一、基础知识填空

JS中的数据类型包含哪些(要求详细)以及基本数据类型和引用数据类型的本质区别：\_基本数据类型（number string boolean null undefined）和引用数据类型\_(对象数据类型（[],{},regexp,date...）和函数数据类型)\_\_区别：基本数据类型存储的是值，引用数据类型存储的是堆内存的地址\_\_\_

JS中如何的检测数据类型(写出四种方法)：typeof instanceof constructor Object.prototype.toString.call()\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ JS中判断某个值是否是一个有效数字的方法是：\_\_isNaN()\_\_\_\_\_\_\_\_\_\_\_\_\_

想要接收函数执行的时候传递的参数，都可以如何的处理：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 将其他的数据类型转化为number类型的方式有那些：\_Number() parseInt() parseFloat()\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

二、写出DOM中常用的方法和属性

有一个div:<div class=”w”name=”h”id=”div1”></div>我想获取这一个div你有几种解决办法(不考虑兼容)：document.getElementById(“div1”) document.getElementsByTagName(“div”)[0]

document.getElementsByClassName(“w”)[0] document.getElementsByName(“h”)[0]\_\_\_\_

document.querySelector(“#div1”) document.querySelectorAll(“#div1”)[0]\_

document.querySelector(“.w”) document.querySelectorAll(“.w”)[0]

document.querySelector(“#div1.w”) document.querySelectorAll“.w”)[0] .......

获取当前浏览器屏幕的宽度和高度(兼容所有的浏览器)：document.documentElement.clientWidth || document.body.clientWidth\_\_\_\_\_\_document.documentElement.clientHeight || document.body.clientHeight\_\_\_\_\_\_\_\_\_ 获取上一个哥哥元素节点（兼容所有的浏览器）: \_previousSibling\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

动态创建一个div标签，并且添加到body的最后面位置：\_var oDiv=document.createElement(); document.body.appendChild(oDiv);\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 继续上一题，把刚才创建的克隆一份一模一样的，添加到刚才创建的div前面：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

三、数组的基础知识和应用

我想删除数组中的最后一项，至少写出两种办法：ary.pop() ary[ary.length-1]=null ary.length=ary.length-1;\_\_

我想在数组的末尾增加一个元素，至少写出两种办法：\_\_ary.push(“newStr”) ary.length=ary.length+1 ary[ary.length-1]=“newStr”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

基于原来的数组复制一份一模一样新的数组，不考虑原有数组是否改变的情况，请写出至少三种方案： \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

数组从小到大排序(可以处理10以上的数字)：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 实现找到第n项到第m项的内容，返回一个新的数组(原有数组不变)：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

四、写出下面表达式的运算结果

[]==false：\_\_\_\_\_true\_\_\_\_\_ ![]==false：\_\_\_\_\_\_true\_\_\_\_\_\_ []==![]：\_\_\_\_true\_\_\_\_\_\_ Number(“13px”)==NaN：\_\_\_\_\_\_true\_\_\_\_\_ 5+”3”-2：\_\_\_\_\_\_51\_\_\_\_\_\_\_\_\_\_\_\_\_

If(“3px”\*3){alert(“true”)}else{alert(“false”)}：\_\_false\_\_\_\_\_\_\_\_\_\_

If(“3px”+3){alert(“true”)}else{alert(“false”)}：\_\_\_true\_\_\_\_\_\_\_\_

If（document.body）{alert(“ok”);}else{alert(“no”)}：\_\_\_\_ok\_\_\_\_\_\_\_\_\_\_\_ alert([])：\_\_\_\_””\_\_\_\_\_ null==undefined：\_\_\_\_\_true\_\_\_\_\_\_\_\_\_\_\_

1. <input type="button" value="按钮1"/><input type="button" value="按钮2"/><input type="button" value="按钮3"/><input type="button" value="按钮4"/><input type="button" value="按钮5"/>

var ary=[“red”,”yellow”,”pink”,”green”,”blue”];

以上是页面中的5个按钮，和一个存储五个颜色值的数组，要求实现一个方法bindEvent，循环给5个按钮绑定点击事件，当点击的时候，整个body的背景颜色正好可以和数组中的指定项颜色对应(例如：点击第三个按钮，body的背景颜色”pink”)，请按照你的思路来实现?

