牛客最新前端 JS 笔试百题

战场小包 前端Q 2022-03-23 08:55

前言

前几天空闲时间抓取了牛客最新的笔试题和面试题，想做一下数据统计，展望一下面试中JavaScript的平凡考点和火爆考点，给未来自己的学习指引一下方向。

不多说了，进入题海吧。

emo3.jpg

单选题

JS基础

js概念与类型检测

1. 以下不属于 typeof 运算符返回值的是?

A. "string"

B. "function"

C. "object"

D. "null"

2. 执行以下代码，错误的输出结果是

A. 输入：typeof {"x":1} 输出："object"

B. 输入：typeof 1 输出："number"

C. 输入：typeof [{x:1}] 输出："array"

D. 输入：typeof NaN 输出："number"

3. 可以用typeof来判断的基本类型有

A. undefined
B. null
C. array
D. object

4. 以下不属于JavaScript基本数据类型的是：

A. Boolean
B. undefined
C. Symbol
D. Array

5. 以下关于JavaScript中数据类型的说法错误的是()

A. 数据类型分为基本数据类型和引用数据类型
B. JavaScript一共有8种数据类型
C. Object是引用数据类型，且只存储于堆(heap)中
D. BigInt是可以表示任意精度整数的基本数据类型，存储于栈(stack)中

答案

DCADC

逻辑判断

1. 请选择结果为ture的表达式?

```
A. null instanceof Object
B. null === undefined
C. null == undefined
D. NaN == NaN
```

2. 下列代码结果为 true 的是?

```
A. Symbol.for('a') === Symbol.for('a')
B. Symbol('a') === Symbol('a')
C. NaN === NaN
D. {} === {}
```

3. 根据如下变量，下列表达式中返回值为true的是

```
var a = 1;
var b = [];
var c = '';
var d = true;
```

```
A. (a || b) === true
B. (b && c) === true
C. (c && d) === true
D. (d || a) === true
```

4. 1==true的返回值是true，这句话是否正确?

```
A. T
B. F
```

5. 下面代码输出为true的是?

```
A. console.log([] === []);
B. console.log(undefined == 0);
```

```
C. console.log(undefined == false);
D. console.log(false == '');
```

6. 浏览器环境下，以下打印结果为true的是

```
A. console.log("12" === 12)
B. console.log (NaN === NaN)
C. console.log (typeof(null) === typeof(window))
D. console.log ([1,2,3] === [1,2,3])
```

注意浏览器环境与node环境的差别，比如C选项

7. 以下表达式，正确的是

```
A. Number('a') == Number('a')
B. -1 == true
C. 3 + '2' === 5
D. [] == ''
```

答案

```
CADADCD
```

Math

1. 如何把 7.25 四舍五入为最接近的整数

```
A. Math.round(7.25)
B. Math.ceil(7.25)
C. round(7.25)
D. Math.rnd(7.25)
```

2. 下面哪个选项可以产生0<=num<=10的随机整数

```
A. Math.floor(Math.random()*6)
B. Math.floor(Math.random()*10)
```

C. Math.floor(Math.random()*11)
D. Math.ceil(Math.random()*10)

3. 以下( )表达式产生一个0~7之间(含0,7)的随机整数

A. Math.floor(Math.random()*6)
B. Math.floor(Math.random()*7)
C. Math. floor(Math.random()*8)

答案

A CD(注意D) C

字符串

1. split() 方法用于把一个字符串分割成字符串数组。

A. T
B. F

2. String对象的哪个方法可以寻找子字符串并返回该子字符串位置

A. match()
B. indexOf()
C. search()
D. concat()

答案

A BC

JSON

1. 下面哪一个是JSON数据?

```
A. {name:"xiaoming",age,"student"}
B. {"name":"xiaoming","age":"student"}
C. {"xiaoming","student"}
D. ["xiaoming","student"]
```

2. 下面分别使用 JSON.stringify 方法，返回值 res 分别是

```
const fn = function(){}
const res = JSON.stringify(fn)
const num = 123
const res = JSON.stringify(num)
const res = JSON.stringify(NaN)
const b = true
const res = JSON.stringify(b)
```

```
A. 'function'、'123'、'NaN'、'true'
B. undefined、'123'、undefined、'true'
C. undefined、'123'、'null'、'true'
D. undefined、'123'、'null'、undefined
```

答案

```
BC
```

数组

1. js数组中不会改变原有数组的方法是()

```
A. push
B. concat
C. sort
D. shift
```

2. 下列哪种数组的方法不会修改数组本身

A. slice
B. splice
C. sort
D. unshift

3. JavaScript中需要往数组末尾处添加一个元素，应该使用以下哪个方法：

A. push
B. pop
C. shift
D. unshift

4. 以下js操作Array的方法中不能添加元素的是：

A. push
B. pop
C. unshift
D. splice

5. 数组以下哪个方法会影响原数组？

A. concat
B. splice
C. slice
D. join

6. JavaScript中，下列哪一个Array的方法的返回值类型和其他不同

A. concat
B. shift
C. filter
D. map

7. 如下的Array.prototype上的方法中，那个方法不会改变原有的数组？

A. push
B. slice
C. splice
D. sort

8. 对于一个数字组成的数组 nums，现在需要执行在不改动 nums 的基础上去重操作，返回一个新的无重复元素的数组，以下几段代码能完成这一操作的是()

```
// (1)
const newNums = Array.from(new Set(nums))
```

```
// (2)
const newNums = nums.filter((n, i) => {
    return nums.indexOf(n) === i
})
```

```
// (3)
const newNums = nums.forEach((n, i) => {
    return nums.indexOf(n) === i
})
```

```
// (4)
const newNums = nums.reduce((acc, n, i) => {
    return [].concat(acc, nums.indexOf(n) === i ? n : []
)
})
```

```
A. (1)、(2)、(3)、(4)
B. (1)、(3)、(4)
C. (1)、(2)、(4)
D. (1)、(4)
```

