

浙江大学城市学院实验报告

课程名称 跨平台脚本开发技术

实验项目名称 实验二 函数

学生姓名 吴成洋 专业班级 软件工程 1404 学号 31401417

实验成绩 指导老师（签名） 日期

注意：

- 务请保存好各自的源代码，以备后用。
- 请把作业保存为 pdf 上传到 BB 平台，请务必在截止日期前提交。

实验目的：

1. 掌握函数的定义与使用；
2. 理解闭包。

实验内容：

1、教材 p38 习题 1,2,3。（必做）

2、运行 p.30 调用栈 chicken egg 函数。（必做）

2.1 请修改程序，记录并显示在溢出前最多可以相互调用次数

2.2 请给程序添加 不同个数，不同类型的参数

2.3 参数个数的变化对调用次数的影响

2.4 参数类型对调用次数的影响

2.5 假设一个 Number 占用 8 字节，请估计调用栈的大小

```
var called = 0; function egg(a,b,c) {  
  called = called + 1 ; chicken(a,b,c); }; console.log(called);
```

3、运行 p33 findSolution 程序（选做）

3.1 请修改程序计算出 1-100 内 调用深度最大的 数字

3.2 画出该数字 p.34 的调用图

4、递归实现考拉兹猜想 Collatz conjecture（选做）

任取一个自然数，如果它是偶数，我们就把它除以 2，如果它是奇数，我们就把它乘 3 再加上 1。在这样一个变换下，我们就得到了一个新的自然数。如果反复使用这个变换，我们就会得到一串自然数。比如说我们先取 5，首先我们

得到 $3*5+1=16$ ，然后是 $16/2=8$ ，接下去是 4，2 和 1，由 1 我们又得到 4，于是我们就陷在 $4 \rightarrow 2 \rightarrow 1$ 这个循环中了。再举个例子，最开始的数取 7，我们得到下面的序列： $7 \rightarrow 22 \rightarrow 11 \rightarrow 34 \rightarrow 17 \rightarrow 52 \rightarrow 26 \rightarrow 13 \rightarrow 40 \rightarrow 20 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$ 这次复杂了一点，但是我们最终还是陷在 $4 \rightarrow 2 \rightarrow 1$ 这个循环中。

4.1 请编程 给出一个数字，输出考拉兹序列

4.2 请输出 1-100 间 考拉兹序列最长的数字

5、理解 第一级对象 First Class Object 4 条特点（必做）

5.1 写出代码，在代码中用注释说明 是哪条特点

6、理解 p.31 闭包（必做）

6.1 写一个 function makeAccount(n) 程序

```
var account = makeAccount(100); account(10); //给帐号存钱
```

```
account(-10); //给帐号扣钱
```

```
account(); //显示当前账户余额
```

6.2 采用部分应用技术,支持多个币种 'RMB' 'EURO' '\$'

```
var rmbAccount = makeAccount('RMB'); var account = rmbAccount(10);
```

