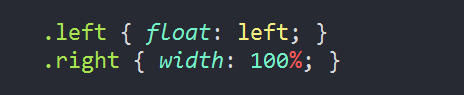
### HTML+ JavaScript

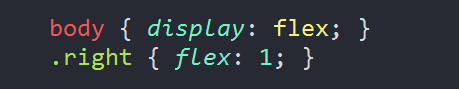
#### 页面布局：

##### 左右布局左边固定，右边自适应

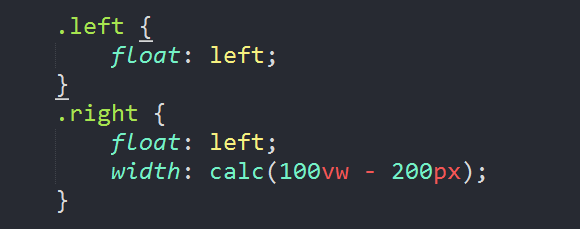
方法一：左边设置左浮动，右边宽度设置100%



方法二：父容器设置 **display：flex；**Right部分设置**flex：1**



方法三：**设置浮动** + 在 css 中使用 **calc()** 函数



1. 浮动。（注意：为了不影响其他元素，别忘了在父级上清除浮动）

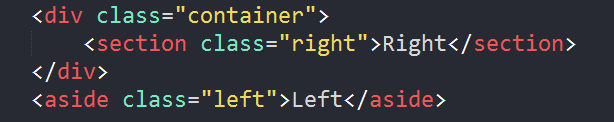
2. **calc() = calc(四则运算)**用于在 css 中动态计算长度值，需要注意的是，运算符前后都需要保留一个空格，例如：width: calc(100% - 10px)；

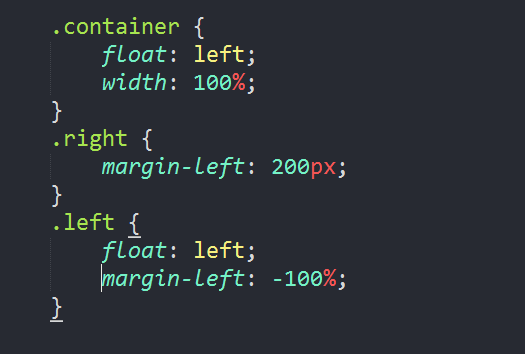
3. **vw**：viewport width。1vw = viewport 宽度的 1%, 100vw = viewport width,

同样的还有 vh： viewport height。1vw = viewport 高度的 1%， 100vh = viewport height。

方法四：**使用负margin**

首先修改页面结构，为自适应部分添加容器 .container, 同时改变左右部分的位置，如下：





### 在data里面使用this继承this指向、

1. 如果代码写在一个文件里面直接使用
2. 如果不在一个文件里面要通过函数接收传递参数的方式

但是接收的时候不能使用this关键字需要用一个别的命名来接收

### 动态添加样式：

<span :class="{'view\_details\_value':oldRecordsDetails.ratio ===newRecordsDetails.ratio,'diff\_color':oldRecordsDetails.ratio !==newRecordsDetails.ratio}">{{newRecordsDetails.ratio}}</span>

{}里面判断条件为真时使用该类名

三元运算符

<li v-for="(item, index) in roomList" :key="index" :class="item.room\_status === 1 ? ' rentedClass' : ''">

### vue项目中数据{\_\_ob\_\_: observer}无法取值

**ob**: Observer是 Vue 对数据监控添加的属性

需要转化为普通对象

解法1：

let resdata=JSON.parse(JSON.stringify(this.checkedproject))

解法2：

let dataList=[]

*this*.tableData.forEach(*v*=>{

         v.info.infoTime= *this*.timestampToTime(v.info.infoTime);

           dataList.push(v.info)

          })

vue版

### 时间戳转化为时间格式

export function formatDate (date, fmt) {

if (/(y+)/.test(fmt)) {

fmt = fmt.replace(RegExp.$1, (date.getFullYear() + '').substr(4 - RegExp.$1.length));

}

let o = {

'M+': date.getMonth() + 1,

'd+': date.getDate(),

'h+': date.getHours(),

'm+': date.getMinutes(),

's+': date.getSeconds()

};

for (let k in o) {

if (new RegExp(`(${k})`).test(fmt)) {

let str = o[k] + '';

fmt = fmt.replace(RegExp.$1, (RegExp.$1.length === 1) ? str : padLeftZero(str));

