JS学习记录

Virgil233 已于 2023-10-14 15:01:49 修改

前端 专栏收录该内容

0 订阅 2 篇文章

目录

一、算法题杂记

（1）charCodeAt(*index*)　#返回指定位置的字符的 Unicode 编码，index为字符下标
（2）str.trim()　#去除字符串左右空格
（3）indexOf()　#返回某个指定的字符串值在字符串中首次出现的位置。如果没有找到匹配的字符串则返回 -1。
（4）isNaN(Number(x))　#判断如["2","1","+"]中数字和运算符，返回true的是运算符，false是数字
（5）Object如果属性是数组会自动将其转换为字符串，通过“ 对象名[数组名] ”获取
Object.values()返回值的数组，Map.values()返回一个值的可迭代对象
Map与Object区别：https://juejin.cn/post/6844903792094232584?searchId=202309111926565C16D4EF4166022D7D66

二、JS组成

ECMAscript（JS语言基础） + DOM + BOM

三、JS书写位置

（1）内部JS：直接写在html文件里，用script标签包住
（2）外部JS：代码写在.js文件里，通过<script src></script>引入，标签中不要写代码，写了也会被忽略，不执行！
（3）内联JS：代码写在标签内部，vue使用此模式

四、输入输出

（1）输出：document.write（如果内容是标签会被解析成网页元素）
alert　#弹出警告对话框
console.log　#控制台打印，调试使用
（2）输入：prompt　#显示对话框，提示用户输入
注：alert和prompt会跳过页面渲染先被执行

五、变量和常量

（1）let 变量名 = 值　#声明变量并赋值（let不允许多次声明一个变量）
（2）const 常量名 = 值　#声明常量并赋值（常量声明时必须赋值）
（3）命名规则：只能由下划线、数字、字母、$组成，且数字不能开头；严格区分大小写
（4）let和const：ES6新增加了两个重要的 JavaScript 关键字: let 和 const。let 声明的变量只在let 命令所在的代码块内有效。const 声明一个只读的常量，一旦声明，常量的值就不能改变。

在 ES6 之前，JavaScript 只有两种作用域：全局变量与函数内的局部变量。使用 var 关键字声明的变量不具备块级作用域的特性，它在 {} 外依然能被访问到。在 ES6 之前，是没有块级作用域的概念的。ES6 可以使用 let 关键字来实现块级作用域。JavaScript 中，var 关键字定义的变量可以在使用后声明（变量提升），let 关键字定义的

变量则不可以在使用后声明。

const 的本质: const 定义的变量并非常量，并非不可变，它定义了一个常量引用一个值。使用 const 定义的对象或者数组，其实是可变的。但是我们不能对常量对象重新赋值。

const优先：尽量使用const声明变量，语义化更好。注：基本数据类型值变化时不能用const，引用数据类型追加或删除可以用const（存地址）

var的变量提升：把所有var声明的变量（不包含赋的值）提升到当前作用域的最前面

六、数组

（1）数组声明：let 数组名 = [数据1，数据2，.....，数据n] / let 数组名 = new Array()

（2）数组新增：

array.push(item1,...,itemn) #将一个或多个元素添加到数组末尾，并返回数组新长度

array.unshift(item1,...,itemn) #将一个或多个元素添加到数组开头，并返回数组新长度

（3）数组删除：

array.pop() #删除最后一个元素，并返回该元素的值

array.shift() #删除第一个元素，并返回该元素的值

array.splice(index,howmany,item1,.....,itemX) #从index处开始删除/添加元素（howmany为删除个数，未规定则一直从index到结尾全删；item为要添加的元素）

