1、 ECMAScript是JavaScriprt的官方命名。使用JavaScript这个名字通常需要遵循以下规则：

（1）JavaScript指的是一种编程语言；

（2）ECMAScript是这种语言的规范的名字。因此，提到此语言的版本时，就是指ECMAScript。JavaScript当前的版本是ECMAScript5，ECMAScript6还在发展中。

2、JavaScript的创造者是Brendan Eich;

3、注意等号的两种不同的用法

（1）一个单独的等于号（=），用于为变量赋值；

（2）三个连续的等于号（===），用于比较两个值

4、语句和表达式

（1）语句“做事情”。程序其实就是一系列的语句的集合。

（2）表达式产生值。他们通常是函数的参数，或是复制的右边部分

5、分号

分号用于结束语句，而不是结束块。分号出现在块之后的情况：函数表达式作为一个表达式时。如果这样的表达式出现在语句的最后，它后面就会跟上一个分号；

Var x=3\*7;var f=function(){};

6、注释

单行注释：//

多行注释：/\* \*/

7、变量和赋值

（1）JS里的变量在声明后使用:var foo=6;

（2）变量的声明和使用可以同时进行；也可以为一个已有变量进行赋值：foo=4;

（3）复合赋值运算符，如+=：x+=1;x=x+1;

（4）标识符与变量

A、标识符区分大小写；标识符的第一个字符可以是任意的Unicode字符、美元符($)或者是下划线（\_）。后面的字符除此之外还可以为任意地Unicode数字。

B、有些标识符为保留字，不能作为变量使用（包括函数名和参数名），如break,case,do,else等等

C、以下三个标识符不是保留字，但同样需要视为保留字：Infinity,NaN,undefined

8、值

点操作符：

1. 可以读取属性：str.length;
2. 用于给属性赋值：obj.name=’Lily’;
3. 调用方法

9、原始值和对象

（1）原始值包括布尔值、数字、字符串、null、undefined

其他值为对象

（2）二者比较方式不同：

每个对象都有唯一的标识且只等于自己

所有的原始值，只要编码相同，则被认为相等

（3）原始值的特点

A、按值进行比较

B、不可改变：其属性不能被改变、添加、或移除

（4）对象（所有的非原始值都是对象）

1>A、简单对象，可以通过对象字面量来创建如，{

firstname:’Jane’,

lastname:’Doe’

}

B、数组，可以通过数组字面量来创建

C、正则表达式，可以通过正则表达式字面量来创建

2>对象特点：

1. 按引用进行比较：比较身份标识；每个值都有各自的身份标识

{}=={} false

1. 默认可变：对象的属性可以很自由的被改变、添加和移除

（5）undefined和null

A、undefined的意思是“没有值”。未被初始化的变量为undefined；丢失的参数也会是undefined；访问不存在的属性，也得到undefined。

B、null的意思是“没有对象”。在用到对象的时候他表示空值（例如参数，对象链中的最后一个元素等）

C、二者都没有属性，连toString()这种标准方法都没有

D、检查undefined或null

通常，函数允许透过undefined或null来表示缺失的值。可以通过以下显式的检查来做到同样的事：if（x===undefined||x===null）{

………

}

也可以利用undefined和null都可以被视为false这一事实来处理if（！x）{

………

}

（6）使用typeof和instanceof对值分类

Typeof主要用于原始值，它的返回值是一个表示这个值“类型”的字符串

|  |  |
| --- | --- |
| Undefined | Undefined |
| Null | object |
| 布尔值 | boolean |
| 数字 | number |
| 字符串 | string |
| 函数 | function |
| 所有其他常规值 | object |
| 引擎创建的值 | JS引擎可以被允许去创建一些值，且typeof的结果可以返回任意字符串（可以与表中列出的结果都不一样） |

typeof null会返回object（是一个bug）

instanceof用于对象:

用法：

value instanceof Constr

如果value是一个通过Constr构造器创建的对象，则返回true

9、在JS中被解释为false的值：

Undefined、null

布尔值：false

数字：-0，NaN

字符串：‘’。

其他所有值（包括所有对象）都会被当成true

10、JS中二元逻辑运算符是短路的。因为如果第一个运算数就足以确定结果的话，则不会对第二个运算数做评估：如false&&foo();true||foo()其中foo()函数永远不会被调用

（1）与（&&）

如果第一个运算数是假值，返回他。否则，返回第二个运算数

123&&‘abc’

‘abc’

(2)或（||）

如果第一个运算数是真值，返回他。否则，返回第二个运算数

11、数字

（1）JS中所有的数字都是浮点数

1===1.0

True

（2）Infinity

多数情况下，也是一个错误的值：

Infinity比任何一个书都要大（NaN除外）。同样的，-Infinity比任何一个数都要小（NaN除外）。这两个数字常用来作为默认值（比如最小值和最大值）

12、字符串

（1）字符串限定在“”或‘’之内。反斜杠（\）用于转义字符及产生一些控制字符。

（2）可以通过方括号来访问字符串中的单个字符：str[1]

（3）length属性可以对字符的个数进行计数

（4）字符串不可变

（5）字符串可以通过加号（+）进行连接

（6）字符串常用方法：

A、slice() 例’abc’.slice(1)//’bc’

B、trim() 例’\t xyz’.trim()//’xyz’

C、toUpperCase()

D、indexOf() 例’abc’.indexOf(‘b’)//1

13、条件语句

Switch语句：case之后跟的“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

14、循环语句

（1）for循环

（2）while循环

（3）do……while循环：由于条件跟在代码体之后，所以，这些代码至少会执行一次

（4）break可以跳出循环

Continue会开始一个新的循环迭代

14、函数

（1）定义函数

A、通过函数声明的方式来定义函数

B、通过给变量赋值为函数表达式的方式来定义函数

（2）函数表达式会产生一个值，因此函数可以作为参数直接传递给另外的函数；

（3）函数声明的提升特性

（4）特殊变量arguments

在JS中，函数的所有参数都可以被自由调用，他会通过arguments变量来使所有的参数可用。Arguments看起来像是个数组，却不具备数组的方法

（5）参数太多或太少

参数太多，额外的参数会被忽略（arguments除外）

参数太少，丢失的参数会得到undefined这个值

（6）可选参数

（7）强制参数长度

想强制一个参数长度（指定的参数长度），可以通过arguments.length来检查

（8）将arguments转换为数组

toArray(arrayLikeObject){

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

}

（9）异常捕获

捕获异常的方式：

Try{

}catch{

}

Throw

使用try语句包裹关键代码，如果try语句有异常会被抛出那么catch语句就会执行

15、严格模式

严格模式激活更多的警告以及是JS变得干净（非严格模式有时候也被叫做‘松散模式’）。要切换到严格模式，在JS文件或者<script>标签第一行输入：‘use strict’;

也可以在每一个函数中激活严格模式：

Function functionInstrictMode(){

‘use strict’;  
}

16、变量作用域和闭包

（1）变量是函数作用域的

一个变量的作用域总是完整的函数（相对于当前块）

（2）变量的提升特性

所有的变量声明都会被提升：声明会被移动到函数的开始处，而赋值则仍然会在原来的位置进行。例：

Function foo(){  
console.log(tmp);

If(false){  
 var tmp=3;

}

}

在程序内部执行过程是这样的

Function foo(){

var tmp;  
console.log(tmp);

If(false){  
 tmp=3;

}

}

（3）闭包

函数以及她所连接的周围作用域中的变量即为闭包。

例：function foo(start){

Return function(){//(1)

Start++;

Return start;

}  
}

函数从（1）这行开始被创建，在创建结束后即离开他的上下文环境，但他仍然保持着和start的连接：var inc=foo(5);所以，foo()的返回其实就是一个闭包。

（4）IIFE模式：引入一个新的作用域

A、想引入一个新的作用域，使用函数；将函数当作类似块的方式来使用，这种模式被称为IIFE（立即调用函数表达式）：

（function(){

Var tmp=………;

