71、window.location.reload()

刷新当前页面

71、BOM对象有六类

wdlnsh win档址导屏史

73、outerHTML

innerHTML加上本身，即整个标签所有内容。

this.appendChild(this.innerHTML)

74、优化代码有五步：

复用NOvar拆函注释内存

75、JS库

Jquery，jquery ui， jquery-easyui，require.js，sea.js，zepto，ext.js，swiper.js，iscroll.js，

Angular，vue，react，flutter

76、js数组方法：

77、html元素添加事件

<1>标签里直接写事件。<2>在js中给元素添加事件方法，<3>在js中事件监听函数添加事件方法

78、闭包点击alert每一列index

for循环里面写一个柯里化函数

79、ff火狐和IE的脚本兼容：

注意点：if 、else if判断条件就是兼容条件是否存在

1、冒泡兼容：e.stopPropagation() window.event.cancelBubble=true

2、浏览器默认行为兼容：e.preventDefault() window.event.returnValue=false，最后还要return false

3、tbody写法：IE中tbody=table.childNodes[0]

4、子节点空白：ff第一个子节点表示空白，ie不会返回空白，通过

If(“ ” != node.nodeName)过滤掉空白子对象

5、模拟点击事件：ie: if(docunment.all) document.getElementById(“a3”).click();

非IE ：先var的= document.**createEvents(**)再**initEvent**()，最后docunment.**getElementById**(“a3”).**dispatchEvent(**var的)

6、事件注册 if(isIE) window.attachEvent() else window。addEventListener()

7、ajax对象获取 ie ActiveXObject else XMLHttpRequest

8、firstElement返回结果不同

80、伪数组

有数组部分特性但用不了数组方法。调用最顶层的Array对象的数组原型中的方法然后call改this调用数组方法

81、正则创建有二宝：

1、RegExp（） 2、var reg =// /内部写正则表达式/

82、slideUp动画有序播放：

先stop(true,true)后slideUP() 两个true分别为清空动画队列和来到动画结束状态

先清空队列 后结束状态

83、js本地对象，内置对象，宿主对象

本地对象：包括可以new的和最顶层的原生对象

内置对象：不可以new的顶层对象，如Math，Gload。内置对象是本地对象的子集

宿主对象：所有的非本地对象，如BOM和DOM对象，例document和window

84、四种减页面加载时间的方法：

1、减少http请求和体积

/1/压缩合并css和js文件，/2/精灵图，/3/使用缓存，/4/使用外部css和js减少服务器压力

2、css和script的位置

/1/ css放在head闭合前。/2/Script放在body闭合前

3、DOM操作优化

/1/避免频繁的DOM操作，/2/缓存DOM查找的结果。/3/尽量选择css动画，和canvas。/4/使用事件代理。/5/避免eval()或Function构造函数。/6/避免字符串拼接，用数组join方法替代

4、css选择符优化

A、避免使用通配符选择器，避免使用后代选择器

B、选择器匹配是从右向左匹配的，选择器语句链越短，浏览器的匹配速度越快。

*其实是因为从叶节点找根节点，避免回溯操作。先找到所有的叶节点如所有的p标签，但不如查找id，class什么的来的方便。找到所有p后再从下往上找。这也是为什么不推荐通配符选择器和后代选择器的原因*

C、减少重排(/回流/重构)（reflow）和重绘（repaint）

D、拒绝@import

E、减少css表达式

85、js作用域链：

全局作用域链，函数作用域链，还有es6的块级作用域链。全局作用域在ES6变成了“顶层”块级作用域了。0级是最顶级的

作用域链：变量的使用，从里到外，层层搜索，搜索到就可以直接使用，搜索到0级作用域还没有找到变量就会报错。

86、use strict

//1全局变量显示声明

2 禁止使用with语句

3 eval中定义的变量都是局部变量

4 禁止在函数内部遍历调用栈

5 严格模式下无法删除变量，只有configurable设置为true才能被删除

6 对对象的只读属性赋值将会报错

7、对一个getter方法读取到的属性进行赋值会报错

8、对禁止扩展的对象添加新属性会报错

9、删除一个不可删除的属性会报错

10 对象有多个重名属性会报错，正常模式最后赋值的会覆盖之前的

11 函数有多个重名参数会报错，正常模式可以用arguments[i]读取

12 整数第一位为0报错，正常模式会将其视为八进制数

13 不允许对arguments赋值

14 arguments不再追踪参数的变化

15 禁止使用arguments.callee

16 不允许在非函数的代码块内声明函数，只允许在全局作用域和函数作用域顶层声明函数

17 新增保留字：implements interface let package private protected publice static yield

优点：

//1、调试更加容易

2、防止意外的全局变量

3、消除this强制

4、不允许重复的属性名称或参数值

5、使eval()更安全

6、在delete使用无效时抛出错误

87、对象查找时不查找原型：hasOwnProperty

88、js模块化规范

**1、requireJs**

#引入模块

require([dependencies],function(){});

