**1.全局对象**

ECMAScript的全局对象是window,node的全局对象是global。window对象可以访问全局属性和方法。

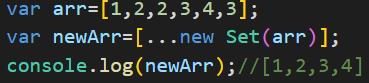




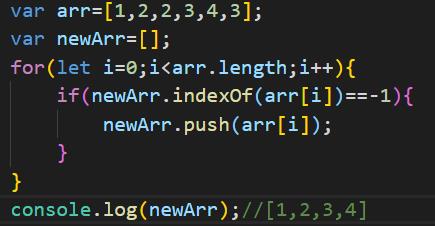
**2.数组去重**

**方法1：es6的Set**

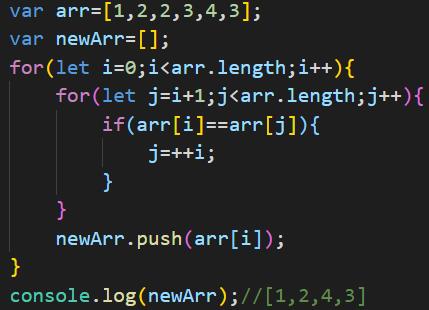
[...new Set(arr)]



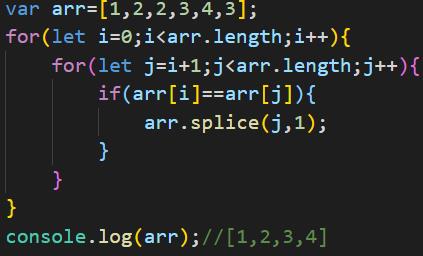
方法2：遍历数组+indexOf法 新建一个数组，循环数组，如果元素已存在新数组中则跳过



方法3：相邻比较法 外层遍历，内层比较，如果和后一个元素相同则跳过后一个元素 可能会导致乱序

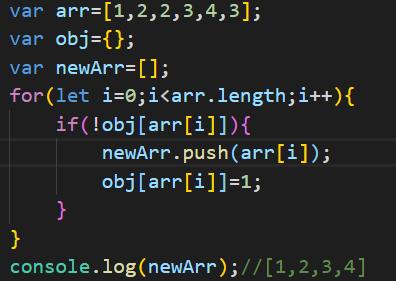


方法4：splice 相邻元素比较，如果相同去掉



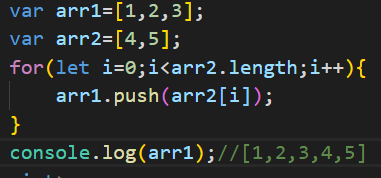
方法5：排序去重 先排序，然后比较相邻元素

方法6：对象键值法 把数组元素添加为对象的一个属性，判断属性是否存在

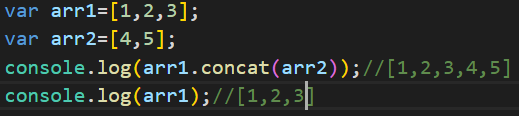


**3.合并数组**

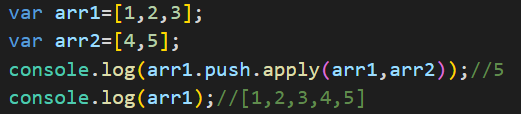
方法1：for循环



方法2：concat



方法3：apply



**4.var声明变量**

引擎负责整个代码的编译以及运行，编译器则负责词法分析、语法分析、代码生成等工作而作用域则如我们熟知的一样，负责维护所有的标识符（变量）。

var a=2;实际分为2步：var a;和a=2;

编译器会询问作用域是否已经存在叫a的变量了，如果不存在，则招呼作用域声明一个新的变量a，若已经存在，则忽略var 继续向下编译，这时a = 2被编译成可执行的代码供引擎使用。

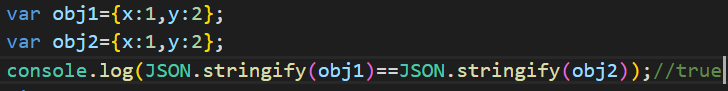
询问在当前的作用域下是否有变量a，若存在，则将a赋值为2（由于第一步编译器忽略了重复声明的var，且作用域中已经有a，所以重复声明会发生值得覆盖而并不会报错）。若不存在，则顺着作用域链向上查找，若最终找到了变量a则将其赋值2，若没有找到，则招呼作用域声明一个变量a并赋值为2

**5.var重复声明**

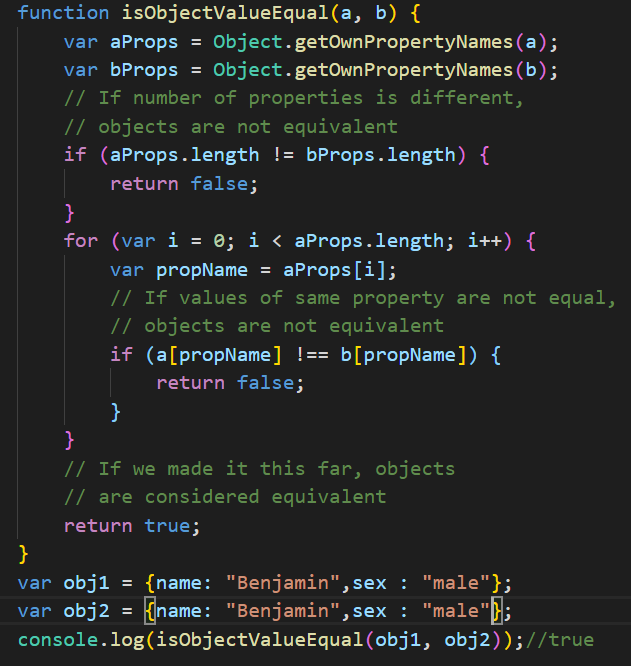
如果是变量，则忽略。如果是函数，则覆盖前面的。

**6.判断对象相等**

方法1：JSON.stringify() 如果对象的属性的顺序不同则不相等



方法2：遍历属性 有3种情况不行：属性值是对象/NaN/undefined



方法3：js库 underscore和Lodash有个方法：\_.isEqual()



**7.flex布局**

弹性盒子 横向是主轴，纵向是交叉轴

容器的属性：

flex-direction项目的排列方向

clipboard.png

flex-wrap如何换行

clipboard.png

flex-flow是flex-direction和flex-wrap的合并

clipboard.png

justify-content 项目在主轴上的排列方式

clipboard.png

align-items 项目在交叉轴上如何对齐

clipboard.png

align-content 多根轴线的对齐方式，如果只有一根轴线会忽略

clipboard.png

项目的属性：

order 排列顺序 越小越靠前

flex-grow 项目的放大比例 如果所有项目的flex-grow属性都为1，则它们将等分剩余空间（如果有的话）。如果一个项目的flex-grow属性为2，其他项目都为1，则前者占据的剩余空间将比其他项多一倍。

flex-shrink 项目的缩小比例

flex-basis 在分配多余空间之前，项目占据的主轴空间（main size）。浏览器根据这个属性，计算主轴是否有多余空间。它的默认值为auto，即项目的本来大小。

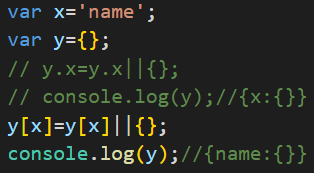
flex属性是flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto。后两个属性可选。

align-self属性允许单个项目有与其他项目不一样的对齐方式，可覆盖align-items属性。默认值为auto，表示继承父元素的align-items属性，如果没有父元素，则等同于stretch。

