事件循环：  
我们设置一个延迟函数的时候，当前脚本并不会阻塞，它只是会在浏览器的事件表中进行记录，程序会继续向下执行。当延迟的时间结束之后，事件表会将回调函数添加至事件队列（task queue）中，  
事件队列拿到了任务过后便将任务压入执行栈（stack）当中，执行栈执行任务。  
事件队列是一个存储着待执行任务的队列，其中的任务严格按照时间先后顺序执行，排在队头的任务将会率先执行，而排在队尾的任务会最后执行。事件队列每次仅执行一个任务，在该任务执行完毕之后，  
再执行下一个任务。执行栈则是一个类似于函数调用栈的运行容器，当执行栈为空时，JS 引擎便检查事件队列，如果不为空的话，事件队列便将第一个任务压入执行栈中运行。  
由于 JS 是单线程的，同步执行任务会造成浏览器的阻塞，所以我们将 JS 分成一个又一个的任务，通过不停的循环来执行事件队列中的任务。这就使得当我们挂起某一个任务的时候可以去做一些其他的事情，  
而不需要等待这个任务执行完毕。所以事件循环的运行机制大致分为以下步骤：  
检查事件队列是否为空，如果为空，则继续检查；如不为空，则执行取出事件队列的首部，压入执行栈；执行任务；检查执行栈，如果执行栈为空，则跳回第 1 步；如不为空，则继续检查；  
事件循环器会不停的检查事件队列，如果不为空，则取出队首压入执行栈执行。当一个任务执行完毕之后，事件循环器又会继续不停的检查事件队列，不过在这间，浏览器会对页面进行渲染。  
这就保证了用户在浏览页面的时候不会出现页面阻塞的情况，这也使 JS 动画成为可能， jQuery 动画在底层均是使用 setTimeout 和 setInterval 来进行实现。想象一下如果我们同步的执行动画，  
那么我们不会看见任何渐变的效果，浏览器会在任务执行结束之后渲染窗口。反之我们使用异步的方法，浏览器会在每一个任务执行结束之后渲染窗口，这样我们就能看见动画的渐变效果了。  
JavaScript 是一门单线程的语言，但是其事件循环的特性使得我们可以异步的执行程序。这些异步的程序也就是一个又一个独立的任务，这些任务包括了 setTimeout、setInterval、ajax、eventListener 等等。关于事件循环，  
我们需要记住以下几点：  
事件队列严格按照时间先后顺序将任务压入执行栈执行；  
当执行栈为空时，浏览器会一直不停的检查事件队列，如果不为空，则取出第一个任务；  
在每一个任务结束之后，浏览器会对页面进行渲染；  
  
前端路由：  
假如我们有一台提供 Web 服务的服务器的网络地址是：10.0.0.1，而该 Web 服务又提供了三个可供用户访问的页面  
http://10.0.0.1/；http://10.0.0.1/about；http://10.0.0.1/concat  
当用户使用 http://10.0.0.1/about 来访问该页面时，Web 服务会接收到这个请求，然后会解析 URL 中的路径 /about，在 Web 服务的程序中，该路径对应着相应的处理逻辑，程序会把请求交给路径所对应的处理逻辑，  
这样就完成了一次「路由分发」，这个分发就是通过「路由」来完成的。  
使用场景,更多用在单页应用上，基本都是前后端分离

优点：  
从性能和用户体验的层面来比较的话，后端路由每次访问一个新页面的时候都要向服务器发送请求，然后服务器再响应请求，这个过程肯定会有延迟。而前端路由在访问一个新页面的时候仅仅是变换了一下路径而已，  
没有了网络延迟，对于用户体验来说会有相当大的提升。  
缺点：  
1、对SEO不是很友好2、在浏览器前进和后退时候重新发送请求，没有合理缓存数据。3，初始加载时候由于加载所有模块渲染，会慢一点。  
  
