#### Node.js的优缺点？

优点：1高并发（最重要的优点）

2适合I/O密集型应用

3 nodejs并没有重新开发运行时环境，而是选择了目前最快的浏览器内核V8作为引擎，保证了nodejs的性能和稳定性。

缺点：1Node是一个相对新的开源项目，所以不太稳定，它总是一直在变。

2缺少足够多的第三方库支持。

#### Node与其他服务器相比有哪些特性？

1异步事件驱动

2非堵塞I/O

3单线程（这里指主线程）

4性能出众

#### Node有没有根目录?

--node当中有个全局变量\_\_dirname ,可通过path引用获得项目根目录。

#### 如何判断当前脚本是运行在浏览器还是Node环境中

exports = typeof window === 'undefined' ? global : window ;

获取全局对象的方式

同理可得，typeof window可以用来判断是不是在浏览器环境中

#### 什么是回调地狱，怎么进行避免?

函数作为参数层层嵌套就是回调地狱.

怎么进行避免?

1保持你的代码简短(给函数取有意义的名字,见名知意,而非匿名函数,写成一大坨)

2模块化(函数封装,打包，每个功能独立,可以单独的定义一个js文件Vue,react中通过import导入就是一种体现)

3处理每一个错误

4创建模块时的一些经验法则

5承诺/生成器/ES6等

#### CSS有哪几种选择器？

标签选择器（如：body,div,p,ul,li）

.类选择器（如：class="head",class="head\_logo"）

ID选择器（如：id="name",id="name\_txt"）

全局选择器（如：\*号）

.组合选择器（如：.head .head\_logo,注意两选择器用空格键分开）

.继承选择器（如：div p,注意两选择器用空格键分开）

伪类选择器（如：就是链接样式,a元素的伪类，4种不同的状态：link、visited、active、hover。）

字符串匹配的属性选择符(^ $ \*三种，分别对应开始、结尾、包含)

#### 宽高200px的div盒子在不同屏幕下，怎么进行上下左右居中

1、absolute，margin: auto，left，right，top，bottom为0

2、relative，left top 50%，transform: translate(-50%,-50%)

3、display:flex和justify-content：center;align-items:center

4、relative，left top 50%，负margin-left margin-top

5、text-align:center,dispaly:table-cell,vertical-align:middle,子元素：设置为行内块元素

#### 拖拽会用到哪些事件？

dragstart 被拖放的元素 开始拖放操作

drag 被拖放的元素 拖放过程中

dragenter 拖放过程中鼠标经过的元素 被拖放元素开始进入本元素的范围内

dragover 拖放过程中鼠标经过的元素 被拖放元素正在本元素范围内移动

dragleave 拖放过程中鼠标经过的元素 被拖放元素离开本元素的范围

drop 拖放的目标元素 有其他元素被拖放到本元素中

dragend 拖放的对象元素 拖放操作结束

#### JS定时器有哪些，区别是什么？

setTimeout 只执行一次

setInterval 会一直重复执行

#### H5的本地缓存

相同点： 缓存数据比cookie的范围大;

localStorage:关闭浏览器数据不会消失，除非手动删除数据

sessionStroage：数据暂时缓存在客户端，关闭浏览器数据丢失

下面是localStorage，sessionStorage常用的方法：

setItem(‘name’,'xsxs');//设置数据

getItem('name');//获取数据

removeItem('name');//移除某条属性的数据

clear();//清空数据

#### Document.write()和innerHtml的区别是什么？

document.write会刷新整个页面的内容，而innerHTML只是找到标签在后面插入一些内容。

#### 原生Ajax分几步？里面有多少种callback

原生Ajax分几步？分五步。

　Ajax请求总共有八种Callback

　　onSuccess

　　onFailure

　　onUninitialized

　　onLoading

　　onLoaded

　　onInteractive

　　onComplete

　　onException

#### 常见的HTTP状态码有哪些？都是做什么的？

200 – 请求成功

301 – 资源(网页等)被永久转移到其它URL

302 – 重定向，需要进一步的操作以完成请求

403 - 没有权限

404 – 请求的资源(网页等)不存在

500 – 内部服务器错误

#### XML和JSON的区别？

json轻便、解析简单，对客户端更加友好。

XML更加笨重，解析为DOM树并遍历节点来取数据，优点是结构清晰，扩展性好。

#### 清除浮动的方式？

1额外标签法（在最后一个浮动标签后，新加一个标签，给其设置clear：both；）（不推荐）

2父级添加overflow属性（父元素添加overflow:hidden）（不推荐）

3使用after伪元素清除浮动（推荐使用）

4使用before和after双伪元素清除浮动

5定义父级div的高度。

#### 左右定宽中间自适应的方式有哪几种？

1浮动加定位

2外框子 Margin 双飞翼模式

3弹性盒子法

#### 前端如何进行网站性能的优化？

尽可能减少HTTP请求：图片合并 （css sprites），Js脚本文件合并、css文件合并

#### PX和em和rem的区别？

px像素（Pixel）。相对长度单位。像素px是相对于显示器屏幕分辨率而言的。

em是相对长度单位。相对于当前对象内文本的字体尺寸。如当前对行内文本的字体尺寸未被人为设置，

则相对于浏览器的默认字体尺寸。

rem单位基于html元素的字体大小。

1、rem 和 em 单位是由浏览器基于你的设计中的字体大小计算得到的像素值。

2、em 单位基于使用他们的元素的字体大小。

3、rem 单位基于 html 元素的字体大小。

4、em 单位可能受任何继承的父元素字体大小影响

5、rem 单位可以从浏览器字体设置中继承字体大小。

6、使用 em 单位应根据组件的字体大小而不是根元素的字体大小。

7、在不需要使用em单位，并且需要根据浏览器的字体大小设置缩放的情况下使用rem。

8、使用rem单位，除非你确定你需要 em 单位，包括对字体大小。

9、媒体查询中使用 rem 单位

10、不要在多列布局中使用 em 或 rem -改用 %。

11、不要使用 em 或 rem，如果缩放会不可避免地导致要打破布局元素。

#### JS哪些操作会造成内存泄漏?

1意外的全局变量引起的内存泄漏

2闭包引起的内存泄漏

3没有清理的DOM元素引用

4被遗忘的定时器或者回调

5子元素存在引用引起的内存泄漏

#### 浏览器的缓存有哪几种？

1、http缓存

2、cookie

3、localstorage

4、sessionstorage

#### 响应式的原理是什么？

媒体查询和栅格系统

#### 什么是事件冒泡？如何阻止事件冒泡？

事件冒泡阶段：事件从事件目标(target)开始，往上冒泡直到页面的最上一级标签。

如何阻止事件冒泡？

EVENT.STOPPROPAGATION()

event.cancelBubble = true;

return false;

#### 什么是闭包？

能够访问另一个函数作用域的变量的函数。

闭包就是一个函数，这个函数能够访问其他函数的作用域中的变量。

function outer(){

var a = '1';

var inner = function(){

cosole.log(a)

}

retrun inner //inner就是一个闭包函数，因为它能够访问到outer函数的作用域

}

#### let和const的区别？

let定义的变量，只能在块作用域里访问，不能跨块访问，也不能跨函数访问。

const用来定义常量，使用时必须初始化(即必须赋值)，只能在块作用域里访问，而且不能直接修改。

因为对象是引用类型的，person中保存的仅是对象的指针，这就意味着，

const仅保证指针不发生改变，修改对象的属性不会改变对象的指针，所以是被允许的。

也就是说const定义的引用类型只要指针不发生改变，其他的不论如何改变都是允许的。

#### 事件委托的原理？

利用事件冒泡机制

#### 什么是面向对象？

面向对象的方法主要是把事物给对象化，包括其属性和行为。

面向对象编程更贴近实际生活的思想。总体来说面向对象的底层还是面向过程，

面向过程抽象成类，然后封装，方便使用就是面向对象。

#### 循环语句有哪些？

while循环()

do-while循环

for循环

for-in循环

forEach

map

#### For in 和for of 的区别？

for...in 循环主要是为了遍历对象而生，不适用于遍历数组

for...of 循环可以用来遍历数组、类数组对象，字符串、Set、Map 以及 Generator 对象

#### 集合和数组的区别？

(1)数组是大小固定的，并且同一个数组只能存放类型一样的数据（基本类型/引用类型）

对象和数组都是属于集合

数组标识有序数据集合，而对象则表示无序数据集合。

还有一个非常重要的区别就是，集合是无名称的，而数组是有名称的。

#### 网页三要素

html（结构）、css（样式）、javascript（行为动作）

#### 创建函数的几种方式

1函数声明

2函数表达式

3匿名函数

#### 创建对象的方式？

1对象字面量方式

2object构造函数

3构造函数模式

4工厂模式

5原型模式

#### 小括号,大括号,中括号和冒号的区别是什么？

小括号

1函数声明时参数表

2和一些语句联合使用以达到某些限定作用

3和new一起使用用来传值(实参)

4作为函数或对象方法的调用运算符(如果定义了参数也可与语义3一样传实参)

大括号

1组织复合语句,这是最常见的

2对象直接量声明

3声明函数或函数直接量

中括号

1声明数组

2取数组成员

3定义对象成员(可以不遵循标识符规则)

4取对象成员

冒号

1声明对象的成员

2switch语句分支

3三元表达式

#### 写一个匿名函数自调用

(function(){

//...

})();

#### 精灵图的原理以及优点和缺点

1> 原理: 把网页中的一些背景图片整合到一张图片文件中, 利用backgroun-position等属性进行背景定位

1> 优点:

<1> 减少http请求, 加快网页的加载速度, 提高用户体验

<2> 减少图片的体积, 减少字节数

<3> 解决设计师图片命名的烦恼

<4> 更换风格方便

2> 缺点:

<1> 开发时需要测量, 比较繁琐

<2> 维护的时候麻烦

<3> 精灵图不能随意改变大小和位置

<4> 宽屏高分辨率的屏幕下, 容易出现背景断裂

#### git常用的命令行？

git clone https://github.com/wlz1244/qingoo.git //下载一个master分支代码

git branch wlz //新建wlz分支

git checkout wlz //切换到wlz分支

git pull origin dev //将远程dev分支更新到本地wlz分支

··············写代码

git add .

git commit -m "XXXXX" //现在还是wlz分支

git checkout dev //切换到dev分支

git pull origin dev //将本地dev分支更新到最新远程dev分支，一摸一样

git merge wlz //将本地wlz分支合并到本地最新dev分支

·············有冲突的话，解决冲突

// git add .

// git commit -m "XXXXX"

git push origin dev //提交到远程dev分支

git checkout wlz //切换到wlz分支

git pull origin dev //更新最新代码到本地

git reset --hard origin/dev 后退已经commit的东西到git服务器的最新代码

#### 定位有哪几种，有什么区别？

STATIC(静态）

HTML元素的默认值，不受top、bottom、left、right属性影响，元素出现在正常的文档流中

RELATIVE(相对）

相对定位, 特点：不脱离文档流的布局，受top、bottom、left、right属性影响,只改变自身的位置，在文档流原先的位置遗留空白区域。定位的起始位置为此元素原先在文档流的位置。

ABSOLUTE(绝对)

绝对定位 特点: 脱离文档流的布局，遗留下来的空间由后面的元素填充。定位的起始位置为最近的父元素(position不为static)，否则为html文档本身。

FIXED(固定)

固定定位, 特点：类似于absolute，但不随着滚动条的移动而改变位置。元素的位置相对于浏览器窗口是固定位置。

#### this的指向问题

window

函数直接执行，函数名+括号 (非箭头函数)

匿名函数自执行

定时器直接调用

箭头函数暴露在全局

事件触发的元素

事件函数(不能是箭头函数)被触发的时候，事件函数内的this，指向事件元素

实例

new了之后就指向实例（默认return 是 this），return上方this指向构造函数的实例

　　　　 构造函数的return 默认是 this ，如果写了一个，return 一个简单类型 ，那return的还是 this

　　　　 如果写了一个，return 一个引用类型 ，那return的就是那个引用类型。

undefined

严格模式

'use strict'

声明变量必须使用var或let

arguments是不和形参相映射的

this指向一般是 undefined

对象的this

就是方法名.前面的对象（前提：不被定时器直接调用）

箭头函数 （文章末尾，简单介绍箭头函数）

　　　　 this指向箭头函数的父级，只看定义时父级的域（比如，箭头函数在全局下，this指向window）

#### ES5和ES6的区别？

ECMAScript 5（ES5）：ECMAScript的第5版，于2009年标准化。该标准已在所有现代浏览器中完全实现

ECMAScript 6（ES6）/ ECMAScript 2015（ES2015）：第6版ECMAScript，于2015年标准化。该标准已在大多数现代浏览器中部分实施。

新增了一些方法和属性

箭头函数和字符串插值

块作用域变量。新的ES6关键字let允许开发人员在块级别定义变量。

默认参数值 默认参数允许我们使用默认值初始化函数。如果省略或未定义参数，则使用默认值 - 意味着null是有效值。

类定义和继承

for-of运算符

Promises Promises提供了一种处理异步操作的结果和错误的机制。你可以用回调来完成同样的事情，但是promises通过方法链接和简洁的错误处理提供了更高的可读性。

模块导出和导入

#### ES6的set是干什么的？

set是数组去重

#### 箭头函数有哪些特性？

1不能作为构造函数，不能使用new

2不能使用argumetns,取而代之用rest参数…解决

3不绑定this，会捕获其定义时所在的this指向作为自己的this。由于在vue中自动绑定 this 上下文到实例中，因此不能使用箭头函数来定义一个周期方法。箭头函数的this永远指向上下文的this，call、apply、bind也无法改变

4箭头函数没有原型对象

#### set和map的区别？

Set 对象类似于数组，且成员的值都是唯一的。

Map 对象是键值对集合，和 JSON 对象类似，但是 key 不仅可以是字符串还可以是对象。

#### HTTP和HTTPS的区别？

1、https协议需要到CA申请证书，一般免费证书较少，因而需要一定费用。

2、http是超文本传输协议，信息是明文传输，https则是具有安全性的ssl/tls加密传输协议。

3、http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。

4、http的连接很简单，是无状态的；HTTPS协议是由SSL/TLS+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。

#### 堆和栈的区别？

基本类型（简单的数据段，存放在栈里面，占固定大小的空间）

引用类型（多个值构成的对象，保存在堆内存中，包含引用类型的变量实际上保存的不是变量本身，而是指向该对象的指针）

#### 栈和队列的区别？

栈和队列都是动态的集合，在栈中，可以去掉的元素是最近插入的那一个。栈实现了后进先出。在队列中，可以去掉的元素总是在集合中存在的时间最长的那一个。队列实现了先进先出的策略。

#### DOCTYPE的作用？

1.告知浏览器文档使用哪种 HTML 或 XHTML 规范。

2.告诉浏览器按照何种规范解析页（如果你的页面没有DOCTYPE的声明，那么compatMode默认就是BackCompat,浏览器按照自己的方式解析渲染页面

#### 工厂模式和构造函数是什么？

工厂模式

　　考虑到在 ECMAScript 中无法创建类，开发人员就发明了一种函数，用函数来封装以特定接口创建对象的细节，

构造函数模式.