clipboard.png

flex:1;代表元素占1/n

**8.给对象添加属性**



原本不存在y[x]，这个等式可以给y添加属性。

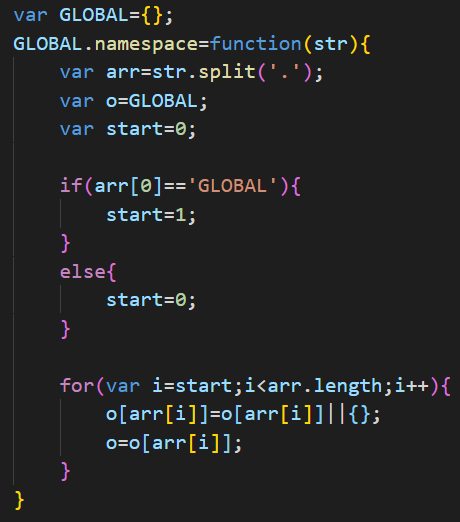
**9.避免js中的变量名冲突**

避免全局变量的泛滥

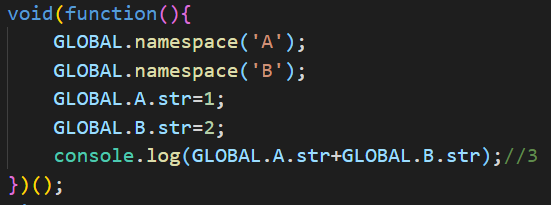
为代码添加必要的注释

合理使用命名空间

定义命名空间：



使用命名空间：

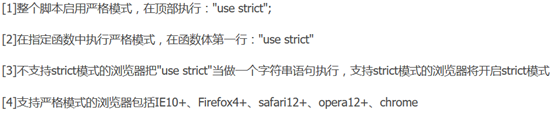


**10.严格模式**

作用：

clipboard.png

如何使用严格模式：



严格模式的限制：

变量：

不允许意外的创建全局变量，如声明变量时忽略var

不能对变量调用delete操作符

对象：

不能为只读属性赋值

不能为不可配置的属性使用delete操作符

函数：

参数必须唯一

修改形参不会影响argument

不允许使用arguments.callee和arguments.caller

注：

arguments是类数组，保存实参信息，函数被调用时才被创建，函数未被调用时为null,如何使用：arguments[0]

caller 执行调用该函数的对象 fn.caller

callee 指向被调用函数本身 arguments.callee

不允许在eval()中创建变量或函数

不允许使用eval和arguments作标识符，也不允许读写他们的值

不允许this的值为null或undefined

不允许使用with语句

不允许使用八进制字面量

**11.for循环**

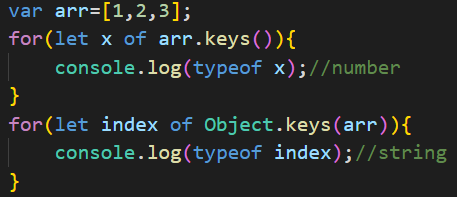
for(;;)麻烦

forEach()不能中途跳出循环

for-in遍历键名 也遍历手动添加的键和原型链上的键，以任意顺序遍历键名，主要遍历对象

for-of 遍历键值 可以跳出循环，提供了遍历所有数据的统一接口

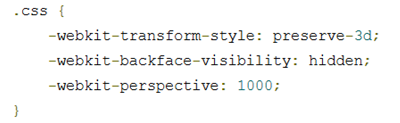
es6提供了3个新方法遍历数组：keys() values() entries() 也可以遍历对象



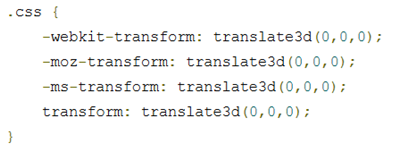
**12.消除transition闪屏**

过渡动画(在没有启动硬件加速的情况下)会出现抖动现象，解决办法：

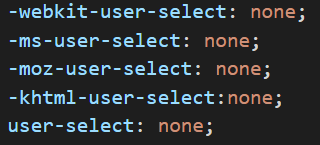
通过改变视角启动硬件加速：



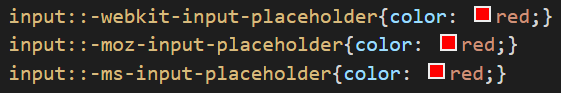
另一种方法：



**13.让图文不可复制**



**14.改变placeholder的颜色**



**15.求数组最大值**

es6拓展运算符

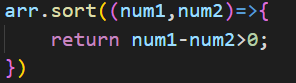
clipboard.png

es5 apply

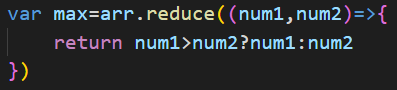
clipboard.png

for循环

sort



reduce



**16.Promise**

Promise对象代表一个异步操作，有3种状态：Pending进行中、Fulfilled已成功、Rejected已失败。Promise的状态改变只有2种可能，从进行中到成功和从进行中到失败，一旦状态改变就不会再变。Promise一旦新建就会立即执行，then方法指定的回调函数将在当前脚本中的所有同步任务都执行完成后再执行。

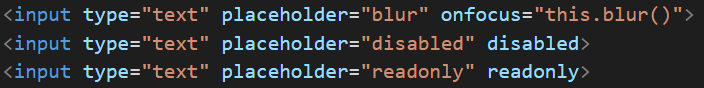
Promise.prototype.then()

Promise实例有then方法，即then方法是定义在原型对象Promise.prototype上的，作用是为Promise实例添加状态改变时的回调函数

Promise.prototype.catch()

Promise.prototype.catch()是.then(null,rejection)的别名，用于指定发生错误时的回调函数。最好用catch，不要用then中reject的回调函数。

**17.让input不能获取焦点和禁用**



disabled时，表单提交时不能被提交

clipboard.png

**18.数字和字符位数**

数字类型在内部被表示为64位浮点数

1===1.0

字符是16位的，即2个字节

**19.文字在div底部**

line-height

position:absoluted

**20.hr边缘不光滑**

clipboard.png

或者用after伪元素代替

**21.伪类和伪元素**

伪元素用::，伪类用:

都是css选择器

eg:

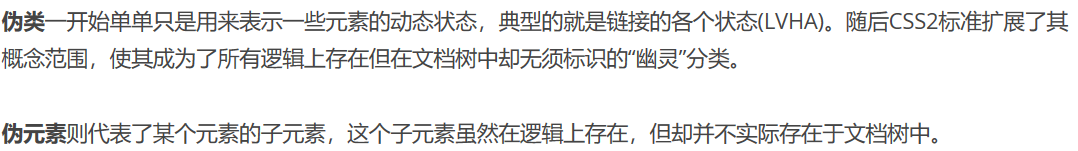
::first-line

如果要描述一个段落的第一行而不用伪元素，要用span标签把第一行包起来，然后加类名。

:first-child

如果要描述ul的第一个子元素而不用类名，不用嵌套新的元素，直接给第一个子元素加类名

所以，两者的根部区别：是否创造了新的元素(抽象)



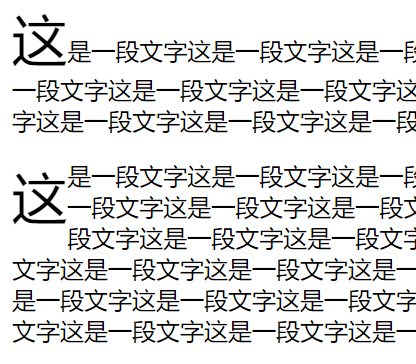
伪类没有数量限制，伪元素在一个选择器里只能出现一次。

注：a标签的伪类的正确顺序：link/visisted/hover/active hover一定要在link/visisted后面

**22.首字母变大**

单行变大 多行变大

clipboard.png



**23.私有变量和特权方法**

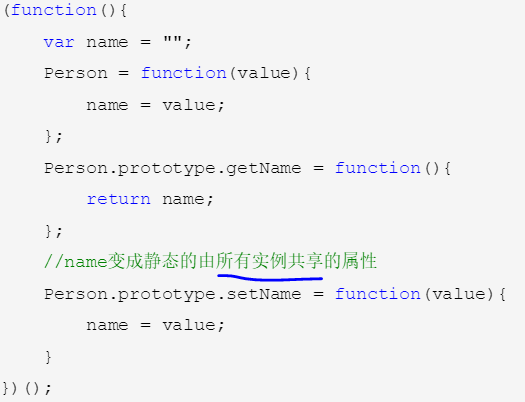
私有变量：在函数内定义的变量，不能在函数外访问到。包括参数，局部变量和函数内部定义的其他函数

