# node

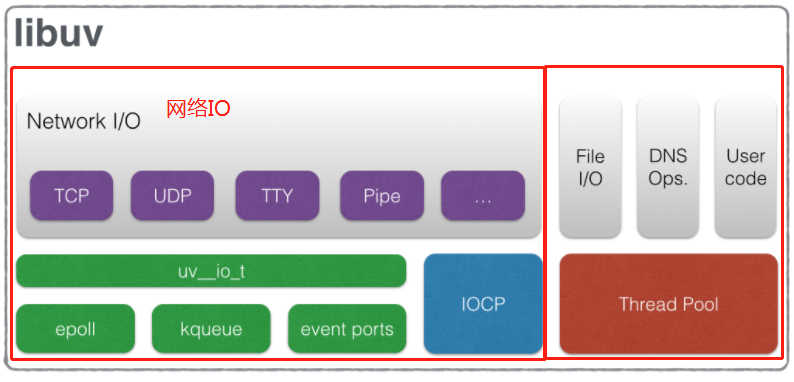
## 学习

是chrome的v8引擎，是js运行环境。外加一个底层I/O API的项目

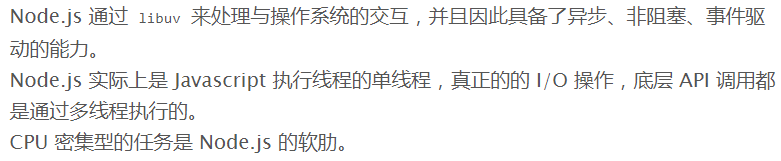
//两大特点：事件驱动和非阻塞IO



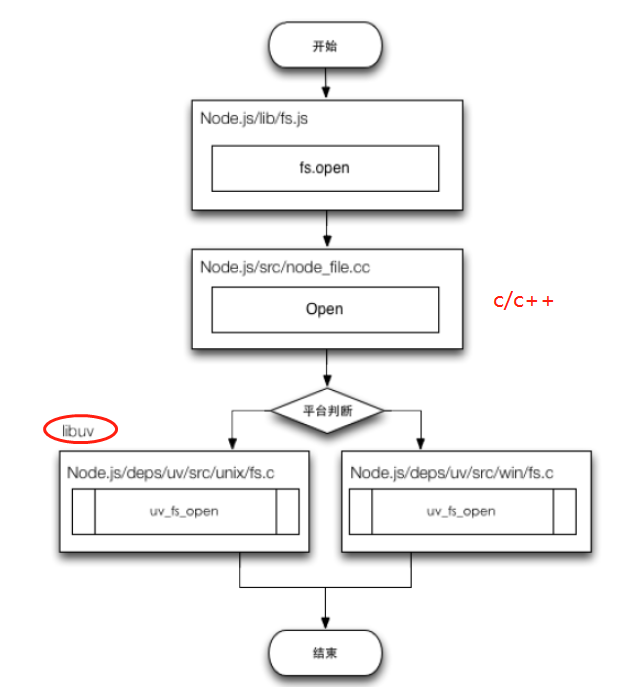


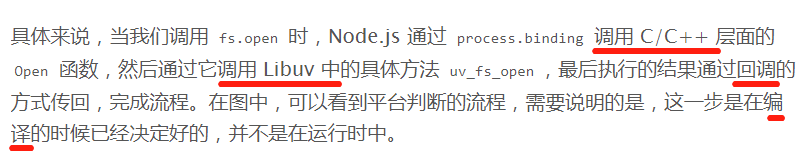


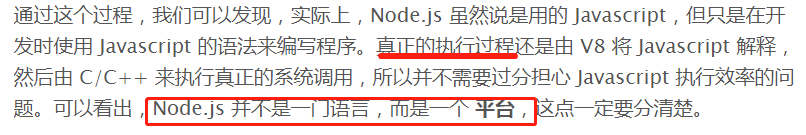
总结：



### 与操作系统的交互







### 异步非阻塞I/O

I/O包括文件io和网络io。

大致流程：

发起I/O调用：

Js代码调用Node核心模块，将参数、回调函数传入；

核心模块将它们封装成一个请求对象；

这个请求对象会被推入到I/O线程池等待执行；

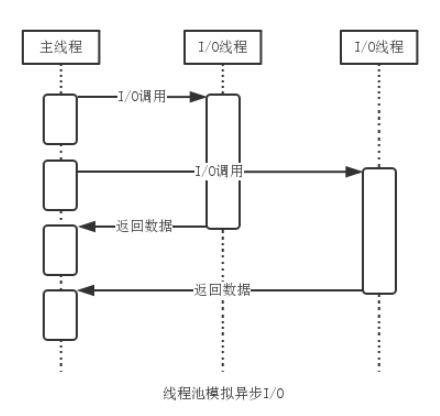
Js发起的异步调用结束，js线程（单线程）继续执行后续操作；

执行回调：

Io操作完成后，结果存储在请求对象的result属性上，并发出操作完成的通知；

每次事件轮询会检查是否有完成的io操作，如果有将这个请求对象加入到io观察者队列，之后当事件处理；

处理io观察者事件，取出封装在请求对象中的回调函数，执行并将result当作参数，完成js回调的目的。



//以上，nodejs单线程，指自身js运行环境的单线程，它并没有给js执行时创建新线程的能力，最终的实际操作是通过libuv及它的事件循环来执行的。所以这也是js是单线程语言，能在nodejs里实现异步操作的原因

### 事件驱动

大部分都用了Events模块，其只暴露了EventEmitter对象，支持多个监听函数绑定。



//解绑 removeListener,参数要一致 removeAllListeners(‘’)

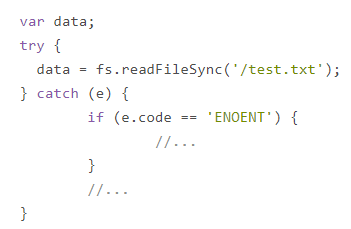
//列举绑定的监听事件： life.listeners(‘’).length

### 同步方法

Nodejs中大部分的异步操作函数都存在其对应的同步版本，只需在其名称后加上Sync即可，不传入回调

var file = fs.readFileSync('/test.txt', {"encoding": "utf-8});

