    <h1>{{reverseTitle()}}</h1>
    <h1>{{reverseTitle2}}</h1>
    <button @click="select">查询</button>
</div>

<script src="js/vue.global.js"></script>
<script>

    var app = Vue.createApp({
        data(){
            return {
                title:"Hi Vue3.x"
            }
        },

        computed: {
            reverseTitle2(){
                return this.title.split("").reverse().join(",");// 全局变量-----9874566
```

```
33              }
34          },
35
36          methods:{
37              reverseTitle(){
38                  return
this.title.split("").reverse().join(",");
39              },
40
41              select(){
42                  // 获取的时候，还会触发
reverseTitle2()行为吗？
43                  let msg1 = this.reverseTitle2;
44                  // 调用方法的reverseTitle还会触发
吗？
45                  let msg2 =
this.reverseTitle();
46                  console.log(msg1)
47                  console.log(msg2)
48              }
49          }
50      }).mount("#app");
51
52  </script>
53
54  </body>
55  </html>
56
```

案例分析 -购物车

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>19、Vue对数组的操作大全.html</title>
    <style>
        table {
            width: 100%;
            border-collapse: collapse
        }

        table tr td {
            border: 1px solid #ccc;
            padding: 10px;
        }

        .container {
            background: #eee;
            padding: 20px;
            margin-bottom: 10px;
```

```
        }
    </style>
</head>
<body>


<div id="app">
    <div class="container">
        <p>商品主键: <input type="text" v-model="product.id"></p>
        <p>商品标题: <input type="text" v-model="product.title"></p>
        <p>商品价格: <input type="text" v-model="product.price"></p>
        <p>商品数量: <input type="text" v-model="product.num"></p>
        <p>添加时间: <input type="text" v-model="product.addtime"></p>
        <p>
            <button @click="pushProduct">push添加商品</button>    
        </p>
    </div>
    <table>
        <thead>
        <tr>
            <td>index</td>
            <td><input type="checkbox"></td>
            <td>ID</td>
            <td>产品名称</td>
```

```
48              <td
    @click="sortMain('sortPrice','price')">产品价格
    {{sortPrice ? "升序" : "降序"}}</td>
49              <td
    @click="sortMain('sortNum','num')">产品的数量</td>
50              <td>小计</td>
51              <td>时间</td>
52              <td>操作</td>
53
54          </tr>
55          </thead>
56          <tbody>
57          <tr v-for="(product,index) in
    newProducts" :key="product.id">
58              <td>{{index}}</td>
59              <td><input type="checkbox"></td>
60              <td>{{product.id}}</td>
61              <td>{{product.title}}</td>
62              <td>{{product.pricestr}}</td>
63              <td><button @click="mius(index)">-
    </button> {{product.num}} <button
    @click="plus(index)">+</button></td>
64              <td>{{product.num * product.price}}
    </td>
65              <td>{{product.addtime}}</td>
66              <td><a href="javascript:void(0);">编
    辑</a> <a href="javascript:void(0);"
67
          @click="delProductByIndex(index)">删除</a>
    </td>
68          </tr>
```

```
        </tbody>
        <tfoot>
        <tr>
            <td colspan="10">总计：{{addSuffix(totalPrice)}}</td>
        </tr>
        </tfoot>
    </table>
</div>

<script src="js/vue.global.js"></script>
<script>
    /*
    * computed/methods函数 做数据二次改造
    * 需求：
    * - 对商品的日期进行格式化处理
    * - 对商品的金额增加人民币符号 ￥
    * - 对商品进行总计和小计
    * */

    var vm = Vue.createApp({
        data() {
            return {
                sortPrice: false,
                sortNum: false,
                product: {id: 0, title: "", price: 0, num: 1,addtime:"2012-11-12 12:12:12"},
                products: []
            }
        },