特权方法：能访问私有变量的公有方法，比如闭包

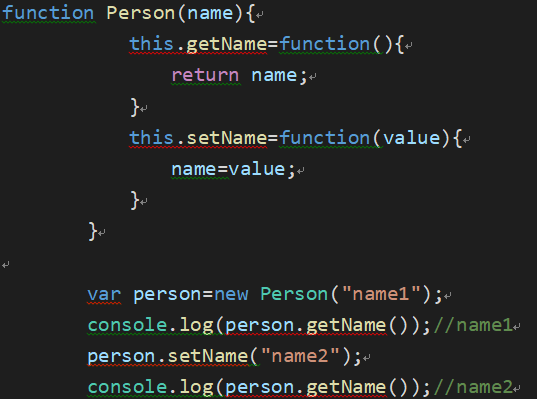
普通的私有变量：



静态私有变量：



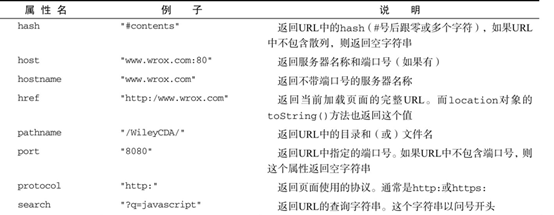
利用私有变量和特权方法，可以隐藏不应该被直接修改的数据

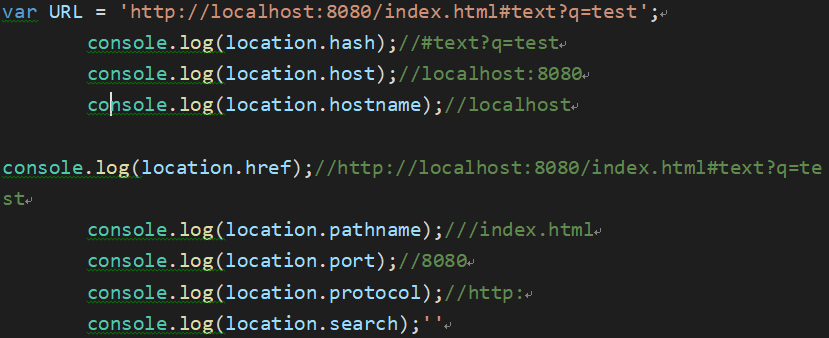


**24.location对象**

location既是window对象的属性，也是document对象的属性，还可以单独使用。

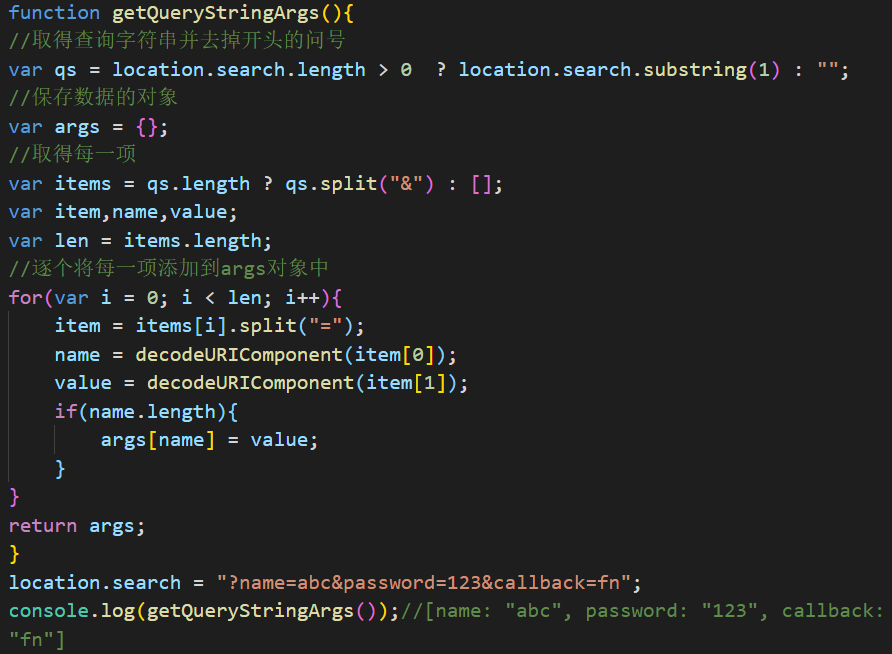
location对象的属性：





location.search只能取到?后面#前面的内容

解析查询字符串：



location的方法：

location.assign(url) 打开url，并在浏览器的历史记录中生成一条记录

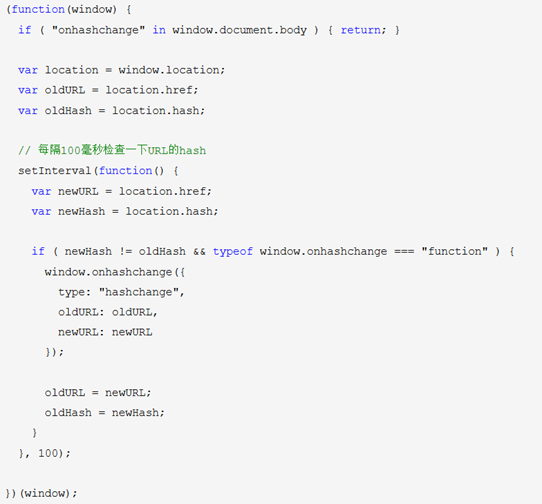
location.replace(url) 位置改变，但不生成记录

location.reload()有可能从缓存中加载 reload之后的代码可能不会执行

location.reload(true)从服务器加载

location.hasChange() h5新增，在url的参数列表(及#后的内容)改变时，通过开发人员

兼容性写法：



**25.calc**

动态计算属性值(长度)

支持加减乘除

运算符前后必须有空格

**26.table**

属性和方法：

<table>:tBodies

<tbody>:rows deleteRow(pos) insertRow(pos)

<tr>:cells deleteCell(pos) insertCell(pos)

**27.hasLayout**

ie7的特有属性，是一种只读属性，true/false。true代表元素有自己布局，false代表布局继承父元素

**28.margin**

没有继承性

ie6下双边距问题

产生原因：block元素+float+margin

解决办法：display:inline/去掉float

margin重叠问题

水平margin不会重叠

margin最大值|最大正margin-绝对值最大的负margin|0-绝对值最大的负margin

元素顶边界与前元素底边界叠加|元素顶边界与父元素底边界叠加|

解决方法：

外层padding

透明边框

绝对定位

外层overflow:hidden

内层float:left;display:inline;

外层zoom:1 zoom是ie的专有属性，设置对象的缩放比例

margin的应用

雪碧图/精灵图 背景定位效果

流动性布局

选项卡边框线

左右分隔线

clipboard.png

clipboard.png

图片与文字对齐问题

默认是居底对齐，vertical-align:top对ie的兼容性不够好，可以使用margin

**29.display:inline-block产生间隙**

把li设置为display:inline-block之后，li之间有8px的间隔

原因：浏览器把inline元素之间的空白字符渲染成一个空格

解决：

为li设置float:left

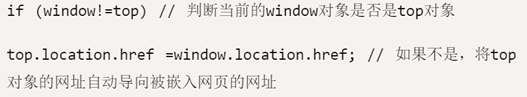
将所有li写在一行

注释掉空白

将ul内的font-size设置为0 需要重新设置其他字符尺寸，而且Safari浏览器依然有空白

ul设置letter-spacing:-8px li设置letter-spacing:normal

**30.防止网页被嵌入代码**



**31.js获取元素的位置和尺寸**

网页大小是一张网页的全面积，包括滚动条隐藏的

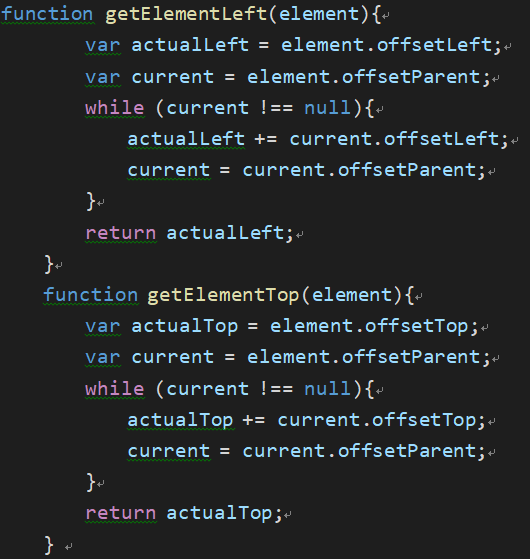
浏览器窗口大小就是用户能看到的大小，也叫视口(viewport)