注意：这种同步操作，没有回调函数，无法处理错误信息，需使用try…catch…来获取。



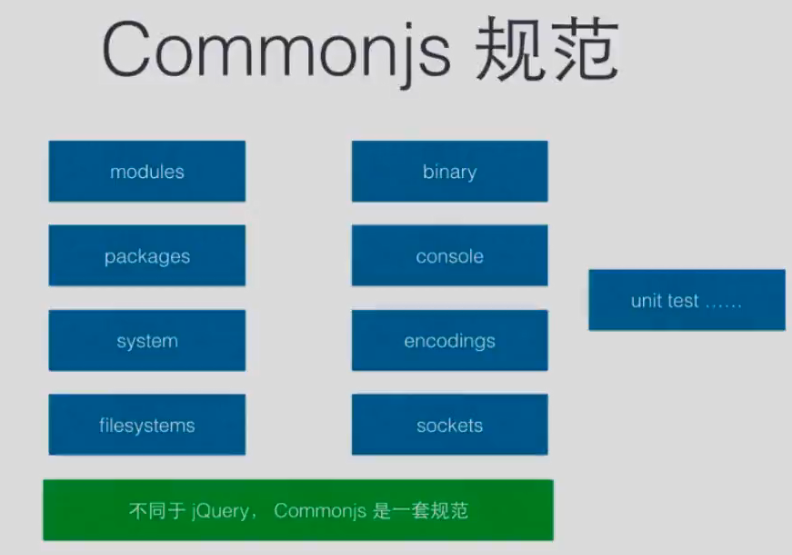
### 宿主全局变量：

命令行 输node， 编写js代码，回车可运行

Process

### Nodejs模块与commonjs

变量、函数 命名空间



Nodejs是对commonjs的实现

模块：核心模块、文件模块、第三方模块



模块流程：

创建模块（tearcher.js）、

导出模块（exports.add=function(){}）、

加载模块（var teacher = require(‘./teacher.js’)） //require的是exports对象

使用模块（teacher.add(‘xiaohong’)）

### Nodejs API

#### http

##### 浏览器输入地址发生了什么？

1. 会先查看自身浏览器DNS缓存
2. 搜索操作系统自身的DNS缓存

3、读取本地host文件

4、浏览器会发起一个dns的系统调用（域名解析请求）

宽带运营商服务器查看本身缓存；运营商服务器发起一个迭代DNS解析请求；

运营商服务器把结果返回操作系统内核同时缓存起来（ip地址）；

操作系统内核把结果返回浏览器；浏览器拿到了url对应的ip地址；

5、浏览器获得域名对应的ip地址后，发起http的‘三次握手’;（tcp连接请求）

6、tcp/ip连接建立后，浏览器可向服务器发送http请求

7、服务器端接受到这个请求后，根据路径参数，后端进行处理后，把结果返回给浏览器端。

8、浏览器拿到了响应结果，解析和渲染页面时，里面的js、css、图片等静态资源，他们同样也是一个个http请求，需重复上面步骤。

9、浏览器根据拿到的资源对页面进行渲染，最终呈现给用户一个完整页面

##### 启动服务



##### Http模块：

支持更多特性、处理流相关、不缓存请求和响应

非阻塞 异步

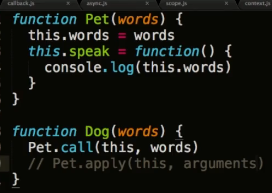
阻塞 同步

事件 事件驱动

事件循环 eventloop 异步回调函数队列 （先进先出）

作用域 跟变量、函数调用有关

上下文 this指向 继承

 call \ apply实现继承父类的speak等方法；

##### http性能小测试

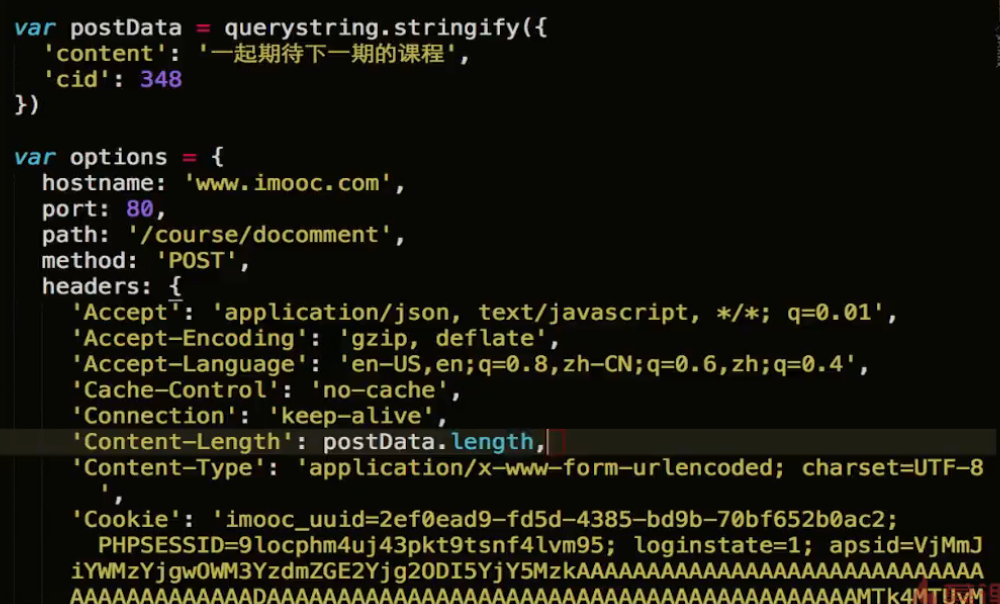
apache自带的ab工具 ab –n1000 –c10 http~

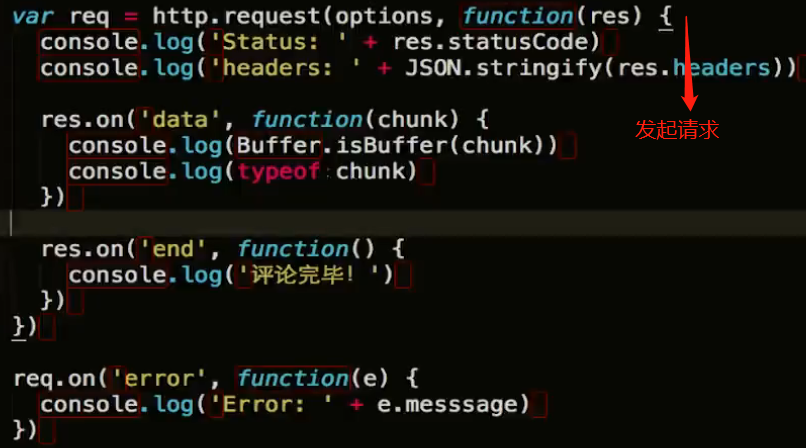
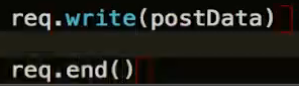
##### http小爬虫

 //结合cheerio 过滤数据

##### 请求 get/request

Get是对request的封装。



#### url .parse()

#### querystring

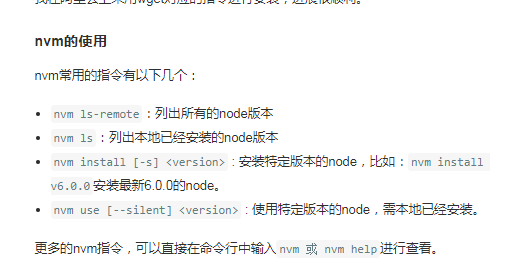
.stringify(json,’splitStr’,’=’) key=value&…

.parse()

.escape() uneacape()



## nvm管理node版本



## 快速启动本地服务器

npm install anywhere –g

在目标文件处 anywhere 8860 （指定端口号）

---浏览器会自动打开页面。 默认打开的是index.html页面。 Any

官方文档： <http://nodejs.cn/> （中文少）

<https://nodejs.org/en/> （英文）

## node event loop(阮一峰) （未完成？）

<http://www.ruanyifeng.com/blog/2018/02/node-event-loop.html>

<https://zhuanlan.zhihu.com/p/34924059> 是event loop 翻译，但是语言更白话一些( 未完成 )

## 重要类

### Buffer

Buffer类的实例类似于整数数组，其大小在创建的时候确定，且不可更改，在v8堆外分配物理内存。

在node.js中是一个全局变量，可直接使用，不需require。

**是什么？**

缓存区buffer是暂时存放输入输出数据的一段内存；

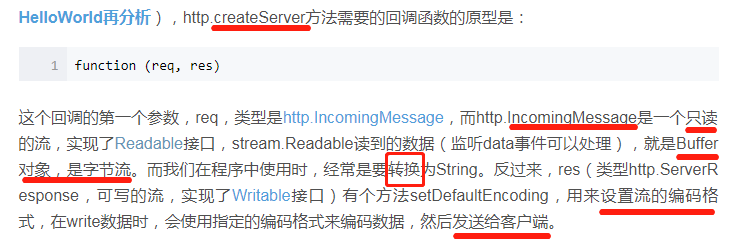
js语言自身只有字串类型，没有二进制类型，但处理tcp和文件流的时候，必须处理二进制数据；

nodejs提供了一个buffer对象来对二进制数据进行操作；

是表示固定内存分配的全局对象，即要放到缓存区的字节数据需要提前确定；

可以有效的在js中存储二进制数据;

字符串是有编码格式的，如utf8,buffer没有编码格式；



**字节、位？**

字节byte是计算机存储时的一种计量单位，一个字节表示8位二进制数；

一位表示0或1，每八位（bit）组成一个字节（byte）;

字节是网络传输信息的单位；

一个字节最大值十进制表示是255 （11111111 2的8次方减1）

一个单词是两个字节，一个汉字是三个字节；

**定位buffer的三种方式**

new Buffer(size) //size表示字节数 🡪 <Buffer 0e> 一个字节

new Buffer(Array) //数组的长度就是buffer的长度 🡪new Buffer([1,2,3]) 即 <Buffer 01 02 03> 三个字节

new Buffer(str,[encoding]) //字符串方式创建

**Buffer常用的方法**

bufStr.toString(‘utf8’) buf转换为字串

new Buffer(‘字串’) 字串转换为buf

Buffer.concat(list[,totalLength]) 合并

buf.copy(target[,targetStart[,sourceStart[,sourceEnd]]]) 复制

buf.slice([start[, end]]) 切片

buf.equals(otherBuffer) 比较相等 返回true/false

buf.compare(otherBuffer) 比较大小 0相等/-1小于/1大于

buf.write(string[,offset][,length][,encoding]) 向缓存区写数据，返回实际写入的字节数，注意不指定offset，总是从0开始写

Buffer.isBuffer(buf)

buf.byteLength / buf.length 字节长度

buf.toJSON() 转换为json格式 🡪当JSON.stringify(buf)时，buf.toJSON会被调用，转换为json字串 ， ‘{“type”:’Buffer’,”data”:[x,x,x]}’

