1. 红宝书的DOM1
2. 避免js中的变量冲突

避免全局变量的泛滥

合理使用命名空间

为代码添加必要的注释

使用命名空间法

var GLOBAL={};

GLOBAL.namespace=function(str){

var arr=str.split('.');

var o=GLOBAL;

var start=0;

if(arr[0]=='GLOBAL'){

start=1;

}

else{

start=0;

}

for(var i=start;i<arr.length;i++){

o[arr[i]]=o[arr[i]]||{};

o=o[arr[i]];

}

}

/\*

\*@method 功能A：实现加法运算

\*@author 程序员甲

\*/

void (function(){

var a=1;

var b=2;

GLOBAL.namespace('A.CAT');

GLOBAL.namespace('A.DOG');

GLOBAL.A.CAT.name='mimi';

GLOBAL.A.DOG.name='xiaobai';

GLOBAL.A.CAT.move=function(){};

GLOBAL.A.str1=a;

GLOBAL.A.str2=b;

console.log(a+b);//3

})();

/\*

\*@method 功能C：实现乘法运算

\*@author 程序员丙

\*/

void (function(){

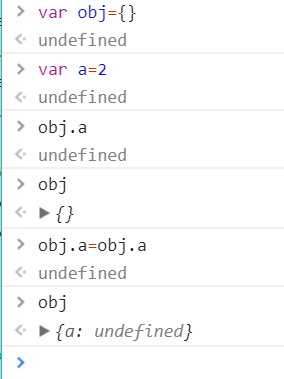
var a=GLOBAL.A.str1;

var b=GLOBAL.A.str2;

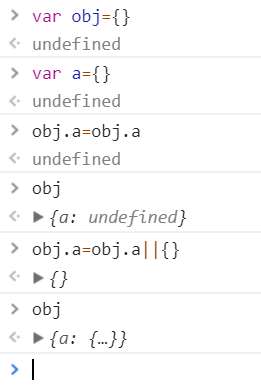
console.log(a\*b);//2

})();

1. obj.a=obj.a||{}即使obj.a不存在在这个等式后obj.a存在了

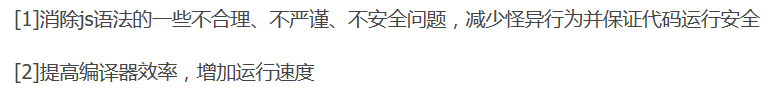


||{} 去空 把undefioned变成{}

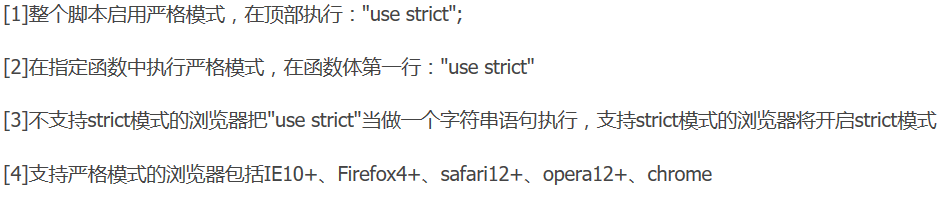


1. js严格模式下的8点规则

严格模式的作用



如何使用严格模式



1. 变量

不允许意外创建全局变量

不能对变量调用delete操作符

"use strict"

str="hello";//Uncaught ReferenceError: str is not defined

var color="red";

delete color;//Uncaught SyntaxError: Delete of an unqualified identifier in strict mode.

1. 对象

不能为只读属性赋值

不能为不可配置的属性使用delete操作符

"use strict"

var person={name:'cook'};

Object.defineProperty(person,'name',{writable:false});

person.name='a';//Uncaught TypeError: Cannot assign to read only property 'name' of object '#<Object>'

Object.defineProperty(person,'name',{configurable:false});

delete person.name;//obj.html:16 Uncaught TypeError: Cannot delete property 'name' of #<Object>

1. 函数

参数必须唯一

修改形参不会影响argument

不允许使用arguments.callee和arguments.caller

"use strict"

function fn(n1,n2){

console.log(n1,n2);//1 1

n1=3;

console.log(arguments[0]);//1

}

fn(1,1);

注：

1. 不允许eval()在包含上下文中创建变量或函数

"use strict"

function fn(){

eval("var x=1");

alert(x);

}

fn();//Uncaught ReferenceError: x is not defined

1. 不运行使用eval和arguments作标识符，也不允许读写他们的值
2. 不允许this的值为null或undefined
3. 不允许使用with语句
4. 不允许使用八进制字面量
5. let

js没有块级作用域，但let声明的变量只在块级内有效

1. for…of

var arr=['a','b','c'];

//for...in循环读取键名

for(let x in arr){

console.log(typeof x);//string

console.log(x);//0 1 2

}

//for...of循环读取键值

for(let y of arr){

console.log(y);//a b c

}

//es6提供了3个新方法用来遍历数组

//keys()-键名 values()-键值

for(let index of arr.keys()){

console.log(typeof index);//number

console.log(index);//0 1 2

}

// for(let elem of arr.values()){

// console.log(elem);//error

// }

for(let elem of Object.values(arr)){

console.log(typeof elem);//string

console.log(elem);//a b c

}

for(let [index,elem] of arr.entries()){

console.log(typeof elem);//string

console.log(index,elem);//0 "a" 1 "b" 2 "c"

}

Array.from可以将类似数组的对象转换为数组

对于普通的对象，for…of不能直接使用，必须部署Iterator接口，但for…in可以遍历键名

let obj={

name:'x',

sex:'woman'

}

for(var key of Object.keys(obj)){

console.log(key,obj[key]);//name x sex woman

}

除了上述方法，还可以使用Generator函数将对象重新包装一下

1. 几种循环的比较

for()麻烦

forEach()不能中途跳出循环

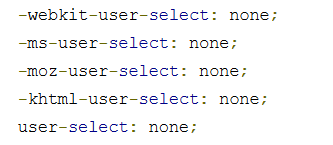
for…in遍历键名 数组键名是数字，甚至遍历手动添加的其他键甚至原型链上的键，以任意顺序遍历键名 主要为遍历对象设计

for…of 没有for…in的缺点，可以跳出循环，提供了遍历所有数据结构的统一接口

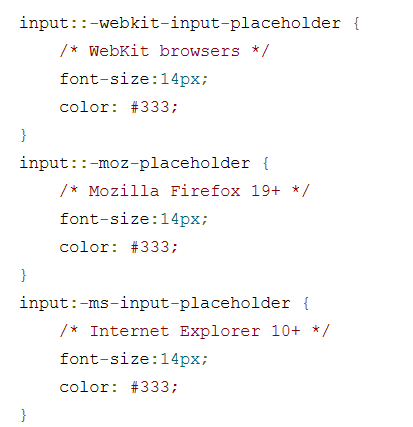
1. 消除transition闪屏



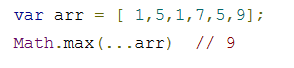
1. 让图文不可复制



1. 改变placeholder的字体颜色大小 只在pc端可以



1. 求数组最大值 最简单



1. 数组去重 最简单