　　ECMAScript中的构造函数可用来创建特定类型的对象，像Array和Object这样的原生构造函数，在运行时会自动出现在执行环境中。

此外，也可以创建自定义的构造函数，从而定义自定义对象的属性和方法。使用构造函数的方法，既解决了重复实例化的问题，又解决了对象识别的问题。

#### 如何实现多个标签页浏览器之间的通信？

方法一：

localstorge在一个标签页里被添加、修改或删除时，都会触发一个storage事件，通过在另一个标签页里监听storage事件，即可得到localstorge存储的值，实现不同标签页之间的通信。

方法二：

使用cookie+setInterval，将要传递的信息存储在cookie中，每隔一定时间读取cookie信息，即可随时获取要传递的信息

#### null和undefined的区别

null是空对象引用，引用指向为空

undefined是只定义了引用

typeof null：”object”

typeof undefined ： “undefined”

null == undefined true

null === undefined false ：值相同，但类型不同

均可以用null或者undefined清空属性

var person = null; // 值为 null(空), 但类型为对象

var person = undefined; // 值为 undefined, 类型为 undefined

#### $.get和$.post的区别是什么

$.get $.post是简单易用的高层实现，我们使用$.get $.post方法，jQuery会自动封装调用底层的$.ajax。

$.get 只处理简单的 GET 请求功能以取代复杂 $.ajax，请求成功时可调用回调函数。不支持出错时执行函数，否则必须使用$.ajax。

$.post 只处理 post请求功能以取代复杂 $.ajax 。请求成功时可调用回调函数。不支持出错时执行函数，否则必须使用$.ajax。

$.get("test.php", { name: "John", time: "2pm" } ) $.get方法在请求时会自动生成queryString提交给服务器(name=John&time=2pm)，

$.post方法提交的数据直接类似表单提交，提交的数据量比$.get更大。

#### 异步加载和延迟加载的区别？

异步加载:

异步加载也叫非阻塞模式加载，浏览器在下载js的同时，同时还会执行后续的页面处理。

在script标签内，用js创建一个script元素并插入到document中，这种就是异步加载js文件了。

同步加载流程是瀑布模型，异步加载流程是并发模型。

延迟加载:

前面解决了异步加载（async loading）问题，再谈谈什么是延迟加载。

延迟加载：有些 js 代码并不是页面初始化的时候就立刻需要的，而稍后的某些情况才需要的。延迟加载就是一开始并不加载这些暂时不用的js，而是在需要的时候或稍后再通过js 的控制来异步加载。

也就是将 js 切分成许多模块，页面初始化时只加载需要立即执行的 js ，然后其它 js 的加载延迟到第一次需要用到的时候再加载。

特别是页面有大量不同的模块组成，很多可能暂时不用或根本就没用到。

就像图片的延迟加载，在图片出现在可视区域内时（在滚动条下拉）才加载显示图片.

#### less是做什么的？它和sass的区别是什么

Less：

是一种动态样式语言. 对CSS赋予了动态语言的特性，如变量、继承、运算、函数。

Less 既可以在客户端上运行 (支持IE 6+, Webkit, Firefox)，也可在服务端运行。

Sass：

是一种动态样式语言，Sass语法属于缩排语法，

比css比多出好些功能(如变量、嵌套、运算,混入(Mixin)、继承、颜色处理，函数等)，更容易阅读。

less和sass的区别:

Less和Sass的主要不同就是他们的实现方式。

Less是基于JavaScript，是在客户端处理的。

Sass是基于Ruby的，是在服务器端处理的。

关于变量在Less和Sass中的唯一区别就是Less用@，Sass用$。

#### 怎么移除onclick事件？

btn1.onclick = null； //去掉绑定的事件

btn2.removeEventListener('click', showMsg, false); //去除绑定

#### jQuery中的addClass，removeClass和toggleClass的区别？

addClass(“className”)方法是用来给指定元素增加类名，也就是说给指定的元素追加样式；

removeClass(“className”)方法是用来给指定的元素移除类名，也就是说给指定元素移除样式；

toggleClass(“className”)方法是用来给指定的元素增加或移除类名（如果元素的类名存在就移除，如果不存在就增加），也就是说用来给指定的元素进行样式切换（如果元素存在样式则去掉，如果不存在则加上样式）

#### 从输入url到页面加载出来整个过程发生了什么？

1、浏览器的地址栏输入URL并按下回车。

2、浏览器查找当前URL的DNS缓存记录。

3、DNS解析URL对应的IP。

4、根据IP建立TCP连接（三次握手）。

5、HTTP发起请求。

6、服务器处理请求，浏览器接收HTTP响应。

7、渲染页面，构建DOM树。

8、关闭TCP连接（四次挥手）

#### Display:none和visibilty区别？

如果给一个元素设置了display: none，那么该元素以及它的所有后代元素都会隐藏，它是前端开发人员使用频率最高的一种隐藏方式。隐藏后的元素无法点击，无法使用屏幕阅读器等辅助设备访问，占据的空间消失。

给元素设置visibility: hidden也可以隐藏这个元素，但是隐藏元素仍需占用与未隐藏时一样的空间，也就是说虽然元素不可见了，但是仍然会影响页面布局。

#### 什么是JS同源策略？

限制从一个源加载的文档或脚本如何与来自另一个源的资源进行交互。这是一个用于隔离潜在恶意文件的关键的安全机制。（来自MDN官方的解释）

简单来说就是：一段脚本只能读取来自于同一来源的窗口和文档的属性，这里的同一来源指的是主机名、协议和端口号的组合

具体解释：

（1）源 ：包括三个部分：协议、域名、端口（http协议的默认端口是80）。如果有任何一个部分不同，则源不同，那就是跨域了。

（2）限制：这个源的文档没有权利去操作另一个源的文档。这个限制体现在：（要记住）

Cookie、LocalStorage和IndexDB无法获取。

无法获取和操作DOM。

不能发送Ajax请求。我们要注意，Ajax只适合同源的通信。

同源策略带来的麻烦：ajax在不同域名下的请求无法实现，

#### 浏览器如何渲染页面？

1、解析文档构建DOM树

2、构建渲染树

3、布局与绘制渲染树

#### 什么是线程和进程？

1.进程是系统进行资源分配的基本单位，有独立的内存地址空间； 线程是CPU调度的基本单位，没有单独地址空间，有独立的栈，局部变量，寄存器， 程序计数器等。

2.创建进程的开销大，包括创建虚拟地址空间等需要大量系统资源； 创建线程开销小，基本上只有一个内核对象和一个堆栈。

3.一个进程无法直接访问另一个进程的资源；同一进程内的多个线程共享进程的资源。

4.进程切换开销大，线程切换开销小；进程间通信开销大，线程间通信开销小。

5.线程属于进程，不能独立执行。每个进程至少要有一个线程，成为主线程。

#### eval()是做什么的？

eval()函数，这个函数可以把一个字符串当作一个JavaScript表达式一样去执行它。

#### 怎么清除字符串前后空格并兼容浏览器？

function trim(str) {

if (str && typeof str === "string") {

return str.replace(/(^\s\*)|(\s\*)$/g,""); //去除前后空白符

}

}

#### 怎么降维数组？

1、遍历数组，将元素一个个放入新数组

2、用apply的特性，将数组作为参数展开

3、利用es6新增的扩展运算符将数组展开

#### CSS和CSS3的区别？

CSS和CSS3都是样式，CSS3是在CSS上新增了样式属性。

CSS3新增特性

CSS3圆角边框：border-radius（解决了原有圆角边框需要截图的问题）

CSS3文本阴影与盒阴影：text-shadow、box-shadow（可以用于实现标签边框或阴影）

CSS3背景尺寸、背景切割和背景原点：background-clip background-origin background-size（移动端用于处理背景图适配问题）

CSS3变形 transform（二维变形以及三维特效）

CSS3动画 animation（实现复杂动画）

CSS3过渡 transition（实现简单动画）

CSS3 Media Query媒体查询（主要用于实现响应式布局）

#### 怎么进行深度克隆？

使用递归

const deepCopy=(obj)=>{

//如果被克隆的是简单数据对象，没必要进行深度克隆

if(typeof obj!=='object'){

return;

}

//判断被克隆的是对象还是数组

let newObj=obj instanceof Array?[]:{};

//对每一个属性进行判断

for (let key in obj){

//只对自身的属性进行赋值

if(obj.hasOwnProperty(key)){

//如果该属性值也是对象，递归调用深度克隆方法

newObj[key]=typeof obj[key]==="object"?deepCopy(obj[key]):obj[key];

}

}

return newObj;

}

let a={

name:"123",

value:123

};

let b=deepCopy(a)

#### 什么是虚拟DOM？

虚拟DOM就是用JS来模拟DOM结构。

虚拟DOM就是为了解决这个浏览器性能问题而被设计出来的，如果一次操作中有10次更新DOM的操作，虚拟DOM不会立即操作DOM，而是将这10次更新的diff内容保存在本地的一个js对象中，最终将这个js对象一次性attach到DOM树上，通知浏览器去执行绘制工作，这样可以避免大量的无谓的计算量。

#### 如何进行深拷贝和浅拷贝

浅拷贝

let a=[0,1,2,3,4],

b=a;

console.log(a===b);

a[0]=1;

console.log(a,b);

深拷贝使用递归

const deepCopy=(obj)=>{

//如果被克隆的是简单数据对象，没必要进行深度克隆

if(typeof obj!=='object'){

return;

}

//判断被克隆的是对象还是数组

let newObj=obj instanceof Array?[]:{};

//对每一个属性进行判断

for (let key in obj){

//只对自身的属性进行赋值

if(obj.hasOwnProperty(key)){

//如果该属性值也是对象，递归调用深度克隆方法

newObj[key]=typeof obj[key]==="object"?deepCopy(obj[key]):obj[key];

}

}

return newObj;

}

let a={

name:"123",

value:123

};

let b=deepCopy(a)

#### 怎么用一行代码实现数组去重？

let array = Array.from(new Set([1, 1, 1, 2, 3, 2, 4]));

console.log(array);

// => [1, 2, 3, 4]

#### JS和jquery的区别？

1、jquery与js最直观的区别就是外观上jQuery对象比js对象多了"$()"。

2、jquery与js操作内容的区别

3、jquery与js操作属性的区别

4、jquery与js操作样式的区别

JS操作样式的方法只能获取内联样式，不能取内嵌的和外部的。

JQUERY操作样式的方法可以是内联的也可以是内嵌的。

#### Jquery和zepto的区别

1、Zepto更轻量级

2、Zepto是jQuery的精简，针对移动端去除了大量jQuery的兼容代码

3、部分API的实现方式不同

4、zepto针对移动端程序

#### 上拉刷新和上拉加载的原理？

实现下拉刷新主要分为三步：

监听原生touchstart事件，记录其初始位置的值，e.touches[0].pageY；

监听原生touchmove事件，记录并计算当前滑动的位置值与初始位置值的差值，大于0表示向下拉动，并借助CSS3的translateY属性使元素跟随手势向下滑动对应的差值，同时也应设置一个允许滑动的最大值；

监听原生touchend事件，若此时元素滑动达到最大值，则触发callback，同时将translateY重设为0，元素回到初始位置。

上拉加载更多数据是在页面滚动时触发的动作，一般是页面滚动到底部时触发，也可以选择在滚动到一定位置的时候触发。

以滚动到页面底部为例。实现原理是分别获得当前滚动条的scrollTop值、当前可视范围的高度值clientHeight以及文档的总高度scrollHeight。当scrollTop和clientHeight的值之和大于等于scrollHeight时，触发callback。

#### 怎么将一个HTML元素加到DOM树中？

jQuery提供了append（）、appendTo（）等相关方法 可以将一个HTML元素添加到DOM树中

JS提供了appendChild()

#### jquery对象和DOM对象是怎么进行转换的？

jquery对象转成dom对象：

1.$("#id名")[0] //id选择器

2.$("#id名").get(0) //id选择器

3.$("标签名").eq(0)[0] 或 $("标签名").get(0) //下标从0开始

dom对象转成jquery对象：

var jq=$(domObj);