答案

```
BAABB
BBC
```

正则

1. 正则表达式 ^d+[^d]+ 能匹配下列哪个字符串？

```
A. 123
B. 123a
C. d123
D. 123def
```

2. 下面哪个不是RegExp对象的方法

```
A. test
B. match
C. exec
D. compile
```

3. 以下哪项可以去除变量str中的所有空格

```
A. str.replace(`/\s*/g,""`)
B. str.replace(`/^\s|\s$/g,""`)
C. str.replace(`/^\s*/, ""`)
D. str.replace(`/(\s*$)/g, ""`)
```

答案

```
CBA
```

其他

1. 下列函数哪个不是JavaScript的全局函数

```
A. encodeURI
B. parseFloat
C. round
D. eval
```

2. 编写高性能JavaScript，以下描述错误的是

```
A. 遵循严格模式："use strict"
B. 将js脚本放在页面顶部，加快渲染页面
```

C. 将js脚本成组打包，减少请求，尽量减少使用闭包
D. 使用非阻塞方式下载js脚本，最小化重绘(repaint)和回流(reflow)

3. 有关JavaScript中系统方法的描述，错误的是？

A. parseFloat方法：该方法将一个字符串转换成对应的小数
B. isNaN方法：该方法用于检测参数是否为数值型，如果是，返回true，否则，返回false。
C. escape方法： 该方法返回对一个字符串编码后的结果字符串
D. eval方法：该方法将某个参数字符串作为一个JavaScript执行题

4. 下面列出的浏览器，无webkit内核的是()

A. chrome
B. Safari
C. 搜狗浏览器
D. Firefox

5. 下列代码哪个能够实现获取形式为 2017-08-01 形式的日期( )？

```
// A
var formatDate=getDate()
// B
var formatDate = new Date()
// C
var formatDate = function (date) {
    var y = date.getFullYear();
    var m = date.getMonth() + 1;

    var d = date.getDate();
    return y + '-' + m + '-' + d;
};
// D
var formatDate = function (date) {
    var y = date.getFullYear();
    var m = date.getMonth() + 1;
    m = m < 10 ? '0' + m : m;
    var d = date.getDate();
    d = d < 10 ? ('0' + d) : d;
    return y + '-' + m + '-' + d;
};
```

6. 下面哪一项不能最小化重绘(repaint)和回流(reflow)

A. 需要对元素进行复杂的操作时，可以先隐藏(display:"none")，操作完成后再显示
B. 需要创建多个DOM节点时，使用DocumentFragment创建完后一次性的加入document
C. 尽量避免用table布局(table元素一旦触发回流就会导致table里所有的其它元素回流)
D. 尽量不要使用 css 属性简写，如：用border-width, border-style, border-color代替border

答案

CBBDDD

JS深入

this

1. 下列哪种方法不能改变this指向()

A. eval
B. apply
C. bind
D. call

2. 在JavaScript中下面选项关于this描述正确的是

A. 在使用new实例化对象时，this指向这个实例对象
B. 将对象的方法赋值给变量A。执行A()时 该方法中的this指向这个对象。
C. 在函数定义时,this指向全局变量
D. 在浏览器下的全局范围内，this指向全局对象

3. 下面有关JavaScript中call和apply方法的描述，错误的是？

A. call与apply都属于Function.prototype的一个方法，所以每个function实例都有call、apply属性
B. 两者传递的参数不同，call函数第一个参数都是要传入给当前对象的对象，apply不是
C. apply传入的是一个参数数组，也就是将多个参数组合成为一个数组传入
D. call传入的则是直接的参数列表。call 方法可将一个函数的对象上下文从初始的上下文改变为由 thisObj 指

答案

```
AAB
```

作用域(闭包)

1. 内存泄漏是 javascript 代码中必须尽量避免的，以下几段代码可能会引起内存泄漏的有()

```
// (1)
function getName() {
    name = 'javascript'
}
getName()
```

```
// (2)
const elements = {
    button: document.getElementById('button')
};
function removeButton() {
    document.body.removeChild(elements.button);
}
removeButton()
```

```
// (3)
let timer = setInterval(() => {
    const node = document.querySelector('#node')
    if(node) {
        clearInterval(timer)
    }
}, 1000);
```

```
A. (1)、(2)、(3)
B. (2)、(3)
C. (1)、(3)
D. (1)、(2)
```

2. 那个操作不会造成内存泄露

```
```

A. 没有清理的DOM元素引用
B. 被遗忘的定时器
C. 事件侦听没有移除
D. 局部变量不用时，没有设为null ✓

3. 下列关于闭包理解错误的是

A. 增加一定的内存消耗
B. 使用不当可能会导致内存泄漏
C. 可以使用闭包模拟私有方法
D. 闭包会改动对象的原型链 ✓

答案

DDD

原型与继承

1. JavaScript实现继承的方式，不正确的是：

A. 原型链继承
B. 构造函数继承
C. 组合继承
D. 关联继承 ✓

2. 所有对象都有原型

A. T
B. F ✓

3. 以下关于原型链的描述正确的是:

A. 通过原型链继承的属性和对象自己定义的属性等效
B. 通过原型链可以模拟对象的私有属性
C. 在对象上访问不存在的属性时,会依次遍历整条原型链 ✓
D. 所有 JavaScript 中的对象都是位于原型链顶端的 `Object` 的实例

答案

```
DBC
```

其他

1. 以下不属于前端数据存储方式的是?

```
A. jsonp
B. cookie
C. localStorage
D. sessionStorage
```

答案

```
A
```

DOM题

事件流

1. 将A元素拖拽并放置到B元素中，B元素需要做哪项操作()?

```
A. event.preventDefault()
B. event.prevent()
C. event.drag()
D. event.drop()
```

2. 以下不支持冒泡的鼠标事件为( )?

```
A. mouseover
B. click
C. mouseleave
D. mousemove
```

3. 在javascript中，用于阻止默认事件的默认操作的方法是

```
A. stopDeafault()
B. stopPropagation()
C. preventDefault()
D. preventDefaultEven()
```

4. 事件传播的三个阶段是什么

```
目标 -> 捕获 -> 冒泡
冒泡 -> 目标 -> 捕获
目标 -> 冒泡 -> 捕获
捕获 -> 目标 -> 冒泡
```

5. 下面有关 javascript 常见事件的触发情况，描述错误的是?

```
A. onchange：用户改变域的内容
B. onkeypress：某个键盘的键被按下或按住
C. onmousedown：某个鼠标按键被按下
D. onblur：元素获得焦点
```

答案

```
ACCDD
```

DOM遍历

1. 下列哪项不属于DOM查找节点的属性()?

```
A. parentObj.firstChild
B. parentObj.children
C. neborNode.previousSibling
D. neborNode.siblings
```

2. DOM中，给父节点添加子节点的正确方法为()?

```
A. appendChild(parentNode,newNode);
B. append(parentNode,newNode);
C. parentNode.append(newNode);
D. parentNode.appendChild(newNode);
```

3. JavaScript中document.getElementById()返回值的类型为？

```
A. Array
B. Object
C. String
D. Function
```

4. DOM中，给父节点添加子节点的正确方法为()？

```
A. appendChild(parentNode,newNode);
B. append(parentNode,newNode);
C. parentNode.append(newNode);
D. parentNode.appendChild(newNode);
```

答案

```
DDBD
```

其他

1. DOM元素的以下属性改变会导致重排(reflows)的是

```
outline
visiblity
font-size
background-color
```

答案

```
C
```

BOM题

1. setInterval(updateClock,60)的含义是( )？

```
A. 每隔60秒调用一次updateClock()
B. 每隔60毫秒调用一次updateClock()
C. 每隔60分钟调用一次updateClock()
D. 每分钟调用60次updateClock()
```

2. 使用方法( )可以获取到地理位置所在的经纬度?

```
A. Geolocation.watchPosition()
B. Geolocation.getCurrentPosition()
C. Geolocation.getPosition()
D. Geolocation.Position()
```

3. setInterval("alert('welcome');",1000);这段代码的意思是

```
A. 等待1000秒后，再弹出一个对话框
B. 等待1秒钟后弹出一个对话框
C. 每隔一秒钟弹出一个对话框
D. 语句报错,语法有问题
```

答案

```
BBC
```

ES6题

箭头函数

1. 下列对js箭头函数描述错误的是()

```
A. 箭头函数没有原型属性
B. 箭头函数不绑定this，会捕获其所在的上下文的this值，作为自己的this值
C. 箭头函数可以作为构造函数，使用new
D. 箭头函数不绑定arguments，取而代之用rest参数解决
```

2. 关于箭头函数下列说法错误的一项是:

```
A. 函数体内this的指向是定义时所在的对象，而不是使用时所在的对象
B. 箭头函数内不能使用arguments对象
```

```
C. 箭头函数不能使用yield命令
D. 可以使用new创建一个箭头函数的实例
```

答案

```
CD
```

promise

1. 关于将 Promise.all 和 Promise.race 传入空数组的两段代码的输出结果说法正确的是：

```
Promise.all([]).then((res) => {
    console.log('all');
});
Promise.race([]).then((res) => {
    console.log('race');
});
```

```
A. all 和 race 都会被输出
B. all 和 race 都不会被输出
C. all 会被输出，而 race 不会被输出
D. all 不会被输出，race 会被输出
```

2. 以下方案中，不是用于解决回调陷阱的的是：

```
A. Promise
B. Generator
C. async
D. Proxy
```

3. 在 ECMAScript6 中，不属于promise的状态是:

```
A. Pending
B. Pause
C. Fulfilled
D. Rejected
```

答案

```
CDB
```

解构赋值

1. 关于ES6解构表达式,描述正确的是()

```
let [a,b, c,d, e] = "hello";
```

```
A. e = "hello";
B. 其它都为undefined
C. 当中 a = "h", b = "e";
D. 语法报错
```

答案

```
C
```

多选题

JS基础

1. 下面哪些数组方法会改变原数组

```
A. push
B. concat
C. splice
D. map
```

2. 下面可以声明数字的js代码是

```
A. const a = 0xa1
B. const a = 076
C. const a = 0b21
D. const a = 7e2
```

