# HTML笔记

## <meta> 元素

元数据（metadata）是关于数据的信息。

<meta> 标签提供关于 HTML 文档的元数据。元数据不会显示在页面上，但是对于机器是可读的。

典型的情况是，meta 元素被用于规定页面的描述、关键词、文档的作者、最后修改时间以及其他元数据。

<meta> 标签始终位于 head 元素中。

元数据可用于浏览器（如何显示内容或重新加载页面），搜索引擎（关键词），或其他 web 服务。

一些搜索引擎会利用meta元素的name和content属性来索引页面

下面的meta元素定义页面的描述：

<meta name=”” content=””/>

移动端的适配viewpoint 怎么写？

<meta name=”viewpoint”content=”width”=device-width,initial-scale=1,maximum-scale=1”>

## <br>or<br/>

使用<br/>更好在xhtml、xml以及未来html中不允许使用未闭合的标签。尽管目前<br>在所有浏览器都没有问题

## 新增被废除的标签和属性

标签：

<center>,<font>,<basefont>,<dir>(ul) ,<acronym>,<applet>,<big>,<frame>,<frameset>,<noframes>,<tt>

属性：

align , bgcolor , color

替换为 text-align, background-color;

HTML5新增标签:

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

## CSS引用

外部样式表，内部样式表，内联样式

1:<link rel=”stylesheet”type”text/css”href=””>

2:<style type=”text/css”></style>

3:<p style=”color:red;”></p>

## <a>标签

<a href=””,target=”\_blank”></a> //\_blank会在点击超链接后从新的窗口打开

name 属性可以为其创建一个索引，点击后会跳转至索引的内容。

<a href="#tips">有用的提示</a>

<a name="tips">基本的注意事项 - 有用的提示</a>

## HTML块级元素内联元素

块级元素

<h1>, <p>, <ul>, <table>

内联元素

<b>, <td>, <a>, <img>,<span>

使用相对路径是个好习惯

## W3C

## SVG和Canvas区别

SVG

SVG 是一种使用 XML 描述 2D 图形的语言。

SVG 基于 XML，这意味着 SVG DOM 中的每个元素都是可用的。您可以为某个元素附加 JavaScript 事件处理器。

在 SVG 中，每个被绘制的图形均被视为对象。如果 SVG 对象的属性发生变化，那么浏览器能够自动重现图形。

Canvas

Canvas 通过 JavaScript 来绘制 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。

Canvas特点

依赖分辨率

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

SVG特点

不依赖分辨率

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

## Html5兼容性问题

html5shiv 是一个针对 IE 浏览器的 HTML5 JavaScript 补丁，目的是让 IE 识别并支持 HTML5 元素。

在低版本IE中（IE8及以前）不允许对未知元素添加样式解决方案使用shiv（谢erv）

<!--[if lt IE 9]>

<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>

<endif]-->

以上代码是一段注释，但是 IE9 的早期版本会读取它（并理解它）

引用 shiv 代码的链接必须位于 <head> 元素中，因为 Internet Explorer 需要在读取之前认识所有新元素。

# CSS笔记

## CSS选择器

（1）id选择器（#myid）

（2）类选择器（.myclassname）

（3）标签选择器（div,h1,p）

（4）后代选择器（h1 p{} div.feng h1{}）

<h1>This is a <em>important</em> heading</h1>

（5）相邻后代选择器（子）选择器（ul>li）

与后代选择器相比，子元素选择器（Child selectors）只能选择作为某元素子元素的元素。 （所有） 就是说 < ul><li><ol><li></li></ol></li><ul> ol后的li没反应。

（6）兄弟选择器（li~a）

指的是li后所有a 不管是不是相邻 同一个父亲 作用是查找某一个指定元素的**后面的所有兄弟结点**。 <li></li><a></a><a></a>

（7）相邻兄弟选择器（li+a）

可选择紧接在另一元素后的元素，且二者有相同父元素。

（8）属性选择器（img[alt]{}）

（9）伪类选择器（a:hover,li:nth-child）

（10）伪元素选择器（::before、::after）

（11）通配符选择器（\*）

《相邻兄弟选择器》

<div>

<ul>

<li>List item 1</li>

<li>List item 2</li>

<li>List item 3</li>

</ul>

<ol>

<li>List item 1</li>

<li>List item 2</li>

<li>List item 3</li>

</ol>

</div>

在上面的片段中，div 元素中包含两个列表：一个无序列表，一个有序列表，每个列表都包含三个列表项。这两个列表是相邻兄弟，列表项本身也是相邻兄弟。不过，第一个列表中的列表项与第二个列表中的列表项不是相邻兄弟，因为这两组列表项不属于同一父元素（最多只能算堂兄弟）。

一个结合符只能选择两个相邻兄弟中的第二个元素 请看：

Li+li{font-weight:bold};

