**函数声明和函数表达式的区别?**

1.函数声明的形式是 function xxx(){},函数表达式的形式是var fn = function(){}

2.函数声明会被预编译提升, 因此在同一个作用域内，不管函数声明在哪里定义，该函数都可以进行调用。而函数表达式的值是在JS运行时确定，并且在表达式赋值完成后，该函数才能调用。 关键在于预编译过程函数声明整体提升,而函数表达式是变量提升,初始值为undefined,不可以提前调用.

**Js的预编译机制,预编译会带来哪些坏处?怎么预防?**

1.js执行代码过程大概分为:1.通篇扫描低级语法错误,不执行;2.预编译, 变量声明提升,但不赋值,函数声明整体提升;3.代码开始解释执行.

2.预编译过程中,会将变量的声明提升, 但不对变量进行赋值,变量的初始值为undefined;而函数声明语句会被提升到作用域顶端. 所以函数能够在定义之前执行，并且不会报错。

3.全局预编译:

生成执行期环境上下文GO对象 GO｛｝

找全局变量和暗示全局变量，将变量作为对象的名，值为undefined；

找函数声明，函数的名作为对象的名，值为函数整体。

4.函数预编译:

创建执行期环境上下文AO对象；

找变量声明，将变量名作为AO的属性名，值为undefined；（函数表达式属于变量声明而不是函数声明）

找形参，将形参名作为AO的属性名，值开始为undefined；然后将实参值和形参统一；

在函数体里面找函数声明，整个函数体赋予属性值；

5.为了预防在变量声明之前就使用变量而导致的不易察觉的错误,可以使用const和let声明变量,这样变量必须先声明，然后才可以使用.

**什么是作用域及作用域链?**

1. 每个javascript函数都是一个对象, 函数对象的特殊属性[ [scope] ]，[ [scope] ]是一个隐式的属性，这个属性就是存储作用域的仓库。
2. [ [scope] ]就是我们所说的作用域，其中存储了多个执行期上下文的集合，这个集合呈链式连接。这种链式连接叫做作用域链。

3.全局作用域:变量在程序的任何地方都能被访问;局部作用域: 在固定的代码片段变量才能被访问; 作用域最大的用处就是隔离变量，不同作用域下同名变量不会有冲突。

4.如果变量在当前作用域中没有查到值，就会向上级作用域去查，直到查到全局作用域.

5.知道作用域链有什么好处?如果作用域链越深, [0] => [1] => [2] => [...] => [n]，如果我们调用的是全局变量，它永远在最后一个(这里是第 n 个)，这样的查找到我们需要的变量会引发多大的性能问题？所以，尽量将全局变量局部化，避免作用域链的层层嵌套，所带来的性能问题。

**什么是闭包?**

1.但凡是内部函数被保存到外部，必定会产生闭包。

2. 闭包就是能够读取其他函数内部变量的函数。因为被保存到外部的函数在函数内部定义,所以这个函数的作用域链包含它的父级函数的作用域和全局作用域的变量,

3.闭包就是将函数内部和函数外部连接起来的桥梁。

4.闭包的应用:做缓存数据,利用闭包实现模块化开发,防止变量污染;

利用闭包实现公有变量,实现多个函数操纵一个变量

5.闭包的危害: 导致原有的作用域链不释放，造成内存泄漏。无法被垃圾回收机制回收,造成性能问题.如果不是特意做成一个闭包,尽量应该避免闭包。

**输出结果**

console.log(test);

function test(test){

console.log(test);

var test =234;

console.log(test);

function test(){}

}

test(1);

var test = 123;

依次输出:

ƒ test(test){

console.log(test);

var test =234;

console.log(test);

function test(){}

}

ƒ test(){}

234

**输出结果**

function bar (){

return foo;

foo = 10;

function foo(){

}

var foo = 11;

}

console.log( bar() )

输出:

ƒ foo(){

}

**输出结果**

console.log(bar());

function bar(){

foo = 10;

function foo(){

}

var foo = 11;

return foo;

}

输出:

11