（4）数组排序：

array.sort(function(a,b){ retrun a-b }) #升序

array.sort(function(a,b){ retrun b-a }) #降序

（5）遍历数组：

array.map(function(ele,index){}) #处理数据并返回新数组，ele为数组元素，index为索引号

array.forEach(function(ele,index){}) #调用数组每个元素，并传递给回调函数，无返回值

（6）其他方法：

array.join()	将数组组合成字符串，括号内为每个元素连接方式，如，“”等（默认为，）
array.filter(function(ele,index){return 筛选条件})	筛选符合条件元素，返回筛选后新数组
array.reduce(function(上一次值，当前值){}，初始值)	累计器，返回累计处理结果，常用于求和
array.find()	返回符合测试条件的第一个数组元素，没有返回undefined
array.every()	检测数组所有元素是否都符合指定条件，是返回true，否返回false

array.some()	检测数组元素是否都符合指定条件，有元素满足返回true，否则返回false
array.concat()	合并两个数组，返回新数组
array.reverse()	反转数组
array.findIndex()	查找元素索引值
Array.from()	伪数组转换为真数组（静态方法）

七、数据类型

（1）JS数据类型分为两大类：

基本数据类型（值类型，存值）：number数字型、string字符串型、boolean布尔型、undefined未定义型、null空类型.....

引用数据类型（存地址）：object对象、function函数、array数组

（2）模板字符串：用``(反引号)，内容拼接变量时用${ }包住变量，支持内容字符串换行输出

（3）检测数据类型：typeof x

（4）隐式转换：+两边有一个字符串则另一个也转成字符串（+作为正号解析可以转成数字型）；除了+以外的算术运算符如- * /等转成数字类型（比较运算符也把字符串转成数字）

（5）显示转换：

转成数字型：Number(x) parseInt(x) parseFloat(x)

注：Number对整个字符串转，存在非数字字符则直接NaN；parse一个一个字符转（如11aa，Number输出NaN,parse输出11）

'' 和null 变为0，undefined变为NaN

转成布尔型：Boolean(x)

注：''、0、undefined、null、false、NaN转换为布尔值后都为false，其余为true

八、匿名函数

（1）函数表达式：将匿名函数赋值给一个变量，通过变量名进行调用。如：let fn = function(){}通过fn()调用，函数调用只能在函数表达式声明后

（2）立即执行函数：用于避免全局变量之间污染，不需要调用，立即执行，必须加；

(function(形参){ console.log(11) })(实参); 或 (function(形参){ console.log(11) }(实参));

九、逻辑中断

只存在于&&和||中，&&左边为false或||左边为true时会让右边代码不执行

十、对象

（1）对象声明：let 对象名 = {} / let 对象名 = new Object()
（2）对象由属性和方法组成，属性成对出现，包括属性名和值，用：隔开，属性间用，隔开
（3）查的两种方法：对象名.属性名　对象名['属性名']
（4）方法由方法名和函数组成，用：隔开，方法间用，隔开，如fn : function(){ }
（5）遍历对象：for(let k in obj){ console.log(k) //打印属性名 console.log(obj[k]) //打印属性值 }
注：取出的k为字符串型，所以必须使用第二种查方法，而不能用.方法，因此也不推荐用于遍历数组！
（6）内置对象Math：

random	生成0-1之间的随机数（包括0不包括1）
ceil	向上取整
floor	向下取整
max	最大
min	最小
pow	幂运算
abs	绝对值

更多参考https://developer.mozilla.org/zh-CN/

十一、DOM

（1）Document Object Model，文档对象模型，用来呈现以及与任意HTML或XML文档交互的API（操作网页内容）
（2）根据CSS选择器获取DOM元素：
document.querySelector(' CSS选择器 ')　#选择匹配的第一个元素
document.querySelectorAll(' CSS选择器 ')　#选择匹配的多个元素
注：querySelectorAll返回的是一个伪数组，有长度有索引号，但是没有pop()、push()等方法，只能用for遍历
（3）操作元素内容：
innerText属性：将文本内容添加或更新到任意标签位置，但是只能显示纯文本，不解析标签
innerHTML属性：将文本内容添加或更新到任意标签位置，可以解析标签，多标签建议使用模板字符
（4）操作元素常用属性，如href、title、src等：对象.属性=值
（5）操作元素样式属性：