}

}

return fmt;

};

function padLeftZero (str) {

return ('00' + str).substr(str.length);

使用

import {formatDate} from '../../date.js';

formatDate(new Date(time \* 1000), 'yyyy-MM-dd hh:mm');

};

注意：小时制HH：24小时 hh：12小时

js版

timestampToTime(*timestamp*) {

        var date = **new** *Date*(timestamp);//时间戳为10位需\*1000，时间戳为13位的话不需乘1000

        var Y = date.getFullYear() + '-';

        var M = (date.getMonth()+1 < 10 ? '0'+(date.getMonth()+1) : date.getMonth()+1) + '-';

        var D = date.getDate() + ' ';

        // var h = date.getHours() + ':';

        // var m = date.getMinutes() + ':';

        // var s = date.getSeconds();

        return Y+M+D;

    },

*this*.moment(nowTime).format('YYYY-MM-DD 00:00:00')

加载效果的函数

 getDatated(){

        const loading = *this*.$loading({

                lock: true,//lock的修改符--默认是false

                text: 'Loading',//显示在加载图标下方的加载文案

                spinner: 'el-icon-loading',//自定义加载图标类名

                background: 'rgba(0, 0, 0, 0.7)',//遮罩层颜色

                target: document.querySelector('.footer-right')//loadin覆盖的dom元素节点

        });

        //成功回调函数停止加载

        setTimeout(() => {

              loading.close();

        }, 90);

    },

过滤器保留两位小数

//过滤器保留两位小数

  filters: {

  numFilter (*value*) {

    // 截取当前数据到小数点后三位

    let tempVal = parseFloat(value).toFixed(3)

    let realVal = tempVal.substring(0, tempVal.length - 1)

    return realVal

  }

},

## el-tab拼接

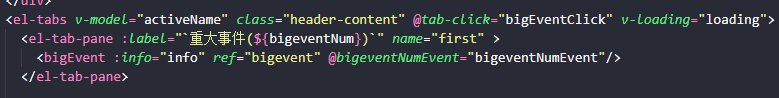
<el-table-column *prop*="contract\_start,'-',contract\_end" *align*="center" >

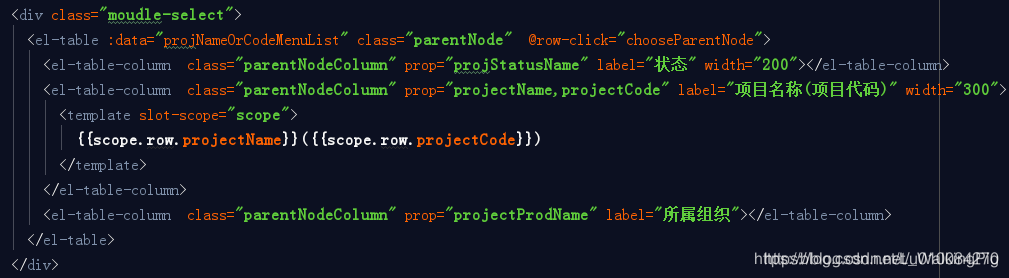
        <template *slot-scope*="scope">

          {{scope.row.contract\_start}}-{{scope.row.contract\_end}}

        </template>

      </el-table-column>



el-table自适应

### Slot-scope 插槽使用

<https://segmentfault.com/a/1190000015884505>

## vue+el-table 根据屏幕大小动态设置max-height属性适合适配多个表格高度出现滚动条

<el-table ref="table" :data="info" :span-method="colspanMethod" :max-height="tableHeight">

mounted:function(){

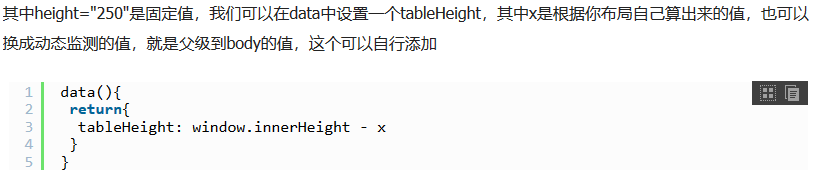
this.tableHeight = window.innerHeight - this.$refs.table.$el.offsetTop - 50;