结合其他选择器

相邻兄弟结合符还可以结合其他结合符：

html > body table + ul {margin-top:20px;}

这个选择器解释为：选择紧接在 table 元素后出现的所有兄弟 ul 元素，该 table 元素包含在一个 body 元素中，body 元素本身是 html 元素的子元素。

## 什么是CSS Sprite（“精灵图”）

CSS Sprites叫 CSS精灵或者雪碧图，是一种网页图片应用处理方式。

CSS Sprites其实就是把网页中一些背景图片整合到一张图片文件中。

使用图像精灵将减少服务器请求的数量并节约带宽。

再利用CSS的"background-image"，“background-repeat”，"background-position"的组合进行背景定位，background-position可以用数字精确的定位出背景图片的位置。

## Css模型

![形状

描述已自动生成

Content-padding-margin-border

元素的总宽度应该这样计算：

元素总宽度 = 宽度 + 左内边距 + 右内边距 + 左边框 + 右边框 + 左外边距 + 右外边距

元素的总高度应该这样计算：

元素总高度 = 高度 + 上内边距 + 下内边距 + 上边框 + 下边框 + 上外边距 + 下外边距

CSS外边距合并

简单地说，外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

当一个元素出现在另一个元素上面时，第一个元素的下外边距与第二个元素的上外边距会发生合并。图示

描述已自动生成

图示

描述已自动生成文本

描述已自动生成表格

描述已自动生成

**注释：**只有普通文档流中块框的垂直外边距才会发生外边距合并。行内框、浮动框或绝对定位之间的外边距不会合并。

## 浮动问题

使用clear属性可以是left，right，top，none 表示框的哪边不挨着浮动框

Clear属性要用在非浮动属性中，而不是浮动本身。

图形用户界面

描述已自动生成

**或者添加一个外边距**

图形用户界面, 应用程序

描述已自动生成

.news {

background-color: gray;

border: solid 1px black;

}

.news img {

float: left;

}

.news p {

float: right;

}

<div class="news">

<img src="news-pic.jpg" />

<p>some text</p>

</div>

浮动元素会脱离文本流，这样会导致div不占据空间。如何让包围元素在视觉上保卫浮动元素呢？需要在这个元素的某个地方使用clear，但是由于没有多余的元素可以清理，所以必须添加一个空元素去清理它（容器内部添加）。因此不得不为了布局添加一个无意义的标签。

图形用户界面, 图示, 应用程序

描述已自动生成

方法2：对容器div进行浮动

.news {

background-color: gray;

border: solid 1px black;

float: left;

}

.news img {

float: left;

}

.news p {

float: right;

}

<div class="news">

<img src="news-pic.jpg" />

<p>some text</p>

</div>

但是这样会影响到下一个元素。为了这个问题这里还可以有第三种方法，就是所有元素都进行浮动，然后使用适当的有意义的元素，例如站点的页脚对这些浮动进行清理。有助于减少不必要的标记。

## CSS定位问题

CSS 有三种基本的定位机制：普通流、浮动和绝对定位。

除非有规定否则所有框都在普通流中定位。

position 属性值的含义：

static

元素框正常生成。块级元素生成一个矩形框，作为文档流的一部分，行内元素则会创建一个或多个行框，置于其父元素中。默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。

relative

元素框偏移某个距离。元素仍保持其未定位前的形状，它原本所占的空间仍保留。生成相对定位的元素，相对于其正常位置进行定位。

absolute

元素框从文档流完全删除，并相对于其包含块定位。包含块可能是文档中的另一个元素或者是初始包含块。元素原先在正常文档流中所占的空间会关闭，就好像元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框。生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。

fixed

元素框的表现类似于将 position 设置为 absolute，不过其包含块是视窗本身。  
生成绝对定位的元素，相对于浏览器窗口进行定位。

Inherit

规定应该从父元素继承 position 属性的值。

提示：相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置。

## Box-Sizing

默认情况下：

* width + padding + border = 元素的实际宽度
* height + padding + border = 元素的实际高度

文本, 应用程序

中度可信度描述已自动生成

box-sizing 属性允许我们在元素的总宽度和高度中包括内边距和边框。

如果在元素上设置了 box-sizing: border-box;，则宽度和高度会包括内边距和边框：

文本

低可信度描述已自动生成

## 伪类和伪元素

伪类用于定义元素的特殊状态 例如 a:hover

顺序 LVHA

:first-child 伪类与指定元素匹配，该元素是另一个元素的第一个子元素

伪元素用于设置元素指定部分的样式

* 设置元素的首字母、首行的样式
* 在元素的内容之前或之后插入内容

::first-line 用于向文本首行添加特殊样式。 这个元素只能用于块级元素

::first-letter 用于向文本首字母添加特殊样式 也只用于块级元素。

::before 在元素内容之前插入一些内容例如图像等

::after 在元素内容后插入一些内容。

::section匹配用户选中的元素部分。

# AJAX笔记

Ajax不是编程语言，他能用于页面加载后请求数据、接收数据。向后台发送数据。不刷新更新网页。

## Ajax工作原理

图示

描述已自动生成

网页中发生一个事件（页面加载、按钮点击）

由 JavaScript 创建 XMLHttpRequest 对象

XMLHttpRequest 对象向 web 服务器发送请求

服务器处理该请求

服务器将响应发送回网页

由 JavaScript 读取响应

由 JavaScript 执行正确的动作（比如更新页面）

创建XMLHttpRequest语法是

Variable =new XMLHttpRequest();

