**HTML5**

# HTML5的概念

HTML5 是下一代 HTML 标准。

## 相较于HTML4的区别

HTML5 中的一些有趣的新特性：

* 新元素
* 新属性
* 完全支持 CSS3
* Video 和 Audio
* 2D/3D 制图
* 本地存储
* 本地 SQL 数据
* Web 应用

## 要求

声明

<!DOCTYPE html>

<!--<!doctype> 声明必须位于 HTML5 文档中的第一行,使用非常简单-->

最小的H5文档

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>文档标题</title>

</head>

<body>

文档内容......

</body>

</html>

# H5浏览器兼容

<!--[if lt IE 9]>

//国外

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

//国内

<script src="http://cdn.static.runoob.com/libs/html5shiv/3.7/html5shiv.min.js"></script>

<![endif]-->

以上代码是一个注释，作用是在 IE 浏览器的版本小于 IE9 时将读取 html5.js 文件，并解析它。

## 将 HTML5 元素定义为块元素

HTML5 定了 8 个新的 HTML 语义（semantic） 元素。所有这些元素都是 块级 元素。

为了能让旧版本的浏览器正确显示这些元素，你可以设置 CSS 的 display 属性值为 block

header, section, footer, aside, nav, main, article, figure {

display: block;

}

## 为HTML5添加新元素

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>为 HTML 添加新元素</title>

<script>document.createElement("myHero")</script>

<style>

myHero {

display: block;

background-color: #ddd;

padding: 50px;

font-size: 30px;

