目录

[Html 1](#_Toc507600412)

[localStorage 和 sessionStorage 1](#_Toc507600413)

[Cookie 、localStorage 和 sessionStorage对比图 2](#_Toc507600414)

[indexDb 2](#_Toc507600415)

[重要概念 2](#_Toc507600416)

[object store 3](#_Toc507600417)

[重要属性 3](#_Toc507600418)

[使用事务 3](#_Toc507600419)

[方法 4](#_Toc507600420)

[过滤器 4](#_Toc507600421)

[WebSocket 5](#_Toc507600422)

[基础搭建 5](#_Toc507600423)

[WebSocket 属性 6](#_Toc507600424)

[WebSocket 事件 6](#_Toc507600425)

[WebSocket 方法 6](#_Toc507600426)

[Css 7](#_Toc507600427)

[单位 7](#_Toc507600428)

[Grid 8](#_Toc507600429)

[属性 9](#_Toc507600430)

[fr 单位(等分) 10](#_Toc507600431)

[repeat() 函数 10](#_Toc507600432)

# Html5特性

## <http://www.w3school.com.cn/html5/index.asp>

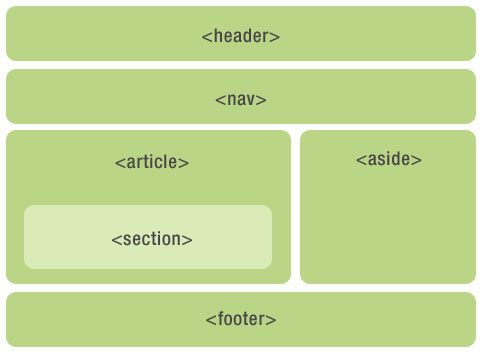
## 1.语义特性（更容易阅读和 和对于SEO的友好度增加）

<http://www.cnblogs.com/mq0036/p/3954172.html>

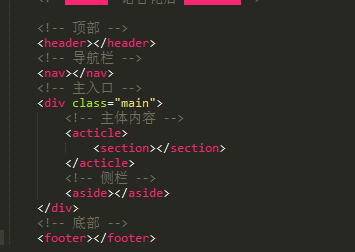
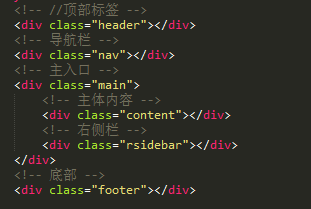
http://www.cnblogs.com/yuzhongwusan/archive/2011/11/17/2252208.html

HTML5赋予网页更好的意义和结构（添加了些新标签 减少了div滥用的情况）

但是在低版本的浏览器中新标签可能不被识别



左边为元素div+css 右边为语意化后的标签



# 本地存储特性（离线存储）

## localStorage 和 sessionStorage

**客户端存储数据的两个对象为**：

**localStorage - 没有时间限制的数据存储（localStorage 对象存储的数据没有时间限制。第二天、第二周或下一年之后，数据依然可用）**

**sessionStorage - 针对一个 session 的数据存储**

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)**;

**谷歌可以在这里看到值**

**控制台 console.log(localStorage)，console.log(sessionStorage) 也可以直接看**



### Cookie 、localStorage 和 sessionStorage对比图



## indexDb IndexedDB （支持存储大量数据，不存在大小限制）

<http://www.ibm.com/developerworks/cn/web/wa-indexeddb/>

<http://www.cnblogs.com/dolphinX/p/3415761.html> 操作网址

<https://www.ibm.com/developerworks/cn/web/wa-indexeddb/> 基础

<http://www.cnblogs.com/dolphinX/p/3416889.html>

<http://www.cnblogs.com/dolphinX/p/3415761.html>

## 重要概念

一个网站可能有一个或多个 IndexedDB 数据库，每个数据库必须具有惟一的名称。

一个数据库可包含一个或多个对象存储。一个对象存储（由一个名称惟一标识）是一个记录集合。每个记录有一个键 和一个值。该值是一个对象，可拥有一个或多个属性。键可能基于某个键生成器，从一个键路径衍生出来，或者是显式设置。一个键生成器自动生成惟一的连续正整数。键路径定义了键值的路径。它可以是单个 JavaScript 标识符或多个由句点分隔的标识符。