老版本IE(5and6)使用activeX对象

Variable = new activeXobject(“Microsoft.XMLHTTP”);

应对所有浏览器应该先判断是否支持XMLhttp对象如果支持创建不支持创建activeX对象

var xhttp;

if (window.XMLHttpRequest) {

xhttp = new XMLHttpRequest();

} else {

// code for IE6, IE5

xhttp = new ActiveXObject("Microsoft.XMLHTTP");

}

## XMLHTTPRequest

向服务器发送请求使用send()向服务器发送请求和open()方法规定请求的类型

Xhttp.open(“GET”,”ajax\_info.txt”,true); true为异步 false为同步

Xhttp.send(); 用于post方法

在大多数情况下GET比POST更快

但以下情况始终使用post

1. 缓存文件不是选项（更新服务器上的文件或者数据）
2. 向服务器发送大量数据（post无大小限制）
3. 发送用户输入（可包含未知字符），post比get更强大安全

GET请求

为了避免请求的缓存获得的都是同一个要向url添加一个唯一id

xhttp.open("GET", "demo\_get.asp?t=" + Math.random(), true);

GET发送

xhttp.open("GET", "demo\_get2.asp?fname=Bill&lname=Gates", true);

POST请求

xhttp.open("POST", "demo\_post.asp", true);

xhttp.send();

POST发送

如果要像表单那样post数据要通过setRequestHeader（）添加一个http头部在send（）方法中规定需要发送的数据

xhttp.open("POST", "ajax\_test.asp", true);

xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded"); （header,value）规定头部名称，规定头部值

xhttp.send("fname=Bill&lname=Gates");

## onreadystatechange 属性

当请求被发送到服务器时，我们需要执行一些基于响应的任务。

每当 readyState 改变时，就会触发 onreadystatechange 事件。

readyState 属性存有 XMLHttpRequest 的状态信息。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

<div id="demo">

<h1>XMLHttpRequest 对象</h1>

<button type="button" onclick="loadDoc()">更改内容</button>

</div>

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("demo").innerHTML = this.responseText;

}

};

xhttp.open("GET", "ajax\_info.txt", true);

xhttp.send();

注意： onreadystatechange 事件被触发 5次每次readystate都发生变化。

## 回调函数

如果您的网站中有多个 AJAX 任务，那么您应该创建一个执行 XMLHttpRequest 对象的函数，以及一个供每个 AJAX 任务的回调函数。

<div id="demo">

<h1>XMLHttpRequest 对象</h1>

<button type="button" onclick="loadDoc('/demo/js/ajax\_info.txt', myFunction)">更改内容

</button>

</div>

<script>

function loadDoc(url, cFunction) { //请求

var xhttp;

xhttp=new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

cFunction(this);

}

};

xhttp.open("GET", url, true);

xhttp.send();

}

function myFunction(xhttp) { //请求

document.getElementById("demo").innerHTML =

xhttp.responseText;

}

</script>

## 响应属性

社交网站的屏幕截图

描述已自动生成

responseText属性

document.getElementById("demo").innerHTML = xhttp.responseText;

responseXML 属性

XML HttpRequest 对象有一个內建的 XML 解析器。

ResponseXML 属性以 XML DOM 对象返回服务器响应。

使用此属性，您可以把响应解析为 XML DOM 对象：

请求文件 [music\_list.xml](https://www.w3school.com.cn/demo/music_list.xml)，并对响应进行解析：

xmlDoc = xhttp.responseXML;

txt = "";

x = xmlDoc.getElementsByTagName("ARTIST");

for (i = 0; i < x.length; i++) {

txt += x[i].childNodes[0].nodeValue + "<br>";

}

document.getElementById("demo").innerHTML = txt;

xhttp.open("GET", "music\_list.xml", true);

xhttp.send();

getAllResponseHeaders() 方法

getAllResponseHeaders() 方法返回所有来自服务器响应的头部信息。

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("demo").innerHTML = this.getAllResponseHeaders();

}

};

getResponseHeader() 方法

getResponseHeader() 方法返回来自服务器响应的特定头部信息。

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("demo").innerHTML = this.getResponseHeader("Last-Modified");

}

};

xhttp.open("GET", "ajax\_info.txt", true);

xhttp.send();