    //window.innerHeight:浏览器的可用高度

    //this.$refs.table.$el.offsetTop：表格距离浏览器的高度

    //后面的50：根据需求空出的高度，自行调整

}



Element 组件样式修改

先找到要修改层级的类（组件自封装类）

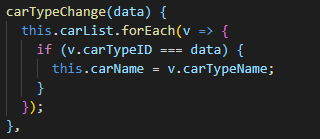
然后/deep/

 /deep/ *.el-input\_\_inner*{

          width: 0.88rem;

        }

选择器的label值如何拿去



## 时间日期选择器

当要禁用当前之前的信息时候：

---vue template里面

<el-date-picker

v-model="value"

type="date"

:editable="false"

placeholder="日期筛选"

value-format="yyyy-MM-dd"

:picker-options="pickerOptions0">

</el-date-picker>

---data里面

pickerOptions0: {

disabledDate(time) {

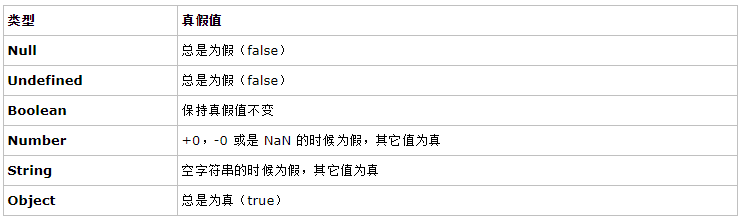
return time.getTime() < Date.now() - 8.64e7; //去掉 8.64e7时禁用当天

}

}

## 数据类型与判断

js中有6个值为false，分别是: 0, '', null, undefined, NaN 和 false



### Js比较规则

* 如果比较的两者中有bool，会把 bool 先转换为对应的 number，即 0 和 1
* 如果比较的双方中有一方为number一方为string，会把string转换为数字
* 把string直接转换为bool的时候，空字符串‘'转换为 false，除此外的一切字符串转换为 true

### 相等性比较

在转换不同的数据类型时，相等和不相等操作符遵循下列基本规则：

1. 如果有一个操作数是布尔值，则在比较相等性之前先将其转换为数值——false转换为0，而true转换为1；

2. 如果一个操作数是字符串，另一个操作数是数值，在比较相等性之前先将字符串转换为数值；

3. 如果一个操作数是对象，另一个操作数不是，则调用对象的valueOf()方法，用得到的基本类型值按照前面的规则进行比较；

### 步骤

首先 'true' == true 符合规则 1，这样就转化成了对 'true' == 1 进行求值

此时表达式符合规则 2

也就是说需要将 'true' 转换成数值之后，再与数字 1 进行比较。把 'true' 转换成数值是使用 Number 函数

[] !==[] ---🡪返回结果为true

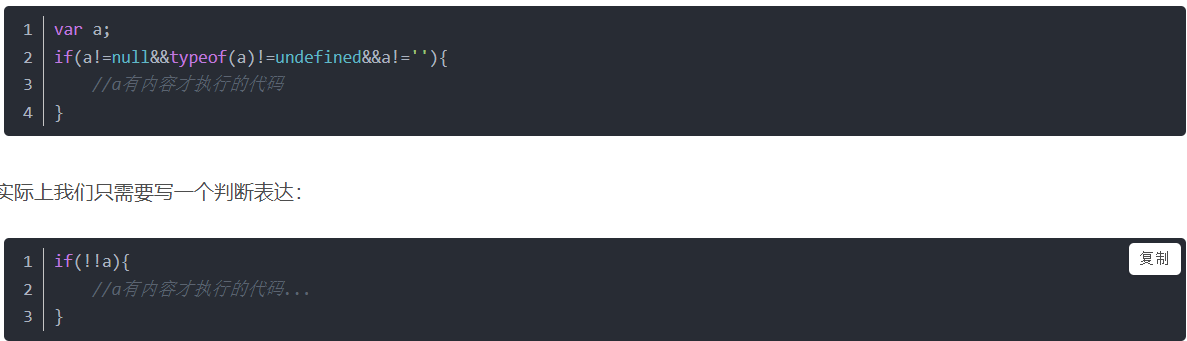
1. 数组是object，用于判断条件时就会被转化为true
2. 任意值与布尔值比较，都会将两边的值转化为Number。如arr与false比较，false转化为0，而arr为空数组，也转化为0：
3. 当空数组作为判断条件时，相当于true。当空数组与布尔值直接比较时，相当于false

