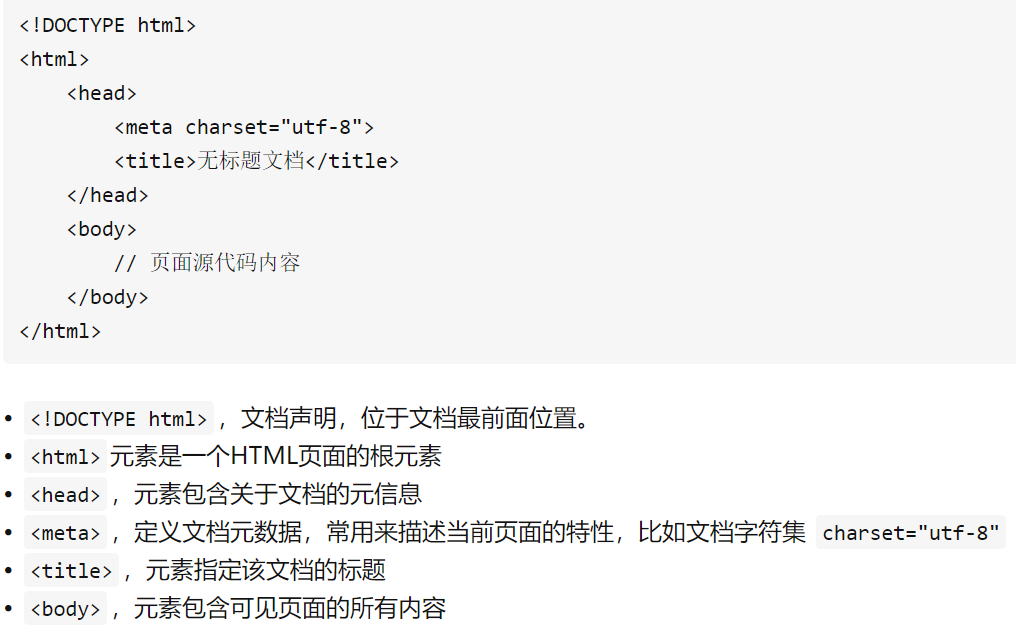
# HTML

## 基本结构



## XHTML

对 HTML 4 的改写，使其成为结构良好的 XML

1.与HTML 4.01几乎相同，是拓展版HTML

2.更严格更纯洁，大小写敏感

3.基于XML

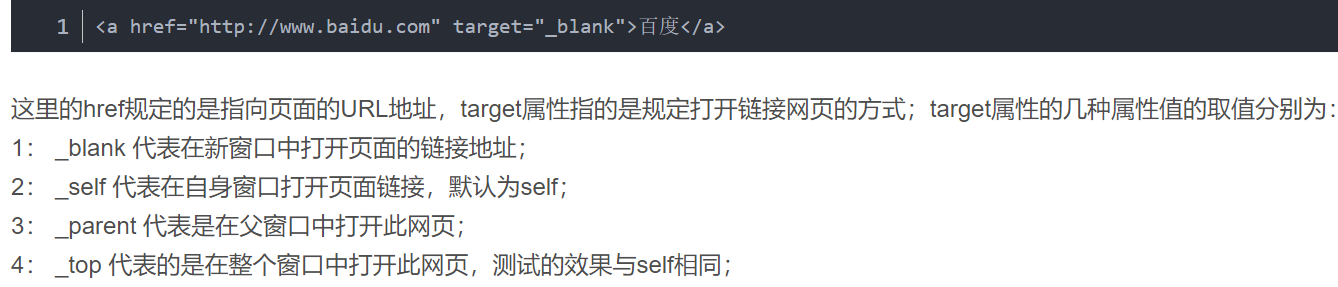
## HTML 4 5

1.声明简化

2.