Function bindEvent(){

var oInput=document.getElementsByTagName(“input”);

For(var i=0;i<oInput.length;i++){

oInput.onclick=(function(i){

Return function(){

(document.documentElement || document.body).style.backgroundColor=ary[i];

}

})(i)

}

}

六、画图分析

var ary=[1,2,3,4];

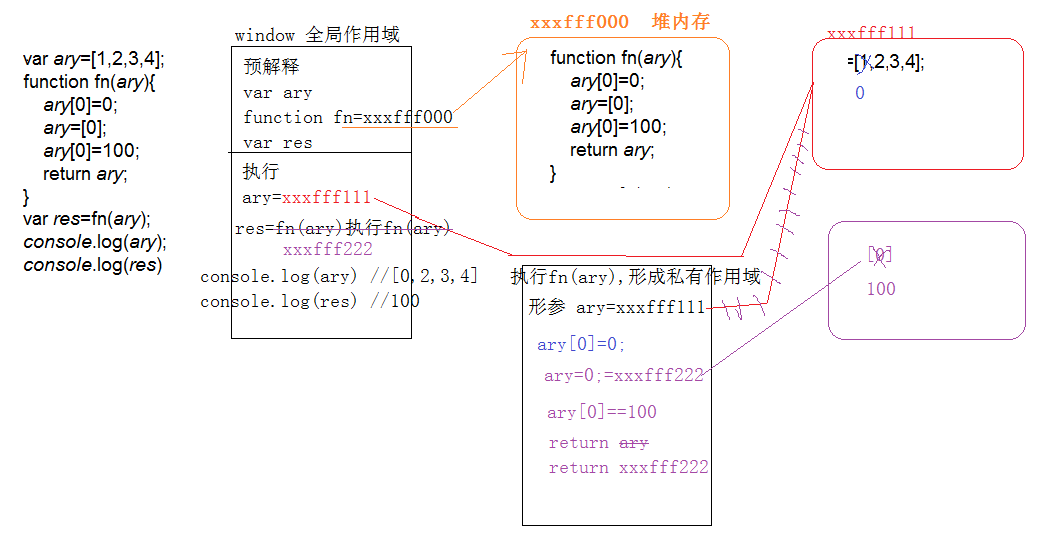
function fn(ary){

ary[0]=0; ary=[0]; ary[0]=100; return ary;

}

var res=fn(ary); console.log(ary); console.log(res); //此题偏难，认真思考(360面试题难度降低版本代码)

**// [0,2,3,4] [100]**



二、选择题：（要求写出分析步骤/画图）

**1、** console.log(a); //undefined

var a=12;

function fn(){

console.log(a); //undefined

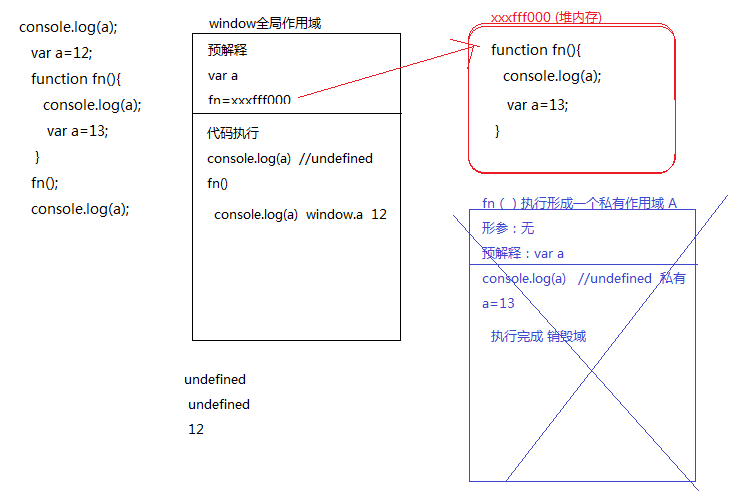
var a=13;

}

fn();

console.log(a); 12

输出的三次分别是多少



A、undefined 12 13 B、undefined undefined 12

C、undefined undefined 13 D、有程序报错

1. console.log(a);

var a=12;