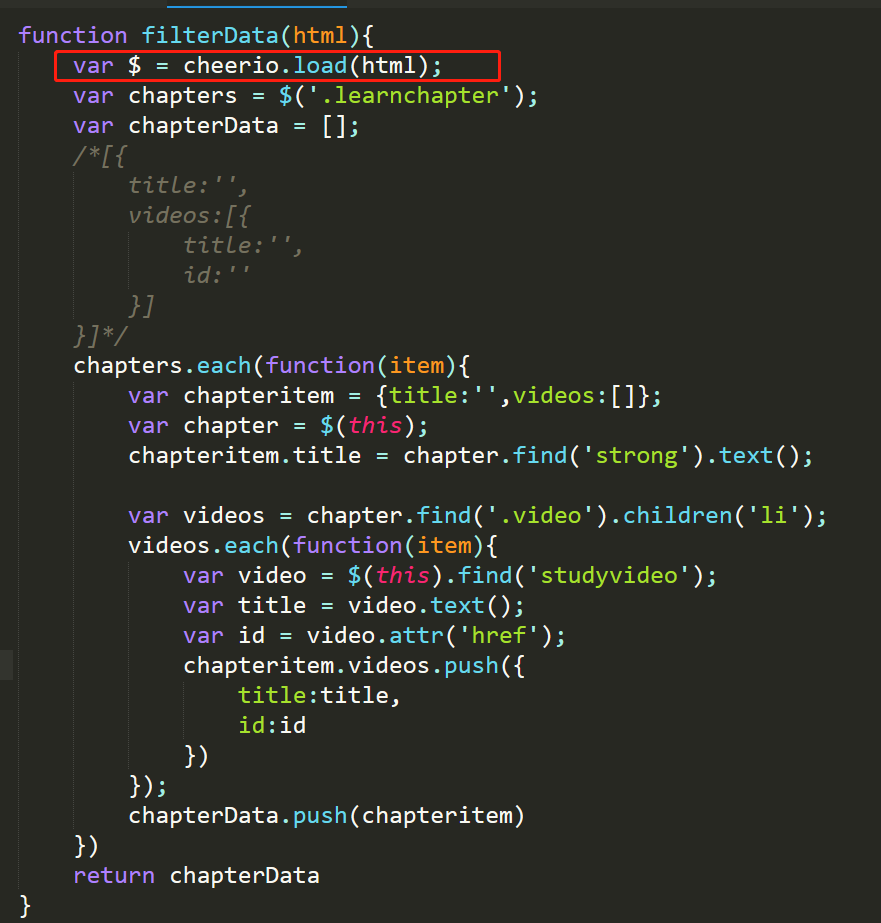
## NPM包（第三方模块）

### 代理请求 http-proxy-middleware模块



### cheerio

服务器端的类jquery核心实现



## 框架

Koa/Thinkjs/Eggjs

# 性能（不看）

## 。。

前后端分离，有时一个页面需要前端发送五六个请求去回填信息内容，有了nodejs，前端可以在nodejs中去代理多个异步请求，这个优化可以让整个渲染效率都提升很多（当然pc上多个请求没什么影响，但在手机端，建立一个http请求开销很大，有了这个优化，性能可以提升很多？？）

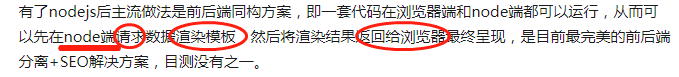
前后端分离：

前端专注于 UI层；控制逻辑、渲染逻辑；交互、用户体验

模板 XTemplate ,前后端通用的

路由共享？？ 在前端做浏览器路由时，可以同时配置服务端的路由 同时也处理了seo问题

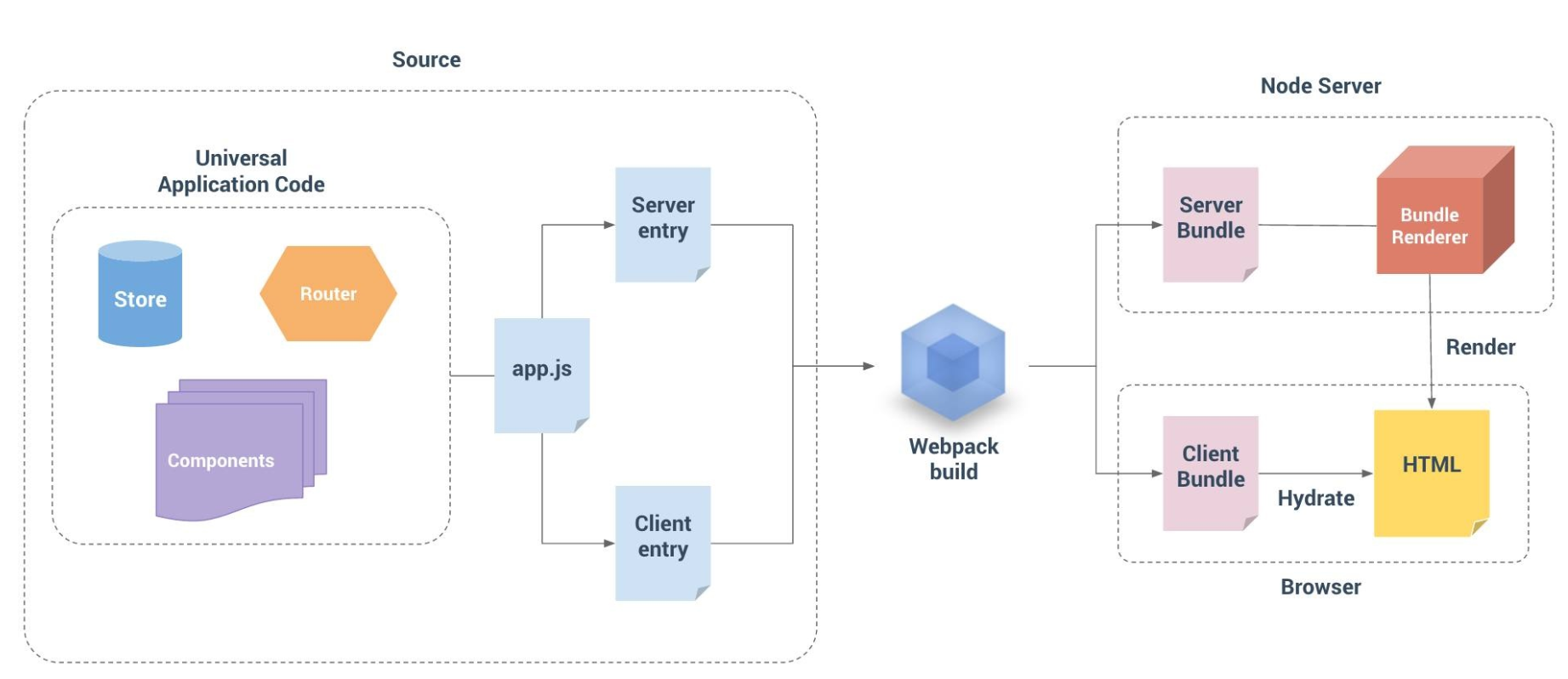
只要是做seo的产品就要做服务端渲染。



## 服务器端渲染VS浏览器端渲染

（<https://zhuanlan.zhihu.com/p/25936718> ）

Vue服务端渲染（SSR）



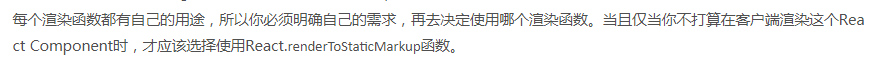
服务端渲染解决seo,更快渲染页面，（在加载js脚本的同时，浏览器就可以进行页面渲染），不必像spa应用，需整个js加载完成后才显示页面内容。

React服务端渲染 （<https://blog.csdn.net/limm33/article/details/50950862> 不懂）

React的虚拟dom是其可被用于服务端渲染的关键，每个reactComponent在虚拟dom中完成渲染，然后react通过虚拟dom来更新浏览器dom中发生变化的那一部分，虚拟dom作为内存中的dom表现，react可以从虚拟dom中生成一个字符串，而不是更新真正的dom，这使得可以在服务端和客户端使用同一个reactComponent。

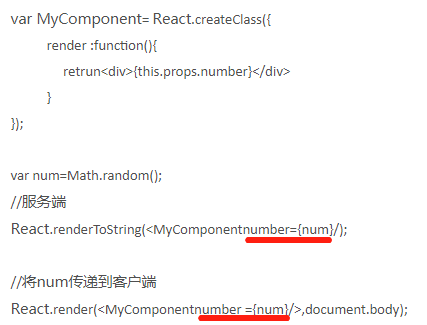
React提供了两个可用于服务端渲染组件的函数：React.renderToString、React.renderToStaticMarkup，服务端渲染考虑以下几点：1、选最优的渲染函数；2、如何支持组件的异步状态；3、如何将应用的初始化状态传递到客户端；4、哪些生命周期函数可用户服务端渲染；5、如何为应用提供同构路由支持；6、单例实例及上下文的用法；







//客户端和服务端打印随机数是相同的



# 其他未用

## 工作组件（皓哥推荐）

### 飞冰

<https://alibaba.github.io/ice/#/>



### nov-wechat（git-lab）

修复了bug。使用rollup替代webpack进行打包，进一步缩小体积（7kb-4.7kb）提供了多种入口类型：cjs、esm、iife。

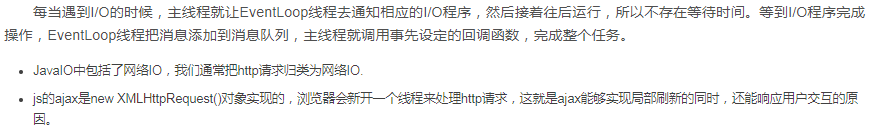
对于小组件或者lib，推荐使用rollup 进行打包

参考文档

- https://rollupjs.org/

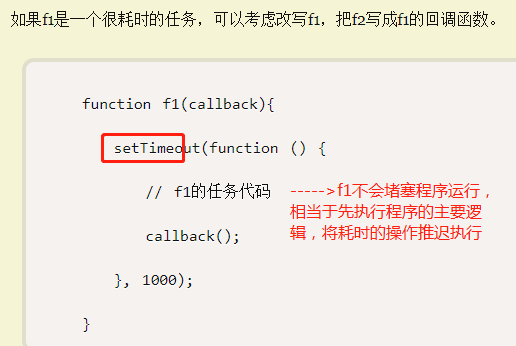
- https://github.com/rollup/rollup-starter-lib

### 异步代码的运行机制



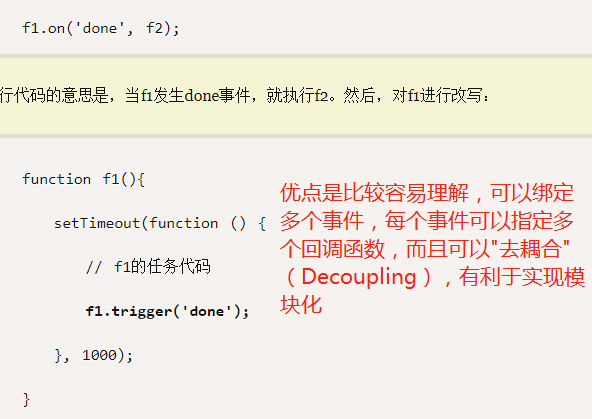
##### 🡪异步编程四种方法

1回调函数

 如: f1( f2 )

