12.09-js05

一、异常

（一）、异常语句：

1、 定义：在ECMAScript-262中规定了7类错误类型。其中error是基类，其它6种是error的子类，继承了error，Error是自定义错误对象。

2、分类：

2.1 Error：普通异常，与try、catch、throw一起使用。属性name可以读写异常类型，massage可以读取异常信息。

2.2 EvalError：不正确使用eval方法导致。

 2.3 SyntaxError：语法错误。（[ˈsɪntæks] n. 句法; 句法规则; 语构; ）

2.4 RangeError：操作数超出合法范围。

 2.5 ReferenceError：读取不存在的变量时。

 2.6 typeError：值的类型发生错误

2.7 URLError：url编码和解码的时候报错。

 （二）、异常处理：

1、try、catch、finally都是异常处理语句

2、try{调试的代码}catch(e){捕获的异常,catch的参数就是异常类型，当异常后执行的代码}finally{后期清理代码块}; 正常情况下，JS按顺序执行try中的代码，如果没有异常发生，将忽略catch直接进入finally。 如果在try中出现的异常，或者是使用throw抛的异常，将会执行catch代码，catch代码的参数是错误类型

二、函数

1、定义：函数是一段被封装的代码，可以被反复调用。

2、类别：可以是一个值、一个对象、或是一个表达式。所以函数可以赋值、运算、可以拥有属性和方法。所以函数也被称为一等公民。

3、声明函数：使用function声明函数 function name([args]){代码块}

A、var语句和function语句都是声明语句，都在预编译时被解析，被用于变量和函数提升。

B、大小括号都必须有

C、函数只有定义在了其它函数中才叫局部函数，否则都是全局的，在哪都可以调用。但是：不允许将函数定义在其它函数中。如：在if中定义一个function函数，在if外也可以调用这个函数（就是说不走if就可以调用这个函数，那这么写有什么意义呢！！！所以不要酱紫在if里面写啦~）

4、匿名函数：

 A、没有名字的函数，又叫函数直接量，只有函数声明、参数和函数体；匿名函数就是一个表达式，所以也叫函数表达式；可以把它作为一个函数 赋值给一个变量或者是一个对象的方法，这个时候变量和对象的方法就是这个函数了。

 B、调用的方法：

1.)()

2.在函数前添加！~+-等一元运算符加小括号自调用，如：！function(){}

3.var ~ = function(){}

5、构造函数：

使用 Function构造函数，var fn1 = new Function(p1,p2,p3,p4.....,body) \* p1---pn 都是函数的参数 \* body是函数体 \* p1--pn 和 body都是字符串格式

三、调用函数

1、函数调用

2、方法调用

3、apply和call

4、new

四、函数返回值

1、函数提供了两个与外界的交互接口，一个是参数入口（用来接收外界的信息），一个是返回值作为出口（把执行结果返回出去）。

2、在函数内部用return语句来操作函数返回值，一旦执行了return，函数将停止运行，return后面的代码看都不会看你一眼！！！return语句不带返回的东西就返回undefined。

3、函数对接收和输出的值的类型没有限制。

五、函数的参数

1、实参：在调用函数的时候传入的参数

2、形参：在定义函数的时候声明的，仅在函数内部可用。以后才会用到的形参可以在定义时给一个默认值(js6以上支持)，目前一般在函数中用三目运算的方法来计算那个形参是不是某 类型（如 ：c = typeof c === "undefined" ? 0 : c;）

3、形参与实参的个数问题

A、实参数 < 形参数， 多出来的形参就是undefined

B、实参数 > 形参数， 多出来的实参不被访问

 4、获取参数的个数

A、使用arguments对象可以获取到函数的实参个数

B、arguments对象表示函数实参的集合，仅能在函数内部访问

C、arguments对象是一个伪数组，不能直接使用数组的方法，可以像数组一样使用下标以及length方法。

D、arguments.length就是实参的个数

E、callee属性是arguments对象的属性，它是引用当前arguments对象所在的函数，使用该属性，可以调用自身函数。

 5、获取形参的个数

A、直接使用函数的length属性获取

 六、函数的作用域

1、函数作用域、变量作用域（scope）指的是变量在程序中可以访问的有效范围，也称变量可见性。

2、全局作用域：变量在整个页面都是可见的，可以被自有的访问

3、局部作用域：变量仅能够在当前函数内部可见（或者是当前函数内部的函数可见），函数外不允许访问

4、var 或者是 function声明的变量和函数 是按照函数划分的

七、执行上下文：

\* JS按照顺序依次从上至下解析，当然在解析之前要先进行预编译（预处理），比如声明提升、函数提升 \* JS可以执行的代码有3种类型，全局代码、函数代码（局部）、eval。 \* 每次JS碰到一段可执行代码都会创建一个执行上下文，在js中会存在很多个执行上下文。所以JS会先创建一个执行上下文栈，用来管理所有的执行上下文 \* 执行上下文是一个抽象的概念，实际上就是内存中的一个区域 \* 当js执行开始解析程序的时候，最先遇到的全局代码，所以在刚开始，先生成一个全局的执行上下文，并且全局的执行上下文只有在程序全部运行结束以后，才清空 \* 当执行到一个函数的时候，会创建一个局部的全局上下文，当函数执行完毕，会把局部上下文从栈里弹出 \* 每一个执行上下文都有3个基本属性：变量对象，作用域链、this

八、练习

1、ATM按键

2、买保险

3、收银程序

九、晨测