3. 以下属于操作符 typeof 的返回值的是:

```
(1)function
(2) object
(3) null
(4) array
(5) NaN
(6) bigint
(7) regexp
(8) undefined
```

```
A. (1)、(2)、(3)、(4)、(5)、(6)、(7)、(8)
B. (1)、(2)、(3)、(8)
C. (1)、(2)、(8)
D. (1)、(2)、(6)、(8)
```

4. 以下()结果等于字符串string

```
A. typeof 'string'
B. String('string').toString()
C. 'string'.split('').sort().join('')
D. (function(string){return string})('string')
E. JSON.parse('{"string":"string"}').string
```

5. 下面的等式成立的是?

```
A. parseInt(46.8) `==` parseFloat(46.8)
B. NaN `!==` NaN
C. isNaN('abc') `==` NaN
D. typeof NaN `===` 'number'
```

6. 以下哪些选项可以将集合A转化为数组?

```
A. Array.from(A)
B. [].slice.apply(A)
```

```
C. [...A]
D. [].map.call(A, o => o)
```

7. 下列结果返回 true 的是

```
A. null == undefined
B. null === undefined
C. null === null
D. NaN == null
E. NaN === NaN
F. Infinity + 1 !== Infinity
```

答案

```
AC ABD D ABDE BD ABCD AC
```

JS深入

1. 关于以下代码，说法正确的有哪些？

function Person() { } var person = new Person();

```
A. 每一个原型都有一个constructor属性指向关联的构造函数。
B. 每一个对象都有一个prototype属性。
C. Object.getPrototypeOf(person) === Person.prototype
D. person.constructor === Person
```

2. 下列在 JS 时间循环机制中属于微任务(microTask)的是？

```
A. process.nextTick
B. promise
C. setTimeout
D. setInterval
```

答案

```
ACD AB
```

ES6

1. 以下关于let和const的说法中正确的是:

```
A. let声明的变量值和类型都可以改变
B. const声明的常量不可以改变
C. 两者都不存在变量提升，同时存在暂时性死区，只能在声明的位置后面使用
D. const可以先声明再初始化，可以后赋值
```

2. 下面关于Promise说法正确的是(注意“返回结果”的意思包含成功或者失败)

```
A. Promise.all在所有给定的promise都fulfilled后才返回结果
B. Promise.race在给定的promise中，某个fulfilled后才返回结果
C. promise.then的回调函数中，可以返回一个新的promise
D. 对于一个向后台获取数据已经产生结果的promise:p1，再次调用p1.then，不会去重新发起请求获取数据
```

答案

```
ABC CD
```

DOM

1. 下列关于使用 JS 修改元素样式的代码，正确的有哪些？

```
document.body.style.['background-color'] = '#fff'
document.body.style.setProperty('background-color', '#fff')
document.body.style = 'background-color': #fff'
document.body.style.fontSize = '14px'
```

2. 下列方法可用于阻止事件冒泡的有

```
A. event.cancelBubble = true;
B. event.stopPropagation();
```

```
C. event.preventDefault();
D. return false;
```

答案

```
BCD ABD
```

填空题

类型检测

1. 在JavaScript中，有var arr = []; typeof arr的结果为
2. 以下使用 typeof 操作符的代码的输出结果为

```
var x = typeof x
var res = typeof typeof x;
console.log(x, res)
```

3. [typeof null, null instanceof Object]的结果是什么
4. typeof typeof 0
5. JavaScript的typeof运算符的可能结果为array？解释为什么
6. 下面代码的输出结果是多少？

```
var arr = [];
console.log(typeof arr, Object.prototype.toString.call(arr));
```

7. console.log(Object.prototype.toString.call(undefined))

类型转换

1. 表达式 "2"+3+4 的值为
2. console.log('5' + 3, 5 + '3')
3. var a=parseInt(“111办公室”);alert(a);
4. ["0x1", "0x2", "0x3"].map(parseInt) 的结果
5. 在js中执行 1+'1'的结果是？

6. 在js中执行 parseInt('77',40)的结果是?

逻辑判断

1. 请给出 [5<6<3,3<2<4] 代码的运行结果
2. (2<3)||(3<2) 表达式将返回值为
3. console.log(true||false&&false, true&&false||true)的输出结果是?

其他

1. 1 + - + + + - + 1 的结果是
2. ['a', ,'b', ,].length 的结果是

程序题

JS基础

1. 下面两个程序的输出结果分别是?

```
// case 1
function showCase(value) {
    switch(value) {
    case 'A':
        console.log('Case A');
        break;
    case 'B':
        console.log('Case B');
        break;
    case undefined:
        console.log('Case undefined');
        break;
    default:
        console.log('Case default');
    }
}
showCase(new String('A'));
```

```
// case 2
function showCase(value) {
    switch(value) {
    case 'A':
        console.log('Case A');
        break;
    case 'B':
        console.log('Case B');
        break;
    case undefined:
        console.log('Case undefined');
        break;
    default:
        console.log('Case default');
    }
}
showCase(String('A'));
```

2. p标签的的内容会显示什么？

```
<html>
    <body>
        <p id="demo"></p>
        <script type="text/javascript">
            var x = 10;
            var y = "10";
            document.getElementById("demo").innerHTML = Boolean(x == y);
        </script>
    </body>
</html>
```

3. document.write的结果会是什么？

```
function funcA(x){
    var temp = 4;

    function funcB(y){
        document.write( ++x + y + (temp--));
    }

    funcB(5);
}
```

```
funcA(6)
```

