二面问题

项目相关：

你们组件库的规范怎么样？（他写过公共组件库）

1.需要他介绍eslint

https://eslint.bootcss.com/docs/rules/

2.是否是用了格式化工具 比如prettier

3.可以说commit规范

4.可以说一下git repo

5.可以说一下ci/cd

6.公共组件库嘛 肯定要有UT以保证其正确性 说一下怎么写UT。

基础：

1.说说继承的原理(extend 原理)

2.手动实现一个extend（手写）

https://confluence.syriusrobotics.cn/x/0YLCDQ中的继承章

3.说一下闭包

4.说一下js中this的指向

5.是否能手动实现一个after()，before()方法

```
Function.prototype.after = function (fn) {
        let self = this;
        return async function () {
            let ret = await self.apply(this, arguments);
            if (ret === "nextSuccessor") {
                return fn.apply(this, arguments);
            }
            return ret;
        }
    };
    Function.prototype.before = function (fn) {
        let self = this;
        return function () {
            fn.apply(this, arguments);
            return self.apply(this, arguments);
        }
    }
```

实战（证明代码能力要么基础好 要么经验丰富）：

三选一：

实现JS限流调度器，方法add接收一个返回Promise的函数，同时执行的任务数量不能超过两个（经验丰富）

```
class Scheduler {
    async add(promiseFunc: () => Promise<void>): Promise<void> {
    }
}
const scheduler = new Scheduler()
const timeout = (time) => {
    return new Promise(r => setTimeout(r, time))
}
const addTask = (time, order) => {
    scheduler.add(() => timeout(time))
        .then(() => console.log(order))
}
addTask(1000, 1)
addTask(500, 2)
addTask(300, 3)
addTask(400, 4)
// log: 2 3 1 4
```

实现:

```
class Scheduler {
  constructor(maxNum) {
    this.taskList = [];
    this.count = 0;
    this.maxNum = maxNum;
  }
  async add(promiseCreator) {
    if (this.count >= this.maxNum) {
      await new Promise((resolve) => {
        this.taskList.push(resolve)
      })
    }
    this.count ++;
    const result = await promiseCreator();
    this.count --;
    if (this.taskList.length > 0) {
      this.taskList.shift()();
    }
    return result;
  }
}
```

基础好：

1.简单题（可以用多重循环或者单调栈）

https://leetcode-cn.com/problems/next-greater-element-i/

2.中等题（公共祖先）

https://leetcode-cn.com/problems/lowest-common-ancestor-of-a-binary-tree/