获取网页大小：

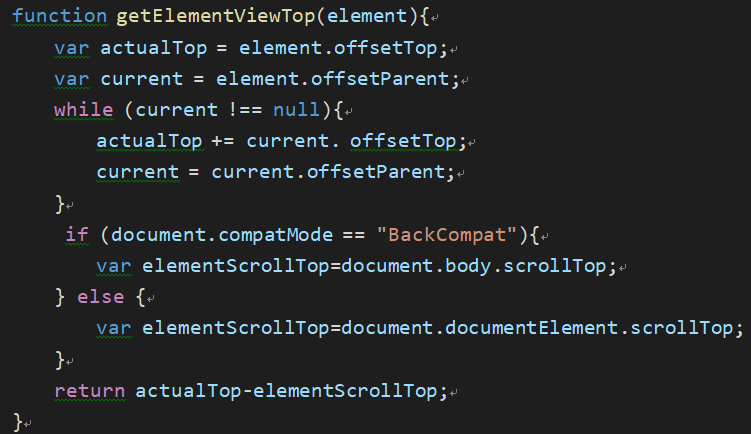
clientWidth/clientHeight 不包括border和滚动条

scrollWidth/scrollHeight 包括滚动条 不包括边框

获取元素的绝对位置 相对于整个网页左上角



获取元素的相对位置 相对于浏览器窗口左上角

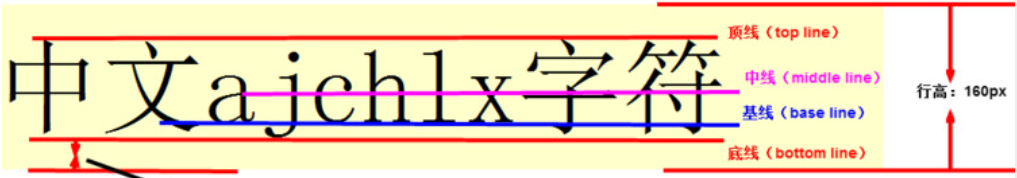


**32.input元素获得焦点是会出现蓝色框**

如果消失：outline:none

**33.复选框/单选框与文字对齐问题**

默认是vertical-align:baseline 改成bottom

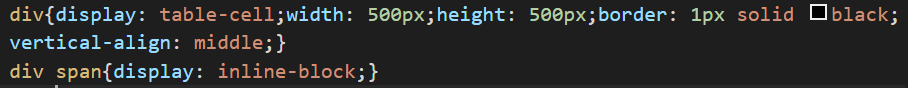


**34.水平垂直居中**

单行文本：水平居中 text-align:center 垂直居中line-height为父元素高度

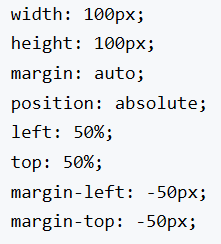
多行文本：父元素高度不固定时，通过设置padding填充上下

父元素高度固定时，通过dispaly:table和vertical-align ie7不支持(position)

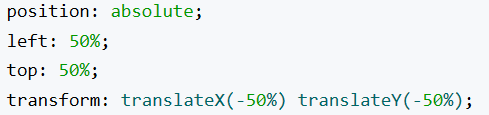


div垂直居中：

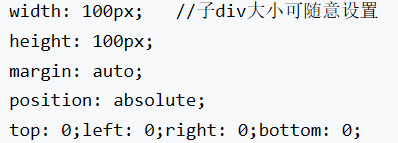
绝对定位+margin



translate



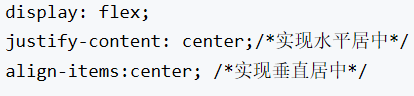
绝对定位



vertical-align

clipboard.png

flex



**35.大小不固定图片水平垂直居中**

透明图片+背景定位

使用一个透明的gif图片做覆盖层，高宽拉伸至需要，设置背景图

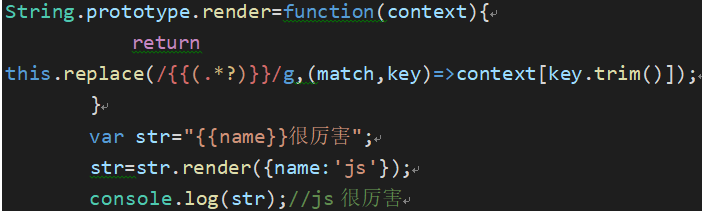
table-cell

display:table-cell;vertical-align:middle;text-align-center;

a

用a标签包裹图片，让a垂直水平居中

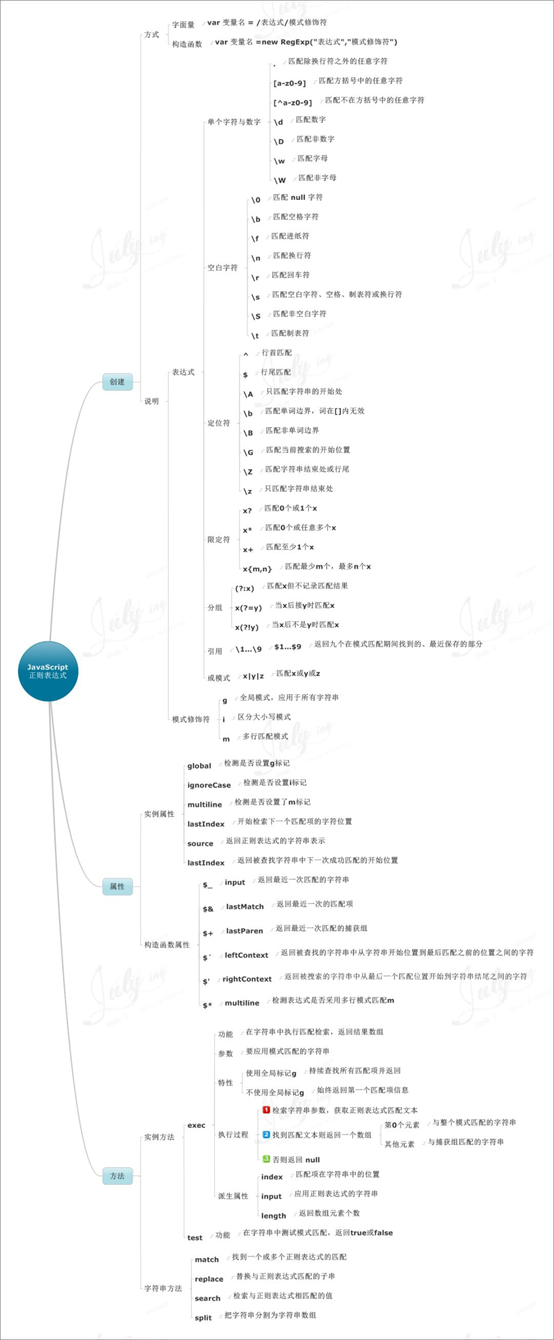
**36.模仿模板字符串**



**37.静态作用域和动态作用域**

静态作用域就是在定义时确定的作用域，动态作用域是运行时确定的作用域，静态作用域也叫词法作用域。js采用静态作用域

**38.正则**



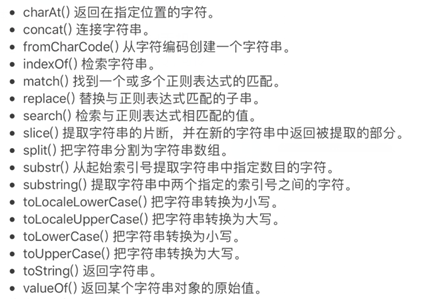
**39.绑定事件方法**

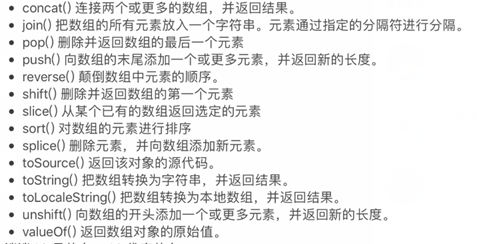
嵌入DOM

直接绑定 eg. onclick

事件监听 addEventListener/attachEvent

**40.字符串和数组常用函数**





**41.深拷贝和浅拷贝**

基本类型是按值传递，对象按引用传递。浅拷贝只是拷贝对象的引用，还会影响原来的对象。

浅拷贝：= Object.assign()

深拷贝：手动复制、如果只有一层可以用Object.assign()、转成JSON再转回来、递归拷贝、Object.create()、jquery的$.extend(true,{},obj)、lodash的\_.cloneDeep

**42.DOM一致性检测**

DOM分为3个级别：DOM1 DOM2 DOM3 。DOM分类：DOM核心、DOMXML、DOM HTML。需要检测浏览器实现了DOM的哪些部分。

var hasCoreDom1 = document.implementation.hasFeature('Core','1.0');

