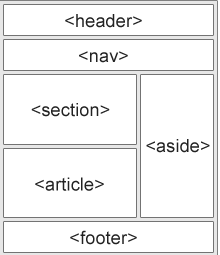
1. H5常见的语义化标签
2. header标签(文档的页眉部分, 介绍信息)
3. nav标签(导航链接部分, 导航栏)
4. main标签(文档主要内容部分, 一个页面只使用一次)
5. section标签(文档中的节, 具有相似主题的一组内容, 如新闻条目)
6. article标签(独立的自包含内容, 如论坛帖子, 包装文章等)
7. aside标签(附注栏, 如引述, 侧栏, 指向文章的一组链接, 广告, 相关产品列表等)
8. footer标签(文档或节的页脚, 如文档作者, 版权信息, 使用条款链接等)
9. 以上标签使用基本案例 -->

 可以使用main标签把例子中的section article aside标签包裹

1. 其他常用标签

(1)hgroup对页面或者区段(section)的标题进行组合

(2)address定义文档或者文章作者或拥有者的联系信息(一般成斜体展示)

(3)details标签用于描述文档或文档某个部分的细节。显示为产生一个下拉菜单, 只设置这一个标签时为显示一个为详细信息为标题的下拉菜单(配合summary标签)

(4)summary属性规定表格内容的摘要, 显示和普通文本没有任何视觉变化

##details和summary标签配合使用说明

details的第一个子标签summary为下拉菜单的主题

1. 处理语义化标签兼容问题
2. IE9以上版本也只是选择支持h5

会把语义化标签解析成inline元素, 兼容需要设置语义标签设置为display: block;

2>IE8以及以下版本完全不支持h5

全不能解析语义化标签, 兼容需要用js手动创建语义化标签, 在将需要为块级元素的标签的style中设为display: block;

3>IE兼容方案, 引入第三方库(解决繁琐操作, 手动JS声明标签)

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

注意：位置需要在使用标签之前(常用在IE8及以下版本兼容方案)

1. H5新增的input表单标签的type属性

1>email 会自动验证 输入的内容必须包含@(合法邮箱)

2>tel不会自动验证(面向全球服务) 移动端自动弹出数字键盘

3>url 会自动验证 输入必须包含http://(合法网站)

4>number只能输入数字(包含小数点) input框右边有+1, -1按钮

其有max属性和min属性

5>search 可以提供更好的体验(input框右边有删除全部内容按钮)

6>range范围 -->  其有max属性和min属性

7>color 拾色器-->

8>time时间 时分秒-->

9>datatime日期时分秒 -->

注意：大部分不支持只有苹果的safari

10>file文件

11>month月week星期

1. H5新增的input表单标签属性

1>placeholder=”要提示的内容” -->

2>autofocus=”on”(打开自动聚焦) off关闭

3>autocpmplate=”on”(打开历史记录) off关闭 注意:需要按提交按钮才会记录

4>required要求输入

5>pattern=”正则验证”(一般配合4>使用且用于验证手机号) -->



6>multiple可以选择多个东西(文件 type=”file”加上这个属性之后可选择多个文件, 也可用 于写入多个邮箱type=”emil”)

注意：一个input框输入多个邮箱时, 号分开

7>from=”form标签的id” 指定控件提交到哪个表单

8>list=”datalist标签的id”(不常用datalist标签的兼容性不好)

1. 多媒体

1>音频播放：audio标签的使用:

(1)属性：

|  |  |  |
| --- | --- | --- |
| [autoplay](http://www.w3school.com.cn/html5/att_audio_autoplay.asp" \o "HTML5 <audio> autoplay 属性) | autoplay | 如果出现该属性，则音频在就绪后马上播放。 |
| [controls](http://www.w3school.com.cn/html5/att_audio_controls.asp" \o "HTML5 <audio> controls 属性) | controls | 如果出现该属性，则向用户显示控件，比如播放按钮。 |
| [loop](http://www.w3school.com.cn/html5/att_audio_loop.asp" \o "HTML5 <audio> loop 属性) | loop | 如果出现该属性，则每当音频结束时重新开始播放。 |
| [preload](http://www.w3school.com.cn/html5/att_audio_preload.asp" \o "HTML5 <audio> preload 属性) | preload | 如果出现该属性，则音频在页面加载时进行加载，并预备播放。  如果使用 "autoplay"，则忽略该属性。 |
| [src](http://www.w3school.com.cn/html5/att_audio_src.asp" \o "HTML5 <audio> src 属性) | *url* | 要播放的音频的 URL。 |

