京城一灯-HTML

目录


每日一题 Html Css JavaScript React Vue Node 工程化 网络&安全 算法 编程题 其它
请说明 Html 布局元素的分类有哪些？并描述每种布局元素的应用场景
说一下减少 dom 数量的办法？一次性给你大量的 dom 怎么优化？
Html5 有哪些新特性？如何处理 Html5 新标签的浏览器兼容问题？如何区分 Html 和 Html5？
html 标签 b 和 strong 的区别
网站 SEO 怎么处理
a 标签默认事件禁掉之后做了什么才实现了跳转
meta 元素都有什么
script 的 async 跟 defer 的区别？
知道语义化吗？说说你理解的语义化，如果是你，平时会怎么做来保证语义化？说说你了解的 HTML5 语义化标签？

1

请说明 Html 布局元素的分类有哪些？并描述每种布局元素的应用场景

一、分类

二、应用场景

请说明 Html 布局元素的分类有哪些？并描述每种布局元素的应用场景

一、分类

1) 内联元素:

span,a,b,strong,i,em,br,input ,textarea

本身属性为 display:inline ;

和其他行内元素从左到右在一行显示,不可以直接控制宽度、高度等其他相关css属性，但是可以直接设置内外边距的左右值

宽高是由本身内容大小决定的（文字、图片等）

只能容纳文本或者其他行内元素，不能嵌套块级元素

2) 块状元素

div,h1-h6,hr,menu,ol,ul,li,dl,table,p,from

本身属性为 display:block ;

独占一行，每一个块级元素都会从新的一行重新开始，从上到下排布 可以直接控制宽度、高度等其他相关css属性，例如（padding系列，margin系列）

在不设置宽度的情况下，块级元素的宽度是它父级元素内容的宽度

在不设置高度的情况下，块级元素的高度是它本身内容的高度

3) 内联块状元素

内联块状元素综合了前两种的特性却又各有取舍。

不自动换行

能够识别 width 和 height , line-height , padding , marign

默认排列方式为从左到右

二、应用场景

- 内联元素：用于不指定宽高，宽高由内容指定；
- 块状元素：用于指定宽高，标签占满一行；
- 内联块状元素：用于指定元素宽高，不占满一行

2

说一下减少 dom 数量的办法？一次性给你大量的 dom 怎么优化？

一、减少DOM数量的方法

二、大量DOM时的优化

1.缓存Dom对象

2.文档片段

3.用innerHtml 代替高频的appendChild

4.最优的layout方案

5.虚拟Dom

说一下减少 dom 数量的办法？一次性给你大量的 dom 怎么优化？

一、减少DOM数量的方法

1. 可以使用伪元素，阴影实现的内容尽量不使用DOM实现，如清除浮动、样式实现等；
2. 按需加载，减少不必要的渲染；
3. 结构合理，语义化标签；

二、大量DOM时的优化

当对Dom元素进行一系列操作时，对Dom进行访问和修改Dom引起的重绘和重排都比较消耗性能，所以关于操作Dom,应该从以下几点出发：

1.缓存Dom对象

首先不管在什么场景下，操作Dom一般首先会去访问Dom，尤其是像循环遍历这种时间复杂度可能会比较高的操作。那么可以在循环之前就将主节点，不必循环的Dom节点先获取到，那么在循环里就可以直接引用，而不必去重新查询。

```
let rootElem = document.querySelector('#app');
let childList = rootElem.child; // 假设全是dom节点
for(let i = 0;i<childList.len;i++){
    /**
    * 根据条件对应操作
    */
}
```

2.文档片段

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。向这个节点添加dom节点，修改dom节点并不会影响到真实的dom结构。

我们可以利用这一点先将我们需要修改的dom一并修改完，保存至文档碎片中，然后用文档碎片一次性的替换真是的dom节点。与虚拟dom类似，同样达到了不频繁修改dom而导致的重排跟重绘的过程。

2.文档片段

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。向这个节点添加dom节点，修改dom节点并不会影响到真实的dom结构。

我们可以利用这一点先将我们需要修改的dom一并修改完，保存至文档碎片中，然后用文档碎片一次性的替换真是的dom节点。与虚拟dom类似，同样达到了不频繁修改dom而导致的重排跟重绘的过程。

```
let fragment = document.createDocumentFragment();
const operationDomHandle = (fragment) =>{
    // 操作
}
operationDomHandle(fragment);
// 然后最后再替换
rootElem.replaceChild(fragment,oldDom);
```