#### jquery是如何设置文本值和属性值？

text() - 设置或返回所选元素的文本内容

html() - 设置或返回所选元素的内容（包括 HTML 标记）

val() - 设置或返回表单字段的值

设置属性 - attr()

jQuery attr() 方法也用于设置/改变属性值。

$("button").click(function(){

$("#w3s").attr("href","http://www.w3school.com.cn/jquery");

});

#### jquery动画怎么显示和隐藏？

使用show()， 不带有参数， 表示让指定的元素直接显示出来。

使用hide()， 不带有参数， 表示让指定的元素直接隐藏。

可以通过 toggle() 便捷的实现显示和隐藏的来回切换

#### 浏览器如何进行前进和后退？

前进: history.forward();=history.go(1); 后退: history.back();=history.go(-1);

#### 常见的布局方式有哪些？

1.静态布局

2.弹性布局

3.自适应布局

4.流式布局

5.响应式布局

#### Git fetch 和 git pull的区别？

etch 只能更新远程仓库的代码为最新的，本地仓库的代码还未被更新，我们需要通过 git merge origin/master 来合并这两个版本，你可以把它理解为合并分支一样的。

pull 操作是将本地仓库和远程仓库（本地的）更新到远程的最新版本。

#### 如何判断是否是一个数组？

1、从原型入手，Array.prototype.isPrototypeOf(obj);

2、也可以从构造函数入手，obj instanceof Array；

3、根据对象的class属性(类属性)，跨原型链调用toString()方法。

4、Array.isArray()方法。

#### ios会有滑动卡顿现象怎么解决？（移动端的兼容问题）

css属性解决

-webkit-overflow-scrolling:touch;

#### JS加载文件位置的区别？

页面嵌入<script>标签内联

页面嵌入<script>标签外联

直接在HTML元素中加入

1、先加载和后加载的区别。

浏览器解析html是从上到下的。如果把javascript放在head里，则先被解析。需要用window.onload起到一个延迟加载的作用。JavaScript脚本写在body里面，也要看具体位置。一般都是写在了body标签的最后面。

2、javascript代码默认是同步执行的，不管是内嵌的还是外部的javascript代码，都会阻塞后续html代码的加载和渲染以及后续javascript的解析。这是个必要的特性，试想一下，如果你要使用某个功能，例如jquery 库的功能，把jquery 库放在你是用的代码之前（一般放在head里），这样你在其后的任何位置都可以无限制的使用jquery 的功能而不用担心其jquery 函数功能是否已经被解析载入。

3、但如果你这有一个javascript文件比较大，而这一部分的功能比较独立，没有他也不会影响页面的显示效果，那就可以考虑把它放在body的最后，即/body之前，这样整个页面载入了，页面完全显示了才开始处理这部分javascript，当然，也可以loadjs 等异步加载方式进行异步加载，这样就不会阻塞其他html和javasscript了。

4、一般情况下最好是单独把javascript放在js文件里，通过head里的链接起来，css则是通过link。这样做的目的是为了让内容-样式-逻辑分离，以便以后能够更好地维护

#### 长时间按住页面会闪退怎么解决？

写一段代码就完事了

element {

-webkit-touch-callout: none;

}

#### ajax中信息如何在浏览器和服务器之间进行传输？

Ajax与服务端是通过字符串或者xml进行交互的

#### XMLHTTP Reuest常用方法又哪些？

readyState

当前HTTP 请求的状态，只读属性。

responseText

从服务器返回的当前HTTP请求的响应信息，或者称为目前为止从服务器接收到的响应体，文本格式的信息，只读属性。

responseXML

从服务器返回的当前HTTP请求的响应信息，解析为 XML 并作为 Document 对象返回，只读属性。

status

从服务器返回的当前HTTP请求的状态码，使用数字表示，只读属性。

statusText

从服务器返回的当前HTTP请求的响应行状态，使用名称表示，只读属性。

responseBody 将响应信息正文以unsigned byte数组形式返回

responseStream 以ADO Stream对象的形式返回响应信息

#### 什么是XML？

XML文件是XML（可扩展标记语言）数据文件。它的格式与.HTML文档非常相似，但使用自定义标记来定义对象和每个对象中的数据。XML文件可以被认为是基于文本的数据库。

#### 移动端的兼容问题和PC端的兼容问题各总结十条？

移动端

① 安卓浏览器看背景图片，有些设备会模糊，原因是手机的分辨率太小

解决方案： 用2X图片来代替img标签，然后background-size: contain

② 防止手机中页面放大或缩小： 在meta中设置viewport user-scalable = no

③ 上下拉滚动条卡顿： overflow-scrolling: touch;

④ 禁止复制选中文本： user-select: none;

⑤ 长时间按住页面出现闪退：-webkit-touch-callout: none;

⑥ 动画定义3D硬件加速： transform: translate 3d(0,0,0);

⑦ formate-detection 启动或禁止自动识别页面中的电话号码，content = "yes/no"

⑧ a标签添加tel是拨号功能

⑨ 安卓手机的圆角失效： background-clip: padding-box;

⑩ 手机端300ms延迟： fastclick

PC端

① rgba不支持IE8 用opacity属性代替rgba设置透明度

② 图片加a标签在IE9中出现边框 解决方案： img{border: none;}

③ IE6不支持display: inline-block 设置为： display: inline

④ position : fixed 不支持IE5/IE6

⑤ IE6，Firfox中，width = width + padding + border

⑥ min-height比兼容IE6/IE7

#### 移动端的300ms怎么解决？

方案一：禁用缩放

<meta name="viewport" content="user-scalable=no">

<meta name="viewport" content="initial-scale=1,maximum-scale=1">

方案二：更改默认的视口宽度

<meta name="viewport" content="width=device-width">

方案三：CSS touch-action

是touch-action这个CSS属性。这个属性指定了相应元素上能够触发的用户代理（也就是浏览器）的默认行为。如果将该属性值设置为touch-action: none，那么表示在该元素上的操作不会触发用户代理的任何默认行为，就无需进行300ms的延迟判断。

#### H5的新特性？

用于绘画的 canvas 元素

用于媒介回放的 video 和 audio 元素

对本地离线存储的更好的支持

新的特殊内容元素，比如 article、footer、header、nav、section

新的表单控件，比如 calendar、date、time、email、url、search

#### CSS3的新特性？

CSS3实现圆角（border-radius），阴影（box-shadow），

对文字加特效（text-shadow、），线性渐变（gradient），旋转（transform）

transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);// 旋转,缩放,定位,倾斜

增加了更多的CSS选择器 多背景 rgba

在CSS3中唯一引入的伪元素是 ::selection.

媒体查询，多栏布局

border-image

#### Call,apply和bind的区别？

call、apply和bind都是用于改变方法中this指向的。

call调用 将方法中的this指向call中第一个参数，当第一个参数为null、undefined时，默认指向window; call中第一个参数之后是要传递给方法的参数列表。

apply与call相似，不同之处在于传递给方法的参数形式不一致。apply传递给方法的参数是数组的形式。

call和apply在改变方法的this指向时，会同时执行方法；而bind不会执行方法，而是返回改变this指向后的新方法。

#### JavaScript中的typeof会返回哪些数据类型？

　undefined

　　string

　　boolean

　　number

　　symbol(ES6)

　　Object

　　Function

#### 强制类型转换和隐式类型转化有哪些？

强制类型转换

Number(参数)把任何类型转换成数值类型

parseInt（参数1，参数2）将字符串转换成整数

parseFloat()将字符串转换成浮点数字

string(参数)：可以将任何类型转换成字符串

Boolean()可以将任何类型的值转换成布尔值

隐式类型转化

+ - \* / == ===

#### ES6中promise的原理是什么？

Promise对象是一个构造函数，用来生成Promise实例。通过在函数内部return 一个 Promise对象的实例，这样就可以使用Promise的属性和方法进行下一步操作了。

#### 解释一下ES6中的模板字符串?

反撇号(键盘上Tab键上面那个)基础知识

除了使用反撇号字符代替普通字符串的引号 ‘ 或 ” 外，它们看起来与普通字符串并无二致。

在最简单的情况下，它们与普通字符串的表现一致。但是我们并没有说：

“原来只是被反撇号括起来的普通字符串啊”。模板字符串名之有理，

它为JavaScript提供了简单的字符串插值功能，从此以后，你可以通过一种更加美观、

更加方便的方式向字符串中插值了。

#### disable和readonly的区别是什么？

（1） 表面上可看到的区别就是当这两个词都设置为true时，都为禁用状态，当鼠标移上时使用disable的相关控件时鼠标出现禁用样式，并且不可做任何操作，而Readonly还可以获取文本框里的焦点。

（2） Disable比readonly的使用范围比广，适用文本框、文本域、下拉框、button按钮、单选框…….而readonly只适用于input（text、passwork、textarea）。

（3） Disable设置为true之后是不可以向后台提交数据的，此时可以选择改用readonly进行禁用，或者在提交数据时取消禁用。

#### $.fn.extend和$.extend的区别是什么？

$.extend(obj);是为了扩展jquery本身，为类添加新的方法

$.fn.extend(obj);给JQUERY对象添加方法。

#### JSON和JSONP的区别？

ajax和jsonp这两种技术在调用方式上“看起来”很像，目的也一样，都是请求一个url，然后把服务器返回的数据进行处理，因此jquery和ext等框架都把jsonp作为ajax的一种形式进行了封装；

但ajax和jsonp其实本质上是不同的东西。ajax的核心是通过XmlHttpRequest获取非本页内容，而jsonp的核心则是动态添加<script>标签来调用服务器提供的js脚本。

所以说，其实ajax与jsonp的区别不在于是否跨域，ajax通过服务端代理一样可以实现跨域，jsonp本身也不排斥同域的数据的获取。

还有就是，jsonp是一种方式或者说非强制性协议，如同ajax一样，它也不一定非要用json格式来传递数据，如果你愿意，字符串都行，只不过这样不利于用jsonp提供公开服务。

jsonp不是ajax的一个特例，哪怕jquery等巨头把jsonp封装进了ajax，也不能改变着一点！

#### json对象和json数组的区别

JSON 对象在花括号中书写，对象可以包含多个名称/值对：

JSON 数组在方括号中书写，数组可包含多个对象：

#### jquery选择器和css选择器有哪些区别？

1、但两者的作用不同，CSS选择器找到元素后为设置该元素的样式，jQuery选择器找到元素后添加行为。

2、jQuery选择器拥有更好的跨浏览器的兼容性。

3、选择器的效率。

#### jquery和jqueryUi的区别？

1.jQuery是一个js库,主要提供的功能是选择器,属性修改和事件绑定等等。

2.jQuery UI则是在jQuery的基础上,利用jQuery的扩展性,设计的插件。提供了一些常用的界面元素,诸如对话框、拖动行为、改变大小行为等等。

3.jQuery本身注重于后台,没有漂亮的界面,而jQuery UI则补充了前者的不足,他提供了华丽的展示界面,使人更容易接受。既有强大的后台,又有华丽的前台。

#### transition动画和animate动画的区别？

transition需要触发一个事件才会随着时间改变其CSS属性；animation在不需要触发任何事件的情况下，也可以显式的随时间变化来改变元素CSS属性，达到一种动画的效果

1）动画不需要事件触发，过渡需要。

2）过渡只有一组（两个：开始-结束） 关键帧，动画可以设置多个。

#### 如何用animation定义一个动画？

第一种方式：

@keyframes myfirst

{

from {background: red;}

to {background: yellow;}

}

第二种方式：

@keyframes myfirst

{

0% {background: red;}

25% {background: yellow;}

50% {background: blue;}

100% {background: green;}

}

调用：

第一种：

div

{

animation-name: myfirst;

animation-duration: 5s;

animation-timing-function: linear;

animation-delay: 2s;

animation-iteration-count: infinite;

animation-direction: alternate;

animation-play-state: running;

}

第二种：

div

{

animation: myfirst 5s linear 2s infinite alternate;

}

#### 什么是作用域？什么是作用域链？

变量的作用域分为两种：全局变量和局部变量。

1、全局作用域： 最外层函数定义的变量拥有全局作用域，即对任何内部函数来说，都是可以访问的；

2、局部作用域： 局部作用域一般只在固定的代码片段内可访问到，而对于函数外部是无法访问的，最常见的例如函数内部。

当代码在一个环境中执行时，会创建变量对象的一个作用域链。

由子级作用域返回父级作用域中寻找变量，就叫做作用域链。

#### 什么是原型？什么是原型链？

JS中每个函数都存在有一个原型对象属性prototype。并且所有函数的默认原型都是Object的实例。

原型链，简单理解就是原型组成的链，对象的\_\_proto\_\_它的是原型，而原型也是一个对象，

也有\_\_proto\_\_属性，原型的\_\_proto\_\_又是原型的原型，就这样可以一直通过\_\_proto\_\_想上找，

这就是原型链，当向上找找到Object的原型的时候，这条原型链就算到头了。

#### async和await的区别？

1）函数前面多了一个aync关键字。await关键字只能用在aync定义的函数内。async函数会隐式地返回一个promise，该promise的reosolve值就是函数return的值。(示例中reosolve值就是字符串”done”)

