2022年前端面试总结

一 自我介绍

这个我觉得不用多说了 就说几个小点吧

- 可以讲述自己的项目，最好有相关的项目展示
- 在工作中遇到过什么样的问题，然后说出自己解决的方法
- 尽量引导面试官问你的长处

二 技术面试

JS相关

1. 什么是闭包？对闭包的理解

闭包就是能够访问另一个函数作用域中的变量的函数。

闭包的三个特性：

1. 函数嵌套函数
2. 函数内部可以引起函数外部的参数和变量
3. 参数和变量不会被垃圾回收机制回收

2. 同步和异步的区别

- 同步任务：是指在主线程上排队执行的任务，只有前一个任务执行完毕，才能继续执行下一个任务，当我们打开网站时，网站的渲染过程，比如元素的渲染，其实就是一个同步任务
- 异步任务：是指不进入主线程，而进入任务队列的任务，只有任务队列通知主线程，某个异步任务可以执行了，该任务才会进入主线程，当我们打开网站时，像图片的加载，音乐的加载，其实就是一个异步任务

3. 深拷贝和浅拷贝

1.浅拷贝： 将原对象或原数组的引用直接赋给新对象，新数组，新对象 / 数组只是原对象的一个引用。

使用：Object.assign();

2.深拷贝： 创建一个新的对象和数组，将原对象的各项属性的“值”（数组的所有元素）拷贝过来，是“值”而不是“引用”。

使用：可以通过JSON.parse(JSON.stringfy(object))来解决。

4. 防抖和节流（并手写任意一个）

防抖：常用于在搜索框，用户多次触发事件，在用户一直触发事件中事件不会执行，只有在停止触发事件一段时间后再执行这个事件一次。

实现（在网上找的例子）

javascript 复制代码

```
/**
* @fn : 要执行的函数
* @delay : 执行函数的时间间隔
* @immediate : 是否立即执行函数 true 表立即执行，false 表非立即执行
*/

function debounce(fn,delay,immediate){
    let timer; // 定时器
    return function(...args){ // 形成闭包  外部执行的函数其实是这个return出去的函数。
        // args 为函数调用时传的参数。
        let context = this; // this 为函数执行时的this绑定。
        timer&&clearTimeout(timer); // 当函数再次执行时，清除定时器，让定时器重新开始计时
        // immediate为true 表示第一次触发就执行
        if(immediate){
            // 执行一次之后赋值为false
            immediate = false;
            fn.apply(context, args)
        }
        // 利用定时器，让指定函数延迟执行。
        timer = setTimeout(function(){
            // immediate 赋值为true  下次输入时 还是会立即执行
            immediate = true;
            // 执行传入的指定函数，利用apply更改传入函数内部的this绑定，传入 args参数
            fn.apply(context,args);
```

```
        },delay)
    }
}
```

复制代码

搜索逻辑就是监听输入事件，等用户输入完成之后把数据发送给后端，后端返回匹配数据，前端显示数据在页面。

节流：是指在一定的时间同一事件只会触发一次，只有超过这个时间才能再次触发。

实现

javascript 复制代码

```javascript
/**
* @fn : 要执行的函数
* @delay : 每次函数的时间间隔
*/
function throttle(fn,delay){
    let timer;    // 定时器

    return function(...args){
        let context = this;
        // 如果timer存在，说明函数还未该执行 也就是距离上次函数执行未间隔指定的时间
        if(timer) return;
        // 如果函数执行之后还有函数还在触发，再延迟执行。
        timer = setTimeout(function(...args){
            // 当函数执行时，让timer为null。
            timer = null;
            fn.apply(context,args);
        },delay);
    }
}
```

复制代码

用户每一次触发事件都会设置一个延迟定时器，但是如果已经设置了延迟定时器就会等上一次延迟定时器执行之后才会开启下

5. 说说对let var const的理解

csharp 复制代码

```
let可以形成块级作用域，不支持变量名提升。
var用来声明全局变量，支持变量名提升，支持预解析。
const用来定义常量的，常量定义以后不允许改变。
```