弊端： 不利于代码维护阅读，高度耦合

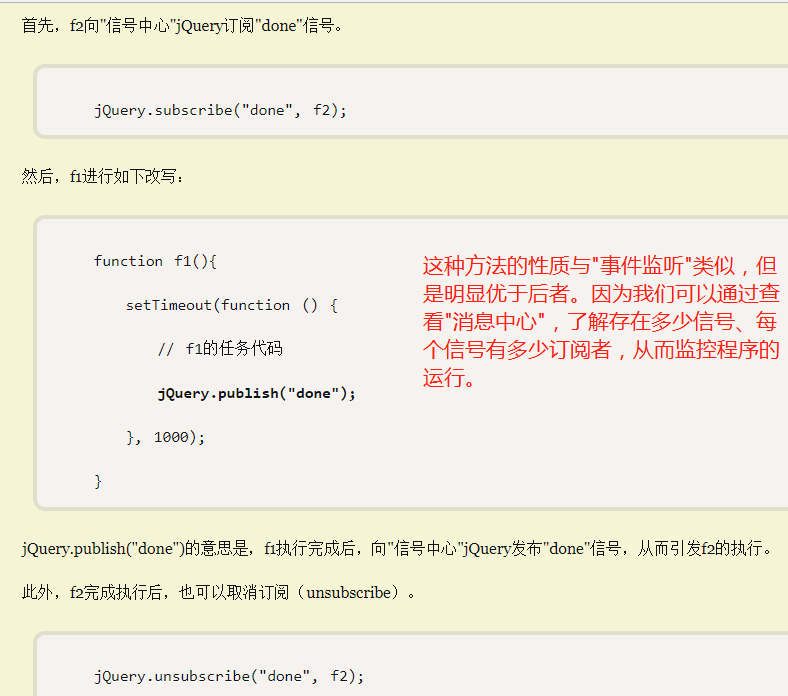
2事件监听（事件驱动模式）



弊端：整个程序变成事件驱动型，运行流程变得很不清晰

3发布订阅

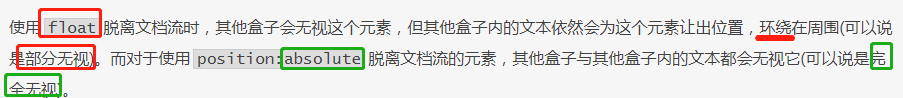




4 Promise对象

## 文档流

博客地址 <https://www.w3cplus.com/css/understand-css-stacking-context-order-z-index.html>





### 层叠上下文

z-index属性决定元素在层叠上下文 在z轴的顺序

每个元素都属于一个层叠上下文，给定层叠上下文中的每个定位元素都有一个层叠层级，数值更大的元素盒子总是在数值小的盒子上面，层级可能有负值。同一层级中的元素盒子按照文档树出现的顺序层叠在一起

