目录

[1 HTML基础 2](#_Toc90564448)

[1.1 HTML语义化 2](#_Toc90564449)

[1.2浏览器乱码的原因是什么 如何解决 3](#_Toc90564450)

[1.3 HTML标准模式和混杂模式的区别 3](#_Toc90564451)

[1.4 HTML标准模式和严格模式的区别 4](#_Toc90564452)

[1.5 如何实现浏览器内多个标签页之间的通信 4](#_Toc90564453)

[1.6 前端页面有哪三层构成，分别是什么 作用是什么 4](#_Toc90564454)

[1.7 XML和JSON的区别 5](#_Toc90564455)

[1.7.1 什么是XML文件 5](#_Toc90564456)

[1.7.2 XML和JSON的区别 5](#_Toc90564457)

[1.8 移动html5开发流行的框架有哪些 6](#_Toc90564458)

[1.9 嵌入视频 6](#_Toc90564459)

[1.10 浏览器解析HTML全过程 6](#_Toc90564460)

[2 HTML关键字 7](#_Toc90564461)

[2.1 src和href的区别 7](#_Toc90564462)

[2.2 defer和async的区别 7](#_Toc90564463)

[2.3 常用的meta标签有哪些 7](#_Toc90564464)

[2.4 img的srcset属性的作用 8](#_Toc90564465)

[2.5 Canvas和SVG的区别 8](#_Toc90564466)

[2.6 meta viewport 是做什么用的 8](#_Toc90564467)

[2.7 defer\async的区别 8](#_Toc90564468)

[2.8 getElementById和getElementsByClassName 9](#_Toc90564469)

[2.9 meta viewport 原理是什么 9](#_Toc90564470)

[3 HTML优化 9](#_Toc90564471)

[3.1 懒加载 9](#_Toc90564472)

[3.2预加载 10](#_Toc90564473)

[3.3 页面有大量图片加载很慢，有哪些方法优化图片的加载 10](#_Toc90564474)

# 1 HTML基础

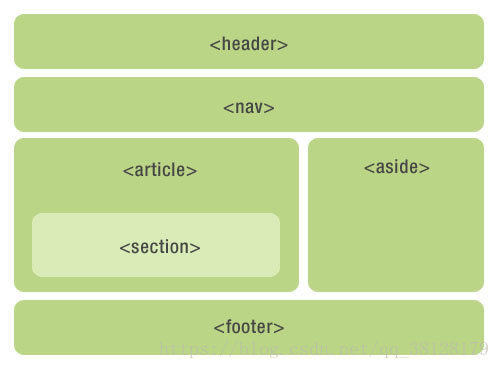
## HTML语义化

**语义化是指根据内容的结构化（内容语义化），选择合适的标签（代码语义化）**。通俗来讲就是用正确的标签做正确的事情。

语义化的优点如下：

* 对机器友好，带有语义的文字表现力丰富，更适合搜索引擎的爬虫爬取有效信息，有利于SEO。除此之外，语义类还支持读屏软件，根据文章可以自动生成目录；
* 对开发者友好，使用语义类标签增强了可读性，结构更加清晰，开发者能清晰的看出网页的结构，便于团队的开发与维护。

HTML常用的语义化标签如下图所示



**<header>元素**

用于定义页面的介绍展示区域，通常包括网站logo、主导航、全站链接以及搜索框。也适合对页面内部一组介绍性或导航性内容进行标记。

**<nav>元素**

定义页面的导航链接部分区域，不是所有的链接都需要包含在<nav>中，除了页脚再次显示顶级全局导航、或者包含招聘信息等重要链接。

**<main>元素**

定义页面的主要内容，一个页面只能使用一次。如果是web应用，则包围其主要功能。

**<article>元素**

定义页面独立的内容，它可以有自己的header、footer、sections等，专注于单个主题的博客文章，报纸文章或网页文章。article可以嵌套article，只要里面的article与外面的是部分与整体的关系。

**<aside>元素**

定义与主要内容相关的内容块。通常显示为侧边栏。

**<footer>元素**

定义文档的底部区域，通常包含文档的作者，著作权信息，链接的使用条款，联系信息等

## 1.2浏览器乱码的原因是什么 如何解决

原因：网页源代码和网页内容中的文字编码不匹配，例如网页源代码是GBK类型的而文字是utf-8类型的。

原因：网页源代码和请求后端后，从数据库返回的字符编码不匹配，例如网页源码是GBK类型的而数据库是UTF-8类型的。

解决方法：把编码置为统一的类型。

## 1.3 HTML标准模式和混杂模式的区别

**盒模型的处理差异**：标准CSS盒模型的宽度和高度等于内容区的高度和宽度，不包含内边距和边框，而IE6之前的浏览器实现的盒模型的宽高计算方式是包含内边距和边框的。因此，对于IE，怪异模式和标准模式下的盒模型宽高计算方式是不一样的；

**行内元素的垂直对齐**：很多早期的浏览器对齐图片至包含它们的盒子的下边框，虽然CSS的规范要求它们被对齐至盒内文本的基线。标准模式下，基于Gecko的浏览器将会对齐至基线，而在quirks模式下它们会对齐至底部。最直接的例子就是图片的显示。在标准模式下，图片并不是与父元素的下边框对齐的，如果仔细观察，你会发现图片与父元素下边框之间存在一点小空隙。那是因为标准模式下，图片是基线对齐的。而怪异模式下，则不存在这个问题。

