W3c：

1. 只能在 HTML 输出中使用 document.write。如果您在文档加载后使用该方法，会覆盖整个文档。

例：<http://blog.csdn.net/a472770699/article/details/49870975>

1. JavaScript 对大小写敏感。
2. JavaScript 中的所有事物都是对象：字符串、数字、数组、日期，等等。在 JavaScript 中，对象是拥有属性和方法的数据。
3. 属性是与对象相关的值。方法是能够在对象上执行的动作。
4. 常用method:.indexOf(),.search(),.replace(),.toUpperCase(),.shift()移除第一个,.unshift()插入到第一个,.pop(),.push()末尾

常用属性：length,

1. js字符串不可变，无效 必须对变量重新赋值
2. document.getElementById(‘’)id查找（document为全局查找）

.getElementsByTagName(‘’)标签查找

1. 改变html内容：document.getElementById(id).innerHTML= new HTML

改变html属性：ex:document.getElementById(id).src=new src

改变 HTML 样式:ex:document.getElementById("p2").style.color="blue"

.style.visibility='hidden'" /.style.visibility='visible'"可改变可见或不可见

没有使用var关键字定义的变量，会被自动创建在全局作用域中，形成全局变量。

1. RegExp()对象 test方法返回true/false compile可添加第二参数，改变检索模式
2. Location:

location.hostname 返回 web 主机的域名

location.pathname 返回当前页面的路径和文件名

location.port 返回 web 主机的端口 （80 或 443）

location.protocol 返回所使用的 web 协议（http:// 或 https://）

location.href 属性返回当前页面的 URL。

location.assign() 方法加载新的文档。

笔记 2019寒假 菜鸟：

Js需懂概念

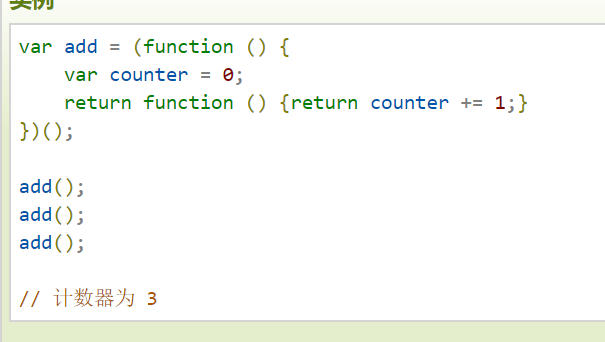
Prototype（神奇又恶心的东西）主要用来一些骚操作，记住是用来给对象加属性的就行。

变量提升与严格模式

Window、this、global区别

## 匿名函数，函数参数规则，arguments对象，使用构造函数调用函数（有意思）

变量声明时如果不使用 var 关键字，那么它就是一个全局变量，即便它在函数内定义。

闭包的一个骚用法

Window对象

JS this：

在方法中，this 为该方法所属的对象。Return this返回 [object Object]，可以通过this.a访问对象中其他属性。

单独使用，this 为全局对象。

在全局声明函数中，this为全局对象。

在全局声明函数中，在严格模式下，this 为(undefined)。

在DOM事件中，this 为接收事件的元素。

类似 call()和apply()方法可以将this引用到任何对象。

## 前面好理解，主要是最后一个，显式函数绑定。在 JavaScript 中函数也是对象，对象则有方法，apply 和 call 就是函数对象的方法。这两个方法异常强大，他们允许切换函数执行的上下文环境（context），即 this 绑定的对象。

一个很棒的例子：

var person1 = {

fullName: function() {

return this.firstName + " " + this.lastName;

}

}

var person2 = { firstName:"John", lastName: "Doe", }

person1.fullName.call(person2); // 返回 "John Doe"

补一补js函数相关：

Js可以用默认参数，没被传入实参的形参会被赋值为undefined。

另一个关于函数调用的例子，用的频率较高的函数作这样的设置，为对象执行相关操作。

var myObject, myArray;

myObject={

name: "hahaha ",

hsk: "en"};

function myFunction(a, b) {

alert(this);

return this.name +this.hsk;

}

myArray = [10, 2]

myObject = myFunction.apply(myObject, myArray);

document.getElementById("demo").innerHTML = myObject;

### Node笔记

看js竟然发现刚学的就全忘得差不多了。。果然语言这东西还是得常看常用。。唉 js特性、要点全忘了，真难受

<https://segmentfault.com/a/1190000005173218> node特色

Node首先要理解回调函数以及阻塞非阻塞、同步异步。

首先是回调函数：

我的理解就是在特定时间或操作的条件下触发的函数。回调也是一种实现异步的方式。

如：jquery 页面加载完毕　　$(function(){"要执行操作"})；

　　jquery iframe页面加载完毕 $('#iframe').load(function(){"要执行操作"})；

　　jquery ajax调用，绑定事件，setTimeout()，setInterval()

同步执行的情况，回调函数一般最后执行，异步执行的情况(ajax)，回调函数不一定会执行(**?**)。

其次是同步异步。理解很简单，主要是异步的实现方式拓展一下。

异步的实现:

①异步回调（多线程实现：异步回调函数）  
 ②消息队列（多线程实现：多线程后台轮询队列处理其中内容实现。）

③协程（时间片分隔实现）

Node中异步编程的直接体现就是回调。将最后一个参数传入闭包函数或者函数名即会当作回调函数。

事件驱动：

var events = require(‘events’);

var eventEmitter = new events.EventEmitter();

//绑定事件及时间处理

eventEmitter.on(‘eventName’,eventHandler);

//触发

eventEmitter.emit(‘eventName‘);

如果一个程序是：

Io操作

Console.log()

事件绑定

则，会先执行io操作之后的操作，因为io操作耗时，所以在异步下会后输出io操作。Io之外的按顺序执行，io之内的读写和console也是按代码顺序，只是io和与其并列的有异步

一个事件如有多个监听器，这事件触发时会被依次调用。如果监听请的参数不同，也会依次调用所有，只是多了的参数无效，少了的为undefined

Buffer类：

很神奇的一个类，没看懂。就知道了toString, toJSON等操作。。

const buf = Buffer.from('runoob', 'ascii');

console.log(buf.toString('hex'));

console.log(buf.toString('base64'));

Stream类：

主要就是读取、写入流，还有管道流、链式流等操作。链式流一般用于管道操作。

比如压缩文件：

fs.createReadStream('input.txt').pipe(zlib.createGzip()).pipe(fs.createWriteStream('input.txt.gz'));

路由：

var requestUrl = request.url;

var pathname = url.parse(requestUrl).pathname; //请求host后的path

var query = url.parse(requestUrl).query; //请求

console.log(querystring.parse(query)["a"]); //获取a参数

全局变量：

按照 ECMAScript 的定义，满足以下条 件的变量是全局变量：

* 在最外层定义的变量；
* 全局对象的属性；
* 隐式定义的变量（未定义直接赋值的变量）。

常用：console、process、\_\_filename、\_\_dirname、setTimeout/setInterval

util常用：

## util.inherits 注意：只会继承基类在原型中定义的函数

## util.inspect 对象转字符串

## util.isArray(object)

## util.isRegExp(object)