形成条件：

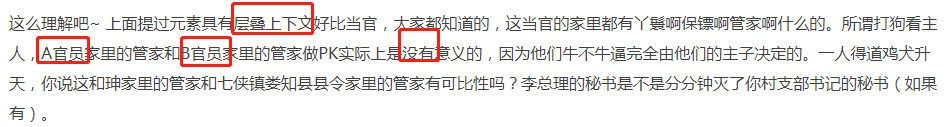


每个网页都有一个默认的层叠上下文。这个层叠上下文的根源就是html元素。

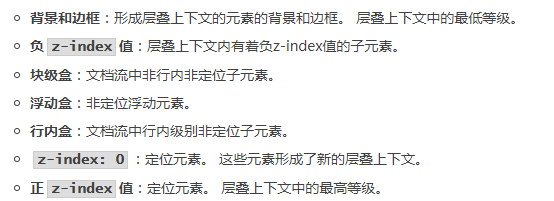
### 层叠水平

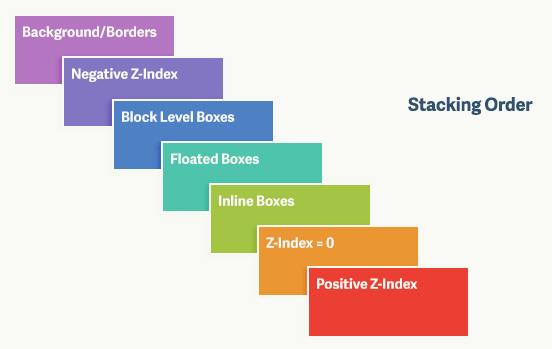
所有的元素都有层叠水平

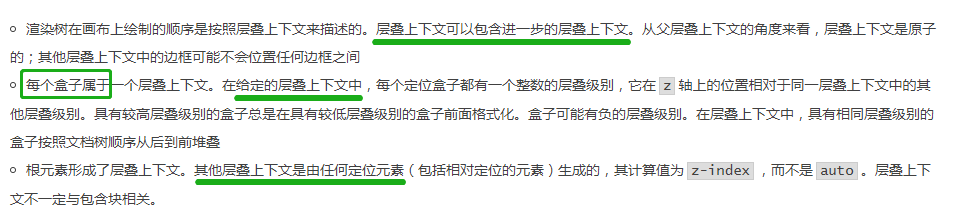




### 层叠次序(Stacking Order)







## 效果(h5)

### 下拉自动加载更多

Scroll监听的是产生滚动区域父容器，父容器一定要设置overflow:auto,形成滚动容器

不指定body\html的height为100%，当内容区过长，出现滚动条，可监听document/window的scroll事件

$(document).height() 内容区的总高度；

$(window).height() 窗口的高度（可视的高度）；

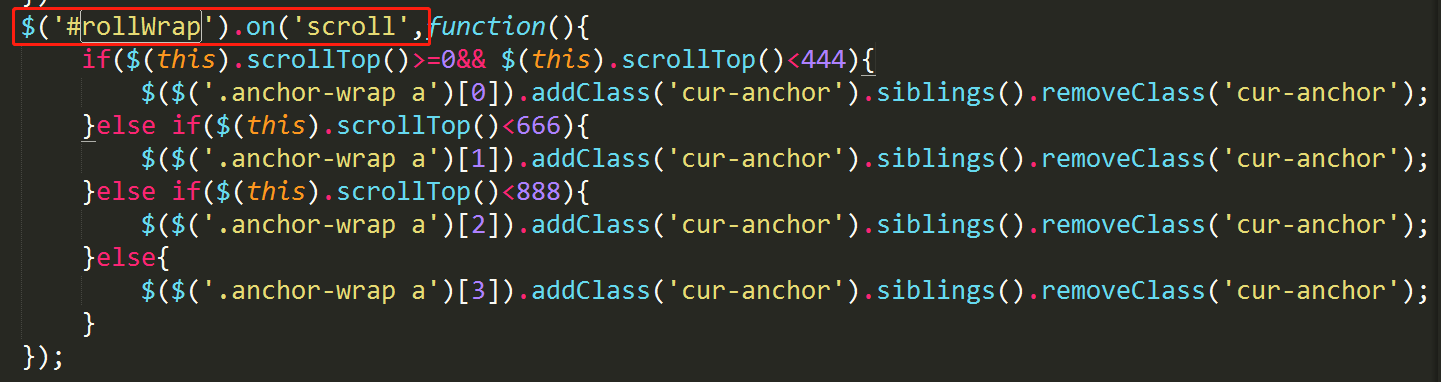
$(this).scrollTop() 滚动条的高度

指定body\html的height为100%，内容区过长时，可监听body的scroll事件

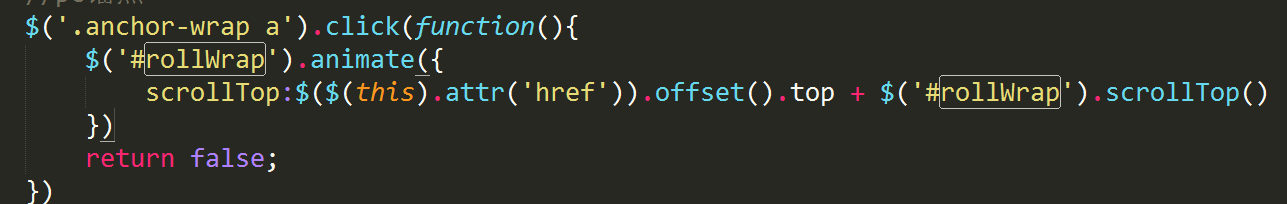
### 锚点



滚动页面，对应的锚点高亮；



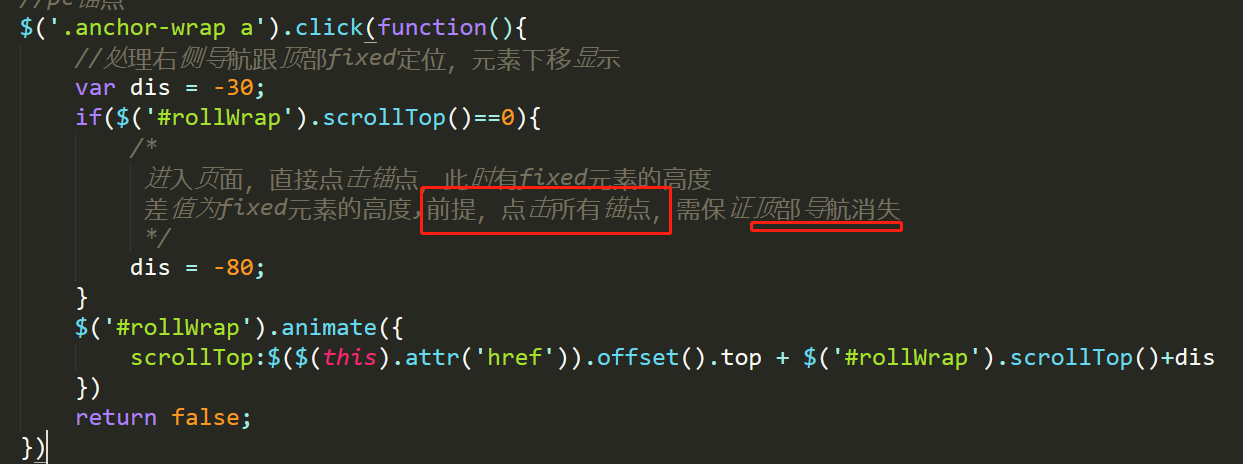
锚点点击，页面滚动到相应的位置

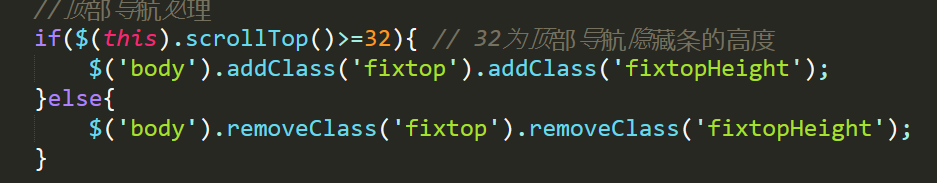


内容区的某元素的offset值是相对于offsetParent父容器说的，这里是相对于滚动盒子（页面滚动，盒子不动，目前设置是全屏），

所以内容区滚动的情况下 该元素的offset值是变化的。最后设置盒子的scrolltop = 元素的offset + 当前的scrolltop

//优化





京东大学学院 有类似效果源码查看： <https://xueyuan.jd.com/> （index.js）

### 页面随手指滑动（性能考虑）

## Meta

外站： <https://www.metatags.org/meta_name_revisit_after>

<https://blog.csdn.net/ttongzw/article/details/13776169> （little）

<meta name="viewport" content="width=device-width,initial-scale=1.0,maximum-scale=1.0,user-scalable=no">

<!-- 忽略识别电话 邮箱 -->

<meta name="format-detection" content="telephone=no">

<meta name="format-detection" content="email=no" />

<!--开启对app的支持 默认是yes,设为no,不显示默认的苹果工具栏和菜单栏？？-->

<meta name="apple-mobile-web-app-capable" content="yes">

<!--在 web app 应用下状态条（屏幕顶部条）的颜色；(改变顶部状态条的颜色)-->

<!--默认值为 default（白色），可以定为 black（黑色）和 black-translucent（灰色半透明）；-->

<meta name="apple-mobile-web-app-status-bar-style" content="black">

<!-- 添加到主屏后的标题 -->

<meta name="app-mobile-web-app-title" content="xx标题">

<!-- 智能app广告条，告诉浏览器这个网站对应的app，并在页面显示下载banner图 -->

<meta name="apple-itunes-app" content="app-id=myappstoreID,affiliate-data=myaffiliatedata,app-argument=myurl">

<!--HTTP-EQUIV类似于HTTP的头部协议，它回应给浏览器一些有用的信息，以帮助正确和精确地显示网页内容-->

<meta http-equiv="Content-Type" content="text/html">

<!-- 搜索引擎索引方式:index文件会被索引，follow页面上的链接会被索引，none都不能检索到-->

<meta name="robots" content="index,follow" />

<!—ok--页面重定向、刷新 以xx秒后页面刷新，加url表示重定向指到x页面-->

<meta http-equiv="refresh" content="0;url=">

<!-- 优先使用ie最新版本 chrome -->

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<!-- 浏览器内核控制，国内很多浏览器都是双内核 webkit和trident -->

<meta name="renderer" content="webkit|ie-comp|ie-stand">

<!-- 站点适配 format根据手机页的协议语言选择一种-->

<meta name="mobile-agent" content="format=[wml|xhtml|html5];url=X">

<!-- 转码申明 用百度打开页面可能会对其进行转码，比如贴上广告，以下避免转码 -->

<meta http-equiv="Cache-Control" content="no-siteapp">

完美适口：

<meta name="viewport" content="

width = [pixel\_value | device-width ] ,

initial-scale = float\_value ,

minimum-scale = float\_value ,

maximum-scale = float\_value ,

user-scalable = [yes | no] ,

target-densitydpi = [dpi\_value | device-dpi | high-dpi | medium-dpi | low-dpi]

/>

## 地图相关

仅限北京的列表：

获取经纬度--------获取城市---------初始化地图（经纬度、城市）

### 高德地图

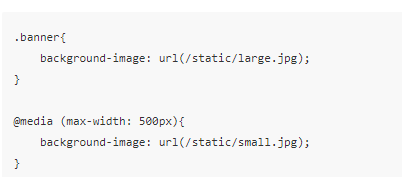
<http://lbs.amap.com/api/>

# 响应式网站（暂未用）

案例学习：<http://www.etycx.com/service.html?audience=151213>



响应式图片：



## 布局单位：（至有道）

1、Px为单位，当浏览器缩放或放大时，布局会乱，

2、em 为单位，一般是以body的font-size为基准，em值= 需要转换的像素值/父元素的font-size;

可解决上面缩放布局变乱问题，但其实是个不确定的值，总是相对于父元素，在任何子元素一直使用em，也不是很方便

3、**rem为单位**，是相对于根元素html的，兼容[Mozilla Firefox 3.6+](http://firefox.com/)、[Apple Safari 5+](http://apple.com/safari/)、[Google Chrome](http://google.com/chrome)、[IE9+](http://windows.microsoft.com/en-US/internet-explorer/products/ie/home)和[Opera11+](http://www.opera.com/)（ie6-8不支持）

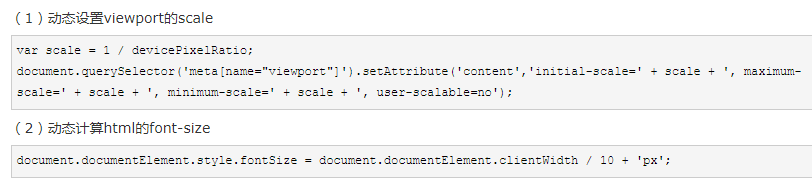
**网易的做法**，参照html的font-size，为了计算方便，设置根字体大小为100px ( 2x下 ),

移动端设计图750，宽为7.5rem，所以html的font-size = deviceWidth/7.5 (iphone6下，font-size = 375 / 7.5= 50px)

**淘宝的做法**：动态设置viewport中的scale 以及html的font-size

device-width = 设备的物理分辨率/（devicePixelRatio\*scale）,scale为1时，device-width = 设备的物理分辨率/devicePixelRatio；

淘宝根据设备像素比动态设置scale,当dpr为2时，scale为0.5；dpr为3时，scale为0.3333；这样做为了保证页面大小与设计稿保持一致

 //没有width=device-width,根字体是75px

计算：（less）



## 服务网站改版响应式：

### rem适配

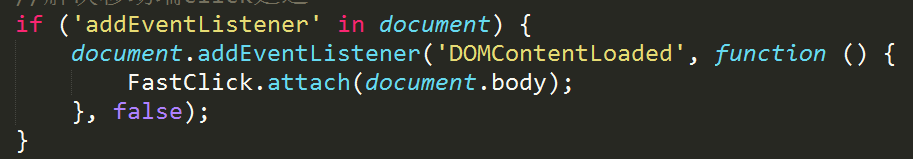
<https://www.cnblogs.com/noobfly/p/6207832.html> 网易适配说明

<**meta** name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0">



### 移动端click延迟

fastclick.js



Github: <https://github.com/ftlabs/fastclick>

### 一些小点

两列布局，inline-block,防止换行，写width,里面的点评标签防止换行，写white-space:nowrap

### SEO优化

URL优化：减少多余层级， http../page2.html

TDK优化：根据页面内容整理标签

<title></title>

<meta name=”keywords” content=””>

<meta name=”description” content=””>

CSS优化：缩减页面体积，提升蜘蛛抓取友好度，降低信躁比

如：可将源码中大量的css采用外部链接样式link引入

JS优化：缩减页面体积……..