6. js中变量和函数的提升

```csharp 复制代码
变量和函数声明的提升会被提升到最顶部去执行；
函数的提升高于变量的提升；
如果在函数内部用var声明了与外部相同的变量，则不向下寻找；
匿名函数不会被提升；
不同块中互不影响
```

7. 说说ajax的原理

Ajax的原理简单来说就是在用户和服务器之间加了一个中间层(Ajax引擎)，由XmlHttpRequest对象来向服务器发异步请求，从服务器获取数据，然后用javascript来操作DOM而更新页面。

8. JavaScript数组去重的方法都有哪些？set如何去重？

- 利用set去重 但不支持对象方法

```javascript 复制代码
function unique(arr){
 //Set数据结构，它类似于数组，其成员的值都是唯一的
 return Array.from(new Set(arr)); // 利用Array.from将Set结构转换成数组
}

var arr = [1,2,2,3,5,3,6,5];
var res = unique(arr)
console.log(res );
```

- 使用两层for循环，在用splice方法删除

```ini 复制代码
function clear(arr){
    for(var i=0; i<arr.length; i++){
        for(var j=i+1; j<arr.length; j++){
            if(arr[i]==arr[j]){//第一个等同于第二个，splice方法删除第二个
                arr.splice(j,1);
                j--;
            }
        }
    }
return arr;
}
var arr = [1,2,2,3];
    console.log(clear(arr));
```

- 新建一个数组，遍历要去重的数组，当值不在新数组的时候（indexOf 为 -1）就加入该新数组中；

```ini 复制代码
function arr1(array) {
    var arr = []; //一个新的数组存放去重后的结果
    for (var i = 0; i < array.length; i++) {
        if (arr.indexOf(array[i]) == -1) { //indexof()方法判断在数组中的位置，若不存在，返回-1
            arr.push(array[i]);
        }
    }
    return arr;
}
var Array = [1, 2, 2, 3];
console.log(arr1(Array));// [1, 2, 3]
```

9. 在工作中如何解决跨域问题？什么情况会产生跨域?

- 只是开发时需要跨域：webpack可以配置devserver
- 上线需要跨域而且只是get请求：用jsonp，动态生成script标签引入
- 上线需要跨域get post都要跨域：后端配置cors跨域，浏览器必须ie10以上版本 因为同源策略限制，不同源会造成跨域。 同源策略是浏览器的一个安全功能，阻止我去请求非同源接口。 协议/子域名/主域名/端口号/ip和网址不同都属于不同源。

10. js事件循环机制

`Event Loop` 即事件循环，是指浏览器或 `Node` 的一种解决 `javaScript` 单线程运行时不会阻塞的一种机制，也就是我们经常使用异步的原理

先执行宏任务队列，然后执行微任务队列，然后开始下一轮事件循环，继续先执行宏任务队列，再执行微任务队列

- 宏任务： `script` / `setTimeout` / `setInterval` / `setImmediate` / `I` / `O` / `UI Rendering`
- 微任务： `process.nextTick()` / `Promise`

11. es6新特性你都用过哪些?

- 箭头函数 简化匿名函数写法
- 函数增强 可以给形参设置默认值
- 模版字符串
- `const` / `let` 都是用来声明变量,不可重复声明，具有块级作用域。存在暂时性死区，也就是不存在变量提升。(`const` 一般用于声明常量);

12. 箭头函数和普通函数的区别

scss 复制代码

```
1.写法上更加简洁
2.箭头函数不能作为构造函数使用
3.箭头函数继承来的this指向永远不会改变
4.箭头函数没有自己的arguments
5.箭头函数没有prototype
6.call()、apply()、bind()等方法不能改变箭头函数中的this指向
```