2）第1点暗示我们不能在最外层代码中使用await，因为不在async函数内。

#### rgba和opacity的区别？

opacity会继承父元素的 opacity 属性，而RGBA设置的元素的后代元素不会继承不透明属性。

#### cookie和session的区别？

cookie原理：

- 从服务器端向客户端浏览器留下信息`设置响应头：set-cookie`；

- 浏览器每次访问服务器时都带上这些信息(自动携带cookie是浏览器的特点)；

session原理：

- 服务器端会为每个用户（浏览器）各自保存一个session（文件）。当服务器保存session之后，会以cookie的形式告诉浏览器，你的session编号是哪一个。它把session号返回给了浏览器，而把真实的数据保存在服务器。

- 下次再来访问服务器的时候，浏览器就会带着它自己的session号去访问，服务器根据session号就可以找到对应的session了。

cookie：优点是节省服务器空间，缺点不安全。不要保存敏感信息。

session：优点是安全，缺点需要服务器空间， 是一种最常见的解决方案。

#### rgba不支持IE8怎么解决？

这里可以使用IE的filter来解决这个问题，css代码如下：background: rgba(255,255,255,0.1); filter:progid:DXImageTransform.Microsoft.gradient(startColorstr=#19ffffff,endColorstr=#19ffffff);这里要解释一下第二句话的意思，就是当上一行的透明度不起作用的时候执行。这句话的意思本来是用来做渐变的。但是这个地方不需要渐变。所以两个颜色都设置成了相同的颜色。

#### 常见的服务器有哪些？

不同的服务器应用对应就有一些常用的服务器软件：

比如：web服务器常用软件有：

提供web服务的IIS，Apache，Tomcat等，

数据库服务的MySQL，oracle，sqlserver等，

服务器程序语言：php、ASP、.net等软件等，

ftp服务的有：Serv-U、FileZilla 等。

#### ES6中的Class和构造函数的区别？

Class 类中不存在变量提升

class内部会启用严格模式

class的所有方法都是不可枚举的

class 必须使用new调用

class 内部无法重写类名

class 的继承有两条继承链

一条是： 子类的\_\_proto\_\_ 指向父类

另一条： 子类prototype属性的\_\_proto\_\_属性指向父类的prototype属性.

es6的子类可以通过\_\_proto\_\_属性找到父类，而es5的子类通过\_\_proto\_\_找到Function.prototype

es5 与 es6子类this的生成顺序不同。

es5的继承是先建立子类实例对象this,再调用父类的构造函数修饰子类实例(Surper.apply(this))。

es6的继承是先建立父类实例对象this，在调用子类的构造函数修饰this。即在子类的constructor方法中必须使用super()，之后才能使用this.

正是因为this的生成顺序不同，所有es5不能继承原生的构造函数，而es6可以继承

#### Iframe标签优缺点？

优点：便于修改，模块分离，像一些信息管理系统会用到。 但现在基本上不推荐使用。除非特殊需要，一般不推荐使用。

缺点：（1）iframe 的创建比一般的 DOM 元素慢了 1-2 个数量级 （2）iframe 标签会阻塞页面的加载，如果页面的 onload 事件不能及时触发， 会让用户觉得网页加载很慢，用户体验不好.在 Safari 和 Chrome 中可以通过 js 动态设置 iframe 的 src 属性来避免阻

#### 如何解决IE8 PNG图片无法显示

加入一个注册表项。

一、新建记事本，写入如下内容：

Windows Registry Editor Version 5.00

[HKEY\_CLASSES\_ROOT\MIME\Database\Content Type\image/png]

"Extension"=".png"

"Image Filter CLSID"="{A3CCEDF7-2DE2-11D0-86F4-00A0C913F750}"

[HKEY\_CLASSES\_ROOT\MIME\Database\Content Type\image/png\Bits]

"0"=hex:08,00,00,00,ff,ff,ff,ff,ff,ff,ff,ff,89,50,4e,47,0d,0a,1a,0a

二、保存，关闭

三、修改此记事本为.reg扩展名

四、双击加入注册表

#### 优雅降级和渐进增强的区别？

优雅降级和渐进增强印象中是随着 css3 流出来的一个概念。由于低级浏览器不支持 css3，但 css3 的效果又太优秀不忍放弃，所以在高级浏览中使用 css3 而低级浏览器只保证 最基本的功能。咋一看两个概念差不多，都是在关注不同浏览器下的不同体验，关键的区别 是他们所侧重的内容，以及这种不同造成的工作流程的差异。 举个例子： a{ display：block； width：200px； height： 100px； background：aquamarine； /\*我就是要用这个新 css 属性\*/ transition：all 1s ease 0s； /\*可是发现了一些低版本浏览器不支持怎么吧\*/ /\*往下兼容\*/ -webkit-transition：all 1s ease 0s； -moz-transition：all 1s ease 0s； -o-transition： all 1s ease 0s； /\*那么通常这样考虑的和这样的侧重点出发的 css 就是优雅降级\*/ } 北京市顺义区京顺路 99 号·黑马程序员 www.itheima.com 第 8 页 共 125 页

a：hover{ height：200px； }/ \*那如果我们的产品要求我们要重低版本的浏览器兼容开始\*/ a{ /\*优先考虑低版本的\*/ -webkit-transition：all 1s ease 0s； -moz-transition：all 1s ease 0s； -o-transition： all 1s ease 0s； /\*高版本的就肯定是渐进渐强\*/ transition：all 1s ease 0s； }“优雅降级”观点认为应该针对那些最高级、最完善的浏览器来设计网站。 “渐进增强”观点则认为应关注于内容本身。

#### CSS3新增的伪类有哪些？

p:first-child 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。

p:last-child 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。

p:nth-child(2n) 选择属于其父元素的第二个子元素的每个 <p> 元素。

:enabled、:disabled 控制表单控件的禁用状态。

:checked，单选框或复选框被选中。

#### 如何阻止事件的默认行为？

w3c的方法是e.preventDefault()，IE则是使用e.returnValue = false;

#### JS的事件流模型？

JavaScript 中的事件流模型 事件冒泡 和 事件捕获，以及 事件委托（也叫事件代理）

#### Undefined 和 not defined的区别？

其中 定义之后，但是未经任何操作或者某对象的一个未定义的属性，都属于 可以拿来运算的 undefined ，可以用来当作逻辑运算   
第二种的变量，完全没有定义，而且没有任何背景的（x.a,虽然x无a属性，但是给x面子），属于报错类型不能被使用，除非用typeof函数   
js中的或运算，并不返回bool值，而是返回js对两边条件最后一个计算的值，比如 1||2 1为真，2不需要计算，所以 1||2 == 1 a||1 报错   
var a; a|| 1 == 1;

#### 节点类型有哪几种？

元素节点 　　 Node.ELEMENT\_NODE(1)

属性节点 　　 Node.ATTRIBUTE\_NODE(2)

文本节点 　　 Node.TEXT\_NODE(3)

CDATA节点 Node.CDATA\_SECTION\_NODE(4)

实体引用名称节点 　 Node.ENTRY\_REFERENCE\_NODE(5)

实体名称节点 　　 Node.ENTITY\_NODE(6)

处理指令节点 　　 Node.PROCESSING\_INSTRUCTION\_NODE(7)

注释节点 　 Node.COMMENT\_NODE(8)

文档节点 　 Node.DOCUMENT\_NODE(9)

文档类型节点 　　 Node.DOCUMENT\_TYPE\_NODE(10)

文档片段节点 　　 Node.DOCUMENT\_FRAGMENT\_NODE(11)

DTD声明节点 Node.NOTATION\_NODE(12)

#### Link和@import的区别？

1.从属关系区别

@import是 CSS 提供的语法规则，只有导入样式表的作用；link是HTML提供的标签，不仅可以加载 CSS 文件，还可以定义 RSS、rel 连接属性等。

2.加载顺序区别

加载页面时，link标签引入的 CSS 被同时加载；@import引入的 CSS 将在页面加载完毕后被加载。

3.兼容性区别

@import是 CSS2.1 才有的语法，故只可在 IE5+ 才能识别；link标签作为 HTML 元素，不存在兼容性问题。

4.DOM可控性区别

可以通过 JS 操作 DOM ，插入link标签来改变样式；由于 DOM 方法是基于文档的，无法使用@import的方式插入样式。

#### overflow的属性有哪几种？

|  |  |
| --- | --- |
| **值** | **描述** |
| visible | 默认值。内容不会被修剪，会呈现在元素框之外。 |
| hidden | 内容会被修剪，并且其余内容是不可见的。 |
| scroll | 内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容。 |
| auto | 如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容。 |
| inherit | 规定应该从父元素继承 overflow 属性的值。 |

#### src和href的区别？

src（source）指向外部资源的位置，指向的内容将会嵌入到文档中当前标签所在位置；在 请求 src 资源时会将其指向的资源下载并应用到文档中，如 js 脚本，img 图片和 iframe 等 元素。 当浏览器解析到该元素时，会暂停其他资源的下载和处理，直到将该资源加载、编译、执行 完毕，类似于将所指向资源嵌入当前标签内。 href（hypertext reference/超文本引用）指向网络资源所在位置，建立和当前元素（锚点） 或当前文档（链接）之间的链接，如果我们在文档中添加<link href="common.cssrel="stylesheet"/>那么浏览器会识别该文档为 css 文件，就会并行下载资源并且不会停止 对当前文档的处理。

#### 什么是函数声明？什么是变量声明提升？

通过function关键字定义的函数和var关键字定义的变量在预解析时会被提升到当前作用域的最顶端

#### detach和remove的区别？

尽管 detach() 和 remove() 方法都被用来移除一个DOM元素, 两者之间的主要不同在于 detach() 会保持对过去被解除元素的跟踪, 因此它可以被取消解除, 而 remove() 方法则会保持过去被移除对象的引用. 你也还可以看看 用来向DOM中添加元素的 appendTo() 方法。

#### 什么是函数节流与防抖？

函数节流：是确保函数特定的时间内至多执行一次。  
函数防抖：是函数在特定的时间内不被再调用后执行。

他们都是可以防止一个函数被无意义的高频率调用

#### 常用数组方法有哪些？

方法 描述

concat() 连接两个或更多的数组，并返回结果。

copyWithin() 从数组的指定位置拷贝元素到数组的另一个指定位置中。

entries() 返回数组的可迭代对象。

every() 检测数值元素的每个元素是否都符合条件。

fill() 使用一个固定值来填充数组。

filter() 检测数值元素，并返回符合条件所有元素的数组。

find() 返回符合传入测试（函数）条件的数组元素。

findIndex() 返回符合传入测试（函数）条件的数组元素索引。

forEach() 数组每个元素都执行一次回调函数。

from() 通过给定的对象中创建一个数组。

includes() 判断一个数组是否包含一个指定的值。

indexOf() 搜索数组中的元素，并返回它所在的位置。

isArray() 判断对象是否为数组。

join() 把数组的所有元素放入一个字符串。

keys() 返回数组的可迭代对象，包含原始数组的键(key)。

lastIndexOf() 搜索数组中的元素，并返回它最后出现的位置。

map() 通过指定函数处理数组的每个元素，并返回处理后的数组。

pop() 删除数组的最后一个元素并返回删除的元素。

push() 向数组的末尾添加一个或更多元素，并返回新的长度。

reduce() 将数组元素计算为一个值（从左到右）。

reduceRight() 将数组元素计算为一个值（从右到左）。

reverse() 反转数组的元素顺序。

shift() 删除并返回数组的第一个元素。

slice() 选取数组的的一部分，并返回一个新数组。

some() 检测数组元素中是否有元素符合指定条件。

sort() 对数组的元素进行排序。

splice() 从数组中添加或删除元素。

toString() 把数组转换为字符串，并返回结果。

unshift() 向数组的开头添加一个或更多元素，并返回新的长度。

valueOf() 返回数组对象的原始值。

#### 常用字符串方法有哪些？

方法 描述

charAt() 返回在指定位置的字符。

charCodeAt() 返回在指定的位置的字符的 Unicode 编码。

concat() 连接两个或更多字符串，并返回新的字符串。

fromCharCode() 将 Unicode 编码转为字符。

indexOf() 返回某个指定的字符串值在字符串中首次出现的位置。

includes() 查找字符串中是否包含指定的子字符串。

lastIndexOf() 从后向前搜索字符串，并从起始位置（0）开始计算返回字符串最后出现的位置。

match() 查找找到一个或多个正则表达式的匹配。

repeat() 复制字符串指定次数，并将它们连接在一起返回。

replace() 在字符串中查找匹配的子串， 并替换与正则表达式匹配的子串。

search() 查找与正则表达式相匹配的值。

slice() 提取字符串的片断，并在新的字符串中返回被提取的部分。

split() 把字符串分割为字符串数组。

startsWith() 查看字符串是否以指定的子字符串开头。

substr() 从起始索引号提取字符串中指定数目的字符。

substring() 提取字符串中两个指定的索引号之间的字符。

toLowerCase() 把字符串转换为小写。

toUpperCase() 把字符串转换为大写。

trim() 去除字符串两边的空白

toLocaleLowerCase() 根据本地主机的语言环境把字符串转换为小写。

toLocaleUpperCase() 根据本地主机的语言环境把字符串转换为大写。

valueOf() 返回某个字符串对象的原始值。

toString() 返回一个字符串。

#### 同步和异步的区别？

**同步**：同步任务指的是，在主线程上排队执行的任务，只有前一个任务执行完毕，才能执行后一个任务。

**异步**：不进入主线程、而进入"任务队列"（task queue）的任务，只有等主线程任务执行完毕，"任务队列"开始通知主线程，请求执行任务，该任务才会进入主线程执行。

#### Window.onload和document.ready的区别

window.onload 是在dom文档树加载完和所有文件加载完之后执行一个函数Document.ready原生种没有这个方法，jquery中有 $().ready(function),在dom文档树加载完之后执行一个函数（注意，这里面的文档树加载完不代表全部文件加载完）。