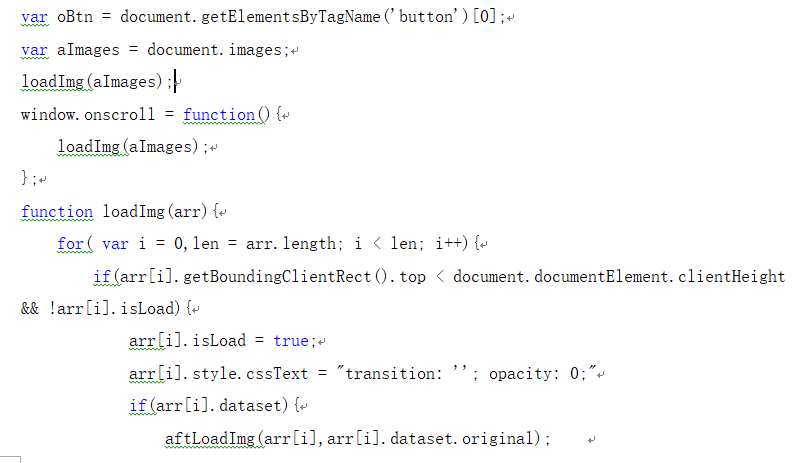
这个方法接受两个参数:要检测的DOM功能的名称及版本号。如果浏览器支持给定名称和版本的功能，则该方法返回true。

**43.预加载和懒加载**

预加载：提前加载图片，用户浏览时获得更好的体验。方法：使用页面无用元素的背景图片进行预加载。或者通过new Image()或document.createElement('img')创建img标签，通过src赋值语句来加载图片。利用图像的onload事件可以确切地知道图片是否被真正加载，并可能在后续执行一系列对图片的操作功能，如获取当前图片的实际宽高及索引等。src赋值必须放在onload后面。

懒加载：如果图片过多，页面显示的慢，浪费用户流量。方法：先把img的src指向空或者一个小图片，图片真实的地址存储在img一个自定义的属性里,等到此图片出现在视野范围内了，获取img元素，把data-src里的值赋给src。offsetTop<scrollTop+clientTop

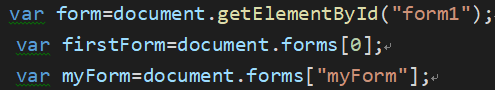
可视区显示图片：





**44.表单**

获取表单



避免重复提交表单方法：在第一次提交表单后禁用按钮，利用onSubmit事件处理程序取消后续的表单操作

获取表单元素个数：oForm.elements.length

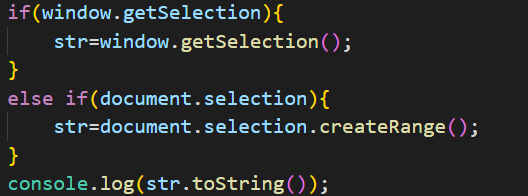
获取表单元素：oForm.elements[0] oForms.elements['color']

自动获取焦点：<input type="text" autofocus />

文本框：

clipboard.png

select() 选中所有文本 触发window.getSelection事件



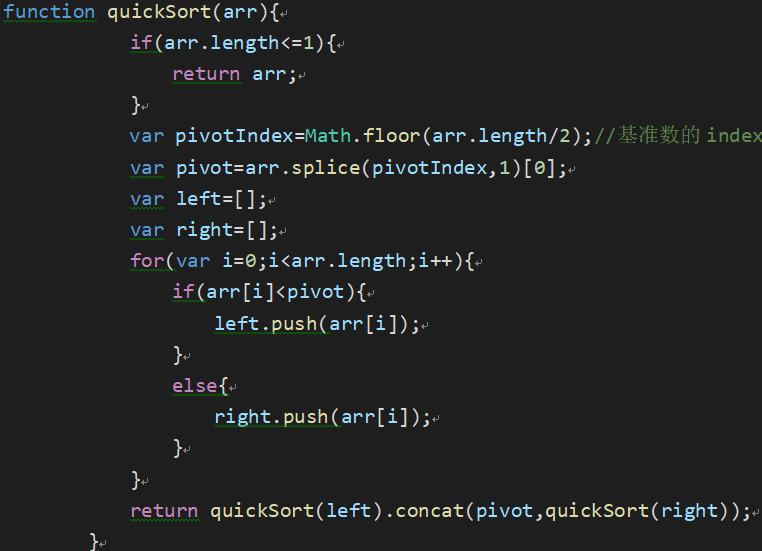
如果要获取选中的文本，可以把上面的代码放在onmouseup事件里

**45.快速排序**

从数据集中选择一个元素作为基准(pivot)

所有小于基准的元素都移到基准的左边，所有大于基准的元素都移到基准的右边

对于基准左右两边的2个子集，不断重复一二步



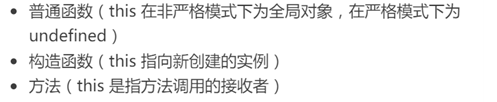
**46.字符串替换**

str.replace(被替换,替换)

**47.this**

分类：在函数内部、在函数外部、eval()中

在函数内部：



在普通函数中，可以通过call、apply、bind进行函数调用，明确this的指向

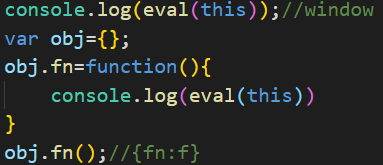
在函数外部：即顶级作用域中

在浏览器环境下，this===window 在node中，this===global

在eval中

直接调用eval()，this指向全局对象

间接调用，this与eval中的环境保持一致

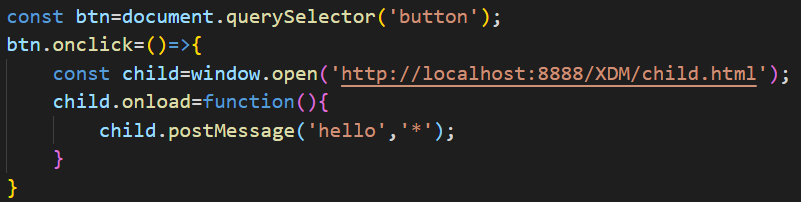


**48.跨文档消息传递**

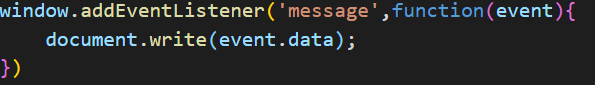
同源策略是指浏览器只允许相同协议、域名、端口号的地址进行通信。

h5新特性跨文档消息传递XDM,可以在不同域的页面间传递消息。

parent.html: 要在本地服务器中打开



child.html:



postMessage()接收2个参数：一条消息和一个表示消息来自哪个域的字符串，第二个参数如果为\*，代表可以把消息发送给来自任何域的文档。

接收到XDM消息时，会触发window对象的message事件。以异步形式触发。message事件的回调函数接收3个参数：传入的数据data 发送消息的文档所在的域origin 发送消息的文档的window对象的代理source

window.parent 父窗口

window.self 当前自身窗口

window.top 顶层窗口

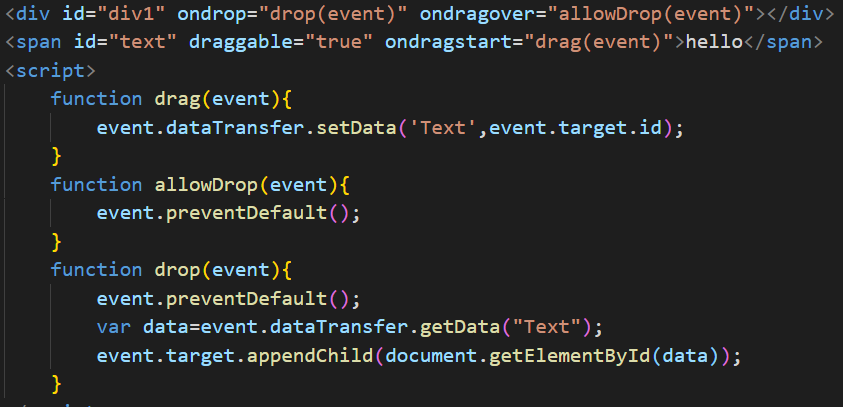
window.opener 打开当前窗口的窗口

**49.拖放**

图像和链接可以拖动，文本在选中的时候可以拖动

h5新增了一个属性-draggable true可以拖动 false不能拖动

拖动某元素时，依次触发dragstart drag dragend 。当某个元素被拖到一个有效的放置目标时，依次触发dargenter dragover dragleave/drop



