【前端面试】你知道JS异步遍历的方法吗?

前言

JavaScript 异步编程可以说是一大核心知识。我们都知道 JavaScript 是一个单线程语言，单线程机制有什么好处和坏处非常的明显，这里就不多说。异步编程思想的出现，让 JS 这个单线程语言又增添了一丝色彩。

异步 JS 带来了非常多的好处，但是如果你没有真正的掌握，那可能就会遇到一些奇奇怪怪的问题，比如我们今天要说的异步遍历的问题。

问题背景：

我们通常使用 for 循环或者 forEach 进行遍历操作，在遍历过程中通常都是做一些同步操作。但是有一些情况是需要在遍历的时候做一些异步操作，比如遍历发送请求，遍历执行 SQL 语句等等这些异步操作，这种情况就有一些问题出现！

今天就来学习以下 JS 异步遍历。

1.经典面试题

有一道非常经典的面试题，虽然它重点考察的是闭包的问题，但是它和 JS 的异步有着不可分割的关系。

问题：

请问下列代码的打印结果?

示例代码：

```
<script>
  for (var i = 0; i < 5; i++) {
    setTimeout(() => {
      console.info(i)
    }, 1000);
  }
</script>
```

输出结果：

```
5
> |
```

上段代码中，我们正常的理解应该是输出 0、1、2、3、4。但是结果却是输出了 5 个 5。这儿有很大部分原因就是因为 setTimeout 是异步的原因。我们每次 for 循环虽然执行了 setTimeout，但是它里面的函数没有执行，知道 for 循环完毕，才会执行里面的函数，此时的 i 已经变为了 5。

2.进入正题

上一节我们举例了一个比较经典的面试题，目的就是为了让大家先简单理解一下异步。接下来我们模拟实际项目中很容易出现的情况，或者说面试中非常容易被问到的问题。

问题背景：

有一个存放了很多异步操作的数组，你如何将它们依次执行和返回结果？

示例代码：

```
<script>
  // 模拟异步操作，实际可能为发送请求等等
  function createPromise(time, value) {
    return new Promise((resolve) => {
      setTimeout(() => {
        resolve(value);
      }, time);
    })
  }
  let asyncArray = [createPromise(3000, "函数 1"), createPromise(1000, "函数
  // 里面执行异步操作
  async function asyncTest() {
    console.time();
    console.info("start");

    // 执行异步函数数组
    // 请编写你的代码

    console.info("end");
    console.timeEnd();
  }
  asyncTest();
</script>
```

上段代码中 createPromise 函数简单模拟了一个异步操作，然后我们声明了一个 asyncArray 数组，里面存放的是很多异步操作函数，每个函数返回的都是 promise 对象。asyncTest 函数用来执行我们数组里面的异步函数。

看到上面那道题目，应该还是有很多人比较熟悉，但是有些小伙伴如果对异步还不了解，或者对 promise 或者 async/await 不熟悉的话，那么可能有点无从下手，不知道面试官想问什么？

解题思路：

- 循环数组，分别执行函数

- 返回 promise，那可以使用 async/await
- 依次返回结果则需要我们串行执行，可以利用 async/await 解决

通过上面的思路，我们有多种解决方案，一起来实现一下。

3.forEach 循环?

这是很多初学者最容易想到得到一种办法，这道题看似很简单，无非就是循环数组，执行函数嘛！那我们一起来看看结果是什么?