## !和!! 详解

1、！可将变量转换成boolean类型，null、undefined和空字符串取反都为true，其余都为false。

2、！！常常用来做类型判断，在第一步!（变量）之后再做逻辑取反运算，在js中新手常常会写这样臃肿的代码：

判断变量a为非空，未定义或者非空串才能执行方法体的内容。



### Undefined比较

Undefined与Undefined无法进行比较

可以：

1. 直接用Undefined做条件
2. 转为字符串然后比较（toString（）转化为字符串）
3. Type of 判断之后比较

注意如果引用的值是Undefined如果继续延伸拿取值的话就会报错阻止代码运行

## 时间对比

### 时间类比较

startTime= new Date(Date.parse(starttime));

endTime=new Date(Date.parse(endTime));

//进行比较

startTime>endTime

### 时间戳比较

//时间戳比较

startTime=Date.parse(starttime);

endTime=Date.parse(endTime);

//进行比较

startTime>endTime

#### 将日期转化为时间戳

var date= new Date();

var time1 = date.getTime();

var time2 = date.valueOf();

var time3 = Date.parse(Date());

console.log(time1);//1528258266412

console.log(time2);//1528258366197

console.log(time3);//1528257868000

### 时间加减运算

time=new Date();

time=time.getFullYear()+"/"+parseInt(time.getMonth()+1)+"/"+time.getDate()+" "+(time.getHours()+2)+":"+time.getMinutes()+":"+time.getSeconds();//给小时加2

## el-form表单验证

this.$refs[ruleForm].validate()

如果DOM还么加载出来的话使用

this.$nextTick(function() {

this.$refs.ticketInfoForm.resetFields();

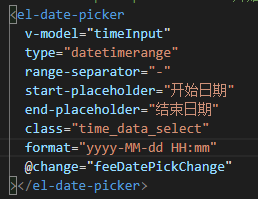
})

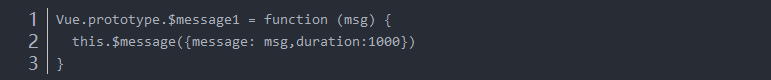
如果使用的是两个或者以上的表单ref名字不能相同不然后一个只会识别一个表单校验规则

有可能this.$refs[ruleForm].validate() 方式不识别。需要使用: this.$refs.ruleForm.validate()

需要通过一个变量开启别的变量的时候

## 时间日期选择器





文本强制显示一行 white-space:nowrap;

## es6对象、数组

### 对象

#### 对象连接Object.assign()

用于对象的合并，如果目标对象与源对象有同名属性，或多个源对象有同名属性，则后面的属性会覆盖前面的属性。

注意点：

Object.assign 方法实行的是浅拷贝

#### 对象遍历

（1）for...in

for...in循环遍历对象自身的和继承的可枚举属性（不含Symbol属性）。

（2）Object.keys(obj)

Object.keys返回一个数组，包括对象自身的（不含继承的）所有可枚举属性（不含Symbol属性）。

（3）Object.getOwnPropertyNames(obj)

Object.getOwnPropertyNames返回一个数组，包含对象自身的所有属性（不含Symbol属性，但是包括不可枚举属性）。

（4）Object.getOwnPropertySymbols(obj)

Object.getOwnPropertySymbols返回一个数组，包含对象自身的所有Symbol属性。

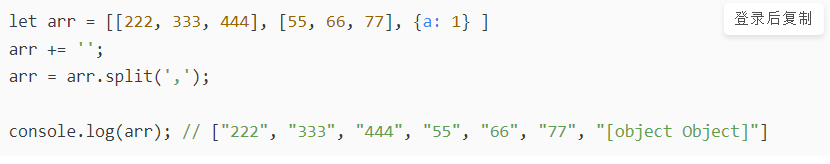
（5）Reflect.ownKeys(obj)