}

</style>

</head>

​

<body>

​

<h1>我的第一个标题</h1>

​

<p>我的第一个段落。</p>

​

<myHero>我的第一个新元素</myHero>

​

</body>

</html>

# H5新元素

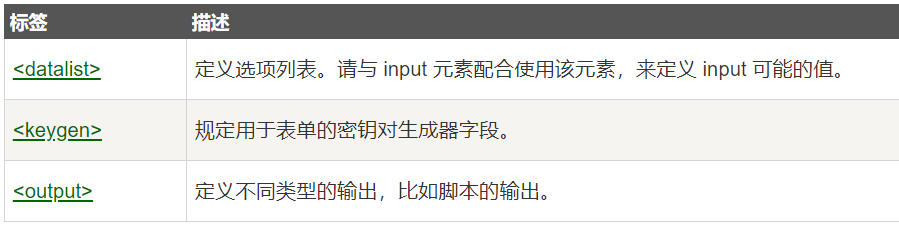
### canvas图形绘制



### 新多媒体元素



### 新表单元素



### 新语义结构元素

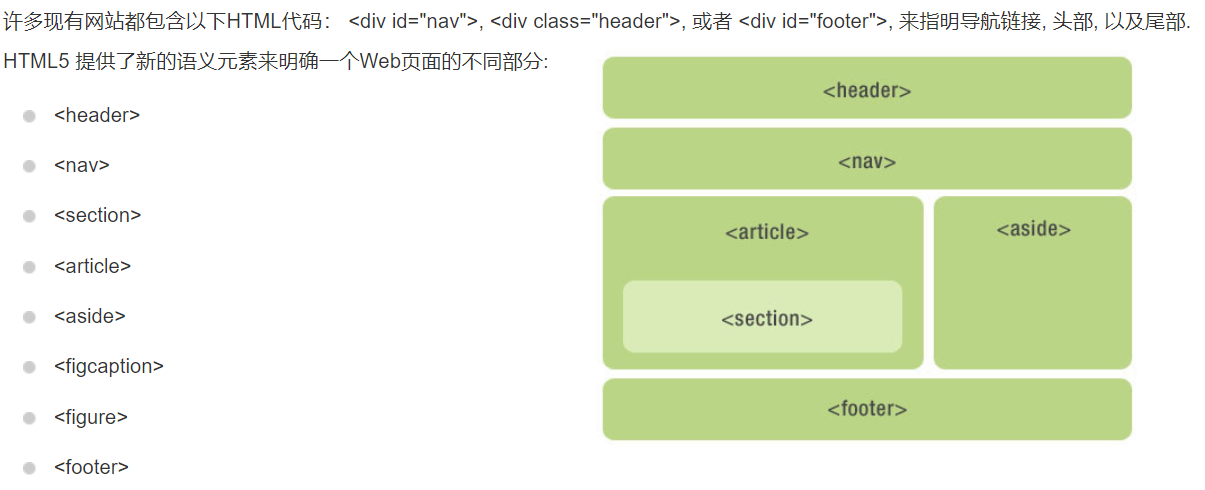


### 删除元素

* <acronym>
* <applet>
* <basefont>
* <big>
* <center>
* <dir>
* <font>
* <frame>
* <frameset>
* <noframes>
* <strike>
* <tt>

### 语义元素

* 语义元素 = 有意义的元素
* 一个语义元素能够清楚的描述其意义给浏览器和开发者
* 无语义 元素实例: <div> 和 <span> - 无需考虑内容
* 语义元素实例: <form>, <table>, and <img> - 清楚的定义了它的内容



**HTML5 <figure> 和 <figcaption> 元素**

* <figure>标签规定独立的流内容（图像、图表、照片、代码等等）。
* <figure> 元素的内容应该与主内容相关，但如果被删除，则不应对文档流产生影响。
* <figcaption> 标签定义 <figure> 元素的标题.
* <figcaption>元素应该被置于 "figure" 元素的第一个或最后一个子元素的位置

# canvas

<canvas>只是图形容器,需要使用脚本控制图形内容

注：canvas 元素本身是没有绘图能力的。所有的绘制工作必须在 JavaScript 内部完成

* 绘制线条
* 绘制矩形
* 绘制圆形
* 绘制渐变
* 绘制图片

# 内联SVG

## SVG

* SVG 指可伸缩矢量图形 (Scalable Vector Graphics)
* 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优势

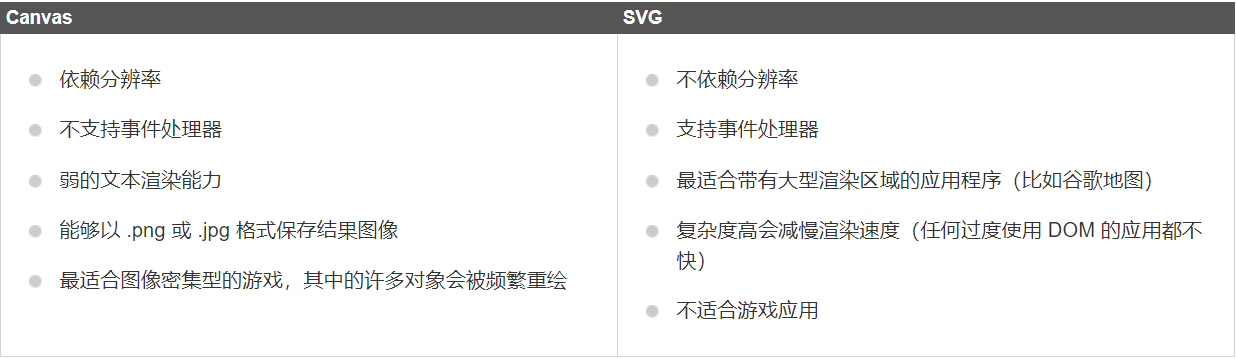
与其他图像格式相比（比如 JPEG 和 GIF），使用 SVG 的优势在于：

* SVG 图像可通过文本编辑器来创建和修改
* 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地打印
* SVG 可在图像质量不下降的情况下被放大

## SVG和canvas比较

区别：

* SVG 是一种使用 XML 描述 2D 图形的语言。
* Canvas 通过 JavaScript 来绘制 2D 图形。
* 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 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。



