学习方法

1. 建立逻辑思维
   1. 多分析
   2. 理解逻辑算法
   3. 多练、多敲、多读
2. 编程习惯
3. 大胆动手
4. 保持好奇好
5. 乐观面对错误

就业能力

1. 分析能力
2. 独立思考能力
3. 程序阅读能力
4. 程序排错能力
5. 独立编写程序能力
6. 团队协作能力
7. 学习能力
8. 善于总结的能力

专业技能

函数（上）

1. 函数的概念及作用
   1. 函数的概述
      1. 概念：对于需要反复使用的功能代码，将其封装成一个独立的模块，这种功能代码块叫函数。
      2. 功能：将代码组织为可利用的单位，可以完成特定的任务并返回数据。
      3. 类型：
         1. 内建(内置、库、系统)函数和自定义函数
      4. JavaScript的函数属于Function对象
2. 定义函数
   1. 语句定义法（任何地方都可调用）（声明法）
      1. 格式  
         function 函数名（【形式参数1，形式参数2，……】）{  
          语句组;  
         }
   2. 表达式定义法（必须先声明，后调用）（赋值法）
      1. 格式  
         var 函数名 = function([形式参数1，……]){  
          语句组;  
         }
   3. 函数的调用：（一般调用和递归调用）
      1. 函数的一般调用
         1. 函数名（【实际参数1，实际参数2，……】）;
      2. 递归调用：让一个函数从其内部调用自身。
         1. 作用：实现循环语句的功能。
   4. 函数的参数
      1. 实际参数（实参）：调用函数时传递的参数。
         1. 实参可以是变量、常量、表达式
         2. 实参数量大于形参数量时，多余的实参自动省略
         3. 实参数量小于形参数量时，多余的形参值为undefined
         4. 传参时，只能由实参传递给形式，即单向传递
      2. 形式参数（形参）：定义函数时接收数据的参数。
         1. 形式参数只能是变量
   5. Return返回值
      1. 函数中可以有多条return语句，但是只能返回一条return语句的值。
      2. 函数中可以没有return 语句。

第二节：应用

1. 编写一个函数，计算两个数字的和、差、积、商  
   要求：使用传参的形式
2. 编写一个函数，计算三个数字的大小，按从小到大顺序输出

第三节

1. 事件的概念种类及作用
   1. 概念：通常鼠标或热键的动作我们称之为事件（event），热键引发的一连串程序的动作，称之为事件驱动(event Driver)。而对事件进行处理的程序或函数，我们称之为事件处理程序。
   2. 种类
      1. 页面事件
         1. onload:页面加载事件
         2. onfocus：得焦事件（获得焦点时触发事件）
         3. onblur:失焦事件（失去焦点时触发事件）
         4. onchange:状态改变事件
      2. 鼠标事件
         1. onclick：鼠标点击事件（当点击鼠标时触发事件）
         2. ondblclick:鼠标双击事件（当双击鼠标时触发事件）
         3. onmouseup:鼠标弹起事件
         4. onmousedown:鼠标按下事件
         5. onmouseover:鼠标移入事件
         6. onmouseout:鼠标移出事件
         7. onmousemove:鼠标移动事件
      3. 键盘事件
         1. onkeyup:键盘弹起事件
         2. onkeydown:键盘按下事件
         3. onkeypress:键盘按过事件
2. 事件和函数的关系，以及综合运用
3. 函数和循环结合打印九九乘法表
4. JS的编译和执行
   1. javascript代码执行顺序是按照脚本标签<script>出现的顺序来确定的，代码是按从上到下的顺序执行的。如：  
      <script type='text/javascript'>  
       alert('one');  
      </script>  
      <script type='text/javascript'>  
       alert('two');  
      </script>  
      <script type='text/javascript'>  
       alert('three');  
      </script>
   2. 变量在预编译时被赋予一个undefined初值，所以下面代码中，第一个变量name在代码中没有被赋值，所以就延用undefined这个值，下面的name被赋予了Zhangsan，所以第二次输出的是Zhangsan这个字符。如：  
      <script type='text/javascript'>  
       alert(name); //显示undefined  
       var name='Zhangsan';  
       alert(name); //显示Zhangsan  
      </script>
   3. 案例如下：  
      <script type='text/javascript'>  
       test(); //输出Hello World!  
       function test(){　　　　　　   
       alert('hello');　　　　　//声明式函数  
       }  
       test(); //输出Hello World!   
       var test=function(){　　　　//赋值式函数  
       alert('test');  
       }  
       test(); //输出test  
       function test(){　　　　　　//声明式函数  
       alert('Hello World!');  
       }  
       test(); //输出test  
      </script>
      1. 从如上结果中我们知道先是连续两次输出Hello Wrold!，最后连续两次输出test，得出这样的结果是因为javascript并非是完全按照顺序执行的，而是在执行之前先进行一个预编译，预编译时声明式函数（语句定义法）被提取出来，优先执行，而且相同的函数会进行覆盖，再执行赋值式函数。
   4. 下面代码显示hello，再显示hello world！，这是因为javascript中的给个代码块是相互独立的，当脚本遇到第一个<script>标签时，则javascript 解析器会等这个代码块加载完成后，先对它进行预编译，然后再执行它，然后javascript解析器准备解析下一个代码块，由于javascript是按块执行的，所以一个javascript调用下一个块的函数或者变量时，会出现错误。如：  
      <script type='text/javascript'>  
       function test(){  
       alert('hello'); //显示hello  
       }  
       test()  
      </script>  
      <script type='text/javascript'>  
       function test(){  
       alert('hello world!'); //显示hello world！  
       }  
       test()  
      </script>
   5. 虽然javascript是按块执行的，但不同的块却属于相同的全局作用域，不同的块的变量和函数是可以相互使用的，也就是某个块可以使用前面块的变量和函数，却不可以使用它之后的块的变量和函数。如：  
      <script type='text/javascript'>  
       alert(name); //显示undefined  
       var name='Zhangsan';  
       function test(){  
       alert('hello');  
       }  
       fun(); //不能调用下一个块的函数  
      </script>  
      <script type='text/javascript'>  
       alert(name); //可以调用上一个块的变量，显示Zhangsan  
       test(); //可以调用上一个块的函数，显示hello  
       function fun(){  
       alert('fun');  
       }  
      </script>
5. 变量的声明提升
   1. 案例1：  
       var a='hello world';  
       var a;  
       console.log(a);
      1. 变量声明提升（如上代码实质上是）：  
         var a;  
         a = “hello world”;  
         console.log(a);
   2. 案例2：  
      console.log(a);  
      var a='hello world';

第四节：应用

1. 编写任意个数字的求和、差、积、商的函数
2. 编写生成4位数字验证码的函数，并生成10次，同时将结果存入数组

第五节：综合应用

1. 编写一个函数，计算任意两个数字之间所能组成的奇数个数，数字必须是个位数  
   比如： 计算0-3之间能组成的奇数个是01、21、03、13、23、31
2. 某个公司采用公用电话传递数据，数据是四位的整数，在传递过程中是加密的，加密规则如下：每位数字都加上5,然后用除以10的余数代替该数字，再将第一位和第四位交换，第二位和第三位交换，请编写一个函数，传入原文，输出密文