Reflect.ownKeys返回一个数组，包含对象自身的所有属性，不管是属性名是Symbol或字符串，也不管是否可枚举。

### 数组

#### 数组降维

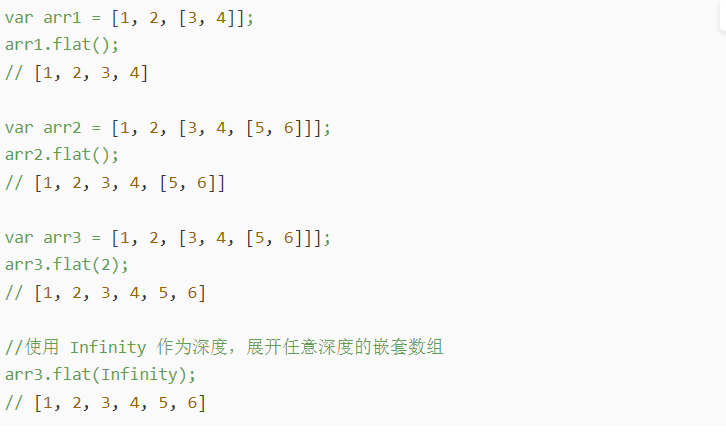
1. 数组字符串话



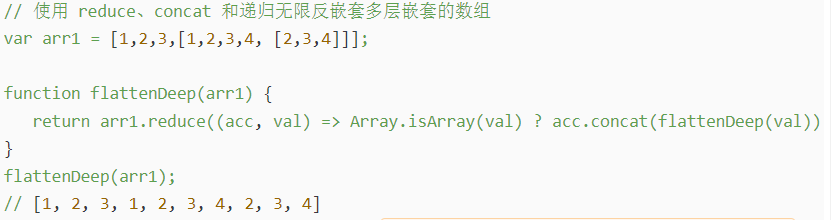
1. 递归



1. Array.prototype.flat()



1. 使用扩展运算符
2. Reduce+concat+递归



#### 数组排序

**按照数值大小进行排序-升序**

[1, 8, 5].sort((a, b) => {

return a-b; // 从小到大排序

});

// [1, 5, 8]

**按照数值大小进行排序-降序**

[1, 8, 5].sort((a, b) => {

return b-a; // 从大到小排序

});

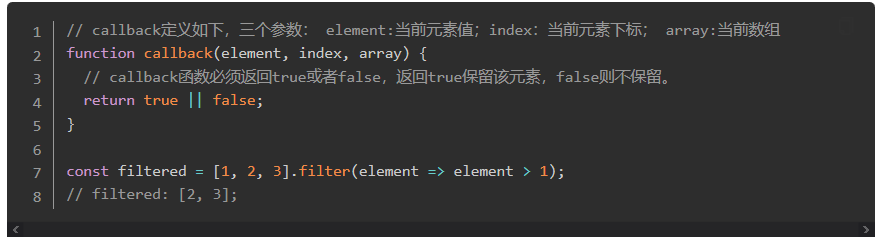
// [8, 5, 1]

#### 数组遍历

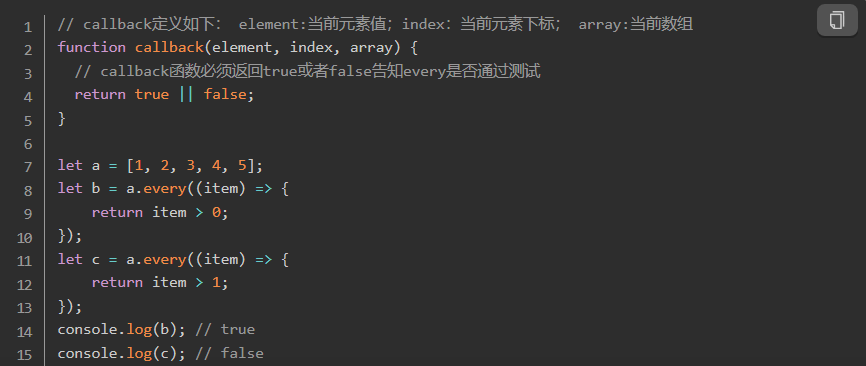
1. array.filter(callback, thisArg)方法使用指定的函数测试所有元素,并创建一个包含所有通过测试的元素的新数组。

参数说明：

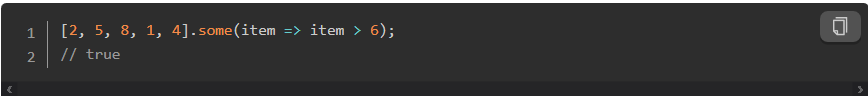
* callback 用来测试数组的每个元素的函数，返回true表示保留该元素（通过测试），false则不保留。
* thisArg 可选。执行 callback 时的用于 this 的值。