**在IndexedDB大部分操作并不是我们常用的调用方法，返回结果的模式，而是请求——响应的模式**

IndexedDB 可以实现数据库的增删改查

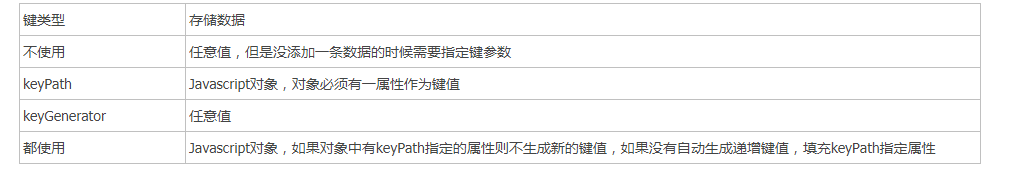
对于不同的网站，数据存储于不同的区域，并且一个网站只能访问其自身的数据。

。

### object store

有了数据库后我们自然希望创建一个表用来存储数据，但indexedDB中没有表的概念，而是objectStore，一个数据库中可以包含多个objectStore，objectStore是一个灵活的数据结构，可以存放多种类型数据。也就是说一个objectStore相当于一张表，里面存储的每条数据和一个键相关联。

我们可以使用每条记录中的某个指定字段作为键值（keyPath），也可以使用自动生成的递增数字作为键值（keyGenerator），也可以不指定。选择键的类型不同，objectStore可以存储的数据结构也有差异



## 重要属性

result： 返回的数据

onerror: 请求失败的回调函数句柄

onsuccess:请求成功的回调函数句柄

onupgradeneeded:请求数据库版本变化句柄

## 使用事务

您需要使用事务在对象存储上执行所有读取和写入操作。类似于关系数据库中的事务的工作原理，IndexedDB 事务提供了数据库写入操作的一个原子集合，这个集合要么完全提交，要么完全不提交。IndexedDB 事务还拥有数据库操作的一个中止和提交工具。

| **模式** | **描述** |
| --- | --- |
| readonly | 提供对某个对象存储的只读访问，在查询对象存储时使用。 |
| readwrite | 提供对某个对象存储的读取和写入访问权。 |
| versionchange | 提供读取和写入访问权来修改对象存储定义，或者创建一个新的对象存储 |

