# 概述

[www.caniuse.com](http://www.caniuse.com)网站按照浏览器的版本提供了详尽的HTML5功能支持情况。若用户通过浏览器访问[www.html5test.com](http://www.html5test.com)的话，该网站会直接显示用户浏览器对HTML5规范的支持情况。 P7

Modernizr，一个js库，它提供了非常先进的HTML5和CSS3的检测功能。 P7

<!DOCTYPE> 声明位于文档中的最前面的位置，处于 <html> 标签之前。此标签可告知浏览器文档使用哪种 HTML 或 XHTML 规范。http://www.w3school.com.cn/tags/tag\_doctype.asp

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title> </title>

</head>

<body>

</body>

</html>

P8

浏览器会根据DOCTYPE来识别该使用哪种模式（怪异、近标准、标准）。P9

内嵌（audio、canvas等）、流（form、hl等）、标题（h1、h2等）、交互（button等）、元数据（script、style等）、短语（mark、kbd、sub等）、片段（article、aside、title等），上述所有类型的元素都可以通过CSS来设定样式。 P9

# Canvas API

如果需要在ie中使用canvas，可以选择explorercanvas开源项目，需要先判断浏览器是否是ie，如果是则在页面中嵌入script标签来加载explorercanvas。 p25

判断浏览器是否支持canvas

try{

document.creatElement("canvas").getContext("2d");

//supported

}catch (e){

//not supported

}

p26

context.save();//这样即使进行了绘制和变换操作，也可以恢复到初始状态 p30

context.translate(70, 140); 通过这个操作，当平移发生时，我们提供的变换坐标会被加到结果坐标上，结果就是将要绘制的对角线移动到新的位置上 p30

context.restore();//**后续的**canvas操作就不会被刚才的平移操作影响 p30

# Geolocation API

位置信息是通过支持html5地理定位功能的底层设备（电脑、手机等）提供给浏览器。 P72

HTML5 Geolocation API不指定设备使用哪种底层技术来定位应用程序得用户。相反，它只是用于检测位置信息的API，而且通过该API检索到的数据只具有某种程度的精确度。并不能保证设备返回的实际位置是精确的。 P73

设备使用的数据来源：ip地址、gps、手机、用户自定义。 p74-75

Ip：任何地方都可用；不精确

Gps：精确；定位时间长耗电大，室内效果不好，需要额外硬件设备

Wifi：精确，室内可用，简单快捷；无线接入点少的地区效果不好

手机定位：优点通wifi；需要能够访问手机或modem的设备，基站少效果不好

自定义：可能不准确，特别是位置变更后

不调用html5Geolocation代码，就不会触发隐私保护机制 p77

定位你的地址原理都是一样的：先寻找你的WiFi接入点和你周围的WiFi接入点来确定位置，如果你周围没有WiFi而且你的电脑也是通过有线线路连接互联网的，那么就是通过IP确定位置了，这样可能就不如WiFi定位来的准。

<http://blog.csdn.net/huanghr_1/article/details/6114400>

参考：<http://www.ivershuo.com/2010/03/geolocation/>

# Communication API

跨文档消息通信可以确保**iframe**、**标签页**、**窗口间**安全地进行跨源通信。它把postMessag API定义为发送消息的标准方式。 p94

当某个消息到达时，通过**检查消息的来源**来决定是否对这条消息进行处理。 p94

父页面和部件通过把彼此的源加到可信源的白名单中，就能收到来自对方的信息。 P96

同源文档间通信时也推荐使用postMessage。 P96

源（origin）由规则（scheme）、主机（host）、端口（port）组成。 P96

跨源通信通过源来确定发送者，这就使得接收方可以忽略或者拒绝来自不可信任源（白名单之外）的消息。 P96

通过调用**目标页面**window对象中的postMessage()函数发送消息。 P98

# WebSockets API

基于同一底层TCP/IP连接在客户端和服务器之间的初始握手阶段，讲HTTP协议升级至WebSocket协议，WebSocke连接就建立完成了。 P116

# WebWorkers API

对多线程支持非常好，可以充分利用多核CPU带来的优势。 p158

Web Workers不能直接访问Web页面和DOM API。 P158

Web workers初始化时会接受一个js文件url地址，其中包含了供Worker执行的代码。可以是相对或者绝对路径，这要同源（协议、主机、端口）即可（不能跨域？）。 p160

使用terminate函数或者启用它的页面终止webworkers，终止后不能被重新启用，但可以使用同样的url创建一个新的worker。P162

# WebStorage API

Cookie在服务器和客户端间来回传送文本值得内置机制。 P175

许多浏览器（ff不支持）不支持从文件系统直接访问文件式的SessionStorage。 P177

只要网页是同源的（规则、主机、端口），基于相同的键，我们都能在其他网页中获得设置在sessionStorage上的数据。在对同一页面后续多次加载的情况也是如此。 P179

当用户关闭窗口或浏览器，sessionStorage数据将被清除。P179

sessionStorage非常适合用于短时存在的流程中，如对话框和向导。如果数据需要存储在多个页面中，同时又不希望用户下一次访问应用程序时重新部署，则可将这些数据存储在sessionStorage中。P179

sessionStorage和localStorage区别：s数据会保存到存储它的窗口和标签页关闭时，数据只在**构建它们**的窗口或标签页内可见。l数据的生命周期比窗口或浏览器的生命期长，数据可被同源的**每个**窗口和标签页共享。P180

如果用户已经关闭了网站的存储（？），或者存储已达到其最大容量，那么此时设置数据将抛出QUOTA\_EXCEEDED\_ERROR错误。P181

删除数据项时不会将原有数据作为结果返回。在删除前请确保已经存储了相应数据副本。p181

HTML5规范中建议浏览器允许每组（每组？）同源页面使用5MB空间。当达到空间配额时，浏览器提示用户分配更多空间。但实际浏览器表现各不相同。P182

某些复杂情况下，多个网页、标签页或者Worker都需要访问存储的数据。此时，应用程序可能会在存储数据被修改后触发一系列操作。Web Storage可以将数据更新通知发送给感兴趣的监听者。无论监听窗口本身是否存储过数据，与执行存储操作的窗口同源的每个窗口的window对象上都会触发WebStorage事件。P182

数据库api细节尚在完善，并有多个方案，Web SQL Database是其中之一。P195

Web SQL Database允许应用程序通过一个异步js接口访问SQLite数据库。 P195

由于标准（？）认定直接执行SQL语句不可取，Web SQL Database已被较新的规范—索引数据库（Indexed Database）所取代。 P195

JSON是一种将对象与字符串可以相互表示的数据转换标准。P198