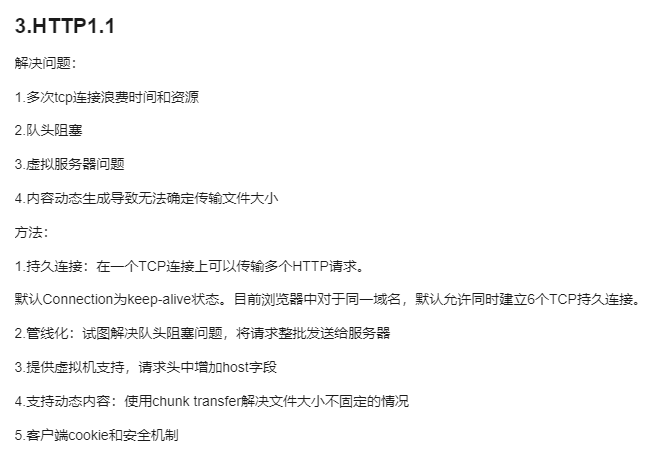
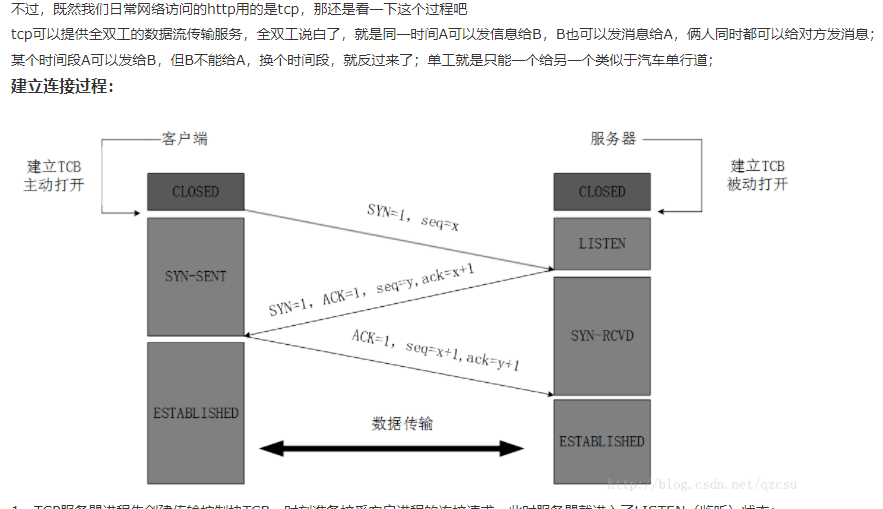
1. http 状态码
2. Http1.1比1.0做了哪些优化

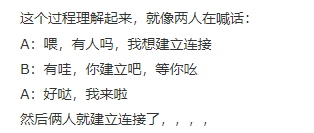
https://www.cnblogs.com/cff2121/p/13369785.html

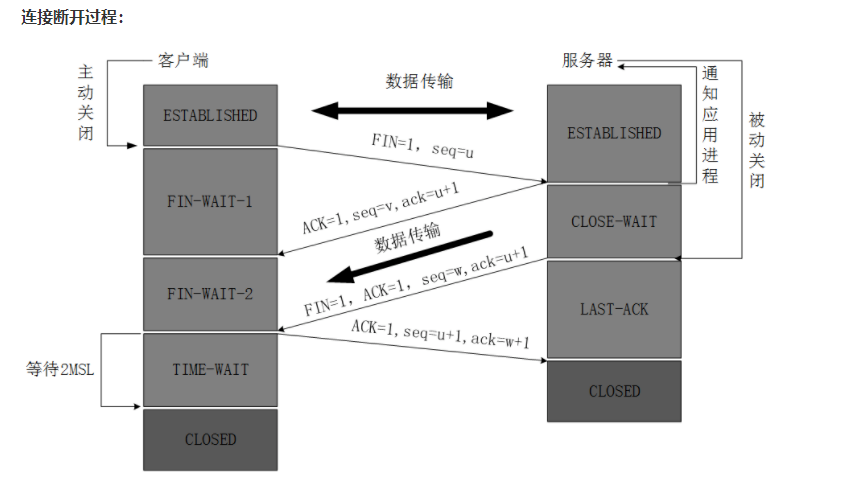


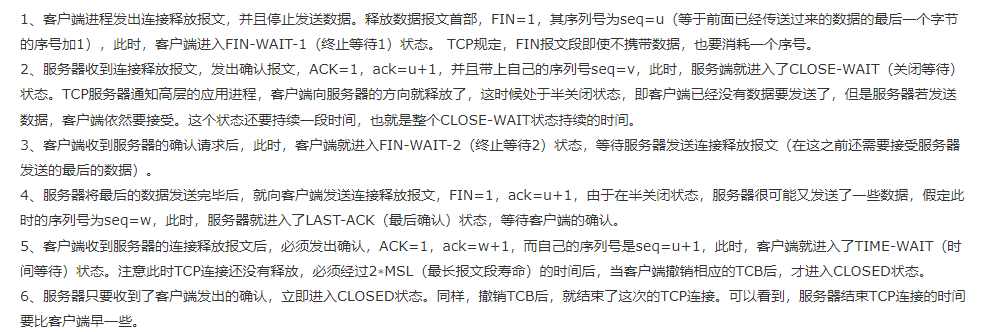
1. http 三次握手、四次挥手

<https://www.cnblogs.com/nevermorewang/p/8906833.html>









1. http缓存设置

<https://blog.csdn.net/u012375924/article/details/82806617>

强制缓存

a/优先级从高到低是 Pragma -> Cache-Control -> Expires

B/Cache-Control  no-store优先级最高

# 缓存校验(协商缓存):

A/Last-Modified

服务端在返回资源时，会将该资源的最后更改时间通过Last-Modified字段返回给客户端。客户端下次请求时通过If-Modified-Since或者If-Unmodified-Since带上Last-Modified，服务端检查该时间是否与服务器的最后修改时间一致：如果一致，则返回304状态码，不返回资源；如果不一致则返回200和修改后的资源，并带上新的时间。

B/etag

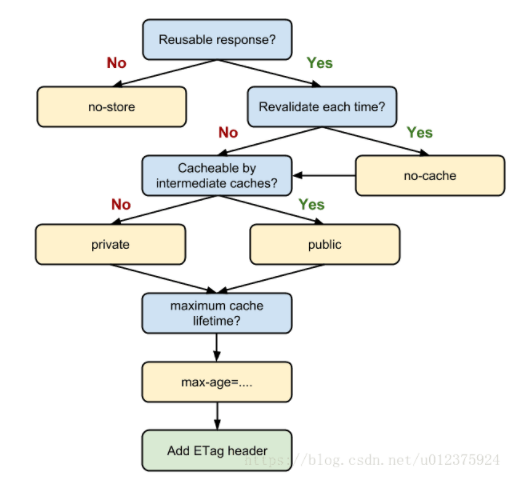
单纯的以修改时间来判断还是有缺陷，比如文件的最后修改时间变了，但内容没变。对于这样的情况，我们可以使用etag来处理。

etag的方式是这样：服务器通过某个算法对资源进行计算，取得一串值(类似于文件的md5值)，之后将该值通过etag返回给客户端，客户端下次请求时通过If-None-Match或If-Match带上该值，服务器对该值进行对比校验：如果一致则不要返回资源。

