

# 50 个JS 必须懂的面试题为你助力金九银十


原创 前端小智 大迂世界 2019-09-01 19:21

南方姑娘  
赵雷 - 赵小雷

作者: Kenmen  
译者: 前端小智  
来源: hackr.io

为了保证的可读性，本文采用意译而非直译。

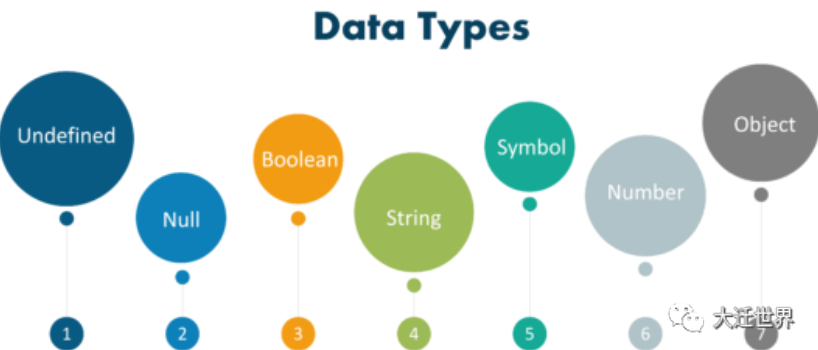
## 问题1: Java和JavaScript有什么不同

Java	JavaScript
Java是一种OOP编程语言。	JavaScript是一种OOP脚本语言。
它创建在虚拟机或浏览器中运行的应用程序。	代码只在浏览器上运行。
需要编译Java代码。	JS代码都是文本的形式。  大迂世界

## 问题2: 什么 Javascript

JavaScript 是一种轻量级的解释型编程语言，具有面向对象的特性，允许各位在其他静态HTML 页面中构建交互性。该语言的通用核心已嵌入Netscape，Internet Explorer和其他Web浏览器中。

## 问题3: JS 支持哪些数据类型



JS 支持的数据类型:

- Undefined
- Null
- Boolean
- String
- Symbol
- Number
- Object

#### 问题4：JavaScript的特性是什么

以下是JS的特性:

- JS 是一种轻量级，解释性编程语言。
- 为了创建以网络为中心的应用程序而设计。
- 补充和集成了 Java
- 补充和集成了 HTML
- 开放和跨平台

#### 问题5：JavaScript是区分大小写

是的，JS是一种区分大小写的语言。关键字、变量、函数名和任何其他标识符必须始终使用一致的大写字母进行使用。

#### 问题6：JS 的优势是什么

以下使用JS的优点:

- **更少的服务器交互** - 在将页面发送到服务器之前，可以验证用户输入,节省了服务器流量，意味着服务器的负载更少
- **立即反馈** - 用户不需要等待页面重新加载来查看是否忘记输入某些内容。
- **增强交互** - 在界面中，当用户使用鼠标悬停或通过键盘激活它们时会做出响应。
- **丰富的接口** - 可以使用JS包含拖放组件和滑块等项，为网站提供丰富的界面。

#### 问题7：如何用JS创建对象

JS支持对象概念，用如下方式创建即可：

```
var emp = {  
  name: "Daniel",  
  age: 23  
};
```

## 问题8：如何用JS创建数组

JS 创建数组也很简单：

```
var x = [];  
var y = [1, 2, 3, 4, 5];
```

## 问题9：JS 中的命名函数是什么以及如何定义：

命名函数在定义后立即声明名称，可以使用 `function` 关键字定义：

```
function named(){  
  // write code here  
}
```

## 问题10：是否可以将匿名函数分配给变量并将其作为参数传递给另一个函数

可以。一个匿名函数可以分配给一个变量，它也可以作为参数传递给另一个函数。

## 问题11：JS中的参数对象是什么&如何获得传递给函数的参数类型

JS 变量 `arguments` 表示传递给函数的参数。使用 `typeof` 运算符，可以获得传递给函数的参数类型。如下：

```
function func(x){  
  console.log(typeof x, arguments.length);  
}  
func(); //==> "undefined", 0  
func(7); //==> "number", 1  
func("1", "2", "3"); //==> "string", 3
```