## 方法

add(Obj)：为当前数据表增加记录。参数为JSON类型，同时返回一个IDBRequest对象

clear()：清除数据表。

delete(recordKey)：根据指定的key值删除记录。

get(recordKey)：根据指定的key值获取完整数据。

getAll()：获取当前数据表中所有数据。

count(KeyRange)：统计。

put(obj，key)：根据指定key值修改数据表数据。

## 过滤器



## .网页多媒体特性(Class: MULTIMEDIA)

支持网页端的Audio、Video等多媒体功能， 与网站自带的APPS，摄像头，影音功能相得益彰。

### 音频



<audio controls="controls">

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

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

您的浏览器不支持 video 标签 请升级至最新版本浏览器

</audio>

### 视频（video）

基本只要.MP4（ie,苹果等）和.webm（谷歌，火狐等）两个格式就可以兼容所有浏览器了



Webkit 内核的最新浏览器 的都支持webm

.Webm 是谷歌开源的一个开放、免费的媒体文件格式,将来可能会成为主流

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

<source src="movie.mp4" >

<source src="movie.webm">

您的浏览器不支持 video 标签 请升级至最新版本浏览器

</video>

注：允许多个 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

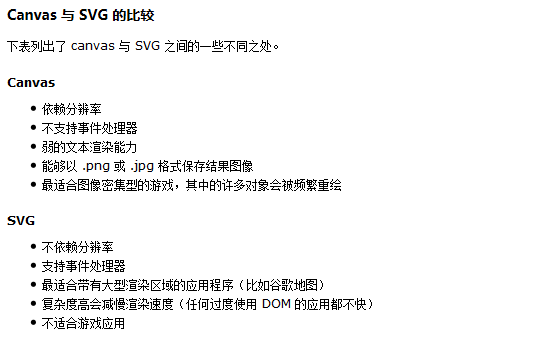
## 三维、图形及特效特性（WebGL,CSS3的3D功能等等）

基于SVG、Canvas、WebGL及CSS3的3D功能，用户会惊叹于在浏览器中，所呈现的惊人视觉效果。

<http://labs.gooengine.com/pearl-boy/> canvas 3D演示

<http://www.html5tricks.com/demo/html5-svg-firebird/index.html> svg

<http://www.html5tricks.com/demo/html5-svg-world-map/index.html> svg地图



## Web Worket（推荐1-3 多线程 ie兼容不好）

 大家都知道js是单线程的，在上一段js执行结束之前，后面的js绝对不会执行,遇到运行效率慢的js的时候,最坏的结果就是在很长的一段时间内用户都不能进行任何操作，所以，退出，关闭。

html5 提出了一个名词：web Worker,按照官方的解释：**web worker 是运行在后台的 JavaScript**，不会影响页面的性能。也就可以理解为两段js同时执行

注意： web worker是不支持跨域加载JS的

**前台页面:**

通过 new Worker( js) 加载一个JS文件来创建一个worker并返回一个worker实例。

通过worker.postMessage( data ) 方法来向worker发送数据。

通过worker.onmessage方法来接收worker发送过来的数据。

worker.terminate() 可以终止worker

**后台js：**

通过postMessage( data ) 方法来向主线程发送数据。

绑定onmessage方法来接收主线程发送过来的数据。

**总结**

web worker看起来很美好，但处处是魔鬼。

**我们可以做什么：**

    1.可以加载一个JS进行大量的复杂计算而不挂起主进程，并通过postMessage，onmessage进行通信

    2.可以在worker中通过importScripts(url)加载另外的脚本文件

    3.可以使用 setTimeout(), clearTimeout(), setInterval(), and clearInterval()

    4.可以使用XMLHttpRequest来发送请求

    5.可以访问navigator的部分属性

**有那些局限性：**

    1.不能跨域加载JS

    2.worker内代码不能访问DOM

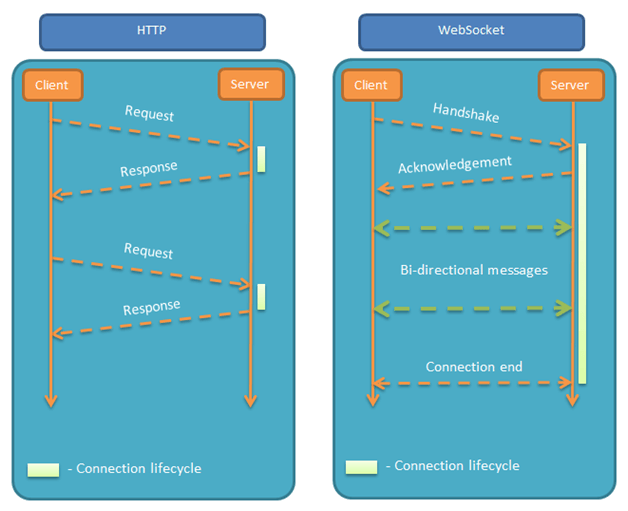
    3.各个浏览器对Worker的实现不大一致，例如FF里允许worker中创建新的worker,而Chrome中就不行

    4.不是每个浏览器都支持这个新特性

# WebSocket

http://www.ruanyifeng.com/blog/2017/05/websocket.html

它的最大特点就是，服务器可以主动向客户端推送信息，客户端也可以主动向服务器发送信息，是真正的双向平等对话，属服务器推送技术的一种



**其他特点包括**：

**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。

## 基础搭建

**1.先装一个nodeJs的模块**

**2.进入目录输入 npm init**

**3.npm install --save nodejs-websocket回车就可以安装好了**

**然后就可以开始建立服务器了，因为有了nodejs-websocket模块，所以很多工作都不用我们自己做，直接调用别人封装好的方法就行了**

## 创建对象

var ws = new WebSocket('ws://127.0.0.1:8082');