示例代码：

```
<script>
  // 模拟异步操作，实际可能为发送请求等等
  function createPromise(time, value) {
    return new Promise((resolve) => {
      setTimeout(() => {
        resolve(value);
      }, time);
    })
  }
  let asyncArray = [createPromise(3000, "函数 1"), createPromise(1000, "函数
  // 里面执行异步操作
  async function asyncTest() {
    console.time();
    console.info("start");

    // 执行异步函数数组

    // forEach
    asyncArray.forEach(async (item) => {
      const res = await item;
      console.info("执行的函数是：",res);
    });

    console.info("end");
    console.timeEnd();
  }
  asyncTest();
</script>
```

上段代码好像乍一看没有什么问题！利用循环，依次执行异步函数，然后利用 await 阻塞，达到串行目的。我们一起来看看打印结果。

输出结果：

```
start
end
default: 0.35009765625 ms
执行的函数是： 函数2
执行的函数是： 函数1
执行的函数是： 函数3
>
```

结果似乎和我们想的不太一样啊！它似乎是以同步的逻辑执行的，先执行了 console，在执行的异步函数。

造成这种结果的原因很简单，因为 forEach 循环根本没有处理异步操作，它根本不支持异步写法。佛 forEach 的原理很简单，它就是简单的执行了一下我们传入的回调函数，并不会去处理异步情况。

注意：这是很多初学者容易范的错误，一定要注意！出了 forEach 外，类似于 map 等直接传入回调函数的循环方式都无法处理异步。

4.for...of

既然 forEach 循环不行，那我们就换一种能行的循环。for...of 循环可以处理异步操作。

示例代码：

```
for (const item of asyncArray) {
  const res = await item;
  console.info("执行的函数是：",res);
}
```

输出结果：

```
start
执行的函数是： 函数1
执行的函数是： 函数2
执行的函数是： 函数3
end
default: 5001.93603515625 ms
>
```

上段代码就满足我们的要求了，依次输出了 1、2、3，即使每个异步函数的处理时间不一样，但是由于使用了 await 阻塞，所以返回的结果也是按照顺序来的。

到这里看起来 for...of 用来遍历异步数组是没有问题的，但是，真实情况是 for...of 还是存在问题，修改一下我们的代码。

修改代码如下：

```
for (const item of asyncArray) {
  // const res = await item;
  console.info("执行的函数是：",await item.then((res) => {
    console.info(res)
```

```
  }));
}
```

输出结果：

```
start
函数1
执行的函数是： undefined
函数2
执行的函数是： undefined
函数3
执行的函数是： undefined
end
default: 5541.421875 ms
>
```

上段代码中，我们在 then 里面做了一些其它操作，且没有返回值，那么我们的输出就会有问题，所以说我们 for..of 的方法还是不太完美。

5.for...await...of

for...await...of 是 ES2018 的新特性，它可以针对异步集合进行操作，它的使用方法基本上和 for..of 一致，只不过多了一个 await 关键词。

示例代码：

```
for await (const item of asyncArray) {
  console.info("执行的函数是：",item);
}
```

输出结果：

```
start
执行的函数是： 函数1
执行的函数是： 函数2
执行的函数是： 函数3
end
default: 5008.5048828125 ms
>
```

上面的输出结果是想要的，而且我们在循环内部没有使用 await 关键词，它不仅可以暂停循环，还允许你做任何操作。

6.Promise.all

上面使用 for...of 的方式处理异步的时候是串行方式，也就是说上一个执行完后在执行下一个。假如有这么一个场景，用户上传很多张图片，我们需要图片并行上传。如果串行上传的话，那用户等待的时间将会大大增长，这是不科学的。

使用 Promise.all 就可以让我们的异步函数并行执行。

示例代码：

```
console.info(await Promise.all(asyncArray))
```

输出结果：

```
start
▶ (3) ['函数1', '函数2', '函数3']
end
default: 5001.43798828125 ms
>
```

如果你运行了上段代码，你会发现控制台是一起返回了三个结果，而使用 for...of 的时候，控制台是依次打印出的结果。

总结

本篇文章我们重点讲的是如何遍历和处理异步数组，主要介绍了 for...await..of 的方式。大家需要理解串行和并行的概念，当然，本篇文章主要以串行为主，并行的话只介绍了 promise.all。最主要的是大家要掌握 for...await...of 的用法。