JavaScript

1. js数据类型

基本类型：字符串（String）、数字(Number)、布尔(Boolean)、对空（Null）、未定义（Undefined）、Symbol。

引用数据类型：对象(Object)、数组(Array)、函数(Function)。

1. 判断类型
   1. typeof a 或 typeof(a)

返回值为类型的字符串： "string"、"number"、"boolean"、"undefined"、"object"（对象、数组、null）、"function"

* 1. a instanceof b

用来判断引用类型，可区分object中的数组类型，如 [] instanceof Array

a为基本数据类型时，结果 为false

* 1. undefined == null 结果为true

undefined === null 结果为 false

4 == [4] 结果为true

1. 作用域：全局作用域、局部作用域（函数作用域，无块级作用域）

未使用var声明的变量为全局变量，推荐尽量使用var

1. 代码解析机制：先预解析，后逐行执行
2. 垃圾回收策略：标记清除（主流）、引用计数
3. 不存在的变量或函数会报错；不存在的属性或方法，返回undefined
4. 本题考察的是变量的解析机制：

以var声明的变量会预解析

以let声明的变量不会进行预解析  
函数的声明会预解析  
函数表达式不进行预解析

1. 删除对象属性（非原型链）：delete对象.属性

判断是否存在该属性："属性" in 对象

遍历对象的属性值：for (var p in 对象) { console.log(对象[p]) }

1. 函数二象性：既是可以被调用的函数，又是对象
2. 函数定义方式：

字面量（function funcName() {}）：function声明和var赋值表达式（因预解析，只能在赋值语句后调用）

构造函数（new Function("参数1", "参数2", "方法体")）

因无块作用域，所以块中不能使用声明方式定义，只能使用赋值表达式方式定义

1. var fn = function () {}

fn是函数对象本身，可以作为其他函数参数传递

fn ()才会执行该函数

fn ()()生成新的函数对象并执行

1. 函数执行方式：

函数名：funcName()

赋值变量：fn()

定义时执行var fn = function () {}();

匿名函数（需有效字符开头）：

(function () {})();

(function () {}());

!+-~function () {}();

console.log(function () {}());

call方式：funcName.call(this, 参数);

apply方式：funcName.apply (this, 参数数组);

1. 对于链式调用，函数定义时需要返回函数本身：return this;
2. 函数中的参数只有值传递：基本类型传值，对象类型传地址
3. arguments为函数中参数类对象（类似数组，通过下标访问）

arguments.callee属性指向函数本身，便于函数内部递归时使用，但不支持严格模式"use strict";

1. 抛出异常

throw new Error("错误信息");

17 闭包是一个拥有许多变量和绑定了这些变量的环境的表达式（通常是一个函数）

优点：有利于封装，可以访问局部变量

缺点：内存占用浪费空间，会引起内存泄漏，有安全隐患

18 声明对象方式

构造模式不会显示创建对象，将属性赋值给this，不需要return对象

工厂模式在方法内部创建object对象，返回object对象，属性和方法都是赋给object对象

原型（prototype）模式可以让所有实例化对象共享属性及方法

19 原型链

？

\_\_proto\_\_ 和 prototype

原型继承

构造继承：在子类内部构造父类的对象实现继承

特权属性和特权方法

20 对象冒充：将父类的属性和方法一起传给子类作为特权属性和特权方法，不能继承prototype添加的共有属性和共有方法

21 正则表达式

test判断正则是否匹配，返回布尔结果

exec执行正则，返回匹配数组

中文字符unicode匹配范围\u4e00-\u9fa5