这样就只会触发一次回流，效率会得到很大的提升。如果需要对元素进行复杂的操作（删减、添加子节点），那么我们应当先将元素从页面中移除，然后再对其进行操作，或者将其复制一个（cloneNode()），在内存中进行操作后再替换原来的节点。

```
var clone=old.cloneNode(true);
operationDomHandle(clone);
rootElem.replaceChild(clone,oldDom)
```

3.用innerHtml 代替高频的appendChild

4.最优的layout方案

批量读，一次性写。先对一个不在render tree上的节点进行操作，再把这个节点添加回render tree。这样只会触发一次DOM操作。 使用 `requestAnimationFrame()` ，把任何导致重绘的操作放入 `requestAnimationFrame`

5.虚拟Dom

js模拟DOM树并对DOM树操作的一种技术。virtual DOM是一个纯js对象（字符串对象），所以对他操作会高效。

利用virtual 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的结构做对比，最终批量的去修改真实的dom结构，尽可能的避免了频繁修改dom而导致的频繁的重排和重绘。

3

Html5 有哪些新特性？如何处理 Html5 新标签的浏览器兼容问题？如何区分 Html 和 Html5？

一、Html5新特性

1. 拖拽释放(Drag and drop) API
2. 语义化更好的内容标签（header,nav,footer,aside,article,section）
3. 音频、视频API(audio,video)
4. 画布(Canvas) API
5. 地理(Geolocation) API
6. 本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失；
7. sessionStorage 的数据在浏览器关闭后自动删除
8. 表单控件，calendar、date、time、email、url、search
9. 新的技术webworker, websocket, Geolocation

二、Html5兼容问题处理

1.使用DOM操作来添加这些标签

既然浏览器不支持，自己来创建一个

```
<!DOCTYPE html>
 <html>
<head lang="en">
     <meta charset="UTF-8">
    <title>测试H5新标签兼容性</title>
     <script>
         document.createElement('header');
         document.createElement('footer');
     </script>
     <style>
        header, footer{display: block; width:50px; height: 50px; background-color:red;}
    </style>
 </head>
 <body>
 <header id="header">header</header>
 <footer id="footer">footer</footer>
```

1.使用DOM操作来添加这些标签

既然浏览器不支持，自己来创建一个

```
<!DOCTYPE html>
 <html>
<head lang="en">
     <meta charset="UTF-8">
    <title>测试H5新标签兼容性</title>
     <script>
         document.createElement('header');
         document.createElement('footer');
     </script>
     <style>
        header, footer{display: block; width:50px; height: 50px; background-color:red;}
    </style>
 </head>
 <body>
 <header id="header">header</header>
 <footer id="footer">footer</footer>

</body>
</html>
```

通过这种方法可以解决H5新标签在老IE浏览器中的兼容问题。但是，这样有个问题，那么多的新标签，如果每个都要通过这种方法去生产的话，是不是太麻烦了呢?

2.封装好的js库 --- html5shiv.js

```
<!-- 引入即可 -->
<script src="js/html5shiv.js"></script>
```

三、如何区分Html 和 Html5

1.文档类型声明

- Html声明： `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`
- Html5声明： `<!doctype html>`

2.结构语义

- Html:没有体现结构语义化的标签，通常都是这样来命名的 `<div id="header"></div>` ，这样表示网站的头部。
- Html5:在语义上却有很大的优势，提供了一些新的HTML5标签比如: `article`、`footer`、`header`、`nav`、`section` ，这些通俗易懂。

html 标签 b 和 strong 的区别

一、b 和 strong 的区别

两者虽然在网页中显示效果一样，但实际目的不同。

- `<b>` 这个标签对应 bold，即文本加粗，其目的仅仅是为了加粗显示文本，是一种样式 / 风格需求；
- `<strong>` 这个标签意思是加强，表示该文本比较重要，提醒读者 / 终端注意。为了达到这个目的，浏览器等终端将其加粗显示；
- `<b>` 为了加粗而加粗，`<strong>` 为了标明重点而加粗。

