目录

一. Css

1. 什么是 BFC?

浮动元素和绝对定位元素,非块级盒子的块级容器以及 overflow 的值不为 visiable 的块级盒子都会为内容创建新的 BFC.
触发的条件:
1) 根元素
2) 浮动元素--float 不取值为 none
3) 绝对定位元素--position 取值为 absolute 或 fixed
4) Displa 取值为 inline-block,inline-flex,flex,table-cell,table-caption 之一的元素
5) Overflow 不取值为 visable 的元素

2. 如何清除浮动?

1) 父级 div 定义高度
2) 结尾处加空 div 标签,clear:both
3) 父级定义伪类 :after 和 zoom
4) 父级 div 定义 overflow:hidden

3. 两边固定宽高,中间自适应如何布局?

1) 采用浮动布局(左边左浮动,右边右浮动,中间 margin0)
2) 绝对定位的方式(左右采用绝对定位,左边 left0,右边 right0,中间 margin0)
3) 给中间加一个 flex:1

4. 两栏布局,左边固定,右边自适应怎么做?

左边给一个宽度,再给一个 float:left,右边给一个 overflow:hidden

5. 如何固定标题栏?

给 div 一个 position:fixed(相对于浏览器窗口进行定位)生成绝对定位

6. 如何让 div 在页面上动起来

css 中,用 transition 动画效果去做
错误！未找到图形项目表。
Js 中,用 setTimeout 定时器去做

7. Opacity 和 rgba 的区别

Opacity 是作用于元素以及元素内所有内容的透明度(具有继承性)
Rgba 只作用于元素的颜色或背景色

8. 简述如何实现一个三角形

给一个 border,给边框的属性,其中三遍给透明色
Border-left:50px solid red;
Border-right:50px solid transparent;
.

9. Css3 新特性

新增各种 css 选择器
圆角(border-radius)
多列布局(column)
阴影和反射(shadow/reflect)
文字特效(text-shadow)
线性渐变(gradient)
旋转,缩放,定位,倾斜(transform)
动画(animation)

10. 弹性布局的属性

1) flex-direction---决定主轴的排列方向
2) Flex-wrap---定义是否换行
3) align-items---定义项目在交叉轴上如何对齐
4) align-conent---定义了多根轴线的对齐方式

11. Position 的属性

Position-relative 相对定位
Position-absolute 绝对定位(要给父级设置相对定位)
Position-fixed 固定定位(相对于 body 定位)

12. Sass 和 less 的区别

1、Less 在 JS 上运行，Sass 在 Ruby 上使用。
Sass 基于 Ruby，需要安装 Ruby。Less 和 Sass 在 Ruby 中构建相似，但它已被移植到 JavaScript 中。为了使用 LESS，我们可以将适用的 JavaScript 文件上载到服务器或通过脱机编译器编译 CSS 表。
2、编写变量的方式不同。
Sass 使用$，而 Less 使用@。

3、在 Less 中，仅允许循环数值。
在 Sass 中，我们可以遍历任何类型的数据。但在 Less 中，我们只能使用递归函数循环数值。
4、Sass 有 Compass，Less 有 Preboot
Sass 和 LESS 有可用于集成 mixins 的扩展（在整个站点中存储和共享 CSS 声明的能力）。
Sass 有适用于 mixins 的 Compass，其中包括所有可用的选项以及未来支持的更新。
LESS 有 Preboot.less，LESS Mixins，LESS Elements，gs 和 Frameless。LESS 的软件支持比 Sass 更加分散，导致许多不同的扩展选项可能不会以相同的方式运行。对于项目，我们可能需要所有列出的扩展以获得与 Compass 类似的性能。

13. 盒子模型

标准盒子模型:只设置 concent 的大小
IE 盒模型:将整个元素看成整体,设置大小
Css 的盒模型由 content,padding,border,margin 组成.大小由除了 margin 以外决定.

14. Animation 的用法

要通过关键帧@keyframes 的规则,关键帧有点类似 css 是写样式的,然后再去调用 animation 的属性

15. 如何使一个 css 样式独立,为什么?