通过style属性操作：对象.style.属性=值（中间带-的属性用小驼峰命名法）
通过className操作：元素.className='css类名'（新值会覆盖旧值）
通过classList操作：元素.classList.add('类名')　#追加一个类
元素.classList.remove('类名')　#删除一个类
元素.classList.toggle('类名')　#切换一个类（有就删掉，没有就加上）
元素.classList.contains()　#是否包含某个类

（6）操作表单元素属性：DOM对象.属性名　#获取　DOM对象.属性名=新值　#修改

（7）自定义属性：以data-开头，用dataset对象方式获取

（8）定时器-间歇函数：
setInterval(函数名，间隔时间)　#开启，单位ms，函数名不加括号，返回值是定时器id数字
let 变量名=setInterval(函数名，间隔时间)→clearInterval(变量名)　#关闭

（9）事件监听：元素对象.addEventListener(' 事件类型 ' ，执行函数)
注：原始版本使用on事件，但on会被覆盖，addEventListener可以绑定多次

（10）鼠标事件：mouseenter　#鼠标经过　mouseleave　#鼠标离开
注：mouseover和mouseout会有冒泡效果，所以不推荐

（11）焦点事件（表单获得光标）：focus　#获得焦点　blur　#失去焦点

（12）键盘事件：keydown　#键盘按下触发　keyup　#键盘抬起触发

（13）文本事件：input　#用户输入事件

（14）获取事件对象：事件监听的执行函数里第一个参数就是事件对象，通常用e表示

（15）事件对象常用属性：
type　#获取当前事件类型
clientX/clienY　#获取光标相对于浏览器可见窗口左上角的位置
offsetX/offsetY　#获取光标相对于当前DOM元素左上角的位置
key　#用户按下的键盘的值

（16）环境对象this：指向调用者（普通函数里this指向window，严格模式下指向undefined）

（17）回调函数：函数A作为参数传给函数B时，A为回调函数

（18）事件流的两个阶段：捕获阶段、冒泡阶段

（19）阻止冒泡：事件对象.stopPropagation()

（20）事件解绑：on事件方式，直接使用null覆盖就可以
addEventListener方式必须使用removeEventListener(事件类型，事件处理函数，[获取捕获或者冒泡阶段])注：匿名函数无法解绑！

（21）事件委托：利用事件冒泡的特点，减少注册次数，提高程序性能（通过e.target.tagName获取点击对象，对象是字符串且大写！）

（22）阻止默认行为：e.preventDefault()比如组织链接跳转、表单域跳转等

（23）页面加载事件：window.addEventListener('load', function(){})　#等页面所有元素加载完毕
document.addEventListener('DOMContentLoaded', function(){})　#等DOM加载完毕，不等样式表、图片等完全加载

（24）页面滚动事件：window.addEventListener('scroll', function(){})
元素.scrollLeft/scrollTop　#获取元素内容往左/往上滚出去看不到的距离（可读写，数字型。不带单位）

注：检测窗口滚动条检测的是html标签，获取方法为document.documentElement
window.scrollTo(x,y)　#滚动到指定坐标

（25）页面尺寸事件：window.addEventListener('resize', function(){})　#窗口尺寸改变时触发
clientWidth和clientHeight：获取元素可见部分宽高（包含padding，不包含border、margin、滚动条等，只读！）
offsetWidth和offsetHeight：获取元素自身宽高（包含padding、border、滚动条等，只读！）
offsetLeft和offsetTop：获取元素距离最近一级带有定位的祖先元素左、上距离（只读！）
element.getBoundingClientRect()　#返回元素大小及其相对于视口（可视区）的位置

（26）实例化日期对象：date = new Date()

getFullYear()	获取年份
getMonth()	获取月份0-11
getDate()	获取天
getDay()	获取星期0-6
getHours()	获取小时0-23
getMinutes()	获取分钟0-59
getSeconds()	获取秒0-59

（27）时间戳：从1970年1月1日0时0分0秒起至现在的毫秒数，用于计算倒计时时间

获取时间戳：date.getTime()　+new Date()　Date.now()