## 1.4 HTML标准模式和严格模式的区别

**严格模式主要有以下限制：**

* 变量必须声明后再使用
* 函数的参数不能有同名属性，否则报错
* 不能使用with语句
* 不能对只读属性赋值，否则报错
* 不能使用前缀0表示八进制数，否则报错
* 不能删除不可删除的属性，否则报错
* 禁止this指向全局对象

## 1.5 如何实现浏览器内多个标签页之间的通信

**调用cookie+setInterval()**

将要传递的信息存储在cookie中，每隔一定时间读取getCookie获取信息，即可随时获取要传递的信息。

## 1.6 前端页面有哪三层构成，分别是什么 作用是什么

分成：结构层、表示层、行为层。

结构层（structural layer）由 HTML 或 XHTML 之类的标记语言负责创建。标签，也就是那些出现在尖括号里的单词，对网页内容的语义含义做出了描述，但这些标签不包含任何关于如何显示有关内容的信息。例如，P 标签表达了这样一种语义：“这是一个文本段。”

表示层（presentation layer）

由 CSS 负责创建。 CSS 对“如何显示有关内容”的问题做出了回答

行为层（behaviorlayer）

负责回答“内容应该如何对事件做出反应”这一问题。这是 Javascript 语言和 DOM 主宰的领域。

## 1.7 XML和JSON的区别

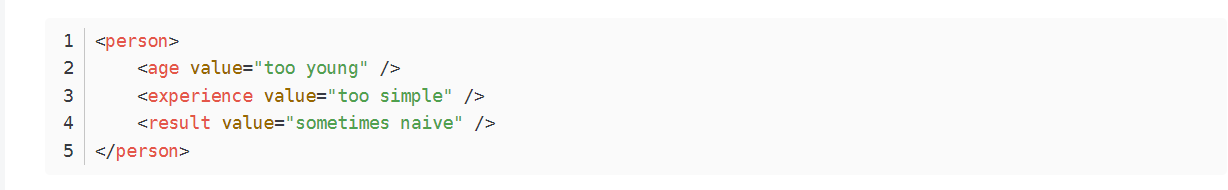
### 1.7.1 什么是XML文件

在XML语言中，它允许用户自定义标签。一个标签用于描述一段数据；一个标签可以分为开始标签和结束标签，在开始标签和结束标签之间，又可以使用其他标签描述其他数据，以此来实现数据关系的描述。

1.XML的出现解决了程序间数据传输的问题：   
比如QQ之间的数据传送，用XML格式来传送数据，具有良好的可读性，可维护性

2.XML可以做配置文件   
XML文件做配置文件可以说非常普遍，比如我们的Tomcat服务器的server.xml，web.xml。再比如我们的structs中的structs-config.xml文件，和hibernate的hibernate.cfg.xml等等。

3.XML可以充当小型的数据库   
XML文件可以做小型数据库，也是不错的选择，我们程序中可能用到一些经常要人工配置的数据，如果放在数据库中读取不合适（因为这会增加维护数据库的工作），则可以考虑直接用XML来做小型数据库。这种方式直接读取文件显然要比读数据库快。比如msn中保存用户聊天记录就是用XML文件。