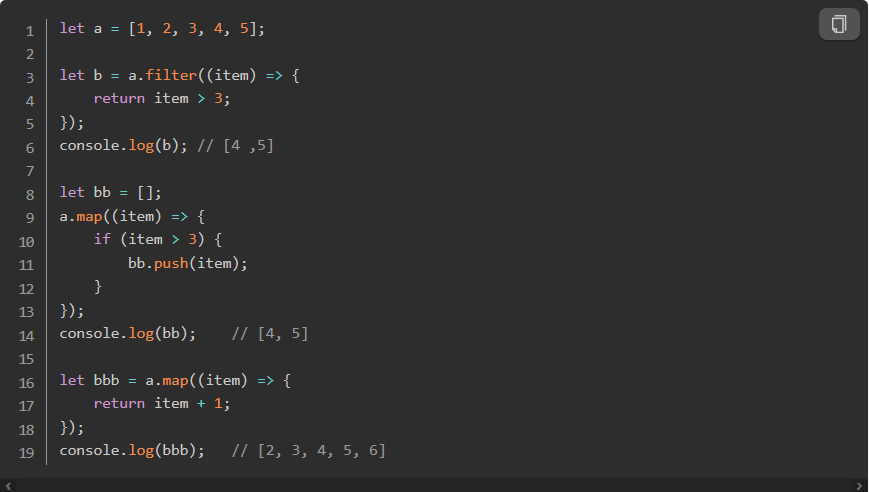
2、array.every(callback[, thisArg])方法检测数组中的每一个元素是否都通过了callback测试，全部通过返回true，否则返回false



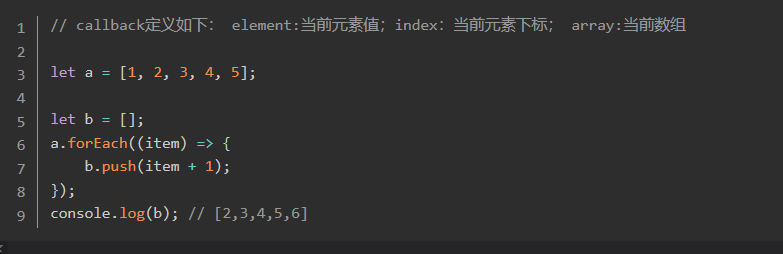
3、array.some(callback[, thisArg])判断数组中是否包含可以通过callback测试的元素，与every不同的是，这里只要某一个元素通过测试，即返回true。callback定义同上。



1. array.map(callback[, thisArg])方法返回一个由原数组中的每个元素调用callback函数后的返回值组成的新数组。



1. array.forEach(callbak)为数组的每个元素执行对应的方法



更多详情

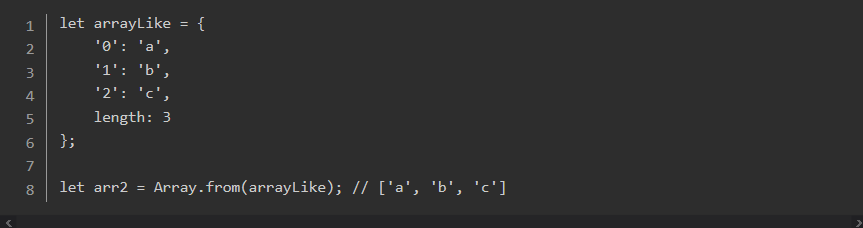
<https://www.jianshu.com/p/e1b43e56de08>

#### 转数组

1、Array.from()方法

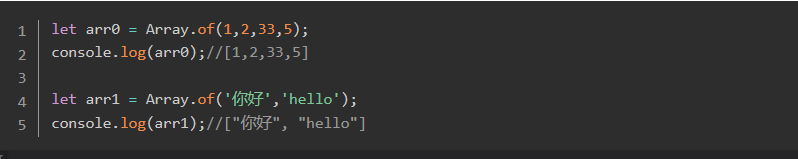
Array.from()方法是用于将类似数组的对象（即有length属性的对象）和可遍历对象转为真正的数组。

比如，使用·Array.from()·方法，可以轻松将·JSON·数组格式转为数组。



2、Array.of()方法

Array.of()方法是将一组值转变为数组。



### 数字

在 JavaScript 中有三个特殊的值，它们虽然是数字，但看起来却跟一般的数字不太一样。

前两个是Infinity和-Infinity，它们代表正无穷和负无穷。 “无穷减一”仍然是“无穷”，依此类推。 尽管如此，不要过分信任基于无穷大的计算。 它在数学上不合理，并且很快导致我们的下一个特殊数字：NaN

NaN代表“不是数字”，即使它是数字类型的值。 例如，当你尝试计算0/0（零除零），Infinity - Infinity或任何其他数字操作，它不会产生有意义的结果时，你将得到此结果。