function fn(){

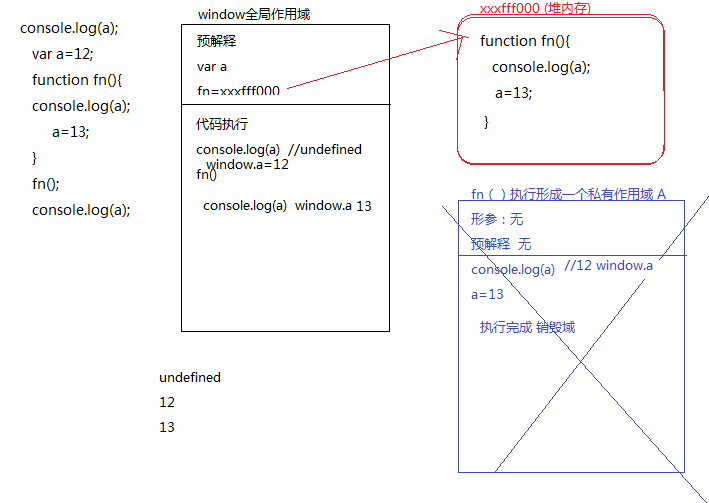
console.log(a);

a=13;

}

fn();

console.log(a);



A、undefined 12 13 B、undefined undefined 12

C、undefined undefined 13 D、有程序报错

1. console.log(a);

a=12;

function fn(){

console.log(a);

a=13;

}

fn();

console.log(a); 输出的三次分别是多少

window下执行

无预解释

console.log(a) //程序报错 a is not defined，以后的代码不再执行

注：作用域链原理：当变量在当前作用域不存在时，逐级向上查找，直到找到，如果至window还没有找到，此时，如果是设置值，则默认为window下的变量，如果获取值，那么就会报错

A、undefined 12 13 B、undefined undefined 12

C、undefined undefined 13 D、有程序报错

1. var foo=1;   
    function bar(){

if(!foo){

var foo=10;

}  
 console.log(foo);

}  
 bar(); 输出的结果是多少

预解释的变态机制之一：不管条件是否成立，都要进行预解释

window下预解释

var foo

bar=xxxfff111

window下从上至下代码执行

foo=1;

bar() 执行，开辟一个私有栈A

A中预解释：不管条件是否成立，都要预解释 var foo

A中代码执行：

if(!foo) ==> if(!undefined) ==>true

foo=10;

console.log(foo) //pring—>10

执行完成，销毁A

A、1 B、10 C、undefined D、报错

1. var n=0;

function a(){

var n=10;

function b(){

n++;

alert(n);

}

b();

return b;

}

var c=a();

c();

alert(n);

弹出三次的结果分别是什么？

window下预解释

var n

a=xxxfff000

var c

window下执行

n=0

var c=a();

a()执行，开辟一个私有栈A

预解释：

var n

b=xxxfff111

执行代码：

A.n=10

b() 执行 开辟一个私有栈B

无形参，无预解释

n++ ==>A.n++ A.n=11

alert(n) //—>print 11

执行完成，销毁B

return b—>return xxxfff111

c=xxxfff111 A中b被c占用 A不销毁

c()执行，形成私有栈C

无形参，无预解释

n++ ==>A.n++ A.n=12

alert(n) //—>print 12

执行完成，销毁C

console.log(a) //window.a=0 print 0

A、1 1 1 B、11 11 0 C、11 12 0 D、11 12 12

1. var a=10,b=11,c=12;

function test(a){

a=1;var b=2;c=3;

}

test(10);

alert(a); alert(b); alert(c); 弹出的三个值分别是多少？

window下预解释

var a,var b,var c

window下代码执行

a=10,b=11,c=12

test(10)执行，开辟私有栈A

形参a=10 —>A.a=10;

预定义：var b

代码执行：

a=1 —>A.a=1;

b=2 —>A.b=2;

c=3 —>window.c=3

执行完成 A销毁

alert(a); 10

alert(b); 11

alert(c); 3

A、1 11 3 B、10 11 12 C、1 2 3 D、10 11 3

7、if(!("a" in window)){

var a=1;

}

alert(a);

预解释变态机制一：不管条件是否成立，都要进行预解释

window下预解释：var a

window下代码执行：

if(!”a” in window) ==>false 跳过

alert(a) //undefined

弹出的a的值是？

A、1 B、undefined C、报错 D、以上答案都不对

8、var a=4;

function b(x,y,a) {

alert(a);

arguments[2]=10;

alert(a);

}

a=b(1,2,3); alert(a); 三次弹出的结果是

window下预解释：

var a

b=xxxfff111

window下执行：

a=4

a=b(1,2,3)

b(1,2,3)执行，开辟私有栈A

形参赋值：A.x=1,A.y=2,A.a=3

alert(a) //—>print 3

arguments[2]=10; A.a=10;

alert(a) //—>print 10

执行完成，销毁A

a=b(1,2,3)=undefined //没有返回值

alert(a) —>print undefined

A、3 3 4 B、3 10 4 C、3 10 10 D、3 10 undefined

9、var foo='hello';