## WebSocket 属性

以下是 WebSocket 对象的属性。假定我们使用了以上代码创建了 Socket 对象：

|  |  |
| --- | --- |
| **属性** | **描述** |
| Socket.readyState | 只读属性 **readyState** 表示连接状态，可以是以下值：   * 0 - 表示连接尚未建立。 * 1 - 表示连接已建立，可以进行通信。 * 2 - 表示连接正在进行关闭。 * 3 - 表示连接已经关闭或者连接不能打开。 |
| Socket.bufferedAmount | 只读属性 **bufferedAmount** 已被 send() 放入正在队列中等待传输，但是还没有发出的 UTF-8 文本字节数。 |

## WebSocket 事件

以下是 WebSocket 对象的相关事件。假定我们使用了以上代码创建了 Socket 对象：

|  |  |  |  |
| --- | --- | --- | --- |
| **事件** | **事件处理程序** | | **描述** |
| open | | Socket.onopen | 连接建立时触发 |
| message | | Socket.onmessage | 客户端接收服务端数据时触发 |
| error | | Socket.onerror | 通信发生错误时触发 |
| close | | Socket.onclose | 连接关闭时触发 |

## WebSocket 方法

以下是 WebSocket 对象的相关方法。假定我们使用了以上代码创建了 Socket 对象：

|  |  |
| --- | --- |
| **方法** | **描述** |
| Socket.send() | 使用连接发送数据 |
| Socket.close() | 关闭连接 |

WebSocket 对象作为一个构造函数，用于新建 WebSocket 实例。

**var ws = new WebSocket('ws://localhost:8080');**

执行上面语句之后，客户端就会与服务器进行连接。

# Css

## 参考

<http://www.runoob.com/css3/css3-user-interface.html> 网站

http://www.runoob.com/css3/css3-gradients.html

<http://www.w3school.com.cn/css3/css3_animation.asp>   动画

<http://www.w3school.com.cn/cssref/pr_transform.asp>   旋转

<http://www.w3school.com.cn/cssref/pr_transition.asp>    过渡

## 厂商前缀

-moz- /\* Firefox 4 \*/

-webkit- /\* Safari 和 Chrome \*/

-o- /\* Opera \*/

## 权级

1.  内联样式表的权值最高 1000；

2.  ID 选择器的权值为 100

3.  Class 类选择器的权值为 10

4.  HTML 标签选择器的权值为 1

A  选择器都有一个权值，权值越大越优先；

B  当权值相等时，后出现的样式表设置要优于先出现的样式表设置；

C  创作者的规则高于浏览者：即网页编写者设置的CSS 样式的优先权高于浏览器所设置的样式；

D  继承的CSS 样式不如后来指定的CSS 样式；

E  在同一组属性设置中标有“!important”规则的优先级最大；示例如下：

## 单位

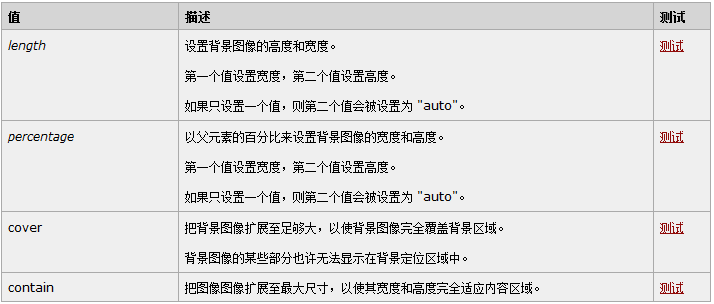
|  |  |
| --- | --- |
| **单位** | **含义** |
| em | 相对于父元素的字体大小 |
| ex | 相对于小写字母"x"的高度 |
| gd | 一般用在东亚字体排版上，这个与英文并无关系 |
| rem | 相对于根元素字体大小 |
| vw | 相对于视窗的宽度：视窗宽度是100vw |
| vh | 相对于视窗的高度：视窗高度是100vh |
| vm | 相对于视窗的宽度或高度，取决于哪个更小 |
| ch | 相对于0尺寸 |
| px | 相对于屏幕分辨率而不是视窗大小：通常为1个点或1/72英寸 |
| in | inch, 表英寸 |
| cm | centimeter, 表厘米 |
| mm | millimeter, 表毫米 |
| pt | 1/72英寸 |
| pc | 12点活字，或1/12点 |
| % | 相对于父元素。正常情况下是通过属性定义自身或其他元素 |

