【前端面试】如何判断两个对象是否相等?

前言

在实际项目开发中，判断两个对象是否相等可能是比较常见的需求了，有些小伙伴会使用第三方库实现，有些小伙伴会自己手动实现。不管怎么实现，只能能满足项目需求，那就是好样的。但是可能有些小伙伴如果对 JS 还不够熟悉，他可能就会有疑问：判断相等不是用==比较就可以了吗？答案肯定是错误的，面试官要是听了你这个回答，估计会当场吐血！

今天就来学一学如何比较两个对象是否相等？

学习目标：实现判断两个对象是否相等，即所有键值对相等。

1.使用===来比较?

很多初学者可能第一个想到的就是这种方式。但是我们需要考虑到 object 是引用类型，我们使用===比较时，实际上是在比较对象的引用地址（内存地址），所以即使两个对象键和值都一样，那么它们的引用地址可能不是一样的。

而我们的值类型就可以使用===比较，因为它们之间的比较就确确实实是通过我们看得见的值比较的。

示例代码：

```
<script>
  let obj1 = {
    name: "小猪课堂",
    age: 26,
    sex: "不知道"
  }
  let obj2 = {
    name: "小猪课堂",
    age: 26,
    sex: "不知道"
  }
  console.info("通过===比较两个对象", obj1 === obj2); // false
</script>
```

上段代码中声明了两个对象 obj1 和 obj2，它们的引用地址是不一样的，虽然它们的键值完全相等，但是最终的比较结果还是 false。

我们将两个对象的引用地址改为一样再来试试。

示例代码：

```
<script>
  let obj1 = {
    name: "小猪课堂",
    age: 26,
    sex: "不知道"
  };
  let obj2 = obj1;
  console.info("通过===比较两个对象", obj1 === obj2); // true
</script>
```

上段代码两个对象的引用地址是一样的，所以使用===比较结果为 true。

综上所述：

使用===判断两个对象是否相等时只能判断引用地址是否相等，无法达到我们的目标。

补充：

我们使用 Object.is(obj1,obj2)方法判断两个对象相等时也是判断的引用地址是否相等。

2.另辟蹊径

既然使用===和 Object.is()的方法不能实现我们的目标，那我们只有另辟蹊径了。我们都知道如果是值类型的数据，那么可以使用===来进行判断是否相等，依照此思路，那么我们可以循环我们的对象，依次比较对象中的每个键值对就行了。

我们封装一个 isEqual()方法，传入两个值，必须是对象类型，专门用来判断对象是否相等。

先来熟悉几个 API：

- Object.prototype.toString.call()

> 用来判断数据类型，返回的是一个字符串，包含类型，如”[object Object]”等等

- Object.keys()

> 以一个数组的形式返回对象的键。

- hasOwnProperty()

> 检测一个属性是否是对象的自有属性.

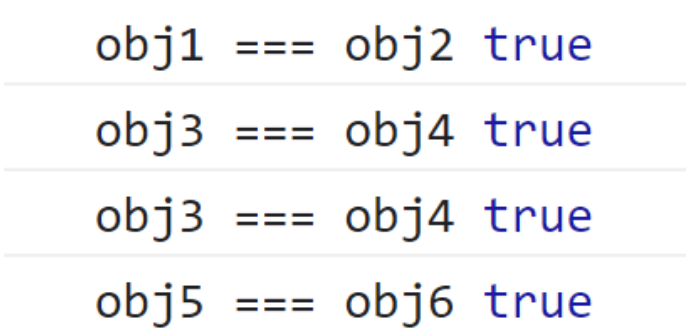
示例代码：

```
<script>
  function isEqual(obj1, obj2) {
    // 判断两个变量是否为对象类型
    let isObj = (toString.call(obj1) === '[object Object]' && toString.call(
    if (!isObj) {
      return false;
    }

```

```
    // 判断两个对象的长度是否相等，不相等则直接返回 fase
    let obj1Keys = Object.keys(obj1);
    let obj2Keys = Object.keys(obj2);
    if (Object.keys(obj1).length !== Object.keys(obj2).length) {
      return false;
    }

    // 判断两个对象的每个属性值是否相等
    for (const key in obj1) {
      // 判断两个对象的键是否相等
      if (obj2.hasOwnProperty.call(obj2, key)) {
        let obj1Type = toString.call(obj1[key]);
        let obj2Type = toString.call(obj2[key]);
        // 如果值是对象，则递归
        if(obj1Type === '[object Object]' || obj2Type === '[object Object]')
          if(!isEqual(obj1[key], obj2[key])) {
            return false;
          }
        } else if (obj1[key] !== obj2[key]) {
          return false; // 如果不是对象，则判断值是否相等
        }
      } else {
        return false;
      }
    }
    return true; // 上面条件都通过，则返回 true
  }
  // 测试用例-1
  let obj1 = {
    name: "小猪课堂",
    age: 26,
    sex: "不知道"
  }
  let obj2 = {
    name: "小猪课堂",
    age: 26,
    sex: "不知道"
  }
  console.info("obj1 === obj2",isEqual(obj1, obj2)); // true
  // 测试用例-2
  let obj3 = {
    name: "小猪课堂",
    age: 26,
    sex: "不知道"
  }
  let obj4 = obj3;
  console.info("obj3 === obj4",isEqual(obj3, obj4)); // true
  obj4.num = 100;
  console.info("obj3 === obj4",isEqual(obj3, obj4)); // true
  // 测试用例-3
  let obj5 = {
```

```
    name: "小猪课堂",
    age: 26,
    sex: {
      type: 1
    }
  }
  let obj6 = {
    name: "小猪课堂",
    age: 26,
    sex: {
      type: 1
    }
  }
  console.info("obj5 === obj6",isEqual(obj5, obj6)); // true
</script>
```

输出结果：



上段代码中我们定义了一个函数用来判断两个对象是否相等，如果传入的参数不是对象，则直接返回 false，如果两个对象的键长度不相等，则直接返回 false，如果两个对象的键对应的值是对象，则使用递归。

上面函数基本上实现了判断两个对象是否相等，但是还是存在一些不足：

- 如果对象中某个键对应的值是数组类型、Data 类型、Set 类型等等，暂未处理。
- 对于某些属性值是 null 或者 undefined 还未做处理。

总体而言：上面的函数基本上能够满足我们常见的需求。

总结

判断对象是否相等相对于判断两个值是否相等要复杂一些，但是其中的基本原理还是比较简单的。我们需要的是理解底层原理，其实主要就是对象中属性值类型的判断。衍生出来的是我们需要了解各个 API 的使用，如 toString.call(obj1)或者 hasOwnProperty 等等，中间就是要知道如何判断一个变量的类型，是 Object 还是 Array，还是 Set、Map、Date 等等。

最后：

最推荐推荐大家使用的还是第三方库，特别是在实际项目中，还是建议使用一些完善的第三方库来实现，毕竟它们将各个方面都考虑到了。比如： lodash 。