# MathML

HTML5 可以在文档中使用 MathML 元素，对应的标签是 <math>...</math> 。

MathML 是数学标记语言，是一种基于XML（标准通用标记语言的子集）的标准，用来在互联网上书写数学符号和公式的置标语言。

# 拖放

拖放（drag&drop）：

拖放是一种常见的特性，即抓取对象以后拖到另一个位置。

在 HTML5 中，拖放是标准的一部分，任何元素都能够拖放。

# 地理位置

地理位置（geolocation）：

HTML5 Geolocation API 用于获得用户的地理位置。

鉴于该特性可能侵犯用户的隐私，除非用户同意，否则用户位置信息是不可用的。

# 多媒体

## 视频

<video width="320" height="240" controls>

<source src="movie.mp4" type="video/mp4">

<source src="movie.ogg" type="video/ogg">

您的浏览器不支持Video标签。

</video>

* <video> 元素提供了 播放、暂停和音量控件来控制视频。
* 同时 <video> 元素也提供了 width 和 height 属性控制视频的尺寸.如果设置的高度和宽度，所需的视频空间会在页面加载时保留。如果没有设置这些属性，浏览器不知道大小的视频，浏览器就不能再加载时保留特定的空间，页面就会根据原始视频的大小而改变。
* <video> 与</video> 标签之间插入的内容是提供给不支持 video 元素的浏览器显示的。
* <video> 元素支持多个 <source> 元素. <source> 元素可以链接不同的视频文件。浏览器将使用第一个可识别的格式：

视频兼容性：



## 音频

<audio controls>

<source src="horse.ogg" type="audio/ogg">

<source src="horse.mp3" type="audio/mpeg">

您的浏览器不支持 audio 元素。

</audio>

* control 属性供添加播放、暂停和音量控件。
* 在<audio> 与 </audio> 之间你需要插入浏览器不支持的<audio>元素的提示文本 。
* <audio> 元素允许使用多个 <source> 元素. <source> 元素可以链接不同的音频文件，浏览器将使用第一个支持的音频文件

# 表单

## input类型

<input type="date" name="bday">

type：

* color：选取颜色
* date：选取日期
* datetime：选择一个日期（UTC 时间）
* datetime-local：选择一个日期和时间 (无时区)
* month
* week
* email
* tel
* search
* number

<input type="number" name="quantity" min="1" max="5">



* range：包含一定范围内数字值的输入域，显示为滑动条

<input type="range" name="points" min="1" max="10">

属性：min、max、step、value

## 表单元素

* **<datalist>**
* <datalist> 元素规定输入域的选项列表。
* <datalist> 属性规定 form 或 input 域应该拥有自动完成功能。当用户在自动完成域中开始输入时，浏览器应该在该域中显示填写的选项：
* 使用 <input> 元素的列表属性与 <datalist> 元素绑定

<input list="browsers">

<datalist id="browsers">

<option value="Internet Explorer">

<option value="Firefox">

<option value="Chrome">

<option value="Opera">

<option value="Safari">

</datalist>

* **<keygen>**
* <keygen> 元素的作用是提供一种验证用户的可靠方法。
* <keygen>标签规定用于表单的密钥对生成器字段。
* 当提交表单时，会生成两个键，一个是私钥，一个公钥。
* 私钥（private key）存储于客户端，公钥（public key）则被发送到服务器。公钥可用于之后验证用户的客户端证书（client certificate）

<form action="demo\_keygen.asp" method="get">

用户名: <input type="text" name="usr\_name">

加密: <keygen name="security">

<input type="submit">

</form>

* **<output>**

<output> 元素用于不同类型的输出

//将计算结果显示在 <output> 元素:

<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">0

<input type="range" id="a" value="50">100 +

<input type="number" id="b" value="50">=

<output name="x" for="a b"></output>

</form>

## 表单属性

<form>新属性：

* autocomplete
* novalidate

<input>新属性：

* autocomplete
* autofocus
* form
* formaction
* formenctype
* formmethod
* formnovalidate
* formtarget
* height 与 width
* list
* min 与 max
* multiple
* pattern (regexp)
* placeholder
* required
* step