```

```
        created(){
            this.products = [
                {id: 1, title: "小米11", price: 125.8, num: 12,addtime:"2012-11-12 12:12:12"},
                {id: 2, title: "小米12", price: 25.8, num: 2,addtime:"2011-12-08 12:12:12"},
                {id: 3, title: "小米13", price: 15.8, num: 1,addtime:"2012-02-10 12:12:12"},
                {id: 4, title: "小米14", price: 5.8, num: 11,addtime:"2019-10-11 12:12:12"},
                {id: 5, title: "小米15", price: 15.8, num: 58,addtime:"2017-02-22 12:12:12"}
            ];
        },

        mounted(){

        },
        computed:{
            //
            totalPrice:function (){
                var total = 0;
                this.products.forEach(product => {
                    total += product.price * product.num
                })
                return total;
            },
            newProducts:function(){
                this.products.map(product=>{
```

```
                    product.pricestr = "￥"+product.price;
                    product.addtime = this.dateFmt(product.addtime,"yyyy-MM-dd")
                    return product;
                });
                return this.products;
            }
        },
        methods: {
            addSuffix(value){
              return "￥"+value;
            },

            dateFmt(dateStr,fmt){
                var date = new Date(dateStr.replace(/-/ig,"/"));
                var o = {
                    "M+" : date.getMonth()+1,  //月份
                    "d+" : date.getDate(),  //日
                    "h+" : date.getHours(),  //小时
                    "m+" : date.getMinutes(),  //分
                    "s+" : date.getSeconds(),  //秒
                    "q+" : Math.floor((date.getMonth()+3)/3), //季度
```

```
143                     "S"  :
    date.getMilliseconds()             //毫秒
144                 };
145                 if(/(y+)/.test(fmt))
146                     fmt=fmt.replace(RegExp.$1,
    (date.getFullYear()+"").substr(4 -
    RegExp.$1.length));
147                 for(var k in o)
148                     if(new RegExp("("+ k
    +")").test(fmt))
149                         fmt =
    fmt.replace(RegExp.$1, (RegExp.$1.length==1) ?
    (o[k]) : (("00"+ o[k]).substr(""+
    o[k]).length)));
150                 return fmt;
151             },
152
153             // 总计
154             countTotalPrice(){
155                 var total = 0;
156                 this.products.forEach(product =>
    {
157                     total += product.price *
    product.num
158                 })
159             },
160
161             // 删除元素
162             delProductByIndex(index) {
163                 this.products.splice(index, 1);
164             },
```

```
165
166             // push添加商品
167             pushProduct() {
168 
    this.products.push(this.product);
169             },
170 
171             plus(index){
172                 this.products[index].num++;
173             },
174 
175             mius(index){
176                 this.products[index].num--;
177             }
178         }
179     }).mount("#app");
180 </script>
181 </body>
182 </html>
```

总结

- 通过上面的一系列的案例的分析，其实computed和methods方法处理，都可以完成数据改造。
- 只不过methods方法用在插值表达式或者指令上去进行调用。当然这个也是最容易理解的方式。
- 而computed不需要，把属性定义和处理做了融合处理，使用起来也非常的方便

- methods毕竟是函数，每次执行和获取都会触发，而computed它一定是关联属性发生变化它才会触发
- 对数据的二次加工和二次改造。

computed和**watch**的区别

- computed --- 1 : N(data的属性)
- watch --- 1(data的属性):1(data的属性)

两者没有任何的关联性，computed底层其实就watch，watch是多个和它有关的属性。

而watch只监听自身的变化。而computed是有缓存机制的。

初始化层面：

- watch 不会触发
    - 如果watch增加了属性的immediate:true，其实和computed监听单属性其实一个意思。
- computed会触发

改变监听属性的时候

- watch会先触发，会执行函数触发
- computed会后触发，会执行函数触发

缓存的问题

两者都会进行数据的缓存，如果数据不会发生变更，是不会再次触发。

两者区别

- watch只能监听某个属性的变化、1:1 自身属性computed都会执行。
    - 是指，watch如果监听的某个属性的时候，也调用别人属性的使用，这个调用的属性发生变化，是不会去触发属性的监听函数。千万不要误解成为watch只能监听一个属性哦。

```
1    watch:{
2        title:function(){
3            this.num++;// 这个num的后面的改变是不会去触发title的监听函数
4        },
5        user:function(){
6
7         }
8    },
```

- computed的行为是可以监听，多个属性的变化。1:N ，如果N任何一个改变computed都会执行。起到更名的效果。

布置一个作业：在地铁上，或者晚上睡觉之前查看一下。有时间就证明和练习。

https://v3.cn.vuejs.org/style-guide/#%E8%A7%84%E5%88%99%E7%B1%BB%E5%88%AB