4. alert的结果会是多少

```
var varArr = function(i,j,str) {
    return j == 0 ? str : varArr(i,--j,(str+= " " + i[j]));
}
var arr = new Array('apple','orange','peach','lime');
var str = varArr(arr,arr.length,"");
alert(str);
```

5. 下面程序的输出结果是多少?

```
function greetingMaker(greeting) {
    function addName(name) {
        greeting  = greeting.split(' ').reverse().join("-");
        return greeting + " " + name;
    }

    return addName;
}

var daytimeGreeting = greetingMaker("Good Day to you");
alert(daytimeGreeting(name));
```

6. 下面程序的输出结果是多少?

```
String.prototype.GetNum = function() {
    var regEx = /[^\d]/g;
    return this.replace(regEx, '');
};

var str = "a1b2c3";
str = str.GetNum();
alert(str);
```

7. 下面程序的输出结果是多少?

```
function sum(a, b) {
  return a + b;
}
sum(1, "2");
```

8. 下面程序的输出结果是多少？

```
var str = "我非常喜欢编程";
str.length = 3;
console.log(str);
```

9. 下面程序的输出结果是多少？

```
let number = 0;
console.log(number++);
console.log(++number);
console.log(number);
```

10. 下面程序的输出结果是多少？

```
function nums(a, b) {
    if (a > b)
        console.log('a is bigger')
    else
        console.log('b is bigger')
    return a + b
}
console.log(nums(4, 2))
console.log(nums(1, 2))
```

11. 下面程序输出结果是多少？

```
function side(arr) {
    arr[0] = arr[2];
}
function func1(a, b, c = 3) {
    c = 10;
    side(arguments);
    console.log(a + b + c);
}
```

```
function func2(a, b, c) {
    c = 10;
    side(arguments);
    console.log(a + b + c);
}
func1(1, 1, 1);
func2(1, 1, 1);
```

12. 下面代码的输出结果是什么？

```
var a = 3;
var b = new Number(3);
var c = 3;

console.log(a == b);
console.log(a === b);
console.log(b === c);
```

13. 执行下列语句后，a.length的值为？

```
var a = [];
a.push(1, 2);
a.shift(3, 4);
a.concat([5, 6]);
a.splice(0, 1, 2);
```

14. 下面这几段代码分别输出结果是多少？为什么？

```
var a = {}, b = '123', c = 123;
a[b] = 'b';
a[c] = 'c';
console.log(a[b]);
// example 2
var a = {}, b = Symbol('123'), c = Symbol('123');
a[b] = 'b';
a[c] = 'c';
console.log(a[b]);
// example 3
var a = {}, b = {key:'123'}, c = {key:'456'};
a[b] = 'b';
a[c] = 'c';
console.log(a[b]);
```

15. 下面每项的返回值是什么？为什么？

```
null == undefined
0.1 + 0.2 == 0.3
typeof NaN
typeof Function
typeof Object
typeof {}
'a' + 1
'a' - 1
Function instanceof Object
Object instanceof Function
```

16. 下面程序的输出结果是多少？

```
var array = []
for(var i = 0; i < 3; i++) {
    array.push(() => i)
}
var newArray = array.map(el => el())
console.log(newArray)
```

17. 下面程序的输出结果是多少？

```
function a(m, n) {
        var b = function (l) {
            return l <= m ? l * b(l + 1) : 1;
        }

        return b(m - n + 1);
    }

console.log(a(4, 2));
```

18. 下面程序的输出结果是多少？

```
console.log(typeof undefined == typeof NULL);
console.log(typeof function () {} == typeof class {});
```

19. 执行后a和b.age的值分别为

```
var a = 10
var b = {
    age: 11
}
function fn(x,y) {
    --y.age;
    return --x;
}
fn(a,b)
```

20. 下面程序的执行结果是：

```
var number = 4;
var numberFactorial = (function (number){
    return (number === 0)? 1: number* factorial(number-1)
})(number)
console.log(numberFactorial)
```

21. 下面程序的输出结果是：

```
var array = []
for(var i = 0; i < 3; i++) {
    array.push(() => i)
}
var newArray = array.map(el => el())
console.log(newArray)
```

22. 下面程序的输出结果是：

```
function addToList(item, list) {
    return list.push(item)
}
const result = addToList("nowcoder", ["hello"])
console.log(result)
```

23. 下面程序的输出结果是：

```
const first = () => { console.log('first'); return false; }
const second = () => { console.log('second'); return true; }
```

```
console.log( first() && second() );
console.log( second() || first() );
```

24. 下面代码的输出结果是：

```
var s='12ab3cd', arr=s.split(/\d/);
console.log(arr[3],arr[4])
```

25. 下面程序的输出结果是：

```
function getAge(...args) {
  console.log(typeof args);
}

getAge(21);
```

26. 下面程序的输出结果是：

```
var arr=[1,2,3];
arr.push(arr.shift())
console.log(arr[1],arr[2])
```

JS深入

this指向

题目解析：**this指向题目解析及扩展**[3]

关于this还可以看看：**可能是最好的 this 解析了...**

1. 下列程序的输出结果是多少？为什么？

```
var x = 1;

var obj = {
    x: 3,
    fun:function () {
        var x = 5;
        return this.x;
    }
```

```
};

var fun = obj.fun;
console.log( obj.fun(), fun() );
```

2. 下列程序的输出结果是多少？你能理清楚test函数的this指向吗？

```
var a = 5;
 function test() {
    a = 0;
    alert(a);
    alert(this.a);
    var a;
    alert(a);
}
new test();
```