13. js中数据类型的判断

- 基本数据类型: 为简单的数据段保存在栈中。分为： `String`，`Number`， `Boolean`，`Symbol`，`Undefined`，`Null`
- 引用数据类型： 指的是那些保存在堆内存中的对象，所以引用类型的值保存的是一个指针，这个指针指向存储在堆中的一个对象。除了上面的 6 种基本数据类型外，剩下的就是引用类型了，统称为 `Object` 类型。细分的话，有： `Object` 类型、 `Array` 类型、 `Date` 类型、 `RegExp` 类型、 `Function` 类型 等
- 判断数据类型的方法：
  - typeof： 是一个操作符。右侧跟一个一元表达式，并且返回这个表达式的数据类型。返回的结果用该类型的字符串形式表示
  - instanceof： 是用来判断A是否为B的实例，表达式为 A instanceof B，如果A是B 的实例，则会返回true， 注意：instanceof检测的是原型，只能用来判断两个对象是否属于实例关系，而不能判断一个对象实例具体属于哪种类型
  - constructor： 当一个函数F被定义的时候，JS引擎会为F添加prototype原型，然后再在prototype上添加一个constructor属性，并且让其指向F的引用
  - to String： 这个是object的原型方法，调用该方法，默认返回当前对象的[Class]。这是一个内部属性，其格式为[objectXxx],其中Xxx就是对象的原型

VUE相关

1. vue的生命周期函数

每个 Vue 实例在被创建之前都要经过一系列的初始化过程。例如需要设置数据监听、编译模板、挂载实例到 DOM 、在数据变化时更新 DOM 等。同时在这个过程中也会运行一些叫做生命周期钩子的函数，给予用户机会在一些特定的场景下添加他们自己的代码:

kotlin 复制代码

```
1、beforeCreate（创建前） ：进行数据和方法的初始化
2、created（创建后） ：已经完成数据和方法的初始化
3、beforeMount（挂载前） ：开始渲染dom
4、mounted（挂载后） ：可以渲染dom
5、beforeUpdate（更新前） ：data中的数据即将被更新
6、updated（更新后）：data中的数据更新完毕；
7、beforeDestroy（销毁前） ：实例即将销毁
8、destroyed（销毁后） ：实例已被销毁
```

2. 计算属性和watch的区别？watch都有什么属性?

- computed 计算属性：依赖其它属性值，并且computed的值有缓存，只有它依赖的属性值发生改变，下一次获取computed的值时才会重新计算computed的值。
- watch 侦听器：更多的是观察的作用,无缓存性,类似与某些数据的监听回调,每当监听的数据变化时都会执行回调进行后续操作 从使用场景上说，computed 适用一个数据被多个数据影响，而 watch 适用一个数据影响多个数据。

watch属性：

- 设置deep: 对对象进行深度监听
- immediate:需要在最初绑定值的时候也执行函数，则就需要用到immediate属性。比如父组件向子组建动态传值，子组件props首次获取到父组件传来的默认值时，也需要执行函数，此时就需要将immediate设为true。

3. vue如何实现数据双向绑定的?

通过object.defineProperty()来劫持vue数据属性，一旦监听到变化，更新数据关联的虚拟dom树。

4. v-show与v-if有什么区别

- v-if 是动态的向DOM树内添加或者删除DOM元素，真正的条件渲染，因为它会确保在切换过程中条件块内的事件监听器和子组件适当地被销毁和重建；也是惰性的：如果在初始渲染时条件为假，则什么也不做——直到条件第一次变为真时，才会开始渲染条件块。

- 不管初始条件是什么，元素总是会被渲染，并且只是简单地基于 CSS 的 `display` 属性进行切换。

5. vue组件传值

- 父传子：父组件通过绑定属性传入子组件，子组件用props接收
- 子传父：子向父传值，通过$emit发回父组件，父组件通过属性绑定方式接收
- 父调子：调用子组建的时候定义一个ref，在父组件通过
- 子调父：通过this.$parent调用父组件方法
- 爷孙：ref或eventBus
- 兄弟可以通过vuex

6. MVVM

MVVM是Model-View-ViewModel的缩写。MVVM是一种设计思想。Model 层代表数据模型，也可以在Model中定义数据修改和操作的业务逻辑；View 代表UI 组件，它负责将数据模型转化成UI 展现出来，ViewModel 是一个同步View 和 Model的对象。

