# 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）**。

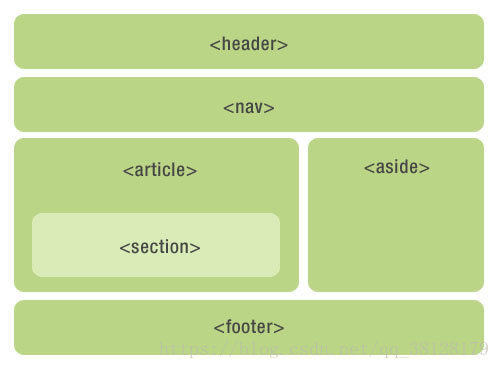
语义化的优点如下：

1.有利于SEO。

2.构建更为简洁清晰的DOM树。

3.增强了可读性，有利于开发与维护。

HTML常用的语义化标签如下图所示



**<header>元素**

定义页面的介绍展示区域，通常包括网站logo、主题、全站链接以及搜索框。

**<nav>元素**

定义页面的导航栏

**<main>元素**

定义页面的主要内容，一个页面只能使用一次。

**<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。

**<section>元素**

用来定义文档的某个区域，比如章节、页眉、页脚或者文档的其他区域。

**<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 HTML标准模式和混杂模式的区别

标准模式以浏览器支持的最高标准呈现页面。content-box

混杂模式向后兼容，使得低版本浏览器也能正常显示。border-box

二者主要在以下两个方面有差异：

**盒模型的处理差异**：

标准模式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

content-box: width=content

混杂模式的盒模型的宽高是包含内边距和边框的

border-box: width=content+padding+margin

**行内元素的垂直对齐**：

标准模式下对齐至基线

混杂模式下对齐至底部

## 1.3 如何实现浏览器内多个标签页之间的通信

**1.localStorage**

在一个标签页里面使用 localStorage.setItem(key,value)添加内容；

在另一个标签页里面监听 storage 事件。即可得到 localstorge 存储的值

localstorage永久有效，存储在磁盘中

**2.sessionstorage**

sessionstorage关闭浏览器后就失效，存储在内存中

**3.cookie**

在浏览器和服务器之前传递数据，有效期到达后失效

## 1.4 cookie和localstorage区别

1.cookie在浏览器与服务器之间来回传递，localStorage仅在本地保存

2.cookie仅在一段时间内有效，localStorage 永久有效

## 1.5 前端页面有哪三层构成，分别是什么 作用是什么

1:结构层：HTML创建页面元素

2.表示层：CSS为页面添加样式

3.行为层：JS为页面元素添加事件

## 1.6 XML和JSON的区别

### 1.6.1 什么是XML文件

在XML语言中，它允许用户自定义标签。通过标签层级的嵌套来描述数据关系

作用：1.程序间传输数据 2.做配置文件  3.充当小型的数据库

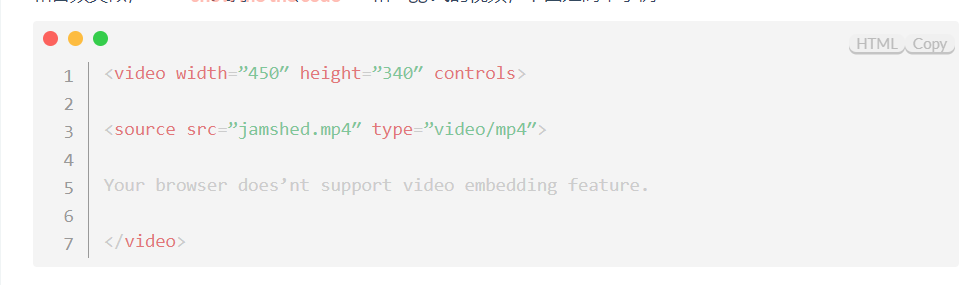
### 1.6.2 JSON

JSON是一种轻量级的数据交换格式。它基于 ECMAScript，独立于编程语言

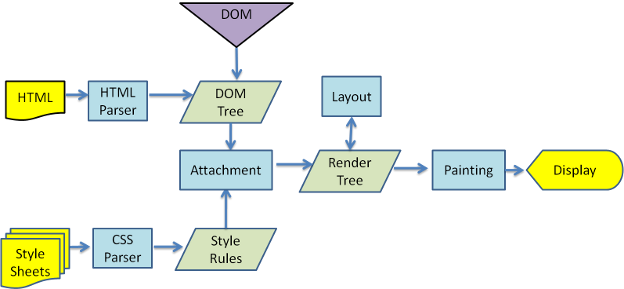
### 1.6.3 XML和JSON的区别

JSON体积小、传输快、与JS交互性较好，而XML对数据的描述更准确

## 1.7 嵌入视频



## 1.8 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

css文件不会阻止html的解析，但会阻止渲染树的生成

js文件会阻止html的解析

内联js脚本阻断html解析，执行内联js脚本，执行后继续html解析

外联普通js脚本：

IMG_256外联defer脚本：

IMG_256外联async脚本：

IMG_256

1. 解析并执行脚本代码。

js之前的css会阻塞js的执行，从而阻塞html的解析

1. DOM树构建完成。//DOMContentLoaded

正常情况下，js和其之前的css执行完毕，才会执行DOMContentLoaded

遇到html结束标签</html>，dom树才算解析完成

1. 加载图片等外部资源。
2. 页面加载完毕。//load

## 1.9 DOMContentLoaded和load的区别：

DOMContentLoaded：

HTML 解析完成之后出发DOMContentLoaded ，而无需等待图像等外部资源

Load：

所有资源全部加载完毕

# 2 HTML关键字

## 2.1 src和href的区别

src和href都表示对外部资源的引用，它们的区别如下：