3. 下列程序的输出结果是多少？为什么？

```
function fun () {
    return () => {
        return () => {
            return () => {
                console.log(this.name)
            }
        }
    }
}
var f = fun.call({name: 'foo'})
var t1 = f.call({name: 'bar'})()()
var t2 = f().call({name: 'baz'})()
var t3 = f()().call({name: 'qux'})
```

4. 执行以下代码，输出结果分别是多少？

```
let obj1 = {
    a: 1,
    foo: () => {
        console.log(this.a)
    }
}
```

```
// log1
obj1.foo()
const obj2 = obj1.foo
// log2
obj2()
```

5. 下面程序的输出结果是什么？为什么？

```
const Person = (name="wang",age=10) => {
this.name = name;
this.age = age;
return this.name +' is '+ this.age + 'years old'
}
let result = new Person('zhang',11)
console.log(result)
```

6. 请表述以下代码的执行结果和原因

```
var person = {
  age: 18,
  getAge: function() {
    return this.age;
  }
};
var getAge = person.getAge
getAge()
```

7. 请按顺序写出打印结果，并说明原因。

```
var name = 'global';
var obj = {
    name: 'local',
    foo: function(){
        this.name = 'foo';
    }.bind(window)
};
var bar = new obj.foo();
setTimeout(function() {
    console.log(window.name);
}, 0);
console.log(bar.name);

var bar3 = bar2 = bar;
bar2.name = 'foo2';
```

```
console.log(bar3.name);
```

8. 下面程序的执行结果是：

```
var obj = {
    name:"zhangsan",
    sayName:function(){
        console.info(this.name);
    }
}

var wfunc = obj.sayName;
obj.sayName();
wfunc();
var name = "lisi";
obj.sayName();
wfunc();
```

9. 下面程序的输出结果是：

```
var name='test'
var a = {
    name: 'ass',
    getName: function() {
        return this.name;
    }
}
var b = a.getName;
b();
```

事件循环

1. 下列程序的输出结果分别是多少？为什么？

```
const promiseA = Promise.resolve('a')
promiseA. then((res) => {
    console.log(res)
}).then((res) => {
    console.log(res)
})
const promiseB = Promise.resolve('b')
```

```
promiseB. then((res) => {
    console.log(res)
})
promiseB. then((res) => {
    console.log(res)
})
```

2. 下面程序的输出结果依次是多少?

```
setTimeout(() => {
    console.log(1)
}, 0)

const P = new Promise((resolve, reject) => {
    console.log(2)
    setTimeout(() => {
        resolve()
        console.log(3)
    }, 0)
})

P.then(() => {
    console.log(4)
})
console.log(5)
```

3. 下面程序的输出结果是

```
setTimeout(function(){
    console.log(1);
}, 0)
new Promise(function(resolve){
    console.log(2);
    resolve();
    console.log(3);
}).then(function(){
    console.log(4);
})
console.log(5);
```

4. 下面程序的输出结果是?

```
(async () => {
    console.log(1);
```

```
    setTimeout(() => {
        console.log(2);
}, 0);
await new Promise((resolve, reject) => {
    console.log(3);
}).then(() => {
    console.log(4);
});
    console.log(5);
})();
```

5. 下面程序的输出结果是:

```
new Promise((resolve) => {
    console.log('1')
    resolve()
    console.log('2')
}).then(() => {
    console.log('3')
})
setTimeout(() => {
    console.log('4')
})
console.log('5')
```

6. 下面程序的输出结果是:

```
var p1 = new Promise(function(resolve, reject){
    resolve("2")
})
setTimeout(function(){
    console.log("1")
},10)
p1.then(function(value){
    console.log(value)
})
setTimeout(function(){
    console.log("3")
},0)
```

7. 下面程序的输出结果是:

```
setTimeout(function() {
  console.log('setTimeout');
}, 0);
Promise.resolve().then(function() {
  console.log('promise1');
}).then(function() {
  console.log('promise2');
});
```

8. 请表述以下代码的执行结果和原因

```
setTimeout(function() {
    console.log(1)
},0)
new Promise(function executor(resolve){
    console.log(2)
    for (var i = 0; i<10000; i++) {
        i - 9999 && resolve()
    }
    console.log(3)
}).then(function() {
    console.log(4)
})
console.log(5)
```

9. 在网页中有两个div块，html代码如下

```
<div class="outer">
 <div class="inner"></div>
</div>
```

对应的js代码如下：

```
var outer = document.querySelector('.outer');
var inner = document.querySelector('.inner');

function onClick() {
    console.log('click');
```

```
    setTimeout(function() {
        console.log('timeout');
    }, 0);

    Promise.resolve().then(function() {
        console.log('promise');
    });

    outer.setAttribute('data-random', Math.random());
}

inner.addEventListener('click', onClick);
outer.addEventListener('click', onClick);
```

当点击class为inner的div块时，控制台依次输出结果是什么？10. 下面程序的输出结果是？

```
(async () => {
    console.log(1);
    setTimeout(() => {
        console.log(2);
}, 0);
await new Promise((resolve, reject) => {
    console.log(3);
}).then(() => {
    console.log(4);
});
    console.log(5);
})();
```

11. 下面程序的输出结果是：

```
setTimeout(() => console.log('a'));
Promise.resolve().then(
   () => console.log('b’);
 ).then(
   () => Promise.resolve('c').then(
     (data) => {
       setTimeout(() => console.log('d'));
       console.log('f');
       return data;
```

```
    }
  )
).then(data => console.log(data));
```

