你不知道的javascript（中卷）知识点总结

1. 类型
2. ECMAScript语言中所有的值都有一个对应的语言类型，语言类型包括Udefined、Null、Boolean、String、Number和Object。
3. Javascript有七种内置类型：空值（null）、未定义（undefined）、布尔值（boolean）、数字（number）、字符串（string）、对象（object）、符号（symnol。ES6中新增）。

例： typeof Symbol（）===”symbol”; //true

typeof null===”object”; //true

1. Null是基本类型中唯一的一个“假值”，检测null值的类型：使用复合条件

Var a = null;

(!a && typeof a ===”object”);//true

1. 函数function是javascript的一个内置类型，实际上是object的一个“子类型”，函数是“可调用对象”，内部属性[[Call]]使其可以被调用。函数对象的length属性是其声明的参数的个数。
2. 数组也是object的一个“子类型”，数组的元素按数字顺序来进行索引，其length属性是元素的个数。
3. Javascript中变量没有类型，只有值才有类型，变量可以随时持有任何类型的值。对变量执行typeof操作时，得到的结果是变量持有的值的类型。
4. 变量在未持有值的时候为undefined，即已在作用域中声明但还没有赋值的变量；还没有在作用域中声明过的变量，是undeclared。

Typeof对undeclared变量的处理方式：typeof b；//“undefined”

Typeof b；没有报错，是因为typeof有一个特殊的安全防范机制，该安全防范机制对在浏览器中运行的javascript代码来说是很有帮助的，因为多个脚本文件会在共享的全局命名空间中加载变量。

第二章 值

1.数组可以容纳任何类型的值，可以是字符串，对象或其他数组（多维数组）；对数组声明后即可向其中加入值，不需要预先设定大小；delete运算符可以将单元从数组中删除，单元删除后，数组的length属性不会发生变化。

2.创建稀疏数组时，a[1];//undefined 与a[1]=undefined;不同。

3.数组通过数字进行索引，也可以包含字符串键值和属性，但是建议使用对象来存放键值/属性值，用对象来存放数字索引值。

4.类数组转换为真正数组，一般通过数组工具函数（indexOf（），concat（），forEach（））实现。ES6中的内置工具Array.from（）也能实现相同功能。

5.字符串和数组都是类数组，都有length属性以及indexOf（）和concat（）方法。

6.Javascript中字符串是不可变的，而数组是可变的。字符串的不可变是指字符串的成员函数不会改变其原始值，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。

7.JavaScript中整数没有小数的十进制数，42.0等同于42。Javascript使用的是“双精度”格式。

8.数字的语法：数字前面的0可以省略，小数点后小数部分的0也可以省略；特别大和特别小的数字默认用指数格式显示，与toExponential（）函数的输出结果相同。

9.数字值可以使用Number对象进行封装，因此数字值可以调用Number.prototype中的方法。toFixed（显示小数位数）方法可指定小数部分的显示位数；toPrecision（位数）方法指定有效数位的显示位数。Eg：42.toFixed（3）；无效语法，点·被视为常量42.的一部分，在 42与点·之间加一个空格，该语句有效。

10.较小的数值：0.1+0.2===0.3//false，浮点数中0.1和0.2并不是十分精确。判断0.1+0.2和0.3是否相等，设置一个误差范围值，这个值通常是2^-52，ES6之后，这个值定义在Number.EPSILON中，可以直接用。

11.最大的浮点数大约是1.789e+308，定义在Number.MAX\_VALUE中；最小的浮点数定义在Number.MIN\_VALUE中，大约是5e-324，不是负数，但无限接近于0！

最大整数是2^53-1，在ES6中被定义为Number.MAX\_SAFE\_INTEGER；最小整数在ES6中被定义为Number.MIN\_SAFE\_INTEGER。

1. 整数检测：ES6中Number.isInteger（）方法
2. 数位运算符|只适用于32位整数，a | θ可以将变量a中的数值转换为32位有符号整数。
3. Undefined和null的异同：undefined类型只有一个值，即undefined，null类型也只有一个值，即null，他们的名称既是类型也是值。Null指空值，是曾赋过值，但是目前没有值，undefined指没有值，从未赋值。Null是一个特殊关键字，不是标识符，我们不能将其作为变量来使用和赋值，但undefined却是一个标识符，可以被当做变量来使用和赋值。
4. Void运算符，通过void运算符可以得到undefined，void并不改变表达式的结果，只是让表达式不返回值。
5. NaN不是一个数字，仍是数字类型，他与自身不相等。isNaN（）判断是否是NaN。“foo”不是一个数字，但也不是NaN，但是window.isNaN（“foo”）；返回true。ES6之后，Number.isNaN（“foo”）；返回false。
6. JavaScript的运算结果有可能溢出，此时结果为Infinity或者-Infinity。Infinity是一个未定义的操作，结果为NaN.
7. （-0）转换为字符串为“0”，“-0”转换为数字为-0.零的比较：0==-0；0===-0
8. ES6中新加入Object.is（a，b）来判断两个值是否绝对相等。
9. Javascript引用指向的是值，如果一个值有10个引用，这些引用指向的都是同一个值，他们相互之间没有引用/指向关系。
10. 基本类型值总是通过值复制的方法来赋值/传递；引用数据类型总是通过引用复制的方来赋值/传递。引用指向的是值本身而非变量，所以一个引用无法更改另一个引用的指向。
11. 如果通过值复制的方式来传递数组，就需要为其创建一个复本，这样传递的就不再是原始值eg：foo（a.slice（））；slice（）不带参数会返回当前数组的一个浅复本。由于传递给参数的是指向该复本的引用，所以不会影响a指向的数组。

第三章 原生函数

第四章 强制类型转换

1.类型转换发生在静态类型语言的编译阶段，而强制类型转换则发生在动态类型语言的的运行时。

2.toString，负责处理非字符串的强制类型转换，var a=[1,2,3];a.toString();//”1,2,3”。

3.JSON.stringify（）在对象中遇到undefined、function和symbol时会自动将其hulue，在数组中则会返回null。对包含循环引用的对象执行JSON.stringify（）会出错。

4.向JSON.stringify（）传递一个可选参数replacer，他可以是数组或者函数，用来指定对象序列化过程中哪些属性应该被处理，哪些应该被排除。如果是一个数组，则必须是一个字符串数组，其中包含序列化要处理的对象的属性名称，除此之外其他的属性则被忽略；如果是一个函数，他会对对象本身调用一次，然后对对象中的每个属性个调用一次，每次传递两个参数，键和值，如果要忽略某个键就返回undefined，否则返回指定的值。

5.ToNumber，将非数字值当做数字来使用，true为1，false为0，undefined为NaN，null为0。