If-None-Match和If-Match的区别是：

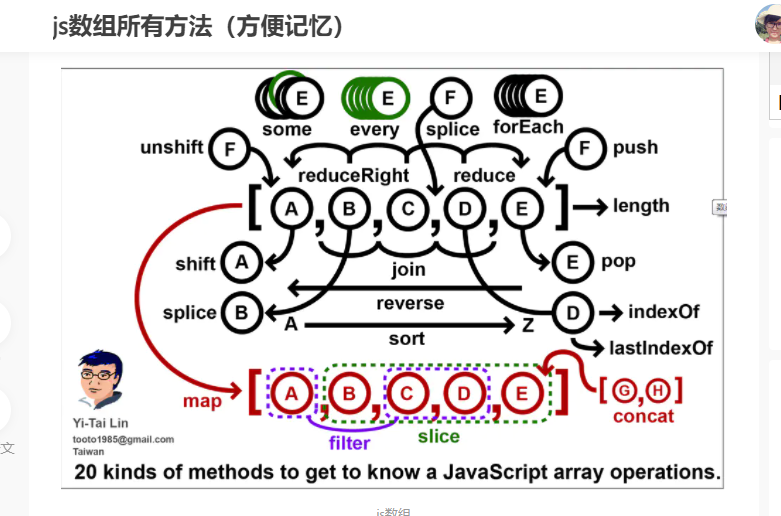
If-None-Match：告诉服务器如果一致，返回状态码304，不一致则返回资源

If-Match：告诉服务器如果不一致，返回状态码412



1. 数组遍历方法

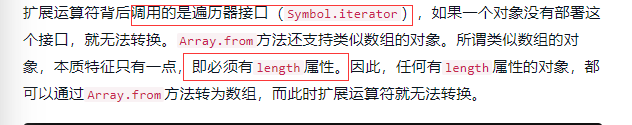
<https://www.jianshu.com/p/8f8ca38ab485>



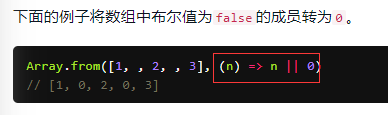
1. 体现自己优点、特质的事情
2. Es6数组的扩展

<https://es6.ruanyifeng.com/#docs/array>

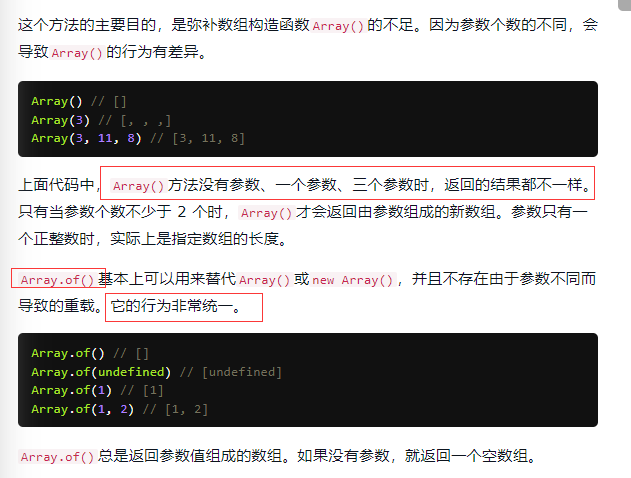




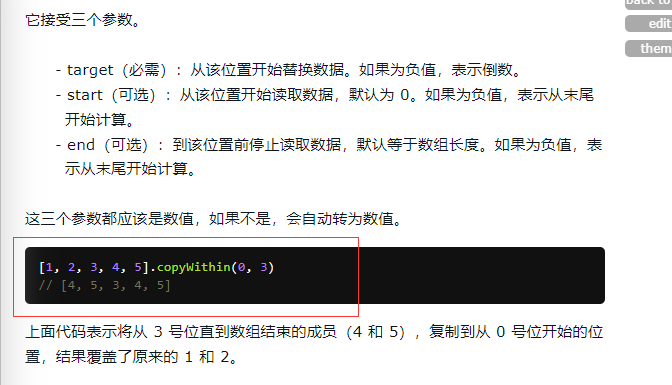








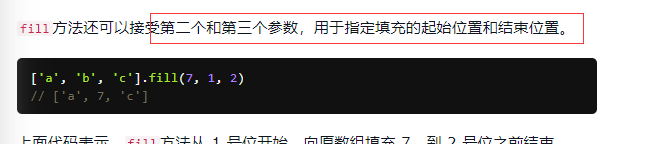
数组实例的copyWithin()方法，在当前数组内部，将指定位置的成员复制到其他位置（会覆盖原有成员），然后返回当前数组。也就是说，使用这个方法，会修改当前数组。

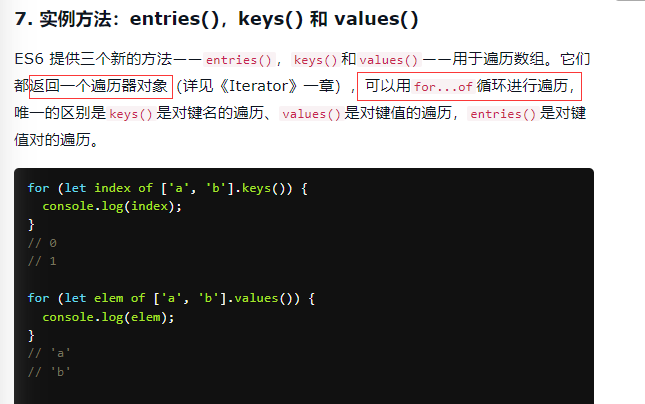


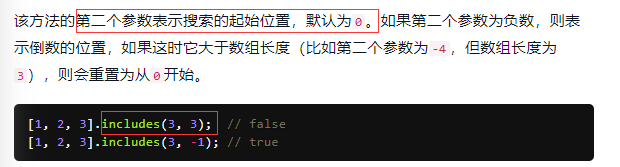
// 将3号位复制到0号位

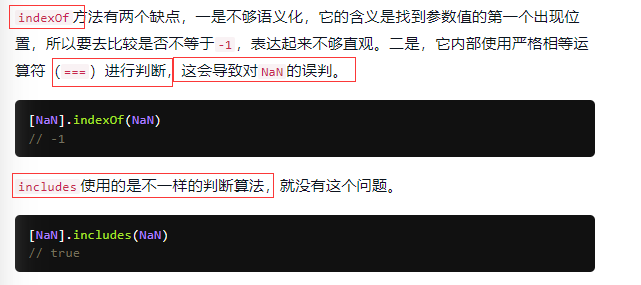
[].copyWithin.call({length: 5, 3: 1}, 0, 3)

// {0: 1, 3: 1, length: 5}

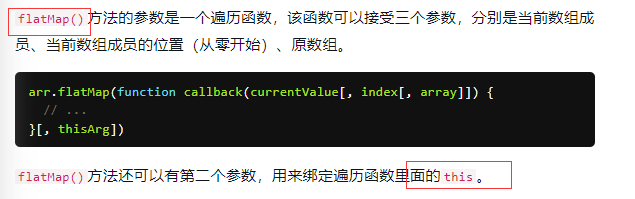
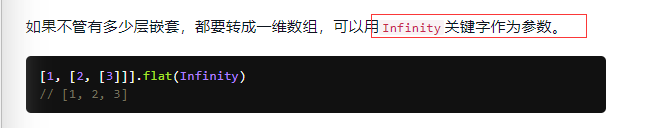