（28）DOM节点类型：
元素节点：所有的标签，比如body、div，html是根节点
属性节点：所有的属性，比如href
文本节点：所有的文本
其他节点

（29）查找DOM节点：
查找父节点：子元素.parentNode，返回最近一级父节点，找不到返回nul
查找子节点：childNodes　#获得所有子节点，包括文本节点（空格、换行）、注释节点等
children属性　#仅获得所有元素节点，返回伪数组
查找兄弟节点：nextElementSibling下一个兄弟　previousElementSibling上一个兄弟

（30）增加DOM节点：document.createElement(' 标签名 ')
追加节点：父元素.appendChild(要插入的元素)　#作为最后一个子元素插入到父元素

父元素.insertBefore(要插入元素，在哪个元素前面)　#插到某个子元素前面

（31）克隆DOM节点：元素.cloneNode(布尔值)，true代表包含后代节点和标签里所有内容一起克隆，false则不包含，默认为false

（32）删除DOM节点：父元素.removeChild(要删除元素)，如果不存在父子关系则删除失败

（33）移动端事件：touchstart　#手指触摸到一个DOM元素时触发

touchmove　#手指在一个DOM元素上滑动时触发

touchend　#手指从一个DOM元素上移开时触发

（34）swiper插件：https://www.swiper.com.cn/

十二、BOM

（1）Browser Object Model，浏览器对象模型

（2）定时器-延时函数：

setTimeout(函数名，等待时间)　#开启，单位ms，函数名不加括号，返回值是定时器id数字

let 变量名=setTimeout(函数名，等待时间)→clearTimeout(变量名)　#关闭

（3）JS执行机制：单线程，同步任务都在主线程上执行，形成一个执行栈；异步任务通过回调函数实现，添加到任务队列中，主要包含以下三种类型：

普通事件：click、resize等；资源加载：load、error等；定时器：setInterval、setTimeout等

机制（事件循环event loop）：先执行完所有同步任务，然后依次读取异步任务放入执行栈执行（当执行栈中有异步任务时，交给对应的异步进程处理（浏览器处理），处理完放入任务队列中

（4）location对象：拆分并保存URL地址各个组成部分

href属性：获取完整URL地址，赋值时用于地址跳转

search属性：获取地址中携带的参数，？后面的部分

hash属性：获取地址中的哈希值，#后面的部分

reload属性：刷新当前页面，传入参数true时表示强制刷新

（5）navigator对象：记录浏览器自身信息

userAgent属性：检查浏览器版本及平台

（6）history对象：管理历史记录，与地址栏操作相对应，如前进、后退、历史记录等

back()方法：后退；forward()方法：前进；go(参数)：参数1前进一个页面，-1后退一个页面

（7）本地存储：存储在用户浏览器中，分为localStorge和sessionStorge

（8）localStorage：永久存储除非手动删除，可以多窗口（页面）共享，键值对形式存储，只能存储字符串类型

localStorage.setItem(' key ',value)　#存储数据

localStorage.getItem(' key ')　#获取数据

localStorage.removeItem(' key ')　#删除数据

（9）sessionStorge：关闭浏览器窗口就消失，同一个窗口（页面）下数据可共享，以键值对形式存储，用法基本同localStorage

（10）本地存储复杂数据类型：JSON.stringify(复杂数据类型)　取：JSON.parse(JSON字符串)

（11）正则表达式：用于匹配字符串中字符组合的模式，也是一种对象，通常用于查找、替换符合正则表达式的文本

const 变量名 = / 表达式 / → 变量名.test(被检测的字符串) 返回true/false

→ 变量名.exec(被检测的字符串）成功返回数组，否则返回null

（12）元字符分类：

边界符：表示位置，必须用什么开头和结尾（^表示以谁开始，$表示以谁结束，同时出现时为精确匹配）

量词：表示重复次数

*	重复零次或更多次
+	重复一次或更多次
?	重复零次或一次
{n}	重复n次
{n,}	重复n次或更多次
{n,m}	重复n到m次

字符类：

[ ]匹配字符集合：字符串包含[ ]内的任意一个（精确匹配且没有量词时只选一个）字符，都返回true。里面可以加-连字符表示一个范围，如：[a-z]表示a到z都可以，[a-zA-Z]表示大小写的a到z都可以，[0-9]表示数字0-9都可以。里面还可以加^表示取反