最重要的区别的就是样式标签与语义化标签的区别。最容易理解的场景就是盲人朋友使用阅读设备阅读网络时：`<strong>` 会重读，`<b>` 不会

二、扩展：Html语义化

很多时候我们写HTML，为了方便都会直接使用div和span标签，再通过class来确定具体样式。网站哪一部分为标题，哪一部分为导航，哪一部分为头部和底部，都只能通过class进行确定。但class命名规范却又没有一套统一的标准，因此导致很多时候无法确定整体网站的结构。

因此，在HTML5出现后，添加了关于页面布局结构的新标签。而在HTML书写过程中，根据不同的内容使用合适的标签进行开发，即为语义化。

在编程中，语义指的是一段代码的含义（这个 HTML 的元素有什么作用，扮演了什么样的角色）。HTML 语义元素清楚地向浏览器和开发者描述其意义，例如 `<form>` 、 `<table>` 以及 `<img>` 等。

2.1 优点

对搜索引擎友好，有了良好的结构和语义，网页内容自然容易被搜索引擎抓取。

2.2 Html5新增语义元素

- `<article>`
- `<aside>`
- `<details>`
- `<figcaption>`
- `<figure>`
- `<footer>`
- `<header>`
- `<main>`
- `<mark>`
- `<nav>`
- `<section>`

2.2 Html5新增语义元素

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>

2.3 为什么要语义化

语义化的优势主要在于下面几点：

1.其他开发者便于阅读代码，通过不同标签明白每个模块的作用何区别； 2.结构明确、语义清晰的页面能有更好的用户体验，在样式（css）没有加载前也有较为明确的结构，更如img这一类的，在图片无法加载的情况下有alt标签告知用户此处图片的具体内容； 3.利于SEO，语义化便于搜索引擎爬虫理解，和搜索引擎建立良好的沟通，能让爬虫爬去更多关键有效的信息； 4.方便其他设备阅读（如屏幕阅读器，盲人设备和移动设备等）。

2.4 如何语义化

一般的网站分为头部、导航、文章（或其他模块）、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

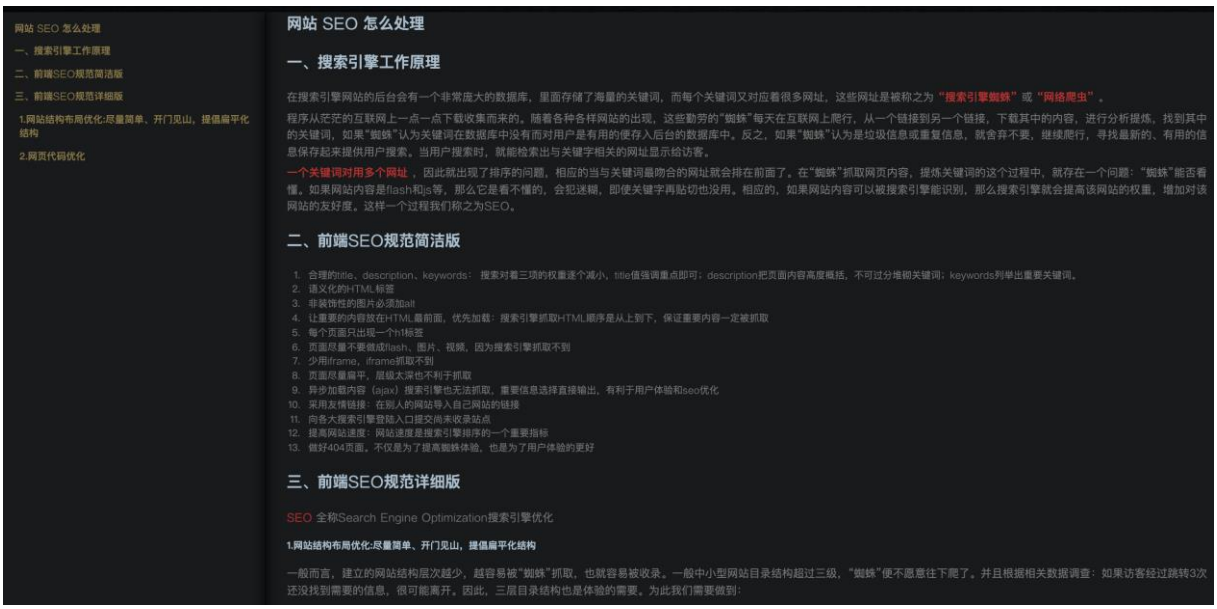
表示具体元素的作用或者意义的标签： a、abbr、address、audio、blockquote、caption、code、datalist、del、detail、ol、ul、figure、figuration、img、input、mark、p 等

