**1. h5的新特性**

首先html5为了更好的实践web语义化，增加了header，footer，nav,aside,section等语义化标签，

在表单方面，为了增强表单，为input增加了color，emial,data ,range等类型，

在存储方面，提供了sessionStorage，localStorage,和离线存储，

通过这些存储方式方便数据在客户端的存储和获取，在多媒体方面规定了音频和视频元素audio和vedio，

另外还有地理定位，canvas画布，拖放，多线程编程的web worker和websocket协议

**2. 算法 快速排序**

**3. http协议是超文本传输协议**

https协议是由http和ssl协议构建的可进行加密传输和身份认证的网络协议，

**4. TCP和UDP的区别**

TCP需要建立连接才能传输数据

UDP不需要建立连接就可以传输数据

TCP传输的数据稳定可靠,不会丢失

UPD传输的数据是尽最多努力交付,不能保证数据一定传输到

TCP协议只能1多1传输

UDP协议可以进行1对1,1对多传输

**5. webSocket**

支持持久连续的数据传输 可以用来做聊天室

保持数据的随时更新

相当于在前端和后端之间建立一个链接。

前端可以主动给后端发消息,后端也可以主动给前端发送消息。

优点:服务器端的压力不大

缺点:如果用户关闭浏览器,服务器端不能第一时间感知。

配套系统: 心跳系统.(每隔一小段时间,检查一下各个连接,看看有没有断)

(1) 轮训 每隔一段时间,发送一次数据请求

(2) 长连接

客户端发出请求,服务端接收到请求。但是服务端不直接返回,把这个请求扣住。

当有新的信息来的时候,进行返回

**6. html5**

drag api

**7. cookie 和 localStorage的区别**

cookie每次请求的时候都会被传递给后台

localStorage存储在浏览器的本地不会传递给后台

cookie存储容量小,只有4k左右

localStorage 可以存储5M或更大

document.cookie = "login=true;expires="+odate+"";

localStorage.setItem('name','wthe');

localStorage.getItem("name");

localStorage.removeItem("name");

**8. HTML5语义化标签是:结构良好，便于阅读**

有利于SEO: 和搜索引擎建立良好沟通,让浏览器的爬虫和机器很好地解析

便于团队的开发和维护

**9. Doctype作用?严格模式与混杂模式如何区分？它们有何意义?**

位于文档的最前面,声明浏览器以何种方式来渲染页面

有<!DOCTYPE html> 说明为严格模式

以该浏览器支持的最高标准运行

混杂模式

向老版本兼容

**10. Cookie如何防范XSS攻击**

HttpOnly属性的cookie变量无法被js获取

防御的话为cookie设置httpOnly属性，对用户的输入进行检查，进行特殊字符过滤

**11. 移动端布局**

(1) 媒体查询布局

@media screen and (max-width: 960px){

body{

background-color:#FF6699

}

}

(2) 百分比布局

(3) 视口 像素 逻辑像素 css、像素

(4) rem

相对于浏览器的根元素（HTML元素）的font-size

**rem是怎么进行移动端布局的**

(5) vw

vw 相对于视窗的宽度，视窗宽度是100vw

vh 相对于视窗的高度，视窗高度是100vh

vmin vw和vh中的较小值

vmax vw和vh中的较大值

**12. 前端优化**

降低请求量：合并资源，减少HTTP 请求数，minify / gzip 压缩，webP，lazyLoad。

加快请求速度：预解析DNS，减少域名数，并行加载，CDN 分发。

缓存：HTTP 协议缓存请求，离线缓存 manifest，离线数据缓存localStorage。

渲染：JS/CSS优化，加载顺序，服务端渲染，pipeline。

减少HTTP请求

使用内容发布网络（CDN）

添加本地缓存

压缩资源文件

将CSS样式表放在顶部，把javascript放在底部（浏览器的运行机制决定）

避免使用CSS表达式

减少DNS查询

使用外部javascript和CSS

避免重定向

图片lazyLoad

**图片的一种懒加载，还有一种方式**

**13. get 和 post 请求**

get参数通过url传递，post放在request body中。

get请求在url中传递的参数是有长度限制的，而post没有。

get比post更不安全，因为参数直接暴露在url中，所以不能用来传递敏感信息。

get请求只能进行url编码，而post支持多种编码方式

get请求会浏览器主动cache，而post支持多种编码方式。

get请求参数会被完整保留在浏览历史记录里，而post中的参数不会被保留。

GET和POST本质上就是TCP链接，并无差别。但是由于HTTP的规定和浏览器/服务器的限制，导致他们在应用过程中体现出一些不同。

**14. 在地址栏里输入一个URL,到这个页面呈现出来，中间会发生什么？**

首先通过找到url对应的ip地址，通过查询DNS服务器得到服务器的ip地址,

通过http协议 传输请求报文,经过tcp协议经过三次握手建立连接

发送请求报文 数据传送完成后 tcp协议四次挥手断开连接

资源返回给浏览器 进行页面的渲染

Dom tree css tree

render Tree (排除非视觉节点)