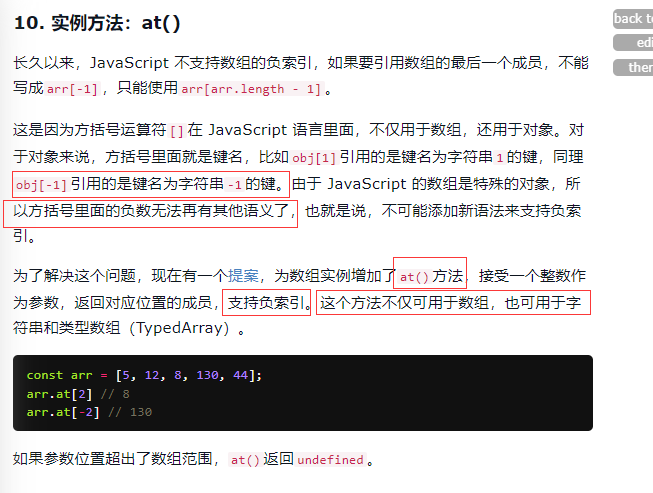












1. new操作符干了什么事情

https://www.jianshu.com/p/55fa58c1a75c

new操作符通过执行自定义构造函数或者js内置构造函数，从而生成一个实例对象。

[mdn](https://links.jianshu.com/go?to=https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/new" \t "_blank)上把内部操作大概分为4步：

1. 创建一个空的简单JavaScript对象（即{ } ）；
2. 链接该对象（即设置该对象的构造函数）到另一个对象 ；
3. 将步骤1新创建的对象作为this的上下文 ；
4. 如果该函数没有返回对象，则返回this。

伪代码

function myNew(ctr) {

let obj = Object.create(ctr.prototype);

const args = [].slice.call(arguments, 1);

let result = ctr.apply(obj, args);

var isObj = (typeof result === 'object' && result !== null);

var isFn = typeof result === 'function';

return (isObj || isFn) ? result : obj;

}

1. 数组去重的方法

https://segmentfault.com/a/1190000016418021?utm\_source=tag-newest

1. 利用ES6 Set去重（ES6中最常用），无法去掉“{}”空对象

function unique (arr) {

return Array.from(new Set(arr))

}

二、利用for嵌套for，然后splice去重（ES5中最常用）

function unique(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;

}

三、利用indexOf去重

function unique(arr) {

if (!Array.isArray(arr)) {

console.log('type error!')

return

}

var array = [];

for (var i = 0; i < arr.length; i++) {

if (array .indexOf(arr[i]) === -1) {

array .push(arr[i])

}

}

return array;

}

四、利用sort()

function unique(arr) {

if (!Array.isArray(arr)) {

console.log('type error!')

return;

}

arr = arr.sort()

var arrry= [arr[0]];

for (var i = 1; i < arr.length; i++) {

if (arr[i] !== arr[i-1]) {

arrry.push(arr[i]);

}

}

return arrry;

}

利用sort()排序方法，然后根据排序后的结果进行遍历及相邻元素比对。

五、利用对象的属性不能相同的特点进行去重（这种数组去重的方法有问题，不建议用，有待改进）

function unique(arr) {

if (!Array.isArray(arr)) {

console.log('type error!')

return

}

var arrry= [];

var obj = {};

for (var i = 0; i < arr.length; i++) {

if (!obj[arr[i]]) {

arrry.push(arr[i])

obj[arr[i]] = 1

} else {

obj[arr[i]]++

}

}

return arrry;

}

var arr = [1,1,'true','true',true,true,15,15,false,false, undefined,undefined, null,null, NaN, NaN,'NaN', 0, 0, 'a', 'a',{},{}];

console.log(unique(arr))

//[1, "true", 15, false, undefined, null, NaN, 0, "a", {…}] //两个true直接去掉了，NaN和{}去重

六、利用includes

function unique(arr) {

if (!Array.isArray(arr)) {

console.log('type error!')

return

}

var array =[];

for(var i = 0; i < arr.length; i++) {

if( !array.includes( arr[i]) ) {//includes 检测数组是否有某个值

array.push(arr[i]);

}

}

return array

}

1. 利用hasOwnProperty

function unique(arr) {

var obj = {};

return arr.filter(function(item, index, arr){

return obj.hasOwnProperty(typeof item + item) ? false : (obj[typeof item + item] = true)

})

}

var obj1 = {} var obj2 = {val: 2}

typeof obj1 + obj1 === 'object[object Object]'

typeof obj2 + obj2 === 'object[object Object]'

八、利用filter

function unique(arr) {

return arr.filter(function(item, index, arr) {

//当前元素，在原始数组中的第一个索引==当前索引值，否则返回当前元素

return arr.indexOf(item, 0) === index;

});

}

九、利用递归去重 --- 略

十、利用Map数据结构去重

function arrayNonRepeatfy(arr) {

let map = new Map();

let array = new Array(); // 数组用于返回结果

for (let i = 0; i < arr.length; i++) {

if(map .has(arr[i])) { // 如果有该key值

map .set(arr[i], true);

} else {

map .set(arr[i], false); // 如果没有该key值

array .push(arr[i]);

}

}

return array ;

}

十一、利用reduce+includes

function unique(arr){

return arr.reduce((prev,cur) => prev.includes(cur) ? prev : [...prev,cur],[]);

}

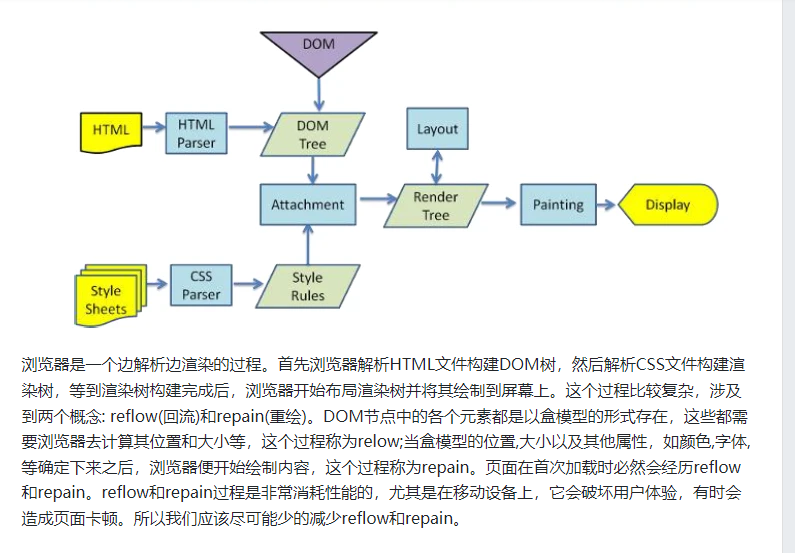
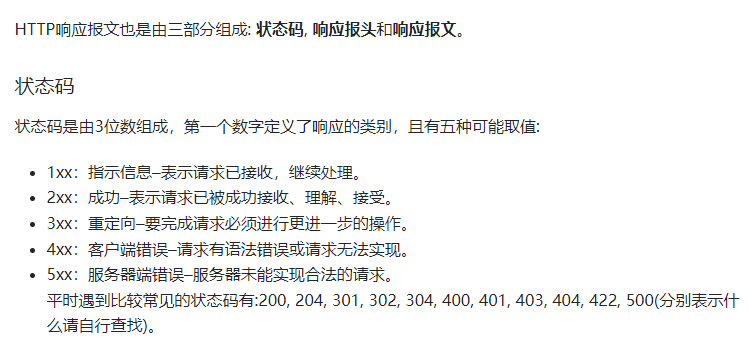
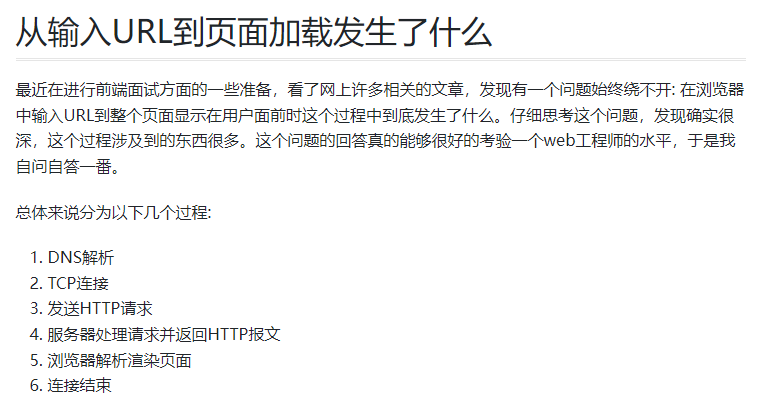
9、src和href的区别