### <form> / <input>：autocomplete（自动填充）

* autocomplete 属性规定 form 或 input 域应该拥有自动完成功能。
* 当用户在自动完成域中开始输入时，浏览器应该在该域中显示填写的选项。
* 提示: autocomplete 属性有可能在 form元素中是开启的，而在input元素中是关闭的。
* 注意: autocomplete 适用于 <form> 标签，以及以下类型的 <input> 标签：text, search, url, telephone, email, password, datepickers, range 以及 color。

### <form>：novalidate（无需验证）

* novalidate 属性是一个 boolean(布尔) 属性.
* novalidate 属性规定在提交表单时不应该验证 form 或 input 域。

### <input>：autofocus （自动获取焦点）

属性规定在页面加载时，域自动地获得焦点

### <input>：form

规定输入域所属的一个或多个表单

//位于form表单外的 input 字段引用了 HTML form (该 input 表单仍然属于form表单的一部分)

<form action="demo-form.php" id="form1">

First name: <input type="text" name="fname"><br> <input type="submit" value="提交">

</form>

Last name: <input type="text" name="lname" form="form1">

### <input>： formaction

The formaction 属性用于描述表单提交的URL地址

The formaction 属性会覆盖<form> 元素中的action属性

### <input>： formenctype / formmethod

* formenctype 属性描述了表单提交到服务器的数据编码 (只对form表单中 method="post" 表单)
* formmethod 属性定义了表单提交的方式
* formenctype 属性覆盖 form 元素的 enctype 属性。
* formmethod 属性覆盖了 <form> 元素的 method 属性。
* 主要: 这两种属性与 type="submit" 和 type="image" 配合使用。

<form action="demo-post\_enctype.php" method="post">

First name:

<input type="text" name="fname"><br>

<input type="submit" value="提交">

<input type="submit" formenctype="multipart/form-data" value="以 Multipart/form-data 提交">

</form>

### <input> ：formnovalidate

* novalidate属性描述了 <input> 元素在表单提交时无需被验证
* formnovalidate 属性会覆盖 <form> 元素的novalidate属性

### <input>： formtarget

* formtarget 属性指定一个名称或一个关键字来指明表单提交数据接收后的展示。
* The formtarget 属性覆盖 <form>元素的target属性.
* 注意: formtarget 属性与type="submit" 和 type="image"配合使用

### <input> multiple

* multiple 属性是一个 boolean 属性.
* multiple 属性规定<input> 元素中可选择多个值。
* 注意: multiple 属性适用于以下类型的 <input> 标签：email 和 file

### <input> height 和 width

* height 和 width 属性规定用于 image 类型的 <input> 标签的图像高度和宽度。
* 注意: height 和 width 属性只适用于 image 类型的<input> 标签。
* 提示:图像通常会同时指定高度和宽度属性。如果图像设置高度和宽度，图像所需的空间 在加载页时会被保留。如果没有这些属性， 浏览器不知道图像的大小，并不能预留 适当的空间。图片在加载过程中会使页面布局效果改变 （尽管图片已加载）

### <input> pattern

pattern 属性描述了一个正则表达式用于验证 <input> 元素的值。

注意：pattern 属性适用于以下类型的 <input> 标签--text, search, url, tel, email, 和 password.

提示：是用来全局 title 属性描述了模式

# WEB存储

**类型：**

客户端存储数据的两个对象为：

* localStorage - 用于长久保存整个网站的数据，保存的数据没有过期时间，直到手动去除。
* sessionStorage - 用于临时保存同一窗口(或标签页)的数据，在关闭窗口或标签页之后将会删除这些数据。

在使用 web 存储前,应检查浏览器是否支持 localStorage 和sessionStorage：

if(typeof(Storage)!=="undefined") {

// 是的! 支持 localStorage sessionStorage 对象! // 一些代码.....