(function(foo){

console.log(foo);

var foo=foo||'world';

console.log(foo);

})(foo);

console.log(foo); 三次分别输出什么？

window下预定义：var foo

window下执行

foo=“hello”

自执行函数执行：开辟一个私有栈A

形参：f00=“hello”

预解释：同名只定义一次

代码执行：

console.log(foo) //hello

f00=f || “world” //foo=“hello”

console.log(foo); //hello

执行完成，销毁A

console.log(foo) //hello

A、hello hello hello B、undefined world hello C、hello world world D、以上答案都不正确

10、var a=9;

function fn(){

a=0;

return function(b){ return b+a++; }

}

var f=fn()

var m=f(5);alert(m);

var n=fn()(5);alert(n);

var x=f(5);alert(x);

alert(a); 弹出的四次结果？

window下预解释：

var a

fn=xxxfff000

var f

var m

var n

var x

window下代码执行

a=9;

f=fn()

执行fn,开辟一个私有栈A

无形参，无预解释，代码执行

a=0; window.a=0

return xxxfff111; —>”function(b){rturn b+a++}”

f=fn()=xxxfff111; A不销毁，被占用

m=f(5);

f(5)执行 开辟一个私有栈 B

形参B.b=5;

return b+a++ B.b+window.a++ —> return 5 window.a=1;

m=5 //print—>5

n=fn()(5)

执行fn,开辟一个私有栈C

无形参，无预解释，代码执行

a=0; window.a=0

return xxxfff111; —>”function(b){rturn b+a++}”

f=fn()=xxxfff111; C不立即销毁

fn()(5)执行 开辟一个私有栈 D

形参D.b=5;

return b+a++ D.b+window.a++ —> return 5 window.a=1;

执行完成，销毁 C D

n=n()(5)=5; //print —>5

x=f(5)

f(5)执行 开辟一个私有栈 E

形参E.b=5;

return b+a++ E.b+window.a++ —> return 6 window.a=2;

执行完成

x=f(5) = 6 //print —>6

alert(a) //window.a=2 print—>2

5 5 6 2

A、6 6 7 2 B、5 6 7 3 C、5 5 6 3 D、以上答案都不正确

三、思考题：

1. document.parentNode的值是什么，为什么
2. document.parentnode的值是什么，为什么 。
3. prototype是谁的属性，它起什么作用？\_\_proto\_\_是谁的属性，它有什么作用？prototype和\_\_proto\_\_有什么联系？
4. hasOwnProperty和in都是用来判断对象的属性的，这两个是怎么用的，有什么区别？
5. 什么是闭包，它有什么作用？
6. **附加题** 画图/分析步骤

1、

function fn(i) {  
 return function (n) {  
 console.log(n + (i++));  
 }  
}  
var f = fn(10);  
f(20);  
fn(20)(40);  
fn(30)(50);  
f(30);

//window下预定义

Fn=xxxfff000

Var f

//window下执行

跳过fn

f=fn(10)

执行fn(10),开辟一个私有作用域 A

形参赋值：i=10;无预解释

执行返回 f=xxxfff111=”function(n){console.log(n+(i++))}“，i被占用,作用域不销毁

执行 f(20),开辟一个私有作用域 B

形参赋值 B.n=20

console.log(n + (i++)); B.n+(A.i++) //print-->30

执行fn(20)(40)

-->执行fn(20)开辟一个私有作用域 C

形参赋值：i=20;无预解释

执行返回 xxxfff222=”function(n){console.log(n+(i++))}“，作用域C不立即销毁

-->执行fn(20)(40) 开辟一个私有作用域 D

形参赋值 D.n=40

console.log(n + (i++)); D.n+(C.i++) //print-->60 作用域D立即销毁 作用域C空闲时销毁

执行fn(30)(50)