## <a></a>标签全部作用

### 1.外部页面链接



在同一个页面中可能会有多个frame，因此会有多个窗口间的父子关系，\_parent在父窗口打开，\_top在根窗口打开

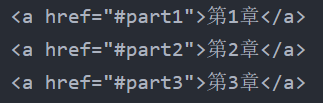
### 2.本地页面链接

在同一个项目中，href的指向为相对路径



### 3.锚点链接

这是页面内部的区域跳转

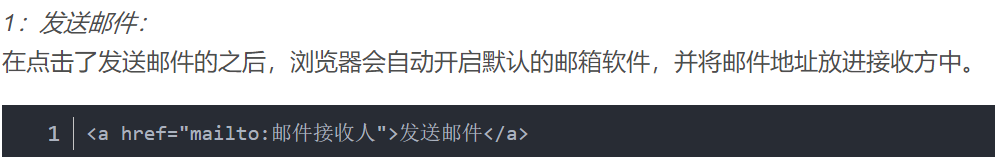


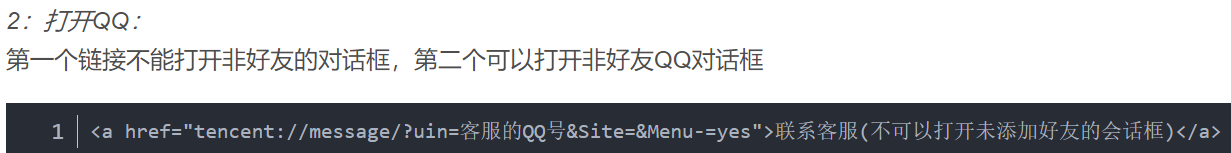
#为跳转到顶部

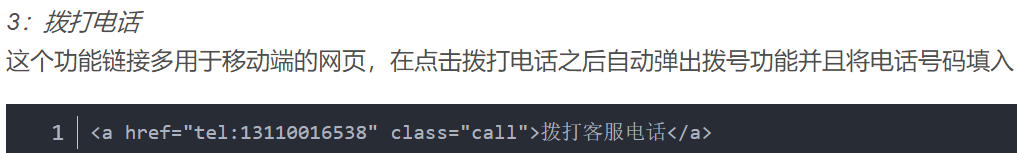
#idname 跳转到id为idname的标签

不支持通过.class跳转

### 4.功能性链接







## 元素种类的划分





### 1.块级元素

**定义：**块级元素是指本身属性为display:block;的元素，因为它自身的特点，我们通常使用块级元素来进行大布局、大结构的搭建。

**块级元素的特点**：

1. 每个块独占一行，不与其他元素共行显示
2. 可以手动设置宽高
3. 默认宽度与父元素保持一致(table除外，table的默认宽度与td单元格内容相关)

### 2.行内元素（内联元素）

**定义：**行内元素是指本身属性为display:inline;的元素，因为它自身的特点，我们通常使用行内元素来进行文字、小图标、小结构的搭建。

**行内元素的特点：**

1. 可以与其他元素共行显示
2. 默认尺寸由内容多少决定，不能手动设置宽高

### 3.行内块元素

**定义：**同时具备内联元素和块状元素的特点。

**特点：**

1. 可以与其他元素共行显示
2. 可以手动设置宽高

比如：<img><input>、

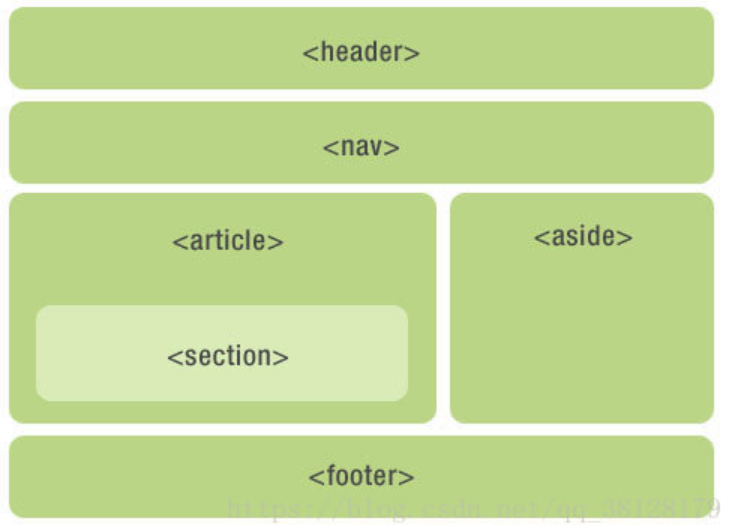
## H5的语义化作用及语义化标签

### 1.定义

语义是指对一个词或者句子含义的正确解释。很多html标签也具有语义的意义，也就是说元素本身传达了关于标签所包含内容类型的一些信息。例如，当浏览器解析到<h1></h1>标签时，它将该标签解释为包含这一块内容的最重要的标题。h1标签的语义就是用它来标识特定网页或部分最重要的标题。