} else {

// 抱歉! 不支持 web 存储。

}

**API：**

不管是 localStorage，还是 sessionStorage，可使用的API都相同，常用的有如下几个（以localStorage为例）：

* 保存数据：localStorage.setItem(key,value);
* 读取数据：localStorage.getItem(key);
* 删除单个数据：localStorage.removeItem(key);
* 删除所有数据：localStorage.clear();
* 得到某个索引的key：localStorage.key(index);
* 提示: 键/值对通常以字符串存储，你可以按自己的需要转换该格式。

## localStorage（长久保存）

// 存储

localStorage.sitename = "菜鸟教程";

// 查找

document.getElementById("result").innerHTML = localStorage.sitename;

// 删除

localStorage.removeItem("sitename");

## sessionStorage（暂时保存）

和localStorage类似

# WEB SQL数据库

核心方法：

* openDatabase：这个方法使用现有的数据库或者新建的数据库创建一个数据库对象。
* transaction：这个方法让我们能够控制一个事务，以及基于这种情况执行提交或者回滚。
* executeSql：这个方法用于执行实际的 SQL 查询。

## 打开数据库

我们可以使用 openDatabase() 方法来打开已存在的数据库，如果数据库不存在，则会创建一个新的数据库，使用代码如下：

var db = openDatabase('mydb', '1.0', 'Test DB', 2 \* 1024 \* 1024);

openDatabase() 方法对应的五个参数说明：

* 数据库名称
* 版本号
* 描述文本
* 数据库大小
* 创建回调
* 第五个参数，创建回调会在创建数据库后被调用。

## 数据库的增删查改

// 创建数据库

var db = openDatabase('mydb', '1.0', 'Test DB', 2 \* 1024 \* 1024);

db.transaction(function (tx) {

// 创建数据库

tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, log)');

// 插入数据

// way 1

tx.executeSql('INSERT INTO LOGS (id, log) VALUES (1, "菜鸟教程")');

// way 2

// tx.executeSql('INSERT INTO LOGS (id,log) VALUES (?, ?)', [e\_id, e\_log]);

// 删除数据

tx.executeSql('DELETE FROM LOGS WHERE id=1');

// 更新数据

tx.executeSql('UPDATE LOGS SET log=\'www.w3cschool.cc\' WHERE id=2');

});

// 查询数据库

db.transaction(function (tx) {

tx.executeSql('SELECT \* FROM LOGS', [], function (tx, results) {

var len = results.rows.length, i;

msg = "<p>查询记录条数: " + len + "</p>";

document.querySelector('#status').innerHTML += msg;

for (i = 0; i < len; i++) {

alert(results.rows.item(i).log);

}

}, null);

});

# 应用程序缓存

**应用缓存（Cache Manifest）**

HTML5 引入了应用程序缓存，这意味着 web 应用可进行缓存，并可在没有因特网连接时进行访问。

应用程序缓存为应用带来三个优势：

* 离线浏览 - 用户可在应用离线时使用它们
* 速度 - 已缓存资源加载得更快
* 减少服务器负载 - 浏览器将只从服务器下载更新过或更改过的资源。

## Cache Manifest 基础

如需启用应用程序缓存，请在文档的<html> 标签中包含 manifest 属性：

<!DOCTYPE HTML>

<html manifest="demo.appcache">

...

</html>

* 每个指定了 manifest 的页面在用户对其访问时都会被缓存。如果未指定 manifest 属性，则页面不会被缓存（除非在 manifest 文件中直接指定了该页面）。
* manifest 文件的建议的文件扩展名是：".appcache"。
* Remark 请注意，manifest 文件需要配置正确的 MIME-type，即 "text/cache-manifest"。必须在 web 服务器上进行配置。

## Manifest 文件

manifest 文件是简单的文本文件，它告知浏览器被缓存的内容（以及不缓存的内容）。

manifest 文件可分为三个部分：

* CACHE MANIFEST - 在此标题下列出的文件将在首次下载后进行缓存
* NETWORK - 在此标题下列出的文件需要与服务器的连接，且不会被缓存
* FALLBACK - 在此标题下列出的文件规定当页面无法访问时的回退页面（比如 404 页面）

**CACHE MANIFEST**

CACHE MANIFEST

/theme.css

/logo.gif

/main.js

上面的 manifest 文件列出了三个资源：一个 CSS 文件，一个 GIF 图像，以及一个 JavaScript 文件。当 manifest 文件加载后，浏览器会从网站的根目录下载这三个文件。然后，无论用户何时与因特网断开连接，这些资源依然是可用的。