选择器

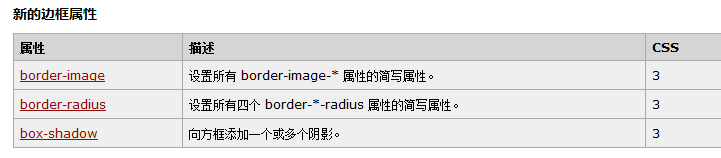
|  |  |  |  |
| --- | --- | --- | --- |
| [.*class*](http://www.runoob.com/cssref/sel-class.html) | .intro | 选择所有class="intro"的元素 | 1 |
| [#*id*](http://www.runoob.com/cssref/sel-id.html) | #firstname | 选择所有id="firstname"的元素 | 1 |
| [\*](http://www.runoob.com/cssref/sel-all.html) | \* | 选择所有元素 | 2 |
| [*element*](http://www.runoob.com/cssref/sel-element.html) | p | 选择所有<p>元素 | 1 |
| [*element,element*](http://www.runoob.com/cssref/sel-element-comma.html) | div,p | 选择所有<div>元素和<p>元素 | 1 |
| [*element* *element*](http://www.runoob.com/cssref/sel-element-element.html) | div p | 选择<div>元素内的所有<p>元素 | 1 |
| [*element*>*element*](http://www.runoob.com/cssref/sel-element-gt.html) | div>p | 选择所有父级是 <div> 元素的 <p> 元素 | 2 |
| [*element*+*element*](http://www.runoob.com/cssref/sel-element-pluss.html) | div+p | 选择所有紧接着<div>元素之后的<p>元素 | 2 |
| [[*attribute*]](http://www.runoob.com/cssref/sel-attribute.html) | [target] | 选择所有带有target属性元素 | 2 |
| [[*attribute*=*value*]](http://www.runoob.com/cssref/sel-attribute-value.html) | [target=-blank] | 选择所有使用target="-blank"的元素 | 2 |
| [[*attribute*~=*value*]](http://www.runoob.com/cssref/sel-attribute-value-contains.html) | [title~=flower] | 选择标题属性包含单词"flower"的所有元素 | 2 |
| [[*attribute*|=*language*]](http://www.runoob.com/cssref/sel-attribute-value-lang.html) | [lang|=en] | 选择一个lang属性的起始值="EN"的所有元素 | 2 |
| [:link](http://www.runoob.com/cssref/sel-link.html) | a:link | 选择所有未访问链接 | 1 |
| [:visited](http://www.runoob.com/cssref/sel-visited.html) | a:visited | 选择所有访问过的链接 | 1 |
| [:active](http://www.runoob.com/cssref/sel-active.html) | a:active | 选择活动链接 | 1 |
| [:hover](http://www.runoob.com/cssref/sel-hover.html) | a:hover | 选择鼠标在链接上面时 | 1 |
| [:focus](http://www.runoob.com/cssref/sel-focus.html) | input:focus | 选择具有焦点的输入元素 | 2 |
| [:first-letter](http://www.runoob.com/cssref/sel-firstletter.html) | p:first-letter | 选择每一个<P>元素的第一个字母 | 1 |
| [:first-line](http://www.runoob.com/cssref/sel-firstline.html) | p:first-line | 选择每一个<P>元素的第一行 | 1 |
| [:first-child](http://www.runoob.com/cssref/sel-firstchild.html) | p:first-child | 指定只有当<p>元素是其父级的第一个子级的样式。 | 2 |
| [:before](http://www.runoob.com/cssref/sel-before.html) | p:before | 在每个<p>元素之前插入内容 | 2 |
| [:after](http://www.runoob.com/cssref/sel-after.html) | p:after | 在每个<p>元素之后插入内容 | 2 |
| [:lang(*language*)](http://www.runoob.com/cssref/sel-lang.html) | p:lang(it) | 选择一个lang属性的起始值="it"的所有<p>元素 | 2 |
| [*element1*~*element2*](http://www.runoob.com/cssref/sel-gen-sibling.html) | p~ul | 选择p元素之后的每一个ul元素 | 3 |
| [[*attribute*^=*value*]](http://www.runoob.com/cssref/sel-attr-begin.html) | a[src^="https"] | 选择每一个src属性的值以"https"开头的元素 | 3 |
| [[*attribute*$=*value*]](http://www.runoob.com/cssref/sel-attr-end.html) | a[src$=".pdf"] | 选择每一个src属性的值以".pdf"结尾的元素 | 3 |