- 尽可能少的使用无语义的标签div和span；
- 在语义不明显时，既可以使用div或者p时，尽量用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等，改用css设置。
- 需要强调的文本，可以包含在strong或者em标签中（浏览器预设样式，能用CSS指定就不用他们），strong默认样式是加粗（不要用b），em是斜体（不用i）；
- 使用表格时，标题要用caption，表头用thead，主体部分用tbody包围，尾部用tfoot包围。表头和一般单元格要区分开，表头用th，单元格用td；
- 表单域要用fieldset标签包起来，并用legend标签说明表单的用途；
- 每个input标签对应的说明文本都需要使用label标签，并且通过为~ input设置id属性，在lable标签中设置for=someId来让说明文本和相对应的input关联起来。

2.5 注意

em，strong，dfn，code，samp，kbd，var，cite 等，虽然这些标签定义的文本大多会呈现出特殊的样式，但实际上，这些标签都拥有确切的语义。我们并不反对使用它们，但是如果您只是为了达到某种视觉效果而使用这些标签的话，我们建议您使用样式表，那么做会达到更加丰富的效果。

4



网站 SEO 怎么处理

一、搜索引擎工作原理

二、前端SEO规范简洁版

三、前端SEO规范详细版

1.网站结构布局优化:尽量简单、开门见山，提倡扁平化结构

2.网页代码优化

网站 SEO 怎么处理

一、搜索引擎工作原理

在搜索引擎网站的后台会有一个非常庞大的数据库，里面存储了海量的关键词，而每个关键词又对应着很多网址，这些网址是被称之为“搜索引擎蜘蛛”或“网络爬虫”。

程序从茫茫的互联网上一点一点下载收集而来的。随着各种各样网站的出现，这些勤劳的“蜘蛛”每天在互联网上爬行，从一个链接到另一个链接，下载其中的内容，进行分析提炼。找到其中的关键词，如果“蜘蛛”认为关键词在数据库中没有而对用户是有用的便存入后台的数据库中。反之，如果“蜘蛛”认为是垃圾信息或重复信息，就舍弃不要，继续爬行，寻找最新的、有用的信息保存起来提供用户搜索。当用户搜索时，就能检索出与关键字相关的网址显示给访客。

一个关键词对用多个网址，因此就出现了排序的问题，相应的当与关键词最吻合的网址就会排在前面了。在“蜘蛛”抓取网页内容，提炼关键词的这个过程中，就存在一个问题：“蜘蛛”能否看懂。如果网站内容是flash和js等，那么它是看不懂的，会犯迷糊，即便关键字再贴切也没用。相应的，如果网站内容可以被搜索引擎能识别，那么搜索引擎就会提高该网站的权重，增加对该网站的友好度。这样一个过程我们称之为SEO。

二、前端SEO规范简洁版

1. 合理的title、description、keywords： 搜索对着三项的权重逐个减小，title值强调重点即可；description把页面内容高度概括，不可过分堆砌关键词；keywords列举出重要关键词。
2. 语义化的HTML标签
3. 非装饰性的图片必须加alt
4. 让重要的内容放在HTML最前面，优先加载：搜索引擎抓取HTML顺序是从上到下，保证重要内容一定被抓取
5. 每个页面只出现一个h1标签
6. 页面尽量不要做成flash、图片、视频，因为搜索引擎抓取不到
7. 少用iframe，iframe抓取不到
8. 页面尽量扁平，层级太深也不利于抓取
9. 异步加载内容（ajax）搜索引擎也无法抓取，重要信息选择直接输出，有利于用户体验和seo优化
10. 采用友情链接：在别人的网站导入自己网站的链接
11. 向各大搜索引擎登陆入口提交尚未收录站点
12. 提高网站速度：网站速度是搜索引擎排序的一个重要指标
13. 做好404页面，不仅是为了提高蜘蛛体验，也是为了用户体验的更好

三、前端SEO规范详细版

SEO 全称Search Engine Optimization搜索引擎优化

1.网站结构布局优化:尽量简单、开门见山，提倡扁平化结构

一般而言，建立的网站结构层次越少，越容易被“蜘蛛”抓取，也就容易被收录。一般中小型网站目录结构超过三级，“蜘蛛”便不愿意往下爬了。并且根据相关数据调查：如果访客经过跳转3次还没找到需要的信息，很可能离开。因此，三层目录结构也是体验的需要。为此我们需要做到：