$(document).ready要比window.onload先执行

window.onload只能出来一次，$(document).ready可以出现多次

#### 文件编码，url编码和Unicode编码的区别是什么？

url编码是一种浏览器用来打包表单输入的格式。浏览器从表单中获取所有的name和其中的值 ，将它们以name/value参数编码（移去那些不能传送的字符，将数据排行等等）作为URL的一部分或者分离地发给服务器。不管哪种情况，在服务器端的表单输入格式样子象这样：

theName=Ichabod+Crane&gender=male&status=missing& ;headless=yes

[Unicode](https://baike.baidu.com/item/Unicode" \t "https://baike.baidu.com/item/Unicode%E7%A0%81/_blank)是一个编码方案，仅仅提供字符与编号间的映射，Unicode 是为了解决传统的字符编码方案的局限而产生的，它为每种语言中的每个字符设定了统一并且唯一的[二进制](https://baike.baidu.com/item/%E4%BA%8C%E8%BF%9B%E5%88%B6" \t "https://baike.baidu.com/item/Unicode%E7%A0%81/_blank)编码，以满足跨语言、跨平台进行文本转换、处理的要求。Unicode 编码共有三种具体实现，分别为utf-8,utf-16,utf-32

#### 实现一个快速排序

// 快排

function quickSort(arr, i, j) {

if(i < j) {

let left = i;

let right = j;

let pivot = arr[left];

while(i < j) {

while(arr[j] >= pivot && i < j) { // 从后往前找比基准小的数

j--;

}

if(i < j) {

arr[i++] = arr[j];

}

while(arr[i] <= pivot && i < j) { // 从前往后找比基准大的数

i++;

}

if(i < j) {

arr[j--] = arr[i];

}

}

arr[i] = pivot;

quickSort(arr, left, i-1);

quickSort(arr, i+1, right);

return arr;

}

}

// example

let arr = [2, 10, 4, 1, 0, 9, 5 ,2];

console.log(quickSort(arr, 0 , arr.length-1));

#### Js的执行过程分为哪几个阶段？

语法分析

预编译阶段

执行阶段

#### 异步执行的时候整个周期是什么？

这个问题看到在写代码的时候首先是没有保证一个函数只处理一件事件,例如:获取缓存路径的应该放到写缓存的外面.作为一个参数传递进来

异步方法中一定要注意使用到的全局变量的生命周期

由于项目比较大，解决这个问题我没有使用调试的方式，Debug断点调试也许就没有这个问题了。使用日志记录调试信息提高了不少效率

#### 基本数据类型存在哪里？垃圾回收时有什么区别？

基本数据类型保存在栈内存，引用数据类型（复杂数据类型）实际上是一个指针，这个指针也保存在栈中，但是这个指针指向的对象则保存在堆内存中。

堆中存放对象，需要手动释放内存。（垃圾回收机制）

栈中存放局部变量，内存的释放是系统控制实现的

#### Pop，push ，shift，unshift的区别是什么

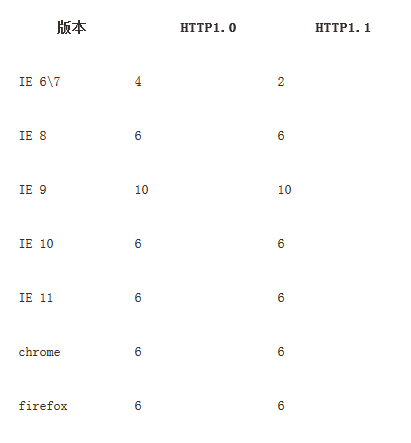
Pop删除数组最后一个元素

Push是向数组最后添加元素

Shift删除数组第一个元素

UNshift向数组第一个位置添加元素

#### 浏览器并发请求资源上限是多少



#### 什么是JSONP

Jsonp并不是一种数据格式，而json是一种数据格式，jsonp是用来解决跨域获取数据的一种解决方案，具体是通过动态创建script标签，然后通过标签的src属性获取js文件中的js脚本，该脚本的内容是一个函数调用，参数就是服务器返回的数据，为了处理这些返回的数据，需要事先在页面定义好回调函数，本质上使用的并不是ajax技术

#### 如何实现HTML5手机端的适配

1、使用meta标签：viewport

H5移动端页面自适应普遍使用的方法，理论上讲使用这个标签是可以适应所有尺寸的屏幕的，

但是各设备对该标签的解释方式及支持程度不同造成了不能兼容所有浏览器或系统。

2、使用css3单位rem

rem是CSS3新增的一个相对单位（root em，根em）,使用rem为元素设定字体大小时，

是相对大小，但相对的只是HTML根元素。通过它既可以做到只修改根元素就成比例地调整所有字体大小，

又可以避免字体大小逐层复合的连锁反应。

3、使用媒体查询

媒体查询也是css3的方法，我们要解决的问题是适应手机屏幕，这个媒体查询正是为解决这个问题而生。

媒体查询的功能就是为不同的媒体设置不同的css样式，这里的“媒体”包括页面尺寸，设备屏幕尺寸等。

4、使用百分比

百分比指的是父元素，所有百分比都是这样的。子元素宽度50%，那么父元素的宽度就是100%；

#### Call,apply,bind的区别是什么？分别适用于什么样的场景

三者都可以把一个函数应用到其他对象上，注意不是自身对象．apply,call是直接执行函数调用，bind是绑定，执行需要再次调用．apply和call的区别是apply接受数组作为参数，而call是接受逗号分隔的无限多个参数列表，call常用在继承中，apply常用在指向数组的时候，bind常用在定时器中。

#### 编写一个方法，去掉数组里重复的内容

function unique(arr) {

var array= arr;

var len = array.length;

array.sort(function(a,b){ //排序后更加方便去重

return a - b;

})

function loop(index){

if(index >= 1){

if(array[index] === array[index-1]){

array.splice(index,1);

}

loop(index - 1); //递归loop，然后数组去重

}

}

loop(len-1);

return array;

}

或者[...new Set(arr)]

#### 如何去除URL里边的#号

vue-router 默认使用 hash 模式，所以在路由加载的时候，项目中的 url 会自带 #。如果不想使用 #， 可以使用 vue-router 的另一种模式 history

new Router({

mode: 'history',

base: process.env.BASE\_URL,

routes

})

#### 常见的图片格式有哪些

图片常见的有很多种像 bmp , jpg ,png, tiff , gif ,pcx,tga,exif,fpx,svg,psd,cdr,pcd,dxf,ufo,eps,ai,raw,WMF等。

#### substr和substring的区别是什么

substr(N1,N2) 这个就是我们常用的从指定的位置(N1)截取指定长度(N2)的字符串;

substring(N1,N2) 这个就是我们常用的从指定的位置(N1)到指定的位置(N2)的字符串;

#### 对于web语义化的理解

1. HTML语义化让页面的内容结构化，结构更清晰，便于对浏览器、搜索引擎解析；

2. 即使在没有样式CSS 的情况下也能以一种文档格式显示，并且是容易阅读的；

3. 搜索引擎的爬虫也依赖于HTML标记来确定上下文和各个关键字的权重，有利于SEO；

4. 使阅读源代码的人更容易将网站分块，便于阅读、维护和理解；

#### Display:inline-block什么时候会显示空隙

元素被当成行内元素排版的时候，元素之间的空白符（空格、回车换行等）都会被浏览器处理，根据white-space的处理方式(默认是normal,空白会被浏览器忽略)，HTML代码中的回车换行被转成一个空白符，在字体不为0的情况下，空白符占据一定宽度，所以inline-block的元素之间就出现了空隙。

解决方法：移除空格

使用margin负值

使用font-size:0

letter-spacing

word-spacing

#### 网页的重构，重排和重绘分别是什么

重绘是一个元素外观的改变所触发的浏览器行为，例如改变visibility、outline、背景色等属性。浏览器会根据元素的新属性重新绘制，使元素呈现新的外观。重绘不会带来重新布局，并不一定伴随重排。

重排是更明显的一种改变，可以理解为渲染树需要重新计算。下面是常见的触发重排的操作：

1.DOM元素的几何属性变化 2. DOM树的结构变化 3. 获取某些属性 4. 此外，改变元素的一些样式，调整浏览器窗口大小等等也都将触发重排。

注：重排对性能有很大的影响

页面重构：编写 CSS、让页面结构更合理化，提升用户体验，实现良好的页面效果和提升性能。

网站重构：在不改变外部行为的前提下，简化结构、添加可读性，而在网站前端保持一致的行为。也就是说是在不改变UI的情况下，对网站进行优化，在扩展的同时保持一致的UI。

对于传统的网站来说重构通常是：

表格(table)布局改为DIV+CSS

使网站前端兼容于现代浏览器(针对于不合规范的CSS、如对IE6有效的)

对于移动平台的优化

针对于SEO进行优化

深层次的网站重构应该考虑的方面：

减少代码间的耦合

让代码保持弹性

严格按规范编写代码

设计可扩展的API

代替旧有的框架、语言(如VB)

增强用户体验

优化响应速度

速度的优化重构：

压缩JS、CSS、image等前端资源(通常是由服务器来解决)

程序的性能优化(如数据读写)

采用CDN来加速资源加载

对于JS DOM的优化

HTTP服务器的文件缓存

#### jquery中的ajax有哪些参数

Url (默认: 当前页地址) 发送请求的地址。

type (默认: "GET") 请求方式 ("POST" 或 "GET")， 默认为 "GET"。注意：其它 HTTP 请求方法，如 PUT 和 DELETE 也可以使用，但仅部分浏览器支持。

timeout 设置请求超时时间（毫秒）。此设置将覆盖全局设置。

async (默认: true) 默认设置下，所有请求均为异步请求。如果需要发送同步请求，请将此选项设置为 false。注意，同步请求将锁住浏览器，用户其它操作必须等待请求完成才可以执行。

beforeSend 发送请求前可修改 XMLHttpRequest 对象的函数，如添加自定义 HTTP 头。XMLHttpRequest 对象是唯一的参数。

cache (默认: true) jQuery 1.2 新功能，设置为 false 将不会从浏览器缓存中加载请求信息。

Complete 请求完成后回调函数 (请求成功或失败时均调用)。参数： XMLHttpRequest 对象，成功信息字符串。

contentType (默认: "application/x-www-form-urlencoded") 发送信息至服务器时内容编码类型。默认值适合大多数应用场合。

Data 发送到服务器的数据。将自动转换为请求字符串格式。GET 请求中将附加在 URL 后。查看 processData 选项说明以禁止此自动转换。必须为 Key/Value 格式。如果为数组，jQuery 将自动为不同值对应同一个名称。dataType 预期服务器返回的数据类型。如果不指定，jQuery 将自动根据 HTTP 包 MIME 信息返回 responseXML 或 responseText，并作为回调函数参数传递，可用值:

"xml": 返回 XML 文档，可用 jQuery 处理。

"html": 返回纯文本 HTML 信息；包含 script 元素。

"script": 返回纯文本 JavaScript 代码。不会自动缓存结果。

"json": 返回 JSON 数据 。

"jsonp": JSONP 格式。使用 JSONP 形式调用函数时，如 "myurl?callback=?" jQuery 将自动替换 ? 为正确的函数名，以执行回调函数。

error (默认: 自动判断 (xml 或 html)) 请求失败时将调用此方法。这个方法有三个参数：XMLHttpRequest 对象，错误信息，（可能）捕获的错误对象。

global (默认: true) 是否触发全局 AJAX 事件。设置为 false 将不会触发全局 AJAX 事件，如 ajaxStart 或 ajaxStop 。可用于控制不同的Ajax事件

ifModified (默认: false) 仅在服务器数据改变时获取新数据。使用 HTTP 包 Last-Modified 头信息判断。

processData (默认: true) 默认情况下，发送的数据将被转换为对象(技术上讲并非字符串) 以配合默认内容类型 "application/x-www-form-urlencoded"。如果要发送 DOM 树信息或其它不希望转换的信息，请设置为 false。

success 请求成功后回调函数。这个方法有两个参数：服务器返回数据，返回状态

#### trim方法是干什么的

去除字符串两端的空格

#### 如何实现去除左空格和右空格

trimLeft trimRigth

#### 三个div等分排在同一行

使用流式布局宽度设置为33.33%

使用弹性布局设置子元素flex属性为1

#### flex有哪些属性值

一.父元素属性

1.display:flex;（定义了一个flex容器）

2. flex-direction（决定主轴的方向）

row（默认值，水平从左到右）colunm（垂直从上到下）row-reverse（水平从右到左）column-reverse（垂直从下到上）

3. flex-wrap（定义如何换行）

nowrap（默认值，不换行）wrap（换行）wrap-reverse（换行，且颠倒行顺序，第一行在下方）

4.flex-flow（属性是 flex-direction 属性和 flex-wrap 属性的简写形式，默认值为row nowrap）

5. justify-content（设置或检索弹性盒子元素在主轴（横轴）方向上的对齐方式）

flex-start（ 默认值、弹性盒子元素将向行起始位置对齐）

flex-end（弹性盒子元素将向行结束位置对齐）

center（弹性盒子元素将向行中间位置对齐。该行的子元素将相互对齐并在行中居中对齐）

space-between（弹性盒子元素会平均地分布在行里）

space-around（弹性盒子元素会平均地分布在行里，两端保留子元素与子元素之间间距大小的一半）

6.align-items（设置或检索弹性盒子元素在侧轴（纵轴）方向上的对齐方式）

flex-start（弹性盒子元素的侧轴（纵轴）起始位置的边界紧靠住该行的侧轴起始边界）

flex-end（弹性盒子元素的侧轴（纵轴）起始位置的边界紧靠住该行的侧轴结束边界）

center（ 弹性盒子元素在该行的侧轴（纵轴）上居中放置。（如果该行的尺寸小于弹性盒子元素的尺寸，则会向两个方向溢出相同的长度））

baseline（如弹性盒子元素的行内轴与侧轴为同一条，则该值与flex-start等效。其它情况下，该值将参与基线对齐。）

stretch（如果指定侧轴大小的属性值为'auto'，则其值会使项目的边距盒的尺寸尽可能接近所在行的尺寸，但同时会遵照'min/max-width/height'属性的限制）