12. 下面程序的输出结果是：

```
console.log('one');
setTimeout(function() { console.log('two'); }, 0);
Promise.resolve()
       .then(function() { console.log('three'); })
 console.log('four');
```

13. 下面程序的执行结果是：

```
setTimeout(function () {
    console.log(C)
},0)
console.log('D')
new Promise(function(resolve){
    console.log('E')
    resolve()
    console.log('F')
}).then(function() {
    console.log('G')
})
console.log('H')
```

14. 有一个输出函数定义如下：

```
function log(msg, time) {
  return new Promise((resolve) => {
    setTimeout(() => {
      console.log(msg);
      resolve();
    }, time);
  });
}
```

则下面三段代码输出的结果是:

```
// 第一段代码:
(async () => {
  for (let i = 0; i < 5; i++) {
    await log(i, 1000);
  }
})();
```

```
// 第二段代码:
(async () => {
  [ 1, 2, 3, 4 ].forEach(async (i) => {
    await log(i, 1000);
  });
})();
```

```
// 第三段代码:
(async () => {
  for (const i of [ 1, 2, 3, 4 ]) {
    await log(i, 1000);
  }
})();
```

原型与原型链

关于原型JS: **看完这篇文章，彻底了解 “原型” & “this”**

传送门: **原型与原型链题目解析**[4]

1. 下面程序的输出结果依次是?

```
function Fn1(name) {
    if(name){
    this.name = name;
    }
}
Fn1.prototype.name="jack"
```

```
let a = new Fn1();
console.log('a:', a.name);

function Fn2(name) {
    this.name = name;
}
Fn2.prototype.name="jack"
let b = new Fn2();
console.log('b:', b.name);
```

2. 下面程序的输出结果是?

```
var Foo = (function() {
    var x = 0;
    function Foo() {}
    Foo.prototype.increment = function() {
        ++x;
        console.log(x);
    };
    return Foo;
})();

var a = new Foo();
a.increment();
a.increment();
var b = new Foo();
a.increment();
```

3. 下面程序的输出结果是?

```
var name = 'Jay'
function Person(name){
    this.name = name;
    console.log(this.name)
}
var a = Person('Tom')
console.log(name)
console.log(a)
var b = new Person('Michael')
console.log(b)
```

4. 请表述以下代码的执行结果和原因

```
class A{}
class B extends A{}
const a = new A()
const b = new B()
a.__proto__
b.__proto__
B. __proto__
B. prototype.__proto__
b.__proto__.__proto__
```

5. 请表述以下代码的执行结果和原因

```
function test() {
    getName = function() {
        Promise.resolve().then(() => console.log(0));
        console.log(1);
    };

    return this;
}
test.getName = function() {
     setTimeout(() => console.log(2), 0);
     console.log(3);
};
test.prototype.getName = function() {

     console.log(4);
};
var getName = function() {
     console.log(5);
};
function getName() {

     console.log(6);
}

test.getName();
getName();
test().getName();
getName();
new test.getName();
new test().getName();
new new test().getName();
```

6. 请表述以下代码的执行结果和原因

```
var tmp = {};
var A = function() {};
A. prototype = tmp;

var a = new A();
A. prototype = {};

var b = Object.create(tmp);
b.constructor = A. constructor;

console.log(a instanceof A);
console.log(b instanceof A);
```

7. 下面程序的执行结果是：

```
function Foo(){}
Foo.prototype.z = 3;
var obj = new Foo();
console.info(obj.z)
obj.z = 10;
console.info(obj.z);
delete obj.z;
console.info(obj.z);
```

8. 下面程序的执行结果是：

```
const Book = {
  price: 32
}
const book = Object.create(Book);
book.type = 'Math';
delete book.price;
delete book.type;
console.log(book.price);
console.log(book.type);
```

作用域与预编译

1. 下面的程序会报错吗？如果不会，输出结果分别是多少？

```
function sayHello() {
    console.log(name);
    console.log(age);
```

```
    var name = "Tom";
    let age = 18;
}
sayHello();
```

2. 下面的程序i的打印结果分别是多少？

```
for (var i = 0; i < 3; i++) {
    setTimeout(_ => {
        console.log(i)
    })
}

for (let i = 0; i < 3; i++) {
    setTimeout(_ => {
        console.log(i)
    })
}
```

3. 下面程序的输出结果是：

```
console.log(a);
var a = 'a';
console.log(b);
let b = 'b';
```

4. 下面程序的输出结果是：

```
var foo = "Hello";
(function(){
    var bar = " World";
    alert(foo + bar);
})();
alert(foo + bar);
```

5. 下面程序的输出结果是：

```
var a = 10;
(function () {
    console.log(a)
    a = 5
    console.log(window.a)
```

```
    var a = 20;
    console.log(a)
})()
```

6. 下面代码的输出结果是:

```
const a = 10
function runFunction() {
    const a = 20
    console.log('inside', a)
}
runFunction()
console.log('outside', a)
```

7. 请描述打印结果并说明原因

```
"use strict"
var name = 'Jay'
var person = {
    name: 'Wang',
    pro: {
        name: 'Michael',
        getName: function () {
            return this.name
        }
    }
}
console.log(person.pro.getName)
var people = person.pro.getName
console.log(people())
```

8. 下面程序的结果是:

```
<ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
</ul>
<script>
var elements = document.getElementsByTagName("li");
for (var i=0;i<elements.length;i++){
    elements[i].onclick =function( ){
    alert(i);
```

```
    };
}
```

