Es6是Js的国际化标准

1. Js的防抖和节流
2. 防抖：当持续触发事件，一定时间内没有在触发事件，事件处理函数才会执行一次，如果设定的时间到来之前，又一次触发了事件，就重新开始延时

实际的应用场景：

1. 使用echarts时，改变浏览器的宽度的时候，希望重新渲染
2. Echarts可以使用防抖函数，提升性能
3. 输入搜索，输入结束n秒才进行搜索，n秒内又输入的内容，就重新计时
4. Js闭包的经典应用
5. 节流：当持续触发事件的时候，保证一段时间内，只调用一次事假处理函数

实际场景：

1. 表单提交
2. js的数据类型
3. 基本数据类型： number、string、undefined、null、booleam、symbol、bigInt

(number: NaN有效数字， isNaN是否为一个有效数字，infinity无穷大的值)

（NaN != NaN， NaN它和谁的不相等， 通过 isNaN来检测是否为有效数字，如果是返回false，否则返回true）

1. 引用数据类型：object、function
2. 数据类型检查：

Typeof 检测数据类型的逻辑运算

1.（注意typeof(null) 返回object; typeof返回的结果都是字符串； typeof不能细分对象类型 ）

2.（typeof typeof typeof([12, 23]) => “string”, 因为typeof([12, 23])=>”object”）

Instanceof 检测是否为某个类的实例

Constructor 检测构造函数

Obje.prototype.toString.call 检测数据类型

1. 把其他数据类型转换为Number类型的方法
2. 强转换： Number() , Number(true)=>1, Number(false)=>0
3. 弱转换: parseInt()、 parseFloat()

parseInt(“”)=> NaN

Number("")=>0

parseInt(null)=>NaN

Number(null)=>0

isNaN("") =>false 隐式（number）转换 isNaN（0）

isNaN(null)=>false 隐式（number）转换 isNaN（0）

parseInt("12px")=>12

Number("12px")=>NaN

isNaN("12px")=>true

parseFloat('1.6px') + parseInt('1.2px') + typeof parseInt(null)=>”2.6number”

isNaN(Number(!!Number(parseInt('0.8')))) => false

typeof !parseInt(null) + !isNaN(null)=>”booleamture”

[]==true=>false([]通过Number([])=>0, true转换为数字为1)

1. == 比较的规律：
2. 对象== 字符串， 对象转为字符串
3. Undefined == null，但是Undefined，null 和其他值不都相等
4. NaN和谁都不相等
5. 剩下两边不同都转为数字

[] == false=> true

![]==false=>true

运算符优先级，![] 转换为布尔值再取反

\*遵循的规律：只有0，NaN,null,undefined,’’,这五个值是false，其他全是true

1. +号运算规则：
2. 在两边出现字符串（或对象）的情况，加号一个是字符串拼接
3. { } + 0 大括号的特殊性：代表对象、代码块；此处把{}当做一个代码块，所以这里就是+0，结果就是0；+值 这个操作是数学运算
4. 0 + {} 是数学运算 "0[object Object]"

10 + false + undefined + [] + ‘xiaoming’ + null + true + {}

10 + false => 10 + 0=>10

10 + undefined =>10+NaN=>NaN

NaN + []=>NaN + ‘’=>’NaN’ (把对象转换为数字，要先转换成字符串)

’NaN’ + ‘xiaoming’ =>’NaNxiaoming’

’NaNxiaoming’ + null=>’NaNxiaomingnull’

’NaNxiaomingnull’+true=>’NaNxiaomingnulltrue’

’NaNxiaomingnulltrue’+ {}=>’NaNxiaomingnulltrue[object object]’

5.对于“+”来说，有两个含义：第一个含义是做字符串拼接，第二个含义是加减法中的加法。

    1，如果操作数里有一个是字符串，其他的值将被转换成字符串；

    2，其他情况，操作数转换成数字执行加法运算。

而对于“ - ”来说，只有一个含义，就是做减法，自然不会转化成字符串了。

1. 会改变数组的方法有：

push()

pop()

shift()

unshift()

splice()

sort()

reverse()

forEach()

1. 不会改变数组的方法有：

filter()

concat()

slice()

map()

七、this有四种情况：

1)当在函数调用的时候指向widow

2)当方法调用的时候指向调用对象

3)当用apply和call上下文调用的时候指向传入的第一个参数

4)构造函数调用指向实例对象

八、JavaScript RegExp 对象有 3 个方法：test()、exec() 和 compile()。

1)test（）用来检测某个字符串是否匹配某个正则表达式，有就说true，没有就说false

2)Exec() 用来检测某个字符串受否匹配某个正则表达式的值，返回一个配置值的数组，否则返回null

3)Compile（）可以在脚本执行中编译正则表达式，也可以改变已有的表达式

# 九、堆栈内存

1）栈内存ECStack（execution context stack）浏览器会在计算中分配一块内存，专门用来供代码执行的。

2）堆内存Heap 浏览器把内置的一些属性方法放在一个单独的内存中（全局对象 Go global object）

\*任何开辟的内存都有一个十六进制的内存地址，方便后期找到这个内存

列：var a = 12这一步是如何操作的呢

先创建值—》再创建变量—》最后让变量和值关联在一起

\*基本类型数据，直接存储到栈中

\*引用类型，是先开辟一个堆内存，把东西存储进去，最后把地址放到栈里面，供变量使用

1. Math.round(）函数返回一个数字四舍五入后最接近的整数

如果参数的小数部分大于0.5，四舍五入到相邻的绝对值更大的整数

如果参数的小数部分小于0.5，四舍五入到相邻的绝对值更小的整数

如果参数的小数部分等于0.5，四舍五入到相邻的在正无穷（+∞）方向上的整数。

例：

x=Math.round(2019.49) ;      //2019

x=Math.round(2019.5);         //2020

x=Math.round(-2019.5);        //-2019

x=Math.round(-2019.51);      //-2020

## History 对象

History 对象包含用户（在浏览器窗口中）访问过的 URL。

History 对象是 window 对象的一部分，可通过 window.history 属性对其进行访问。

注意： 没有应用于 History 对象的公开标准，不过所有浏览器都支持该对象。

History 对象属性

|  |  |
| --- | --- |
| length | 返回历史列表中的网址数 |

History 对象方法

|  |  |
| --- | --- |
| 方法 | 说明 |
| back() | 加载 history 列表中的前一个 URL |
| forward() | 加载 history 列表中的下一个 URL |
| go() | 加载 history 列表中的某个具体页面 |

## JavaScript实现继承共6种方式：

原型链继承、借用构造函数继承、组合继承、原型式继承、寄生式继承、寄生组合式继承。



1. 不支持冒泡的事件

常见的不支持冒泡事件有：

①focus

②blur

③mouseenter

④mouseleave

⑤load

⑥unload

⑦resize

1. js的全局函数



1. js对象可以分为三类
2. 内置对象：Regexp、 Date、 Error、 Function、 Arry、 Boolean、 Number 、String、Object、Math、Arguments
3. 宿主对象：（运行环境提供的对象）Window、Form、Image、Element、Documen
4. 自定义对象：开发人员自己定义的对象