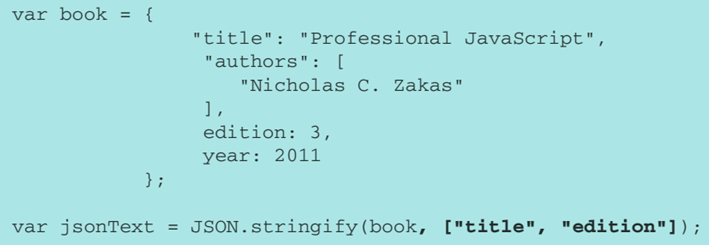
**50.JSON**

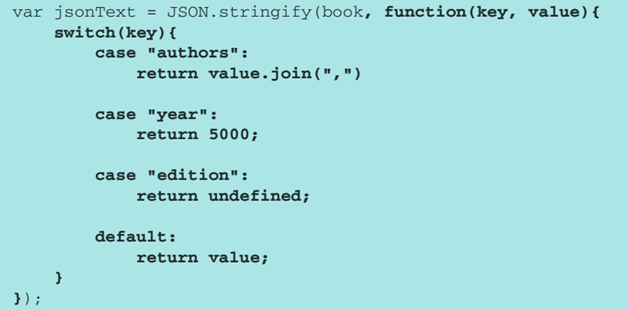
json是一种数据格式

json的属性值必须有双引号

JSON.stringify()将对象变成json字符串，JSON.parse()将json变成对象

stringify还可以接收2个参数，第一个参数是过滤器-数组/函数，第二个参数是缩进的空格数。





toJSON()可以将Date对象转换成json字符串

**51.序列化**

序列化就是将对象转换成字符串

对象序列化

将对象转换成(json)字符串

表单序列化

jquery中：serialize() serializeArray()

clipboard.png

clipboard.png

**52.js的缺陷**

开发只用了10天，设计的初衷是为了解决简单的网页互动。没有先例，结合了函数式编程和面向对象编程。唯一使用prototype继承模型的语言。过早的标准化，js推出一年半之后标准就出现了。

不适合开发大型程序

没有命名空间，难模块化，允许重复定义，不利用模块加载

非常小的标准库

标准库里的函数只能完成一些基本的操作

null和undefined

null代表对象为空，undefined表示变量未定义，NaN是一种特殊的数字

null==undefined true

null===undefined false

全局变量难以控制，任何一个函数内部都可能产生全局变量

解释器为语句在行尾自动加分号

+遇字符串变连接符

NaN 不等于任何值(包括自身)

数组和函数的区分

数组：

Array.isArray(arr)

obj instanceof Array

Object.propotype.toString.call(obj)===’[object Array]’

obj.constructor===Array

函数：

Object.propotype.toString.call(obj)===’[object Function]’

==和===

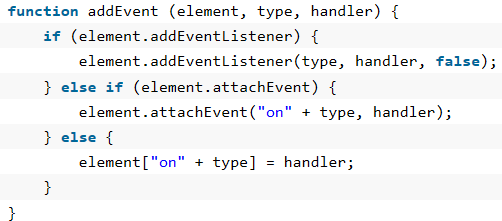
==比较时会自动转换数据类型

基本类型的包装对象

Boolean Number String

**53.惰性载入函数**

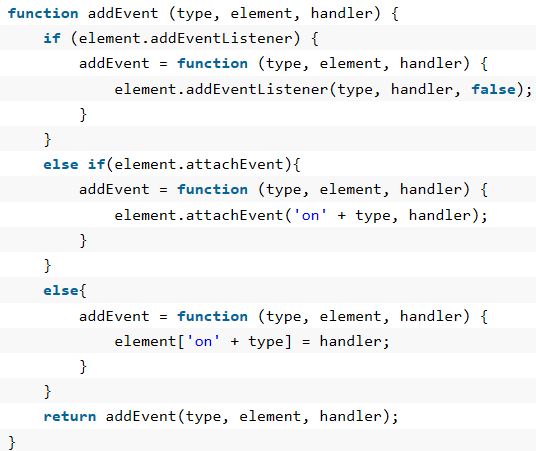
为了解决浏览器的兼容性问题，有时候需要使用if-else语句检测浏览器特性



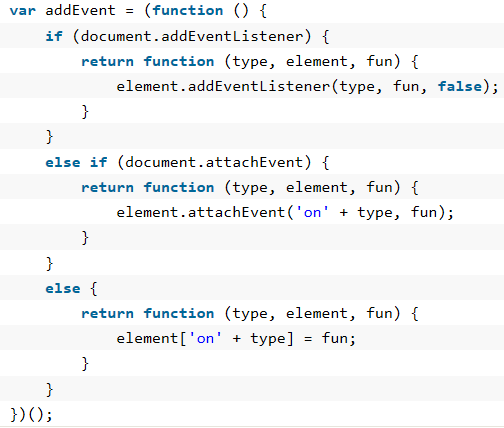
每次执行addEvent的时候都要在if-else里进行判断，惰性载入即每个分支只执行一次