7.align-content（设置或检索弹性盒堆叠伸缩行的对齐方式）

flex-start（各行向弹性盒容器的起始位置堆叠。弹性盒容器中第一行的侧轴起始边界紧靠住该弹性盒容器的侧轴起始边界，之后的每一行都紧靠住前面一行）

flex-end（各行向弹性盒容器的结束位置堆叠。弹性盒容器中最后一行的侧轴起结束界紧靠住该弹性盒容器的侧轴结束边界，之后的每一行都紧靠住前面一行）

center（各行向弹性盒容器的中间位置堆叠。各行两两紧靠住同时在弹性盒容器中居中对齐，保持弹性盒容器的侧轴起始内容边界和第一行之间的距离与该容器的侧轴结束内容边界与第最后一 行之间的距离相等）

space-between（各行在弹性盒容器中平均分布。第一行的侧轴起始边界紧靠住弹性盒容器的侧轴起始内容边界，最后一行的侧轴结束边界紧靠住弹性盒容器的侧轴结束内容边界，剩余的行则 按一定方式在弹性盒窗口中排列，以保持两两之间的空间相等）

space-around（ 各行在弹性盒容器中平均分布，两端保留子元素与子元素之间间距大小的一半。各行会按一定方式在弹性盒容器中排列，以保持两两之间的空间相等，同时第一行前面及最后 一行后面的空间是其他空间的一半）

stretch（各行将会伸展以占用剩余的空间。剩余空间被所有行平分，以扩大它们的侧轴尺寸）

二.子元素上属性

1.order（默认情况下flex order会按照书写顺训呈现，可以通过order属性改变，数值小的在前面，还可以是负数）

2.flex-grow（设置或检索弹性盒的扩展比率,根据弹性盒子元素所设置的扩展因子作为比率来分配剩余空间）

3.flex-shrink（设置或检索弹性盒的收缩比率,根据弹性盒子元素所设置的收缩因子作为比率来收缩空间）

4.flex-basis (设置或检索弹性盒伸缩基准值，如果所有子元素的基准值之和大于剩余空间，则会根据每项设置的基准值，按比率伸缩剩余空间)

5.flex (flex属性是flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto。后两个属性可选)

6.align-self (设置或检索弹性盒子元素在侧轴（纵轴）方向上的对齐方式，可以覆盖父容器align-items的设置)

#### transition和magin的百分比是根据什么设置的

transition是根据自身设置margin是根据定位来设置

#### 怎么实现秒针转动效果

<style>

.second-bar-box{

width: 100px;

height: 100px;

border-radius: 50%;

border: 1px solid #db192a;

position: relative;

}

.second-bar{

position: absolute;

left: 50%;

top: 0;

width: 2px;

height: 50px;

margin-left: -1px;

background-color: #ccc;

animation: barMove 60s linear infinite; /\*动画名称、持续时间、运动方式、重复形式（也可以写成具体次数，例如：5）\*/

transform-origin: center bottom; /\*控制以哪个点为中心点\*/

}

@keyframes barMove {

0% {

transform: rotateZ(0deg);

}

100% {

transform: rotateZ(360deg);

}

}

</style>

<div class="second-bar-box">

<div class="second-bar"></div>

</div>

#### 行内元素有哪些

<a>标签可定义锚

<abbr>表示一个缩写形式

<acronym>定义只取首字母缩写

<b>字体加粗

<bdo>可覆盖默认的文本方向

<big>大号字体加粗

<br>换行

<cite>引用进行定义

<code>定义计算机代码文本

<dfn>定义一个定义项目

<em>定义为强调的内容

<i>斜体文本效果

<img>向网页中嵌入一幅图像<input>输入框

<kbd>定义键盘文本

<label>标签为

<input> 元素定义标注（标记）

<q>定义短的引用

<samp>定义样本文本

<select>创建单选或多选菜单<small>呈现小号字体效果

<span>组合文档中的行内元素

<strong>语气更强的强调的内容

<sub>定义下标文本

<sup>定义上标文本

<textarea>多行的文本输入控件

<tt>打字机或者等宽的文本效果

<var>定义变量

#### 判断数组有哪些方法

1. a instanceof Array

2.a.consturctor ==Array

3.Object.prototype.toString.call(a) =='[Object Arral]'

#### 什么是图片懒加载

当打开一个有很多图片的页面时，先只加载页面上看到的图片，等滚动到页面下面时，再加载所需的图片。这就是图片懒加载。减少或延迟请求数，缓解浏览器的压力，增强用户体验。

#### ES6的promise

Promise 对象是一个代理对象（代理一个值），被代理的值在Promise对象创建时可能是未知的。它允许你为异步操作的成功和失败分别绑定相应的处理方法（handlers）。 这让异步方法可以像同步方法那样返回值，但并不是立即返回最终执行结果，而是一个能代表未来出现的结果的promise对象

#### promise和ajax的区别是什么

Promise实际是充当ajax获取后台数据后执行回调的异步执行顺序的一个媒介，客观来说就是避免ajax回调地狱（反复嵌套回调），不然代码不够直观。

#### 如何实现继承

1. 借用构造函数。也叫伪造对象或经典继承。 思路：在子类构造函数的内部调用超类型构造函数。可以通过使用 apply()和call()方法 在新创建的对象上执行构造函数。 缺点：方法都在构造函数中定义，函数的复用就无从谈起。在超类型的原型中定义的方 法，对子类而言也是不可见的，结果所有的类型都只能使用构造函数模式。

2. 组合继承。也叫伪经典继承。指的是将原型链和借用构造函数的技术组合到一起， 从而发挥二者之长。 思路：使用原型链实现对原型属性属性和方法的继承，通过借用构造函数来实现实例属 性的继承。 优点：既通过在原型上定义方法实现了函数复用，又能保证每一个实例都有它自己的数 组。 组合继承避免了原型链和借用构造函数的缺陷，融合了他们的优点，成为 JavaScript 中常用的继承模式。

3. 原型链继承。 思路：借助原型可以基于已有的对象创建对象，同时还不必因此创建自定义类型。 在object()函数内部，先创建一个临时的构造函数，然后将传入的对象作为这个构造函 数的原型，最后返回了这个临时类型的一个新实例。

4. 寄生式继承。 思路：创建一个仅用于封装继承过程的函数，该函数在内部以某种方式来增强对象，最 后再像真的是它做了所有的工作一样返回对象。缺点：使用寄生式继承来为对象添加函数，会由于不能做到函数复用二降低效率，这一 点和构造函数模式类似。

5. ）寄生组合式继承。是JavaScript 最常用的继承模式。 思路：通过借用构造函数来继承属性，通过原型链的混成形式来继承方法。 本质上，就是使用寄生式继承来继承超类型的原型，然后再将结果指定给子类型的原型。 开发人员普遍认为寄生组合式继承时引用类型最理想的继承范式。 extend（）方法才用了这样的方式。

#### 类数组和数组的区别

类数组有索引和length，可以遍历，但是不是真正的数组，不能用数组中的方法

#### 常见的类数组有哪些

1：function里的arguments

2:通过getElementByTagName(),document.childNodes(),getElementByName(),document.children()等方式获取的NodeList集合的对象都属于伪数组

3：特殊写法的对象

eg: var obj={}

obj[0]="一" obj[1]="二" obj[2]="三"

#### 怎么将类数组转换为数组

一：

假设这里有个伪数组：pagis

let arr = [].slice.call(pagis)

console.log(arr) 这时arr就是真数组了。

二：

let arr = Array.prototype.slice.call(pagis);

利用了slice传一个数组/集合，就会直接返回这个集合的原理。拿到的也是数组。

也就可以使用数组的各种方法了。

三：

var arr1 = [],

len1 = pagis.length;

for (var i = 0; i < len1; i++) {

arr1.push(pagis[i]);

}

就是简单的for循环，把类数组的每一项都push到真正的数字arr1中

与之类似的另一种写法：

（转换函数中的arguments伪数组为真数组，是在学习es6时，将扩展运算符的收集功能在经过babel转换后得到的es5代码）

for (var \_len = arguments.length, arr = new Array(\_len), \_key = 0; \_key < \_len; \_key++) {

arr[\_key] = arguments[\_key];

}

#### 什么是观察者模式？什么是发布者模式

观察者模式: 一个目标物件管理所有相依于它的观察者物件，并且在它本身的状态改变时主动发出通知。这通常透过呼叫各观察者所提供的方法来实现。此种模式通常被用来实现事件处理系统

发布-订阅者模式: 相比在观察者模式中，发布者和订阅者之间多了一个发布通道；一方面从发布者接收事件，另一方面向订阅者发布事件；订阅者需要从事件通道订阅事件,解决了主体对象与观察者之间功能的耦合, 以此避免发布者和订阅者之间产生依赖关系。

总结：在观察者模式中，观察者是知道Subject的，Subject一直保持对观察者进行记录。然而，在发布订阅模式中，发布者和订阅者不知道对方的存在。它们只有通过消息代理进行通信。

在发布订阅模式中，组件是松散耦合的，正好和观察者模式相反。

观察者模式大多数时候是同步的，比如当事件触发，Subject就会去调用观察者的方法。而发布-订阅模式大多数时候是异步的（使用消息队列）。

观察者模式需要在单个应用程序地址空间中实现，而发布-订阅更像交叉应用模式。

#### 常见的浏览器内核有哪些

    Trident内核：主要代表为IE浏览器

    Gecko内核：主要代表为Firefox

    Presto内核：主要代表为Opera

    Webkit内核：产要代表为Chrome和Safari

而这些不同内核的浏览器，CSS3属性（部分需要添加前缀的属性）对应需要添加不同的前缀，也将其称之为浏览器的私有前缀，添加上私有前缀之后的CSS3属性可以说是对应浏览器的私有属性：

Trident内核：前缀为-ms

Gecko内核：前缀为-moz

Presto内核：前缀为-o

Webkit内核：前缀为-webkit

#### 怎么创建一个新节点

var node=document.createElement("li")

var textnode=document.createTextNode("王昭君")

#### H5新增的表单控件

• ①email、②url、③number、④range

• ⑤Date pickers (date, month, week, time, datetime, datetime-local)

• ⑥search、⑦color、⑧tel

#### 怎么区分HTML和html5

　1.在文档类型声明上

　　html:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "

<html xmlns="

　　HTML5:

<!DOCTYPE html>

　　说明：在文档声明上，html代码很长，而html5只有简简单单的声明。

　　2.在结构语义上

　　html4.0：没有体现结构语义化的标签，通常都是这样来命名的,这样表示网站的头部。

<divid="header"></div>

　　html5：在语义上却有很大的优势。提供了一些新的html5标签

　　比如:

<header> 、<nav>、<article>、<aside>、<footer>..

#### 常见的空元素有哪些

常见的空元素：

<br> <hr> <img> <input> <link> <meta>

鲜为人知的：

<area> <base> <col> <command> <embed> <keygen> <param> <source> <track> <wbr>

#### 什么是三次握手和四次挥手

**三次握手**

第一次握手：建立连接时，客户端发送syn包（syn=x）到服务器，并进入SYN\_SENT状态，等待服务器确认；SYN：同步序列编号（Synchronize Sequence Numbers）。

第二次握手：服务器收到syn包，必须确认客户的SYN（ack=x+1），同时自己也发送一个SYN包（syn=y），即SYN+ACK包，此时服务器进入SYN\_RECV状态；

第三次握手：客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK(ack=y+1），此包发送完毕，客户端和服务器进入ESTABLISHED（TCP连接成功）状态，完成三次握手。

**四次挥手**

1）客户端进程发出连接释放报文，并且停止发送数据。释放数据报文首部，FIN=1，其序列号为seq=u（等于前面已经传送过来的数据的最后一个字节的序号加1），此时，客户端进入FIN-WAIT-1（终止等待1）状态。 TCP规定，FIN报文段即使不携带数据，也要消耗一个序号。

2）服务器收到连接释放报文，发出确认报文，ACK=1，ack=u+1，并且带上自己的序列号seq=v，此时，服务端就进入了CLOSE-WAIT（关闭等待）状态。TCP服务器通知高层的应用进程，客户端向服务器的方向就释放了，这时候处于半关闭状态，即客户端已经没有数据要发送了，但是服务器若发送数据，客户端依然要接受。这个状态还要持续一段时间，也就是整个CLOSE-WAIT状态持续的时间。

3）客户端收到服务器的确认请求后，此时，客户端就进入FIN-WAIT-2（终止等待2）状态，等待服务器发送连接释放报文（在这之前还需要接受服务器发送的最后的数据）。

4）服务器将最后的数据发送完毕后，就向客户端发送连接释放报文，FIN=1，ack=u+1，由于在半关闭状态，服务器很可能又发送了一些数据，假定此时的序列号为seq=w，此时，服务器就进入了LAST-ACK（最后确认）状态，等待客户端的确认。