## 问题12：JS中变量的作用域是什么

变量的作用域是程序中定义它的区域，JS变量只有两个作用域：

- **全局变量** - 全局变量具有全局作用域，这意味着它在JS代码中的任何位置都可见。
- **局部变量** - 局部变量仅在定义它的函数中可见，函数参数始终是该函数的本地参数。

## 问题13：JS 中“this”运算符的用途是什么？

`this` 关键字引用它所属的对象。根据使用位置，它具有不同的值。在方法中，这指的是所有者对象，而在函数中，这指的是全局对象。

## 问题14：什么是回调

回调函数是作为参数或选项传递给某个方法的普通JS函数。它是一个函数，在另一个函数完成执行后执行，因此称为回调。在JS中，函数是对象，因此，函数可以接受函数作为参数，并且可以由其他函数返回。

## 问题15：什么是闭包？举个例子

只要在某个内部作用域内访问在当前作用域之外定义的变量，就会创建**闭包**。它允许你从内部函数访问外部函数的作用域。在JS中，每次创建函数时都会创建闭包。要使用闭包，只需在另一个函数内定义一个函数并暴露它。

## 问题16：列出一些内置方法及其返回的值。

## 问题17：JS中的变量命名约定是什么？

在JS中命名变量时要遵循以下规则：

1. 咱们不应该使用任何JS保留关键字作为变量名。例如，`break` 或 `boolean` 变量名无效。
2. JS 变量名不应该以数字 (`0-9`) 开头。它们必须以字母或下划线开头。例如，`123name` 是一个无效的变量名，但 `123name` 或 `name123` 是一个有效的变量名。
3. JS 变量名区分大小写。例如，`Test` 和 `test` 是两个不同的变量。

## 问题18：TypeOf 运算符是如何工作的

`typeof` 运算符用于获取其操作数的数据类型。操作数可以是文字或数据结构，例如变量，函数或对象。它是一个一元运算符，放在它的单个操作数之前，可以是任何类型。它的值是一个字符串，表示操作数的数据类型。

## 问题19：如何使用 JS 创建 cookie

创建 `cookie` 的最简单方法是为 `document.cookie` 对象分配一个字符串值，如下所示：

```
document.cookie = "key1 = value1; key2 = value2; expires = date";
```

## 问题20：如何使用JS读取cookie

读取 `cookie` 就像写入 `cookie` 一样简单，因为 `document.cookie` 对象的值是 `cookie`。

- `document.cookie` 的值是由分号分隔的 `name=value` 对的列表，其中 `name` 是cookie的名称，`value` 是其字符串值。
- 可以使用 `split()` 方法将字符串分解为键和值。

## 问题21：如何使用 JS 删除 cookie

如果要删除 `cookie` 以便后续尝试读取 `cookie`，则只需将过期日期设置为过去的时间。咱们应该定义 `cookie` 路径以确保删除正确的 `cookie`。如果未指定路径，某些浏览器将不允许咱们删除 `cookie`。

## 问题22：Attribute 和Property之间有什么区别

- **Attribute**——提供关于元素的更多细节，如id、类型、值等。
- **Property** —— 分配给属性的值，如 `type = "text"`，`value = 'Name'` 等。

## 问题23：列出在JS代码中访问HTML元素的不同方式

下面是在JS代码中访问 html 元素的方法列表：

- `getElementById('idname')`: 按 `id` 名称获取元素
- `getElementsByClass('classname')`: 获取具有给定类名的所有元素
- `getElementsByTagName('tagname')`: 获取具有给定标记名称的所有元素
- `querySelector()`: 此函数采用css样式选择器并返回第一个选定元素

