## JS 代码片段

for循环还有一个特别之处，就是循环语句部分是一个父作用域，而循环体内部是一个单独的子作用域。

**for** (**let** i = 0; i < 3; i++) {

**let** i = 'abc';

console.log(i);

}*// abc// abc// abc*

只要块级作用域内存在let命令，它所声明的变量就“绑定”（binding）这个区域，不再受外部的影响。

**var** tmp = 123;

**if** (true) {

tmp = 'abc'; *// ReferenceError*

**let** tmp;

}

ES6明确规定，如果区块中存在let和const命令，这个区块对这些命令声明的变量，从一开始就形成了封闭作用域。凡是在声明之前就使用这些变量，就会报错。

总之，在代码块内，使用let命令声明变量之前，该变量都是不可用的。这在语法上，称为“暂时性死区”（temporal dead zone，简称 TDZ）。

**typeof** x; *// ReferenceError*

**let** x;

****交换变量的值****

**let** x = 1;**let** y = 2;

[x, y] = [y, x];

### [计算数组中每个元素出现的次数](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce" \l "%E8%AE%A1%E7%AE%97%E6%95%B0%E7%BB%84%E4%B8%AD%E6%AF%8F%E4%B8%AA%E5%85%83%E7%B4%A0%E5%87%BA%E7%8E%B0%E7%9A%84%E6%AC%A1%E6%95%B0" \o "Permalink to 计算数组中每个元素出现的次数)

var names = ['Alice', 'Bob', 'Tiff', 'Bruce', 'Alice'];

var countedNames = names.reduce(function (allNames, name) {

if (name in allNames) { allNames[name]++; } else {allNames[name] = 1; }

return allNames;}, {});// countedNames is:// { 'Alice': 2, 'Bob': 1, 'Tiff': 1, 'Bruce': 1 }

类似数组的对象，将它转为真正的数组。

**let** arrayLike = { '0': 'a', '1': 'b', '2': 'c', length: 3 };

*// ES5的写法***var** arr1 = [].slice.call(arrayLike); *// ['a', 'b', 'c']*

*// ES6的写法***let** arr2 = Array.from(arrayLike); *// ['a', 'b', 'c']*

[...arrayLike]

# **vue事件方法之$on方法的实现原理**

$on也是采用了经典的发布订阅者设计模式，首先定义一个事件中心，通过$on订阅事件，将事件存储在事件中心里面，然后通过$emit触发事件中心里面存储的订阅事件。是采用了经典的发布订阅者设计模式，首先定义一个事件中心，通过$on订阅事件，将事件存储在事件中心里面，然后通过$emit触发事件中心里面存储的订阅事件。

Vue.prototype.$on = function (event, fn) {

const vm: Component = this

if (Array.isArray(event)) {

for (let i = 0, l = event.length; i < l; i++) {

this.$on(event[i], fn)

}

} else {

(vm.\_events[event] || (vm.\_events[event] = [])).push(fn)

}

return vm

}

this.bus.\_events 获取$on注册的事件

实例的\_events是什么？这就是在事件初始化的时候，initEvents函数中绑定了\_event属性并给其赋值为空对象。这个\_events属性就是用来作为当前实例的事件中心，所有绑定在这个实例上的事件都会存储在事件中心\_events属性中。

对vue-antd中的table进行全部行的替换和是行展示的实现

manageColumns(columns) {

    columns = columns.map((e) => {

        let { dataIndex, customRender, scopedSlots = {} } = e;

        let { customRender: slotCustomRender = "" } = scopedSlots;

        if (typeof customRender === "function") {

            e.customRenderFn = customRender;

            delete e.customRender;

            e.scopedSlots = Object.assign({}, scopedSlots, { "customRender": dataIndex });

        } else {

            if (slotCustomRender === "function") { e.customRenderFn = slotCustomRender; }

            e.scopedSlots = Object.assign({}, scopedSlots, { "customRender": dataIndex });

        }

        return e;

    });

    return columns;

},

getColumnsText(record, columns) {

    let { customRender, dataIndex, customRenderFn } = columns;

    let text = record[dataIndex];

    if (typeof customRenderFn === "function") { text = customRenderFn(text, record); }

    return text;

},

table的slot替换实现

<template #[col.dataIndex]="{record}" v-for="(col, i) in columns">

    <span :key="i">{{getColumnsText(record, col)}}</span>

</template>

<template #[slot]="{text, record, index, expanded}" v-for="(slot, i) in Object.keys($scopedSlots)">

    <span :key="i" v-if="tableOption.showSubtotal">

        <slot :name="slot" v-bind="{record, text, index, expanded}"></slot>

    </span>

    <slot :name="slot" v-bind="{record, text, index, expanded}" v-else></slot>

</template>