三、前端SEO规范详细版

SEO 全称Search Engine Optimization搜索引擎优化

1.网站结构布局优化:尽量简单、开门见山，提倡扁平化结构

一般而言，建立的网站结构层次越少，越容易被“蜘蛛”抓取，也就容易被收录。一般中小型网站目录结构超过三级，“蜘蛛”便不愿意往下爬了。并且根据相关数据调查：如果访客经过跳转3次还没找到需要的信息，很可能离开。因此，三层目录结构也是体验的需要。为此我们需要做到：

1）控制首页链接数量

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。

2）扁平化的目录层次

尽量让“蜘蛛”只要跳转3次，就能到达网站内的任何一个内页。

3）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，做到即使图片未能正常显示时，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中的位置，帮助用户很快了解网站组织形式，从而形成更好的位置感，同时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结构，同时还增加了大量的内部链接，方便抓取，降低跳出率。

4）网站的结构布局----不可忽略的细节

页面头部: logo及主导航，以及用户的信息。页面主体：左边正文，包括面包屑导航及正文；右边放热门文章及相关文章，好处：留住访客，让访客多停留，对“蜘蛛”而言，这些文章属于相关链接，增强了页面相关性，也能增强页面的权重。

页面底部 版权信息和友情链接。

特别注意: 分页导航写法，推荐写法：“首页 1 2 3 4 5 6 7 8 9 下拉框”，这样“蜘蛛”能够根据相应页码直接跳转，下拉框直接选择页面跳转。而下面的写法是不推荐的，“首页 下一页 尾页”，特别是当分页数量特别多时，“蜘蛛”需要经过很多次往下爬，才能抓取，会很累、会容易放弃。

5）利用布局，把重要内容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，广告等不重要代码放在下边。例如，在左栏和右栏的代码不变的情况下，只需改一下样式，利用float:left;和float:right;就可以随意让两栏在展现上位置互换，这样就可以保证重要代码在最前，让爬虫最先抓取。同样也适用于多栏的情况。

6）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，页面加载速度慢。当速度很慢时，用户体验不好，留不住访客，并且一旦超时，“蜘蛛”也会离开。

2.网页代码优化

2.网页代码优化

1）突出重要内容

合理的设计 `title`、`description` 和 `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。
- `<meta keywords>` 标签：关键词，列举出几个页面的重要关键字即可，切记过分堆砌。
- `<meta description>` 标签：网页描述，需要高度概括网页内容，切记不能太长，过分堆砌关键词，每个页面也要有所不同。

2）语义化书写HTML代码，符合W3C标准

尽量让代码语义化，在适当的位置使用适当的标签，用正确的标签做正确的事。让阅读源码者和“蜘蛛”都一目了然。比如：h1-h6 是用于标题类的，`<nav>` 标签是用来设置页面主导航，列表形式的代码使用ul或ol，重要的文字使用strong等。

3）`<a>` 标签

页内链接，要加 `title` 属性加以说明，让访客和“蜘蛛”知道。而外部链接，链接到其他网站的，则需要加上 `el="nofollow` 属性，告诉“蜘蛛”不要爬，因为一旦“蜘蛛”爬了外部链接之后，就不会再回来了。

4）正文标题要用

`<h1>` 标签：h1标签自带权重“蜘蛛”认为它最重要，一个页面有且最多只能有一个H1标签，放在该页面最重要的标题上面，如首页的logo上可以加H1标签。副标题用 `<h2>` 标签，而其它地方不应该随便乱用 h 标题标签。

5. `<img>` 应使用 "alt" 属性加以说明

当网络速度很慢，或者图片地址失效的时候，就可以体现出alt属性的作用，他可以让用户在图片没有显示的时候知道这个图片的作用。同时为图片设置高度和宽度，可提高页面的加载速度。

6）表格应该使用 `<caption>` 表格标题标签

caption 元素定义表格标题。caption 标签必须紧随 table 标签之后

7）`<strong>`、`<em>` 标签

需要强调是使用。

`<strong>` 标签在搜索引擎中能够得到高度的重视，它能突出关键词，表现重要的内容，`<em>` 标签强调效果仅次于 `<strong>` 标签；`<b>`、`<i>` 标签：只是用于显示效果时使用，在SEO中不会起任何效果。