### 2.常见元素

HTML5提供了新的语义元素来定义网页的不同部分，它们被称为“切片元素”，如图所示：



**<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。

**<section>元素**

元素用于标记文档的各个部分，例如长表单文章的章节或主要部分。

**<aside>元素**

定义与主要内容相关的内容块。通常显示为侧边栏。

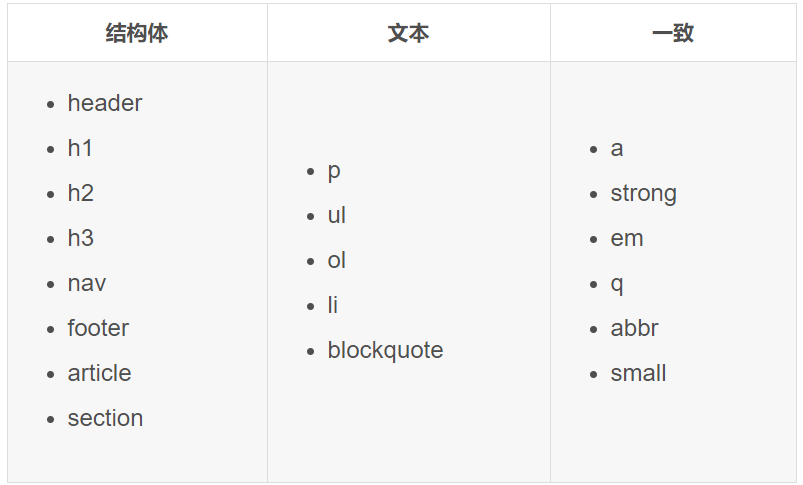
**<footer>元素**

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

**<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

语义元素：



**<h1>~<h6>元素**

定义页面的标题，h1元素具有最高等级，h6元素具有最低的等级

**<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。

**<small>元素**

为较不重要的内容定义小字体。如果被包围的字体已经是字体模型所支持的最小字号，那么 <small> 标签将不起任何作用。

**<strong>元素**

把文本定义为语气更强的强调的内容，以表示内容的重要性。

**<em>元素**

标记内容着重点（大量用于提升段落文本语义），通常呈现为斜体文字。

**<blockquote>元素**

定义块引用，浏览器会在 blockquote 元素前后添加换行，并增加外边距。cite属性可用来规定引用的来源

**<abbr>元素**

解释缩写词。使用title属性可提供全称，只在第一次出现时使用就ok。

### 3.优势

1. 代码结构: 使页面没有css的情况下，也能够呈现出很好的内容结构
2. 有利于SEO（Search Engine Optimization:搜索引擎优化）: 爬虫依赖标签来确定关键字的权重，因此可以和搜索引擎建立良好的沟通，帮助爬虫抓取更多的有效信息
3. 提升用户体验： 例如title、alt可以用于解释名称或者解释图片信息，以及label标签的灵活运用。
4. 便于团队开发和维护: 语义化使得代码更具有可读性，让其他开发人员更加理解你的html结构，减少差异化。
5. 方便其他设备解析: 如屏幕阅读器、盲人阅读器、移动设备等，以有意义的方式来渲染网页。

## Web Worker和Web Socket

### 1.Web Worker

#### （1）定义

为 JavaScript 创造多线程环境，允许主线程创建 Worker 线程，将一些任务分配给后者运行。在主线程运行的同时，Worker 线程在后台运行，两者互不干扰。等到 Worker 线程完成计算任务，再把结果返回给主线程。

Worker 线程一旦新建成功，就会始终运行，不会被主线程上的活动（比如用户点击按钮、提交表单）打断。

#### （2）限制

1）**同源限制**

分配给 Worker 线程运行的脚本文件，必须与主线程的脚本文件同源。

2）**DOM 限制**

Worker 线程所在的全局对象，与主线程不一样，无法读取主线程所在网页的 DOM 对象，也无法使用document、window、parent这些对象。但是，Worker 线程可以navigator对象和location对象。