(2)示例:播放音频

(2)示例:播放音频

<audio src="../mp3/See.mp3" controls autoplay></audio>

2>视频播放：video标签的使用

(1)属性：

|  |  |  |
| --- | --- | --- |
| [autoplay](http://www.w3school.com.cn/html5/att_video_autoplay.asp" \o "HTML5 <video> autoplay 属性) | autoplay | 如果出现该属性，则视频在就绪后马上播放。 |
| [controls](http://www.w3school.com.cn/html5/att_video_controls.asp" \o "HTML5 <video> controls 属性) | controls | 如果出现该属性，则向用户显示控件，比如播放按钮。 |
| [height](http://www.w3school.com.cn/html5/att_video_height.asp" \o "HTML5 <video> height 属性) | *pixels* | 设置视频播放器的高度。 |
| [loop](http://www.w3school.com.cn/html5/att_video_loop.asp" \o "HTML5 <video> loop 属性) | loop | 如果出现该属性，则当媒介文件完成播放后再次开始播放。 |
| [preload](http://www.w3school.com.cn/html5/att_video_preload.asp" \o "HTML5 <video> preload 属性) | preload | 如果出现该属性，则视频在页面加载时进行加载，并预备播放。  如果使用 "autoplay"，则忽略该属性。 |
| [src](http://www.w3school.com.cn/html5/att_video_src.asp" \o "HTML5 <video> src 属性) | *url* | 要播放的视频的 URL。 |
| [width](http://www.w3school.com.cn/html5/att_video_width.asp" \o "HTML5 <video> width 属性) | *pixels* | 设置视频播放器的宽度。 |

(2)视频播放

<video src="../mp3/561902ae6ac6e6649.mp4" controls></video>

(3)说明：由于版权等原因，不同的浏览器可支持播放的格式是不一样的



3>常用方法：

load() 加载、 play() 播放、 pause() 暂停

4>常用属性：

(1)currentTime 视频播放的当前进度、

(2)duration:视频的总时间 100000/60

(3)paused:视频播放的状态.

5>常用事件：

(1)oncanplay: 事件在用户可以开始播放视频/音频（audio/video）时触发。

(2)ontimeupdate:通过该事件来报告当前的播放进度.

(3)onended:播放完时触发—重置

注意：Jq没有提供对视频播放控件的方式，也就意味着如果要操作视频播放，必须使用原生的js方法—dom元素

1. H5操作元素扩展

1>H5新增表单事件

(1)oninput 只要内容改变就会触发该事件

(2)oninvalid 当前验证不通过时候触发一般用于改变input验证默认信

2>H5新增操作元素类

(1)dom.classList.add(“class”) 添加class

(2)dom.classList.romove('class') 移除class

(3)dom.classList.toggle('class') 切换class，有则移除，无则添加

注意：()只能操作一个class需要操作多个需重新创建

(4)dom.classList.contains('class') 检测是否存在class, 有则true, 无则false

(5)dom.classList.item(index) 找dom的第索引位class, 返回其类名

3>H5自定义属性

(1)设置自定义属性

##标签内设置(和标签添加class一样)

data-自定义属性名="设置的属性值"

##在js中设置

dom.dataset.自定义属性名 = “设置的属性值”

(2)获取其属性(js中获取)

##dom.dataset.自定义的属性名

(3)注意设置属性名时候的规范

##data-(标签内设置时), js中获取其属性驼峰式获取

1. H5新增接口

1>网络接口(事件, 与click用法一样)

(1)online 网络连接触发

(2)onoffline 网络断开触发

注意：为了兼容问题一般使用addEventListener绑定

2>全屏接口(方法)