**NETWORK**

# 下面的 NETWORK 小节规定文件 "login.php" 永远不会被缓存，且离线时是不可用的：

NETWORK:

login.php

# 可以使用星号来指示所有其他资源/文件都需要因特网连接：

NETWORK:

\*

**FALLBACK**

下面的 FALLBACK 小节规定如果无法建立因特网连接，则用 "offline.html" 替代 /html5/ 目录中的所有文件：

FALLBACK:

/html/ /offline.html

# 第一个 URI 是资源，第二个是替补。

## 更新缓存

一旦应用被缓存，它就会保持缓存直到发生下列情况：

* 用户清空浏览器缓存
* manifest 文件被修改（参阅下面的提示）
* 由程序来更新应用缓存

## 实例 - 完整的 Manifest 文件

CACHE MANIFEST

# 2012-02-21 v1.0.0

/theme.css

/logo.gif

/main.js

NETWORK:

login.php

FALLBACK:

/html/ /offline.html

提示：以 "#" 开头的是注释行，但也可满足其他用途。应用的缓存会在其 manifest 文件更改时被更新。如果您编辑了一幅图片，或者修改了一个 JavaScript 函数，这些改变都不会被重新缓存。更新注释行中的日期和版本号是一种使浏览器重新缓存文件的办法。

# WEB Worker

当在 HTML 页面中执行脚本时，页面的状态是不可响应的，直到脚本已完成。

web worker 是运行在后台的 JavaScript，独立于其他脚本，不会影响页面的性能。您可以继续做任何愿意做的事情：点击、选取内容等等，而此时 web worker 在后台运行。

**实例：**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>菜鸟教程(runoob.com)</title>

</head>

<body>

<p>计数： <output id="result"></output></p>

<button onclick="startWorker()">开始工作</button>

<button onclick="stopWorker()">停止工作</button>

<p><strong>注意：</strong> Internet Explorer 9 及更早 IE 版本浏览器不支持 Web Workers.</p>

<script>

var w;

function startWorker() {

if(typeof(Worker) !== "undefined") {

if(typeof(w) == "undefined") {

w = new Worker("demo\_workers.js");

}

w.onmessage = function(event) {

document.getElementById("result").innerHTML = event.data;

};

} else {

document.getElementById("result").innerHTML = "抱歉，你的浏览器不支持 Web Workers...";

}

}

function stopWorker()

{

w.terminate();

w = undefined;

}

</script>

</body>

</html>

**demo\_workers.js**

var i=0;

function timedCount()

{

i=i+1;

postMessage(i);

setTimeout("timedCount()",500);

}

timedCount();

# 服务器发送事件

**服务器发送事件(Server-Sent Events)**

Server-Sent 事件指的是网页自动获取来自服务器的更新。

以前也可能做到这一点，前提是网页不得不询问是否有可用的更新。通过服务器发送事件，更新能够自动到达。

例子：Facebook/Twitter 更新、股价更新、新的博文、赛事结果等。

**实例：**

var source=new EventSource("demo\_sse.php");

source.onmessage=function(event)

{

document.getElementById("result").innerHTML+=event.data + "<br>";

};

**php**

<?php

header('Content-Type: text/event-stream');

header('Cache-Control: no-cache');

$time = date('r');

echo "data: The server time is: {$time}\n\n";