3）**通信联系**

Worker 线程和主线程不在同一个上下文环境，它们不能直接通信，必须通过消息完成。

4）**脚本限制**

Worker 线程不能执行alert()方法和confirm()方法，但可以使用 XMLHttpRequest 对象发出 AJAX 请求。

5）**文件限制**

Worker 线程无法读取本地文件，即不能打开本机的文件系统（file://），它所加载的脚本，必须来自网络。

#### （3）用法

##### 1）主线程

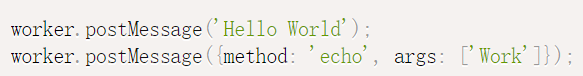
**创建Worker线程**

Worker()构造函数的参数是一个脚本文件，该文件就是 Worker 线程所要执行的任务。由于 Worker 不能读取本地文件，所以这个脚本必须来自网络。



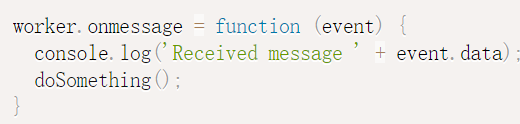
**向Worker发送消息**

worker.postMessage()方法的参数，就是主线程传给 Worker 的数据。它可以是各种数据类型，包括二进制数据。

****

**接收来自Worker的消息**

主线程通过worker.onmessage指定监听函数，接收子线程发回来的消息。事件对象event的data属性可以获取 Worker 发来的数据。



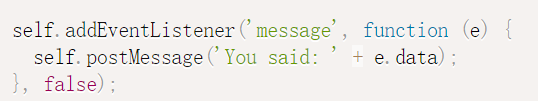
**关闭Worker**

****

##### 2）Worker线程

**创建监听函数**

监听message事件



self代表子线程自身，即子线程的全局对象。

除了使用self.addEventListener()指定监听函数，也可以使用self.onmessage指定。

监听函数的参数是一个事件对象，它的data属性包含主线程发来的数据。

self.postMessage()方法用来向主线程发送消息。

**自关闭**

self.close()用于在 Worker 内部关闭自身。

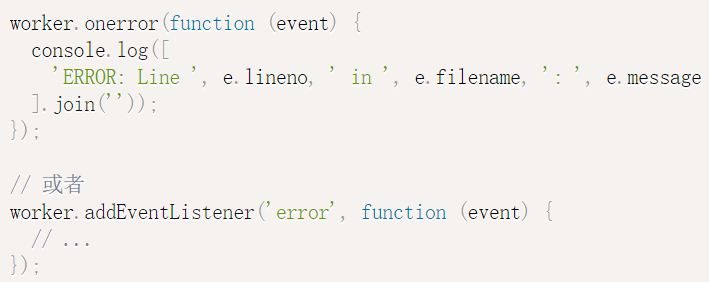
**加载脚本**

Worker 内部如果要加载其他脚本，有一个专门的方法importScripts()。

****

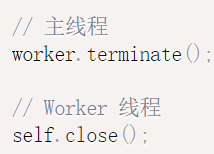
**错误处理**

主线程可以监听 Worker 是否发生错误。如果发生错误，Worker 会触发主线程的error事件。



**关闭 Worker**

使用完毕，为了节省系统资源，必须关闭 Worker。

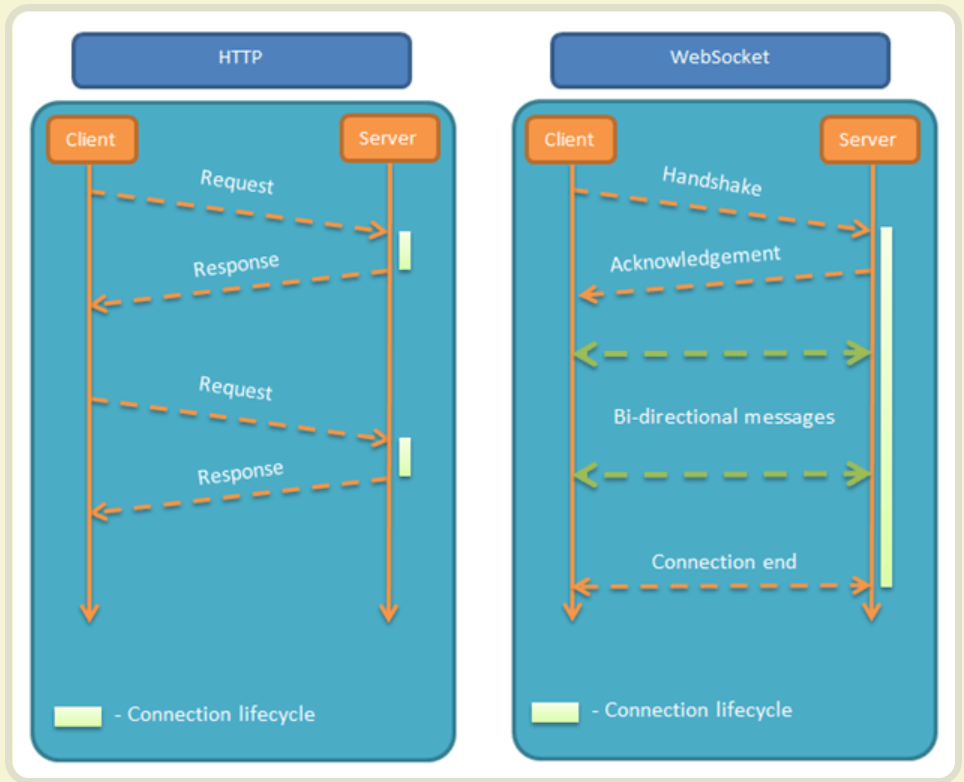


### 2.WebSocket

#### （1）定义

 HTTP 协议有一个缺陷：通信只能由客户端发起，客户端要获知就只能使用["轮询"](https://www.pubnub.com/blog/2014-12-01-http-long-polling/)：每隔一段时候，就发出一个询问，了解服务器有没有新的信息。