9. 下面程序的输出结果是

```
compute(10,100);
var compute = function(A,B) {
    console.info(A * B) ;
};
function compute(A,B){
    console.info(A + B);
}
function compute(A,B){
    console.info((A + B)*2);
}
compute(2,10);
```

10. 下面程序的执行结果是：

```
meili()
function meili() {
    console.log("meili")
}
mogu()
var mogu = function() {
    console.log("mogu")
}
```

11. 下面两个代码片段输出结果有什么区别？为什么？

```
// 片段1
check('first');
function check(ars){
    console.log(ars);
}
// 片段2
check('second');
var check= function(ars){
    console.log(ars);
}
```

ES6

对象

1. 下面代码的输出结果是?

```
const student = {name: 'ZhangSan'}
Object.defineProperty(student, 'age', {value: 22})
console.log(student)
console.log(Object.keys(student))
```

generator

1. 下列程序的输出结果是多少？为什么？

```
function * cb(x, y) {
    for(let i = Math.ceil(x); i <= y; i++) {
        yield i;
    }
}

var a = cb(6, 9);
console.log(a.next());
console.log(a.next());
```

扩展运算符

1. 下面程序的输出结果是:

```
function fn(...args) {
  console.log(typeof args);
}
fn(21);
```

promise

```
Promise.reject(0)
       .catch(e => e)
       .catch(e => console.log(e))
```

class

1. 请写出下面ES6代码编译后所生成的ES5代码

```
class Person {
     constructor (name) {
          this.name = name;
     }
     greet () {
          console.log(`Hi, my name is ${this.name}`);
     }
     greetDelay (time) {
          setTimeout(() => {
               console.log(`Hi, my name is ${this.name}`);
          }, time);
     }
}
```

标签模板

1. 下面程序的输出结果是多少?

```
function getPersonInfo (one, two, three) {
    console.log(one)
    console.log(two)
    console.log(three)
}
const person = 'Lydia'
const age = 21
getPersonInfo `${person} is ${age} years old`
```

module

1. 请写出index里面的输出结果

```
// module.js
export default () => "Hello world"
export const name = "nowcoder"
// index.js
import * as data from "./module"
console.log(data)
```

2. 有a.js和b.js两个文件，请写出b文件中代码的输出

```
// a.js
let a = 1
let b = {}
setTimeout(() => {
a = 2
b.b = 2
}, 100)
module.exports = { a, b }

// b.js
const a = require('./a')
console.log(a.a)
console.log(a.b)
setTimeout(() => {
    console.log(a.a)
    console.log(a.b)
}, 500)
```

其他

1. 输出结果是：

```
<div id="box1">
    <div id="box2">
        content
    </div>
</div>
<script>
const $ = document.querySelector.bind(document);
const box1 = $('#box1');
const box2 = $('#box2');

box1.addEventListener('click', () =>{
```

```
    console.log('box1 true');
}, true);

box1.addEventListener('click', () =>{
    console.log('box1 false');
}, false);

box2.addEventListener('click', () =>{
    console.log('box2 true');
}, true);

box2.addEventListener('click', () =>{
    console.log('box2 false');
}, false);
</script>
```

2. 输出结果是:

```
$(function () {
    function fn1( value ) {
        alert( value );
    }
    function fn2( value ) {
        fn1("A");
        return false;
    }
    var callbacks = $.Callbacks();
    callbacks.add( fn1 );
    callbacks.fire( "B" );
    callbacks.add( fn2 );
    callbacks.fire( "C" );
})
```

3. 实现在p元素后添加“Hello World!”，则横线处应使用的方法为( )?

```
<html>
    <head>
        <script type="text/javascript" src="/jquery/jquery.js"></script>
        <script type="text/javascript">
            $(document).ready(function(){
                $("button").click(function(){
```

```
                $("<b>Hello World!</b>")._____("p");
            });
        });
    </script>
</head>
<body>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
    <button>在每个p元素的结尾添加内容</button>
</body>
</html>
```

4. 输出结果是:

```
<div id="box1">
  <div id="box2">
    content
  </div>
</div>
<script>
const $ = document.querySelector.bind(document);
const box1 = $('#box1');
const box2 = $('#box2');
box1.addEventListener('click', () => {
  console.log('box1 true');
}, true);
box1.addEventListener('click', () => {
  console.log('box1 false');
}, false);
box2.addEventListener('click', () => {
  console.log('box2 true');
}, true);
box2.addEventListener('click', () => {
  console.log('box2 false');
}, false);
</script>
```

5. 请选择下面代码输出1的次数

```
var vm = new Vue({
el: '#example',
data: {
```

```
    message: 'Hello'
},
computed: {
    test: function () {
        console.log(1)
        return this.message
    }
},
created: function (){
        this.message = 'World'
        for (var i = 0; i < 5; i++) {
            console.log(this.test)
        }
    }
})
```

关于本文

作者：战场小包

https://juejin.cn/post/7023271065392513038

往期推荐

- 10 个 Node.js 最佳实践：来自 Node 专家的启示
- 2022互联网大厂薪资大比拼
- 第一次拿全年年终奖的前端女程序员的2021

最后

- 欢迎加我微信，拉你进技术群，长期交流学习...
- 欢迎关注「前端Q」,认真学前端，做个专业的技术人...

前端Q

我是winty，专注分享前端知识和各类前端资源，乐于分享...

118篇原创内容

公众号

点个在看支持我吧

阅读原文