加一个 scoped,因为 scoped 是 style 标签上的一个特殊属性,表示当前样式只属于当前模块.
1) 加上 scoped 后,会给 html 的 dom 节点添加一个不重复的 data 属性,去唯一标识这个 dom 元素
2) 在每句 css 选择器的末尾添加一个当前组件的 data 属性选择器来私有化样式

二. ES6

16. ES6 的新特性

1) let&&const 都有块级作用域
2) 箭头函数
3) 模板字符串 反引号`标识
4) 解构赋值
5) For of 循环 可以遍历数组
6) Set map 数据结构
7) Class 类
8) Promise

17. Var,let,const 的区别

Var 可以重复声明,let 不可以,const 声明以后必须赋值,否则会报错
Var 不受限块级,let 受限,const 定义不可变的量,改变会报错
Var 可以在声明上访问变量,const 不行,let 有暂存死区访问也会报错

18. Set 和 map 的区别

Set 用于数据重组,map 用于数据储存
Set 中成员不能重复,只有键值没有键名,类似数组,可以遍历方法有 add,delete
Map 本质是键值对的集合,类似集合,可以遍历,可以跟各种数据格式转换

19. Promise 是什么?promise.all 如何理解

Promise 对象是一种规范,目的是为异步编程提供统一接口,每一个异步任务返回一个 promise 对象,该对象有一个.then 方法,允许指定回调函数
Promise 有三种状态:pending(进行中)fulfilled(已成功)rejected(已失败)
Then 方法会接收两个参数,一个是成功时候的回调(resolve),一个是失败时候的回调(reject)
Promise 主要是引入微任务:采用异步回调替代同步回调解决浪费 cpu 性能的问题,然后放到当前的宏任务中最后执行,解决了回调执行的实时性问题
Promise.all
1) 接收一个 promise 实例的数组或具有 evator 接口的对象作为参数
2) 会返回一个新的 promise 对象
3) 遍历传入的参数,用 promise.resolve()将参数包一层,使其变成一个对象
4) 参数所有回调成功才算成功,其中一个失败则触发失败状态,第一个失败 promise 错误信息,就作为 promise.all 错误信息
5) 一般来说 promise.all 用来处理多个并发请求,为了页面数据结构的方便
6) Promise.race 中与 promise.all 的不同之处是,只要有一个 promise 执行完成,就直接 resolve 并停止执行

20. Set 如何去重?

Let 一个数组,然后再 let 一个新数组,令新数组=[...new set(数组)]

21. forEach,for in,for of 三者区别

ForEach 用来遍历数组
For in 一般用来遍历对象或 json
For of 数组,对象都能遍历,遍历对象通过 object.keys()
For in 循环处的是 key for of 循环出的是 value

22. 解释一下 async 和 await

Async/awiat 被称为 js 中,异步终极解决方案
Async 是一个通过异步执行并隐式返回 promise 作为结果的函数,await 被 js 转换成了 promise,然后调用了 resolve,reject 任务,进入微任务队列,然后 js 暂停当前的协程运行,把线程的执行权交给父协程,回到父协程后,父协程对 await 返回的 promise 调用.then 去监听这个 promise 的状态改变

23. 如何异步操作转同步处理

在使用 aysnc 和 await 进行将异步改成同步处理时,所使用的函数返回值一定是返回的 promise 对象的数据,不然达不到异步改同步的操作处理

三. JavaScript

24. JS 的数据类型

基础:string,number,boolearn,null,undefined
引用:object(arry,math,date,function)

25. 如何判断引用类型,以及原理是什么?

用 instanceof
因为 instanceof 的原理是基于原型链的查询,只要处在原型链中永远为 true

26. 什么是闭包,优缺点是什么?应用场景?

闭包就是能够读取其他函数内部变量的函数
闭包的特性:
函数内嵌套函数,内部函数可以引用外层的参数和变量,参数和变量不会被垃圾回收机制收回
优点:可以实现封装和缓存
缺点:会消耗内存,使用不当还会造成内存泄漏
解决办法:在退出函数之前,将不再使用的局部变量都删除掉
(闭包一般使用在异步操作,使用回调函数的时候,比如定时器,ajax 的请求,事件监听等...)