(1)requestFullScreen(); 开启全屏显示

(2)cancelFullScreen(); 退出全屏显示

(3)fullScreenElement; 是否全屏状态

注意：(1)三者都有兼容问题

##谷歌需加上webkit, -->div.webkitRequestFullScreen()

##IE需加上ms,

##火狐须加上moz,

##opera需加上o

(2)requestFullScreen()选中某个dom操作,

cancelFullScreen()和fullScreenElement;需对document操作

3>FileReader接口

(1)readAsText()

##读取文本文件(需可以用Txt打开的文件),

##返回文本字符串

##默认编码为utf-8

(2)readAsBinaryString()

##读取任意类型的文件

##返回二进制字符串,

##一般不是读取文件给用户看, 而是存储文件(传递给后台, 后台接收后存储)

(3)readAsDataURL()

##读取文件获取一段以data开头的字符串, 这段字符串的本质是DataURL,

##DataURL是一种将文件(这个文件一般指的是图像或者能够嵌入到文档的文件

格式)嵌入到文档的方案, 能将服务器资源转化为base64编码 的字符形式, 且将这些内容直接存储到url中》》优化网站加载速度和执行效率

##无返回值, 需要传递一个参数(二进制的大文件, 如图文或者能嵌入文档的文件)

##读取完文件之后会将读取的结果存储在读取文件对象的result中

##可以使用onload事件判断是够读取完成文件

##使用方式 -->img(获取的imgDOM).src = reader(读取文件对象).result

##创建文件对象方式 -->var reader = new FileReader();

(4)abort() 中断文件读取

4>拖拽接口

(1)将需要拖拽的元素设置其属性draggable=”true”(图片和超链接默认可以拖拽)

(2)添加拖拽事件

##应用于拖拽元素上的事件(绑定在拖拽元素身上)

++ondrag 整个拖拽过程都会调用(持续触发)

++ondragstart 当拖拽开始时调用

++ondragleave 当鼠标离开拖拽元素时调用

++ondragend 当拖拽结束时调用

##应用于目标元素上的事件(绑定在目标元素身上)

++ondragenter 当拖拽元素进入时触发

++ondragover 当拖拽元素停留在目标元素时触发

++ondrop 当拖拽元素放在目标元素松开鼠标时触发

++ondragleave 当鼠标离开目标元素时触发

注意：想要ondrop事件触发不要阻止ondragover的默认行为(实现拖拽主要靠这两个事件, 其他事件主要为了给拖拽过程实现效果)

(3)拖拽实例总结

注意：在拖动元素时，每隔 350 毫秒会触发 ondrag 事件

实现要点: ##使用一个对象接收被拖拽的元素用于插入目标元素

##利用ondragstart和ondragend事件实现拖拽时的效果

##利用ondragover和ondrop事件实现拖拽功能

##优化(实例演示无说明)

++为了防止定义的对象影响全局变量(优化后可不定义对象)

++使用dataTransfer方法来存取拖拽元素

--setData存(ondragstart事件中存) -->

// 通过dataTransfer来存储数据和获取

// setData(format, data)

// fomrmat: 数据类型(text/html text/url-list)

// data: 数据(一般时字符串值)

e.dataTransfer.setData("text/html", e.target.id);

--getData取(ondrop事件中取)

/\*通过e.dataTransfer.setData存储的数据，只能在drop事件中获取\*/

var id = e.dataTransfer.getData("text/html");

e.target.appendChild(document.getElementById(id));

(4)实例 -->



1. 地图接口

(1)概念

在HTML规范中，增加了获取用户地理信息的API，这样使得我们可以基于用户位置开发互联网应用，即基于位置服务 (Location Base Service), 浏览器会自动以最优方式去获取用户地理信息。

(2)定位方式优缺点



(3)隐私: 推送通知

HTML5 Geolocation(地理位置定位) 规范提供了一套保护用户隐私的机制。必须先得到用户明确许可，才能获取用户的位置信息

(4)API说明

## navigator.getCurrentPosition(successCallback, errorCallback, options)

获取当前地理信息

##navigator.watchPosition(successCallback, errorCallback, options)

重复获取当前地理信息

##当成功获取地理信息后，会调用succssCallback，并返回一个包含位置信息的对象position。Coords(坐标)

