第一章和第二章总结：

JavaScript 的发展历史： 1995年，只是为了做一个验证，之后发现向服务器发送请求前可以做出详细的验证要求，可以大大减轻服务器的压力，这样就开始发展JavaScript这个脚本语言，并迅速的成长为现代浏览器的不可获取的脚本语言，增加了与后台交互和渲染页面等强大的功能，主要分为三个方面： 1，ECMAScript 规定的基本语法和规定 ；2，DOM1 修改页面中的节点 ； 3 ，DOM2 鼠标事件和点击事件等

用script标签引入js的方式用两种： 一是直接在标签内部写js代码，二是用src属性来引入外部js文件。 ； script 标签内type属性一般都是text/javascript ,若script中没有type属性时，默认的也是text/javascript ,要注意的是引入外部文件的js时，script标签内部写的js代码是无效的； asaync： 异步加载属性，只对外部脚本有效，表示立即下载脚本，但不妨碍页面的其他操作； defer:脚本延迟属性，用来延迟脚本的执行时间，知道HTML文档已经全部解析和显示之后再执行，只对外部脚本文件有效； 平时咱们最好使用src属性来引入外部js 文件，原因： 1.可维护性高，2可以统一做修改，3，利于其他属性的添加，如defer属性等

第三章总结：

ECAMScript 的规范如：

标识符：只能是字母、下划线或$符号开头，不能使关键字

JS的基本数据类型： undefined NAN 字符串 数字 布尔值

Undefined： 声明变量但是没有初始化。

NAN: 数字计算错误

字符串： toString() String(params)

数字： Number parseInt parseFloat 计算超过计算机的范围是最大值： Infinity ;

最小值： -Infinity;

布尔值： true false

操作符： typeof也属于操作符

一位操作符如： ++i ;

加：+； 减： -； 求余： %；

基本语法： if..else do..while for..in(列举出对象的属性) for

逻辑运算符： && 与； || 或；

函数： return 返回值 ； 函数中括号中的参数； arguments.length为函数实参个数，arguments.callee引用函数自身

虽然arguments对象并不是一个数组，但是访问单个参数的方式与访问数组元素的方式相同

第四章 总结

通常的变量基本都是通过数值赋值，若变量为对象时，是应用的方式，它们应用的都是同一个堆中的对象

变量为数值是固定的大小，存放到内存中的栈中，而变量为对象时，存放到堆中。

函数都是通过数值来传递参数的

函数内部如果是对象的话也是通过数值来传递的，而不是引用传递

作用域链：函数内部局部变量在函数内部没有找到的情况下，会逐级向上查找，如果没有全局也没有找到报错

只有C和C++ 语言才需要去手动启动垃圾回收机制，而JavaScript 无需手动清除，隔一段时间会自动启动垃圾回收机制。

垃圾回收机制: 标记清除： 先标记所有，用到的去除标记，之后吧所有带有标记的都清除，来解放内存。

引用计数： 引用到的就计数，没有引用到的不计数，之后清除计数为0的对象，这种清除内存的方式有一个弊端就是对象之间的属性相互引用时，就成了循环引用，永远不会被清除，

当然，有时候咱们自己可以手动来清除对象，让其作为null,之后若是垃圾回收机制启动就会把这部分置空的对象清除，来释放内存。

第五章 引用类型

创建有两种方式：一种是 new 出一个新的对象，另一种是字面量方式创建对象

对象中取值有两种方式： 一种是点的方式取值，另一种是中括号方式取值，中括号中是字符串；两者的区别是中括号的方式可以去变量的值，而点的方式不可以。

数组拼接：

concat(): 把一个数组和另一个数组拼接在一起，返回拼接好的数组；

join(): 把数组中的每一项 按照指定的分隔符拼接成字符串；

排序：

reverse(): 倒序数组 返回值倒序数组 原有数组改变

sort(): 匿名函数进行冒泡排序 b-a倒序 a-b升序

Array 类型：

push() ： 往数组最后一位添加；

pop(): 去除数组中的最后一位；

shift(): 去除数组中的第一个；

unshift(): 往数组中第一位添加；

slice(): 若只有个数字时就是从这个数字的下标截取到结束，若是两个数字时，从开始数字的下标到结束数字的下标，（注：截取的数字不包括结束数组的下标）；

splice(): 若是一个数字的时候，表示从这个数字的下标删除到最后；有两个数字，表示第一个数是开始下标，第二个数是删除几个； 若第二个数字后还有就是往里面添加的这些；

indexOf(): 返回获取项在数组中的索引；

lastIndexOf: 返回获取到数组中出现的最后一次索引；

reduce(): 从数组的第一项开始逐个遍历到最后；

reduceRight(): 从数组的最后一项开始，向前遍历到第一项；

reduce()和reduceRight的函数接受4个参数： 前一个值、当前值、项的索引和数组对象。这个函数返回的任何值都会作为第一个参数自动传给下一项。第一次迭代发生在数组的第二项，因此第一个参数是数组的第一项，第二个参数就是数组的第二项。

Date 类型

Date 有两个方法 Date.parse()和Date.UTC() ；一般我们不需要写会自动转换

如：var person = new Date(2000)和var person = new Date(Date.parse(2000)) 效果一样。

时间的中间差的计算：

//开始

var start = new Date();

//调用函数

doSomething();

//结束

var end = new Date();

时间差： end – start

Date 有很多方法来获取或设置日期，如： getTime(),setTime(), gethours(),sethours()等还有很多；

Function 类型

每一个函数都是由Function 来创建的实例；

函数没有重载，若前面已经定义了一个函数，后面定义的函数与前面的相同，就会覆盖之前的函数；

函数表达式和函数声明的不同之处：

var num = function(){} 函数表达式 ： 在JS运行时确定，并且在表达式赋值完成后，该函数才能调用。

function sum(){} 函数声明： 函数声明在JS解析时进行函数提升，因此在同一个作用域内，不管函数声明在哪里定义，该函数都可以进行调用。

函数中的参数，可以的一个或多个，可以是函数；