手写 JS 深拷贝，考虑 set map 函数等

如果不考虑 NaN、Infinity、Symbol、bigint、函数、set、map, 则使用 JSON.stringify 和 JSON.parse 效率很高。否则就不能使用 JSON.stringify，因为会出现很多问题：

- `bigint`：处理不了大整数，TypeError: Do not know how to serialize a BigInt
- `symbol`: symbol 丢失
- `NaN 和 Infinity`: NaN 和 Infinity 变成了 null
- `Set 和 Map`：Set 和 Map 变成了空对象 {}
- `function`: 函数丢失

```
let obj = {
    a: 1,
    b: NaN,
    c: Infinity,
    d: '1',
    e: true,
    f: false,
    g: Symbol('g'),
    h: 10n,
    i: function() {},
    j: [1, 2],
    k: {x: 1, y: 2},
    l: new Set([1, 2]),
    m: new Map([[1, 2]])
};
// bigint 处理不了，会报错

// 去掉 bigint
let obj2 = {
    a: 1,
    b: NaN,
    c: Infinity,
    d: '1',
    e: true,
    f: false,
    g: Symbol('g'),
    i: function() {},
    j: [1, 2],
    k: {x: 1, y: 2},
    l: new Set([1, 2]),
    m: new Map([[1, 2]])
};

JSON.parse(JSON.stringify(obj2));
{
    "a": 1,
    "b": null,
    "c": null,
    "d": "1",
```

```
    "e": true,
    "f": false,
    "j": [
        1,
        2
    ],
    "k": {
        "x": 1,
        "y": 2
    },
    "l": {},
    "m": {}
}
```

```
export function deepClone(obj: any, map = new WeakMap()): any {
    if (typeof obj !== 'object' || obj == null) {
        return obj;
    }
    // 避免循环引用
    const objFromMap = map.get(obj);
    if (objFromMap) {
        return objFromMap;
    }

    let target: any = {};
    map.set(obj, target);

    // Map
    if (obj instanceof Map) {
        target = new Map();
        obj.forEach((v, k) => {
            const v1 = deepClone(v, map);
            const k1 = deepClone(k, map);
            target.set(k1, v1);
        });
    }

    // Set
    if (obj instanceof Set) {
        target = new Set();
        obj.forEach((v) => {
            const v1 = deepClone(v, map);
            target.add(v1);
        });
    }

    // Array
    if (obj instanceof Array) {
        target = obj.map(item => deepClone(item, map));
    }
```

```
    // Object
    for (const key in obj) {
        const val = obj[key];
        const val1 = deepClone(val, map);
        target[key] = val;
    }
    return target;
}
```

Object.assign() 和对象扩展运算符是深拷贝吗

Object.assign() 和对象扩展运算符不是深拷贝，如果对象的所有属性值都是基本类型，它表现为深拷贝。**如果对象的某一属性有引用类型，那么它是浅拷贝，因为改变深层的值，另一个对象也会跟着改变。**