任务17-函数

1.函数声明和函数表达式有什么区别 （*）

函数声明是使用function声明一个函数；

函数表达式是下面这种形式：

```
var printName = function(){
   console.log('Byron');
};
```

两种区别在于函数声明会有一个声明提前的过程，即便是调用写在函数前面也可以正常运行。

函数表达式就没有声明提前的过程。

2.什么是变量的声明前置？什么是函数的声明前置 （**）

如果你写了一个

```
var a=1;
function fn(a){
   console.log(a);
}
fn(a);
```

就相当于

```
var a;
function fn(a){
   console.log(a);
}
a=1;
fn(a);
```

这就是变量的声明前置和函数声明前置。

注意：变量声明在函数声明之前。

3.arguments 是什么 （*）

arguments其实就是相当于函数参数的数组，他默认写在每一个函数内部，可以随时调用。获取参数的信息，还可以修改信息。

```
function fn(a,b,c){
   console.log(arguments);  //[1,2,4]
   console.log(arguments.length); //3
```

```
    arguments[2]=3;
    console.log(c);  //3
}
fn(1,2,4);
```

4.函数的重载怎样实现 （**）

```
function sum(a,b,c){
    var sum=0;
    for(var i=0;i<arguments.length;i++){
        sum+=arguments[i];
    }
    return sum;
}
console.log(sum(1,2));
console.log(sum(1,2,3));
```

5.立即执行函数表达式是什么？有什么作用 （***）

```
(function fn(a){
    console.log(a);
})(3);  //3
```

作用在于立即执行函数内部形成的变量是局部变量；

不会进行提前声明；

不能重复调用。

6.什么是函数的作用域链 （****）

作用域分为全局作用域和局部作用域。

例如如下代码

```
a=1;
function fn1(){
    console.log(a);
    console.log(b);
}
function fn2(){
    var a=2;
    console.log(a);
}
fn1();
```

```
fn2();
```

在fn1里 console.log(a)，查找fn1内部有没有a,结果没有，到上一级即全局变量查找a ,的到结果 1，console.log(b)，在fn1内查找b，没有，到上一级查找，也没有，显示错误。

在fn2里，console.log(a)，查找内部的a ，找到a=2，输出结果2。

问答题

1.以下代码输出什么？（难度**）

```
function getInfo(name, age, sex){
    console.log('name:',name);
    console.log('age:', age);
    console.log('sex:', sex);
    console.log(arguments);
    arguments[0] = 'valley';
    console.log('name', name);
}

getInfo('hunger', 28, '男');
getInfo('hunger', 28);
getInfo('男');
```

结果

```
name:hunger age:28 sex:男 ['hunger', 28, '男'] name:valley
name:hunger age:28  sex:undefined ['hunger', 28,] name valley
name:男 age:undefined sex:undefined ['男'] name valley
```

2.写一个函数，返回参数的平方和？如（难度**）

```
function sumofSquares(){
    var sumofSquares=0;
    for(var i=0;i<arguments.length;i++){
        sumofSquares+=arguments[i]*arguments[i];
    }
    return sumofSquares;
}
sumofSquares(2,3,4); //29
```

3.如下代码的输出？为什么（难度*）

```
console.log(a); //undefined
var a = 1;
console.log(b);  //b is not defined
```

改写成如下

```
var a ;
console.log(a);
a=1;
console.log(b);
```

4.如下代码的输出？为什么（难度*）

```
sayName('world');
    sayAge(10);
    function sayName(name){
        console.log('hello ', name);
    } //hellow world
    var sayAge = function(age){
        console.log(age);
    }; //sayAge is not defined
```

sayName作为声明函数，有提前声明的效果。

sayAge是表达式函数，不能提前声明，因此把sayAge(10)放它前面没作用。

5.如下代码的输出？为什么（难度**）

```
function fn(){}
var fn = 3;
console.log(fn);  //3
```

还原

```
var fn;
function fn(){}
fn=3;
console.log(fn);
```

6.如下代码的输出？为什么（难度***）

```
function fn(fn2){
    console.log(fn2);
```

```
    var fn2 = 3;
    console.log(fn2);
    console.log(fn);
    function fn2(){
        console.log('fnnn2');
     }
  }
  fn(10);
```

还原

```
function fn(fn2){
  var fn2;
  function fn2(){
    console.log('fnnn2');
  }
  console.log(fn2);
  fn2=3;
  console.log(fn2);
  console.log(fn);
}
fn(10)
/* function fn2(){
        console.log('fnnn2');
    }
  3
  function fn(fn2){
  var fn2;
  function fn2(){
    console.log('fnnn2');
  }
  console.log(fn2);
  fn2=3;
  console.log(fn2);
  console.log(fn);
}*/
```

7.如下代码的输出？为什么（难度***）

```
 var fn = 1;
  function fn(fn){
     console.log(fn);
  }
  console.log(fn(fn)); //fn is not a function(...)
```

还原

```
var fn;
function fn(fn){
        console.log(fn);
    }
    fn=1
     console.log(fn(fn));
```

8.如下代码的输出？为什么（难度**）

```
console.log(j); //undefined var j
console.log(i);//undefined  var i
for(var i=0; i<10; i++){
        var j = 100;
    }
console.log(i);//10 全局变量
console.log(j);//100  全局变量
```

9.如下代码的输出？为什么（难度****）

```
    fn();
    var i = 10;
    var fn = 20;
    console.log(i);
    function fn(){
        console.log(i);
        var i = 99;
        fn2();
        console.log(i);
        function fn2(){
            i = 100;
        }
    }
```

还原

```
var i;
var fn;
function fn(){
        var i;
        function fn2(){
            i = 100;
        }
        console.log(i);//undefined
        i = 99;
        fn2();
        console.log(i);//100
```

```
        }
i=10;
fn=20;
fn();
console.log(i);//10
```

10.如下代码的输出？为什么 （难度***** ）

```
var say = 0;
  (function say(n){
      console.log(n);
      if(n<3) return;
      say(n-1);
  }( 10 ));
  console.log(say);
```

结果

10 9 8 7 6 5 4 3 2

0

立即执行函数跟外界无关。