█预处理

●

经典问题

<script>

var a = 1

function xx () {

console.log(a)

var a = 2

}

xx() //undefined

</script>

问题详解

<script>

        console.log(a) //undefined(全局预处理后a的值是undefined)

        var a = 1

        console.log(a) //1 (正常执行a的赋值语句后，a被赋值为1)

        function xx () {

            console.log(a) //函数内部预处理后，此时a的值是undefined

            var a = 2

            console.log(a) //执行a的复制语句后，此时a的值是2

        }

        xx()

        // 执行xx后输出如下

        //undefined

        //2

    </script>

●JS的解析与执行过程

全局预处理阶段

全局执行阶段

函数预处理阶段

函数执行阶段

◎重要说明：JS执行一段代码，首先是全局预处理，然后一行一行全局执行代码。当执行到函数调用时，进入该函数，进行该函数预处理，然后一行一行执行该函数的代码。该函数代码执行完毕后跳出该函数，回到全局代码继续执行。

●全局预处理和执行

◎全局预处理阶段，扫描全局代码的以下内容：

1. 用声明的方式创建的函数
2. 用var定义的变量

◎讲扫描到的函数和方法保存到全局作用域的变量对象中：

1. 用声明的方式创建的函数的值是对该函数的引用
2. 用var定义的变量的值是undefined

例如：

全局代码中有如下内容：

var a = 2;

function b (){alert(1);}

那么预处理扫描后，会把以下内容加到全局变量对象中：

window{

a:undefined,

b:对b函数的引用

}

<script>

alert(g===undefined); //弹出true，因为g是表达式方式声明的函数，在预处理阶段只被认为是var声明的变量，其值被保存为undefined

f(); //会输出1，因为f是用声明的方式创建的函数，被预处理保存在全局变量对象window中了

g(); //会报错，"Uncaught TypeError: undefined is not a function"。因为g是用表达式的方式声明的函数，只是被认为是var声明的变量，所以值被保存为undefined

function f (){

console.log(1);

}

var g = function(){

console.log(2);

}

</script>

◎如果标识符有冲突的处理：

全局预处理会先扫描函数声明，然后再扫描var变量。

处理函数声明有冲突，会覆盖

处理变量有冲突，会忽略（因为值都是undefined）

处理函数和变量的冲突，会以函数为重

<script>

alert(a); //会打印出第二个函数a的代码，因为函数声明冲突会采用覆盖

alert(b); //会打印undefined，因为var变量冲突会忽略，因为值都是undefined

alert(c); //会打印函数c的代码，因为函数声明和变量冲突时，会以函数为重

function a (){

console.log(1);

}

function a (){

console.log(2);

}

var b=3;

var b=5;

function c (){

console.log(3);

}

var c=7;

</script>

◎预处理后的执行

预处理后，开始一行一行执行代码，代码的执行过程中，不断更新全局变量对象中的内容

<script>

//经过预处理,全局变量对象是这样的: window{a:undefined, f:引用, g:undefined}

            alert(a); //打印undefined

        //alert(b); //报错，因为window中没有b

            alert(f); //打印函数f的代码

            alert(g); //打印undefined

            var a =5;

            b = 6;

        //预处理完成后,开始一行一行执行代码,执行到这里时,用执行后的结果更新了window{a:5, f:引用, b:6，g：undefined}

            alert(b); //打印6

            function f(){

                console.log(1);

            }

            var g =function(){

                console.log(2);

            }

        //代码执行到这里后,window中又更新了g的值window{a:5, f:引用, b:6，g：引用}

            alert(g); //打印函数g的引用

</script>

●函数预处理和执行

◎函数预处理阶段

函数每调用一次，就产生一个活动对象，函数执行完毕后活动对象被销毁（闭包除外）

活动对象中包括：

1. 函数的参数（值是undefined，执行时如果传入具体值，则用具体值来更新undefined）
2. 函数内部声明式函数
3. 函数内部var变量
4. arguments

处理冲突与全局处理方式一样

<script>

function f (a, b, c) {

            console.log(a) //1 因为实参是1，并且var冲突忽略，所以忽略了var a = 100, 所以预处理a的值是1

            console.log(b) //函数引用 虽然实参是2， 但是预处理阶段会被函数声明b所覆盖，所以预处理b的值是函数b的引用

            console.log(c) //undefined 调用函数f时没有传第三个参数，所以c的值是undefined，并且var冲突忽略，所以忽略了var c = 200,所以预处理c的值是undefined

            var a = 100

            function b () {

                console.log('b')

            }

            var c = 200

        }

        f(1, 2)

</script>

◎执行函数时，如果函数中不用var定义变量，例如b=2;则执行该行代码后，b被加到全局变量对象中。（没执行时预处理不会添加）

<script>

function f1 () {

            function f2 () {

                a = 99

            }

            f2()

        }

        f1()

        console.log(window.a) //99 在f2函数内声明的a，没有var，则会添加到最外部的变量对象中，即window对象里

</script>