*如何语义化：*

* 一般的网站分为头部、导航、文章、侧栏、底部，根据不同的部位，使用不同的标签进行书写。
* 表示页面不同位置的标签：header、nav、article、section、footer、aside。
* 尽可能少的使用无语义的div和span标签
* 在语义不明显时候，既可以使用div也可以使用p的时候，尽可能的使用p，因为p在默认情况下有上下间距，对兼容特殊终端有利。
* 不要使用纯样式标签，如b、font、u，改用css设置。
* 表单域要用fieldset包裹起来，并用legend标签说明其用途。
* 每一个input标签的说明文本都需要使用label标签，并且input设置id和label的for关联起来

备注：strong、em、dfn、code、samp、kbd、var、cite，虽然这些标签定义的文本大多会呈现特殊的样式，但是实际上这些标签都拥有明确的含义。

我们并不反对使用他们，但是如果你只是为了达到某一个视觉效果而使用他们，建议使用css样式表完成。

1. **Html5有哪些新特性？如何处理html5新标签的浏览器兼容性问题？如何区分html和html5？**
2. 新特性

* 拖拽释放API
* 语义化更好的标签内容
* 音频audio、视频video
* 画布canvas API
* 地理API
* localStorage，浏览器关闭之后数据不丢失
* sessionStorage，浏览器关闭之后数据丢失
* 表单控件，date、time、calendar、email、url、search
* 新技术，webworker、websocket

1. Html5兼容性问题处理

可以通过js的方式（createElement）去手动创建一个。这种方式可以解决新标签在老版本IE浏览器中不兼容的问题。

生产环境中，直接引入html5shiv.js库即可。

1. 如何区分html和html5

文档类型声明，html5是<!DOCTYPE>，结构语义。

1. **Meta元素都有什么？**

Meta元数据。

元数据用来构建html文档的基本结构，以及就如何处理文档向浏览器提供信息和指示。

Title、link、base、meta都是元数据元素。

Meta元素可以提供有关页面的元信息，比如针对搜索引擎和更新频度的描述和关键词。

网站首页是权重最高的地方，如果首页链接太少，没有“桥”，“蜘蛛”不能继续往下爬到内页，直接影响网站的收录数量。但是首页链接也不能太多，一旦太多，没有实质性的链接，很容易影响用户体验，也会降低网站首页的权重，收录效果也不好。

1. 扁平化的目录层次

尽量让“蜘蛛”只要跳三次，就能到达网站内的任何一个内页。

1. 导航优化

导航应该尽量采用文字方式，也可以搭配图片导航，但是图片代码一定要进行优化，<img>标签必须添加alt和title属性，告诉搜索引擎导航的定位，做到即使图片未能显示，用户也能看到提示文字。

其次，在每一个网页上应该加上面包屑导航，好处：从用户体验方面来说，可以让用户了解当前所处的位置以及当前页面在整个网站的位置，帮助用户很快了解我咋会呢的组织形式，从而形成更好的位置感，同事提供了返回各个页面的接口，方便用户操作；对“蜘蛛”而言，能够清楚的了解网站的结构，同时还增加了大量的内部链接，方便抓取，降低跳出率。

1. 网站的结构布局

页面头部：logo、主导航、用户信息。

页面底部：版权信息和友情链接。

分页导航推荐写法：“首页 1 2 3 4 5 下拉框”，这样“蜘蛛”可以根据相应页码直接跳转，下拉框直接选择页面跳转。不推荐这种：“首页 下一页 尾页”，这样当分页数量很多的时候，“蜘蛛”需要经过多次往下爬，才能抓取，会很累、会容易放弃。

1. 利用布局，把重要的内容html代码放在最前

搜索引擎抓取html内容是从上到下，利用这一特点，可以让主要代码优先读取，广告等不重要的代码放在下边。

1. 控制页面的大小，减少http请求，提高网站的加载速度

一个页面最好不要超过100k，太大，页面加载速度会很慢。当速度很慢的时候，用户体验不好，留不住访客，并且一旦超时，“蜘蛛”也会离开。

* 网页代码优化

1. 突出重要内容

合理的设计title、description、keywords

Title：标题，只强调重点即可，尽量把重要的关键词放在前面，关键词不要重复出现，尽量做到每个页面的title标题中不要设置相同的内容。

Keywords：关键词，列举出几个页面的重要关键词即可，切记过分堆砌。

Description：网页描述，需要高度概括网页内容，切记不能过长，过分堆砌关键词，每个页面也要有所不同。

1. 语义化书写html代码，符合w3c标准
2. <a>标签

利用document.createDocumentFragment()方法创建文档碎片节点，创建的是一个虚拟的节点对象。向这个节点中添加dom节点，修改dom节点并不会影响到真实dom节点。与虚拟dom类似，同样达到了不频繁修改dom而导致的重排和重绘的过程。

* 利用innerHtml代替高频的appendChild
* 使用requestAnimationFrame()，把任何导致重绘的操作放入其中
* 虚拟Dom

利用虚拟dom，将dom抽象为虚拟dom，在dom发生变化的时候先对虚拟dom进行操作，通过dom diff算法将虚拟dom和原虚拟dom进行对比，最终批量修改真实的dom结构，尽可能的避免了频繁的修改dom而导致的重绘和重排。

1. **Script的async和defer的区别？**

浏览器在执行html的时候如果遇到<script>时会停止页面的渲染，去下载和执行js文件，知道遇到</script>的时候才会继续渲染页面。故浏览器在执行js文件的时候浏览器表现为一片空白，为了解决这个问题，ecmascript定义了defer和async两个属性用于控制js的下载和执行。

Defer

这个属性的用途表明脚本在执行的时候不会影响页面的构造。也就是说，脚本会被延迟到整个页面都解析完毕后在执行。

因此，在<script>元素设置defer属性，相当于告诉浏览器立即下载，但延迟执行。

Html5规范要求脚本按照它们出现的顺序先后执行，因此第一个延迟脚本会先于第二个延迟脚本执行，而这两个脚本会先于DOMContentLoaded事件执行。

在现实中，延迟脚本并不一定会按照顺序执行，也不一定会在DOMContentLoaded事件触发前执行，因此最好只包含一个延迟脚本。

MDN：如果缺少src属性（即内嵌脚本），该属性不应被使用，因为这种情况下它不起作用。

Async

与defer类似，都用于改变处理脚本的行为。

Async只适用于外部脚本，并告诉浏览器立即下载文件。但是与defer不同的是，标记为async的脚本并不保证按照它们的先后顺序执行。

第二个脚本文件可能会在第一个脚本文件之前执行，因此确保二者之间互不依赖非常重要。指定async属性的目的是不让页面等待两个脚本的下载和执行，从而异步加载页面其它内容。

MDN：该属性对没有src的script无效。

区别

相同点

* 加载文件时不阻塞页面渲染
* 对于inline的script无效，当script中间有代码时，两个属性都不会起作用
* 使用这两个属性的脚本中不能调用document.write方法