https://blog.csdn.net/binlety/article/details/81448195



1. 浏览器输入url，到页面渲染，做什么事

<https://segmentfault.com/a/1190000006879700>



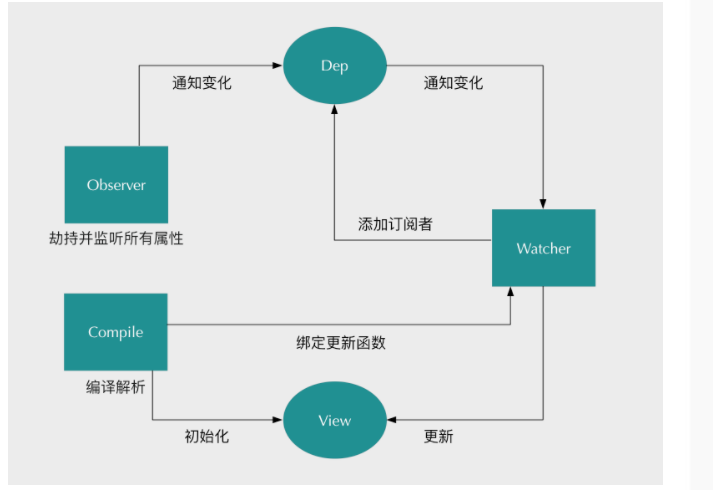
1. h5新特性

<https://www.cnblogs.com/shangjun6/p/10867941.html>



1. vue双向数据绑定原理

<https://www.cnblogs.com/canfoo/p/6891868.html>



双向包括视图和数据，a、视图变化驱动数据变化；b、数据变化驱动视图变化；

1. 通过事件监听的方式绑定回调，视图变化的时候触发回调函数，进而修改响应数据；
2. 数据变化驱动视图变化，实现原理包括数据劫持和订阅者模式。

主要包括三个东西：

监听器Observer，利用Object.defineProperty进行数据劫持，通知变化；

编译器Compiler，负责初始化数据，对模板进行编译、解析，比如把模板里的c替换成实际的值123,

并且把初始化的逻辑作添加到订阅者里，等待触发；

c、订阅者Watcher,添加到Observer里面去，当数据变化时，会被调用，从而触发关于编译器的更新函数；

1. Linux常用命令

<https://www.cnblogs.com/xuxinstyle/p/9609551.html>

A、find 基本语法参数如下find [PATH] [option] [action]

find /root -mtime 0 # 在当前目录下查找今天之内有改动的文件

B、 ls 命令，展示文件夹内内容，参数如下：

-a ：全部的档案，连同隐藏档( 开头为 . 的档案) 一起列出来～

-A ：全部的档案，连同隐藏档，但不包括 . 与 .. 这两个目录，一起列出来～

-d ：仅列出目录本身，而不是列出目录内的档案数据

-f ：直接列出结果，而不进行排序 (ls 预设会以档名排序！)

-F ：根据档案、目录等信息，给予附加数据结构，例如：

\*：代表可执行档； /：代表目录； =：代表 socket 档案； |：代表 FIFO 档案；

-h ：将档案容量以人类较易读的方式(例如 GB, KB 等等)列出来；

-i ：列出 inode 位置，而非列出档案属性；

-l ：长数据串行出，包含档案的属性等等数据；

-n ：列出 UID 与 GID 而非使用者与群组的名称 (UID与GID会在账号管理提到！)

-r ：将排序结果反向输出，例如：原本档名由小到大，反向则为由大到小；

-R ：连同子目录内容一起列出来；

-S ：以档案容量大小排序！

-t ：依时间排序

C、cd 命令

cd ./path # 切换到当前目录下的path目录中，“.”表示当前目录

cd ../path # 切换到上层目录中的path目录中，“..”表示上一层目录

1. cp 命令

-r ：递归持续复制，用于目录的复制行为

cp -r c f 把c的文件夹子，复制一份，叫f

1. rm命令作用为删除，参数：

-f ：就是force的意思，忽略不存在的文件，不会出现警告消息

-i ：互动模式，在删除前会询问用户是否操作

-r ：递归删除，最常用于目录删除，它是一个非常危险的参数

1. pwd命令，作用为查看”当前工作目录“的完整路径
2. mkdir命令创建目录：

-m, --mode=模式，设定权限<模式> (类似 chmod)，而不是 rwxrwxrwx 减 umask

-p, --parents 可以是一个路径名称。此时若路径中的某些目录尚不存在,加上此选项后,系统将自动建立好那些尚不存在的目录,即一次可以建立多个目录;

-v, --verbose 每次创建新目录都显示信息

1. rmdir 命令删除目录：

rmdir [选项]... 目录...

-p 递归删除目录dirname，当子目录删除后其父目录为空时，也一同被删除。如果整个路径被删除或者由于某种原因保留部分路径，则系统在标准输出上显示相应的信息。

-v --verbose 显示指令执行过程

1. Typescript

<https://www.runoob.com/typescript/ts-tutorial.html>

1. 自我介绍

名字，籍贯，毕业时间，项目经历

面试官好，我叫xxx, 来自广东，今年xx岁，20xx年毕业于xx大学，毕业到现在，做的都是前端开发的工作，有移动端的还有pc端的，

第一家公司是惠州的一家互联网公司，在里面，维护一个用原生js写的pc端项目，以及两个用vue写的移动端项目；

19年来深圳找了一家武汉的公司，维护两个使用vue全家桶的pc端项目，

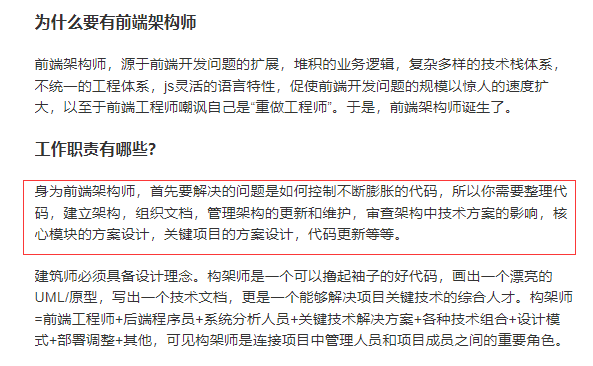
最近的一家公司，做两个项目，一个是用react + mobx + ts的移动端项目，最近的一个是用vue全家桶的pc端项目；