5）客户端收到服务器的连接释放报文后，必须发出确认，ACK=1，ack=w+1，而自己的序列号是seq=u+1，此时，客户端就进入了TIME-WAIT（时间等待）状态。注意此时TCP连接还没有释放，必须经过2∗∗MSL（最长报文段寿命）的时间后，当客户端撤销相应的TCB后，才进入CLOSED状态。

6）服务器只要收到了客户端发出的确认，立即进入CLOSED状态。同样，撤销TCB后，就结束了这次的TCP连接。可以看到，服务器结束TCP连接的时间要比客户端早一些。

#### javascript和typescript的区别

**JavaScript**是一种基于客户端浏览器的，基于对象、事件驱动式的脚本语言。稍提一下，JavaScript和Java没有任何关系，两者相当于雷峰塔和雷锋的关系。

JavaScript是一种脚本编写语言，无需编译，只要嵌入HTML代码中，就能在浏览器中加载执行。

JavaScript是一种基于对象的语言，可以创建对象同时使用现有对象。但是JavaScript并不支持面向对象语言所承载的继承和重载功能。

JavaScript使用的变量是弱类型。

JavaScript语言较为安全，仅在浏览器端执行，不会访问本地硬盘数据。

JavaScript语言具有动态性。JavaScript是事件驱动的，只根据用户的操作做出相应的反应处理。

JavaScript只依赖于浏览器，与操作系统的因素无关。因此JavaScript是一种跨平台的语言。

JavaScript兼容性极好，能够与其他技术（如XML、REST API等）一起使用

**TypeScript**

TypeScript是JavaScript类型的超类，它可以编译成纯JavaScript。TypeScript可以在任何浏览器、任何计算机和任何操作系统上运行，并且是开源的。

TypeScript是Microsoft推出的开源语言，使用Apache授权协议

TypeScript增加了静态类型、类、模块、接口和类型注解

TypeScript可用于开发大型的应用

TypeScript易学易于理解

**JavaScript和TypeScript的主要差异**

TypeScript可以使用JavaScript中的所有代码和编程概念，TypeScript是为了使JavaScript的开发变得更加容易而创建的。

TypeScript从核心语言方面和类概念方面的模塑方面对JavaScript对象模型进行扩展。

JavaScript代码可以在无需任何修改的情况下与TypeScript一同工作，同时可以使用编译器将TypeScript代码转换为JavaScript。

TypeScript通过类型注解提供编译时的静态类型检查。

TypeScript中的数据要求带有明确的类型，JavaScript不要求。

TypeScript提供了缺省参数值。

TypeScript引入了JavaScript中没有的“类”概念。

TypeScript中引入模块的概念，可以把声明、数据、函数和类封装在模块中。

#### 如何检测一个变量是字符串

方法1、

function isString(obj){

return typeof(obj) === "string"? true: false;

// returntypeof obj === "string"? true: false;

}

方法2、

function isString(obj){

return obj.constructor === String? true: false;}

方法3、

function isString(obj){

return Object.prototype.toString.call(obj) === "[object String]" ? true : false;

}

如：

var isstring = isString('xiaoming');

console.log(isstring); // true

#### Slice和splice的区别

1.slice(start,end)：方法可从已有数组中返回选定的元素，返回一个新数组，包含从start到end（不包含该元素）的数组元素。

注意：该方法不会改变原数组，而是返回一个子数组，如果想删除数组中的一段元素，应该使用Array.splice()方法。

2.splice()：该方法向或者从数组中添加或者删除项目，返回被删除的项目。（该方法会改变原数组）

splice（index,howmany,item1,...itemX）

#### 常见的表单选择器有哪些

:input表单选择器

:text表单文本选择器

:password表单密码选择器

:radio单选按钮选择器

:checkbox复选框选择器

:submit提交按钮选择器

:image图像域选择器

:button表单按钮选择器

:checked选中状态选择器

:selected选中状态选择器

#### Jquery中的delegate函数是干什么的

delegate()[函数](https://www.baidu.com/s?wd=%E5%87%BD%E6%95%B0&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)用于为指定元素的一个或多个事件绑定事件处理[函数](https://www.baidu.com/s?wd=%E5%87%BD%E6%95%B0&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)。

即使是执行delegate()函数之后新添加的元素，只要它符合条件，绑定的事件处理函数仍然对其有效。  
  
此外，该函数可以为同一元素、同一事件类型绑定多个事件处理函数。触发事件时，jQuery会按照绑定的先后顺序依次执行绑定的事件处理函数。

delegate()函数并不是为当前jQuery对象匹配的元素绑定事件处理函数，而是为它们的后  
代元素中符合选择器selector参数的元素绑定事件处理函数。delegate()函数并不是直接为这些后代元素挨个绑定事件，而是"委托"给当前  
  
要删除通过delegate()绑定的事件，请使用undelegate()函数

#### Css3的盒模型

当元素设置了 box-sizing:border-box;这个样式，例如下面一小段代码：

.ele{

box-sizing:border-box;

}

width和height就不仅仅是内容的宽高，而是代表着整个盒子的宽高，这里已经包含了padding和border。

#### 降低页面加载时间有哪些方法

1.优化图片

2.图像格式的选择（GIF：提供的颜色较少，可用在一些对颜色要求不高的地方）

3.优化CSS（压缩合并css，如margin-top,margin-left...)

4.网址后加斜杠（如www.campr.com/目录，会判断这个“目录是什么文件类型，或者是目录。）

5.标明高度和宽度（如果浏览器没有找到这两个参数，它需要一边下载图片一边计算大小，如果图片很多，浏览器需要不断地调整页面。这不但影响速度，也影响浏览体验。 当浏览器知道了高度和宽度参数后，即使图片暂时无法显示，页面上也会腾出图片的空位，然后继续加载后面的内容。从而加载时间快了，浏览体验也更好了。）

6.减少http请求（合并文件，合并图片）。

#### 说一下对Prototype的理解

prototype是一个属性对象，只要创建一个新函数，就会根据特定的规则为该函数创建一个prototype属性，这个属性指向函数的原型对象。在默认情况下，所有原型对象都会自动获得一个constructor(构造函数)属性，这个属性包含一个指向prototype属性所在函数的指针。

#### 怎么解决跨域问题

Cors

JSONP 方式

HttpClient 请求转发

nginx 转发（代理）

#### Meta viewport的原理

viewportde 原理：移动端浏览器通常都会在一个比移动端屏幕更宽的虚拟窗口中渲染页面，这个虚拟窗口就是viewport; 目的是正常展示没有做移动端适配的网页，让他们完整的展示给用户；

meta viewport 标签的作用是让当前viewport的宽度等于设备的宽度，同时不允许用户进行手动缩放

#### 七点十五分小于180度的夹角

67.5°

#### 前端如何进行异常检测

前端捕获异常分为全局捕获和单点捕获。全局捕获代码集中，易于管理；单点捕获作为补充，对某些特殊情况进行捕获，但分散，不利于管理。

a、全局捕获

通过全局的接口，将捕获代码集中写在一个地方，可以利用的接口有：

* window.addEventListener(‘error’) / window.addEventListener(“unhandledrejection”) / document.addEventListener(‘click’) 等
* 框架级别的全局监听，例如aixos中使用interceptor进行拦截，vue、react都有自己的错误采集接口
* 通过对全局函数进行封装包裹，实现在在调用该函数时自动捕获异常
* 对实例方法重写（Patch），在原有功能基础上包裹一层，例如对console.error进行重写，在使用方法不变的情况下也可以异常捕获

b、单点捕获

在业务代码中对单个代码块进行包裹，或在逻辑流程中打点，实现有针对性的异常捕获：

* try…catch
* 专门写一个函数来收集异常信息，在异常发生时，调用该函数
* 专门写一个函数来包裹其他函数，得到一个新函数，该新函数运行结果和原函数一模一样，只是在发生异常时可以捕获异常

#### 如何实现两个大数相加

对两个超大的整数进行加法运算，但又想输出一般形式，使用+是无法达到的，因为一旦结果值超过Number.MAX\_SAFE\_INTEGER就会被转换成科学计数法，并且数字精度相比之前将会有误差。

此时需要自己写一套加法算法，原理就是小学学习加减法时候的算法，相应位数字相加，满十进一。

function addBigNum(a, b){

//将a,b两个数转换为数组，并且将数组倒叙。

let arrA = a.split('').reverse()

let arrB = b.split('').reverse()

//获取两个数组的长度

let aLen = arrA.length

let bLen = arrB.length

let shortArr, shortLen, longArr, longLen

let result = []

if(aLen<bLen){

shortArr = arrA

shortLen = aLen

longArr = arrB

longLen = bLen

}else {

shortArr = arrB

shortLen = bLen

longArr = arrA

longLen = aLen

}

let add = 0//用于记录进位数

//将前面的数相加

for(let i = 0; i < shortLen; i++){

let temp = (parseInt(arrA[i])+parseInt(arrB[i])+add)%10//相加后模10取余数

result.push(temp)

add = Math.floor((parseInt(arrA[i])+parseInt(arrB[i])+add)/10)//求进位数

}

if(shortLen===longLen){//如果两个数的长度相等，结果直接加上进位数即可

result.push(add)

}else{//如果两数字长度不相等

for(let j = shortLen; j < longLen; j++){

if(j===longLen-1){//最后一位直接等于长数组最后一位加上进位数字

result.push(parseInt(longArr[j])+add)

}else{

let temp = (parseInt(longArr[j])+add)%10//长数组后面的数字加上进位模10取余数

result.push(temp)

add = Math.floor((parseInt(longArr[j])+add)/10)//求进位数字

}

}

}

return parseInt(result.reverse().join(''))//将结果数组倒叙，然后转换为字符串，最后转化为整数

#### Require模块中amd和cmd的区别

AMD 即Asynchronous Module Definition，中文名是异步模块定义的意思。它是一个在浏览器端模块化开发的规范

由于不是JavaScript原生支持，使用AMD规范进行页面开发需要用到对应的库函数，也就是大名鼎鼎RequireJS，实际上AMD 是 RequireJS 在推广过程中对模块定义的规范化的产出

CMD 即Common Module Definition通用模块定义，CMD规范是国内发展出来的，就像AMD有个requireJS，CMD有个浏览器的实现SeaJS，SeaJS要解决的问题和requireJS一样，只不过在模块定义方式和模块加载（可以说运行、解析）时机上有所不同

#### 引用类型有哪几种

Function,Array,Object

#### 数据加密的方式有哪几种

MD5:全称是Message Digest Algorithm 5，译为“消息摘要算法第5版”效果：对输入信息生成唯一的128位散列值（32个字符）

Base64加密方式(可逆)

钥匙串加密方式

对称加密算法

RSA加密(非对称加密算法)

#### 什么是单线程？什么是多线程

单线程在程序执行时，所走的程序路径按照连续顺序排下来，前面的必须处理好，后面的才会执行。

多线程（英语：multithreading），是指从软件或者硬件上实现多个线程并发执行的技术。具有多线程能力的计算机因有硬件支持而能够在同一时间执行多于一个线程，进而提升整体处理性能。

#### js中的迭代方式有哪几种

js比较好用的几种迭代方法

1.every() ：对数组中的每一项运行给定函数，如果该函数对每一项都返回 true，则返回 true。

2.filter() ：对数组中的每一项运行给定函数，返回该函数会返回 true 的项组成的数组。

3.forEach() ：对数组中的每一项运行给定函数。这个方法没有返回值。

4.map() ：对数组中的每一项运行给定函数，返回每次函数调用的结果组成的数组。

5.some() ：对数组中的每一项运行给定函数，如果该函数任意一项返回 true，则返回 true。

6.reduce() //数组求和 从左向右+

7.reduceRight() //数组求和 从右向左+

#### 什么是数组扁平化

数组扁平化是指将一个多维数组变为一维数组

#### Node的中间件有哪些

- 应用级中间件

- 路由级中间件

- 错误处理中间件

- 内置中间件

- 第三方中间件

#### 如何和mysql进行通信

var mysql = require('mysql'); //调用MySQL模块

//创建一个connection

var connection = mysql.createConnection({

host : '192.168.0.200', //主机

user : 'root', //MySQL认证用户名

password : 'abcd', //MySQL认证用户密码

port: '3306', //端口号

});

//创建一个connection

connection.connect(function(err){

if(err){

console.log('[query] - :'+err);

return;

}

console.log('[connection connect] succeed!');

});

//执行SQL语句

connection.query('SELECT 1 + 1 AS solution', function(err, rows, fields) {

if (err) {

console.log('[query] - :'+err);

return;

}

console.log('The solution is: ', rows[0].solution);

});

//关闭connection

connection.end(function(err){

if(err){

return;

}

console.log('[connection end] succeed!');

});

#### Iframe 标签是用来干什么的

<iframe> 标签是一个内联框架，即用来在当前 HTML 页面中嵌入另一个文档的，且所有主流浏览器都支持iframe标签。

#### W3C标准是什么

W3C的意思是万维网联盟（World Wide Web Consortium），创建于1994年10月，是一个会员组织，它的工作是对web进行标准化--->W3C 致力于实现所有的用户都能够对 web 加以利用，W3C 同时与其他标准化组织协同工作

     Web标准不是指一个标准，而是一系列的标准。网页主要有三部分组成： 结构，行为，表现，对应的标准也分三方面：结构化标准语言主要包括XHTML和XML，表现标准语言主要包括CSS，行为标准主要包括对象模型（如W3C DOM）、ECMAScript等。

