1. 《深入理解JavaScript》
2. 基础JavaScript
3. ECMAScript是JavaScript的官方命名。

JavaScript指的是一种编程语言，ECMAScript是这种语言的规范名字。

1. 一个等于号=，用于为变量赋值。

三个等于号===，用于比较两个值。

1. 语句“做事情”。程序其实就是一系列的语句的集合。

例：var foo; 用来声明（创建）一个变量foo

表达式产生值。通常是函数的参数，或是赋值的右边部分。例：3\*7

在JavaScript中，表达式可以用在所有需要语句的

例：foo(7,1) 整个这行就是一条语句（表达式语句），这个函数调用foo(7,1)也是一个表达式。

1. 分号：用于结束语句，而不是语句块。

函数表达式作为一个表达式时，分号出现在块之后。

1. 单行注释// 多行注释/\* \*/
2. 变量声明 var foo;

声明与赋值同时进行：var foo = 6;

为一个已有变量进行赋值：foo = 6;

1. 复合赋值运算符：如+= x + = 1;意义为x = x + 1;
2. 标识符：JavaScript中各种语法的名称。

由字母、数字、下划线\_或美元符号$表示；

不能以数字开头；区分大小写；

1. 属性：一条记录的字段。通过点(.)操作符可以读取属性。

也可通过点操作符给属性赋值，也可调用方法。

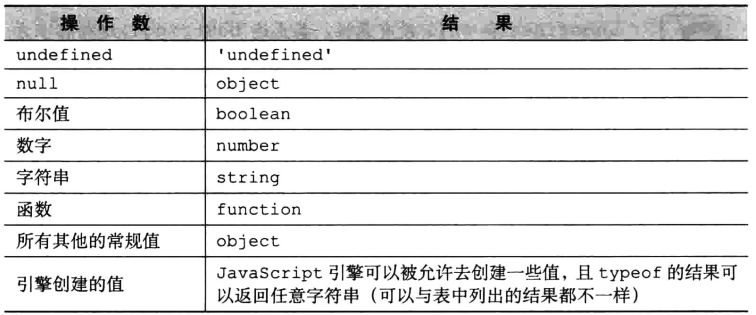
1. 原始值（基本类型值）：数字、布尔值、字符串、null和undefined

其他值都是对象。每个对象都有唯一的标识且只（严格地）等于自己。

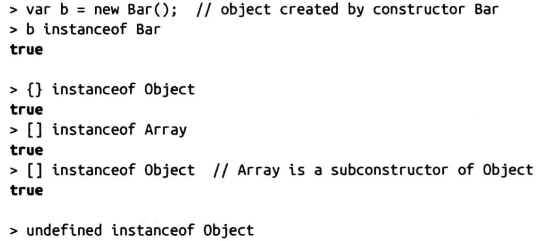
1. 原始值特点：
2. 按值进行比较
3. 不可改变。其属性不能被改变、添加或移除
4. 对象：
5. 简单对象，可以通过对象字面量来创建。
6. 数组，可以通过数组字面量来创建。
7. 正则表达式，可以通过正则表达式字面量来创建。
8. 对象特点：
9. 按引用进行比较。{}==={}为false，这是不同的空对象。
10. 默认可变，对象属性可以很自由地被改变、添加和移除。
11. 基本数据类型：真实数值存在栈区中。

对象类型：地址存在栈区，真实数值存在堆区中。

1. undefined：未被初始化的变量、丢失的参数、访问不存在的属性时，会得到。没有属性。
2. null：在用到对象的时候它表示空值（例如参数、对象链中的最后一个元素等）。没有属性。
3. typeof



1. instanceof





1. 布尔值true和false。以下运算符会产生布尔值。
2. 二元逻辑运算符：&&与，||或
3. 前置逻辑运算符：!非
4. 比较运算符：相等运算符：===、!==、==、!=

排序运算符（针对字符串及数字）：> , >= , < , <=

1. 真值与假值

false：undefined、null；布尔值：false；数字：-0、NaN；空字符串

其他的所有值（包括所有对象）都会被当成true。

被解释为false的值可被称为假值，被解释为true的值被称为真值。

1. 二元逻辑运算符
2. 与（&&）

如果第一个运算数是假值，返回它。如果第二个运算数是真值，返回第二个运算数。

1. 或（||）

如果第一个运算数是真值，返回它。如果第二个运算数是假值，返回第二个运算数。

1. 等式运算符
2. 严格相等运算符（===） 仅当两个操作数的类型相同且值相等为 true
3. 宽松相等运算符（==） 在进行比较之前，将两个操作数转换成相同的类型
4. JS中所有的数字都是浮点数。也包含特殊的数字NaN。Infinity多数情况下也是一个错误的值。Infinity比任何数都要大，-Infinity比任何一个数都要小（NaN除外）。常用作表示默认值。
5. 算数运算符：加法+ 减法- 乘法\* 除法/ 取模% 增量++变量/变量++ 减量--变量/变量-- 负值-值 转变成数字+值

