【高频面试题】谈谈JavaScript中this的指向机制

一.前言

大家上午好，今天我们来说一下在面试中经常被面试官问到的一个代码输出题，谈一谈JavaScript中的this指向机制。要知道，this这个关键字非常重要，如果不理解它的机制，大部分开发任务都无法完成。

二.区分执行上下文、作用域和词法环境

首先，在了解这个机制之前，我们要简单了解什么叫执行上下文、作用域和词法环境。

执行上下文

官方一点地说，执行上下文就是一个评估和执行JavaScript代码的环境的抽象概念。**任何代码**在JavaScript中运行时，都在执行上下文中运行，我们一般认为它是动态的。

作用域

函数天生就有作用域，且作用域是个可见的属性。作用域是在运行时代码中的某些特定部分中变量，函数和对象的可访问性。换句话说，作用域决定了代码区块中变量和其他资源的可见性。 需要注意的是：作用域是**静态**的，这个在编写代码的时候就确定了。

词法环境

词法环境就比较复杂了，在JavaScript中，每个运行的函数、代码块以及脚本，都有一个被称为**词法环境**的内部的关联对象。

首先，词法环境是静态的。一定要记住，它是静态的，不要和**执行上下文**搞混。一个词法环境由**环境记录器**和一个可能的**引用外部词法环境的空值**组成。我们可以认为，词法环境就是指相应代码块内标识符与变量值、函数值之间的关联关系的一种体现。

三.this的指向机制

我们可以先记住两点：

1：this一定会指向一个对象；

2：this的指向完全取决于函数调用的位置；

关键是在JavaScript语言之中，一切皆对象，那么我们的问题就是，确定this到底指向哪个对象。现在我们看到this的这几种绑定方式，在绑定方式中确认它到底指向哪个对象。

1.默认绑定

非严格环境下，this指向**全局对象**（在浏览器环境下是window对象，在Node.js环境下是global对象）。

```
1 js
2 复制代码
3 function foo() {    var b = 1;    bar()}function bar() {
  console.log(this.b);}foo()
```

这个例子中无论函数声明在哪，在哪调用，由于函数调用时前面并未指定任何对象，这种情况下this指向全局对象window。因此输出 `this.b` 为 `undefined` 。

2.隐式绑定

当函数被一个对象所拥有，再调用时，会触发隐式绑定，此时，this会指向该对象.

```
1 js
2 复制代码
3 function foo() {    console.log(this.a);}var obj = {    a: 2,    foo:
  foo}obj.foo()
```

此时函数调用时前面存在调用它的对象，那么this就会隐式绑定到这个对象上，因此 `this.a` 输出为 `2` 。

3.隐式绑定的隐式丢失

当函数被多个对象链式调用，this指向最终引用函数的对象

4.显式绑定

通过函数的call、apply或bind方法，可以显式地绑定this的值。call和apply立即调用函数，而bind返回一个新的函数，可以稍后调用。

```
1 js
2 复制代码
3 function foo() {    console.log(this.a);}var obj = {    a: 2,}foo.call(obj)
```

此时输出 `2`

```
1 js
2 复制代码
3 function foo(n) {    console.log(this.a, n);}var obj = {    a:
  2,}foo.apply(obj, [100, 200])
```

此时输出 `2 100`

```
1 js
2 复制代码
3 function foo(n) {    console.log(this.a, n);}var obj = {    a: 2,}var bar =
  foo.bind(obj, 100, 200)bar()
```

如果我们给bind输入两个参数，此时n为第一个输入的参数，因此输出为 `2 100`

5.new 绑定

当使用new关键字调用构造函数时，JavaScript会创建一个新对象，并将该对象绑定到this上。

如果在某个上下文中存在多种绑定方式，优先级如下：new绑定 > 显式绑定 > 隐式绑定 > 默认绑定。

需要注意的是，箭头函数没有自己的this值，它会继承外部函数的this值。这在处理回调函数、避免this指向改变等情况下非常有用。