27. 解释一下原型,原型链

每个对象都会在其内部初始化一个属性,就是 prototype(原型)
当我们访问一个对象的属性的时候,如果这个对象不存在这个属性,那么它就会去 prototype 中寻找这个属性,prototype 会有自己的 prototype,会一层一层找下去,这就是原型链

28. == 和===的区别

==是数据类型的比较,===更严谨,不仅比较数据类型,还要比较值

29. Ajax 原理

简单来说就是在用户和服务器之间加了一个中间层,通过 xmlHttpRequest 对象向服务器发异步请求,从服务器获取数据,然后用 js 操作 DOM 更新页面,使用户操作与服务器响应异步化

30. 数组元素去重方法

利用 ES6 中的 Set 方法去重
Set 为 ES6 新增的一个对象，允许存储任何类型（原始值或引用值）的唯一值
使用双重 for 循环，再利用数组的 splice 方法去重（ES5 常用）
利用数组的 indexOf 方法去重
利用数组的 sort 方法去重（相邻元素对比法）
根据排序后的结果进行遍历及相邻元素比对。
利用 ES6 中的 Map 方法
去重创建一个空 Map 数据结构，遍历需要去重的数组，把数组的每一个元素作为 key 存到 Map 中。由于 Map 中不会出现相同的 key 值，所以最终得到的就是去重后的结果

31. Js 事件循环

同步任务和异步任务分别进入不同的执行环境,同步进入主线程,即主执行栈异步的进入任务队列,主线程内的执行完毕后,会去任务队列读取对应的任务,推入主线程执行,这个过程不断的循环就是事件循环.

32. 节流和防抖以及应用场景

节流:间隔一段时间执行一次回调的场景
例:滚动加载,加载更多或到底监听,谷歌的搜索框,搜索联想功能,高频的点击提交,表单的重复提交(鼠标的滑动,拖拽)

```
复制// 思路：在规定时间内只触发一次
function throttle (fn, delay) {
// 利用闭包保存时间
let prev = Date.now()
return function () {
let context = this
let arg = arguments
let now = Date.now()
if (now - prev >= delay) {
fn.apply(context, arg)
prev = Date.now()
}
}
}
function fn () {
console.log('节流')
}
```

防抖:连续的事件,只触发一次回调的场景
例:搜索框搜索输入,只需用户最后一次输完再发送请求

手机号,邮箱号等验证输入检测
鼠标的 mousemove,mouseover(input 的模糊搜索)

```
复制// 思路:在规定时间内未触发第二次，则执行
function debounce (fn, delay) {
// 利用闭包保存定时器
let timer = null
return function () {
let context = this
let arg = arguments
// 在规定时间内再次触发会先清除定时器后再重设定时器
clearTimeout(timer)
timer = setTimeout(function () {
fn.apply(context, arg)
}, delay)
}
}
function fn () {
console.log('防抖')
}
addEventListener('scroll', debounce(fn, 1000))
```

33. 深拷贝和浅拷贝以及应用场景

浅拷贝只拷贝对象的第一层属性,如果对象中还有对象只是拷贝内存的地址,两者修改会互相影响
深拷贝是拷贝对象的多层属性,如果对象中还有对象会继续拷贝下去

应用场景:想要使用这个对象内的数据,但是不修改原本的对象的数据内容时,可以使用深拷贝来复制数据,并对新的对象内的数据进行操作

34. js 为什么是单线程

与它的用途有关,js 作为浏览器脚本语言,主要用途就是与用户互动以及操作 dom,这决定了它只能是单线程,不然会带来很多复杂的同步问题

Js 异步的原因
Js 选择成为单线程语言,本身就不可能异步,但 js 的宿主环境比如浏览器就是多线程的,宿主环境通过事件驱动使得 js 具备了异步的属性
Js 如何异步
Js 是单线程语言,浏览器只分配给下一个主线程用来执行任务,但一次只能执行一个任务,多个任务就形成了任务队列,等候执行,这个时候,像网络请求,定时器,事件监听这种任务是很耗时的,排队等下去会让执行的效率非常低,甚至导致页面崩溃,所以浏览器为这样耗时的任务开辟了一个另外的线程,这些耗时的任务

可以异步完成后通过回调函数让主线程知道

35. Js 的事件机制

事件是用来实现 js 与 html 之间交互的,可以用侦听器或处理程序来预定事件,一遍事件发生时执行相应的代码

36. 箭头函数 this 指向,改变指向的方法

指向箭头函数定义时所处的对象 三种方法:bind call apply
call,apply 会立即调用,bind 方法不会立即执行,而是返回一个改变了上下文 this 的函数便于稍后调用
Call 的第一个参数是绑定 this 的值,第二个参数是一个参数列表
Apply,bind 的第二个参数是一个参数的数组