7、理解 纯函数和副作用函数的区别,写出一个纯函数（数学函数）和有副作用的函数（访问外面变量，修改外部变量，输入输出）。（必做）

7.1 说明使用纯函数的好处

8、请举例说明 原始值类型 Number String Boolean 比较与 复合类型 [] {} 比较的不同。（必做）

9、学习 es6 函数部分（<http://es6.ruanyifeng.com/#docs/function>）（选做）

10、学习使用帮助手册 devdocs（<http://devdocs.io/>）（选做）

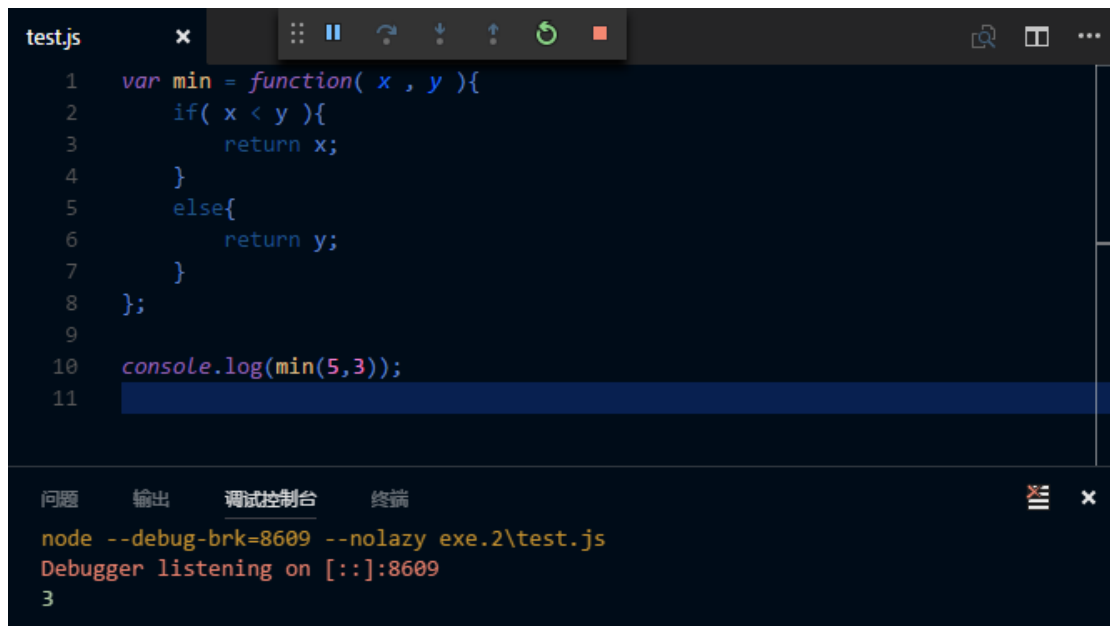
10.1 查找帮助，配置(enable) 帮助文件

10.2 将帮助文档下载到本地

实验步骤：

1、

1.1



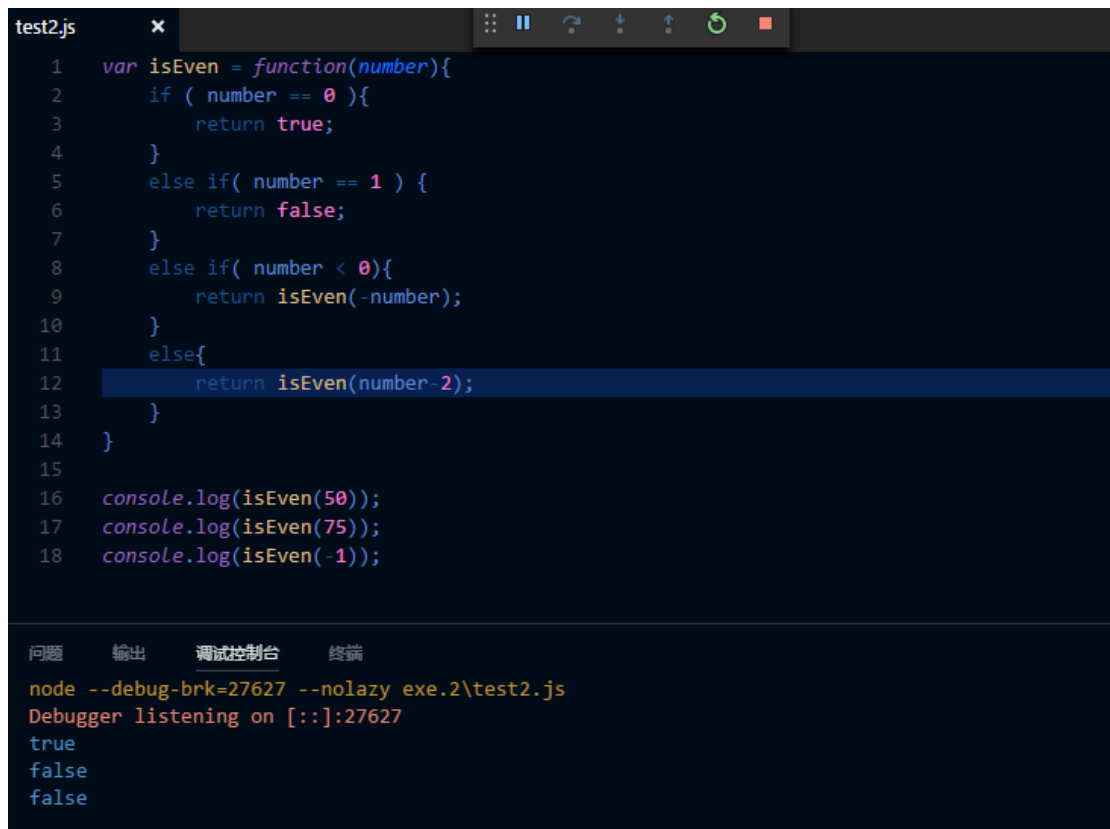
The screenshot shows a code editor with a file named `test.js`. The code defines a function `min` that takes two arguments `x` and `y`. It uses an `if` statement to return the smaller of the two. Below the function, `console.log(min(5,3));` is called. The bottom panel shows the command `node --debug-brk=8609 --nolazy exe.2\test.js` and the output `3`.

```
1  var min = function( x , y ){
2      if( x < y ){
3          return x;
4      }
5      else{
6          return y;
7      }
8  };
9
10 console.log(min(5,3));
11
```

问题 输出 调试控制台 终端

```
node --debug-brk=8609 --nolazy exe.2\test.js
Debugger listening on [::]:8609
3
```