substr、substring、slice、split、splice区别  
slice()  
可以从已有的数组中返回选定的元素，使用范围Array和String  
obj.slice(start, end)  
start: 必需。规定从数组（字符串）的哪个index开始选取，如果是负数，则从数组（字符串）尾部算index，比如，-1指最后一个元素，-2指倒数第二个元素。  
end:可选。规定到哪个index之前结束。如果没有指定该参数，那么切分的数组（字符串）包含从start到数组（字符串）结束的所有元素。如果这个参数是负数，那么它规定的是从数组（字符串）尾部开始算起的元素。  
返回值：  
返回一个新的数组（字符串），包含从start到end（不包含改元素）的obj中的元素。  
  
splice()有三个参数 第一个索引 第二数量，第三新加元素  
arrayObject.splice(index, howmany, ite1,......, itemxX)  
向/从数组中添加、删除项目，然后返回被删除的项目，可以有负数  
该方法会改变原始数组；  
返回值是会返回被截取之后的数组  
使用范围 array  
[$1, $1+$2-1]  
  
split() 用于把一个字符串分割成字符串数组  
适用String  
stringObject.split(seperator, howmany)  
seperator: 必需。字符串或者正则表达式，从该参数指定的地方分割stringObject  
返回值 一个字符串数组  
该数组是在separator指定的边界处将字符串stringObject分割成子串创建的。返回的数组中的子串不包括seperator自身。但是，如果seperator是包含子表达式的正则表达式，那么返回的数组中包括与这些子表达式匹配的字符串（但不包括与整个正则表达式匹配的文本）。  
  
substring() 不接受负的参数  
用于提取字符串中介于两个指定index之间的字符  
使用范围 String  
stringObject.substring(start, stop)  
start: 必需。一个非负的正数，规定要提取的子串的第一个字符在stringObject中的位置。  
stop: 可选。一个非负的正数，比要提取的子串的最后一个字符在stringObject中的位置多1.如果省略该参数，那么返回的子串会一直到字符串的结尾。  
  
substr()  
stringObject.substr(start, length)  
一个新的字符串，包含从stringObject的start处开始的length哥字符。  
  
移动端标准1像素 \*  
移动端必问  
一般在移动端，由于dpr（设备像素比）不为1，在PC端显示1像素的边框，在移动端其实显示为2px。解决这个问题，主要思想是：使用伪元素设置1px的边框，然后使用媒体查询，根据dpr的大小，对边框进行缩放（scaleY）。  
  
  
移动端与PC端有什么不同  
先说布局方面：PC端咱们最常用的就是固定宽度980px（也有960，1000,1200），然后水平居中 width:980px;margin:0 auto;  
但移动端就不能这么用了，因为很多网页都是可以横屏看，也可以竖屏看；很多屏幕的分辨率不一样；所以只要牵涉到移动端，就要牵涉到响应式（也叫自适应）；  
如果是只针对移动端的项目，我平时主要考虑的是320px宽 到 750px宽的兼容；https://www.zhihu.com/question/34364365/answer/58460054  
  
  
移动端常见适配方案  
  
1.使用rem单位来做网页适配，这个是我比较推荐的一种，效果很好，浏览器的兼容性也不错  
改变html的font-size,  
2、第二种就是使用浏览器的缩放功能来做，  
先按设计稿还原html，然后再设置viewport属性，  
3、适应响应式布局来做，用媒体查询，  
4.使用自适应百分比来布局，

移动端点击穿透原理以及解决方案问题：  
  
移动浏览器提供一个特殊的功能：双击（double tap）放大  
300ms的延迟就来自这里，用户碰触页面之后，需要等待一段时间来判断是不是双击（double tap）动作，而不是立即响应单击（click），等待的这段时间大约是300ms。  
点击穿透问题：点击蒙层（mask）上的关闭按钮，蒙层消失后发现触发了按钮下面元素的click事件  
蒙层的关闭按钮绑定的是touch事件，而按钮下面元素绑定的是click事件，touch事件触发之后，蒙层消失了，300ms后这个点的click事件fire，event的target自然就是按钮下面的元素，因为按钮跟蒙层一起消失了  
跨页面点击穿透问题：如果按钮下面恰好是一个有href属性的a标签，那么页面就会发生跳转  
因为a标签跳转默认是click事件触发，所以原理和上面的完全相同  
另一种跨页面点击穿透问题：这次没有mask了，直接点击页内按钮跳转至新页，然后发现新页面中对应位置元素的click事件被触发了  
和蒙层的道理一样，js控制页面跳转的逻辑如果是绑定在touch事件上的，而且新页面中对应位置的元素绑定的是click事件，而且页面在300ms内完成了跳转，三个条件同时满足，就出现这种情况了  
  