8）重要内容不要用JS输出

因为“蜘蛛”不会读取JS里的内容，所以重要内容必须放在HTML里。前端框架针对SEO的缺陷，可通过服务端渲染弥补

9）尽量少使用iframe框架

因为“蜘蛛”一般不会读取其中的内容。

10）搜索引擎会过滤掉display:none其中的内容

10）搜索引擎会过滤掉display:none其中的内容

11）蜘蛛只能抓取a标签中href

```
<a href="Default.aspx?id=1">测试</a>
```

最好后面不要带参数

```
<a href="Default.aspx">测试</a>
```

如果带上参数 蜘蛛不会考虑的。这样的话，就需要用到URL重写了。

12）蜘蛛不会执行JavaScript

换句话说 如果在a标签中使用了onclick 蜘蛛是不会抓到的。

13）蜘蛛只能抓到get请求的页面，不会抓到post请求的页面

14）我们希望网页的前台页面全部被蜘蛛抓到

但是不希望后台页面被蜘蛛抓到，蜘蛛可没有那么智能，知道你的网站哪个是前台页面，哪个是后台页面。

这里就需要创建一个名为“robots.txt”（注意robots.txt是一个协议，不是命令，一般最好要遵守的robots.txt是搜索引擎搜索该网站时的第一个文件。

5

6

2.1 用途

meta 元素除了 charset 属性外，都是http-equiv属性或name属性结合content来使用

2.1.1 name 指定名/值对定义元数据

```
<meta name="参数" content="具体描述信息">
```

name属性与content属性结合使用，name用来表示元数据的类型，表示当前meta 标签的具体作用；content属性用来提供值。

```
 <head>
      <title>示例</title>
      <meta name="keywords" content="描述网站内容的关键词，以逗号隔开，用于SEO搜索">
      <meta name="application name      " content="当前页所属Web应用系统的名称">
      <meta name="description" content="当前页的说明">
      <meta name="author" content="当前页的作者名">
      <meta name="copyright" content="版权信息">
      <meta name="renderer" content="renderer是为双核浏览器准备的，用于指定双核浏览器默认以何种方式渲染页面">
      <meta name="viewreport" content="它提供有关视口初始大小的提示，仅供移动设备使用">
</head>
```

2.1.2 charset 声明字符编码

charset属性为HTML5新增的属性，用于声明字符编码,有两种写法

```
<!-- HTML5 推荐方式 -->
<meta charset="utf-8">
 <!-- 旧的HTML -->
<meta http-equiv="content-Type" content="text/html;charset=utf-8">
```

> 在理论上，可以使用任何字符编码，但并不是所有浏览器都能够理解它们。

2.1.3 http-equiv 模拟http标头字段

http-equiv属性与content属性结合使用，http-equiv属性为指定所要模拟的标头字段的名称，content属性用来提供值。

```
<meta http-equiv="参数" content="具体的描述">
<!-- content-Type 声明网页字符编码 -->
<meta http-equiv="content-Type" content="text/html charset=UTF-8">
<!-- refresh 指定一个时间间隔(以秒为单位)，在此时间过去之后从服务器重新载入当前页面，也可以另外指定一个页面。 -->
<!-- 2秒后在当前页跳转到百度 -->
<meta http-equiv="refresh" content="2;URL=http://www.baidu.com">
<!-- X-UA-Compatible 浏览器采取何种版本渲染当前页面 -->
<!-- 指定IE和Chrome使用最新版本渲染当前页面 -->
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<!-- catch-control 用于指定所有缓存机制在整个请求/响应链中必须服从的指令 -->
<meta http-equiv="cache-control" content="no-cache">
```