require（【依赖的数组】，回调函数）

#定义模块

define(id?,dependencies?,factory);

define（模块名？当前模块依赖的模块？模块初始化函数或对象）

**2、seaJs**

#声明模块

define(id?,d?,factory)

- factory :函数 function(require,exports,module)

define(模块名?,依赖的依赖?,函数（获取其他模块的接口，对外提供的接口，存储模块相关属性和方法）)

#使用模块

seajs.use('mymodel.js',function(mymodel){

});

**3、commonJs**

#引入

const modu = require(‘./b.js’)

#导出

单个：exports.foo = “变量” exports.add = function(){}

多个module.exports = {

} 多个会覆盖单个导出

**4、ES6模块化规范**

#导出

//导出单个成员

export let name = 'leo';

//导出多个成员

export {name, age};

//导出默认成员

export default kk;

#引入

import kk,{name,age} from 'a.js'

89、gulp、grunt、webpack

同：前端自动化构建工具：

小同：

gulp和grunt类似，gulp强调前端开发的工作流程，核心是基于一个个任务task进行项目构建。先通过配置一系列task，再定义task处理的事务，然后定义task执行顺序。最后gulp执行这些task，构建整个前端开发流程。

先配置task，再定义事务，然后排序task，最后执行task

异：

Webpack把项目看做一个整体，前端模块化方案，侧重于模块打包。把开发中的资源看成模块。通过loader和插件对资源进行打包处理

90、requireJs的核心思想

核心是创建srcipt元素：

1、通过指定script元素的src属性实现加载模块

2、通过正则匹配模块以及模块之间的依赖关系，保证文件加载的先后顺序

3、并且根据文件路径对加载过的文件做缓存

先加载，再处理依赖，然后排序，最后缓存

91、AMD(requireJs)、CMD(seaJs)

AMD依赖前置，CMD依赖就近，

AMD提前执行，CMD延迟执行，

AMD没有延迟，用户体验好，CMD性能好，需要时才执行

92、ECMAScript6即ES6新增

1、箭头函数

2、块级作用域

3、解构赋值

4、Symbol

5、模板字符串

6、展开运算符

7、类

8、Promise

9、ES6迭代器

10、ES6模块化

93、Object对象常用方法

1、Object.hasOwnProperty() 检查是否被继承

2、Object.isPrototypeOf() 一个对象是否是另一个对象的原型

3、Object.propertyIsEnumerable() 是否可以通过for/in循环看到属性。是否可枚举

4、Object.toLocaleString() 返回对象的本地字符串展示

5、Object.toString() 定义一个对象的字符串显示

6、Object.valueOf() 指定对象原始值

94、高性能JS代码编写十一技

1、DocumentFragment文档碎片优化多次append

2、模板元素clone替代createElement

3、使用一次innerHTML赋值替代构建DOM元素

4、使用firstChild和nextSibling替代childNodes遍历Dom元素。nextSibling 属性可返回某个元素之后紧跟的元素（处于同一树层级中）。如果无此节点，则属性返回 null。

5、使用Array作为StringBuffer（字符串变量），代替字符串拼接

6、将循环控制量保存到局部变量，item用let声明

7、顺序无关的遍历时，用while替代for

8、将条件分支，按可能性顺序从高到低排列

9、在条件分支比较多的时候，优选switch

10、三目运算符替代条件分支

11、需要不断执行，优先setInterval定时器

文档碎片；clone；inner HTML；first；next替代遍历；Array拼接；局部声明i；while循环；条件分支三代目；setInterval

95、this

this指向调用函数的对象。但具体要细谈

1、普通函数调用--- window

2、定时器调用 ----window

3、构造函数调用 ----实例对象

4、对象方法调用 ---调用方法的对象

5、事件绑定的方法 ----绑定事件的对象

6、箭头函数 ----声明函数时所在作用域的this

96、数组方法：split() slice() splice() join()

split() 把字符串分离开，以数组方式存储。字符串的方法

97、jQuery.fn的init方法返回的this。

fn是prototype原型，不是顶层jquery对象

this执行init构造函数本身，也就是jQuery实例对象，返回this实现jquery链式操作

98、jQuery属性拷贝（extend）利用递归赋值实现深拷贝

递归赋值

99、一个函数满足两个调用需求

函数柯里化：把多个参数拆分成一个函数接收部分参数，然后返回一个函数去处理剩余参数

100、let，const，var区别和使用

1、var有变量提升，并且var声明的变量属于函数作用域，let和const属于块级作用域

2、var可以重复声明，在同一作用域下let不能重新声明，const不能改变

101、["1","2","3"].map(parseInt)