WebSocket是一种持久化协议，只需要一次握手（发送HTTP请求头）就可以建立持久性连接，并进行双向数据传输。



#### （2）性质

1）建立在 TCP 协议之上，服务器端的实现比较容易。

2）与 HTTP 协议有着良好的兼容性。默认端口也是80和443，并且握手阶段采用 HTTP 协议，因此握手时不容易屏蔽，能通过各种 HTTP 代理服务器。

3）数据格式比较轻量，性能开销小，通信高效。

4）可以发送文本，也可以发送二进制数据。

5）没有同源限制，客户端可以与任意服务器通信。

6）协议标识符是ws（如果加密，则为wss），服务器网址就是 URL。



#### （3）客户端使用

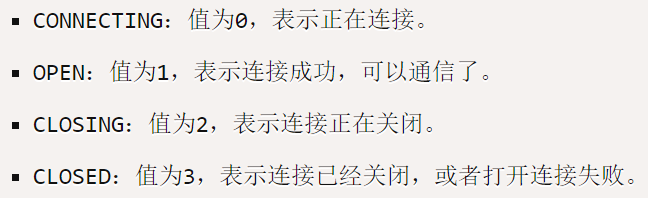
**WebSocket 构造函数**

****

执行上面语句之后，客户端就会与服务器进行连接。

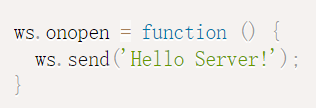
**WebSocket.readyState**

返回实例对象的当前状态



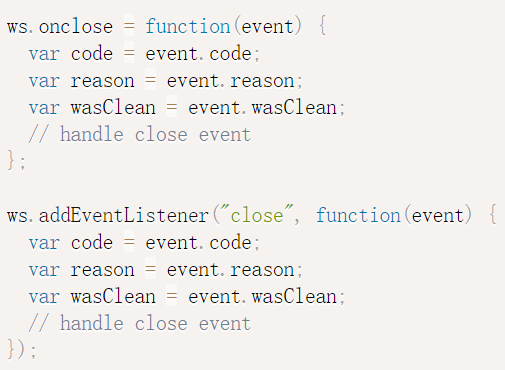
**WebSocket.onopen**

实例对象的onopen属性，用于指定连接成功后的回调函数。



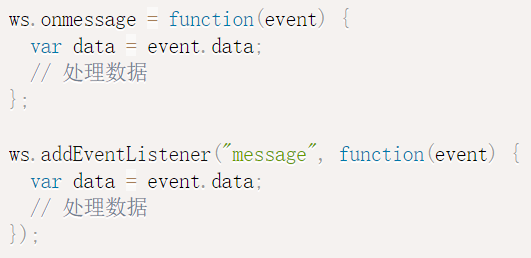
**WebSocket.onclose**

实例对象的onclose属性，用于指定连接关闭后的回调函数。



**WebSocket.onmessage**

实例对象的onmessage属性，用于指定收到服务器数据后的回调函数。