2.2常用meta属性

```
  <!-- 声明文档使用的字符编码 -->
  <meta charset='utf-8'>
  <!-- 优先使用 IE 最新版本和 Chrome -->
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
  <!-- 页面描述 -->
  <meta name="description" content="不超过150个字符" />
  <!-- 页面关键词 -->
  <meta name="keywords" content="" />
  <!-- 网页作者 -->
  <meta name="author" content="name, email@gmail.com" />
  <!-- 搜索引擎抓取 -->
  <meta name="robots" content="index,follow" />
  <!-- 为移动设备添加 viewport -->
  <meta name="viewport" content="initial-scale=1, maximum-scale=3, minimum-scale=1, user-scalable=no">
  <!-- `width=device-width` 会导致 iPhone 5 添加到主屏后以 WebApp 全屏模式打开页面时出现黑边 http://bigc.at/ios-webapp-viewport-meta.orz -->

  <!-- iOS 设备 begin -->
  <meta name="apple-mobile-web-app-title" content="标题">
```

```
    <!-- iOS 设备 begin -->
    <meta name="apple-mobile-web-app-title" content="标题">
    <!-- 添加到主屏后的标题（iOS 6 新增） -->
    <meta name="apple-mobile-web-app-capable" content="yes"/>
    <!-- 是否启用 WebApp 全屏模式，删除苹果默认的工具栏和菜单栏 -->

    <meta name="apple-itunes-app" content="app-id=myAppStoreID, affiliate-data=myAffiliateData, app-argument=myURL">
    <!-- 添加智能 App 广告条 Smart App Banner（iOS 6+ Safari） -->
    <meta name="apple-mobile-web-app-status-bar-style" content="black"/>
    <!-- 设置苹果工具栏颜色 -->
    <meta name="format-detection" content="telphone=no, email=no"/>
    <!-- 忽略页面中的数字识别为电话，忽略email识别 -->
    <!-- 启用360浏览器的极速模式(webkit) -->
    <meta name="renderer" content="webkit">
    <!-- 避免IE使用兼容模式 -->
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <!-- 不让百度转码 -->
    <meta http-equiv="Cache-Control" content="no-siteapp" />
    <!-- 针对手持设备优化，主要是针对一些老的不识别viewport的浏览器，比如黑莓 -->
    <meta name="HandheldFriendly" content="true">
    <!-- 微软的老式浏览器 -->
    <meta name="MobileOptimized" content="320">
    <!-- uc强制竖屏 -->
    <meta name="screen-orientation" content="portrait">
    <!-- QQ强制竖屏 -->
    <meta name="x5-orientation" content="portrait">
    <!-- UC强制全屏 -->
    <meta name="full-screen" content="yes">
    <!-- QQ强制全屏 -->
    <meta name="x5-fullscreen" content="true">
    <!-- UC应用模式 -->
    <meta name="browsermode" content="application">
    <!-- QQ应用模式 -->
    <meta name="x5-page-mode" content="app">
    <!-- windows phone 点击无高光 -->
    <meta name="msapplication-tap-highlight" content="no">
      <!-- iOS 设备 end -->
```

```
  <!-- iOS 设备 end -->
<meta name="msapplication-TileColor" content="#000"/>
<!-- Windows 8 磁贴颜色 -->
<meta name="msapplication-TileImage" content="icon.png"/>
<!-- Windows 8 磁贴图标 -->

<link rel="alternate" type="application/rss+xml" title="RSS" href="/rss.xml"/>
<!-- 添加 RSS 订阅 -->
<link rel="shortcut icon" type="image/ico" href="/favicon.ico"/>
<!-- 添加 favicon icon -->

<!-- sns 社交标签 begin -->
<!-- 参考微博API -->
<meta property="og:type" content="类型" />
<meta property="og:url" content="URL地址" />
<meta property="og:title" content="标题" />
<meta property="og:image" content="图片" />
<meta property="og:description" content="描述" />
<!-- sns 社交标签 end -->
```

7

script 的 async 跟 defer 的区别?

async & defer

1.1 defer

1.2 async

1.3 区别

script 的 async 跟 defer 的区别?

async & defer

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的文件直接遇见 `</scirpt>` 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

1.1 defer

红宝书中的解释：

这个属性的用途是表明脚本在执行时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再运行。因此，在 `<script>` 元素中设置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先后顺序执行，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。在现实当中，延迟脚本并不一定会按照顺序执行，也不一定会在DOMContentLoad事件触发前执行，因此最好只包含一个延迟脚本。

MDN上的解释：

defer，这个布尔属性被设定用来通知浏览器该脚本将在文档完成解析后，触发DOMContentLoaded事件前执行。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。

1.2 async

红宝书中的解释：

这个属性与defer类似，都用于改变处理脚本的行为。async只适用于外部脚本文件，并告诉浏览器立即下载文件。但与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。因此确保两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下载和执行，从而异步加载页面其他内容。

MDN上的解释：

async，该布尔属性指示浏览器是否在允许的情况下异步执行该脚本。该属性对于内联脚本无作用（即没有src属性的脚本）

1.3 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标签中间有代码时，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
- 有脚本的onload的事件回调

