1.eval用法及作用域？

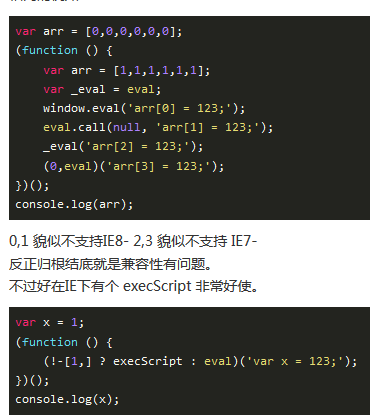
eval(string) 可计算某个字符串，并执行其中的的 JavaScript 代码。返回值：通过计算 string 得到的值（如果有的话）。

该方法只接受原始字符串作为参数，如果 string 参数不是原始字符串，那么该方法将不作任何改变地返回。因此请不要为 eval() 函数传递 String 对象来作为参数。

如果试图覆盖 eval 属性或把 eval() 方法赋予另一个属性，并通过该属性调用它，则 ECMAScript 实现允许抛出一个 EvalError 异常。

如果参数中没有合法的表达式和语句，则抛出 SyntaxError 异常；如果非法调用 eval()，则抛出 EvalError 异常；如果传递给 eval() 的 Javascript 代码生成了一个异常，eval() 将把该异常传递给调用者。

严格模式下eval为独立作用域，正常模式eval为当前函数作用；早函数作用域中将eval设置为独立作用域的方法有四种：https://www.cnblogs.com/52cik/p/js-eval.html



2.documents,callee,caller区别

argument代表当前函数的参数数组；

arguments.callee表示引用当前正在执行的函数

functionName.caller返回一个函数的引用，该函数调用了当前函数。functionName 对象是所执行函数的名称。如果为null,表示该函数是由顶层调用的。（对于函数来说，claller只有在函数被调用时才有定义，如果函数是有顶层调用的，那么caller包含的就是null,如果在字符串上下文中调用，那么caller属性，那么结果和 functionName.toString 一样，也就是说，显示的是函数的反编译文本。）

3.ES7和ES8的新特性？

https://blog.csdn.net/sinat\_17775997/article/details/78823550

ES7：\*\*指数操作符(ES7之前使用递归或者Math.pow()方法)

includes()判断数组是否包含某个值(ES7之前使用indexOf()判断)

ES8：Object.values()

Object.entries()

padStart()

padEnd()

Object.getOwnPropertyDescriptors()(ES8之前只能获取某个属性的描述器)

函数参数列表结尾允许逗号

Async/Await 异步操作(ES8之前使用函数或Promise完成异步操作)

4.Object对象的属性方法

https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global\_Objects/Object/getOwnPropertyDescriptors

5.十个最佳ES6特性

https://blog.fundebug.com/2017/08/21/10-best-es6-feature/

函数参数默认值

模板字符串

多行字符串

解构赋值

对象属性简写

箭头函数

Promise

Let与Const

类

模块化

6.height()、offsetHeight、clientHeight、scrollHeight、innerHeight()、outerHeight()的区别？

(Chrome选择元素区域显示的宽度和高度是包含border+padding+content部分--标准盒模型)

offsetHeight：JS对象的方法，高度是content+padding+border。

clientHeight：JS对象方法，返回内容的可视高度，即content+padding，不包含边框、边距或滚动条。

scrollHeight：JS对象方法，返回content+padding+滚动条。

height()：JQuery对象，返回元素的高度或者说这个div的内容的高度。即content部分高度。

innerHeight()：JQuery对象，返回content+padding。

outerHeight()：JQuery对象，不加参数返回content+padding+border。参数为true，返回content+padding+border+margin的高度。

7.JQuery中offset和position的区别？

offset()：获取匹配元素在当前视口的相对偏移。总是计算相对于文档的位置， 无论元素的父元素或祖先元素的position属性是什么。

返回的对象包含两个整形属性：top 和 left。

此方法只对可见元素有效。

position()：获取匹配相对父元素的偏移。

8.js中的变量提升和函数提升

首先 javascript 是一种弱类型、动态的、解释型的脚本语言。

弱类型：类型检查不严格，偏向于容忍隐式类型转换。   
 强类型：类型检查严格，偏向于不容忍隐式类型转换。   
 动态类型：运行的时候执行类型检查。   
 静态类型：编译的时候就知道每个变量的类型。   
 解释型：程序不需要编译，程序在运行的时候才翻译成机器语言，每执行一次都要翻译一次，因此效率比较低，但是跨平台性好。   
 编译型：程序在执行之前需要一个专门的翻译过程，把程序编译为机器语言的文件，运行时直接使用编译的结果就行了。   
 标记语言：标记语言的存在就是用来被读取(浏览)的，而其本身是没有行为能力的，在标记语言里你会看到<和>这些尖括号，这是用来写出“层次”和”属性”的，换句话说，它是被动的。并不具备与访问者互动的能力。   
 编程语言：它是具有逻辑性和行为能力，这是主动的。说通俗一点，它是有思想的。

