**VUE前端工程师笔试题目-最终**

**谢愫15118173775**

**1-2题目请直接用文字作答：**

1、web storage和cookie的区别

**答：**

web storage分为sessionStorage以及localStorage，是H5新增加的属性，他们优化了cookie的许多问题。

**Cookie：**

a、他是存储在客户端的一种键值对数据，因为http本身他是无状态的，为了实现保持状态的功能，从而出现了cookie。

b、他可以从客户端和服务端生成，服务端通过响应报文的形式返回给客户端，客户端读取响应头的set-cookie拿到后存储。

**缺点：**

1. 容量不大。cookie在浏览器的存储上限是 4KB 。

2. 过多的带 cookie请求会带来性能浪费。同域名下的请求都会携带同域名的 cookie，哪怕这个请求不需要 cookie，比如一些对 css、图片的请求。

3. HTTP 是明文传输，cookie传递存在安全性问题。

**web storage：**

web storage解决了cookie的这几个缺点

1. 一样是以键值对的方式存储，但是存储上线是5MB。

2. 只在客户端使用，不会发送给服务端。也就是说需要开发者在发送请求时自行添加。

sessionStorage以及localStorage的区别：

1. localStorage是长期存储，需要手动清除；sessionStorage顾名思义，是会话存储，会话结束后自动清除。（一般情况下浏览器关闭即会话关闭）

2. localStorage可在同源页面里共享；sessionStorage在同一标签页下的同源页面里共享（iframe打开的页面可共享）

2、Vuex的工作流程，以及它的作用，使用场景

答：

**未曾阅读过VueX的源码，讲讲与之类似的rudux：**

**（源码只读过react，但大部分项目还是使用vue编写的）**

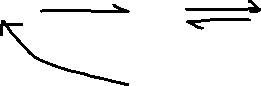
**工作流程：**

② Dispatch ④旧数据，action

Action ③store reducer

⑤新数据

⑥数据变化通知渲染 ⑥subscribe



①用户交互

Views

1. 用户进行交互后通知action
2. action派发(dispatch)行为(action)给store

③ store是数据存储的地方，他是一颗obj树，是redux唯一的数据源，同时它只读，不进行数据的计算和处理

④ store将旧数据提取，与action一起发送给reducer，reducer内存有action对应的操作⑤ 与旧数据计算出结果后，将新数据交给store，store将旧数据覆盖

⑥ 观察者发现数据变化，视图渲染

**作用和场景：**

在项目越来越庞大的时候，共享一个状态会变得困难，因为每一个模块间的数据基本上都在父子之间的传递，也就是说需要一层层的进行转发。

这时就需要一个统一的数据存储地点，将需要共享的数据统一存取。

**3-8题请用JavaScript或者TypeScript作答：**

3、请两种方法对以下数组进行合并const arr1 = ['a', 'b']; const arr2 = ['c']; const arr3 = ['d', 'e']

const arr1 = ["a", "b"];

const arr2 = ["c"];

const arr3 = ["d", "e"];

// 方法一

const ans1 = arr1.concat(arr2, arr3);

console.log('方法一:', ans1)

// 方法二

// 会改变原数组

// for(let i of arr2) {

//     arr1.push(i)

// }

// for(let i of arr3) {

//     arr1.push(i)

// }

// console.log('方法二:', arr1);

// 方法三

// 会改变原数组

arr1.push(...arr2, ...arr3)

console.log('方法三:',arr1)

4、将以下数组去重let a = ['1', '2', '3', 1,NaN,NaN,undefined,undefined,null,null, 'a', 'b', 'b']

let a = ['1', '2', '3', 1, NaN, NaN, undefined, undefined, null, null, 'a', 'b', 'b']

// 方法一（最优）

// 利用set数组

console.log([...new Set(a)])

// 方法二

// 当前索引为当前第一次出现的次数，需要单独判断NaN，因为indexOf会返回-1

// 除了用filter还能用for循环嵌套等其它方式，原理相同

function unique(arr) {

    let NaNFlag = true;

    return arr.filter(function (item, index, arr) {

        let i = arr.indexOf(item, 0);

        if(i == -1 && window.isNaN(item)) {

            if(NaNFlag) {

                NaNFlag = false;

                return true;

            }

            return false;

        }

        return  i === index;

    });

}

console.log(unique(a));

5、js 统计以下字符串中出现频率最高的字母/数字let str = 'aslxjkgakfakerkxlkfeikakdck'

let str = 'aslxjkgakfakerkxlkfeikakdck'

// 利用map

let map = new Map();

let maxCount = 0, maxChar = [];

for (let i of str) {

    // 存在该元素

    if (map.has(i)) {

        let c = map.get(i) + 1;

        // 考虑同一个字母次数相同的情况

        if (c > maxCount) {

            maxCount = c;

            maxChar = [i];

        } else if (c == maxCount) {

            maxChar.push(i);

        }

        map.set(i, c);

    } else {

        map.set(i, 1);

    }

}

console.log(map);

console.log('出现最多的字母：', maxChar.join(''), '次数：', maxCount)

// 其他方法：

// 都需要记录每个字母出现的次数，只是记录方式的不同

// 1、与map类似，将字母以对象的属性方式挂上去（map能优化查找，更优一点）

// 2、建一个长度为26的数组，将字母转为ascii码后按顺序存储，相当于arr[0] = 1为第一个字母a出现一次

6、请从2022-04-23T13:24:06 Europe/Paris提取出["2022","04","23","13","24","06"]

let str = '2022-04-23T13:24:06 Europe/Paris';

// 提取字符串中的数字(最优)

var numArr = str.match(/\d+/g);

console.log(numArr);

// 其他方法

// 逐个遍历字符串，将连续的数字提取出来放入答案

7、数组中最大差值let arr = [6, 8, 16, 7, 55, 26, 6, 4, 88, -3]

let arr = [6, 8, 16, 7, 55, 26, 6, 4, 88, -3]

// 就是找最大值以及最小值

let max = Math.max(...arr), min = Math.min(...arr);

console.log(max - min);

8、对以下数组进行排序let arr = [22, 3, 7, 5, 12, 2]

let arr = [22, 3, 7, 5, 12, 2]

// 方法一

// 使用原生

// arr.sort((a, b) => a - b);

// console.log(arr)

// 方法二（其他排序算法

// 快排

const quickSort = function (arr) {

    if (arr.length <= 1) {

        return arr;

    }

    var p = arr.splice(Math.floor(arr.length / 2), 1)[0];

    var left = [];

    var right = [];

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

        if (arr[i] < p) {

            left.push(arr[i]);

        } else {

            right.push(arr[i]);

        }

    }

    return quickSort(left).concat([p], quickSort(right));

};

console.log(quickSort(arr))