解决办法：

在函数第一次调用时，重写函数，覆盖原来的函数



在声明函数时就指定适当的函数



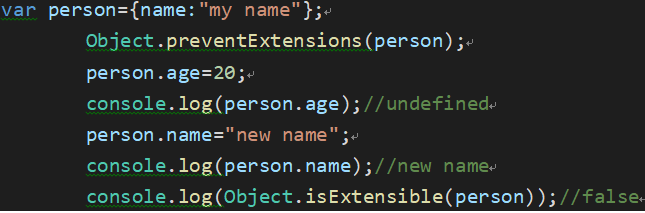
**54.函数柯里化**

把接受多个参数的函数变换成接收单一参数的函数，并且返回接受余下的参数而且返回结果的新函数

**55.防篡改对象**

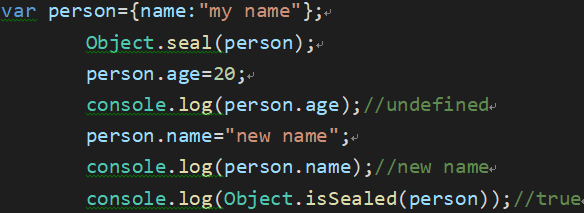
不可扩展对象

不能添加属性，可以删除和修改已经存在的属性



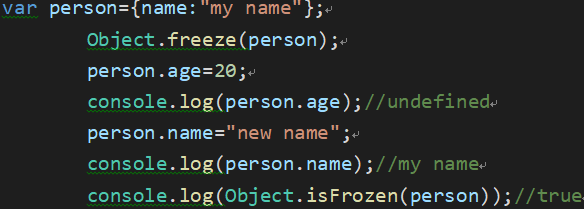
密封对象

不能添加和删除属性，可以修改属性值

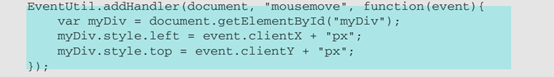


冻结对象

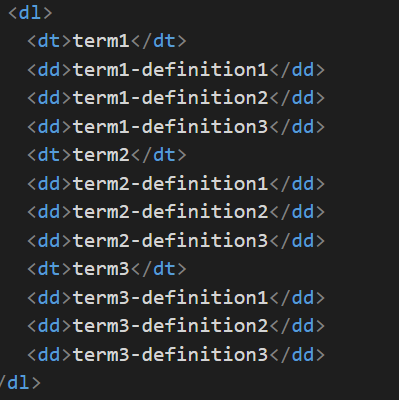
不能增加和删除属性，也不能修改属性值

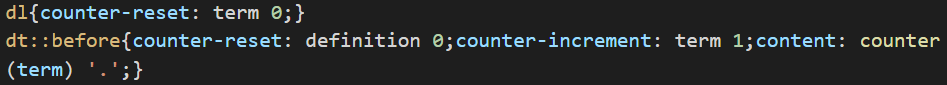


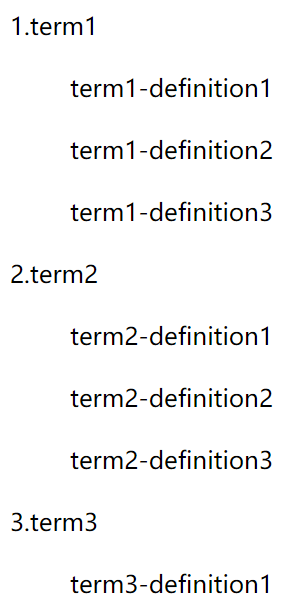
**56.鼠标拖尾**



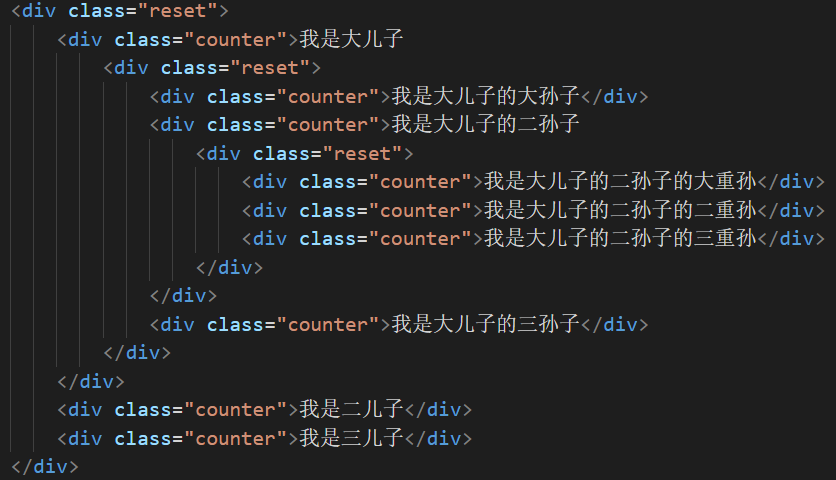
**57.css计数器counter**

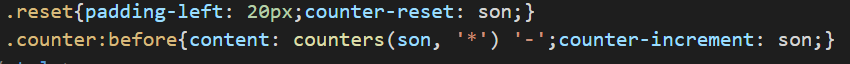


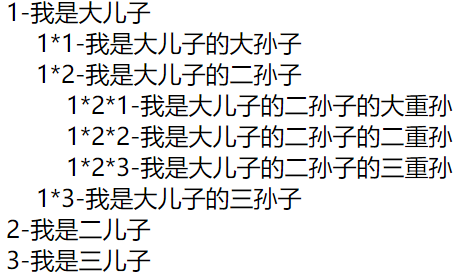


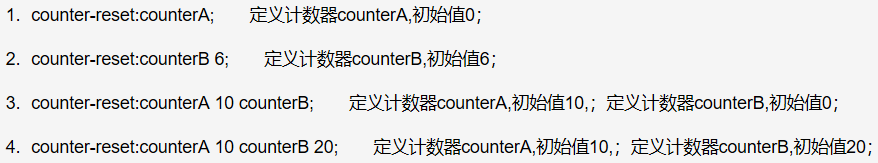


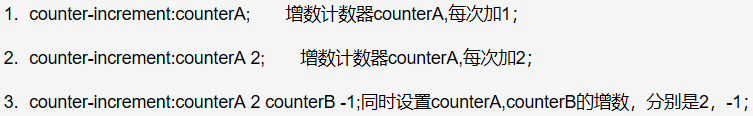
eg2.

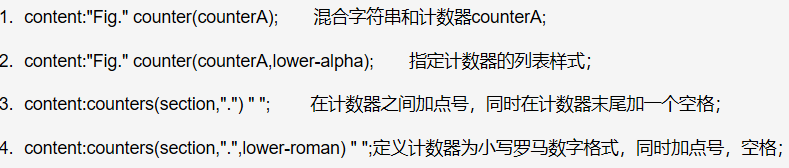


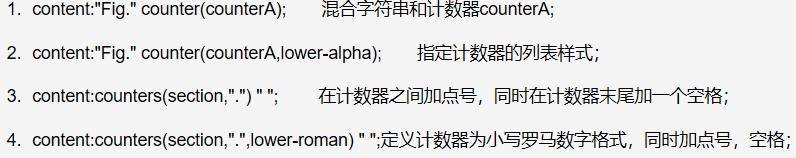






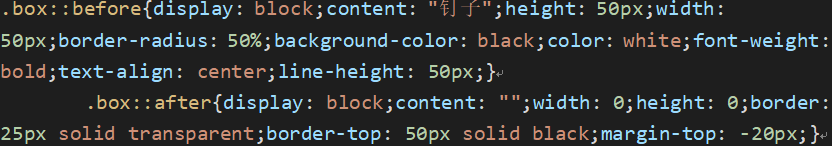




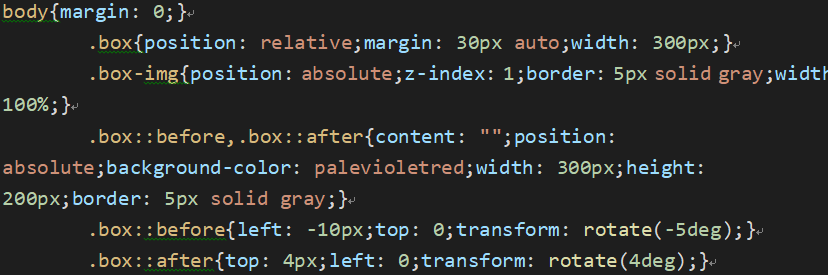


**58.利用伪元素做CSS效果**

钉子：



图片叠加效果：



**59.能转换为布尔值false的只有6个**

0  “”  null  undefined false NaN

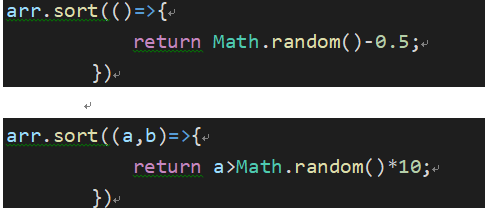
**60.undefined**

undefined不是关键字，能被赋值，所以是不可靠的

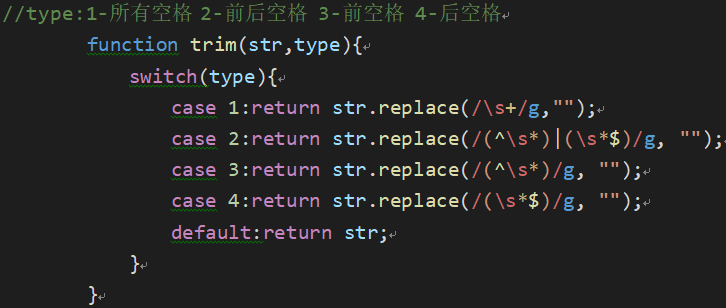
所以应该用void 0 或void(0)代替undefined

**61.创建一个随机数组或打乱数组**

借助Math.random()



**62.去除前后空格**



**63.作用域和闭包**

作用域是什么

作用域是根据名称查找变量的一套规则。

如果查找的目的是对变量进行赋值，那么就会使用LHS赋值。如果目的是获取变量的值，就会使用RHS查询。

编译分3个阶段：词法分析(将代码拆分为词法单元)—语法分析(转换成一个抽象语法树AST)—代码生成(将AST转换成可执行代码即机器语言)。

任何js片段在执行前都会进行编译，大部分情况下编译发生在代码执行前几微秒。

引擎从头到尾负责整个js程序的编译及执行过程。编译器负责语法分析及代码生成，作用域负责收集并维护由所有声明的标识符组成的一系列查询，并实施一套非常严格的规则，确定当前执行的代码对这些标识符的访问权限。

var a=2;首先编译器会在当前作用域中声明一个变量(如果之前没有声明过)，然后在运行时，引擎会在作用域中查找该变量，如果能够找到就会对它赋值。

当一个块或者函数嵌套在另一个块或者函数中，就发生了作用域的嵌套。遍历嵌套作用域链的规则：引擎从当前的执行作用域中开始查找变量，如果找不到就向上一级继续查找，当抵达最外层的全局作用域时，无论找没找到，查询都会停止。

如果RHS查询在所有嵌套的作用域中遍寻不到所需的变量，引擎就会抛出ReferenceError异常。当引擎执行LHS查询时，如果在全局作用域中也找不到目标变量，全局作用域就中就会创建一个具有该名称的变量(非严格模式下)，严格模式下会产生ReferenceError异常。