7. vuex

`Vuex` 是一个专为 `Vue.js` 应用程序开发的状态管理模式。它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化. `Vuex` 解决了多个视图依赖于同一状态和来自不同视图的行为需要变更同一状态的问题，将开发者的精力聚焦于数据的更新而不是数据在组件之间的传递上

- `state` ： 存储数据

  `Vuex` 使用单一状态树,即每个应用将仅仅包含一个 `store` 实例，但单一状态树和模块化并不冲突。存放的数据状态，不可以直接修改里面的数据。

- `mutations` ： 更新数据的方法

  定义的方法动态修改 `Vuex` 的 `store` 中的状态或数据。

- `actions` ： 调用 `mutations` 方法，更新 `state` 数据

- `getters`：对 `state` 中的数据进行预处理

- modules：项目特别复杂的时候，可以让每一个模块拥有自己的 state 、 mutation 、 action 、 getters ,使得结构非常清晰，方便管理。

8. vue-router

有 3 种路由模式： hash 、 history 、 abstract

- hash：使用 URL hash 值来作路由。支持所有浏览器，包括不支持 HTML5 History Api 的浏览器
- history ：依赖 HTML5 History API 和服务器配置。具体可以查看 HTML5 History 模式
- ◦ abstract ：支持所有 JavaScript 运行环境，如 Node.js 服务器端。如果发现没有浏览器的 API ，路由会自动强制进入这个模式.

9. keep-alive的作用是什么?

keep-alive 是 Vue 内置的一个组件，可以使被包含的组件保留状态，避免重新渲染 ，其有以下特性：

- 缓存组件
- 提供 include 与 exclude 属性，允许组件有条件地进行缓存，其中 exclude 的优先级比 include 高， max 最多可以缓存多少组件实例

10. nextTick 是什么?

vue实现响应式并不是数据发生变化后dom立即变化，而是按照一定的策略来进行dom更新。*nextTick*是在下次*DOM*更新循环结束之后执行延迟回调，在修改数据之后使用nextTick，则可以在回调中获取更新后的 DOM

11.你有对 Vue 项目进行哪些优化？怎么实现的?

- 图片懒加载

```
css 复制代码
先自定义属性：data-src，存放着照片的路径，通过js判断图片是否已加载，未加载再去获取属性data-src的值赋给src
```

- v-if 和 v-show
- computed 和 watch
- 第三方插件的按需引入

12.vue 中 key 值的作用?

vue 中 key 值的作用可以分为两种情况来考虑。

第一种情况是 v-if 中使用 key。由于 Vue 会尽可能高效地渲染元素，通常会复用已有元素而不是从头开始渲染。因此当我们使用 v-if 来实现元素切换的时候，如果切换前后含有相同类型的元素，那么这个元素就会被复用。如果是相同的 input 元素，那么切换前后用户的输入不会被清除掉，这样是不符合需求的。因此我们可以通过使用 key 来唯一的标识一个元素，这个情况下，使用 key 的元素不会被复用。这个时候 key 的作用是用来标识一个独立的元素。

第二种情况是 v-for 中使用 key。用 v-for 更新已渲染过的元素列表时，它默认使用“就地复用”的策略。如果数据项的顺序发生了改变，Vue 不会移动 DOM 元素来匹配数据项的顺序，而是简单复用此处的每个元素。因此通过为每个列表项提供一个 key 值，来以便 Vue 跟踪元素的身份，从而高效的实现复用。这个时候 key 的作用是为了高效的更新渲染虚拟 DOM。

13.vue父子组件的渲染顺序

在组件开始生成到结束生成的过程中，如果该组件还包含子组件，则自己开始生成后，要让所有的子组件也开始生成，然后自己就等着，直到所有的子组件生成完毕，自己再结束。如下顺序:

父beforeCreate->父created->父beforeMount->子beforeCreate->子created->子beforeMount->子mounted->父mounted