| [[*attribute*\*=*value*]](http://www.runoob.com/cssref/sel-attr-contain.html) | a[src\*="runoob"] | 选择每一个src属性的值包含子字符串"runoob"的元素 | 3 |
| [:first-of-type](http://www.runoob.com/cssref/sel-first-of-type.html) | p:first-of-type | 选择每个p元素是其父级的第一个p元素 | 3 |
| [:last-of-type](http://www.runoob.com/cssref/sel-last-of-type.html) | p:last-of-type | 选择每个p元素是其父级的最后一个p元素 | 3 |
| [:only-of-type](http://www.runoob.com/cssref/sel-only-of-type.html) | p:only-of-type | 选择每个p元素是其父级的唯一p元素 | 3 |
| [:only-child](http://www.runoob.com/cssref/sel-only-child.html) | p:only-child | 选择每个p元素是其父级的唯一子元素 | 3 |
| [:nth-child(*n*)](http://www.runoob.com/cssref/sel-nth-child.html) | p:nth-child(2) | 选择每个p元素是其父级的第二个子元素 | 3 |
| [:nth-last-child(*n*)](http://www.runoob.com/cssref/sel-nth-last-child.html) | p:nth-last-child(2) | 选择每个p元素的是其父级的第二个子元素，从最后一个子项计数 | 3 |
| [:nth-of-type(*n*)](http://www.runoob.com/cssref/sel-nth-of-type.html) | p:nth-of-type(2) | 选择每个p元素是其父级的第二个p元素 | 3 |
| [:nth-last-of-type(*n*)](http://www.runoob.com/cssref/sel-nth-last-of-type.html) | p:nth-last-of-type(2) | 选择每个p元素的是其父级的第二个p元素，从最后一个子项计数 | 3 |
| [:last-child](http://www.runoob.com/cssref/sel-last-child.html) | p:last-child | 选择每个p元素是其父级的最后一个子级。 | 3 |
| [:root](http://www.runoob.com/cssref/sel-root.html) | :root | 选择文档的根元素 | 3 |
| [:empty](http://www.runoob.com/cssref/sel-empty.html) | p:empty | 选择每个没有任何子级的p元素（包括文本节点） | 3 |
| [:target](http://www.runoob.com/cssref/sel-target.html) | #news:target | 选择当前活动的#news元素（包含该锚名称的点击的URL） | 3 |
| [:enabled](http://www.runoob.com/cssref/sel-enabled.html) | input:enabled | 选择每一个已启用的输入元素 | 3 |
| [:disabled](http://www.runoob.com/cssref/sel-disabled.html) | input:disabled | 选择每一个禁用的输入元素 | 3 |
| [:checked](http://www.runoob.com/cssref/sel-checked.html) | input:checked | 选择每个选中的输入元素 | 3 |
| [:not(*selector*)](http://www.runoob.com/cssref/sel-not.html) | :not(p) | 选择每个并非p元素的元素 | 3 |
| [::selection](http://www.runoob.com/cssref/sel-selection.html) | ::selection | 匹配元素中被用户选中或处于高亮状态的部分 | 3 |
| [:out-of-range](http://www.runoob.com/cssref/sel-out-of-range.html) | :out-of-range | 匹配值在指定区间之外的input元素 | 3 |
| [:in-range](http://www.runoob.com/cssref/sel-in-range.html) | :in-range | 匹配值在指定区间之内的input元素 | 3 |
| [:read-write](http://www.runoob.com/cssref/sel-read-write.html) | :read-write | 用于匹配可读及可写的元素 | 3 |
| [:read-only](http://www.runoob.com/cssref/sel-read-only.html) | :read-only | 用于匹配设置 "readonly"（只读） 属性的元素 | 3 |
| [:optional](http://www.runoob.com/cssref/sel-optional.html) | :optional | 用于匹配可选的输入元素 | 3 |
| [:required](http://www.runoob.com/cssref/sel-required.html) | :required | 用于匹配设置了 "required" 属性的元素 | 3 |
| [:valid](http://www.runoob.com/cssref/sel-valid.html) | :valid | 用于匹配输入值为合法的元素 | 3 |
| [:invalid](http://www.runoob.com/cssref/sel-invalid.html) | :invalid | 用于匹配输入值为非法的元素 | 3 |