37. Js 的继承

1) 原型链继承
让新实例的原型等于父类的实例
2) 借用构造函数继承
在子类函数中做了父类函数的自执行
3) 组合继承
可传参,可复用
4) 原型式继承
用一个函数包装一个对象,然后返回这个函数的调用,就可以随意添加属性的实例或对象,object.create()就是这个原理
5) 寄生式继承
就是给原型式继承外面套了一个壳子
6) 计生组合式继承
修复了组合继承的问题

38. Null 与 undefined 的区别

Undefined 表示不存在这个值,null 表示一个对象被定义了,值为空值
Null 是一个对象,undefined 是一个表示’无’的原始值或表示’缺少值’,应该有值但是未定义

39. 事件修饰符

.stop 阻止事件冒泡
.prevent 阻止默认行为
.once 不论点击多少次 只执行一次

40. 箭头函数与普通函数的区别

普通函数可以有匿名函数，也可以有具体名函数，但是箭头函数都是匿名函数。箭头函数不能用于构造函数，不能使用 new 在普通函数中，this 总是指向调用它的对象，如果用作构造函数，this 指向创建的对象实例

41. 编写一个 b 继承 a 的方法

```
function a() {
    this.run = function() {
        console.log("我是 run 方法");
    }
}
function b() {
}
b.prototype = new a();
var nb = new b();
nb.run();
```

42. Js 递归写 1-100 的和

```
function num(n)
{ if (n == 1) return 1;
return num(n - 1) + n; }
let sum = num(100);
console.log(sum, "sum")
```

43. 垃圾回收器为什么没有把闭包清除掉

因为闭包本身就建立在一个函数的内部作用域的子函数,由于可以访问上级作用域的原因,即使上级函数执行完作用域也不会被清除,这时子函数就是闭包,拥有了访问上级作用域变量的权限,即使上级函数执行完,作用域内的值也不会被销毁

四. Vue

44. 路由跳转怎么传参?

Params 和 query
两者之间的小区别:
1、用法上的
query 要用 path 来引入，params 要用 name 来引入，接收参数都是类似的，分别是 this.$route.query.name 和 this.$route.params.name。
注意接收参数的时候，已经是$route 而不是$router 了哦！！
2、展示上的
query 更加类似于我们 ajax 中 get 传参，params 则类似于 post，说的再简单一点，前者在浏览器地址栏中显示参数，后者则不显示

45. Watch 和 computed 的区别

1) watch 中的函数不需要调用,computed 内部的函数调用时不需要加()
2) Watch 的属性是监听,监听属性值的变化,computed 是计算属性,通过属性计算得来的
3) Computed 是当一个属性受多个属性影响时用 computed,像购物车结算,watch 是当一条数据影响多条数据的时候用 watch,像搜索数据

46. MVVM 是什么?

Model-view-viewmodel 的缩写,传统前端会将数据手动渲染到页面上,MVVM 模式不需要用户手动操作 dom 元素,将数据绑定在 viewmodel 层中,会自动将数据渲染到页面上,视图层变化会通过 viewmodel 更新数据,viewmodel 是 MVVM 的桥梁

47. Vue 的双向绑定原理

是采用数据劫持结合发布者到订阅者模式的方式,通过 object.definepropery()来劫持各个属性的 setter/getter,在数据变动时发布消息给订阅者,触发相应的监听回调

48. v-router 有几种模式,它们的区别是什么?

History 和 hash
Hash 回车刷新会加载到地址栏对应的页面,history 会 404(需要配置 nigax 的重定向)
Hash 兼容 IE8 history 兼容 IE10
Hash 是通过浏览器监听事件变化的
History 是通过 h5 中 history 新增的两个 API 和一个事件去监听 URL 的变化

49. 简述生命周期

创建,挂载,更新,销毁
Created:data 和 methods 创建完毕,最早可以使用处
Mountd:最早可以在这里更新,创建 DOM
更新阶段还有一个 keep-alive 相关
Activated:被 keep-alive 缓存的组件激活时调用
Deactivated:被 keep-alive 缓存的组件停用时调用
beforeDestroy:清除定时器时候使用