ReferenceError同作用域判别失败有关，TypeError代表作用域判别成功了但对结果的操作是不合法的。

词法作用域

词法作用域意味着作用域是由书写代码时函数声明的位置来决定的。

2个机制可以欺骗词法作用域：eval()和with

这2个机制的副作用：引擎无法在编译时对作用域查找进行优化从而导致运行变慢，在严格模式下失效。

作用域2种：词法作用域和动态作用域

词法作用域就是定义在词法阶段的作用域。

没有任何函数可以同时出现在2个父级函数中

在多层嵌套的作用域中可以定义同名标识符，这是遮蔽效应，即内部标识符遮蔽了外部的标识符。

词法作用域只会查找一级标识符，即foo.bar.baz词法作用域先查找foo标识符，然后对象属性访问规则查找bar…

with可能会造成全局作用域变量污染，隐式声明全局变量。

函数作用域和块作用域

函数的作用域是指，属于这个函数的全部变量都可以在整个函数的范围内使用及复用。

从所写的代码中挑选任意片段，然后用函数声明对它进行包装，实际上就是把这些代码隐藏起来了。遵循最小特权原则(最小授权或最小暴露)。避免同名标识符的冲突。

避免变量冲突的方法：命名空间、模块管理

如果function是声明中的第一个词，那么就是一个函数声明，否则就是一个函数表达式。

匿名函数

arguments.callee调用函数自身

IIFE代表立即执行函数表达式。进阶用法：把它们当作函数调用并传参数进去。还可以解决undfined标识符的默认值被错误覆盖导致的异常。还可以倒置代码的运行顺序，把需要运行的函数放在第二位，在IIFE执行之后当参数传递进去。

块级作用域：with try-catch let const

如果用let代替var则需要在代码重构中付出额外的精力。

提升

包括变量和函数在内的所有声明都会在任何代码被执行前首先被处理。

var a=2;àvar a;a=2;第一个定义声明是在编译阶段进行的，第二个赋值声明会被留在原地等待执行。

每个作用域都会进行提升操作。变量只能提升到当前作用域的最前面。

函数声明会提升，但函数表达式不会提升。

函数首先被提升，然后才是变量。

作用域闭包

当函数可以记住并访问所在的词法作用域，即使函数是在当前词法作用域之外执行，这时就产生了闭包。

模块2个主要特征：为创建内部作用域而调用了一个包装函数，包装函数的返回值必须至少包括一个对内部函数的引用。

一般情况下，函数执行后内部作用域会被销毁。但闭包的话，这个内部作用域不会被销毁。

IIFE严格来讲不是闭包，因为它不是在它本身的词法作用域外执行的，是在定义时所在的定义域内执行的。里面的变量是通过普通的词法作用域查找而非闭包被发现的。

闭包循环

for(var i=0;i<5;i++){

            (function(j){

                setTimeout(function(){

                    console.log(j);

                },1000);

            })(i)

        }

        for(var i=0;i<5;i++){

            let j=i;

            setTimeout(function timer(){

                console.log(j);

            },1000)

        }

        for(let i=0;i<5;i++){

            setTimeout(function timer(){

                console.log(i);

            },1000)

        }

**64.精度丢失**

clipboard.png

es6新增属性极小的常量 Number.EPSILON 判断是否为可忽略的误差

只要差值小于最小数就说明正确

Number.MIN\_VALUE ECMAScript所能表示的最小数值

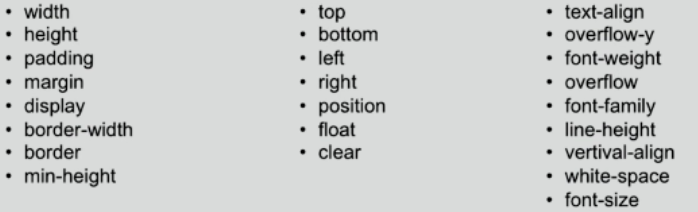
**65.回流与重绘**

回流：当render tree 的一部分或全部的元素因改变了自身的宽高，布局，显示或隐藏，或者元素内部的文字结构发生变化 导致需要重新构建页面的时候，回流就产生了

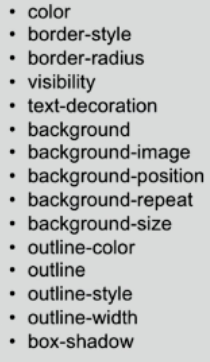
重绘：当一个元素自身的宽高，布局，及显示或隐藏没有改变，而只是改变了元素的外观风格的时候，就会产生重绘。例如你改变了元素的background-color....

回流必定触发重绘，重绘不一定触发回流

触发回流：



触发重绘：



**66.CSS制作图像**

<http://www.cnblogs.com/xiaohuochai/p/5028101.html>

**67.this**

this指向调用它的对象

函数内丢失this的解决方法

var that=this

箭头函数

bind

this绑定的4条规则

由new调用 绑定到新创建的对象

由call apply bind调用 绑定到指定对象

由上下文对象调用 绑定到那个上下文对象

默认：严格模式下绑定到undefined 非严格模式下绑定到全局对象

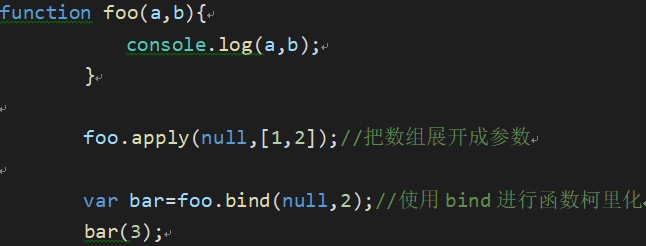
箭头函数会继承外层函数的this绑定，像es5的self=this机制

参数传递是一种隐式赋值 其和赋值容易导致丢失绑定对象

call函数中，如果传入了一个原始值当作this的绑定对象，这个原始值会被转换为它的对象形式，称为装箱。string->new String()

如果把null undefined作为this的绑定对象传入call apply bind 这些值在调用时会被忽略，实际应用的是默认规则。更安全的是传入一个特殊的对象，把this绑定到这个对象上不会有影响，即MDZ-非军事区，一个空的非委托对象。

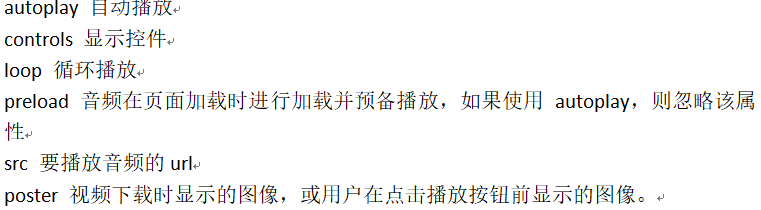
用Object.create(null)创建



this可能会导致隐形变量

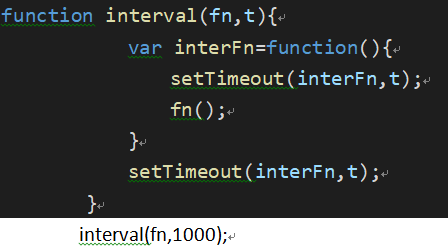


**68.audio/video**

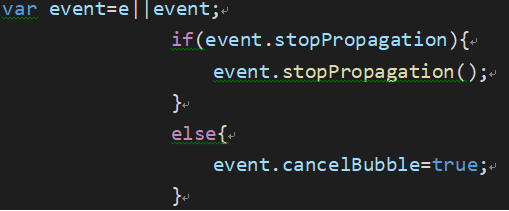


audio.duration 直接获取是NaN 所以要把获取放在oncanplay事件中

**69.利用setTimeout代替setInterval**

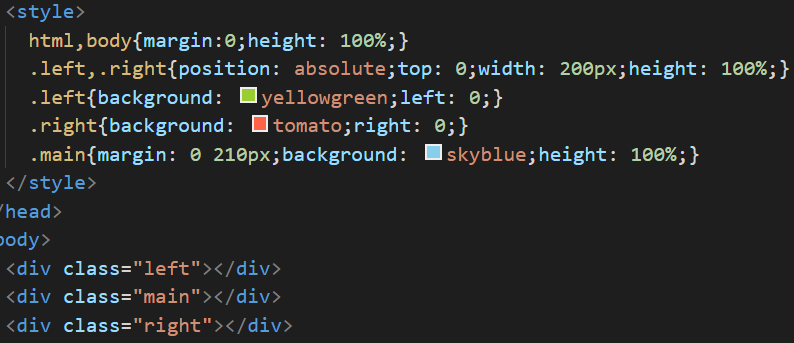


**70.取消冒泡**



**71.三列布局 中间自适应-两边固定**

绝对定位



包裹+margin负值



浮动