上图就是一个典型的xm格式文件

### 1.7.2 XML和JSON的区别

(1).数据体积方面。

JSON相对于XML来讲，数据的体积小，传递的速度更快些。

(2).数据交互方面。

JSON与JavaScript的交互更加方便，更容易解析处理，更好的数据交互。

(3).数据描述方面。

JSON对数据的描述性比XML较差。

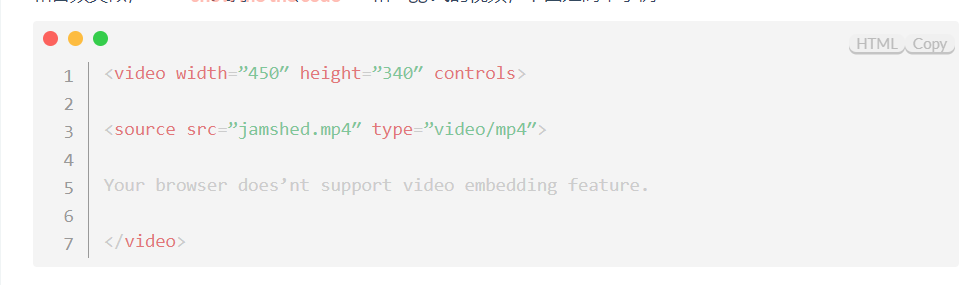
(4).传输速度方面。

JSON的速度要远远快于XML。

## 1.8 移动html5开发流行的框架有哪些

* jQueryMobile，jQueryMobile是jQuery在手机上和平板设备上的版本。jQueryMobile不仅会给主流移动平台带来jQuery核心库，而且会发布一个完整统一的jQuery移动UI框架。支持全球主流的移动平台。jQueryMobile开发团队说：能开发这个项目，非常兴奋。移动web太需要一个跨浏览器的框架，让开发人员开发出真正的移动web网站。
* jQTouch，jQTouch是一个jQuery的插件，主要用于手机上的Webkit浏览器上实现一些包括动画、列表导航、默认应用样式等各种常见UI效果的Java库。支持包括iPhone、Android等手机。

## 1.9 嵌入视频



## 1.10 浏览器解析HTML全过程

DOM文档通常加载的步骤：

1.解析HTML结构。

2.加载外部脚本和样式表文件

3.解析并执行脚本代码。

4.DOM树构建完成。//DOMContentLoaded

5.加载图片等外部资源。

6.页面加载完毕。//load

# 2 HTML关键字

## 2.1 src和href的区别

src和href都是**用来引用外部的资源**，它们的区别如下：

**src：** 表示对资源的引用，它指向的内容会嵌入到当前标签所在的位置。src会将其指向的资源下载并应⽤到⽂档内，如请求js脚本。当浏览器解析到该元素时，会暂停其他资源的下载和处理，直到将该资源加载、编译、执⾏完毕，所以⼀般js脚本会放在页面底部。

**href：** 表示超文本引用，它指向一些网络资源，建立和当前元素或本文档的链接关系。当浏览器识别到它他指向的⽂件时，就会并行下载资源，不会停⽌对当前⽂档的处理。 常用在a、link等标签上。

## 2.2 defer和async的区别

如果没有defer或async属性，浏览器会立即加载并执行相应的脚本。它不会等待后续加载的文档元素，读取到就会开始加载和执行，这样就阻塞了后续文档的加载。**defer 和 async属性都是去异步加载外部的JS脚本文件，它们都不会阻塞页面的解析**，其区别如下：

* **执行顺序：** 多个带async属性的标签，不能保证加载的顺序；多个带defer属性的标签，按照加载顺序执行；
* **脚本是否并行执行：async属性，表示后续文档的加载和执行与js脚本的加载和执行是并行进行的**，即异步执行；defer属性，加载后续文档的过程和js脚本的加载(此时仅加载不执行)是并行进行的(异步)，js脚本需要等到文档所有元素解析完成之后才执行，DOMContentLoaded事件触发执行之前。

## 2.3 常用的meta标签有哪些

meta 标签由 name 和 content 属性定义，**用来描述网页文档的属性**，比如网页的作者，网页描述，关键词等。

charset，用来描述HTML文档的编码类型：

<meta charset="UTF-8" >

 keywords，页面关键词：

<meta name="keywords" content="关键词" />

description，页面描述：

<meta name="description" content="页面描述内容" />

## 2.4 img的srcset属性的作用

响应式页面中经常用到根据屏幕密度设置不同的图片。这时就用到了 img 标签的srcset属性。srcset属性用于设置不同屏幕密度下，img 会自动加载不同的图片。用法如下：

<img src="image-128.png" srcset="image-256.png 2x" />

## 2.5 Canvas和SVG的区别

**SVG：** SVG可缩放矢量图形（Scalable Vector Graphics）是基于可扩展标记语言XML描述的2D图形的语言，SVG基于XML就意味着SVG DOM中的每个元素都是可用的，可以为某个元素附加Javascript事件处理器。

* 不依赖分辨率，可以动态改变大小，不会失真
* 支持事件处理器
* 最适合带有大型渲染区域的应用程序（比如谷歌地图）
* 复杂度高会减慢渲染速度（任何过度使用 DOM 的应用都不快）
* 不适合游戏应用

**Canvas：** Canvas是画布，通过Javascript来绘制2D图形，是逐像素进行渲染的。其位置发生改变，就会重新进行绘制。

* 依赖分辨率，不能动态改变大小
* 不支持事件处理器
* 弱的文本渲染能力
* 能够以 .png 或 .jpg 格式保存结果图像
* 最适合图像密集型的游戏，其中的许多对象会被频繁重绘