如：将源码中的大段js采用外部链接样式引入

Img 添加alt,必要时添加title

A添加title,外部链接 rel=”nofollow”

## 适配（至有道）

### 一些基本概念(移至有道笔记)

#### CSS像素和设备像素

Css pixel:是浏览器使用的抽象单位

Device pixed: 显示屏幕的最小物理单位，每个dp包含自己的颜色、亮度

两者之间没有任何关系，为了编写页面引入一下视口概念，visual viewport 和layout viewport 都是以css像素绘制的。

#### 像素密度（ppi）->设备像素大小

像素密度(pixels per inch),指的是每英寸的长度上排列的像素点数量。

Ppi越高，代表屏幕显示效果越精细（如Retina屏），物理设备的一个像素点越小（指单个设备的像素大小是不定的）

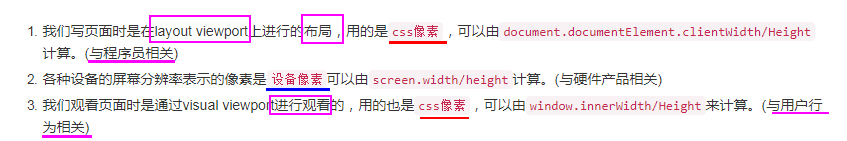


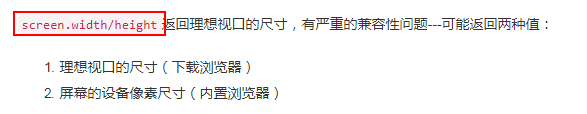
#### 视口（viewport）

PC浏览器，viewport等于浏览器窗口的宽度和高度，但在移动端设备上，有点复杂； （两者是联动的）

移动端的viewport太窄，为了能更好的css布局，提供了两个viewport：虚拟的visualviewport和布局的layoutviewport； （两者是独立的）

##### 0、总结：





##### 1、布局视口（****layout viewport****）：

A: 把layout viewport想象成为不会变更大小和形状的大图 ，想象拿一个小的框看这张大图；

**document.documentElement.clientWidth/Height可得到**

B: Width:100%,100%这种**百分比**是继承父元素而来的，区别于pc和移动端继承自哪里呢？

1:在pc中，有一个用来约束css布局适口的东西，叫做初始包含块，除去margin\padding布局视口与可视窗口宽度一致，也与浏览器本身宽度一致。

2:在移动端，布局一个20%，80%的块，body为100%，我们会看到iphone6下body的宽度是980，而浏览器的宽度只有375px,这里的**980px**就是移动端的布局视口，**并且被缩放在**了屏幕内。（无meta标签）

浏览器厂商为了让用户在小屏幕下网页也能够显示地很好，所以把布局视口宽度设置地很大，一般在768px ~ 1024px之间，最常见的宽度是980px。

这个宽度可以通过document.documentElement.clientWidth得到。

##### 2、视觉视口（visual viewport即虚拟视口）： ->可通过缩放改变

即页面（大图）当前显示在屏幕上的部分，用户可通过缩放来改变visual viewport的大小；

**window.innerWidth/Height可得到**

##### 3、缩放操作影响

缩放操作：

在pc端，缩放对应的布局宽度和视觉窗口宽度是联动的（放大操作，出滚动条 ----前提：元素指定写死的宽，不是使用百分比）

在移动端，**当放大页面，视觉视口变小，根本原因是css像素变大了，但是布局视口大小没变，意味着布局视口能容纳的css像素变少了；**

**窗口的大小发生变化的时候，layout viewport的大小也会改变，能够容纳的css像素越多.**

**一般默认设备的layout viewport大于 viaual viewport,这样不会破坏原有的页面布局，用户可通过平移或缩放来看网页的其他部分。**

##### 4、完美视口

相同像素密度的设备下，Css像素在**不放大或缩小**的情况下，在任何设备中**1个css像素都是一样大**的；但是不同像素密度的设备下，1个设备像素大小不相同，所以在没有任何设置的情况下，单位设备像素与单位css像素扯不上什么关系。

设置以下**meta**，可让两者扯上关系（width指的是layout viewport的宽度，与**设备的宽度**相等=设备转换后的大小=物理设备像素/dpr）

<meta name="viewport" content="width=device-width,initial-scale=1,maximum-scale=1,user-scalable=no">

//定义理想视口是浏览器的事情，并不能简单地认为是开发者定义的，开发者只能使用。

//**需要注意**的是： ios10+以上，尽管开发者设置user-scalable=no,safari还是允许用户通过手势来缩放（安卓手机各大厂商内置的浏览器也逐渐开放用户缩放，即使使用了meta标签进行设置）。

解决办法：只需要检测touch相关事件来阻止事件的触发即可

window.onload = function() {

// 同时按下两个手指

document.addEventListener('touchstart', function(event) {

if(event.touches.length > 1) {

event.preventDefault()

}

})

var lastTouchEnd = 0;

// 特别注意300ms时差的设置

document.addEventListener('touchend', function(event) {

var now = (new Date()).getTime();

if(now-lastTouchEnd <= 300) {

event.preventDefault();

}

lastTouchEnd = now;

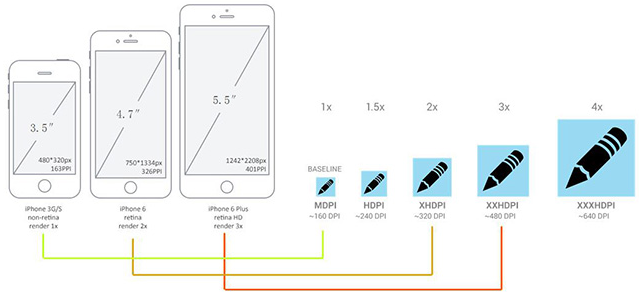
})

}

#### 像素密度与转换系数

Ios与android都会通过转换系数让控件适应屏幕尺寸。不同的像素密度PPI 转化系数不同

如：iphone6 plus的屏幕分辨率是1242×2208,转化系数为3 1242/3 = 414，这是设备自身进行的转化，我们设置后的layout viewport的大小也等于设备转换后的大小。



#### 设备独立像素

不同于设备像素（物理像素），它是虚拟化的。比如说css像素，我们常说的10px其实指的就是它--🡪即布局视口像素

CSS像素也可以称为**设备独立像素**(device-independent pixels)，简称为dips，单位是dp。

#### 设备像素比 （dpr）

缩写简称dpr，也就是我们经常在谷歌控制台移动端调试顶端会看到的一个值。设备像素比 = 设备像素 / css像素（垂直方向或水平方向）。可以通过JS来获取：window.devicePixelRatio

BavaScript# , window. devicepixe1Ratio FGlEÄÜiQEä5dpro fiE-CSS* , -webkit-device- 
-webkit-min-device-pixel -ratio -webkit-max-device-pixel-ratio , 
pi xel -ratio 
dpräfiiQE , 



独立设备像素 375 dpr=2 物理设备像素 750

某元素width：2px , 不同的屏幕上，css像素所呈现的物理尺寸是一致的，但对应的物理像素的个数不一致。

普通屏下，1个css = 1个物理像素，Retina屏下，1个css = 4个物理像素。

##### 1、Retina屏和普通屏（与模糊）

如iphone6,dpr=2,设备独立像素是375\*667,则设备像素是750\*1334



无论是高清屏还是普通屏，css像素所呈现的大小是一致的，不会受dpr的影响；但一个css像素所对应的物理个数是不同的；所以dpr低的屏幕，显示会模糊

##### 2、位图像素

是栅格图像（png\jpg等）的最小数据单元，包含显示位置，颜色，透明度等；

理论上1个位图像素对应1个物理像素，图片**可完美清晰的展示**；

但高清屏下，1个位图像素对应多个物理像素，如iphone6下，1个位图像素对应4个物理像素，由于位图像素已经是最小的单位了，于是为了能够显示出来，只能就近取色，从而导致所谓的图片模糊问题；

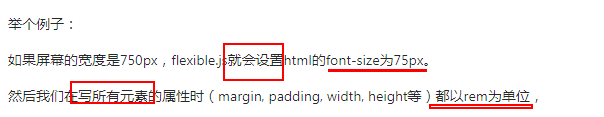
同样在普通屏下，

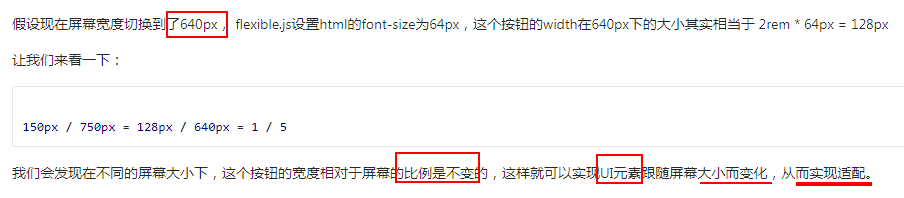
如何解决：使用跟dpr同个倍数大小的图片，比如iphone6,一个200×300的img标签,原图就要提供400×600的大小。

### Rem（移至有道笔记）

#### flexbile.js原理（ git地址： <https://github.com/amfe> ； 淘宝案例：<https://www.w3cplus.com/mobile/lib-flexible-for-html5-layout.html?utm_source=tuicool&utm_medium=referral> ）

flexible.js的原理就是根据屏幕的宽度设置html元素的font-size，具体做法是计算出屏幕的宽度然后除以10，赋给html标签的font-size属性。(以上这种算法只在屏幕宽度小于540px的情况下发生，这是为了避免在宽屏时，UI元素过大的情况，在大于540px的时候，html的font-size一直保持54px不变。)





##### 1、Rem布局与字体处理

flexbile实现，字体使用的是px,UI元素宽高等使用的rem为单位，rem实现了等比例的存在，所以在小屏幕下，存在小屏幕字体更小的情况，不利于我们去阅读，违背了适配的初衷。所以为了字体的适配，更好的做法是使用px和媒体查询来进行适配。

如下：对font-size进行放大处理 （scss）

@mixin font-dpr($font-size) {

font-size: $font-size;

[data-dpr="2"] & {

font-size: $font-size \* 2;