1.2



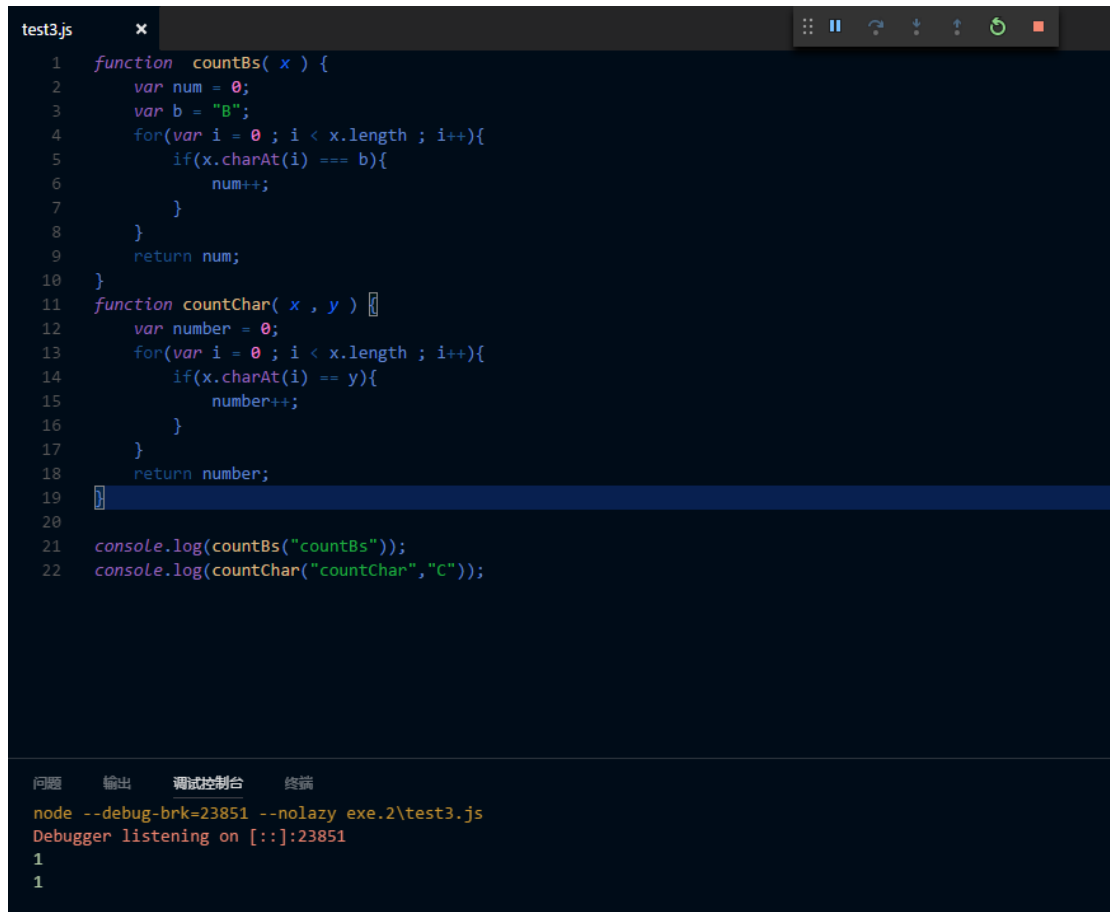
The screenshot shows a code editor with a file named `test2.js`. The code defines a recursive function `isEven` that checks if a number is even. It uses `if` and `else if` statements to handle different cases, including a recursive call `isEven(-number)`. Below the function, three `console.log` statements are used to test the function with `50`, `75`, and `-1`. The bottom panel shows the command `node --debug-brk=27627 --nolazy exe.2\test2.js` and the output `true`, `false`, and `false`.

```
1  var isEven = function(number){
2      if ( number == 0 ){
3          return true;
4      }
5      else if( number == 1 ) {
6          return false;
7      }
8      else if( number < 0 ){
9          return isEven(-number);
10     }
11     else{
12         return isEven(number-2);
13     }
14 }
15
16 console.log(isEven(50));
17 console.log(isEven(75));
18 console.log(isEven(-1));
```

问题 输出 调试控制台 终端

```
node --debug-brk=27627 --nolazy exe.2\test2.js
Debugger listening on [::]:27627
true
false
false
```

1.3



The image shows a code editor window titled 'test3.js' with a dark theme. The code defines two functions: 'countBs' and 'countChar'. 'countBs' counts the number of 'B's in a string, and 'countChar' counts the number of a specific character 'y' in a string 'x'. Below the functions, there are two console.log statements: 'console.log(countBs("countBs"));' and 'console.log(countChar("countChar", "C"));'. The editor has a toolbar with icons for running, stepping through, and other debugging actions. At the bottom, there is a '调试控制台' (Debug Console) tab showing the command 'node --debug-brk=23851 --nolazy exe.2\test3.js' and the output 'Debugger listening on [::]:23851' followed by two '1' values, corresponding to the console.log statements.