## 问题24：JS代码在HTML文件中可以以多少种方式使用

主要有三种：

- 行内
- 内部
- 外部

行内方式：

```
...  
<input type="button" value="点击有惊喜" onclick="javascript:alert('哈哈哈哈哈')">  
...
```

内部方式:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<script type="text/javascript">
//声明一个函数(整个文档都可以使用)
function surprise() {
alert('恭喜你中了一百万')/*弹出框*/
}
</script>
</head>
...
</html>
```

外部方式:

```
...
<!--很多html页面都可以调用js4.js页面-->
<script src="../../js/js4.js" type="text/javascript" charset="utf-8">
...
```

## 问题25: 在JS中定义变量的方法有哪些

在 JS 中声明变量有三种方式:

- **var** – **var** 语句用于声明一个变量, 咱们可以选择初始化该变量的值。例子: **var a =10;** 变量声明在代码执行之前处理。
- **const** - 常量的值不能通过重新赋值来改变, 并且不能重新声明。
- **let** - 语句声明一个块级作用域的本地变量, 并且可选的将其初始化为一个值。

## 问题26: 什么是类型化语言

类型化语言中, 值与值关联, 而不是与变量关联, 它有两种类型:

- **动态**: 在这种情况下, 变量可以包含多种类型, 如在JS中, 变量可以取 **number**, **string** 类型。
- **静态**: 在这种情况下, 变量只能包含一种类型, 就像在Java中声明为 **string** 的变量只能包含一组字符, 不能包含其他类型。

## 问题27: Cookie 与 Local storage 与 Session storage 区别

### 问题28: '==' 和 '===' 区别

**==** : 两边值类型不同的时候, 要先进行类型转换, 再比较。

**===** : 不做类型转换, 类型不同的一定不等。

一言以蔽之: **==** 先转换类型再比较, **===** 先判断类型, 如果不是同一类型直接为 **false** 。

### 问题29: null 和 undefined 区别

**undefined** 是基本数据类型 表示未定义 缺少的意思。

**null** 是引用数据类型, 是对象, 表示空对象

**undefined** 是从 **null** 派生出来的 所以 **undefined===null** 为 **true**

### 问题 30: undeclared 和 undefined 区别?

**undeclared** 的变量是程序中不存在且未声明的变量。如果程序尝试读取未声明变量的值, 则会遇到运行时错误。**undefined**的变量是在程序中声明但未赋予任何值的变量, 如果程序试图读取未定义变量的值, 则返回 **undefined** 的值。

### 问题 31: 列出一些JS框架

JS框架是用JavaScript编写的应用程序框架, 它与控制流中的JS库不同, 一些最常用的框架是:

- Vue
- Angular
- React

### 问题 32: window 与 document 的区别:

**window:**JS 的 window 是一个全局对象，它包含变量、函数、 **history** 、 **location** 。

**document:** **document** 也位于 **window** 之下，可以视为 **window** 的属性。

### 问题 33: innerHTML 和 innerText 的区别

**innerHTML:**也就是从对象的起始位置到终止位置的全部内容,包括Html标签。

**innerText:**从起始位置到终止位置的内容, 但它去除Html标签

### 问题 34: JS中的事件冒泡是什么

事件冒泡是HTML DOM API中事件传播的一种方式，当一个事件发生在另一个元素中的一个元素中，并且两个元素都注册了该事件的句柄时。通过冒泡，事件首先由最内部的元素捕获和处理，然后传播到外部元素。执行从该事件开始，并转到其父元素。然后执行传递给父元素，以此类推，直到body元素。

### 问题 35: NaN 是什么

**NaN** 即非数值（Not a Number）， **NaN** 属性用于引用特殊的非数字值，该属性指定的并不是不合法的数字。

**NaN** 属性 与 **Number.NaN** 属性相同。

**提示:** 请使用 **isNaN()** 来判断一个值是否是数字。原因是 **NaN** 与所有值都不相等，包括它自己。

### 问题 36: JS的原始/对象类型如何在函数中传递？

两者之间的一个区别是，原始数据类型是通过值传递的，对象是通过引用传递的。

- **值传递:** 意味着创建原始文件的副本。把它想象成一对双胞胎:他们出生的时候一模一样，但是双胞胎中的老大在战争中失去了一条腿，而老二却没有。
- **引用传递:** 意味着创建原始文件的别名。当我妈妈叫沙雕的时候，虽然我的名字叫小智，但这并不是说我就突然就克隆了一个自己:我仍然是我，只是可以用不同名字来称呼我而已。

### 问题 37: 如何在JS中将任意基的字符串转换为整数

**parseInt(string, radix)** 将一个字符串 **string** 转换为 **radix** 进制的整数，**radix** 为介于 **2-36** 之间的数,如下:

```
parseInt("4F", 16)
```

### 问题 38: 2+5+ 3的结果是什么



由于 2 和 5 是整数，它们将以数字形式相加。因为 3 是一个字符串，它将与 7 拼接，结果是 73。

### 问题 39: export 和 import 是什么

import 和 export 有助于咱们编写模块化JS代码。使用 import 和 export ，咱们可以将代码拆分为多个文件，如下：

```
//----- lib.js -----</span>
export const sqrt = Math.sqrt;</span>
export function square(x) {</span>
  return x * x;</span>
}
export function diag(x, y) {
  return sqrt(square(x) + square(y));
}

//----- main.js -----</span>
{ square, diag } from 'lib';
console.log(square(5)); // 25
console.log(diag(4, 3)); // 5
```

### 问题40: JS中的“严格”模式是什么以及如何启用?

严格模式是在代码中引入更好的错误检查的一种方法。

- 当使用严格模式时，不能使用隐式声明的变量，或为只读属性赋值，或向不可扩展的对象添加属性。
- 可以通过在文件，程序或函数的开头添加 “use strict” 来启用严格模式

### 问题41: JS 中的 prompt 框是什么

提示框是允许用户通过提供文本框输入输入的框。prompt() 方法显示一个对话框，提示访问者输入。如果您希望用户在输入页面之前输入值，则通常会使用提示框。弹出提示框时，用户必须在输入输入值后单击“确定”或“取消”才能继续。

### 问题42: 下面代码的输出是什么?

```
var Y = 1;
if (eval(function f(){}))
{
  y += typeof F;
}
console.log(y);
```

打印 1undefined 。 if 条件语句使用 eval 求值，因此 eval(function f(){}) 返回函数 f(){ } (为真)。因此，在 if 语句中，执行 typeof f 返回undefined，因为 if 语句代码在运行时执行，而 if 条件中的语句在运行时计算。

### 问题43: call 和 apply有什么区别

`call` 和 `apply` 可以用来重新定义函数的执行环境，也就是 `this` 的指向；`call` 和 `apply` 都是为了改变某个函数运行时的 `context`，即上下文而存在的，换句话说，就是为了改变函数体内部 `this` 的指向。

`call()` 调用一个对象的方法，用另一个对象替换当前对象，可以继承另外一个对象的属性，它的语法是：

```
Function.call(obj[, param1[, param2[, [,...paramN]]]]);
```

**说明：**`call` 方法可以用来代替另一个对象调用一个方法，`call` 方法可以将一个函数的对象上下文从初始的上下文改变为 `obj` 指定的新对象，如果没有提供 `obj` 参数，那么 `Global` 对象被用于 `obj`

`apply()` 和 `call()` 方法一样，只是参数列表不同，语法：

```
Function.apply(obj[, argArray]);
```

**说明：**如果 `argArray` 不是一个有效数组或不是 `arguments` 对象，那么将导致一个 `TypeError`，如果没有提供 `argArray` 和 `obj` 任何一个参数，那么 `Global` 对象将用作 `obj`。

## 问题44：如何在JS中清空数组

有许多方法可以用来清空数组：

方法一：