14.路由守卫你用过哪些

- 全局守卫

```vbnet
1. router.beforeEach((to,from,next)=>{})
2. 回调函数中的参数，to：进入到哪个路由去，from：从哪个路由离开，next：函数，决定是否展示你要看到的路由页面。
3. main.js中设置全局守卫
4. 全局后置钩子router.afterEach((to,from)=>{}) 只有两个参数，to：进入到哪个路由去，from：从哪个路由离。
```

- 组件内的守卫

```vbnet
1. 到达这个组件时，beforeRouteEnter:(to,from,next)=>{}
这里的next回调函数略有不同，访问不到我们的data属性。所以next()会给一个对应的回调，帮助完成。
2. 离开这个组件时，beforeRouteLeave:(to,from,next)=>{}
点击其他组件时，判断是否确认离开。确认执行next()；取消执行next(false)，留在当前页面。
```

css相关

1.重构和回流

重构： 当元素的某些属性发生变化，这些属性又只影响元素的外观和风格，而不改变元素的布局、大小比如颜色、背景。此时触发的浏览器行为称作重构。

回流： 当元素的布局、大小规模和显示方式发生改变时，触发的浏览器行为叫回流。而且，每个页面都会在第一次加载时触发回流。

2.清除浮动你用什么方式?

给父级添加overflow：hidden或overflow：auto

3.水平文本居中对齐方式有几种?

- text-align：center；
- margin:0 auto;
- justify-content:center;

4.rem em px的区别

- px是像素相对于显示器屏幕分辨率
- em是根据基准来缩放字体的大小。em是相对单位，一般都是以 `<body>` 的字体大小作基准的。em是相对于父元素的属性来计算的，这样就会存在一个问题，就是每一层父元素都必须有它的数值。
- rem是相对于根元素html,而此时我们只需要以rem为基准就可以了

5.三种定位方式的定位原点

- absolute：绝对定位，脱离文档流
- relative：相对定位，不脱离文档流，只改变自身位置
- fixed：固定定位，不随滚动条移动改变位置

6.让一个元素垂直居中有几种

- align-items:center
- height和line-height相等

7.使用CSS画一个三角形

xml 复制代码

```xml
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <style type="text/css">
        /* css3绘制三角形 */
        .triangle{
            width: 0px; /*设置宽高为0，所以div的内容为空，从才能形成三角形尖角*/
            height: 0px;
            border-bottom: 200px solid #00a3af;
            border-left: 200px solid transparent; /*transparent 表示透明*/
            border-right: 200px solid transparent;
        }
    </style>
</head>
<body>
    <div class="triangle"></div>
</body>
</html>
```

html相关

1.请描述一下 cookies， sessionStorage 和 localStorage 的区别?

javascript 复制代码

```
相同点：
cookie，localStorage，sessionStorage都是在客户端保存数据的，存储数据的类型：都是字符串。
不同点：
cookie是网站为了标示用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）
cookie数据始终在同源的http请求中携带（即使不需要），记会在浏览器和服务器间来回传递
sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存
大小限制：cookie大小限制在4KB，非常小；localstorage和sessionStorage在5M
```

2.h5的新特性你用过哪些?

附加题

1.获取某一个用户进入到当前页面停留的时间，做了哪些操作？然后需要吧这些数据封装给后台

2.http请求头有哪些字段

3.针对ie9兼容性问题，不能使用es6有啥方法?

babel-polyfill转换es5

4.http和https有什么区别?

ini 复制代码

```
1、HTTPS协议需要CA证书,费用较高;而HTTP协议不需要
2、HTTP协议是超文本传输协议,信息是明文传输的,HTTPS则是具有安全性的SSL加密传输协议;
3、使用不同的连接方式,端口也不同,HTTP协议端口是80,HTTPS协议端口是443;
4、HTTP协议连接很简单,是无状态的;HTTPS协议是具有SSL和HTTP协议构建的可进行加密传输、身份认证的网络协议,比HTT
```