-->执行fn(30)开辟一个私有作用域 E

形参赋值：i=30;无预解释

执行返回 xxxfff333=”function(n){console.log(n+(i++))}“，作用域E不立即销毁

-->执行fn(30)(50) 开辟一个私有作用域 F

形参赋值 F.n=50

console.log(n + (i++)); F.n+(E.i++) //print-->80 作用域E立即销毁 作用域F空闲时销毁

执行 f(30),开辟一个私有作用域 G

形参赋值 G.n=30

console.log(n + (i++)); G.n+(A.i++) //print-->41

30 60 80 41

2、

var i = 10;  
function fn() {  
 return function (n) {  
 console.log(n + (++i));  
 }  
}  
var f = fn();  
f(20);  
fn()(20);  
fn()(30);  
f(30);

window下预解释

var i

fn=xxxfff000

var f

window下执行

window.i=10;

f=fn(); =xxxfff111

fn()执行，开辟一个私有栈内存A

形参无，预解释无

执行代码：

return xxxfff111—>”function(n){console.llg(n+(++i))}”

f=xxxfff111,，return 的内容被f占用，A不销毁

f(20)执行 开辟一个私有栈内存B

形参 n=20 ,无预解释

console.log(n+(++i)); B.n+(++window.i) //print—>31 window.i=11

执行完成 销毁B

fn()(20)

执行fn(),形成一个私有栈内存C

形参无，预解释无

执行代码：

return xxxfff222—>”function(n){console.llg(n+(++i))}”

执行完成，作用域C暂不销毁

执行fn()(20) 开辟一个私有栈内存D

形参 n=20 ,无预解释

console.log(n+(++i)); C.n+(++window.i) //print—>32 window.i=12

执行完成 销毁C D

fn()(30)

执行fn(),形成一个私有栈内存E

形参无，预解释无

执行代码：

return xxxfff333—>”function(n){console.llg(n+(++i))}”

执行完成，作用域E暂不销毁

执行fn()(30) 开辟一个私有栈内存F

形参 n=30 ,无预解释

console.log(n+(++i)); E.n+(++window.i) //print—>43 window.i=13

执行完成 销毁C D

f(30)执行 开辟一个私有栈内存G

形参 n=30 ,无预解释

console.log(n+(++i)); G.n+(++window.i) //print—>44 window.i=14

执行完成 销毁G

3、

var num = 10;  
var obj = {num: 20};  
obj.fn = (function (num) {  
 this.num = num \* 3;  
 num++;  
 return function (n) {  
 this.num += n;  
 num++;  
 console.log(num);  
 }  
})(obj.num);  
var fn = obj.fn;  
fn(5);  
obj.fn(10);  
console.log(num, obj.num);

Window下执行

预解释：

var num

var obj

var fn

window下代码从上至下执行

window.num=10;

obj=xxxfff000 —>{num:20}

obj.fn=

自执行函数执行，开辟一个私有栈A,自执行函数的this是window

形参 A.num=obj.num=20;

this.num=num\*3 window.num=A.num\*3=60

num++; A.num=21;

return xxxfff111—>’function(n){this.num+=n;num++,console.log(num)}’

obj.fn=xxxfff111 return后的值被占用，A不销毁

fn=obj.fn=xxxfff111

fn(5) 执行，开辟一个私有栈B 里面的this是window

形参B.n=5;

this.num+=5 window.num=65

num++ A.num=22;

console.log(num) //print—>22

执行完成 B销毁

obj.fn(10)执行，开辟一个私有栈C,里面的this是obj

形参C.n=10;

this.num+=n; obj.num=30;

num++; A.num=23;

console.log(num) //print—>23

执行完成 C销毁

console.log(num,obj.num) //print—>65 30

22

23

65 30

4、

function Fn() {  
 this.x = 100;  
 this.y = 200;  
 this.getX = function () {  
 console.log(this.x);  
 }  
}  
Fn.prototype.getX = function () {  
 console.log(this.x);  
};  
Fn.prototype.getY = function () {  
 console.log(this.y);  
};  
var f1 = new Fn;  
var f2 = new Fn;  
console.log(f1.getX === f2.getX); //false  
console.log(f1.getY === f2.getY); //true  
console.log(f1.\_\_proto\_\_.getY === Fn.prototype.getY); //true  
console.log(f1.\_\_proto\_\_.getX === f2.getX); //false  
console.log(f1.getX === Fn.prototype.getX); //false  
console.log(f1.constructor); //Fn这个函数  
console.log(Fn.prototype.\_\_proto\_\_.constructor); //Object这个函数  
f1.getX(); //100  
f1.\_\_proto\_\_.getX(); // undefined  
f2.getY(); //200  
Fn.prototype.getY(); // undefined

1. 我想通过下面的代码实现选项卡，请问是否可以实现？不能的话，说明原因，那么你有几种办法可以解决? (面试必考题)