50. v-model 原理

View 层输入值影响 data 的属性值,data 属性值改变会更新 view 层的数值变化

51. Axios 的理解

是基于 promise 的 http 库,可以工作于浏览器,也可以在 node.js 中使用
Axios 本质是对原生 XHR 的封装,是 promise 的实现版本
特点:
从浏览器创建 XMLHttpRequest
从 node.js 创建 http 请求
支持 promise API
拦截请求和响应
转换请求和响应数据
取消请求
自动转换成 JSON 数据
客户端支持防止 XSRF/CSRF

52. Data 为什么是一个函数

是为了在重复创建实例的时候避免共享同一数据,造成数据污染

因为组件是可复用的.如果 data 不是一个函数的话,那么大家共享一个 data 属性值,如果 data 改变了就会影响其他的组件

53. 路由的钩子函数

全局前置守卫 beforeEach
组件路由守卫 beforeRouterEnter
路由独享守卫 beforeEnter

54. Vuex 的理解

vuex 是 vue 专用的状态管理库。它以全局方式集中管理应用的状态，并且可以保证状态变更的可预测性。
vuex 主要解决的问题是多组件之间状态共享的问题，利用各种组件通信方式，我们虽然能够做到状态共享，但是往往需要在多个组件之间保持状态的一致性，这种模式很容易出现问题，也会使程序逻辑变得复杂。 vuex 通过把组件的共享状态抽取出来，以全局单例模式管理，这样任何组件都能用一致的方式获取和修改状态，响应式的数据也能够保证简洁的单向数据流动，我们的代码将变得更结构化且易维护。
vuex 并非必须的，它帮我们管理共享状态，但却带来更多的概念和框架。如果我们不打算开发大型单页应用或者我们的应用并没有大量全局的状态需要维护，完全没有使用 vuex 的必要。一个简单的 Store 模式就足够了。反之， Vuex 将会成为自然而然的选择。引用 Redux 的作者 Oan Abramov 的话说就是：Flux 架构就像眼镜：您自会知道什么时候需要它。
我在使用 vuex 过程中有如下理解：首先是对核心概念的理解和运用，将全局状态放入 state 对象中，它本身一棵状态树，组件中使用 Store 实例的 state 访问这些状态；然后有配套的 mutation 方法修改这些状态，并且只能用 mutation 修改状态，在组件中调用 commit 方法提交 mutation ；如果应用中有异步操作或者复杂逻辑组合，我们需要编写 action ，执行结束如果有状态修改仍然需要提交 mutation ，组件中调用这些 action 使用 dispatch 方法派发。最后是模块化，通过 modules 选项组织拆分出去的各个子模块，在访问状态时注意添加子模块的名称，如果子模块有设置 namespace ，那么在提交 mutation 和派发 action 时还需要额外的命名空间前缀。
vuex 在实现单项数据流时需要做到数据的响应式，通过源码的学习发现是借用了 vue 的数据响应化特性实现的，它会利用 Vue 将 state 作为 data 对其进行响应化处理，从而使得这些状态发生变化时，能够导致组件重新渲染。

55. Vue 组件传参方式有哪些?

父子传参 子组件通过 props 接收数据
子父传参 $emit 方法传递参数
非父子传参,兄弟组件 eventbus
eventbus 就是创建一个服务中心,相当于中转站,可以用来传递和接收项目(适合比较小的项目)
Vuex 全局数据管理库

56. v-if 和 v-show 的区别?

v-if 通过删除 dom 元素实现元素的隐藏,v-show 则是通过设置元素的 css 样式,实现隐藏
v-show 适合一些频繁的操作,v-if 不适合

57. 为什么 v-for 跟 v-if 不能一起用

v-for 的优先级比 v-if 高,如果一起使用的话,每次渲染都要遍历整个列表,如果列表的数据有很多,就会造成性能低,页面卡顿的情况,所以要把 v-if 提到 v-for 外的一层上

58. $route 和$router 的区别

Route 是一个跳转的路由对象
Router 是 vueRouter 的一个对象

59. Vue 的两个核心

数据驱动和组件化开发

60. Vue 的常见指令

1、v-if：根据表达式的值的真假条件渲染元素。在切换时元素及它的数据绑定/组件被销毁并重建。
2、v-show：根据表达式之真假值，切换元素的 displayCSS 属性
3、v-for：循环指令，基于一个数组或者对象渲染一个列表，vue2.0 以上必须需配合 key 值使用。
4、v-bind：动态地绑定一个或多个特性，或一个组件 prop 到表达式。
5、v-on：用于监听指定元素的 DOM 事件，比如点击事件。绑定事件监听器。
6、v-model：实现表单输入和应用状态之间的双向绑定
7、v-once：只渲染元素和组件一次。随后的重新渲染，元素/组件及其所有的子节点将被视为静态内容并跳过。这可以用于优化更新性能