## 2.6 meta viewport 是做什么用的

是为了在移动端不让用户缩放页面使用的

## 2.7 **defer\async的区别**

首先都是让js文件能够异步下载，不阻塞页面的渲染 区别就是defer必须等待整个文档渲染完成后才执行 而async在下载完成后，会暂停html的解析，转去执行js。

## 2.8 getElementById和getElementsByClassName

区别：

Js可以通过ById获取innerText和innerHTML，但是ByClassName不可以

## 2.9 meta viewport 原理是什么

meta viewport 标签的作用是让当前 viewport 的宽度等于设备的宽度，同时不允许用户进行手动缩放。是为了在移动端不让用户缩放页面使用的。

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale-1, minimum-scale=1">

* with=device-width 将布局视窗（layout viewport）的宽度设置为设备屏幕分辨率的宽度
* initial-scale=1 页面初始缩放比例为屏幕分辨率的宽度
* maximum-scale=1 指定用户能够放大的最大比例
* minimum-scale=1 指定用户能够缩小的最大比例

# 3 HTML优化

## 3.1 懒加载

懒加载也就是延迟加载。

当访问一个页面的时候，先把img元素或是其他元素的背景图片路径替换成一张占位图的路径，这样就只需请求一次，只有当图片出现在浏览器的可视区域内时，才设置图片正真的路径，让图片显示出来。

**优点：很多页面，内容很丰富，页面很长，图片较多。比如瀑布流，图片数量多且大，如果一次性加载完毕，用户等待时间过长。 懒加载页面加载速度快、可以减轻服务器的压力、节约了流量，用户体验好。**

一张图片就是一个<img>标签，浏览器是否发起请求图片是根据<img>的src属性，所以实现懒加载的关键就是，在图片没有进入可视区域时，先不给<img>的src赋值，这样浏览器就不会发送请求了，等到图片进入可视区域再给src赋值。

## 3.2预加载

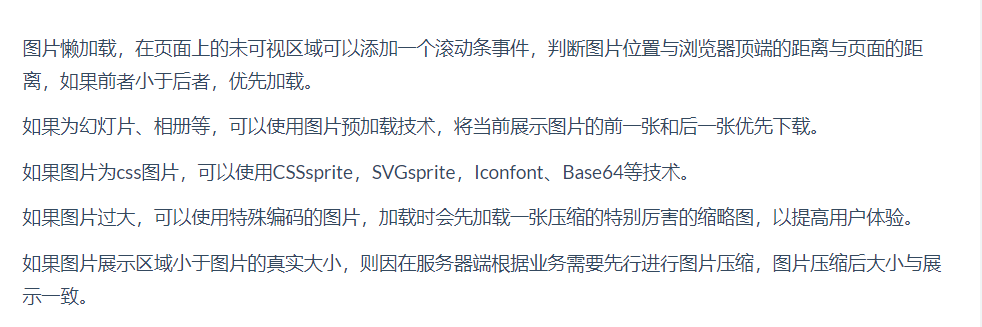
预加载也就是提前加载图片

具体表现为，当用户需要查看时可直接从本地缓存中渲染

一般用于移动端 用于图片比较多的情况 避免在项目中加载缓慢 影响用户体验 所以需要在进入项目前提前加载图片

做法是把所有图片的路径 放在一个数组中 然后使用循环遍历这个数组 每次创建一个图片对象 并把路径赋值给这个图片对象的src属性 这样就完成加载 其中使用 onload事件判断每个图片是否加载完成。

## 3.3 页面有大量图片加载很慢，有哪些方法优化图片的加载



## 3.4 常见的web优化方式

3.4.1 Web语义化优化

特点：增加代码的可读性，方便网页的爬取，简洁化DOM的标签，减少页面渲染的时间消耗。

语义化标签是浏览器内置就能解析识别的标签，最终的实现也会dom的结构简洁话，所以就会减少时间消耗哦

3.4.2 异步加载js

把不需要立刻加载的js异步加载，例如在script标签上增加async标签，实现异步加载。

JS不要写在head里，而是要卸载html的结尾处，因为html解析是自上而下解析的，所以html遇到js会暂停对DOM的解析转而去执行js，会降低页面加载速度。

3.4.3 CSSDOM优化

内部样式尽量写在head中（或者说成写在文件尽可能的上方）因为是文档是子上而下的解析，如果写在下方，或者中间，遇到js会阻塞dom和cssom操作，所以避免这个情况，应该规范在head中。

尽量少使用内联样式，原因如上

尽可能的不使用 @import 属性，import属性会在页面加载完成之后，或者使用到该css代码块的时候采取加载对应的文件，等效于把css写到的文档的底部，所以应该避免这种情况。