手写 flatten 方法

flatten 表示拍平一个数组，将数组里面的嵌套数组展开。

ES6 新增了这样一个方法，`Array.prototype.flat()` 传参表示提取嵌套数组的结构深度，默认值为 1。

```
[1, 2, [3, 4, 5, [6, 7, [8, 9, [10]]]]].flat(1)
// (6) [1, 2, 3, 4, 5, [6, 7, [8, 9, [10]]]]

[1, 2, [3, 4, 5, [6, 7, [8, 9, [10]]]]].flat(2)
// (8) [1, 2, 3, 4, 5, 6, 7, [8, 9, [10]]]

[1, 2, [3, 4, 5, [6, 7, [8, 9, [10]]]]].flat(3)
// (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, [10]]

[1, 2, [3, 4, 5, [6, 7, [8, 9, [10]]]]].flat(4)
// (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

[1, 2, [3, 4, 5, [6, 7, [8, 9, [10]]]]].flat(Infinity)
// (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

上面代码表明，使用 `Infinity` 参数，可以拍平任意嵌套层级的数组。

可以使用 forEach 循环来实现 flat() 方法

```
// forEach 遍历数组会自动跳过空元素
const eachFlat = (arr = [], depth = 1) => {
  const result = []; // 缓存递归结果
  // 开始递归
  (function flat(arr, depth) {
    // forEach 会自动去除数组空位
    arr.forEach((item) => {
      // 控制递归深度
      if (Array.isArray(item) && depth > 0) {
        // 递归数组
        flat(item, depth - 1)
      } else {
        // 缓存元素
        result.push(item)
      }
    })
  })(arr, depth)
  // 返回递归结果
  return result;
}
```

其它方式

```
function deepFlat2(arr) {
    let isDeep = arr.some(item => Array.isArray(item)); // 是否有成员还是数组
    if (!isDeep) {
        return arr;
    }

    // const flat = [].concat.apply([], arr);
    const flat = [].concat(...arr);
    return deepFlat2(flat);
}
```