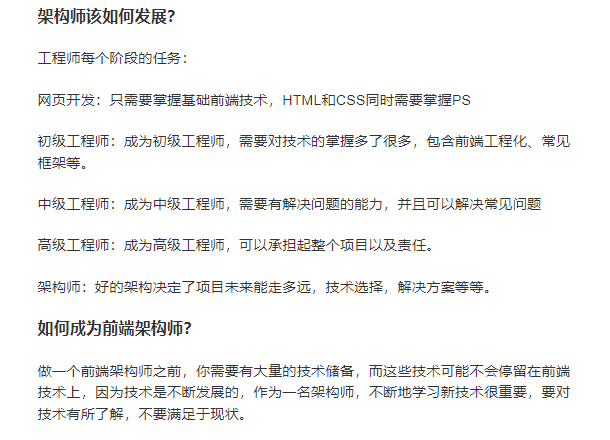
1. 上家公司离职原因

最主要的原因是我想找一个更有发展前景的平台，另外的原因是，入职之前hr说的平时加班费并不存在，而且在甲方场地办公，甲方的需求太随意，连评审都不用，做完又是三番五次改，挺心烦的，所以就离职了。

1. 浏览器兼容
2. 最近三年的奋斗目标

<https://baijiahao.baidu.com/s?id=1688463473932807216&wfr=spider&for=pc>

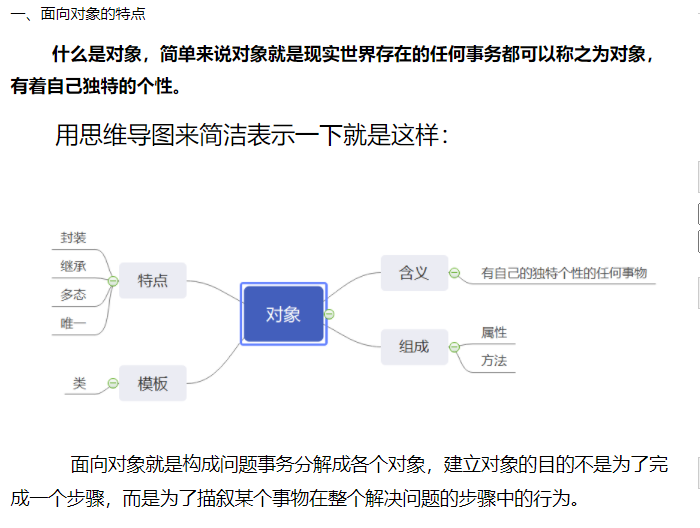


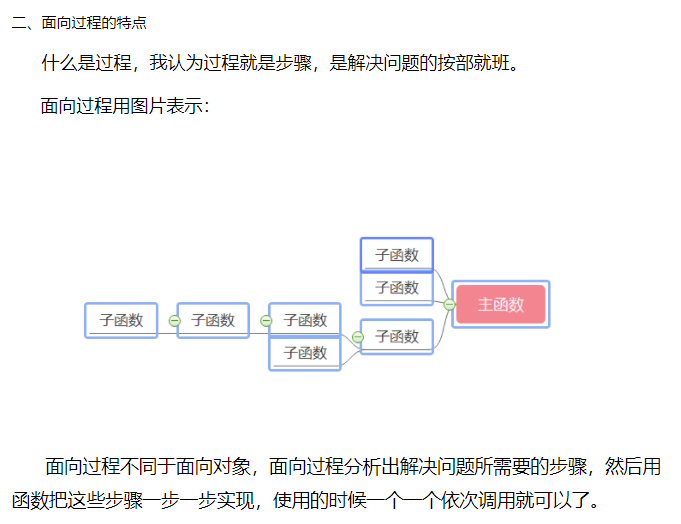


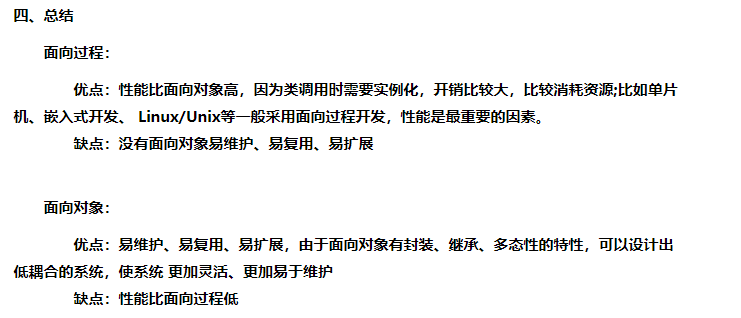
工作中常做的事情：

1. 组件化：能组件化的要组件化，input输入框要输入不同位数的小数；
2. 公用方法：防抖节流；
3. 接口要求：跟后台对接的接口返回值要求统一，同样的功能接口格式尽量一样，前端代码处理逻辑才能相同，比如一样的删除接口，不同的后台用的方法不相同，用post、put，参数提交也不同，用query、body，要求统一，不然前端要搞好几份不同的代码。
4. 代码文件夹：代码按照同功能区分，两种工序放到一个文件夹下，我件里两个文件夹单独存放，就不需要去区分辨别了。
5. 面向对象与面向过程的区别

<https://www.cnblogs.com/pylearner/p/10903266.html>

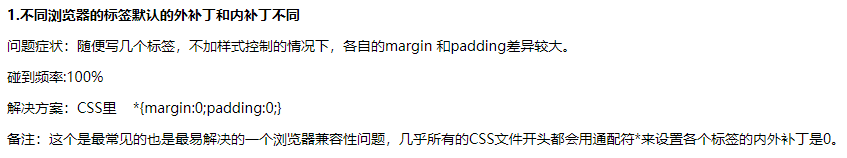


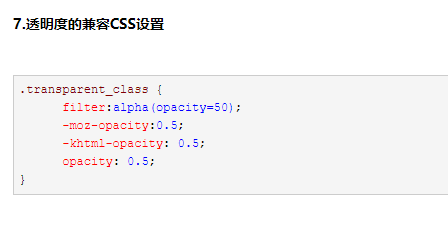




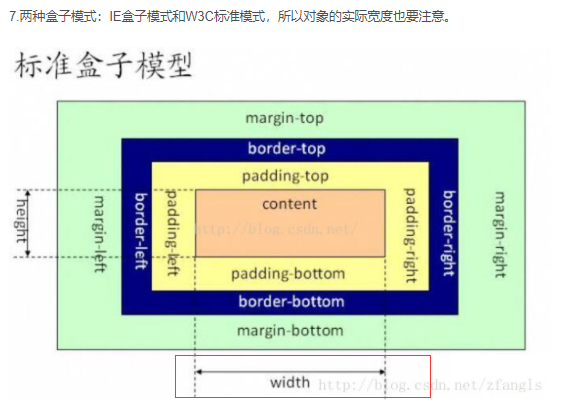
1. 常见的浏览器兼容性问题

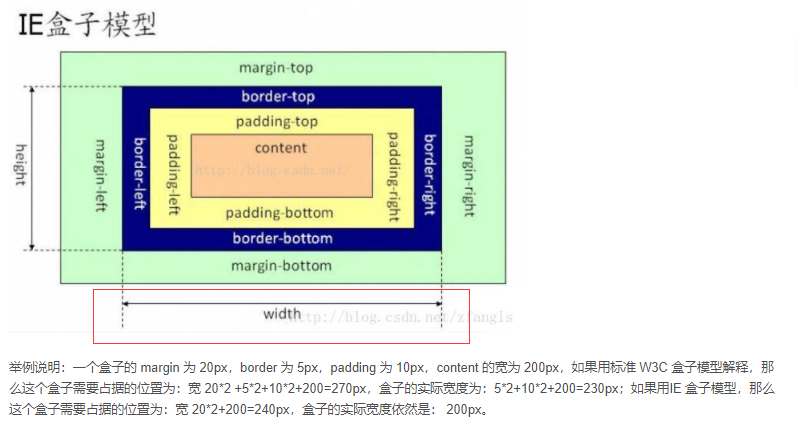
<https://www.cnblogs.com/angel648/p/11392262.html>