##position.coords.latitude纬度

##position.coords.longitude经度

##当获取地理信息失败后，会调用errorCallback，并返回错误信息error

##可选参数 options 对象可以调整位置信息数据收集方式

1. H5存储
2. window.sessionStorage存储

(1)sessionStorage约5M

(2)生命周期为关闭浏览器窗口：相当于存储在当前页面的内内存中(在同一个窗口下数据可以共享(在当前页面下可以获取到，换另外一个页面下不能获取到))

2>window.localStorage存储

(1)localStorage约20M

(2)永久生效，除非手动删除：存储在硬盘上, 可以多窗口共享。但是不能跨浏览器

3>两者共同点

(1)只能存储字符串，可以将对象JSON.stringify() 编码后存储

(2)方法

## SetItem(key,value):设置数据，以键值对的方式

## getItem(key):通过指定的键获取对应的值内容

## removeItem(key):删除指定的key及对应的值内容

## clear():清空所有存储内容

1. 使用案例

<script>  
 **var** userData=document.getElementById("userName");  
 //存储数据  
 document.getElementById("setData").onclick=**function**(){  
 window.sessionStorage.setItem("userName",userData.value);  
 }  
 //获取数据  
 document.getElementById("getData").onclick=**function**(){  
 **var** value=window.sessionStorage.getItem("userName");  
 alert(value);  
 }

</script>

1. H5缓存

1>概念

通过创建 cache manifest 文件, 可以轻松地创建 web 应用的离线版本

2>基本使用

(1)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>

(2)创建manifest 文件的建议的文件扩展名是：".appcache"

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

(3)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）

3>Manifest 文件

(1)概念：

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)

(2)manifest 文件可分为三个部分

##CACHE MANIFEST – 开始

CACHE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)

(3)三部分说明

##CACHE MANIFEST说明：

CACHE MANIFEST，放置在第一行，是必需的 -->

CACHE ：

/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说明：

NETWORK小节规定文件"login.asp"永远不会被缓存, 且离线时是不可用的 -->

NETWORK:

login.asp

##FALLBACK说明：

FALLBACK小节规定如果无法建立因特网连接, 就使用指定的资源代替所请求的url的资源, -->

FALLBACK:

/html5/ /404.html

注释：当html5资源在离线状态下无法请求的时候，就使用404.html代替

(4)总结说明

##CACHE: 可以省略，这种情况下将需要缓存的资源写在CACHE MANIFEST

##可以指定多个CACHE: NETWORK: FALLBACK:，无顺序限制

## #表示注释, 只有当demo.appcache文件内容发生改变时或者手动清除缓存后，才会重新缓存。

##chrome 可以通过chrome://appcache-internals/工具和离线（offline）模式来调试管理应用缓存

(5)更新缓存

一旦文件被缓存，则浏览器会继续展示已缓存的版本，即使修改了服务器上的文件。为了确保浏览器更新缓存，也需要更新 manifest 文件，也就意味着一旦应用被缓存，它就会保持缓存直到发生下列情况：

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

说明：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

后面查看HTML5-三~五天笔记

1. 适配方案(通过视口, 配合流式布局)
2. 流式布局

宽度使用百分比(百分比布局), 高度用px来固定住

1. viewport(视觉窗口, 即视口)

(1)简介

viewport是移动端特有的, 是一个虚拟的区域, 承载网页, 浏览器承载视口, 视口承载网页(使得PC端页面放到移动端不出现滚动条, 原理是缩放)

缺点：内容也会缩放(采用viewport默认值会)

(2)标准适配方案(改变viewport默认设置以达到内容正常显示, 且无滚动条)

##注意点：

++设置viewport需紧跟在编码(<meta charset=”UTF-8”>)后面

++s通过meta标签的content属性来传递参数来改变其默认值 -->

<meta content=”里面设置参数”>

++开启默认缩放(user-scalable=yes)之后maximum和maximum才有用

##viewport的主要功能(参数)

++width 设置宽度 (device-width当前设备的宽度)

++height 设置高度

++initial-scale 缩放比例

++user-scalable 是否允许用户缩放 (no, 0用户不能缩放, yes, 1用户能缩放)