. 匹配除换行符之外的任何单个字符

预定类	说明
\d	0-9,相当于[0-9]
\D	0-9以外的，相当于[^0-9]
\w	任意字母、数字、下划线，相当于[A-Za-z0-9_]
\W	除任意字母、数字、下划线，相当于[^A-Za-z0-9_]
\s	空格符、换行符、制表符等，相当于[\t\r\n\v\f]
\S	相当于[^\t\r\n\v\f]

（13）修饰符：/表达式/修饰符

i是ignore缩写，匹配时字母不区分大小写；g是global缩写，匹配满足所有正则表达式结果

（14）替换：字符串.replace(/正则表达式/，'替换的文本')

十三、作用域

（1）作用域分为全局作用域和局部作用域。局部作用域分为函数作用域和块作用域，let和const会产生块作用域，var不会产生块作用域

（2）作用域链本质上是底层的变量查找机制

十四、垃圾回收机制

（1）内存生命周期：内存分配、内存使用、内存回收。一般情况下，全局变量不会回收，关闭页面时回收；局部变量不用时被自动回收

（2）内存泄漏：程序中分配的内存由于某种原因未释放或无法释放

（3）垃圾回收算法：引用计数法、标记清除法

引用计数法：看一个对象是否有指向它的引用，没有就回收，IE采用。但存在嵌套引用（循环引用）的问题，当两个对象互相引用时，引用次数永远不会是0

标记清除法：从根部（全局对象）出发，无法到达的对象标记为不再使用

十五、闭包

（1）闭包=内层函数+外层函数的变量（内层函数用到了外层函数的变量）

（2）闭包作用：封闭数据，提供操作，外部可以访问函数内部的变量

（3）闭包可能存在内存泄漏的问题！

十六、函数其余参数

（1）动态参数：arguments，是函数内置的伪数组变量，包含调用时传入的所有实参

（2）剩余参数：...形参名，将一个不定数量的参数表示为一个真数组

（3）展开运算符：... ，将一个数组展开，通常用于求数组最大最小值（Math.max(...arr)），合并数组arr = [...arr1, ...arr2]等

十七、箭头函数

（1）适用于需要匿名函数的地方

（2）语法：只有一个形参时可以省略小括号

函数体只有一行可以省略大括号写到一行上，且无需写return直接返回值

加上小括号可以直接返回一个对象　fn1 = uname => ({uname: uname})

（3）参数：没有动态参数，但是有剩余参数

（4）箭头函数不会创建自己的this，只会从作用域链的上一层沿用this

十八、解构赋值

（1）数组解构：将数组单元值快速批量赋值给一系列变量
（2）数组解构语法：左侧的[ ]用于批量声明变量，右侧数组单元值赋值给左侧变量
注：数组开头，前面有语句一定加；　如：交换变量；[b,a] = [a,b]
（3）对象解构：将对象属性和方法快速批量赋值给一系列变量
（4）对象解构语法：左侧的{ }用于批量声明变量，右侧对象属性值赋值给左侧变量，变量名要与属性名相同，否则会undefined
（5）对象结构改变量名：　旧变量名：新变量名

十九、构造函数

（1）可以通过构造函数快速创建多个类似的对象，命名以大写字母开头，只能由new操作符执行
（2）实例化执行过程：创建新的空对象→构造函数this指向新对象→执行构造函数代码，修改this，添加新的属性→返回新对象
（3）实例成员：实例对象中的属性和方法
静态成员：构造函数中的属性和方法
（4）内置构造函数：引用类型：Object、Array、RegExp(正则表达式)、Date等
包装类型：String、Number、Boolean等
（5）Object常用静态方法：
Object.keys　#获取对象中所有属性，返回一个数组
Object.values　#获取对象中所有值，返回一个数组
Object.assign(新对象，被拷贝的对象/新增属性)　#对象拷贝或给对象添加属性
（6）String常用实例方法：