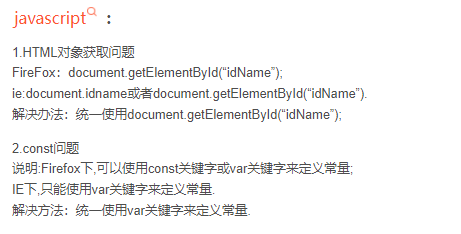


<https://blog.csdn.net/zfangls/article/details/53908063>







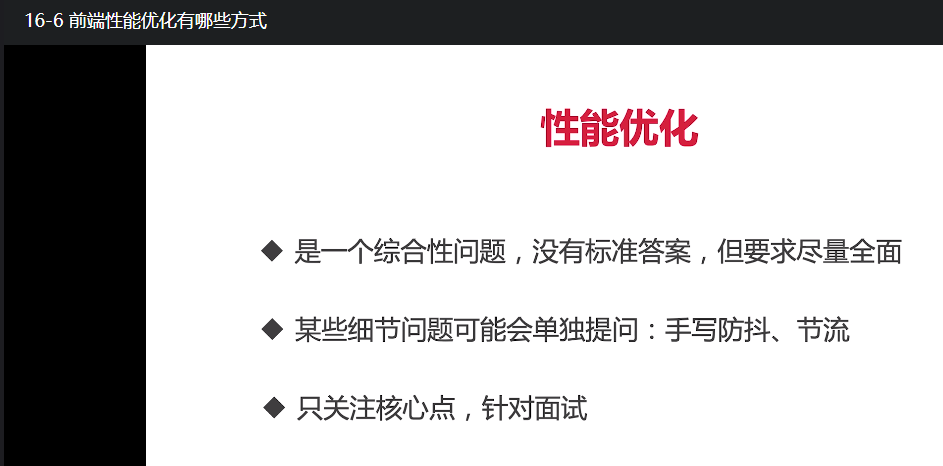


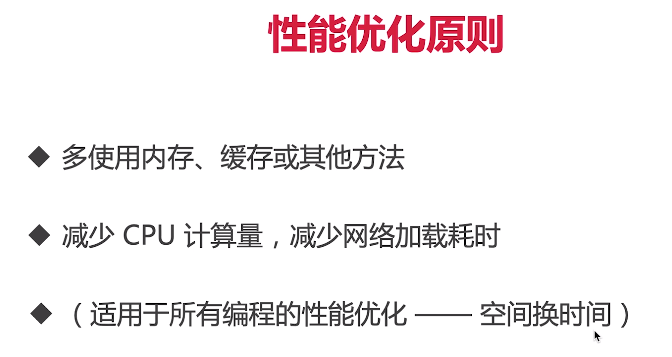






1. 前端性能优化







让加载更快：

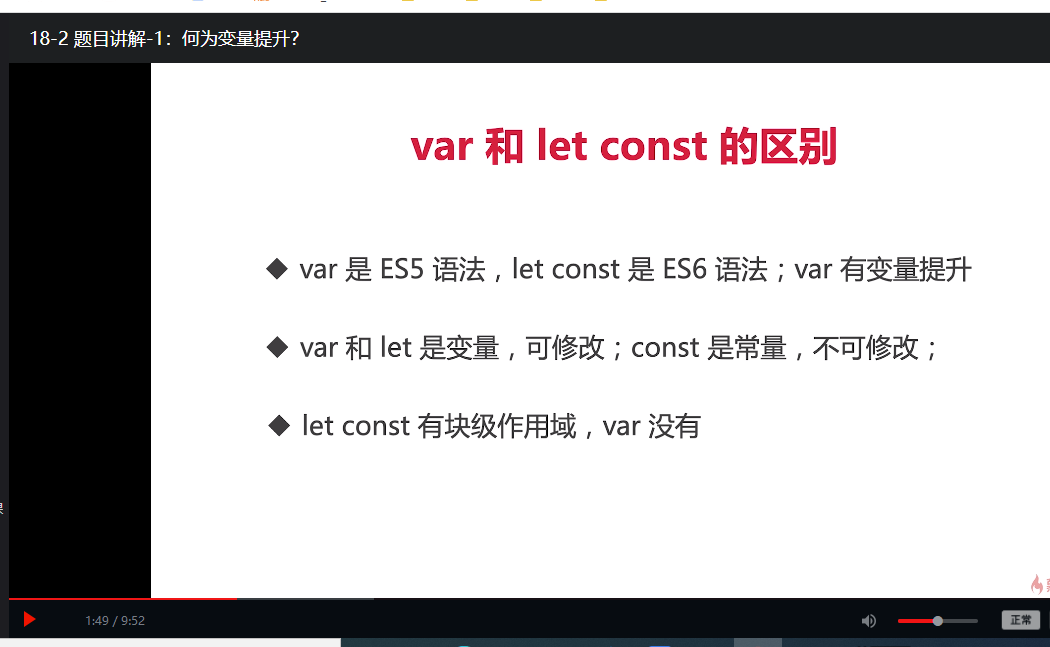
1. 减少资源体积：压缩代码，如gzip, 能压缩到1/3左右
2. 减少访问次数：合并代码（比如三个js文件webpack打包到一个js文件，雪碧图，css合并，），ssr服务端渲染(后台一次性把所需要的html, js，css一次性返回，可以不需要在解析html的时候再去请求js等文件)，缓存
3. 使用更快的网络：CDN（根据区域选择近的服务器）

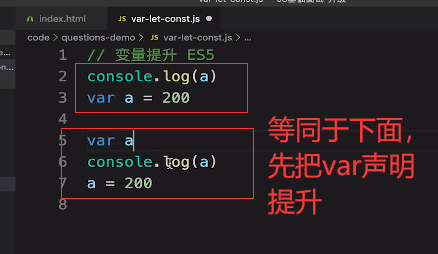
让渲染更快：

1. css放在head, js放到body最下面
2. 尽早开始执行js,用DOMContentLoaded触发
3. 懒加载，图片懒加载，上滑加载更多；
4. 对DOM查询进行缓存
5. 频繁DOM操作，合并到一起插入DOM结构
6. 节流throttle, 防抖 debounce, 让渲染更加流畅
7. 解决跨域的方法