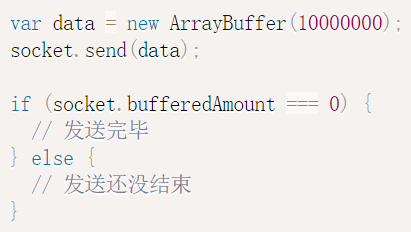
**webSocket.send()**

实例对象的send()方法用于向服务器发送数据。



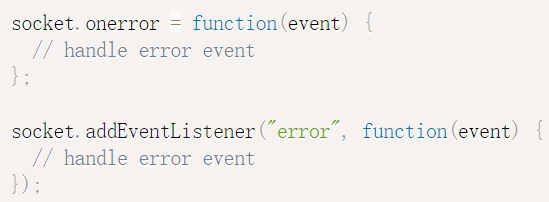
**webSocket.bufferedAmount**

实例对象的bufferedAmount属性，表示还有多少字节的二进制数据没有发送出去。它可以用来判断发送是否结束。



**webSocket.onerror**

实例对象的onerror属性，用于指定报错时的回调函数。

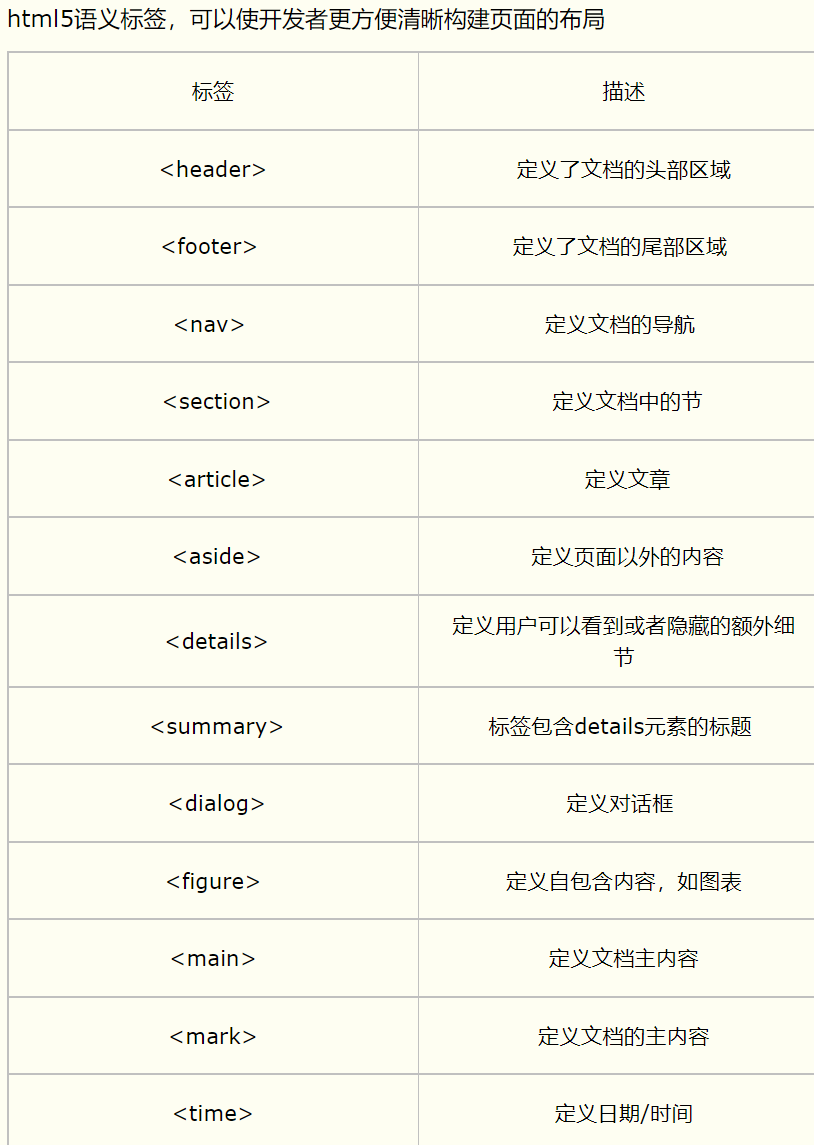


## SEO的概念及实现

**定义：**搜索引擎优化（简称seo）,指为了提升网页在搜索引擎自然搜索结果中（非商业性广告结果）的收录数量及排序位置而作的优化行为，这一行为的目的，是为了从搜索引擎中获得更多的免费流量，以及更好地展现网站形象。关键字/链接

## HTML5的新特性

### 一、语义标签

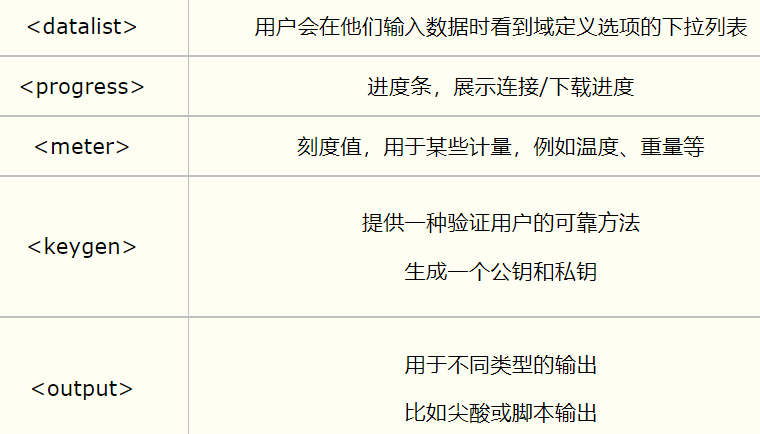


### 二、增强型表单

html5修改一些新的input输入特性，改善更好的输入控制和验证



html5新增了五个表单元素



html5新增表单属性



### 三、视频和音频



### 四、Canvas绘图

通过js绘制2D图形

Canvas 是逐像素进行渲染的。在 canvas 中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括任何或许已被图形覆盖的对象。