```
arrayList = []
```

上面的代码将把变量 `arrayList` 设置为一个新的空数组。如果在其他任何地方都没有对原始数组 `arrayList` 的引用，则建议这样做，因为它实际上会创建一个新的空数组。咱们应该小心使用这种清空数组的方法，因为如果你从另一个变量引用了这个数组，那么原始的引用数组将保持不变。

方法二：

```
arrayList.length = 0;
```

上面的代码将通过将其 `length` 设置为 `0` 来清除现有数组。这种清空数组的方式还会更新指向原始数组的所有引用变量。因此，当你想要更新指向 `arrayList` 的所有引用变量时，此方法很有用。

方法三：

```
arrayList.splice(0, arrayList.length);
```

此处方法也行，当然这种清空数组的方法也将更新对原始数组的所有引用。

方法四：

```
while(arrayList.length)
{
    arrayList.pop();
}
```

上面的实现也可以空数组，但通常不建议经常使用这种方式。

### 问题45：以下代码的输出什么

```
var output = (function(x)
{
    delete x;
    return x;
})(0);
console.log(output);
```

打印 `0`。`delete` 操作符用于从对象中删除属性。这里 `x` 不是一个对象，而是一个局部变量，删除操作符不影响局部变量。

### 问题46：以下代码的输出什么

```
var X = { foo : 1};
var output = (function()
{
    delete X.foo;
    return X.foo;
})();
console.log(output);
```

输出 `undefined`。`delete` 操作符用于删除对象的属性。`X` 是一个具有 `foo` 属性的对象，由于它是一个自调用函数，所以咱们将从对象 `X` 中删除 `foo` 属性。这样做之后，当咱们试图引用一个被删除的 `foo` 属性时，结果是 `undefined`。

### 问题47：以下代码的输出什么

```
var foo = function Bar()
{
    return 7;
};
typeof Bar();
```

输出将是 `引用错误`。函数定义只能有一个引用变量作为其函数名。

### 问题49：为什么要将JS源文件的全部内容包装在一个函数中

这是一种越来越普遍的做法，被许多流行的JS库所采用。这种技术围绕文件的整个内容创建一个闭包，最重要的是，它可以创建一个私有命名空间，从而有助于避免不同JS模块和库之间潜在的名称冲突。

该技术的另一个特性是允许为全局变量提供一个简单的别名，这在jQuery插件中经常使用。

## 问题50：JS中的转义字符是什么

JS转义字符使咱们能够在不破坏应用程序的情况下编写特殊字符。转义字符( \ )用于处理特殊字符，如单引号、双引号、撇号和 & 号，在字符前放置反斜杠使其显示。

如：`document.write("I am a \"good\" boy")`

代码部署后可能存在的BUG没法实时知道，事后为了解决这些BUG，花了大量的时间进行log 调试，这边顺便给大家推荐一个好用的BUG监控工具 Fundebug。

原文：<https://hackr.io/blog/javascript-interview-questions>

## 交流

干货系列文章汇总如下，觉得不错点个Star，欢迎 加群 互相学习。

<https://github.com/qq449245884/xiaozhi>

我是小智，公众号「大迂世界」作者，对前端技术保持学习爱好者。我会经常分享自己所学所看的干货，在进阶的路上，共勉！

关注公众号，后台回复福利，即可看到福利，你懂的。

## 延伸阅读

JS 前20个常用字符串方法及使用方式

如何使用useReducer Hook

ES新提案：双问号操作符

JS 可选链的好处

面试 10

面试 · 目录

上一篇

下一篇

看完这几道 JavaScript 面试题，让你与考官对答如流（下）

36 个JS 面试题为你助力金九银十(面试必读)

喜欢此内容的人还喜欢

90后程序员辞职搞灰产，一年获利超700万，结局很刑！  
大迂世界



100 个鲜为人知的 CSS 技巧汇总整理合集  
大迂世界



Vite 为何短短几年内变成这样？  
大迂世界