61. Vue 的常用修饰符

.trim 输入框过滤首尾空格
.number 先输入数字就会限制只能输入数字
.lazy 输入框改变数据就会改变,lazy 在光标离开 input 才更新数据

62. Vue 中 key 的作用

高效的更新虚拟 dom
当 vue 中用 v-for 更新已渲染过的元素列表时,默认就地复用策略,如果数据项的数据被改变,vue 将不会移动 dom 元素来匹配数据项的顺序,而是简单的复用此处每个元素,并且确保它在特定的索引下显示已被渲染过的每个元素

63. jQuery 和 vue 的区别

jQuery 是使用选择器选取 dom 对象,然后对 dom 对象进行操作赋值,取值事件绑

定等
Vue 是通过 vue 对象实现数据和视图的双向绑定
Vue 侧重数据绑定,可以应用于复杂数据操作的后台页面,jQuery 侧重样式操作,动画效果等,可以应用于一些 html5 的动画页面,一些需要 js 操作页面样式的页面

64. 单向,双向数据流的区别

v-bind 单向数据流绑定 插值形式
v-model 双向数据流 用户对 view 层的更改会直接同步到 model 层

65. 说一下 nextTick

当修改了 data 值,然后马上更新这个 dom 值,是获取不到的,需要使用 nextTick 这个回调,让修改后的 data 渲染更新 dom 元素以后再获取.

66. 如何扩展组件

mixins , extends , 自定义插槽(slot)

67. 说一下 keep-alive

keep-alive 是一个抽象组件,自身不会渲染 dom 元素,也不会出现在父组件链中,使用 keep-alive 包裹动态组件时,会缓存不活动的组件实例,而不是销毁

68. Vue 和 react 的区别

1) 监听数据变化的实现原理不同
Vue 是通过 getter/setter 以及一些函数的劫持,可以精确快速的算出 virtual DOM 的差异,由于在渲染过程中,会跟踪每一个组件的依赖关系,不需要重新渲染整个 dom 树
React 是默认通过比较引用的方式进行的,如果不优化每当应用状态被改变时,全部的子组件都会重新渲染,导致大量的不必要组件渲染,vue 使用的是可变数据,react 更强调数据的不可变
2) 数据流不同
Vue 默认双向绑定,在父子组件传参中,props 是单向数据流
React 一直提倡的都是单向数据流,onchange/setstate()模式
3) 模板渲染方式不同
React 通过 JSX 渲染模板
Vue 通过一种扩展的 html 语法进行渲染
React 是在组件 js 代码中,通过原生 js 实现模板中的常见语法,比如插值,条件,循环,vue 是在和组件 js 代码分离的单独模板中,通过指令实现,比如条件语句 v-if 实现

69. 介绍 mixin,一般在什么情况使用?

Less 中允许你用一个类定义样式,然后把他当做变量,在另一个类中只要引用变量名字就能使用它的所有属性,less 把这种特性称作 mixin
当有多个类,这多个类之间不存在继承关系,但是彼此之间存在可以复用的代码,这个时候就可以考虑使用 mixin 复用代码

70. Vue 中代码优化有哪些

懒加载,代码模块化,for 循环设置 key 值,使用 keep-alive,防抖节流,图片的懒加载,无状态组件标记为函数式组件

71. ajax 和 axios 的区别

Axios 是一个基于 promise 的 http 库,而 ajax 是对原生 XHR 的封装
Ajax 技术实现了对局部数据的刷新,而 axios 实现了对 ajax 的封装

五. 浏览器

72. http 和 https 的区别

传输信息安全不同,连接方式不同,端口不同,证书申请方式不同

73. Js 的垃圾回收机制

JavaScript 具有自动垃圾回收机制,会定期对那些不再使用的变量,对象所占用的内存进行释放,原理就是找到不再使用的变量,然后释放掉其占用的内存
不过当函数局部变量被外部函数使用的时候,有一种情况叫闭包,在函数执行结束以后,函数外部的变量依然指向函数内部的局部变量,此时,函数变量还在使用,所以不会回收.
浏览器通常的垃圾回收方式有标记清除和引用计数