### 五、SVG绘图

**什么是SVG**

  SVG指可伸缩矢量图形

  SVG用于定义用于网络的基于矢量的图形

  SVG使用XML格式定义图形

  SVG图像在放大或改变尺寸的情况下其图形质量不会有损失

  SVG是万维网联盟的标准

**SVG的优势**

与其他图像格式相比，是哟个SVG的优势在于：

    SVG图像可通过文本编译器来创建和修改

   SVG图像可被搜索、索引、脚本化或压缩

    SVG是可伸缩的

    SVG图像可在任何的分辨率下被高质量的打印

    SVG可在图像质量不下降的情况下被放大

**SVG与Canvas区别**

SVG适用于描述XML中的2D图形的语言

Canvas随时随地绘制2D图形（使用javaScript）

SVG是基于XML的，意味这可以操作DOM，渲染速度较慢

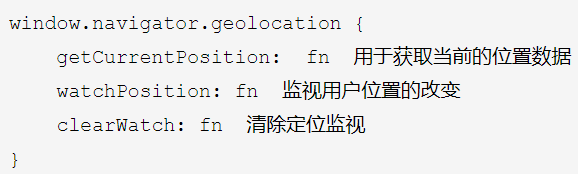
在SVG中每个形状都被当做是一个对象，如果SVG发生改变，页面就会发生重绘

Canvas是一像素一像素地渲染，如果改变某一个位置，整个画布会重绘。

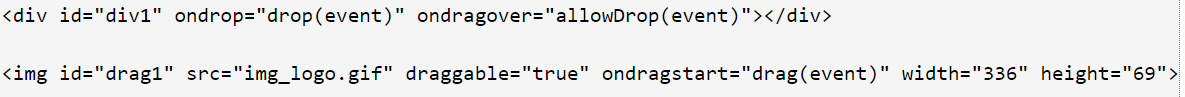


### 六、地理定位

HTML5 Geolocation（地理定位）用于定位用户的位置。



### 七、拖放API





### 八、WebWorker

Web Worker可以通过加载一个脚本文件，进而创建一个独立工作的线程，在主线程之外运行。

 基本使用：

     Web Worker的基本原理就是在当前javascript的主线程中，使用Worker类加载一个javascript文件来开辟一个新的线程，

起到互不阻塞执行的效果，并且提供主线程和新县城之间数据交换的接口：postMessage、onmessage。

### 九、WebStorage

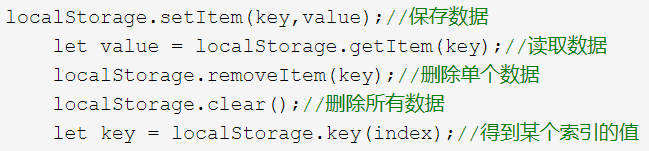
WebStorage是HTML新增的本地存储解决方案之一。WebSorage的意图在于解决本来不应该cookie做，却不得不用cookie的本地存储。

websorage拥有5M的存储容量，而cookie却只有4K，这是完全不能比的。

客户端存储数据有两个对象，其用法基本是一致：

localStorage：没有时间限制的数据存储

sessionStorage:在浏览器关闭的时候就会清除。



### 十、WebSocket

WebSocket协议为web应用程序客户端和服务端之间提供了一种全双工通信机制。

特点：

  （1）握手阶段采用HTTP协议，默认端口是80和443

  （2）建立在TCP协议基础之上，和http协议同属于应用层

  （3）可以发送文本，也可以发送二进制数据。

  （4）没有同源限制，客户端可以与任意服务器通信。

  （5）协议标识符是ws（如果加密，为wss），如ws://localhost:8023

## href、src

**href:** 用于建立链接或关系 (link、a)

**src:**  将某资源嵌入到当前文档元素定义的位置(img、script、iframe）

rel：对href进行说明描述

## 存储方式

**Web Storage：**

一种类似cookie的存储方式，但是不需要

|  |  |  |
| --- | --- | --- |
|  | localStorage | sessionStorage |
| 生命周期 | 永久的，只要是在某个浏览器内存储，只要不主动删除，就永远可以访问到；无法跨域； | 同源的窗口中始终存在（即使刷新）。浏览器窗口关闭后销毁，同时独立的打开同一个窗口同一个页面，sessionStorage也是不一样的。 |
| 存储大小 | 5MB | 5MB |
| 存储类型 | 字符串 | 字符串 |
| 存储位置 | 本地 | 本地 |
| 获取方式 | window.localStorage | window.sessionStorage |