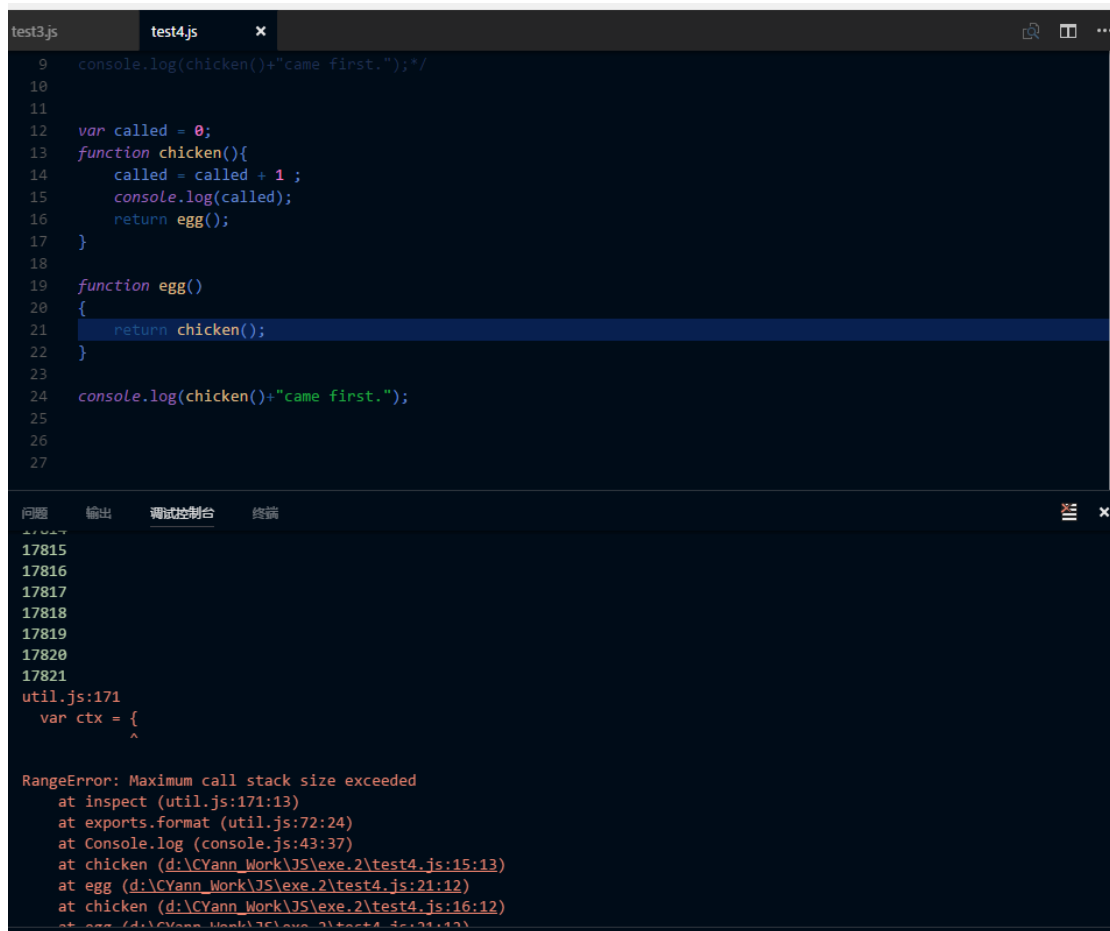
```
1 function countBs( x ) {
2   var num = 0;
3   var b = "B";
4   for(var i = 0 ; i < x.length ; i++){
5     if(x.charAt(i) === b){
6       num++;
7     }
8   }
9   return num;
10 }
11 function countChar( x , y ) {
12   var number = 0;
13   for(var i = 0 ; i < x.length ; i++){
14     if(x.charAt(i) == y){
15       number++;
16     }
17   }
18   return number;
19 }
20
21 console.log(countBs("countBs"));
22 console.log(countChar("countChar", "C"));
```

问题 输出 调试控制台 终端

```
node --debug-brk=23851 --nolazy exe.2\test3.js
Debugger listening on [::]:23851
1
1
```

