1.给定 csv 文件，转换成对象结构(并提供测试用例)

```
/**
* interface Person {
* name: string;
* age: number;
* parent: Person[];
* children: Person[];
* }
*/
function parseCsv(str) {
        const rowArr = str.trim().split("\n");
        const data = [];
        if (rowArr.length > 0) {
                const headers = rowArr.shift().split(",");
                rowArr.forEach((item) => {
                        const obj = {};
                        obj.parent = [];
                        obj.children = [];
                        const temp = item.split(",");
                        for(let j = 0; j < temp.length; j++){
                                if(headers[j] === 'parent') {
                                        temp[j] && obj.parent.push(temp[j]);
                                } else {
                                        obj[headers[j]] = temp[j];
                                }
                        }
                        data.push(obj);
                })
        }
        const len = data.length;
        for(let i = 0; i < len; i ++) {
                for(let j = 0; j < len; j ++) {
                        if(data[i].parent.includes(data[j].name) && !data[j].children.includes(data[i].name)) {
                                data[j].children.push(data[i].name)
                        }
                        if(data[i].children.includes(data[j].name) && !data[j].parent.includes(data[i].name)) {
                                data[j].parent.push(data[i].name)
                        }
                }
        }
        return data;
}

//测试
const csv = `
name,age,parent
Bob,30,David
David,60,
Anna,10,Bob
`;
console.log(parseCsv(csv))
```

2. 请实现 find 函数，使下列的代码调用正确。

约定：

title 数据类型为 String

userId 为主键，数据类型为 Number

```
var find = function (origin) {
    if (!Array.isArray(origin)) {
        return null
    }
    this.data = origin;
    this.where = function (rule) {
        for (let p in rule) {
            this.data = this.data.filter(function (item) {
                return item[p] && item[p].match(rule[p])
            })
        }
        return this
    };
    this.orderBy = function (key, method) {
        if (method == "desc") {
            return this.data.sort(function (a, b) {
                return b[key] - a[key];
            });
        } else {
            return this.data.sort(function (a, b) {
                return a[key] - b[key];
            });
        }
    };
    return this;
}
// 查找 data 中，符合条件的数据，并进行排序
var result = find(data).where({ 'title': /\d$/ }).orderBy('userId', 'desc');

// 测试
var data = [
{ userId: 8, title: 'title1' },
{ userId: 11, title: 'other' },
{ userId: 15, title: null },
{ userId: 19, title: 'title2' }
];
console.log(result); // [{ userId: 19, title: 'title2'}, { userId: 8,title: 'title1' }];
```

3. 实现一个前端缓存模块，主要用于缓存 xhr 返回的结果，避免多余的网络请求浪费

要求：
生命周期为一次页面打开
如果有相同的请求同时并行发起，要求其中一个能挂起并且等待另外一个请求返回并读取该缓存

【思路：使用 sessionStorage 存储 dataCatch，dataCatch 存储结构如下：

```
dataCatch: {
    'url': {
        data: data,
        status: 'pedding' || 'success' || 'fail',
        successCbs: [],
        failCbs: []
    }
}
```

其中 fail 状态或者没有该 url 请求对象，则发起请求，pedding 状态缓存回调函数，在请求结果成功或者失败时调用，success 状态至今返回存储的结果】

```
function cacheRequest(url, successCb, failCb) {
    let dataCatch = JSON.parse(sessionStorage.getItem('dataCatch')) ||{};
    const current = dataCatch[url]
    if(!current || current.status === 'fail') {
        dataCatch[url] = {
            data: null,
            status: 'pedding',
            successCbs: [],
            failCbs: []
        }
        fetch(url)
        .then(res => {
            dataCatch[url].data = res;
            dataCatch[url].status = 'success'
            successCb(res)
            while(dataCatch[url].successCbs.length) {dataCatch[url].successCbs.shift()(res) }
            dataCatch[url].failCbs = []
        })
        .catch(err => {
            dataCatch[url].status = 'fail'
            failCb(err)
            while(dataCatch[url].failCbs.length) {dataCatch[url].failCbs.shift()(err) }
            dataCatch[url].successCbs = []
        })
        .finally(() => {
            sessionStorage.setItem('dataCatch', JSON.stringify(dataCatch))
        })
    }
    if(current.status === 'pedding') {
        dataCatch[url].successCbs.push(successCb)
        dataCatch[url].failCbs.push(failCb)
        return
    }
    return Promise.resolve().then(() => {successCb(current.data)})
}
```