++maximum 最大缩放比

++maximum 最小缩放比

(3)国际标准适配方案



(4)生成标准适配方案快捷键meta:vp+tab

1. 移动开发注意点
2. 屏幕像素和px区别

屏幕像素：物理像素, 即像素点, 设备屏幕的最小可视颗粒

px:页面的尺寸单位

1. 移动开发可以使用jquery但不推荐使用

原因：jquery做了很多PC端浏览器兼容, 特别是ie, 移动端没有ie

1. 移动端内核都是webkit或者blink 兼容都是加-webkit-前缀
2. 建议使用h5的pai或者使用 zepto.js的库(基于高版本浏览器开发)
3. 常用box-sizing原因

(1)移动开发常用流式布局(宽度百分比, 高度固定)

(2)普通盒模型无法准确计算盒子大小 -->

.box{ wdith: 100%; height: 200px; border: 20px solid red; }

普通盒子无法计算盒子大小会导致出现横向滚动条

ie盒模型其大小为100%, 200px无横向滚动条

(3)设置IE盒模型之后都是从边框开始计算盒子尺寸的

6>在项目中初始化文件中常见设置

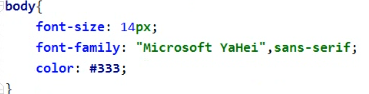
(1)设置成IE盒模型

\*, \*::before, \*::after{ box-sizing: border-box; -webkit-box-sizing: border-box}

(2)清除高亮属性(移动端特有的默认样式, 注意和伪类:active区分)

-webkit-tap-highlight-color: transparent;清除高亮

(3)设置字体

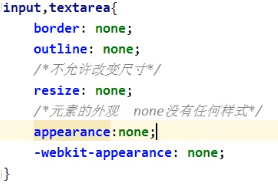


没有Microsoft YaHei就用系统自带的字体

(4)去列表原点



(5)输入框



7>px是css单位, 不是真机上的分辨率,

8>1px单位里面可能可以放多个物理像素点

1. 物理像素比 非矢量网站可能会失真, (标准视口使用二倍图可以解决失真问题)

1. 移动端常见多栏布局
2. 双飞翼(圣杯模式)

两边固定中间不固定

2>弹性盒子布局flex

用法见HTML第五天笔记

3>响应式

1. 触摸事件

1>事件

(1)touchstart 当手指触摸屏幕时候触发

(2)touchmove 当手指在屏幕来回滑动时候触发

(3)touchend 当手指离开屏幕时候触发

(4)touchcancel 当被迫中止滑动时候触发, 如来电, 弹消息

2>事件对象

(1)changedTouchs 改变后的触摸点集合

(2)targetTouches 当前元素的触摸点集合

(3)touches 页面上所有触发点的集合

(4)三者关系

##名字都是touchList(触摸点的集合, 一个手指一个触摸点)

##changedTouchs在接触屏幕和离开屏幕都会有记录而targetTouches 和targetTouches不会有记录( length: 0 )

3>实战说明

(1)通过绑定touchstart和touchmove计算其e.touches[0].clientX(鼠标坐标)值计算其左右滑动手势。

1. tap事件(移动端轻击)
2. 前言

tap事件是touch事件衍生出来的(其封装), 为了解决移动端click事件延迟300ms(移动端click事件延迟300ms为了区分用户是点击还是滑动) , 谷歌浏览器看不到延迟效果, 在真机才有300ms延迟效果

1. 可以使用第三方插件引入就可以直接使用了

(1)fastclick.js插件, 提供移动端响应速度, 引入之后在需要轻击事件的页面执行 -->

Document.addEventListener(“DOMContentLoaded”, function () {

FastClick.attach(document.body);

})

既可以使用, 页面使用click绑定事件即使轻击事件

(2)zepto.js库(基于高版本开发)里面也有tap轻击事件的封装

3>封装



1. iScroll插件区域滚动插件注意点
2. 使用插件new(“父容器(小容器)”, function () { 配置参数} )
3. 需要滚动的子容器必须要大于父容器
4. 父容器和子容器之前需要夹一层容器 -->

div.wrap(父容器)>ul>li(需要滚动的容器)