注意：

在 JavaScript 中，只有一个值不等于其自身，那就是NaN（Not a Number，非数值）

NaN用于表示非法运算的结果，正因如此，不同的非法运算结果也不会相等。

## 冒泡造成的问题

自己写了一个抽屉的侧拉框，给body添加了点击事件触发到全页面，（点击页面任意位置抽屉回拉），然后使用element的对话框组件el-dialog组件，但是问题来了点击关闭弹框的时候抽屉也回拉了，后来发现当点击关闭弹框的时候事件冒泡到父级的body上面了

解决办法就是给el-dialog组件外层包裹一层div就解决了

### 阻止事件冒泡

* 阻止事件冒泡： event.stopPropagation() 或 event.cancelBubble = true (IE)
* 阻止浏览器默认行为： event.preventDefault()
* @click.stop : 阻止事件冒泡
* @click.prevent : 阻止事件默认行为
* @click.self : 事件只作用在元素本身，而不是其子元素

 @click.capture : 使用事件捕获模式

 @click.once : 事件只触发一次 2.1.4新增

 @scroll.passive : 告诉浏览器这个事件的默认行为不会被取消 2.3.0新增

 @keyup.enter : 按键修饰符，enter被按下的时候触发

 @click.ctrl : 系统修饰符，Ctrl 被按下的时候触发 2.1.0新增

 @click.ctrl.exact : 有且只有 Ctrl 被按下的时候才触发 2.5.0新增

### 滚动条

给子组件是el-tree 父组件在el-tabs里面调用子组件添加滚动条

1. 在el-tree的父容器里面给定固定高度 以及overflow：scroll
2. 在el-tree的子节点写内敛样式width：max-content

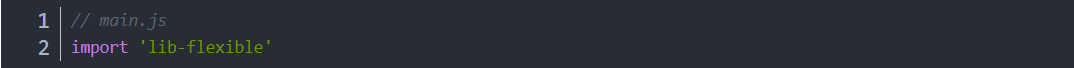
## 屏幕适配

进入项目目录中 (cd 项目名) 安装 lib-flexible 和 px2rem-loader

npm i lib-flexible -S

npm i px2rem-loader -D

安装好了之后还需要在项目的入口文件 main.js 里引入 lib-flexible



## Vue组件类型问题

强制刷新

this.$forceUpdate()

## 项目总结IFM：

### 项目遇到问题描述

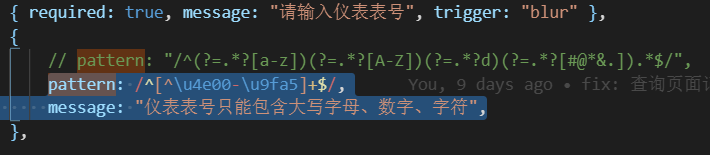
1. 项目复杂程度倾向于复杂的逻辑交互
2. 频繁的调取接口
3. 表单校验规则的管理
4. 针对想功能、权限管理
5. Element 组件的应用编辑
6. 父子组件的通信及处理数据

### 提供经验（以后开发需要注意的方向）

1. 开发项目前需要先明确自己的任务和开发方向

细节交互很重要、需要特别关注

1. UI设计稿的还原度关系到验收，在界面布局和画面既视感方面需要完全遵循UI设计稿、功能类需要遵循原型（交互）设计稿
2. 不管项目初期调用接口的数量多少，先自己封装一个接口文件存放接口、实现接口的统一管理
3. 表单校验



正则校验需要放在／／里面



Validator尽量声明一个常量放在return外面也可以封装一个规则组件方便校验规则的统一管理

功能性触发校验

this.$refs.addMeterForm.validate((valid ) =>｛｝)

记得主动取消校验记录

this.$refs.［ｒｅｆ绑定值］．resetFields();

６、父子组件通信时，尽量

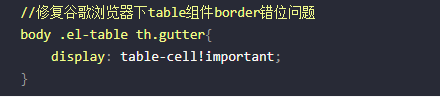
将数据处理放在父组件完成然后传给子组件这样会减少很多不必要的问题

## 浏览器兼容问题

### IE 浏览器不兼容es6 语法

### Chrome浏览器下ElementUI的table组件错误问题

处理方法：



## Vue3.0之坑

由于vue3.0在插件install函数的入参从vue原型（类）改成了app（vue实例）导致element-ui中的Vue.prototype.\* 这样的代码已经全部失效了

