一.简介

1.ECMAScript，简称ES，是对javascript的一种标准规范，各个浏览器都需要对这种规范进行实现，这种实现成为js引擎。chrome浏览器使用v8去实现，速度最快。

2.JavaScript包括了三个部分，ECMA规范，DOM（操作页面），BOM（操作浏览器）。

3.HelloWorld：

```
<script type="text/javascript">
 alert("HelloWorld");//弹框
 document.write("Hello");//在body内输出，即在页面上输出文字
 console.log("HelloWorld");//在控制台输出
</script>
```

二.语法

1.js编写位置（和CSS非常类似）

1.和css一样，可以写在html标签属性中，但是这种做法不好；

2.在<body>里的<script>标签里，就和css写在<style>里一样；

3.写在外部的js文件里，然后再html里引入。就和css文件一样。

```
<script type="text/javascript" scr="js/script.js"></script>
```

这种做法的好处是可以被多次使用，而且可以利用浏览器的缓存机制。

2.基本语法

1.注释，和java一样

2.声明变量，js中使用var关键字来声明变量，如var a = 1；

值得注意的是，如果我们只是声明变量var a，而没有给a赋值，那么a就是undefined

3.标识符：在js中所有我们可以自主命名的东西，都是标识符，

例如：变量名，函数名，属性名。

3.数据类型

1.String

在js中需要用引号引起来，例如：

```
var str = 'hello';
```

如果出现字符串中有引号的情况，可以考虑使用转义字符\，java也是\转义：

```
var str = "我说：\"今天天气不错\"";
```

2.Number

在js中，整数和浮点数都是number类型。js有几个特殊的数字：

Infinity：表示无穷大，这里的无穷大是超过了js能表示的最大数值

NaN：Not a number，不是一个数字

3.Boolean

4.Null

null专门用来表示空的对象。

5.Undefined

当声明一个变量又没有给它赋值时，就是Undefined类型

6.强转为String

```
var num = 123;
var str = num.toString();
console.log(typeOf str);
```

toString()方法不适用于undefined和null，类似于java的空指针异常。

除了toString()外，还有一种方法是使用String()函数。

```
var a = undefined;
var str = String(a);
```

String()函数，对于boolean和number来说就是调用toString()，而对于null和undefined则是给变量加双引号，变成字符串。

7.强转为Number

和String类似的，可以调用Number()函数。但是如果这个变量不是一个数字，比如

```
var str = "123abc";
```

那么如果调用Number()，会得到一个NaN，即not a number。

对于类似 "123px"的这种字符串，可以用另一种方法：

```
var num = parseInt("123abc");
//这样得到的就是123
```

8.强转为Boolean

Number—0或NaN会转成false；

String—空字符串会转成false；

Undefined—转成false；

Null—转成false；

Object—转成true;

4.运算符

1.typeof

返回一个变量的类型，结果是一个字符串类型

```
var a = 123;
console.log(typeof a);
//可以得到number，而这个number是字符串
```

2.算术运算符 + - * /

加减乘除。特殊情况：任何变量和字符串做加法，都是做字符串的拼接。除了这种情况之外，其他所有情况都是把变量转换成Number再做运算。

```
var result = 12 + "34"; //结果是1234，因为是加法，做字符串拼接
var result = "34" - 12; //结果是22，因为不是加法，全部转换成Number
```

3.自增和自减 ++ --

可以建立一种概念，比如a++，不管++在前面还是在后面，都会立刻使变量自增1，不同的是表达式本身的值不一样，a++的值是自增前a的值，而++a是自增后a的值，换言之这里区分了变量a的值，和表达式a++的值，这么两种概念。

4.逻辑运算符 && ||

（1）非运算：！，可以对一个Boolean值进行取反操作；

（2）与运算：&&，可以对两个Boolean值进行与运算，都为true才返回true；

短路特性：第一个值为false就不会进行第二个值的运算；

（3）或运算：||，两个都是false才返回false，同样拥有短路特性。

5.赋值运算符 = += -=

6.关系运算符 > < >= <=

NaN和任何变量比较都是false；

如果两个非Number类型，会先转换成Number类型再进行比较。但是有一种特殊情况：如果两边是字符串，会比较unicode编码的大小，这时可能会得到预期之外的结果，所以在这种情况下有必要手动把字符串转为Number类型，例如 +"5"；