2、

2.1



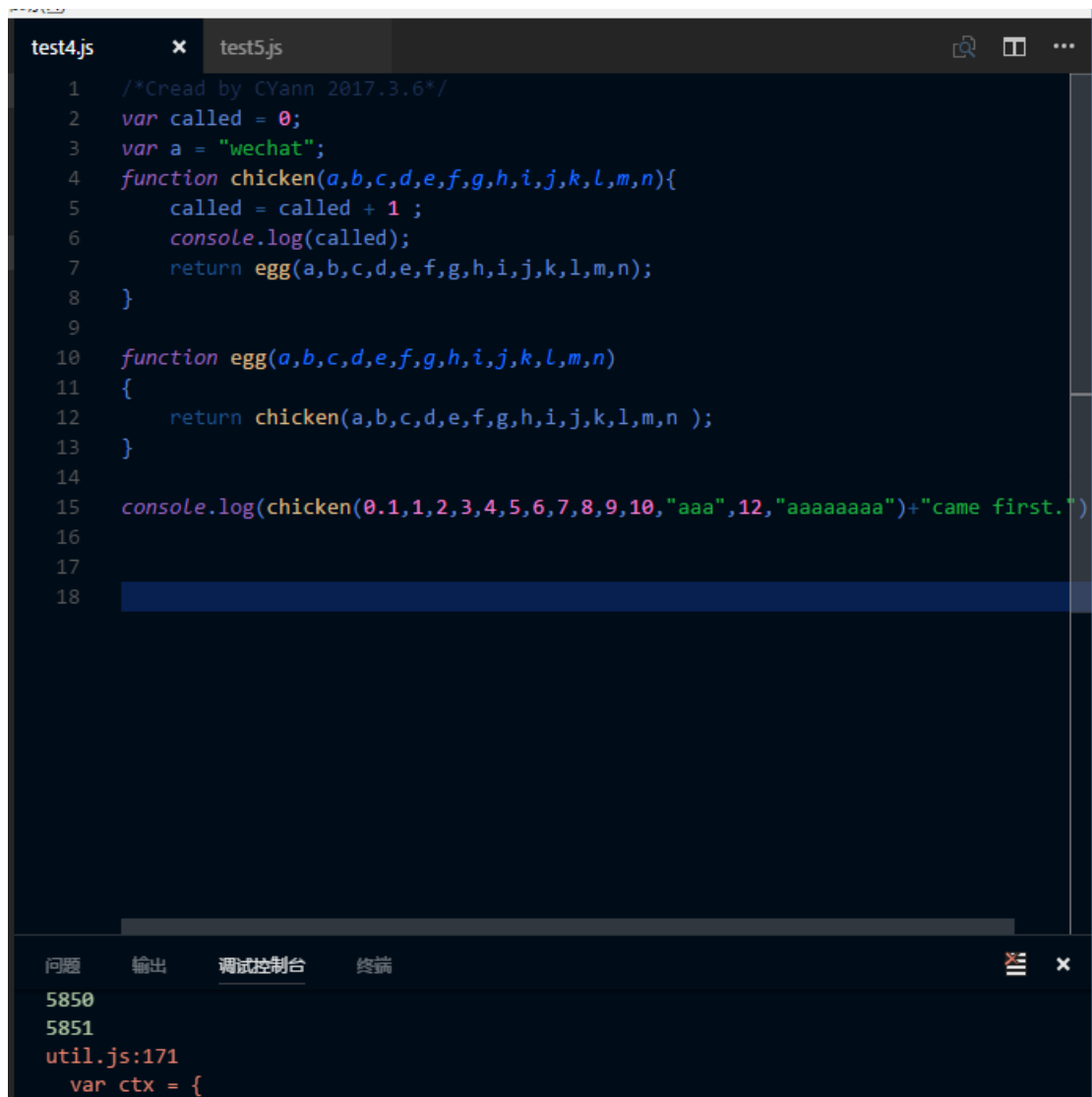
```
9 console.log(chicken()+"came first.");
10
11
12 var called = 0;
13 function chicken(){
14   called = called + 1;
15   console.log(called);
16   return egg();
17 }
18
19 function egg()
20 {
21   return chicken();
22 }
23
24 console.log(chicken()+"came first.");
25
26
27
```

问题 输出 调试控制台 终端

17815
17816
17817
17818
17819
17820
17821
util.js:171
var ctx = {
 ^

RangeError: Maximum call stack size exceeded
at inspect (util.js:171:13)
at exports.format (util.js:72:24)
at Console.log (console.js:43:37)
at chicken (d:\CYann_Work\JS\exe.2\test4.js:15:13)
at egg (d:\CYann_Work\JS\exe.2\test4.js:21:12)
at chicken (d:\CYann_Work\JS\exe.2\test4.js:16:12)
at egg (d:\CYann_Work\JS\exe.2\test4.js:21:12)