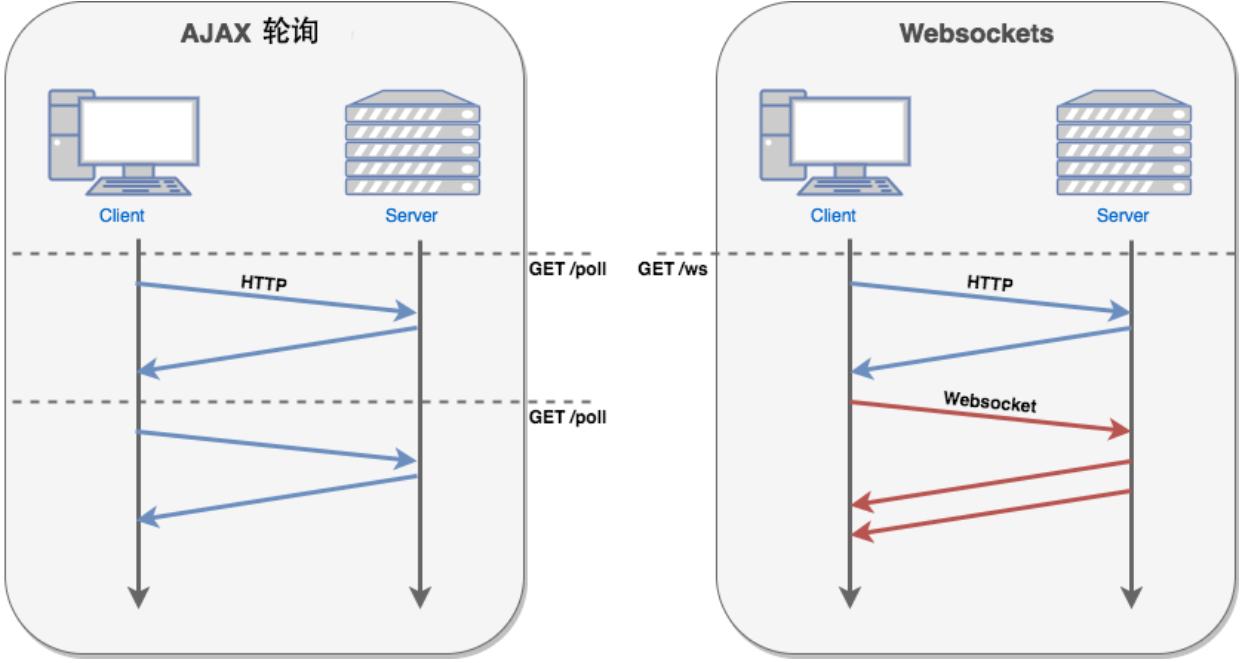
flush();

?>



# WebSocket

* WebSocket 是 HTML5 开始提供的一种在单个 TCP 连接上进行全双工通讯的协议。
* WebSocket 使得客户端和服务器之间的数据交换变得更加简单，允许服务端主动向客户端推送数据。在 WebSocket API 中，浏览器和服务器只需要完成一次握手，两者之间就直接可以创建持久性的连接，并进行双向数据传输。
* 在 WebSocket API 中，浏览器和服务器只需要做一个握手的动作，然后，浏览器和服务器之间就形成了一条快速通道。两者之间就直接可以数据互相传送。
* 现在，很多网站为了实现推送技术，所用的技术都是 Ajax 轮询。轮询是在特定的的时间间隔（如每1秒），由浏览器对服务器发出HTTP请求，然后由服务器返回最新的数据给客户端的浏览器。这种传统的模式带来很明显的缺点，即浏览器需要不断的向服务器发出请求，然而HTTP请求可能包含较长的头部，其中真正有效的数据可能只是很小的一部分，显然这样会浪费很多的带宽等资源。
* HTML5 定义的 WebSocket 协议，能更好的节省服务器资源和带宽，并且能够更实时地进行通讯。



WebSocket 协议本质上是一个基于 TCP 的协议

为了建立一个 WebSocket 连接，客户端浏览器首先要向服务器发起一个 HTTP 请求，这个请求和通常的 HTTP 请求不同，包含了一些附加头信息，其中附加头信息"Upgrade: WebSocket"表明这是一个申请协议升级的 HTTP 请求，服务器端解析这些附加的头信息然后产生应答信息返回给客户端，客户端和服务器端的 WebSocket 连接就建立起来了，双方就可以通过这个连接通道自由的传递信息，并且这个连接会持续存在直到客户端或者服务器端的某一方主动的关闭连接。