全局对象Math，还会以函数方式提供更多算术运算。

1. 字符串

通过字符串字面量单引号’’或双引号””来创建。

反斜杠\用于转义字符及产生一些控制字符。可通过方括号来访问字符串中的单个字符。

字符串的length属性可以对字符的个数进行计数。

1. 字符串运算符

加号（+）连接

1. 字符串方法

slice(i)：从第i个字符到最后 slice(a,b)：从第a个字符到第b个字符之前，不包括b

toUpperCase()：转大写

toLowerCase()：转小写

indexOf(‘a’)：查询a字符的下标 若查询不到，则返回-1

1. 条件语句

If(){

}else if(){

}else{

}

switch(fruit){

case ‘banana’: //...

case ‘apple’: //...

default: //...

}

case后的“运算数”可以是任意表达式，在switch里的参数会通过===来进行比较。

1. 循环语句
2. for循环

for( , , , ) 初始化在循环开始前执行；条件在每次循环迭代前做检查；后迭代在循环迭代后执行。

1. while循环

while(){} 在条件成立的时候会持续循环。

1. do-while循环

do{

}while(condition){

}

先执行一次，判断是否满足条件，条件成立则持续循环。

1. break可以跳离循环
2. continue会开始一个新的循环迭代
3. 函数
4. 声明（3种）

普通函数声明 function 函数名(){ };

函数表达式声明 var 函数名=function(){ };

构造函数声明 var 变量名=new Function(“参数1”,”参数2”,...,”参数n”,”函数体”);

函数声明具有提升特性，它们的实体会被移动到所在作用域的开始处。

1. 变量arguments 类数组，有length、index属性

1. 传入参数太多 额外的参数会被忽略（arguments除外）

传入参数太少 丢失的参数会得到undefined这个值

1. 可选参数 x=x||0;y=y||0; ||运算符会在x为真值的时候（非null、undefined）返回x，否则，会返回第二个运算数。
2. 将arguments转换为数组

function toArray(arrayLikeObject){

return Array.prototype.slice.call(arrayLikeObject);

}

1. 异常捕获
2. 变量作用域
3. 全局作用域 全局作用域中无法访问到局部作用域中的变量
4. 局部作用域 局部作用域中可以访问到全局作用域中的变量
5. 块作用域 任何一对花括号 { } 中的语句集都属于一个块，在这之中定义的所有变量在代码块外都是不可见的。
6. 声明提升
7. 函数声明整体提前
8. 变量声明提前，赋值留在原地
9. 函数首先被提升，之后是变量
10. 函数声明有冲突，会覆盖；变量声明有冲突，会忽略
11. 闭包
12. 定义：函数及其它所连接的周围作用域中的变量即为闭包。

是由一个函数以及创建该函数的词法环境组合而成。这个环境包含了这个闭包创建时所需访问的所有局部变量。

1. IIFE模式：立即调用函数表达式
2. 《你不知道的JavaScript（中卷）》
3. 类型

1.1内置类型

1. 内置类型：数字(number)、布尔值(boolean)、字符串(string)、空值(null)、未定义(undefined)、对象(object)、符号(symbol,ES6中新增)
2. 除对象之外，其他统称为“基本类型”。用typeof运算符来查看值的类型，返回类型的字符串值。
3. 函数是“可调用对象”，有属性length、call等属性，length属性是其声明的参数的个数。

数组也是对象。

1.2值和类型

1. 变量没有类型，值才有。变量可以随时持有任何类型的值。
2. undefined和undeclared

已在作用域中声明的变量未持有值的时候为undefined。

还没有在作用域中声明过的变量是undeclared的。

在我们试图访问”undeclared”变量报错，但JS处理中typeof都是undefined

1. typeof undeclared

通过typeof的安全防范机制（阻止报错）来检查undeclared变量。

1. 数组

1.1数组

1. 数组可容纳任何类型的值，字符串、数字、对象，甚至是其他数组。

数组可通过数字进行索引，也可以包含字符串键值和属性，这些并不算数组长度。

还是建议使用对象来存放键值/属性值

1. 类数组（一组通过数字索引的值）
2. 字符串

JS中字符串是不可变得，而数组可变。

字符串不可变是指字符串得成员函数不会改变原始值，而是创建并返回一个新的字符串，而数组的成员函数都是在其原始值上进行操作。（字符串是基本数据类型，不可变。数组是对象，可变）