2.2

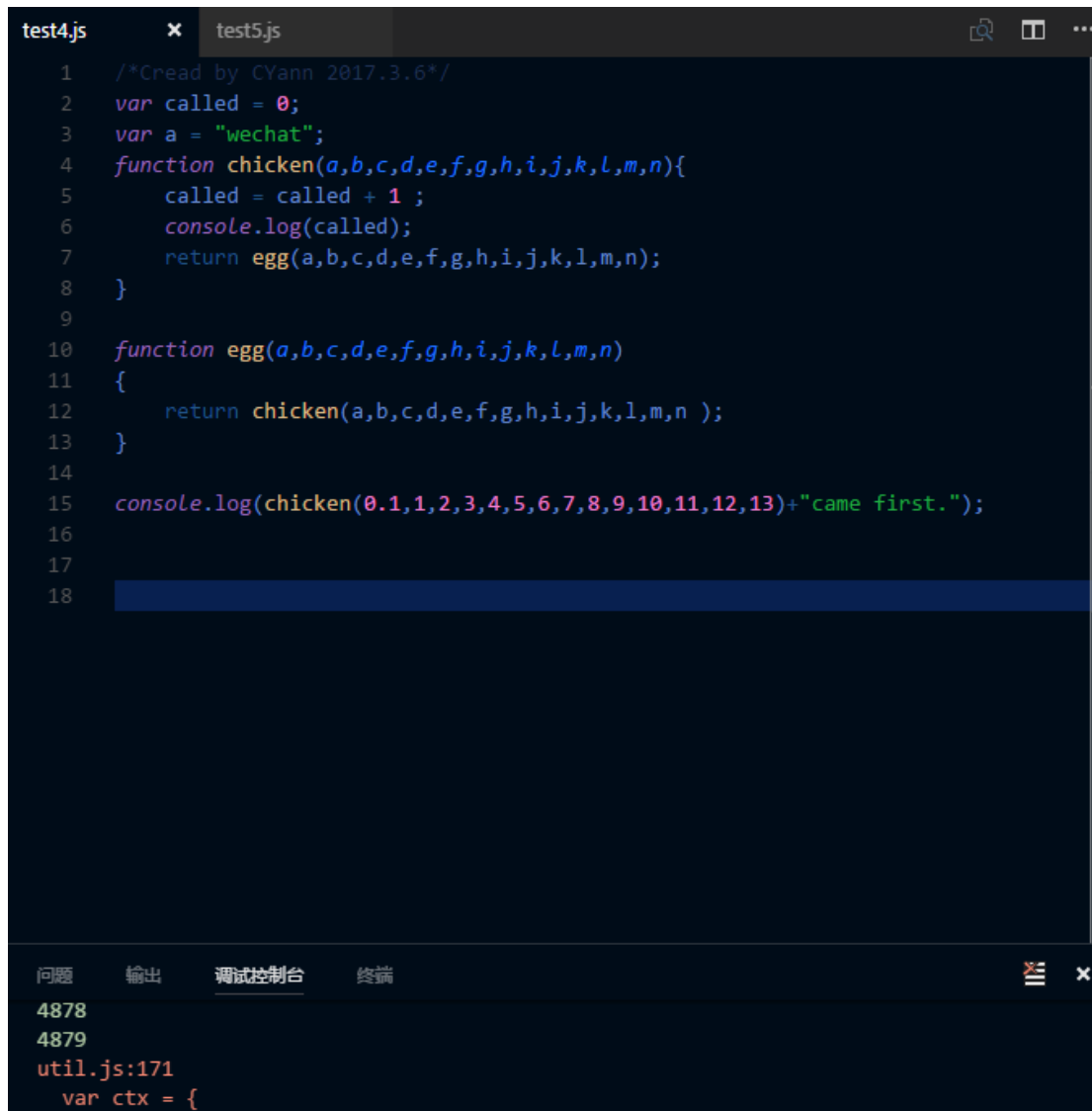


```
test4.js  x  test5.js  [icons]
1  /*Cread by CYann 2017.3.6*/
2  var called = 0;
3  var a = "wechat";
4  function chicken(a,b,c,d,e,f,g,h,i,j,k,l,m,n){
5      called = called + 1 ;
6      console.log(called);
7      return egg(a,b,c,d,e,f,g,h,i,j,k,l,m,n);
8  }
9
10 function egg(a,b,c,d,e,f,g,h,i,j,k,l,m,n)
11 {
12     return chicken(a,b,c,d,e,f,g,h,i,j,k,l,m,n );
13 }
14
15 console.log(chicken(0.1,1,2,3,4,5,6,7,8,9,10,"aaa",12,"aaaaaaaa")+ "came first.")
16
17
18
```

问题 输出 调试控制台 终端

```
5850
5851
util.js:171
var ctx = {
```

2.3



The image shows a code editor with two tabs: 'test4.js' and 'test5.js'. The 'test5.js' tab is active, displaying the following JavaScript code:

```
1  /*Cread by CYann 2017.3.6*/
2  var called = 0;
3  var a = "wechat";
4  function chicken(a,b,c,d,e,f,g,h,i,j,k,l,m,n){
5      called = called + 1 ;
6      console.log(called);
7      return egg(a,b,c,d,e,f,g,h,i,j,k,l,m,n);
8  }
9
10 function egg(a,b,c,d,e,f,g,h,i,j,k,l,m,n)
11 {
12     return chicken(a,b,c,d,e,f,g,h,i,j,k,l,m,n );
13 }
14
15 console.log(chicken(0,1,1,2,3,4,5,6,7,8,9,10,11,12,13)+"came first.");
16
17
18
```

Below the code editor is a debug console with tabs for '问题' (Issues), '输出' (Output), '调试控制台' (Debug Console), and '终端' (Terminal). The '调试控制台' tab is selected, showing the following output:

```
4878
4879
util.js:171
    var ctx = {
```