1.3 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标签中间有代码时，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
- 有脚本的onload的事件回调

不同点：

— html4.0中定义了defer；html5.0中定义了async — 浏览器支持不同 — 每一个async属性的脚本都在它下载结束之后立刻执行，同时会在window的load事件之前执行。所以就有可能出现脚本执行顺序被打乱的情况；每一个defer属性的脚本都是在页面解析完毕之后，按照原本的顺序执行，同时会在document的DOMContentLoaded之前执行。

- 当一个script标签内同时包含defer与async属性时,只会触发async，不会触发defer，除非浏览器不兼容async。

8

知道语义化吗？说说你理解的语义化，如果是你，平时会怎么做来保证语义化？说说你了解的 HTML5 语义化标签？

Html语义化

1.优点

2.Html5新增语义元素

3.为什么要语义化

4.如何语义化

5.注意

知道语义化吗？说说你理解的语义化，如果是你，平时会怎么做来保证语义化？说说你了解的 HTML5 语义化标签？

Html语义化

很多时候我们写HTML，为了方便都会直接使用div和span标签，再通过class来确定具体样式。网站哪一部分为标题，哪一部分为导航，哪一部分为头部和底部，都只能通过class进行确定。但class命名规范却又没有一套统一的标准，因此导致很多时候无法确定整体网站的结构。

因此，在HTML5出现后，添加了关于页面布局结构的新标签。而在HTML书写过程中，根据不同的内容使用合适的标签进行开发，即为语义化。

在编程中，语义指的是一段代码的含义（这个 HTML 的元素有什么作用，扮演了什么样的角色）。HTML 语义元素清楚地向浏览器和开发者描述其意义，例如 `<form>` 、 `<table>` 以及 `<img>` 等。

1.优点

对搜索引擎友好，有了良好的结构和语义，网页内容自然容易被搜索引擎抓取。

2.Html5新增语义元素

- `<article>`
- `<aside>`
- `<details>`
- `<figcaption>`
- `<figure>`
- `<footer>`
- `<header>`
- `<main>`
- `<mark>`
- `<nav>`
- `<section>`
- `<summary>`
- `<time>`

3.为什么要语义化

语义化的优势主要在于下面几点：

1.其他开发者便于阅读代码，通过不同标签明白每个模块的作用何区别； 2.结构明确、语义清晰的页面能有更好的用户体验，在样式（css）没有加载前也有较为明确的结构，更如img这一类的，在图片无法加载的情况下有alt标签告知用户此处图片的具体内容； 3.利于SEO，语义化便于搜索引擎爬虫理解，和搜索引擎建立良好的沟通，能让爬虫爬去更多关键有效的信息； 4.方便其他设备阅读（如屏幕阅读器，盲人设备和移动设备等）。

4.如何语义化

一般的网站分为头部、导航、文章（或其他模块）、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`

表示具体元素的作用或者意义的标签： `a`、`abbr`、`address`、`audio`、`blockquote`、`caption`、`code`、`datalist`、`del`、`detail`、`ol`、`ul`、`figure`、`figuration`、`img`、`input`、`mark`、`p` 等

- 尽可能少的使用无语义的标签div和span；
- 在语义不明显时，既可以使用div或者p时，尽量用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等，改用css设置。
- 需要强调的文本，可以包含在strong或者em标签中（浏览器预设样式，能用CSS指定就不用他们），strong默认样式是加粗（不要用b），em是斜体（不用i）；
- 使用表格时，标题要用caption，表头用thead，主体部分用tbody包围，尾部用tfoot包围。表头和一般单元格要区分开，表头用th，单元格用td；
- 表单域要用fieldset标签包起来，并用legend标签说明表单的用途；
- 每个input标签对应的说明文本都需要使用label标签，并且通过为 input设置id属性，在lable标签中设置for=someId来让说明文本和相对应的input关联起来。

5.注意

`em`，`strong`，`dfn`，`code`，`samp`，`kbd`，`var`，`cite` 等，虽然这些标签定义的文本大多会呈现出特殊的样式，但实际上，这些标签都拥有确切的语义。

我们并不反对使用它们，但是如果您只是为了达到某种视觉效果而使用这些标签的话，我们建议您使用样式表，那么做会达到更加丰富的效果。