7.相等运算符 ===

相等运算符有两个，一个是==，另一个是===，它们有所不同，不同在于：

==不会进行类型比较，如果两端变量的类型不同，会先转换成相同类型再比较；

===会先进行类型比较，再做值比较，如果类型不同就得到false结果;

!= 和 !==的关系类似。

8.三元运算符 a>b?a:b

5.流程控制语句

1.if语句

```
if(a>b){
 alert("a比b大");
}else if(a=b){
 alert("a等于b");
}else {
 alert("a比b小")
}
```

2.switch语句

```
switch(key){
 case value1:
 ...
 break;
 case value2:
 ...
 break;
 default:
 ...
  break;
}
```

3.while语句

```
while(条件表达式){
  循环体;
}
```

4.for语句

三.对象

上面学到了5种基本数据类型：string,number,boolean,null,undefined。而object对象类型是引用类型，属于一种复合的数据类型，在对象中可以保存多种不同数据类型的属性。

1.对象分类

1.内建对象

由ES标准中定义的对象，在任何的ES实现中都可以使用，比如Math，String，Function；

2.宿主对象

主要是指DOM，BOM等浏览器提供的对象，如console.log，document.write

3.自定义对象

由开发人员自己定义的对象

2.操作对象

1.创建对象

```
let person = {};//这个是es6的语法
```

2.给对象添加属性

```
person.name = 'sean';
```

3.给对象删除属性

```
delete person.name
```

4.给对象设置属性的另一种方式，重要!

```
person['age'] = 18;
person.age = 18;
```

使用中括号赋值的好处在于可以使用变量作为属性名，这一场景在项目中遇到过，举例为：

```
const n = 'age';
person[n] = 18;
```

当然平时还是用.来赋值或者获取属性值比较习惯。

5.属性名in对象

检查对象中是否含有指定属性，如：

```
console.log('name' in obj);
```

判断obj对象中是否有name这个属性。

3.基本数据类型和引用数据类型

```
let obj = {name: 'sean', age: 18, gender: 'male'};
let obj2 = obj;
obj2.name = 'hello';
console.log(obj);
//结果会发现obj的name也变成了hello
```

这里提到栈内存和堆内存的概念，基本数据类型的变量都存储在栈内存，每一个变量都有其对应的值，修改一个变量不会影响另一个变量的值。

```
let obj2 = {};
```

而引用数据类型（对象），其变量依然保存在栈，但是其值并不是对象本身，而是一个指向堆内存对象的地址值（new Object时会在堆内存创建一个空间保存对象），每个变量对应一个内存地址。当我们对对象进行赋值时，其实并没有创建新的对象，obj和obj2指向同一个地址（对象引用），因此改变obj的属性会使obj2也一起改变。

要想让两个对象不相互印象，就必须new一个新的对象，代码如下:

```
let obj2 = Object.assign({},obj);
let obj2 = {...obj};
```

但是注意，obj2 = null 并不会使得obj指向的对象消失，obj2 = null 只是把obj2的值（地址值）变为null，即obj2和对象的指向关系消失了。

```
obj2 = null;
//不会使obj产生影响
```

四.函数

函数也是对象，在es6中可以用lambda表达式（箭头函数）来表示。函数可以封装一些功能（方法），在需要时被调用。

1.创建函数对象

```
const fun = (str) => {
 console.log(str)
};

function fun2(str2) {
 console.log(str2);
}
```

第一种是lambda表达式风格，第二种是函数声明风格。还有一种new Function的面向对象风格并不常用，就不提了。

2.函数返回值

```
function sum(a,b){
 return a+b;
}
const result = sum(1,2);
```

如果一个函数没有返回值，或者return后面没有跟任何变量，那么返回值就是undefined