74. 解释用户单点登录

用户登录时,验证用户的账号和密码,生成一个 token,保存在数据库中,将 token 写到 cookies 里,将用户的数据保存在 session 中,请求时带上 cookies,检查有没有登录,如果已登录则放行

75. http 状态码

200 成功
300 系列是重定向
400 客户端错误
401 需要认证
403 拒绝请求访问
404 找不到资源

76. Get 和 post 的区别

1) get 在浏览器退回是无害的,post 会再次提交请求
2) Get 产生的 URL 地址可以被 bookmark,post 不可以
3) Get 请求会被浏览器主动 cache,post 不会,需要手动设置
4) Get 请求只会进行 URL 编码,post 支持多种编码方式
5) Get 比 post 更不安全,因为参数直接暴露在 URL 中,不能用来传递敏感信息
6) Get 参数通过 URL 传递,post 通过 request body
7) Get 产生一个 TCP 包,post 产生两个

77. http 的请求头有什么?

Accept:请求报头域，指定客户端可以接收的类型有哪些

Accept-Charset:浏览器能够显示的字符集
Accept-Encoding：指定客户端可接收的内容编码
Accept-Language：指定客户端可接收的编码语言
Connection：浏览器与服务器之间连接的类型
host:用户指定请求资源的主机和端口号
Cookie：用于存储本地的数据
CookieHost：发出请求的页面所在的域
Referer：用于标识这个请求是从哪个页面来的，如登录前是从主页来的，就显示主页的信息
User-Agent:用于识别用户的使用的客户端版本等信息

78. 如何解决跨域问题

Proxy 代理

79. 什么是 spa 单页面

是一种网络应用程序或网站的模型，它通过动态重写当前页面来与用户交互，这种方法避免了页面之间切换打断用户体验在单页应用中，所有必要的代码(HTML、JavaScript 和 CSS）都通过单个页面的加载而检索，或者根据需要（通常是为响应用户操作)动态装载适当的资源并添加到页面页面在任何时间点都不会重新加载，也不会将控制转移到其他页面
我们熟知的 JS 框架如 react,vue,angular,ember 都属于 SPA
单页应用优缺点
优点：
具有桌面应用的即时性、网站的可移植性和可访问性
用户体验好、快，内容的改变不需要重新加载整个页面
良好的前后端分离，分工更明确
缺点：
不利于搜索引擎的抓取
首次渲染速度相对较慢
实现一个 spa 的原理:
监听地址栏中 hash 变化驱动界面变化
用 pushsate 记录浏览器的历史，驱动界面发送变化
核心通过监听 url 中的 hash 来进行路由跳转

	单页面应用（SPA）	多页面应用（MPA）
组成	一个主页面和多个页面片段	多个主页面
刷新方式	局部刷新	整页刷新
url 模式	哈希模式	历史模式
SEO 搜索引擎	难实现，可使用 SSR 方式	容易实现

	单页面应用（SPA）	多页面应用（MPA）
优化	改善	
数据传递	容易	通过 url、cookie、localStorage 等传递
页面切换	速度快，用户体验良好	切换加载资源，速度慢，用户体验差

80. 页面的回流与重绘

引起 dom 树结构变化,页面布局变化的行为叫做回流,回流一定伴随着重绘,只是样式发生了变化不会引起 dom 树结构变化,页面布局变化的行为叫做重绘,重绘不一定有回流

81. 描述输入 URL 后的加载过程

1) 浏览器根据请求的 URL 交给 DNS 域名解析,找到真实的 ip 后,向服务器发起请求
2) 服务器交给后台处理完成后返回数据,浏览器接收文件
3) 浏览器对加载的资源进行语法解析,建立相应的内部数据结构
4) 载入解析到的资源文件,渲染页面

82. 浏览器存储的方式有哪些?

