**箭头函数this的指向**

1.箭头函数中的this指向，内外相同

2.一般函数中this:自动指向正在调用当前函数的对象

3.new 可将构造函数中的this自动指向新对象

4.一个函数如果不调用，则this指向window。在定时器中，this默认指向window, 如果希望回调函数中的this和外部的this保持一致时，就可改为箭头函数

5.也可以自行改变this的指向，利用call apply bind的方法

**vue传参**

1.父->子: 属性下行 （props down）

(1)在父组件的子组件标签中将自己的模型变量赋值给子组件的props属性变量

(2)子组件通过 props:[“模型变量”]获取父组件传过来的参数

2.子->父: 事件上行 （events up）

(1)在父组件的子组件标签中定义自定义事件

<子组件 @自定义事件名="处理函数"

然后在methos中调用处理函数

处理函数(参数){

根据参数修改this.模型变量

}

(2)在子组件中任意位置 this.$emit(“自定义事件名”，参数值)

会触发父组件template中，提前绑定的@自定义事件，并将参数传递给父组件中的处理函数

3.兄弟间

(1)采用先给父，再由父绑定到另一个子

(2)

①新建一个eventBus空Vue对象

内容为： import Vue from ‘vue’;

export default new Vue();

②在需要传递参数的两个组件中都引入eventBus

③在发出参数的组件中

eventBus.$emit('myEvent',参数)

④在接受参数的组件中

eventBus.$on('myEvent',(e)=>{

console.log(“接收到的参数是”+e)

})

**如何定义一个各个组件都可用的方法**

在main.js中给vue添加方法

将该方法添加到vue的原型对象中

Vue. prototype.方法名=function(){}

**移动端屏幕适配**

https://www.w3cplus.com/mobile/lib-flexible-for-html5-layout.html

**移动端click事件**

<https://www.cnblogs.com/imwtr/p/5882166.html>

**闭包(3h+w回答类似问题套路)**

什么是:既重用变量，又保护变量不被污染的一种机制

为什么: 全局变量和局部变量都有不可兼得的优缺点

全局变量: 优: 可反复使用！ 缺:随处可用 极容易被篡改

局部变量: 优: 仅函数内可用,不会被篡改！ 缺: 不可重用！

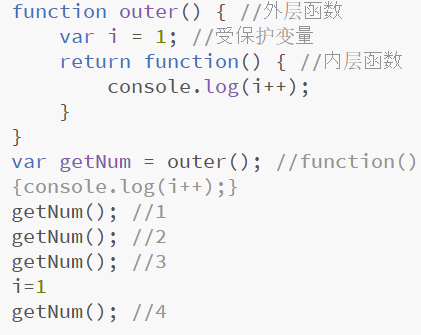
何时: 为一个函数永久绑定一个专属的可重用变量时

如何: 3步:

1. 用外层函数包裹受保护的变量和内层函数

2. 外层函数返回内层函数对象

3. 调用外层函数获得内层函数对象



闭包形成的原因: 外层函数的作用域对象无法释放

因为被内层函数对象的scope引用着

闭包缺点:

1. 比普通函数多占用一块内存——外层函数的作用域对象

2. 不会自动释放！易造成内存泄漏

解决: 一旦不再使用闭包！应主动释放对内层函数对象的引用

**作用域 作用域链 原型链**

作用域(scope):

什么是: 使用: 一个变量的可用范围

存储: 一个保存变量的对象！

为什么: 避免不同范围内的变量间互相干扰！

原理:

程序运行前:

1.创建执行环境栈数组(ECS): 用于储存将要调用的函数

2.调用浏览器软件的主程序: main()函数

3.main()函数会创建并引用window对象

在window中保存所有内置对象: Array, Date, console,..

定义函数时:

1.在window之外创建函数对象保存函数定义

2.在window中添加函数名变量

3.函数名变量引用函数对象

4.函数对象用scope属性引用回window

调用函数时:

1.在执行环境栈数组(ECS)中记录本次函数调用的函数名

2.为本次函数调用临时创建函数作用域对象

函数作用域对象的parent属性指向window

3.在函数作用域对象中临时创建局部变量

4.设置函数作用域对象的parent属性指向函数的scope

变量的使用顺序:

优先使用函数作用域对象中的局部变量

局部没有，才延parent属性向下找

调用后:

ECS中本次函数调用的记录出栈

导致函数作用域对象释放！

导致所有局部变量一同释放！

所以: 局部变量仅在函数调用时，函数内可用

不可重用！

作用域链(scope chain): 由多级作用域对象逐级引用形成的链式结构

保存着: 所有的全局变量和局部变量

控制着: 变量的使用顺序: 先局部后全局

原型链:

什么是: 由多级父对象逐级继承,形成的链式结构

保存着: 所有对象的属性

控制着: 对象成员的使用顺序和使用范围

顺序: 先自有，再向上

范围: 越上层的成员，共享范围越大！

vs 作用域链:

保存着所有不带.的变量

控制着变量的使用顺序: 先局部，后全局

总结:

只要不用.就可直接访问的变量，默认仅在作用域链中找

必须用.才能访问的变量，都在各自的原型链中查找

**数组api**

https://www.cnblogs.com/xhy-steve/p/5871182.html

**面向对象的封装 继承 多态**

https://www.cnblogs.com/meetup/p/5724334.html