}

[data-dpr="3"] & {

font-size: $font-size \* 3;

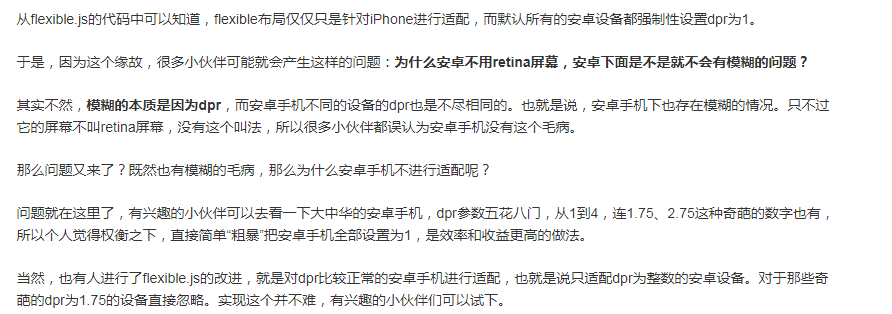
}

}

//由于retina屏幕下dpr的不同，我们又想显示的字体一样大，于是就给字体再增大dpr的倍数，这样当缩小dpr倍的时候，那么字体也就和设计稿所示的大小一样大了，在不同的手机中显示的大小也是一致的。

//（为什么是扩大dpr倍数，布局宽=物理设备宽/dpr, **rem= 布局宽/10**，即根字体大小； 所以适配字体时，可设置font-size再扩大相应的dpr，可保证页面里的字体大小相同，此时的字体非改变根字体大小。）

##### 2、Retina屏幕下的处理与安卓手机的适配



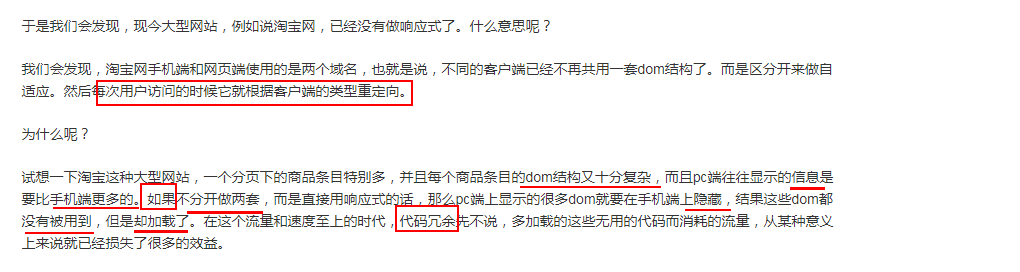
但是总是计算rem值，太过麻烦， 下面介绍gulp中的sass编译时px to rem的转换

#### px to rem

### 响应式与自适应的选择（弊端）（至有道）

响应式针对的是不同分辨率设备而进行的适配式设计，以利用**@media**规则为主要手段；

自适应则忽略@media，以比例布局为主，以适应不同浏览器窗口大小；



#### 1、@media

是依据布局视口宽度来设置的

### 简述方案 （<https://github.com/riskers/blog/issues/18> 不需要细看了）

目前有三种方案：

#### 固定高度，宽度自适应（至有道）

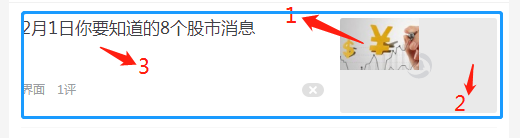
垂直方向使用定值，水平方向用百分比、定值、flex都行；

例如： <https://xw.qq.com/index.htm>l //腾讯网

学习点：

box-sizing 默认值 content-box，width、height为元素的内容框，在其外绘制元素的内边距和边框。‘



div的高度 写在了伪元素中（padding-top），2右浮动，且相对定位；

1的wrap 相对定位，left/bottom/left/right为0，占满（width:100%; height:100%也可完成）

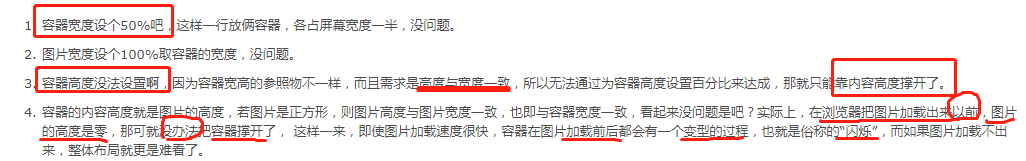
3为普通div,无特殊样式，布局先写右浮动的2，再写div1，完成两列布局；

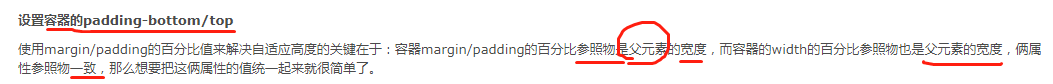
整体外面的高度是由2的高度撑开的，且需清浮动，浮动的元素（虽开启了相对定位，但是为了布局方便）不占文档流。

**结论**：巧用margin/padding的百分比实现高度自适应（多用于占位，避免闪烁）。

**当margin/padding取形式为百分比的值时，无论是left/right，还是top/bottom，都是以父元素的width为参照物的！**

**展示图片列表的页面，pc端可px写死宽高，但在移动端得靠百分比自适应；**





//**一般**是进行以下设置，给容器的伪元素设置padding,容器的高度撑开了；； 可解决给父容器设置padding，height为0，max-height设置失效问题；

//自适应的重点在于元素的宽高必须维持一个固定的比例，对于宽高不一致的情况，相应的修改margin/padding的百分比值便可，

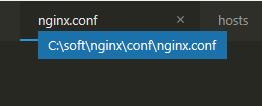


//其他宽高自适应方案：

CSS3推出的长度单位vw，都是以屏幕宽度为参照物的，只要给元素的width/height上都使用vm单位，只是有部分兼容性

# nginx: (解决跨域)

本地配置路径：





Host文件：



//接口访问写相对路径即可 api即可

# 移动端所需

## 调试工具

vconsole

也可以vue中使用： import vConsole form 'vconsole'; vue.use(vConsole)

## Web应用打包为app

Phonegap平台

参考链接 <https://www.cnblogs.com/w-wanglei/p/5565305.html>

## 一些小点

### ios12且微信6.7.4出现键盘弹出，收起时，页面下面有空白，fixed定位的元素，仍停留原位置

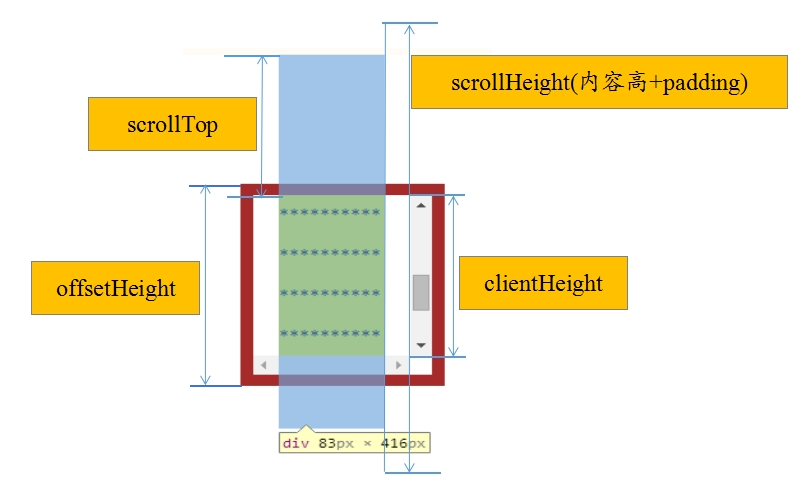


## 跨平台开发

### Hybrid app(混合应用)

兼具native app良好的用户交互体验 和 web app跨平台开发的优势

## 盒子模型 (scroll)

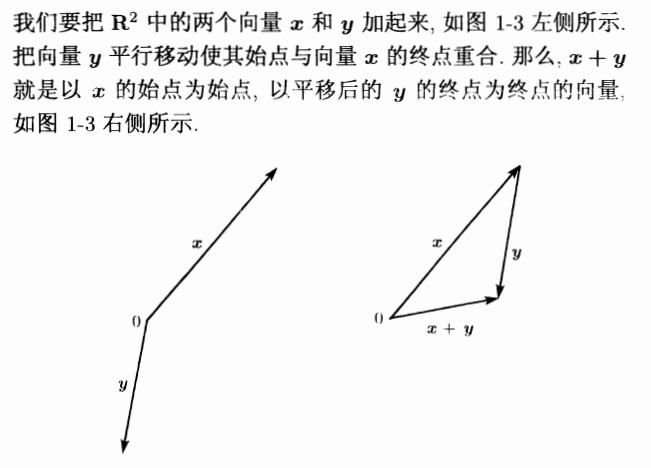


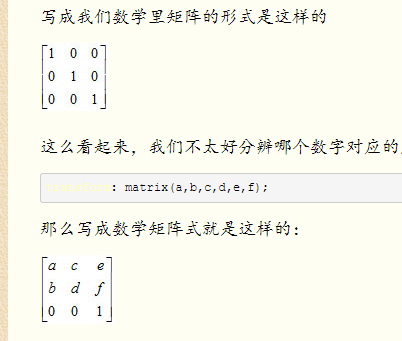
# Emmet快捷编辑





# 矩阵







# 服务网站开发前准备：

0 拉代码(IDEA)

VCS --> checkout from version 🡪 tfs 🡪

建一个虚拟workspace空间 🡪 连上对应服务器 下载对应代码

本地做映射处理

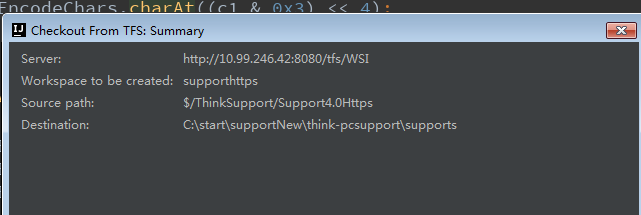
1 下载代码：

服务器：10.99.246.42

csuat\gaojj8，Lenovo2015

csuat\wanghao29，password01!