Cookies,sessionstorage,localstroage

83. Cookies,seesionstorage,localstroage 的区别

Cookies 存储小只有 4k,其他两种存储有 5M
Cookies 可以服务器生成设置过期的时间,sessionstorage 只要不清除就会一直都在,localstorage 在页面关闭的时候就清理了
Cookies 是网络为了标识用户身份而存储在用户本地终端上的数据
Cookies 数据始终在同源的 http 请求中携带,会在浏览器和服务器之间来回传递
Sessionstorage 和 localstorage 仅在本地保存,不会把数据传给服务器

84. 什么是浏览器缓存

1) 先根据这个资源法人一些 http,header 判断是否命中强缓存,命中则直接从本地获取
2) 当强缓存没有命中时,客户端会发送请求到服务器,服务器通过另一些 request,header 验证,如果命中,服务器的请求会返回,不返资源,而且让客户端

直接从缓存中获取,客户端收到返回以后就会从缓存中获取资源
强缓存和协商缓存的共同之处在于,命中缓存都不会重新返回资源
区别是强缓存不发送请求到服务器,协商缓存会
当协商缓存也没有命中的时候,服务器会将资源发送到客户端
当 F5 刷新的时候,会跳过强缓存,但是会检查协商缓存
当 Ctrl+F5 强制刷新时,直接从服务器加载,跳过强缓存和协商缓存

85. 大文件如何上传?

断点上传 利用Blob.prototype.slice的方法,和数组的slice方法相似,将大文件切成小文件去上传

86. 什么操作会造成内存泄漏?

1）意外的全局变量引起的内存泄露
2）闭包引起的内存泄露
3）没有清理的 DOM 元素引用
4）被遗忘的定时器或者回调
怎样避免内存泄露
1）减少不必要的全局变量，或者生命周期较长的对象，及时对无用的数据进行垃圾回收；
2）注意程序逻辑，避免“死循环”之类的 ；
3）避免创建过多的对象　 原则：不用了的东西要及时归还。

六. HTML5

87. h5 新特性

语义标签
增强型表单
视频和音频
Canvas 绘图
SVG 绘图
地理定位
拖放 API
WebWorker
WebStorage
WebSocket

88. Canvas 有哪些用法

基础图形的绘制
文字的绘制
图形的变形和图片的合成

图片和视频的处理
动画的实现
小游戏的制作

89. SVG 和 canvas 的区别

1.什么是 Canvas?
Canvas 是 H5 新出来的标签
Canvas 画布，利用 JavaScript 在网页绘制图像

在标签中给上宽高： 不用加单位
如果在 css 中给宽高 会对图像造成拉伸 （默认宽高 300px*150px）
通过过去绘制工具 .getContext(“2d”) 来在画布中绘制图形
2.什么是 SVG?
SVG 可缩放矢量图形（Scalable Vector Graphics），基于可扩展标记语言 XML 出来的时间比较老
SVG 用来定义用于网格的基于矢量的图形
Canvas 和 SVG 区别
1.绘制的图片格式不同
Canvas 的工具 getContext 绘制出来的图形或传入的图片都依赖分辨率，能够以 .png 和 .jpg 格式保存存储图像，可以说是位图
SVG 可以在 H5 中直接绘制，但绘制的是矢量图
由于位图依赖分辨率，矢量图不依赖分辨率，所以 Canvas 和 SVG 的图片格式的不同实际上是他们绘制出来的图片的格式不同造成的。
2.Canvas 不支持事件处理器，SVG 支持事件处理器
Canvas 绘制的图像 都在 Canvas 这个画布里面，是 Canvas 的一部分，不能用 js 获取已经绘制好的图形元素
Canvas 就像动画，每次显示全部的一帧的内容，想改变里面某个元素的位置或者变化需要在下一帧中全部重新显示。
而 SVG 绘图时，每个图形都是以 DOM 节点的形式插入到页面中，可以用 js 或其他方法直接操作
SVG 中 每个被绘制的图形都被视为对象，如果 SVG 对象的属性发生变化，那么浏览器能够自动重现图形。
3.适用范围不同
由于 Canvas 和 SVG 的工作机制不同，
Canvas 是逐像素进行渲染的，一旦图形绘制完成，就不会继续被浏览器关注。而 SVG 是通过 DOM 操作来显示的。
所以 Canvas 的文本渲染能力弱，而 SVG 最适合带有大型渲染区域的应用程序，比如地图。
而 Canvas 最适合有许多对象要被频繁重绘的图形密集型游戏。
而 SVG 由于 DOM 操作 在复杂度高的游戏应用中 会减慢渲染速度。所以不适合在游戏应用。