**小鹅通前端实习生笔试题**

1. 使用JS编写判断函数：输入值为变量a和变量b，变量皆为对象；输出值为布尔值，true则为输入变量的值完全一致

function isEqual(a, b) {

    if (typeof a !== 'object' || typeof a !== 'object') {

        return false

    }

    const propsA = Object.keys(a)

    const propsB = Object.keys(a)

    if (propsA.length !== propsB.length) {

        return false;

    }

    for (let props of propsA) {

        if (typeof a[props] === "object" && typeof b[props] === "object") {

            if (!isEqual(a[props], b[props])) {

                return false;

            }

        } else if (a[props] !== b[props]) {

            return false;

        }

    }

    return true;

}

2、使用JS编写排序函数（冒泡或快排皆可）：输入随意数字数组；输出值为数组，数组内数字按从大到小排列

function quickSort(arr) {

    if (arr.length <= 1) {

        return arr;

    }

    const mid = Math.floor(arr.length / 2)

    const key = arr[mid];

    const left = [];

    const right = [];

    for (let i = 0; i < arr.length; i++) {

        if (arr[i] > key) {

            right.push(arr[i]);

        } else if (arr[i] < key) {

            left.push(arr[i]);

        }

    }

return [...quickSort(right), key, ...quickSort(left)];

}

1. vue或react等前端框架的数据绑定响应，实现原理大致是怎样的？

答：在vue2中，双向数据绑定是通过数据劫持和发布订阅模式实现的。

首先是基于Object.defineProperty()来追踪对象的属性的变化，它会对数据对象进行遍历，包括子属性对象的属性，对属性都加上setter和getter，当访问对象属性的时候，就会触发getter，当修改对象属性的时候，就会触发setter，从而监听到数据的变化。

其次，实现一个解析器Compile，解析Vue 模板指令，将模板中的变量都替换成数据，然后初始化渲染页面视图，并将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知，调用更新函数进行数据更新。

再次，实现一个订阅者Watcher，订阅者是通过Oberserve和Compile之间通信的桥梁，主要的任务是订阅监听器中的属性值的变化，当收到属性值变化的消息的时候，触发解析器中对应的更新函数。

最后，实现一个订阅器，订阅器采用发布订阅设计模式，用来收集订阅者Watcher，对监听器和订阅者进行统一的管理

4、有用户访问网站页面，出现了页面白屏无法访问的情况，现需要排查是什么导致的。用户这个问题可能是怎么导致的？可以通过什么方法排查确认？

**答：**

**导致的原因：**

1：页面中JS的代码错误，导致无法正常的进行加载

2：网络异常，网速过慢，资源太大导致加载速度过慢，导致白屏时间过长

3：页面中依赖的文件比如CSS，JS文件等加载错误或路径错误，导致页面无法正常加载

4：所依赖的服务端出现了错误

**排查方法：**

1：打开浏览器的开发者工具来查看控制台输出、网络请求和其他相关信息。

2：查看网速，网络请求是否失败

3：在开发者工具中查看页面的资源加载状况

4：与后端人员进行沟通，查看服务端的日志，确定是否有服务器错误或异常的情况

5、有直播间聊天室页面，同时可支撑1w人同时在线，用户进去后可实时聊天，请分析需要用到什么技术？实现过程中需要注意什么问题？（可列举技术点及问题/解决思路)

**答：**选择高性能的，支持大规模并发的聊天需求的后端技术，比如Java，Node.js等

使用WebSocket协议来实现实时双向通信。使用WebSocket来保持客户端与服务器的实时连接。客户端可以通过监听WebSocket事件来接收实时的聊天消息。

使用Nginx或负载均衡器，将流量分发到多个后端服务器上，达到负载均衡技术

将聊天记录存储到数据库中，以便后续的消息检索和历史记录展示

**需要注意的地方：**

网络的延迟问题，大规模并发的通信可能会导致网络延迟，需要合理的网络架构和优化来降低延迟。

资源消耗问题，大规模的并发通信会占用大量的服务器资源，因此，需要合理地进行资源地分配

安全性问题，需要对用户身份进行多重地验证和权限控制

防止恶意地攻击，对用户输入的消息进行合法性检查，防止XXS攻击