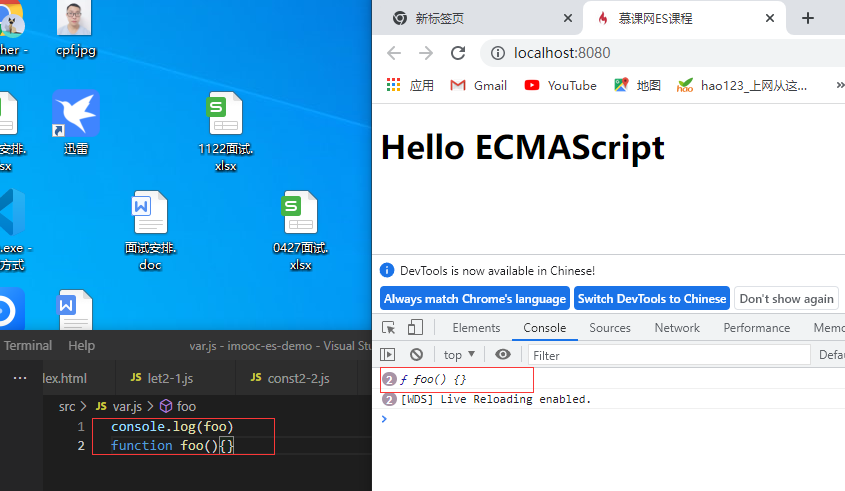
JSONP，CORS，网络代理，nginx，postMessage

1. Var let const区别

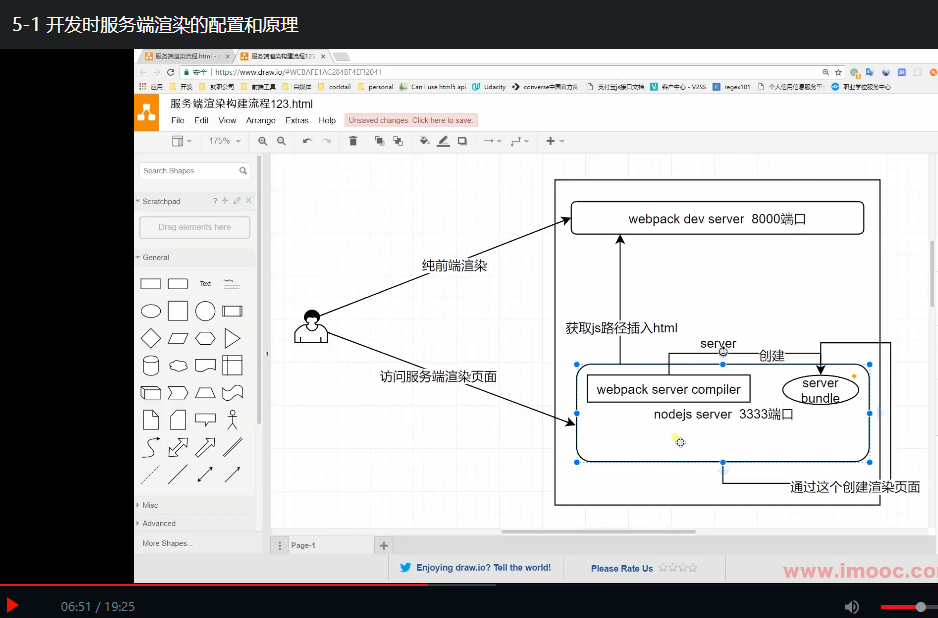




函数声明提升

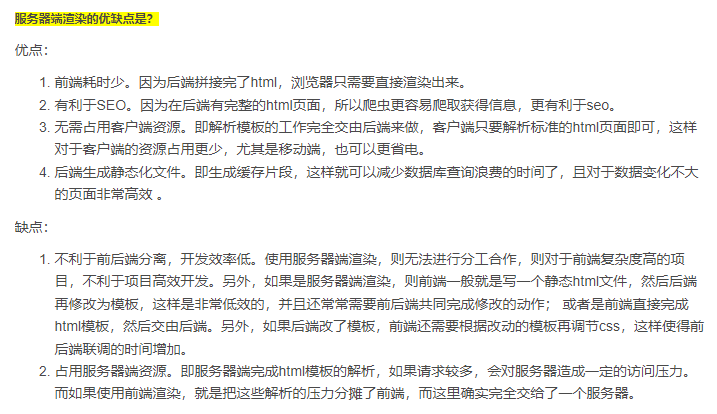


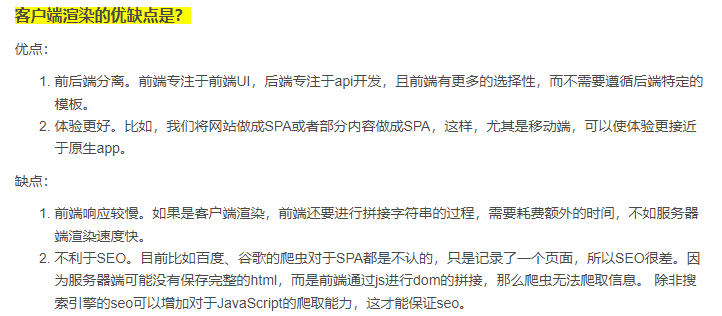
1. 服务端渲染



服务端渲染&客户端渲染优缺点：

<https://www.cnblogs.com/muzishijie/p/11198315.html>





1. 掌握最深的技术，以及在工作中的应用

用postMessage拿到跨域的x-token

最近的一个项目，在前端页面的一个弹窗要打开一个后端代码里的一个页面activity流程图，

页面能打开，但是要获取流程图的数据，需要用户的token信息，不能放到sessionStorage，

也不能放到url上，最后我就是想了用postMessage, 流程图页面用postMessage给主页面发个消息，并且进行监听，主页面收到消息后，就把token发给流程图页面，流程图页面接收后存在自己的sessionStorage里，然后重新加载就可以获取到了。

阅读源码改bug:

我做了一个表单嵌套表格的页面，有个前端也要做个类似的，然后就把我的代码拿过去修改，

快要做完的时候发现有个问题，他的页面表单校验不通过，就算填了数据，还是提示必填，

定位了好久也没找出什么问题，然后就来问我，我一时半会也没发现，好像一模一样的代码，怎么会有校验的问题？然后我就找到校验的代码，打断点进去看源码，最后才知道，我写的校验信息是个空格，空格是true, 是他写的是空字符串，空字符串是false，所以就校验不通过了。所以有有些奇奇怪怪的问题，要调试一下源码才能发现。

1. 体现自己优点的例子

尊重领导: 项目没有测试人员，前后端联调完了之后，任务点完成状态，项目经理就来测试了，会发现一些bug，不管是前端还是数据的问题，都往我身上推，我感觉想跟他吵架，但是人家是项目经理，在这么多人面前，我也不好驳他的面子，所以就没跟他顶嘴了。

能够较好的处理同事之间的矛盾：跟一个后台同事接口联调的时候，我真是急坏了，同个功能，他改完我一测又是问题，他也是挺烦，然后快下班的时候，他接口写完了，希望我调一下再下班，我就说我先去吃饭，然后他就对我说了几句脏话，并且说我别拖延他下班的时间，我挺生气的，就没理他，去楼下饭店，不过我为了节省时间，没在店里吃，而是打了包回办公室，然后就继续调接口了。因为这个事情，我感觉跟他之间有隔阂，不过想想，大家是同事，矛盾、口角是避免不了的，工作该交流还是继续交流。有一次我们办公事喝奶茶的时候，我就把闹矛盾的事情给拿出来调侃一下，他也说了他的想法，然后我们就没什么隔阂了。

1. Es6都有哪些语法

<https://www.jianshu.com/p/fe2aec6f307d>

语法：

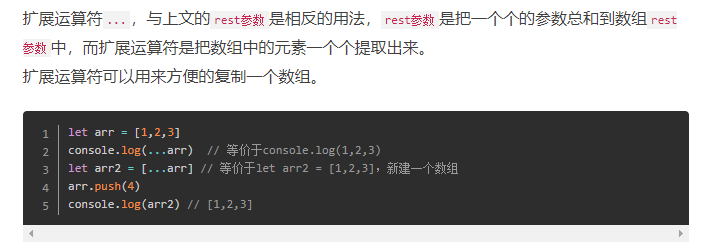
A/变量



B/解构赋值



C/ ...扩展运算符、rest参数



D/ Promise

E/ Generator函数

F/ async await

G/ Iteractor 和 for ... of ...