str.split()	将字符串拆分成数组，括号内为每个元素连接方式，如，“”等
str.length()	获取字符串长度
str.substring(要截取的第一个字符索引[，结束索引号])	字符串截取，截取部分不包含结束索引号对应字符
str.startsWith(检测字符串[，检测位置索引号])	检测是否以某字符开头
str.includes(搜索字符串[，检测位置索引号])	判断字符串是否包含在另一字符串中，返回true或false
str.toUpperCase()	字母转成大写

str.toLowerCase()	字母转成小写
str.indexOf()	检测是否包含某字符
str.endsWith()	检测是否以某字符结尾
str.replace()	替换字符串，支持正则匹配
str.match()	查找字符串，支持正则匹配

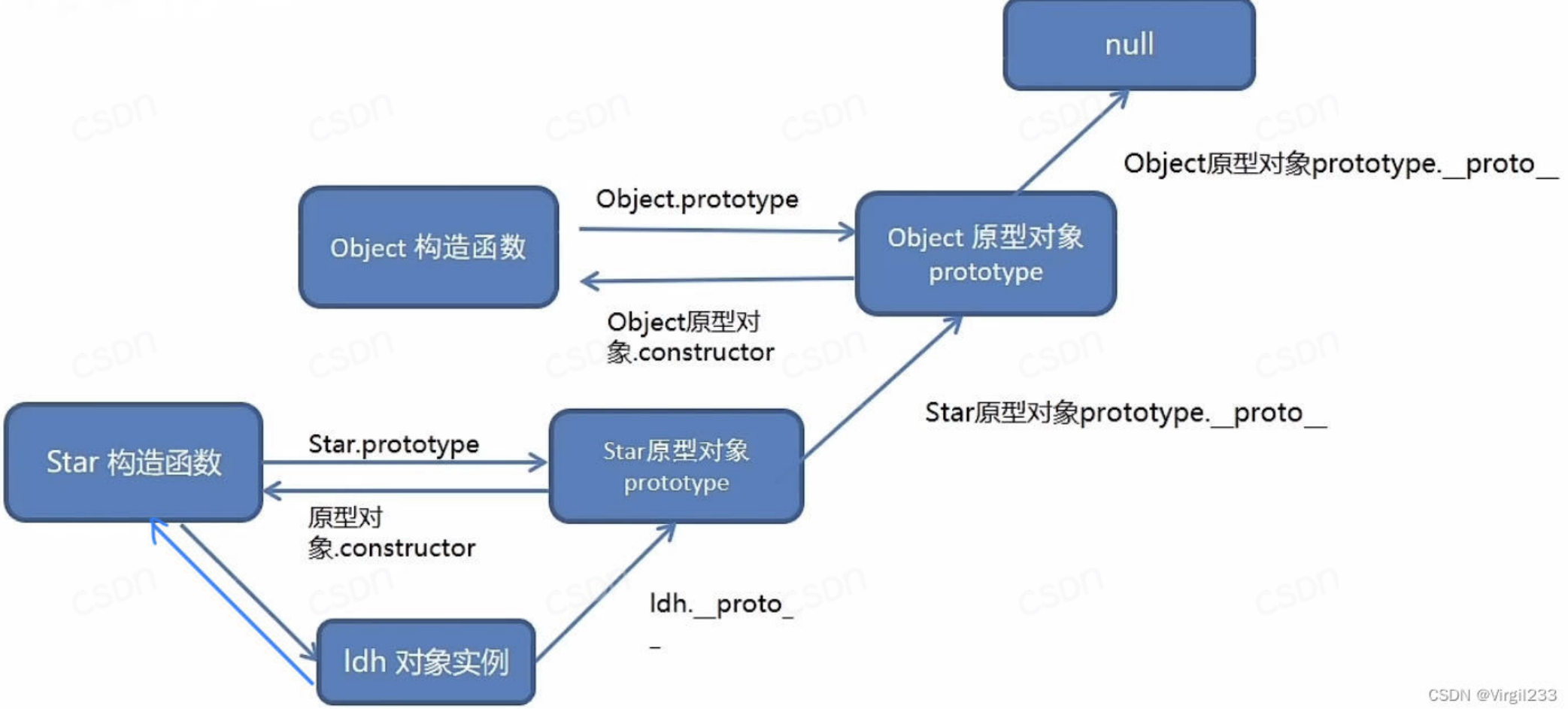
（7）number.toFixed()　#设置保留小数位的长度，四舍五入

二十、面向对象

（1）面型对象具有灵活、代码可复用、容易维护和开发的优点，更适合多人合作的大项目
（2）面向对象特性：封装性、继承性、多态性
（3）JS实现面向对象通过构造函数，但存在内存浪费的问题