2.4

$8 \times 13 \times 4879 / 1024 = 495$

3、

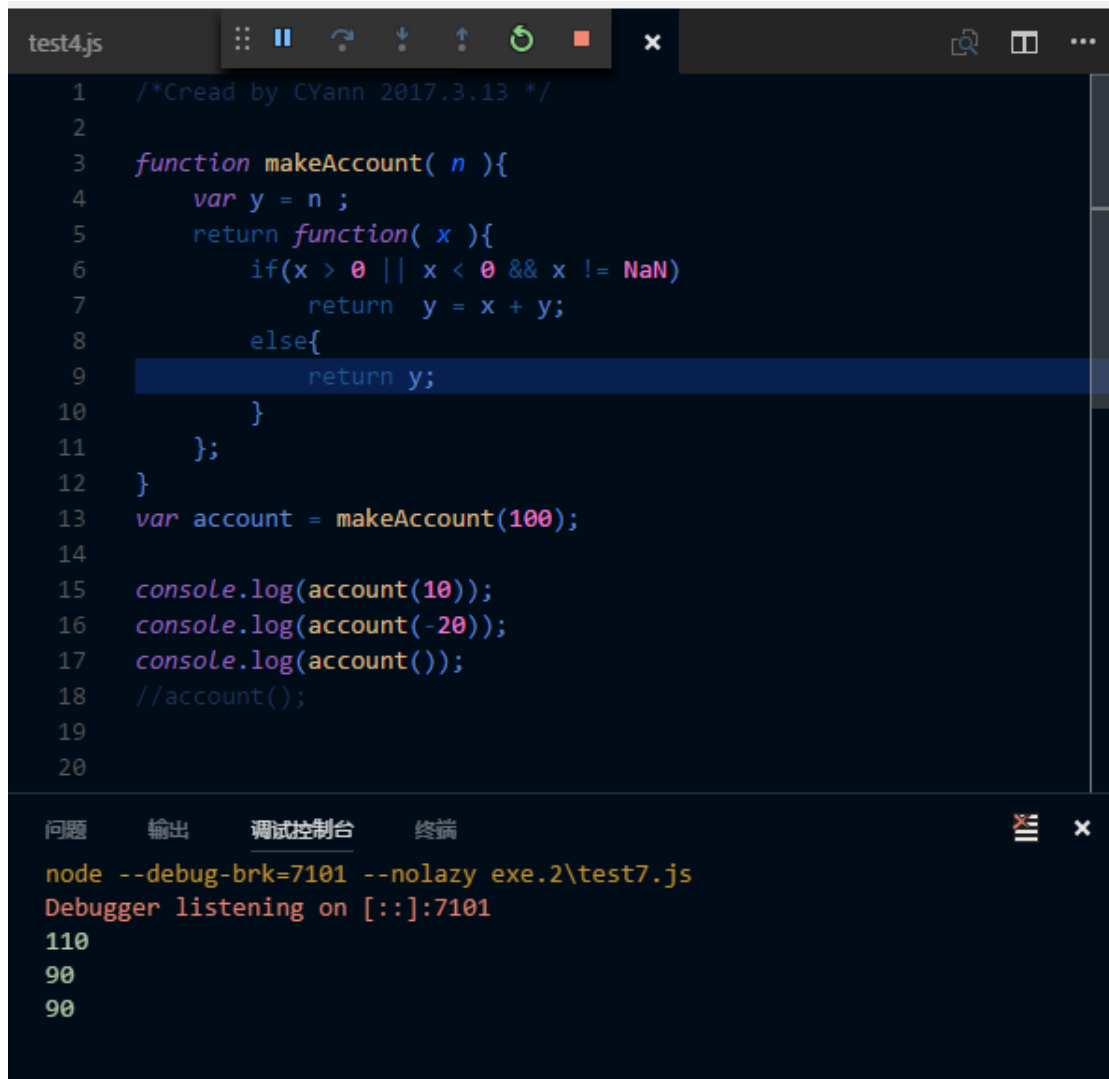
4、

5、

第一级对象 First Class Object

某种编程结构，构造子 Constructor，支持如下特性，被称为 第一级对象

- 赋值给变量
- 放置于数据结构
- 作为参数传递给其他函数

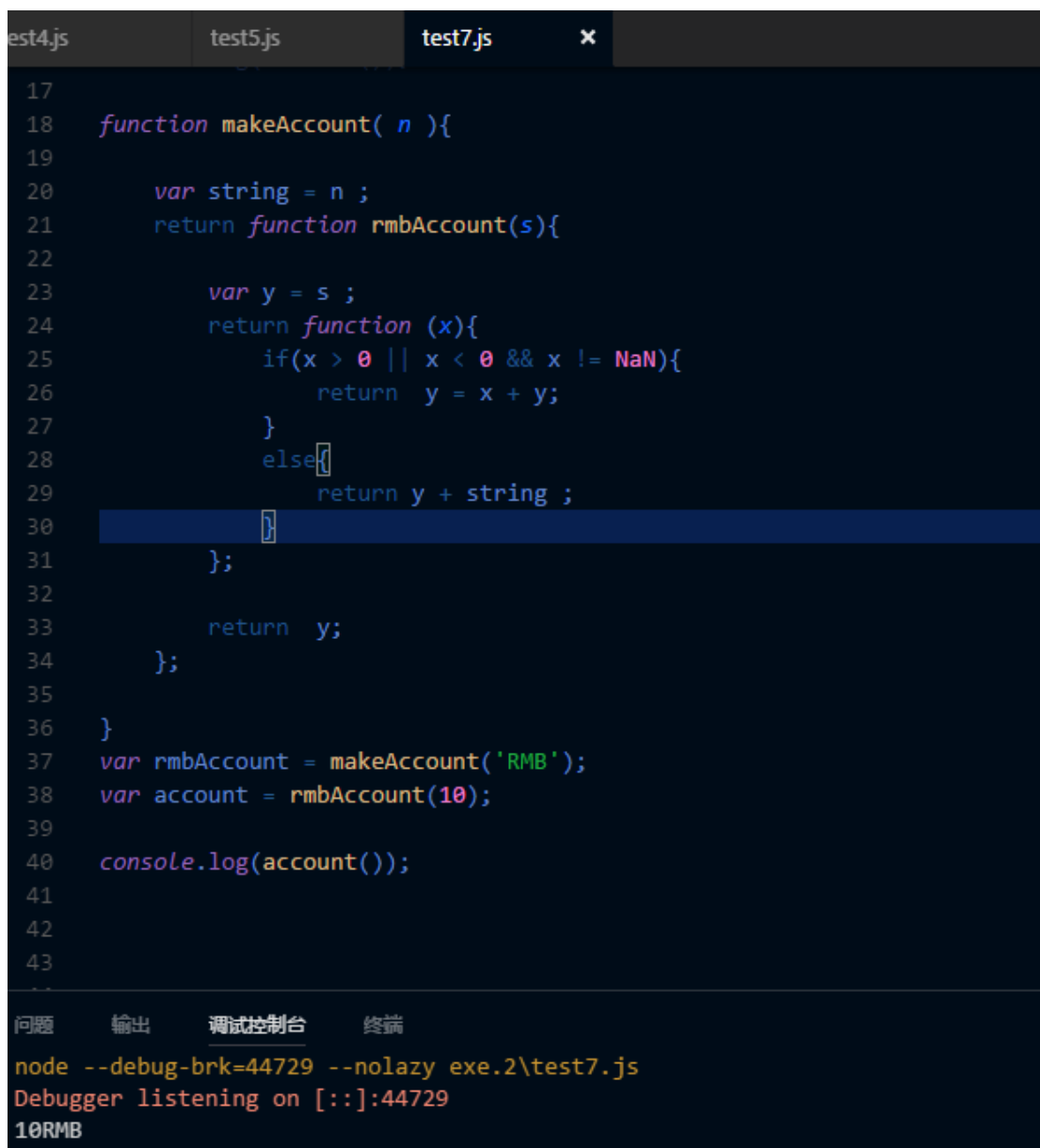


```
test4.js
1  /*Cread by CYann 2017.3.13 */
2
3  function makeAccount( n ){
4      var y = n ;
5      return function( x ){
6          if(x > 0 || x < 0 && x != NaN)
7              return y = x + y;
8          else{
9              return y;
10         }
11     };
12 }
13 var account = makeAccount(100);
14
15 console.log(account(10));
16 console.log(account(-20));
17 console.log(account());
18 //account();
19
20
```

问题 输出 调试控制台 终端

```
node --debug-brk=7101 --nolazy exe.2\test7.js
Debugger listening on [::]:7101
110
90
90
```

6.2、



```
17
18 function makeAccount( n ){
19
20     var string = n ;
21     return function rmbAccount(s){
22
23         var y = s ;
24         return function (x){
25             if(x > 0 || x < 0 && x != NaN){
26                 return y = x + y;
27             }
28             else{
