【前端面试】你有几种方式实现数字千分位...

前言

这既是一道常见的面试题，也是实际工作中常见的一个需求。这虽然不是一道算法题，但是它是一道发散性思维的题目，想要实现这个功能有很多种方法，这就要看你能够想出几种方法了，本篇文章只列出常见的几种，就相当于抛砖引玉了。

1.实现目标

首先让我们明白这道题目要干什么，接下来我们才好分析思路。

题目描述：

> 假如我们有这样一串数字：43243232.3232143，我们需要把它格式化为
>
> 43,243,232.3232143

输出：

> 12343243

输出：

> 12,343,243

题目总之非常的简单，接下来就让我们一起来实现吧！

2.数组逆序法

数组逆序方法是一种比较容易想到的方法，实现起来也比较简单。

代码如下：

```
function format1(num) {
  // 处理数字，主要是分割出小数和整数
  const str = num.toString().split('.'); // 处理小数
  const intNum = str[0]; // 整数部分
  const fraction = str[1] || ''; // 小数部分

  const arr = intNum.split('').reverse(); // 转化为数组，并且逆序
  let result = ''; // 最终需要返回的数据
  arr.forEach((item, index) => {
    // 不是第一位，且是3的倍数添加“,”
    if (index !== 0 && index % 3 === 0) {
```

```
      result = item + ',' + result;
    } else {
      result = item + result;
    }
  })
  return result + (fraction ? '.' + fraction : ''); // 加上小数部分
}

console.log(format1(2343143.123)); // 2,343,143.123
console.log(format1(2343143)); // 2,343,143
console.log(format1(234)); // 234
```

上段代码中我们有几点需要注意：

- 小数点后面的数字我们不需要分割
- 我们需要将整串数字逆序，因为我们分割的结构是从后往前数，每三位添加一个逗号

3.字符串分割法

相较于数组逆序法，字符串分割法性能要相较于而言高一点，因为数组操作特别是一些打乱顺序得到操作其实是比较消耗性能的。

在JS中，字符串本身就是可以循环的，而且本导题目输出的也是字符串，所以我们可以直接操作字符串。

代码如下：

```
function format2(num) {
  // 处理数字，主要是分割出小数和整数
  const str = num.toString().split('.'); // 处理小数
  const intNum = str[0]; // 整数部分
  const fraction = str[1] || ''; // 小数部分

  const length = intNum.length; // 获取字符串长度
  // 逆序循环
  let result = '';
  for (let index = length - 1; index > + 0; index--) {
    const j = length - index; // 循环到第几位了
    if (j % 3 === 0) {
      if (index === 0) {
        result = intNum[i] + result;
      } else {
        result = ',' + intNum[i] + result;
      }
    } else {
      result = intNum[i] + result;
    }
  }
  return result + (fraction ? '.' + fraction : ''); // 加上小数部分
}
console.log(format1(2343143.123)); // 2,343,143.123
```

```
console.log(format1(2343143)); // 2,343,143
console.log(format1(234)); // 234
```

上段代码中需要注意的点如下：

- 循环字符串时我们需要逆序循环
- 当循环到3的倍数的时候，我们还需要判断当前是不是首位数

4.toLocaleString()

我们可以借助Number.prototype.toLocaleString()方法来实现该题目，toLocaleString()是一个原生的API，使用它来实现题目非常的简单。

我们可以简单看一下这个API做什么的，**官网解释如下：**

> toLocaleString() 方法返回这个数字在特定语言环境下的表示字符串。

官网这段话估计小伙伴们看了也不太明白，所以还是建议去官网好好学习一下这个API：toLocaleString()

代码如下：

```
function format3(num) {
  return num.toLocaleString();
}
console.log(format3(2343143.123)); // 2,343,143.123
console.log(format3(2343143)); // 2,343,143
console.log(format3(234)); // 234
```

5.使用正则

正则也是大家很容易想到的一种方式，不过如果这道题出现在面试中的话，我建议大家还是不要使用正则，因为正则的效率没有大家想象的那么好，当然如果在实际项目中，什么用起来简单用什么。

代码如下：

```
function format4(num){
  return num.toString().replace(/\B(?=(\d{3})+(?!\d))/g, ",");
}
console.log(format3(2343143.123)); // 2,343,143.123
console.log(format3(2343143)); // 2,343,143
console.log(format3(234)); // 234
```

总结

想要实现该需求，方式多种多样，具体使用哪一种方式需要结合实际应用场景，比如说需要性能好一点，或者说需要代码简洁一点等等。出了上面几种，还有很多种方式可以实现，这就需要大

家自己下去挖掘了。