**src：** 将外部资源下载并嵌入到当前标签所在的位置。常用于js，暂停html解析

**href：**建立外部资源和当前元素的链接关系。常用于css，不会暂停html解析

## 2.2 defer和async的区别

**defer 和 async属性都是去异步加载JS，加载过程中不会阻塞页面的解析**

区别如下：

**执行顺序：** defer按照加载顺序执行；async不能保证加载顺序

**是否阻止html解析：**async加载后立即执行，无论html是否解析完毕，可能造成html解析暂停。defer在html解析后、DOMContentLoad之前执行

## 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。例如：charset、keywords和description

## 2.4 Canvas和SVG的区别

**SVG：**可缩放矢量图形（Scalable Vector Graphics），基于XML

* 可以动态缩放，不会失真
* 支持事件处理器
* 渲染能力强
* 渲染速度慢

**Canvas：** HTML5新特性。Canvas是画布，通过Javascript来绘制2D图形

* 不能动态缩放
* 不支持事件处理器
* 渲染能力弱
* 渲染速度快

## 2.5 getElementById和getElementsByClassName

getElementById是通过id来获取元素，只能获取一个元素（id唯一）。

getElementsByClassName是通过类名来获取元素，可以获取多个元素

2.6 img的alt属性

1.无法显示图像时：显示替代文本

2.正常显示图像时：显示悬停文字

# 3 HTML优化

## 3.1 懒加载

又称延迟加载。图片没有进入可视区域时，先不给<img>的src赋值，减少发送请求。等到图片进入可视区域再给src赋值，请求真正的图片

**优点：加载速度快、减轻服务器的压力、减少用户等待时间**

## 3.2预加载

预加载也就是提前加载图片，当页面渲染时直接从本地缓存中读取

预加载是以增大服务器压力的方式，来换取更好的用户体验

## 3.3 常见的web优化方式

1 适用语义化标签

增加代码的可读性，有利于SEO，DOM更清晰，减少页面渲染时间

2 异步加载js

使用defer和async，减少对html解析的阻塞

3 js和css位置

css尽量写在head中，JS放在html的结尾处

css会构建cssom树，html会解析成dom树，二者合并成为渲染树，css放在前面可以让页面尽快的显示出内容，css不会阻止html解析，但会阻止html渲染。如果css放在后面，有两个问题：

1.html解析后才能解析css，造成时间浪费

2.会先渲染没有样式的页面，然后再渲染有样式的，页面闪烁

js放在结尾，两个原因：

1.**浏览器会先渲染js之前的部分页面内容，不至于造成长时间等待**

2.js放在后面，方便操作dom，放在前面dom还没有构建，获取不到dom

## 3.4 HTML5新特性

1.语义化标签，比如 article、footer、header、nav、section

2.Websocket

3.web storage:sessionstorage、localstorage

4.新的表单控件，比如 calendar、date、time、email、url、search

5.canvas

6.媒体元素audio和video

## 3.5浏览器的线程

JavaScript 单线程指的是浏览器中负责解释和执行 JavaScript 代码的只有一个线程，即为 JS引擎线程，但是浏览器的渲染进程是提供多个线程的，如下

1.渲染引擎线程：负责页面的渲染

2.JS引擎线程：负责JS的解析和执行

3.定时触发器线程：处理定时事件，比如setTimeout，setInterval

4.事件触发线程：处理DOM响应事件

5.异步http请求线程：处理http请求

三个常驻的线程：渲染线程、js引擎线程、事件触发线程

**渲染线程和js引擎线程不能同时运行，一个运行时，另一个被挂起，GUI更新会被保存在一个队列中等到JS引擎空闲时立即被执行。HTML和CSS的解析是渲染引擎线程，而js的解析则是js引擎线程，二者同一时刻只能有一个在工作**

浏览器端JavaScript是以单线程的方式执行的，也就是说JavaScript和UI渲染占用同一个主线程，那就意味着，如果JavaScript进行高负载的数据处理，UI渲染就很有可能被阻断，浏览器就会出现卡顿，降低了用户体验。

Ajax通过http请求线程得到结果后，将回调函数加入消息队列尾部，等到js引擎空闲，再去执行消息队列中的回调函数，因此，Ajax并不是真正的异步

setTimeout和setInterval也是一样

## 3.6 获取标签内的文字

let d=document.getElementById("a");

console.log(d.innerHTML);