29                 return y + string ;
30             }
31         };
32
33         return y;
34     };
35 }
36
37 var rmbAccount = makeAccount('RMB');
38 var account = rmbAccount(10);
39
40 console.log(account());
41
42
43
...
问题 输出 调试控制台 终端
node --debug-brk=44729 --nolazy exe.2\test7.js
Debugger listening on [::]:44729
10RMB
```

7、

纯函数(PureFunction)是这样一种函数——输入输出数据流全是显式(Explicit)的。

显式(Explicit)的意思是，函数与外界交换数据只有一个唯一渠道——参数和返回值；函数从函数外部接受的所有输入信息都通过参数传递到该函数内部；函数输出到函数外部的所有信息都通过返回值传递到该函数外部

非纯函数（ Impure Function ）

与之相反。 隐式（Implicit）的意思是，函数通过参数和返回值以外的渠道，和外界进行数据交换。比如读取/修改全局变量，都叫作以隐式的方式和外界进行数据交换。

引用透明（ Referential Transparent ）

引用透明的概念与函数的副作用相关，且受其影响。 如果程序中两个相同值得表达式能在该程序的任何地方互相替换，而不影响程序的动作，那么该程序就具有引用透明性。它的优点是比非引用透明的语言的语义更容易理解，不那么晦涩。纯函数式语言没有变量，所以它们都具有引用透明性。

```
function f(x,y){  
    return Math.pow(x,y);  
}
```

```
var a = 5;  
function fun(){  
    a = 10;  
}  
fun(); // a 变成了10
```

7.1、

纯函数的好处：

一是不必为函数命名，避免了污染全局变量；二是 IIFE 内部形成了一个单独的作用域，可以封装一些外部无法读取的私有变量。

8、

原始值类型 Number String Boolean 复合类型有函数，数组，类，
为对象类型

JS 基本数据类型的变量存放的是基本类型数据的实际值；而引用数据类型
的变量保存对它的引用，即指针

9、

10、