数据类型：

A/ 函数



B/ Symbol

C/ Set

D/ Map

E/ Proxy

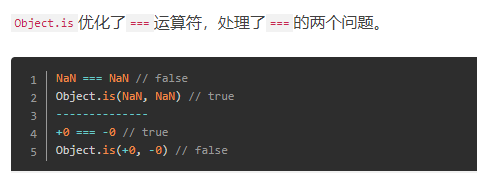
F/ Reflect

G/ String:



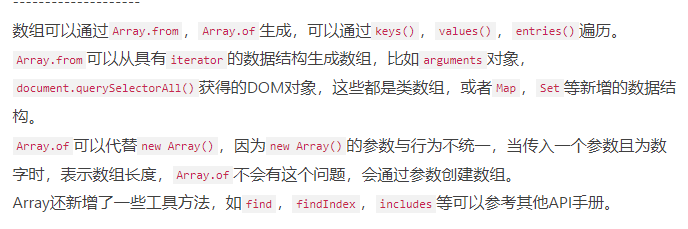


H/ Object:

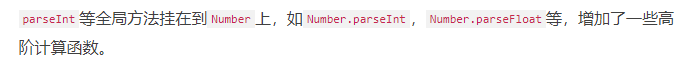




I/ Array:

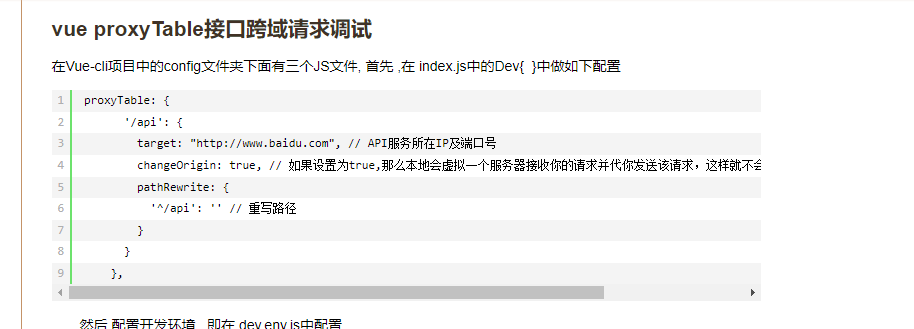


J/ Number:



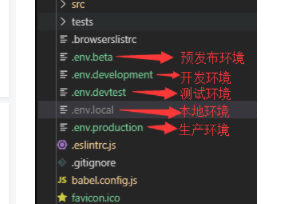
1. Vue 代理配置

<https://www.cnblogs.com/momeak/p/10563833.html>



1. Vue配置环境变量

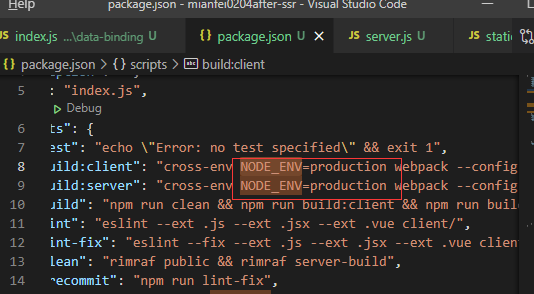
<https://blog.csdn.net/qq_34543252/article/details/106002332>



、



或者



1. vue路由两种模式的区别

<https://zhuanlan.zhihu.com/p/337073166>

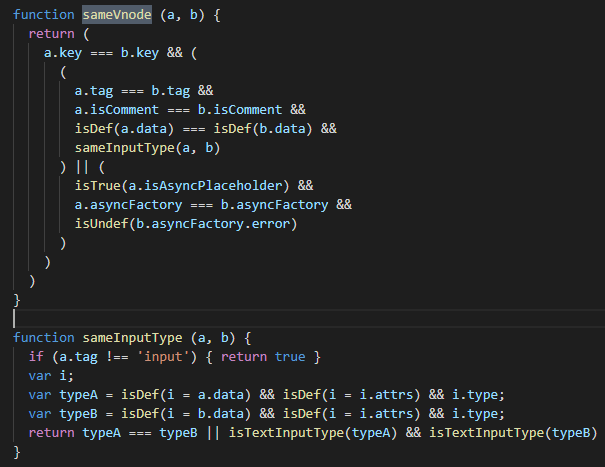


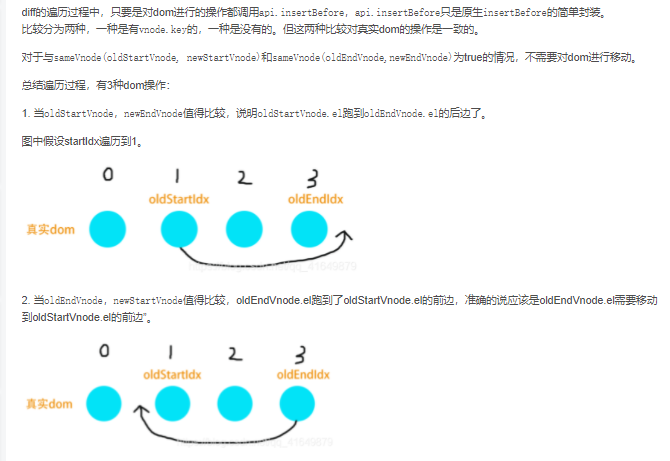


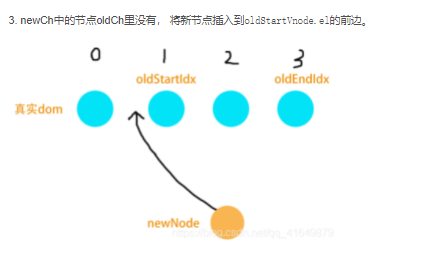


1. Patchvnode的逻辑

<https://blog.csdn.net/qq_41649879/article/details/83450695>



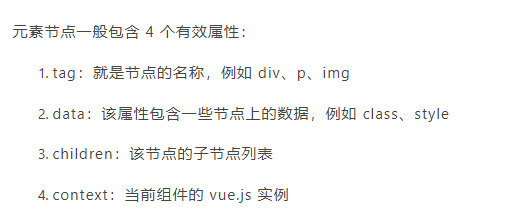
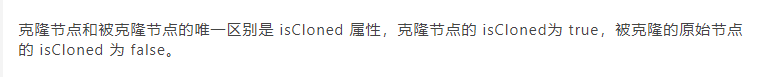
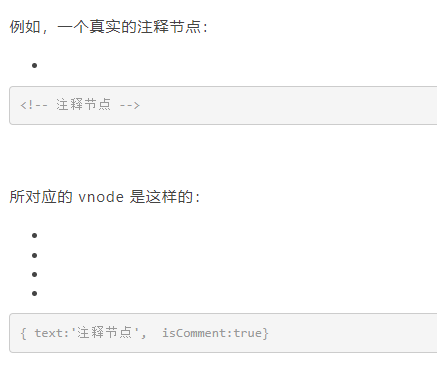
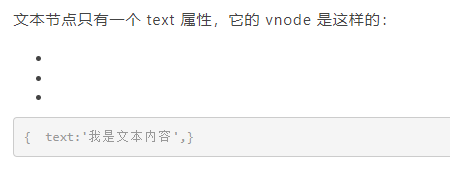






1. Vue VNode

<https://www.136.la/jingpin/show-208588.html>



1. F
2. F
3. F
4. f