二十一、原型

（1）本质上是一个对象，构造函数通过原型分配的函数所有对象共享，每一个构造函数都有一个prototype属性，对象实例化不会多次创建原型上的函数，节约内存。构造函数和原型里面的this指向实例化对象
（2）每个原型对象都有一个constructor属性(构造方法)，指向该原型对象的构造函数
　　constructor使用场景：如果有多个对象的方法，可以给原型对象采取对象的形式赋值，但这样会覆盖构造函数原型对象原来的内容，修改后的constructor不再指向构造函数。此时，可以在修改后的原型对象中，添加一个constructor指向原来的构造函数
（3）每一个实例对象都有一个属性__proto__（只读，只能获取，不能赋值）指向构造函数的prototype原型对象，因此对象可以使用构造函数prototype原型对象的属性和方法
　　__proto__对象原型里也有一个constructor属性，指向创建该实例对象的构造函数
（4）原型链：



查找规则：当访问一个对象的属性或方法时，首先查找这个对象自身有没有；如果没有则查找它的原型（__proto__指向的prototype原型对象）；还没有则查找原型对象的原型（Object的原型对象）；以此类推直到Object为止（null）

注：只要是对象就有__proto__，指向上一层原型对象；只要是原型对象就有constructor，指向创造它的构造函数

使用instanceof运算符检测构造函数的prototype属性是否出现在某个实例对象的原型链上

二十二、深浅拷贝

（1）浅拷贝拷贝的是地址，常见方法：拷贝对象：Object.assign()/{...ob} 拷贝数组：Array.prototype.concat()/[...arr]，单层对象可用，多层（对象里面还有对象）会出现问题

（2）深拷贝拷贝的是对象，常见方法：通过递归实现；lodash/cloneDeep；通过JSON.stringify()

注：利用递归实现时，要先考虑数组的情况再考虑Object的情况，因为数组也是一种Object

二十三、改变this指向

（1）fun.call(thisArg,arg1,arg2...）调用函数同时指定被调用函数this的值，返回值就是函数返回值

（2）fun.apply(thisArg,[argsArray])调用函数同时指定被调用函数this的值，返回值就是函数返回值

（3）fun.bind(thisArg,arg1,arg2...）不调用函数，返回由指定的this值和初始化参数改造的原函数拷贝（新函数）

二十四、防抖与节流

（1）防抖：单位时间内，频繁触发事件，只执行最后一次。例如：搜索框搜索输入，等用户最后一次输入完才发送请求

（2）防抖方法：lodash/debounce(func, [wait])　延迟wait毫秒之后调用func方法

（3）手写防抖：声明一个定时器变量→每次触发事件先判断是否有定时器(setTimeout)，有则先清除之前的定时器，没有则开启定时器，并保存到变量中→在定时器里调用要执行的函数

注：将声明变量后的操作放到一个function中return，这样才能每次触发事件都执行

（4）节流：单位时间内，频繁触发事件，只执行一次，适用于鼠标移动、页面尺寸缩放、滚动条滚动等高频事件

（5）节流方法：lodash/throttle(func, [wait])　在wait毫秒内最多执行一次func

（6）手写节流：声明一个定时器变量→每次触发事件先判断是否有定时器(setTimeout)，有则不开启新的定时器，没有则开启定时器，并保存到变量中→在定时器里调用要执行的函数并清空定时器

注：清除定时器不能用clearTimeout，而应该用timer = null，因为setTimeout中定时器还在运作，无法清除