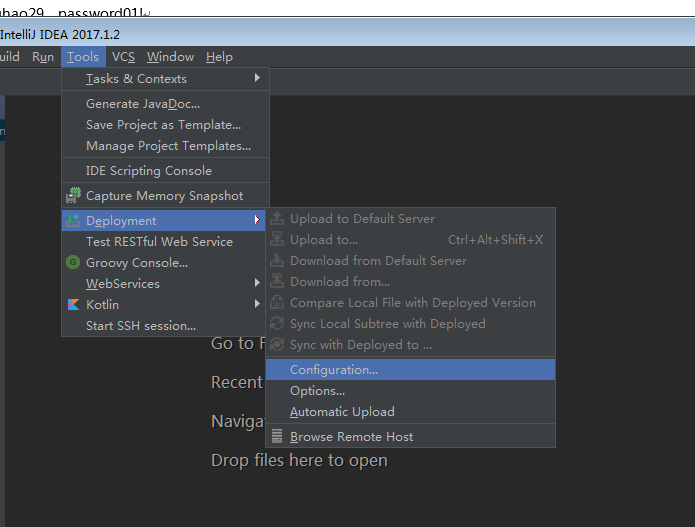
如果是在IDEA中配置，使用8080端口，domain=CSUAT

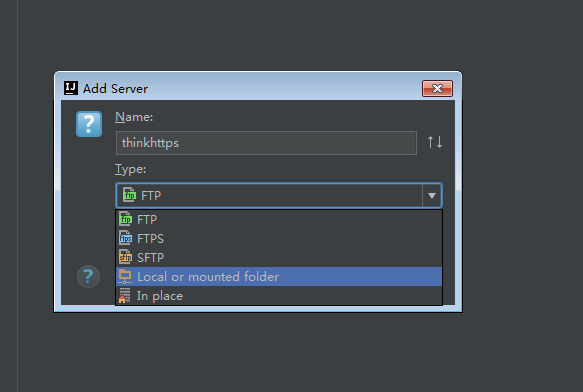


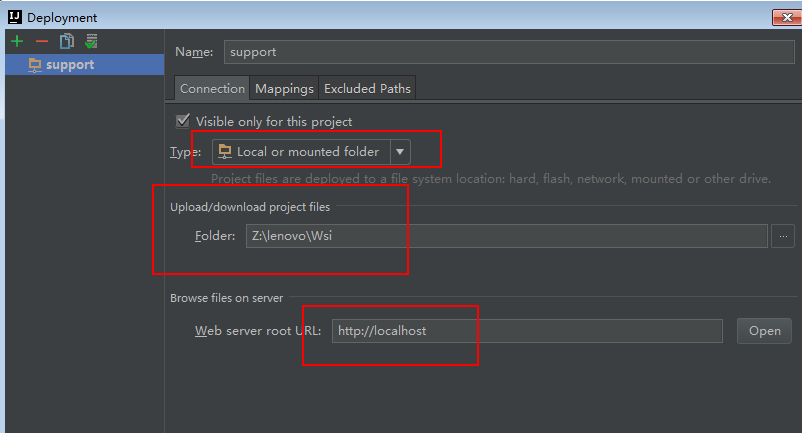
1. 创建本地测试映射

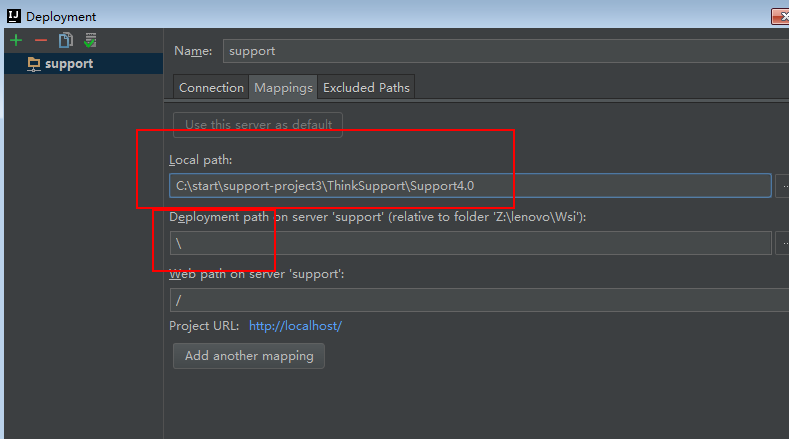
测试服务器：\\10.99.101.218，wsi\_uat/1234qwer!

1. idea配置 将本地项目与映射对应

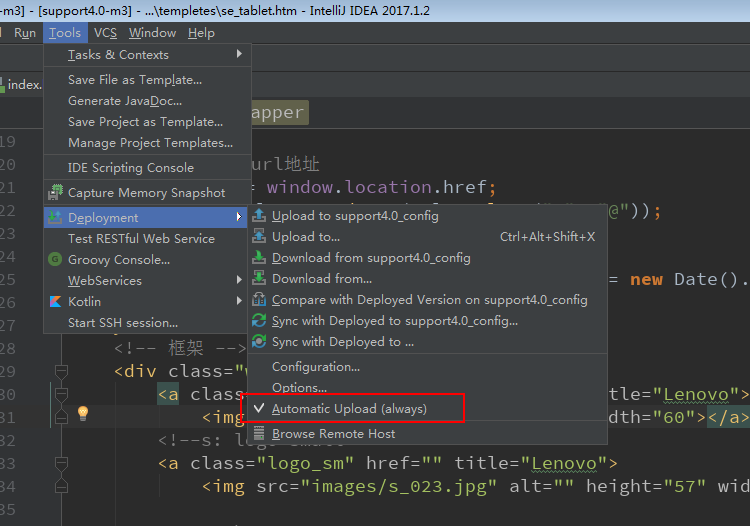








1. 设置本地修改 自动上传代码到本地测试映射



1. 本地访问：
   1. <http://supportuat.lenovo.com.cn>
2. IDEA 快捷键

Ctrl+Shift+N 查找

### 获取经伟度：

ios下只支持https方式 使用h5原生方式获取经纬度

安卓设备下，也支持http的方式获取经纬度

<script src="//pv.sohu.com/cityjson?ie=utf-8"></script> 获取ip地址

### 监听浏览器返回按钮

if (window.history && window.history.pushState) {

$(window).on('popstate', function() {

var hashLocation = location.hash;

var hashSplit = hashLocation.split("#!/");

var hashName = hashSplit[1];

if (hashName !== '') {

var hash = window.location.hash;

if (hash === '') {

alert('後退按鈕點擊');

window.location.href = '/'

}

}

});

window.history.pushState('forward', null,'personCenter\_setAccount.html');

}

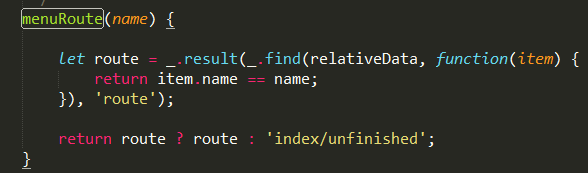
# 权限：

**提示功能未开放**的情况是：

**菜单写活**，展示的就是后台传回的数据，

本地自己写菜单对应的路由，点击新增的A模块，如果前端代码尚未部署，显示功能未开放

**逻辑处理**： 点击从本地菜单配置里找对应的A模块名对应的路由，找到则显示，未找到则返回 /unfinished 路由



# 其他地方新小点小记

3、<http://www.html5tricks.com/category/html5-demo/page/2>



-🡪 按钮波纹，<http://www.html5tricks.com/demo/svg-css3-ripple-button/index2.html>

<http://www.html5tricks.com/demo/svg-image-wave-effect/index.html>

向日葵： <http://www.html5tricks.com/demo/html5-svg-dancing-flower/index.html>

# ant

1、事件：onClick={this.showModalEditAdd()} 加了括号，事件中需return一个函数，若无括号，则事件的编辑直接编写触发，无return关键字

2、异象

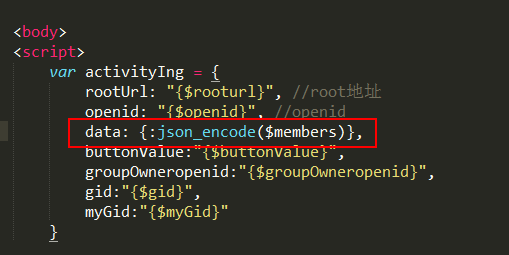


已经输入关键字，点击查找，展示新的列表，但上面的输入变空了，所以需要此时数据的记忆。

# 其他相关（php等）

## Redis

## PHP



## Idea激活

http://blog.csdn.net/u013772876/article/details/70763759

# 继续探索

### 跨域



## 博客学习

<http://blog.csdn.net/applebomb/article/details/60765329> （webpack）