进行页面的渲染

**15. 继承的几种实现方式**

(1) 原型链继承

(2) 通过构造函数继承 call、apply继承

(3) 共享原型的继承方式

(4) 圣杯模式的继承

(5) \_\_proto\_\_进行继承

(6) Es6中实现继承的方式

Object.setPrototypeOf(Son.prototype,Father.prototype)

**16. 如何解决异步回调地狱**

promise、generator、async/await

**promise底层实现原理**

**17. 防抖 节流**

防抖:在百度搜索东西，在打字过程中就进行数据请求

但是这不是我想要的结果

就可以利用防抖的思想

在input触发事件一秒内没触发的时候

再进行数据请求

每次触发input事件就会清除上一次事件

节流函数，只允许一个函数在 X 毫秒内执行一次。

节流函数于防抖函数多了个限制条件，在多少毫秒内只能执行一次

**18. JS中的垃圾回收机制**

JavaScript的解释器可以检测到何时程序不再使用一个对象了，

当他确定了一个对象是无用的时候，他就知道不再需要这个对象，可以把它所占用的内存释放掉了

**19. 前端模块化**

Commonjs：开始于服务器端的模块化，同步定义的模块化，

每个模块都是一个单独的作用域，模块输出，

modules.exports，

模块加载require()引入模块。

AMD：中文名异步模块定义的意思。

**webpack,gulp一些基本的插件和原理需要熟悉**

**20. 深度克隆**

(1) 原始方式

function deepClone(target,origin){

var tostr = Object.prototype.toString;

var arr = '[object Array]';

var obj = '[object Object]';

for(var prop in origin){

if(origin.hasOwnProperty(prop)){

if(origin[prop] != undefined && typeof origin[prop]){

if(tostr.call(origin[prop]) == arr){

target[prop] = [];

}else{

target[prop] = {};

}

deepClone(target[prop],origin[prop])

}else{

target[prop] = origin[prop];

}

}

}

}

(2) JSON.stringify() 将对象转为 字符串

JSON.parse() 将字符串转为 新的对象

(3) Es6中的 ... 扩展运算符

只试用于层数比较少的克隆

(4) $.extend(true,obj1,obj2)

**21. 原始值 引用值**

number string boolean null undefined object symbol

var obj = {

0:'hwt',

1:'age',

2: 175,

length:3,

[Symbol.iterator]:function(){

var curIndex = 0;

let next = ()=>{

return {

value:this[curIndex],

done:this.length == ++curIndex,

}

}

return {

next:next,

}

}

}

**22. 监听属性值的变化**

**Vue的底层实现原理**

Es5 Object.defineProperty(obj,'name',{

set(){

},

get(){

},

enumerable,

configurable,

})

Es6 var newObj = new Proxy(obj,{

set:function(target,key,value,proxyObj){

Reflect.set(target,key,value)

},

get:function(target,key,proxyObj){

return Reflect.get(target,key);

},

})

23. Object.is(NaN,NaN) 可以比较自身

**24.大多数电脑显示器的刷新频率是60Hz，大概相当于每秒钟重绘60次**

requestAnimation()

（1）requestAnimationFrame会把每一帧中的所有DOM操作集中起来，

在一次重绘或回流中就完成，并且重绘或回流的时间间隔紧紧跟随浏览器的刷新频率。

（2）在隐藏或不可见的元素中，requestAnimationFrame将不会进行重绘或回流，

这当然就意味着更少的CPU、GPU和内存使用量

（3）requestAnimationFrame是由浏览器专门为动画提供的API，

在运行时浏览器会自动优化方法的调用，并且如果页面不是激活状态下的话，

动画会自动暂停，有效节省了CPU开销。

**24. arguments.callee 代表自身的函数**

**25. generate、 async await 、promise**

**26. 箭头函数的this**

**27. reduce 执行完会返回一个值** 第一个参数：function ,第二个参数 ，initialValue 初始化值

两个参数都是 必填

function 中 有四个值， prevValue(之前的值) curValue(当前的值) index self

function transCookie(str){

var newStr = str.split('; ') ;

var obj = {};

newStr.reduce(function(prevValue,curValue,index,self){

var \_arr= curValue.split('=');

prevValue[\_arr[0]]=\_arr[1] ;

return prevValue;

},obj);

return obj;

}

**27. 数组去重**

indexOf循环去重

ES6 Set去重；Array.from(new Set(array))

**Set、Map底层原理**

**28. JS实现跨域**

(1) Jsonp 动态创建script标签

script.src = './index.js?cb=demo';告诉后端回调函数的名字

(2) 服务器代理中转的方式

(3) iframe

跨域: 子窗口得到父窗口的数据可以通过loacation.hash进行数据的传递

父窗口得到子窗口的数据

引入一个和父窗口同源的页面,在子窗口将数据存储在window.name上之后

iframe.src 在引入此时同源的窗口,就可以得到传递的数据了

iframe.contentWindow.name

1. domain

**29. vue的生命周期**

-->