结果为 [1,NaN,NaN]

先是map返回一个新数组

parseInt(val,radix)解析字符串，并返回整数。有两个参数，第一个参数入参值，第二个参数是代表数学几进制的。范围是2~36。如果传的参数超过两个，就会转换为多个parseInt，第二个参数radix变为自然数序号。即相当于　[parseInt("1",0),parseInt("2",1),parseInt("3",2)]

102、Cookie和Session

Cookie和Session是服务器开发中会话管理的数据存储技术。Cookie和Session的主要区别如下：

1. 大小：cookie ---4k ， session ---5M
2. cookie存在客户端，Session保存在服务器，
3. cookie存字符串，session存对象
4. cookie同一网站不同路径互相访问不到，session不区分路径，同一用户在这一个网站哪个地方都能访问到session
5. session需要借助cookie，客户端没有cookie拿不到session
6. session在用户会话结束时关闭，cookie存在客户端，在设定期限内一直保存

103、bootstrap的栅格系统

将屏幕分为12列，响应式，移动设备优先的流式栅格系统，还包含了预定义class易于使用，还有mixin用于生成更具语义布局

104、decodeURL 及编码解码

URI里有中文或特殊符号 / & 时，发送请求时要对URI进行编码发给服务器。编码利于二进制用于传输。而我们需要使用时就需要解码成我们看得懂的来使用

编码三杰：encodeURI() , encodeURIComponent() , unescaped()

解码三俊：dencodeURI() ; dencodeURIComponent() , escape()

105、js的内存管理 垃圾回收机制

**1、引用计数法：**

给一个对象开一块内存空间。每有一个其他函数或对象引用此对象时，给对象的指针数加1。

当对象指针数为0，就相当于没有其他代码引用它，此时该对象的内存就会被释放。

缺点：循环引用的问题无法解决，内存泄漏

**2、标记清除法：**

垃圾回收器构建roots列表。Roots通常为代码中保留引用的全局变量，如window对象。从roots出发开始递归检查并标记active给所有对象。此时所有不被标记为active的内存就视为垃圾了。推荐使用

106、浏览器四进程

1、主进程Browser：onlyONe，负责资源下载，页面展示等主要基础功能

2、GPU进程：3D图示绘制

3、第三方插件进程：第三方插件处理

4、渲染进程：js执行，页面渲染

渲染五线程：

1. GUI渲染线程：渲染页面
2. JS引擎线程：处理js脚本程序
3. 事件触发线程：控制事件循环。等js引擎空闲执行
4. 定时器触发线程：计时。计时完毕等js引擎空闲执行
5. 异步http请求线程：在XMLHttpRequest连接后通过浏览器新开一个线程请求。在检测状态变更后将回调函数放入事件队列。

107、js检测变量是否为String类型

1、typeof 只能判断基本类型，引用类型返回object

2、Object原型的toString方法

Object.prototype.toString.call(参数)

3、构造器

obj.constructor === String

108、$(this)和this在jQuery中的不同

this是js的dom对象，调用dom操作的API，如innerHTML等

$(this)表示把dom对象转换为jquery对象，可以调用jQuery方法。如text（）获取文本。Val（）获取值。

109、document.load()和document.ready()的区别

DOM文档解析过程:

1、解析html结构

2、加载js和css

3、解析并执行js

4、构造html的DOM模型 //document.ready表示文档结构加载完成，没有非文字媒体文件，dom加载完成后就可以进行dom操作

5、加载图片等外部资源文件

6、页面加载完成 //document.load。onload。页面所有元素加载完成

不同点主要在于图片等有没有加载

110、jQuery中$.get()和$.post()提交的区别

同：异步请求获取服务端数据

异：1、请求方式：一个get一个post

2、参数传递方式：get请求参数跟在URL上，POST放在请求体中

3、数据传输大小不同：get传输数据不能超过2kb，POST则很大

4、安全问题：GET请求的数据会被浏览器缓存，有安全问题