#### HTML全局属性有哪些

类：为元素设置类标识，多个类名用空格分开，CSS和JavaScript的可通过类属性获取元素

contenteditable：指定元素内容是否可编辑

contextmenu：自定义鼠标右键弹出菜单内容

data- \*：为元素增加自定义属性

dir：设置元素文本方向

draggable：设置元素是否可拖拽

dropzone：设置元素拖放类型：复制，移动，链接

hidden：表示一个元素是否与文档。样式上会导致元素不显示，但是不能用这个属性实现样式效果

id：元素id，文档内唯一

lang：元素内容的的语言

spellcheck：是否启动拼写和语法检查

style：行内css样式

tabindex：设置元素可以获得焦点，通过tab可以导航

title：元素相关的建议信息

translate：元素和子孙节点内容是否需要本地化

#### HTML5新增的画布canvas和svg的区别是什么

一句话总结：都是2D做图，svg是矢量图，canvas是位图。Canvas 是逐像素进行渲染的，适合游戏。

SVG

　　　　SVG 是一种使用 XML 描述 2D 图形的语言。

　　　　SVG 基于 XML，这意味着 SVG DOM 中的每个元素都是可用的。您可以为某个元素附加 JavaScript 事件处理器。

　　　　在 SVG 中，每个被绘制的图形均被视为对象。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。

　　　　特点：

　　　　 　　不依赖分辨率

　　　　 　　支持事件处理器

　　　　 　　最适合带有大型渲染区域的应用程序（比如谷歌地图）

　　　　　　 复杂度高会减慢渲染速度（任何过度使用 DOM 的应用都不快）

　　　　 　　不适合游戏应用

Canvas

　　　　Canvas 通过 JavaScript 来绘制 2D 图形。

　　　　Canvas 是逐像素进行渲染的。

　　　　在 canvas 中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括任何或许已被图形覆盖的对象。

　　　　特点：

　　　 　　依赖分辨率

　　　　 　　不支持事件处理器

　　　　　　 弱的文本渲染能力

　　　　　　 能够以 .png 或 .jpg 格式保存结果图像

　　　　 　　最适合图像密集型的游戏，其中的许多对象会被频繁重绘

#### 为什么要初始化css样式

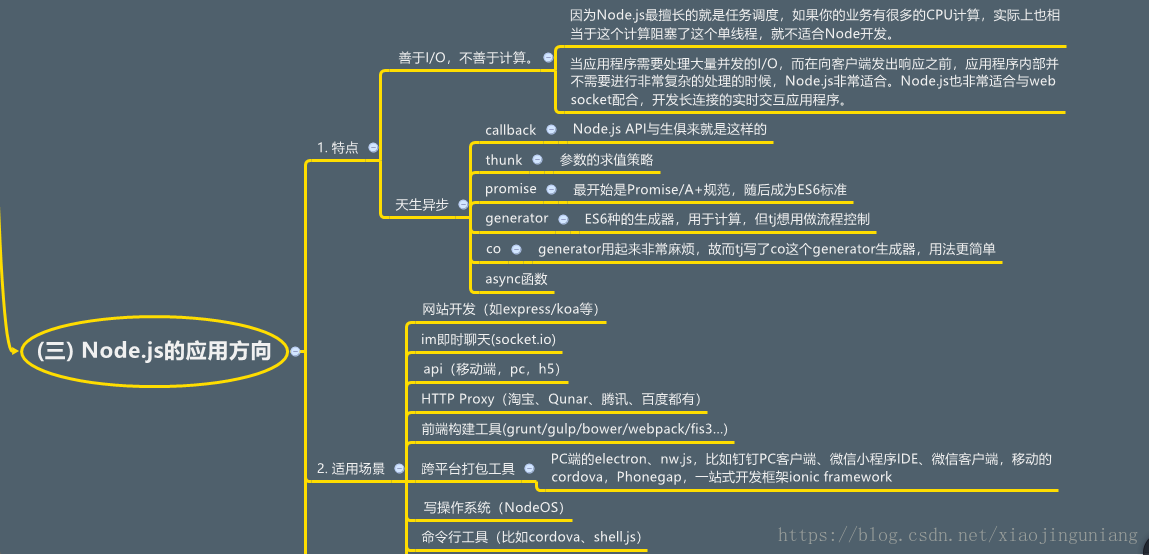
1.浏览器差异 不同浏览器对有些标签的默认值是不同的,如果没对css初始化会出现浏览器之间的页面显示差异

2.提高编码质量 如果不初始化,整个页面做完会很糟糕,重复的css样式很多 最简单的初始化方法是:(不建议) 淘宝样式 样式初始化

#### box-sizing有哪些属性

box-sizing: content-box|border-box|inherit;

#### Node的应用场景有哪些



#### 改变数组长度的方法有哪些

arr.push()

arr.pop()

arr.unshift()

arr.shift()

#### js清除浏览器缓存的方法

方式一：用ajax请求服务器最新文件，并加上请求头If-Modified-Since和Cache-Control,如下:

$.ajax({

url:'www.haorooms.com',

dataType:'json',

data:{},

beforeSend :function(xmlHttp){

xmlHttp.setRequestHeader("If-Modified-Since","0");

xmlHttp.setRequestHeader("Cache-Control","no-cache");

},

success:function(response){

//操作

}

async:false

});

方法二，直接用cache:false,

$.ajax({

url:'www.haorooms.com',

dataType:'json',

data:{},

cache:false,

ifModified :true ,

success:function(response){

//操作

}

async:false

});

方法三：用随机数

在 URL 参数后加上 "?timestamp=" + new Date().getTime();

#### babel是干什么的

[Babel](https://babeljs.io/" \t "https://www.cnblogs.com/superAnny/p/_blank)是一个广泛使用的转码器，可以将ES6代码转为ES5代码，从而在现有环境执行。

babel就是为了支持原有的旧的环境。

#### 图片压缩的原理是什么

减少图像数据中的冗余信息从而用更加高效的格式存储和传输数据。

#### express中间件如何实现的

中间件本质就是一个函数，它被当作 app.use(中间件函数) 的参数来使用，或者是某个路由处理函数中使用。a.从上往下依次匹配请求路径，如果匹配成功则执行该中间件 ​ b.如果这个中间件中调用了：next() ，则会继续往下匹配 ​ c.如果所有的中间件都无法匹配，则会自动进入一个兜底的中间件响应返回404 not found错误

#### 怎么实现一个三角形

#css3-triangle{

width: 0;

height: 0;

border-left: 100px solid transparent;

border-right: 100px solid transparent;

border-bottom: 150px solid #232323;}

#### js如何实现倒计时

左侧输入为大于0的整数，单位为毫秒，右侧显示倒计时的时间，间隔是1000ms,注意小于1000ms的时间的处理。中间是开始的按钮可以重复点击，重复点击后则从新计算

<input type="text" id="leftTime" placeholder="请输入0以上的整数">

<button id="btn">开始</button>

<input type="text" id="rightTime" placeholder="开始倒计时">

<script>

var btn = document.getElementById("btn");

var leftTime = document.getElementById("leftTime");

var rightTime = document.getElementById("rightTime");

btn.onclick=function(){

if(leftTime.value > 0){

rightTime.value=leftTime.value

}else{

rightTime.value = 1

}

var cle = setInterval(function(){

rightTime.value--;

if(rightTime.value == 0){

window.clearInterval(cle)

}

},1000)

}

</script>

#### 网络协议分为哪几层

7 应用层

6 表示层

5 会话层

4 传输层

3 网络层

2 数据链路层

1 物理层

#### 动态获取数组的长度

arr.length

#### 移动端对图片的优化有哪几种方式

懒加载，使用CSS Sprites合并为一张大图，首先从图片格式方面着手，webp(google官方网址)是谷歌推出的一种图片格式，优点在于同等画面质量下，体积比jpg、png少了25%以上，去掉无意义的修饰，使用矢量图替代位图。

按照HTTP协议设置合理的缓存。

#### 程序中捕获异常的方式

window.error

try{}catch(error){}finally{}

#### Ajax如何处理json数据

使用eval() 或者JSON.parse() 鉴于安全性考虑，推荐使用JSON.parse()更靠谱，对数据的安全性更好。

#### Git和svn的区别？

1.GIT是分布式的，而SVN是集中式的

2.GIT把内容按元数据方式存储，而SVN是按文件：因为git目录是处于个人机器上的一个克隆版的版本库，它拥有中心版本库上所有的东西，例如标签，分支，版本记录等。​

3.GIT分支和SVN的分支不同：svn会发生分支遗漏的情况，而git可以同一个工作目录下快速的在几个分支间切换，很容易发现未被合并的分支，简单而快捷的合并这些文件。

4.GIT没有一个全局的版本号，而SVN有

5.GIT的内容完整性要优于SVN：GIT的内容存储使用的是SHA-1哈希算法。这能确保代码内容的完整性，确保在遇到磁盘故障和网络问题时降低对版本库的破坏。



#### 什么是git？

Git是一个开源的分布式版本控制系统，可以有效、高速地处理从很小到非常大的项目版本管理。当这个系统的任何一个客户端出现问题的时候，都可以从另外的客户端（即使服务器挂了）获取所有的代码。

#### 在git中你如何还原已经push并公开的提交？

删除或修复新提交中的错误文件，并将其推送到远程存储库。这是修复错误的最自然方式。对文件进行必要的修改后，将其提交到我将使用的远程存储库

git commit -m "commit message"

创建一个新的提交，撤消在错误提交中所做的所有更改。可以使用命令：

git revert <name of bad commit>

#### 怎么样将N次提交压缩成一次提交？

将N个提交压缩到单个提交中有两种方式：

如果要从头开始编写新的提交消息，请使用以下命令：

git reset –soft HEAD~N &&

git commit

如果你想在新的提交消息中串联现有的提交消息，那么需要提取这些消息并将它们传给 git commit，可以这样：

git reset –soft HEAD~N &&

git commit –edit -m"$(git log –format=%B –reverse .HEAD@{N})"

#### Git提交代码的步骤？

- 通过git add 将文件 工作区 ---》暂存区 (本地)  
- 通过git commit 将文件 暂存区 ---》分支 (本地)  
- 通过git push 将文件 分支 ---》远程库 (github)

#### 提交时发生冲突，你能解释冲突是如何产生的吗？你是如何解决的？

场景如下，甲乙都是根据point.java 文件进行了开发。甲开发出了版本2，并且提交了代码；乙开发出了版本3，也需要提交了代码，此时将会报错存在冲突。

为什么呢？因为甲开发完了版本，提交了版本之后，此时远端的代码已经是版本2点代码了，而乙是基于版本1进行的开发出了版本3。所以，乙想要提交代码，势必要将自己的代码更新为版本2的代码，然后再进行提交，如果存在冲突则解决冲突后提交。

解决冲突通常使用如下的步骤即可:

情况1 无冲突

先拉取远端的代码，更新本地代码。然后提交自己的更新代码即可。

情况2 有冲突

拉取远端代码。存在冲突，会报错。

此时我们需要将本地代码暂存起来 stash；

更新本地代码，将本地代码版本更新和远端的代码一致即可；

将暂存的代码合并到更新后的代码后，有冲突解决冲突(需要手动进行解决冲突)；

提交解决冲突后的代码。

#### 如果本次提交误操作，如何撤销？

情况一：文件被修改了，但未执行git add操作（working tree内撤销）

git checkout fileName

git checkout .

情况二：同时对多个文件执行了git add操作，但本次只想提交其中一部分文件

$ git add \*

$ git status

# 取消暂存

$ git reset HEAD <filename>

情况三：文件执行了git add操作，但想撤销对其的修改（index内回滚）

# 取消暂存

git reset HEAD fileName

# 撤销修改

git checkout fileName

情况四：修改的文件已被git commit，但想再次修改不再产生新的Commit

# 修改最后一次提交

$ git add sample.txt

$ git commit --amend -m"说明"

情况五：已在本地进行了多次git commit操作，现在想撤销到其中某次Commit

git reset [--hard|soft|mixed|merge|keep] [commit|HEAD]

#### 如果我想修改提交的历史信息，应该用什么命令？

1、git log --oneline -5

    查看最近5次commit的简要信息，输出信息为：简短commitID commit\_message，可以根据需要查看最近n次的提交

    也可以git log -5，输出信息相对详细些，commitID为完整的，这里只需要加上参数--oneline查看简短commitID即可

2、git rebase -i <简短commitID>

    如果需要修改从上往下第2个commit\_message，这里的简短commitID为上面输出信息的第3个，以此类推

    在弹出的窗口中，以VIM编辑方式显示了最近两次的提交信息

3、（按照VIM操作）按i键，进入编辑模式，将想要修改的提交前的pick改为reword，如果需要修改多个，也可以将对应的多个pick改为reword

4、（按照VIM操作）按ESC键        再按 shift + :        然后输入wq（w是保存，q是退出）        按回车键

5、在弹出的窗口中，按i进入编辑模式，就可以修改commit\_message了

6、（按照VIM操作）按ESC键        再按 shift + :        然后输入wq（w是保存，q是退出）        按回车键（同第4步）

    如果第3步中修改了多个pick为reword，则会多次弹出修改界面，重复第5~6步即可

7、再使用第1步的命令查看一下修改结果，git log --oneline -5或者git log -5，查看修改是否已经完成

8、最后强制push上去git push --force

#### 如何把本地仓库的内容推向一个空的远程仓库？

1.登录git账户，新建一个代码仓库

2.进入自己要推送的项目目录下初始化

git init

3.将信息提交到本地仓库

git add .

git commit -m 'first' -n

4.将本地和远程厂库关联起来

git remote add origin git@github.com:地址

5.将本地代码推送到库上

git push -u origin master