Js中数据类型分原始数据类型与引用数据类型。

原始数据类型：布尔类型、Number、String、Nul、Undefine、Symbol

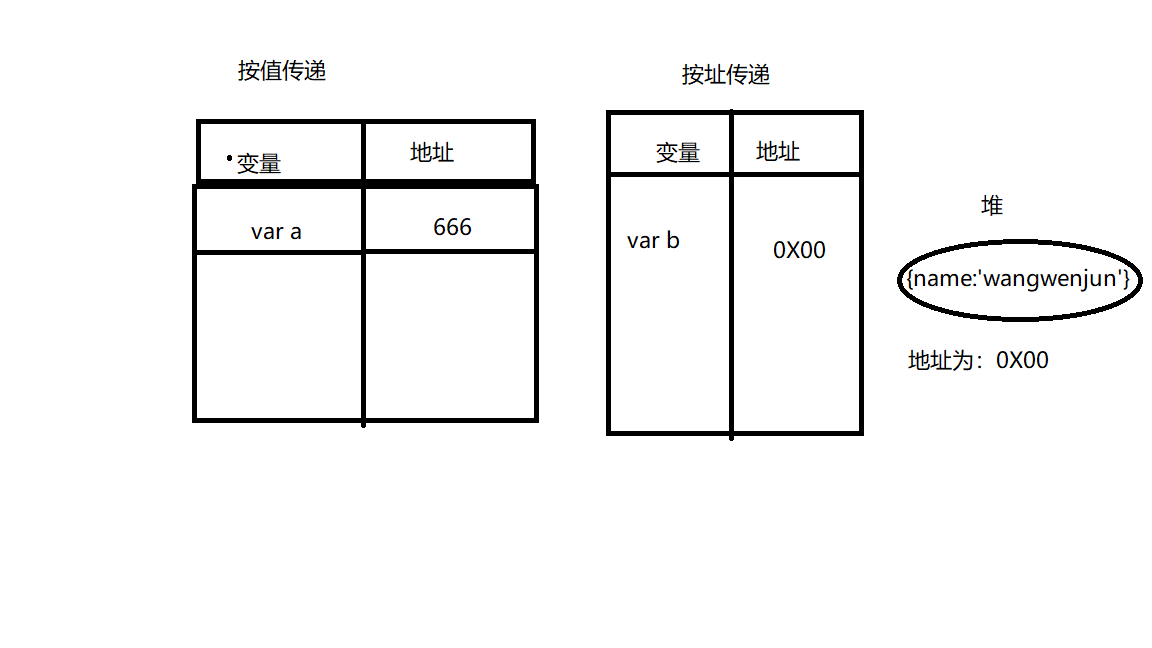
引用数据类型:Function、Date、Object、Array、正则

1. 按值传递与按址传递。

Js中一种数据类型一旦被定义，计算机会分给变量一块地址。

计算机内存中原始数据类型是按值传递的，定义一个变量 var a=666,内存地址直接将数值666放在内存地址中，并且直接将数值赋值给变量a;

引用数据类型是按址传递的，顾名思义，址指的是地址，如定义变量var b={name:’wangwenjun’}, 对象存放在计算机中的堆中，并且会给这个对象一个16进制的地址0X00，定义引用数据类型是，会将这个对象的地址传给变量b,这就叫按址传递。如下图；



如var a={name:“wang”,age:18}; var b=a; b.name=”zhu”;当为对象b赋值的时候，原始数据类型a的数据也变了，因为赋值b=a的时候，是将a的内存地址赋给了b;

明白按值传递与按址传递的原理，就明白了项目中为什么会出现有时候数据发生变化的原因。这也解释为什么在vue项目中，父子props down传递的数据是引用数据类型的时候，在子组件是可以直接修改的，而不需要再emit，实质也是传递的是应用数据的地址。

1. 浅拷贝与深拷贝

浅拷贝：浅拷贝是拷贝了对象的应用，当对象的属性是引用类型的时候，实质是复制其引用，当引用值发生改变时，拷贝值也跟着改变

深拷贝：深拷贝时另外申请了一块内存，拷贝对象的内容与原来一摸一样，更改原对象，拷贝对象也跟着变化。

3.递归实现深拷贝

实现拷贝的方式有很多，如for in ,es6的扩展运算符，Object.assign、JSON.parse(JSON.stringtify(obj))等等，总结了下，从严格意义上说，都算不上深拷贝，当然很多书本上讲JSON序列化算是深拷贝，你仔细研究会发现当一个对象的属性时null，或者undefine和function的时候，用序列化复制对象，拷贝的对象这些属性会丢失。如下时用递归实现的深拷贝；

var obj = {

    name: "wwj",

    age: 18,

    friend: {

        name: 'dog',

        age: 2

    },

    info: function () {

        console.log("i am boy")

    },

    car: null,

    girlfriend: undefined

}

var deepClone = function (obj) {

    let newObj = Array.isArray(obj) ? [] : {};

    if (obj && typeof obj == 'object') {

        for (let key in obj) {

            if (obj.hasOwnProperty(key)) {

                if (obj[key] && typeof obj[key] === 'object') {

                    newObj[key] = deepClone(obj[key])

                } else {

                    newObj[key] = obj[key]

                }

            }

        }

    }

    return newObj

}

var other = deepClone(obj);

console.log(other)