## background-size（背景 ie9+）

http://www.w3school.com.cn/tiy/c.asp?f=css\_background-size&p=7



border-radius | box-shadow | border-image （边框属性 ie9+ 需要写厂家前缀）



### 6.2 word-wrap:break-word; （英文文本换行 ie9+）

## 媒体查询@media （排版能手 但是手机端不推荐写太多 媒体查询 IE9+ ）

<http://www.runoob.com/try/try2.php?filename=trycss3_mediaqueries_ex4>

使用 @media 查询，你可以针对不同的媒体类型定义不同的样式。

@media 可以针对不同的屏幕尺寸设置不同的样式，特别是如果你需要设置设计响应式的页面，@media 是非常有用的。

当你重置浏览器大小的过程中，页面也会根据浏览器的宽度和高度重新渲染页面。



例

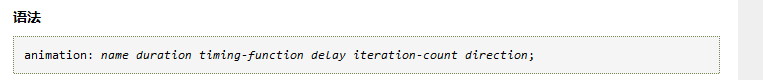
## @media screen and (min-width: 480px) {     body {         background-color: lightgreen;     } }

### 6.3 animation属性介绍 (动画属性 ie10+ 必须加厂家前缀)

<http://www.w3school.com.cn/css3/css3_animation.asp>



属性简写顺序：

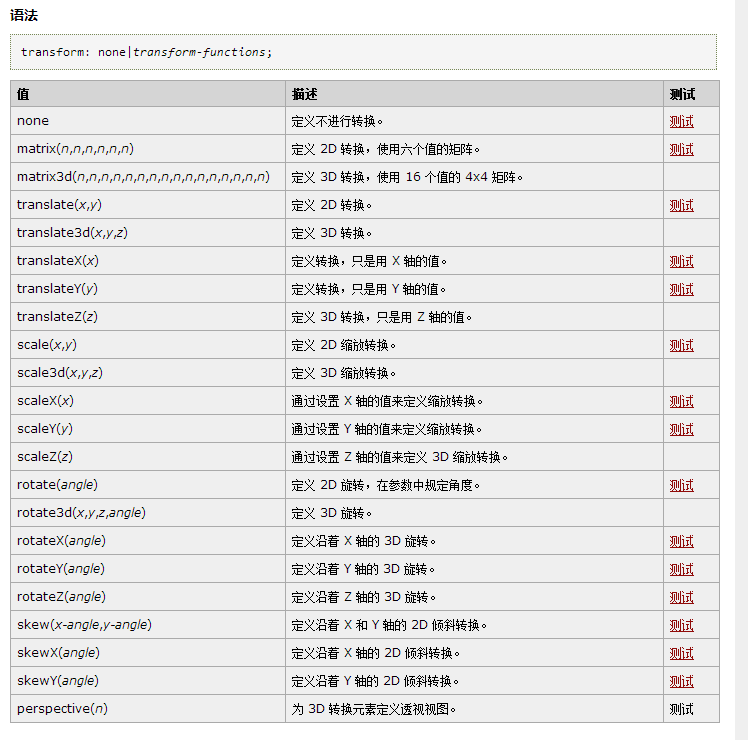


animation: *namedurationtiming-functiondelayiteration-countdirection*;