**Cookie、Session、Token:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Cookie** | **Session** | **Token** |
| **过程** | 1、用户登录后，服务器端返回头中设置Set-Cookie：cookieId，date等  2、浏览器按照返回头生成Cookie  3、剩下的看程序员的计划 | 1、用户登录后，服务器给用户一个sessionid（一般通过cookie传送），自己存下session，两者对应；  2、用户获得sessionid，存于cookie  3、用户再此登录时携带cookie中的sessionid  4、服务器验证发现sessionid与session对应，免登录（或其他） | 1、用户登录后，服务端为其设置userid，并加密生成一个签名，userid+签名=token全部发给用户  2、用户得到token，下次请求携带token（userid+签名）  3、服务端将userid加密后，与签名比较，相同就验证成功。 |
| **大小** | 4KB | Sessionid 4KB  Session 服务端决定 | 1、请求头Authorization中传送（chrome250kb)  2、url传送(至少的为2KB) |
| **类型** | 字符串 | Sessionid同Cookie  Session在服务端，自定义 | 字符串 |
| **时间** | set-cookie中设置  不设置默认浏览器未关闭 | 服务端决定set-cookie | 将时间戳放入token，验证时取出与当前时间判断 |
| **跨域** | 各网站不同，不可跨域共享 | **？** | **？** |
| **优势** |  |  |  |
| **缺点** | CSRF | 多服务器不共享session  服务其存过多session，压力过大 |  |

## 跨域

### 1.定义

不满足同源策略的请求

同源：协议相同、域名相同、端口相同

恶意网站无法获取cookie、localstorage、DOM、ajax发送

### 2.解决

#### （1）Cookie

两个网页一级域名相同，只是二级域名不同，浏览器允许通过设置document.domain共享 Cookie。

#### （2）iframe

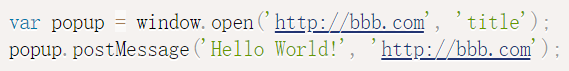
父子窗口间无法通信

如果两个网页一级域名相同，只是二级域名不同，设置document.domain

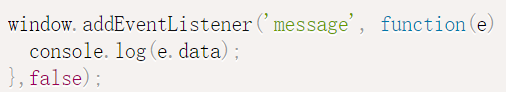
如果域名完全不同：

<1>window.postMessage

发送：



接收：



<2>window.name

<3>fragment indentifier

修改子窗口的src，通过hashchange监听

#### （3）localstorage

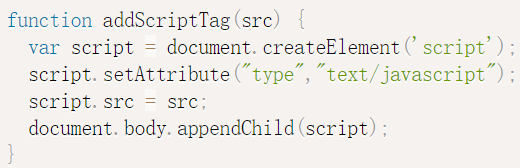
通过postMessage

#### （4）Ajax

<1>JSONP

添加一个<script>元素，向服务器请求JSON数据，这种做法不受同源政策限制

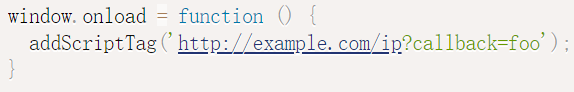
动态创建script标签



定义回调函数



执行回调函数



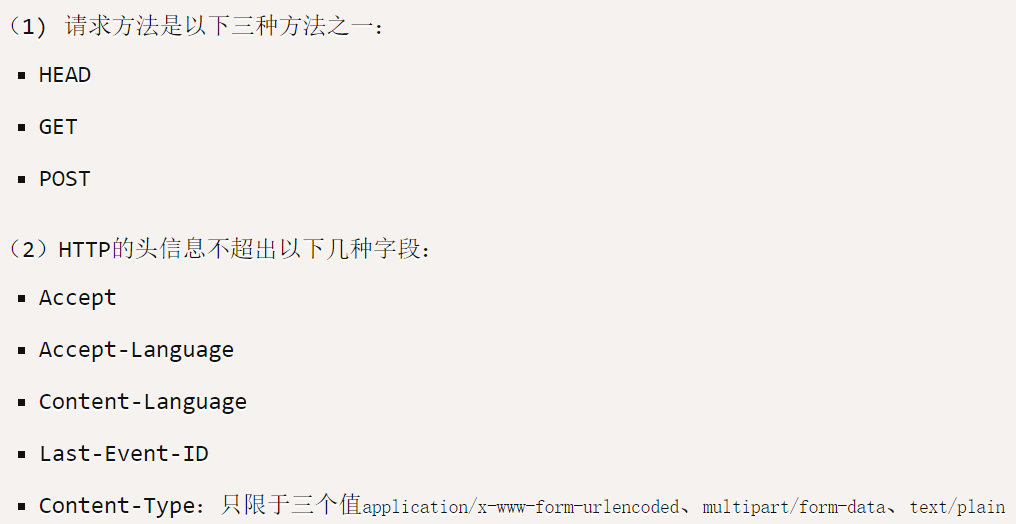
<2>websocket

WebSocket是一种通信协议，使用ws://（非加密）和wss://（加密）作为协议前缀。该协议不实行同源政策，只要服务器支持，就可以通过它进行跨源通信。



<3>CORS (Cross-origin resource sharing）

简单请求：



头信息自动添加Origin字段

非简单请求：

