1.组件导入外部css文件

<style>

@import './assets/css/init.css';

</style>

2.background-position的x,y坐标：

看成一个框在一个图片上移动，上加下减，左加右减

3.去除input聚焦后的边框：

input:focus{

border: 0;

outline: none;

}

4.vue没有hover事件，只能分成mouseenter，mouseleave

5.float和position可以同时使用

6.在vue中可以通过给标签加ref属性，就可以在js中利用ref去引用它，

从而操作该dom元素，只要想要在Vue中直接操作DOM元素，就必须用ref属性进行注册

例子: 注册时**<p ref="p">hello</p>** 获取时**this.$ref.p**

7.js判断对象是否为空对象的几种方法

**JSON.stringify(obj1) == "{}"**

**Object.keys(obj1).length==0**

8.vue 组建里给img标签的src属性设置v-bond的时候，url不能用原先的

"../assets/img/bunner/bunner1.png"应该用**require("../assets/img/bunner/bunner1.png")**

require("../assets/img/bunner/bunner(1).jpg")只能在created阶段或者在组建的data里使用

9.vue里监听事件页面滚动:

mounted: function( ){

window.addEventListener('scroll', this.handleScroll);

//首先在组建挂载时创建监听事件

}，

methods: {

handleScroll () {

let scrollY = window.pageYOffset || document.documentElement.scrollTop;

//获取页面超出窗口的高度

if (scrollY > 100) {dosometing... }

else {dosometing... }

},

}，

beforeDestroy(){

//一定要在这步不然跳转其他页面还在监听会找不到methods绑定的dom

window.removeEventListener('scroll', this.topbarShow);

},

10.**event.target**返回触发事件的元素

**event.currentTarget**返回绑定事件的元素

11.在请求执行成功后执行回调函数(res)中的内容，

回调函数(res)处于其它函数的内部this不会与任何对象绑定，为undefined。

12.不使用事件监听的情况下，想启用某个函数(函数中需要得到动态数据)，一般放在updated钩子函数里

13.vue路由跳转 定时器 没有清掉 ？给组件添加销毁前beforeDestroy的钩子函数，在组件销毁前关闭掉定时器

14.router传参用params必须在router.js的path中申明参数如:

**{path:'/plist/:pid/:pno',component:Plist}**

**<router-link to="/plist/101/6">**

or

**this.$router.push("/plist/101/6");**

**this.$route.params----> { pid:'101', pid:'6'}**

如果用query则不用，因为会直接显示在URL上，如

**this.$router.push({path:'/testDemo',query:{setid:123456}});**

**this.$route.query.setid** 接收

15.HTML5新增class修改方法：

**el.classList.add("类名");**

**el.classList.remove("类名");**

**el.classList.toggle("类名"); 切换**

16.Js 数据去掉小数点后边的00000，保留非零位数

a=parseFloat(a)

php 去除小数点后面的0，保留非零

floatval($arg)

17.vue组建中异步得到的数据绑定不到dom上--------无报错

原因：？？？？？？？？

元代码：

for(var i=8;i<58;i++){

this.products.[i-8]=res.data[i];

this.products[i-8].pic=require("@/"+res.data[i].pic);

}

改进后代码：

for(var i=8;i<58;i++){

this.products.push(res.data[i]);

this.products[i-8].pic=require("@/"+res.data[i].pic);

}

18.获取鼠标相对元素的坐标：@mousemove="move($event)"

methods:{

move(e){

console.log(e.offsetY+"\_"+e.offsetX);

}

//注意不能用e.target.offsetY/X

}

19.路径里总是出现'#'，原因vue-router处于hash模式，改为h5 的history模式就行

export default new Router({

mode:'history',

routes: [

{path:'/',component:Main}

]

})

20.使用vuex管理登录状态：

首先要安装vuex：npm cnpm install vuex –S

在main.js中引入：

|  |
| --- |
| import Vue from 'vue'  import App from './App'  import Vuex from 'vuex'  Vue.use(Vuex)  const store = new Vuex.Store({  state: {相当于data，但无法直接赋值修改需要mutation里修改}，  getters: {相当于组建里的computed计算属性},  mutations:{专用于修改state里的数据的方法},  actions: {同mutations，但是这个可以有异步操作，mutations不行}  })  new Vue({  el: '#app',  store,  render: h => h(App)  }) |

使用的时候，在组建里：

|  |
| --- |
| <template>  <footer class="footer">  <p>  Copyright&nbsp;&copy;&nbsp;**{{author}}** - 2016 All rights reserved  </p>  </footer>  </template>  <script>  export default {  name: 'footerDiv',  data () {  return {}  },  **computed: {**  **author () {**  **return this.$store.state.author**  **}**  **}，**  **methods:{**  **add(){**  **this.$store.commit(“mutations方法名”,value);**  **}**  **}**  }  </script> |