### 6.4 transform属性 （旋转属性ie10+ 必须加厂家前缀）

<http://www.w3school.com.cn/cssref/pr_transform.asp>

<http://www.runoob.com/css3/css3-3dtransforms.html> 3d转换



### 6.5 transition属性（过渡属性ie10+ 必须加厂家前缀）

<http://www.w3school.com.cn/cssref/pr_transition.asp>



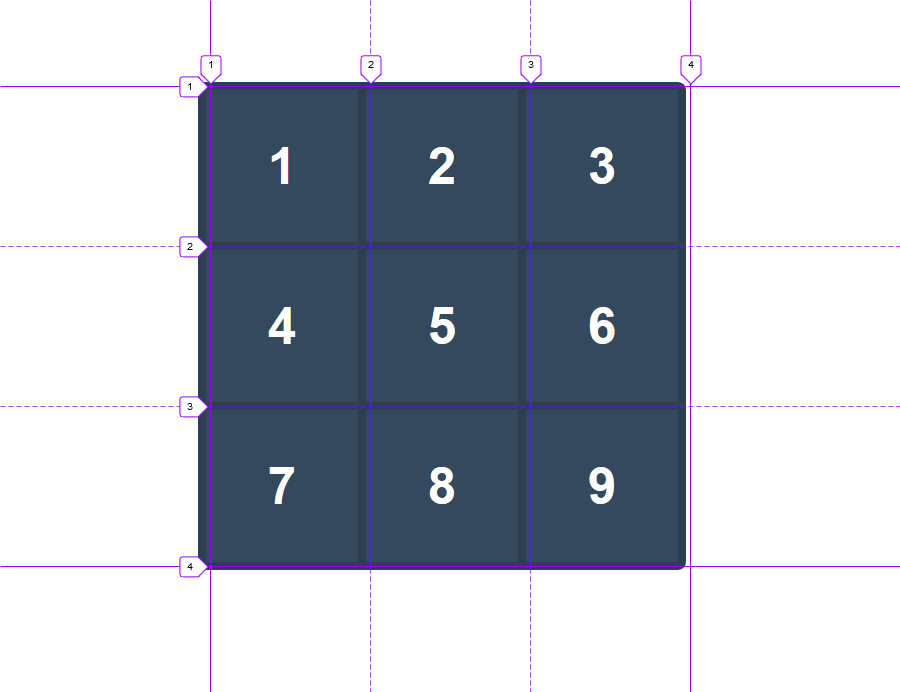
## Grid

<http://www.css88.com/archives/8661>

Grid(网格) 模块为 display 属性提供了一个新的值：grid。当你将任何元素的 display 属性设置为 grid 时，那么这个元素就是一个 网格容器(grid container)，它的所有直接子元素就成了 网格项(grid items)。。

注意

1. 使用disaplay: grid之后元素会默认占领一行（占满父元素）
2. 网格项的padding和margin都是向内计算（与box-sizing: border-box类型并且）



### 属性

#### 网格容器属性(grid container)

grid-template-rows：指定网格中的行数及行的高度

grid-template-columns： 指定网格中的列数及列的宽度

grid-template：grid-template-rows 和 grid-template-columns 的简写语法

例

1. {
2. /\*设置每个元素的宽度\*/
3. grid-template-columns: 100px 100px 100px;
4. /\*设置每个元素的高度,\*/
5. grid-template-rows: 100px 100px 100px 100px;
6. }

等价

1. {
2. //4列3行
3. grid-template: repeat(4, 100px) / repeat(3, 100PX);
4. }

## 网格项属性(grid items)

grid-column-start

grid-column-end

grid-row-start

grid-row-end

前两个属性对应于垂直网格线，也就是列网格线的开始和结束。 最后两个属性是指水平网格线，也就是行网格线的开始和结束

例

1. {
2. grid-row-start: 1;
3. grid-row-end: 2;
4. grid-column-start: 2;
5. grid-column-end: 3;
6. }

等价于

1. {
2. grid-row: 1 / 2;
3. grid-column: 2 / 3;
4. }

### fr 单位(等分)

fr 是为网格布局定义的一个新单位。它可以帮助你摆脱计算百分比，并将可用空间等分

### repeat() 函数

repeat 函数，就像任何一个循环重复多少次输出某个给定值。它有两个参数。第一个是迭代次数，第二个是要重复的值

### 兼容情况