解决方案：  
不要混用touch和click  
既然touch之后300ms会触发click，只用touch或者只用click就自然不会存在问题了  
吃掉（或者说是消费掉）touch之后的click  
依旧用tap，只是在可能发生点击穿透的情形做额外的处理，拿个东西来挡住、或者tap后延迟350毫秒再隐藏mask、pointer-events、在下面元素的事件处理器里做检测（配合全局flag）等等，能吃掉就行  
最简单的解决方案，完美解决点击穿透问题  
把页面内所有click全部换成touch事件（touchstart、’touchend’、’tap’），需要特别注意a标签，a标签的href也是click，需要去掉换成js控制的跳转，或者直接改成span + tap控制跳转。如果要求不高，不在乎滑走或者滑进来触发事件的话，span + touchend就可以了，毕竟tap需要引入第三方库  
  
  
  
事件流描述的是从页面中接收事件的顺序。  
事件发生时会在元素节点与根节点之间按照特定的顺序传播，路径所经过的所有节点都会收到该事件，这个传播过程即DOM事件流。  
  
浏览器能同时并发多少请求  
浏览器的并发请求数目限制是针对同一域名的。即一时间针对同一域名下的请求有一定数量限制。超过限制数目的请求会被阻塞。

请求头的常用属性：  
http请求中的常用头（请求头）的含义：  
Accept：告诉服务器，客户端支持的数据类型。  
Accept-Charset：告诉服务器，客户端采用的编码。  
Accept-Encoding：告诉服务器，客户机支持的数据压缩格式。  
Accept-Language：告诉服务器，客户机的语言环境。  
Host：客户机通过这个头告诉服务器，想访问的主机名。  
If-Modified-Since:客户机通过这个头告诉服务器，资源的缓存时间。  
Referer:客户机通过这个头告诉服务器，它是从哪个资源来访问服务器的。（一般用于防盗链）  
User-Agent:客户机通过这个头告诉服务器，客户机的软件环境。  
Cookie：客户机通过这个头告诉服务器，可以向服务器带数据。  
Connection：客户机通过这个头告诉服务器，请求完后是关闭还是保持链接。  
Date：客户机通过这个头告诉服务器，客户机当前请求时间。  
  
http请求中常用的响应头的含义：  
Location:这个头配合302状态码使用，告诉用户端找谁。  
Server:服务器通过这个头，告诉浏览器服务器的类型  
Content-Encoding:服务器通过这个头，告诉浏览器数据采用的压缩格式。  
Content-Length:服务器通过这个头，告诉浏览器回送数据的长度。  
Content-Language：服务器通过这个头，告诉服务器的语言环境。  
Content-Type:服务器通过这个头，回送数据的类型  
Last-Modified:服务器通过这个头，告诉浏览器当前资源的缓存时间。  
Refresh:服务器通过这个头，告诉浏览器隔多长时间刷新一次。  
Content-Disposition:服务器通过这个头，告诉浏览器以下载的方式打开数据。  
Transfer-Encoding:服务器通过这个头，告诉浏览器数据的传送格式。  
ETag:与缓存相关的头。  
Expires:服务器通过这个头，告诉浏览器把回送的数据缓存多长时间。-1或0不缓存。  
Cache-Control和Pragma：服务器通过这个头，也可以控制浏览器不缓存数据。  
Connection:服务器通过这个头，响应完是保持链接还是关闭链接。  
Date:告诉客户机，返回响应的时间。