脚本语言：它介于标记语言和编程语言之间，脚本语言不需要编译，可以直接用，由解释器来负责解释。

js解析原则：首先js引擎在读取js代码时会进行两个步骤，第一个步骤是解释，第二个步骤是执行。 所谓解释就是会先通篇扫描所有的Js代码，然后把所有声明提升到顶端，第二步是执行，执行就是操作一类的。

变量提升（把变量声明提升到当前执行环境的最顶端）；函数声明提升 （函数声明提升直接把整个函数提到执行环境的最顶端）；函数提升在变量提升之上。

9. CSS设置文字超出部分显示缩略号？

使用属性overflow:hidden; text-overflow:ellipsis; white-space:nowrap

10.定义对象属性时其描述器的属性值默认是true还是false？

只有使用Object.defineProperty定义对象属性，则描述器属性标签默认为false；其他方式均为true；

11.JS中escape和unescape函数？

escape(string) 函数可对字符串进行编码，编码格式为Unicode，这样就可以在所有的计算机上读取该字符串。

unescape(string) 函数可对通过 escape() 编码的字符串进行解码。

注释：ECMAScript v3 已从标准中删除了 unescape() 函数，并反对使用它，因此应该用 decodeURI() 和 decodeURIComponent() 取而代之。

12.JS中encodeURI()、decodeURI()、encodeURIComponent()、decodeURLComponent()函数？

https://www.cnblogs.com/shuiyi/p/5277233.html

[encodeURI](https://www.w3.org/html/ig/zh/wiki/ES5/%E6%A0%87%E5%87%86_ECMAScript_%E5%86%85%E7%BD%AE%E5%AF%B9%E8%B1%A1#encodeURI) 和 [decodeURI](https://www.w3.org/html/ig/zh/wiki/ES5/%E6%A0%87%E5%87%86_ECMAScript_%E5%86%85%E7%BD%AE%E5%AF%B9%E8%B1%A1" \l "decodeURI) 函数操作的是完整的 URI；这俩函数假定 URI 中的任何保留字符都有特殊意义，所有不会编码它们。（通常两次转换encode(encode(string))）

[encodeURIComponent](https://www.w3.org/html/ig/zh/wiki/ES5/%E6%A0%87%E5%87%86_ECMAScript_%E5%86%85%E7%BD%AE%E5%AF%B9%E8%B1%A1#encodeURIComponent) 和 [decodeURIComponent](https://www.w3.org/html/ig/zh/wiki/ES5/%E6%A0%87%E5%87%86_ECMAScript_%E5%86%85%E7%BD%AE%E5%AF%B9%E8%B1%A1" \l "decodeURIComponent) 函数操作的是组成 URI 的个别组件；这俩函数假定任何保留字符都代表普通文本，所以必须编码它们，所以它们（保留字符）出现在一个完整 URI 的组件里面时不会被解释成保留字符了。

13.Object.is(a,b)函数与===的区别？

==：等同，比较运算符，两边值类型不同的时候，先进行类型转换，再比较；

===：恒等，严格比较运算符，不做类型转换，类型不同就是不等；

Object.is()是ES6新增的用来比较两个值是否严格相等的方法，与===的行为基本一致。不过有两处不同：+0不等于-0; NaN等于自身。

Object.is(NaN, NaN) true

NaN === NaN false

NaN == NaN false

Object.is(+0,-0) false

+0 === -0 true

+0 == -0 true

14.JS中构造函数和普通函数的区别？

构造函数使用new操作符调用，普通函数直接调用；

函数调用：(1)构造函数内部会创建一个实例，普通函数不会创建新对象。

(2)构造函数内部的this指向是新创建的实例，而普通函数内部的this指向调用函数的对象(如果没有对象调用，默认为window)。

返回值：构造函数的返回值如果是基本数据类型，则返回值为创建的实例；如果是复杂数据类型，则返回值是return的对象。（？？？如果返回的是函数？）

15.JQuery 监听input的value值改变：

(1).onchange，该事件必须由鼠标或者键盘激发

$("xxx").on("change",function(){

alert("变化了");

})

(2) onpropertychange，ie有专属的方法，仅支持ie内核的浏览器，同样必须由鼠标或者键盘激发

(3)ie之外的其他浏览器，比如firefox，opera，提供了oninput，同样必须有鼠标或者键盘激发，若想同时支持ie,firefox等浏览器，可以同时绑定两个事件

$("xxx").on(" input propertychange",function(){

alert("变化了");

})

16.使用new+构造函数创建的对象和只使用构造函数创建的对象的区别？

Object/Function/Array的对比结果都好说，因为他们构建出来的都是新的对象。

Number：实际上new Number()构建出来的变量n是一个特殊的对象，Number对象。但是Number()构建出来的是一个数值。所以new Number()==Number()为true，new Number()===Number()为false。

Boolean: new Boolean()构建出来的是一个Boolean对象，但是Boolean()构建出来的是一个布尔值。

String: new String()构建出来的是一个String对象，但是String()构建出来的是一个字符串。