1. 各种UI框架不兼容
2. Axios不兼容（目前可以通过sdn的方式单页面引用）
3. 函数分离将所要写的代码包在一个函数里然后return最后在setup里面输出
4. <script>
5. import axios from 'axios'
6. import { shallowReactive } from 'vue'
7. export default {
8. name: "Show",
9. setup() {
10. let {state} = recommendData()
11. return {
12. state
13. }
14. }
15. }
16. function recommendData() {
17. let state = shallowReactive({
18. recommList:[]
19. })
20. axios.post('http://www.51houniao.com/product/product/getProductRecommendUser',{orderBy: "TOP", desc: false, pageNow: 1, pageSize: 600}).then(res => {
21. console.log(res,'数据')
22. state.recommList = res.data
23. }).catch((error => {
24. console.log('error:',error)
25. }))
26. return {state}
27. }
28. </script>

## Vue读取Excel文件内容

### input通常方式

安装依赖

import XLSX from 'xlsx'

<input type="file" ref="upload" accept=".xls,.xlsx" class="outputlist\_upload">

Data中声明变量

data() {

return {

outputs: []

}

},

mounted中绑定事件

mounted() {

this.$refs.upload.addEventListener('change', e => {//绑定监听表格导入事件

this.readExcel(e);

})

},

readExcel(e) {//表格导入

var that = this;

const files = e.target.files;

console.log(files);

if(files.length<=0){//如果没有文件名

return false;

}else if(!/\.(xls|xlsx)$/.test(files[0].name.toLowerCase())){

this.$Message.error('上传格式不正确，请上传xls或者xlsx格式');

return false;

}

const fileReader = new FileReader();

fileReader.onload = (ev) => {

try {

const data = ev.target.result;

const workbook = XLSX.read(data, {

type: 'binary'

});

const wsname = workbook.SheetNames[0];//取第一张表

const ws = XLSX.utils.sheet\_to\_json(workbook.Sheets[wsname]);//生成json表格内容

console.log(ws);

that.outputs = [];//清空接收数据

//编辑数据

for(var i= 0;i<ws.length;i++){

var sheetData = {

address: ws[i].addr,

value: ws[i].value

}

that.outputs.push(sheetData);

}

this.$refs.upload.value = '';

} catch (e) {

return false;

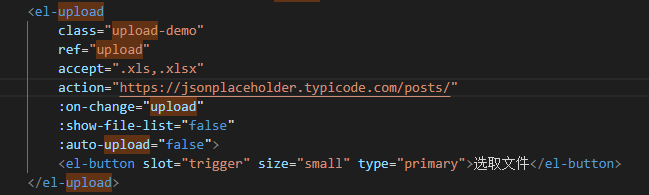
}

};

fileReader.readAsBinaryString(files[0]);

}

### 结合elementUI的upload



upload(file,fileList){

console.log("file",file);

console.log("fileList",fileList);

let files = {0:file.raw}

this.readExcel1(files);

}

readExcel1主函数

readExcel1(files) {//表格导入

var that = this;

console.log(files);

if(files.length<=0){//如果没有文件名

return false;

}else if(!/\.(xls|xlsx)$/.test(files[0].name.toLowerCase())){

this.$Message.error('上传格式不正确，请上传xls或者xlsx格式');

return false;

}

const fileReader = new FileReader();

fileReader.onload = (ev) => {

try {

const data = ev.target.result;

const workbook = XLSX.read(data, {

type: 'binary'

});

const wsname = workbook.SheetNames[0];//取第一张表

const ws = XLSX.utils.sheet\_to\_json(workbook.Sheets[wsname]);//生成json表格内容

console.log(ws);

// that.peopleArr = [];//清空接收数据

// if(that.peopleArr.length == 1 && that.peopleArr[0].roleName == "" && that.peopleArr[0].enLine == ""){

// that.peopleArr = [];

// }

//重写数据

try{

}catch(err){

console.log(err)

}

this.$refs.upload.value = '';

} catch (e) {

return false;

}

};

fileReader.readAsBinaryString(files[0]);

},

## **Github 相关问题总结**

### Github 登录不进去

C:\Windows\System32\drivers\etc

Hosts 文件最后位置

增加github 的映射就行了

例如：

192.30.255.112  github.com git   
185.31.16.184 github.global.ssl.fastly.net

或者

140.82.113.3 github.com

199.232.5.194 github.global.ssl.fastly.net

54.231.114.219 github-cloud.s3.amazonaws.com