}()）

IIFE是在一个定义之后就被立即调用的函数表达式。在函数内部，会有一个新的作用域，以防止tmp变成全局变量

17、对象和构造函数

（1）单一对象

A、可直接通过对象字面量去创建普通对象：

var jane={

name:’jane’;

singing:function(){

return ……;

}

}

1>可以获取以及设置这些属性；以函数作为值的属性被称为方法。他们使用this对调用他们的对象进行引用

2>使用in运算符检查属性是否存在：‘name’ in jane

True;

3>如果读取一个不存在的属性，会得到undefined；

4>使用delete运算符移除属性

（2）如果想用其他字符串作为属性名，则必须将他们用引号引起来，再通过对象字面量和方括号来获取或设置这个属性；

方括号可以动态计算属性键名

（3）提取方法

使用bind()方法，所有函数都支持。他会创建一个this总是指向给定值的新函数：var fun=jane.singing.bind(jane);

fun();

（4）方法中的函数在嵌套函数的内部不能访问方法中的this变量

解决方法：

1. 将this保存在不同的变量中；
2. 利用forEach的第二个参数，它可以给this指定一个值

（5）构造函数：对象工厂

**构造函数的定义**

（1）使用function关键字声明

* 1. 为了区别普通函数，将构造函数首字母大写
  2. 构造函数的形参用于赋给对象的属性

（2）在构造函数中，使用this指代实例对象

* 1. 对象的属性和方法必须通过this访问

**使用构造函数创建对象**

* 1>构造函数不能被直接调用执行

2>使用 new 关键字来通过构造函数创建对象（实例化）

a>将要赋给对象属性的值通过构造函数的参数传递给对象

b>创建原料: 隐式创建一个空对象，并且 this 指代这个空对象

c>产品输出: 隐式将加工过的对象 return 返回

18、数组

（1）数组字面量

A、可以通过整数索引来访问数组中的元素

B、Length属性表明数组有多少元素，可以通过使用它来添加或者删除元素

C、In操作符也可以在数组中正常使用：

Var arr=[‘a’,’b’,’c’];

1 in arr//true

D、注意数组是对象，可以拥有对象属性

（2）数组方法

|  |  |
| --- | --- |
| **方法** | **作用** |
| join( ) | 把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。 |
| pop( ) | 删除并返回数组的最后一个元素 |
| push( ) | 向数组的末尾添加一个或更多元素，并 返回新的长度 |
| shift( ) | 删除并返回数组的第一个元素 |
| unshift( ) | 向数组的开头添加一个或更多元素，并返回新的长度 |
| 方法 | 作用 |
| concat( ) | 连接两个或更多的数组，并返回结果。 |
| reverse( ) | 颠倒数组中元素的顺序 |
| slice( ) | 从某个已有的数组返回选定的元素 |
| splice( ) | 删除元素，并向数组添加新元素 |

（3）遍历数组forEach以及map方法

ForEach迭代数组并且将当前的元素和元素的index扔到一个函数中；

Map通过应用一个函数映射到现有的数组的每个已经存在的元素创建一个新的数组；例：[1,2,3].map(function (x)){

Return x\*x;

}

//[1,4,9]

19、正则表达式

（1）test()方法：匹配吗

（2）exec()方法：匹配以及捕获分组

（3）replace()方法：搜索和替换

Replace的第一个参数必须是一个带着/g标志的正则表达式；否则将之替换第一次出现的内容。还可使用一个函数完成替换

20、Math：是一个包含运算功能的对象

Math.abs(-2);//2

Math.pow(3,2)//9

Math.max(2,1,-1)//2

Math.round(1.9)//2

Math.PI//3.141592653……

Math.cos(Math.PI)//-1

21、标准库的其他功能

（1）Date:一个日期构造器，主要功能是解析和创建日期字符串和访问组件的日期（年，小时等）

（2）JSON:一个可以解析和生成JSON数据的对象

（3）console.\*系列方法：这些浏览器特定的方法不是这个语言的一部分，但是有一些同样可以运行在Node.js中。

22、