**01 HTML5简介：**

HTML5也是一个web标准【最新的一项】

<img src="foo" alt="bar" /> <p class="foo">Hello World</p> 完全正确

<img src="foo" alt="bar"> <p class="foo">Hello World img是单标签 没有p的结束标签

<IMG SRC="foo" ALT="bar"> <P CLASS="foo">Hello World</P> 都是大写

<img src=foo alt=bar> <p class=foo>Hello World</p> 属性值没加双引号

但是现在HTML5都是正确的，主要还是推荐第一种严格的规范书写代码【辨识度高】

可以省略结束标签的元素：li、p、option、tr、th、td、thead、tbody、tfoot

可以省略全部标签的元素：html、head、body

【注意】即使标签被省略了，该元素还是以隐式的方式存在的。body省略不写了，也是有的。可以通过document.body进行访问。

HTML5 应用跨设备、跨平台支持【兼容性好】

**02 HTML5标签及特性：** 【重点：增加的结构性和功能性元素】->面试简答题里

<!DOCTYPE html>

<meta charset="utf-8" />

<title> 第一个HTML5页面 </title>

<p>Hello,World</p>

浏览器中查看也是有head、body等标签

新增结构性元素：

header 定义头部，通常包含导航和一些引导信息【可有多个header元素】

nav 定义页面导航的链接组【设置导航栏】----配置着ul和li【可有多个nav元素，作为页面整体或不同部分的导航】

article 定义可以独立被外部引用的内容

section 定义文档中的区域【嵌套在article里面的】

aside 用来表示当前页面或文章的附属信息【包含在article里/外】

article里：<article>

<h1>…</h1>

<p>…</p>

<aside>…</aside>

</article> 【有关资料等】

article外：<aside>

<h2>…</h2>

<ul>

<li>…</li>

</ul>

</aside> 【友情链接、广告等】

footer 定义页脚。通常包括相关区块的脚注信息，坐着、文档的创作日期以及版权信息等。

新增功能性元素：HTML4加入了对音频和视频的原生支持

video：播放视频

andio：播放音频

各种属性：src=' ' controls='controls' antoplay='antoplay'

audio/video 组合source 元素。source 元素可以链接不同的音频文件，也可以为同一个媒体数据指定多个播放格式

使用play()、load()、pause()等常用控制函数可以灵活控制音/视频，做一个播放器

details：用于描述文档或文档某个部分的细节【目前只有谷歌和Safari6支持这个标签】

mark：定义带有记号的文本【高亮显示】<p> 每天都要记得吃<mark>早饭</mark> </p> 【默认为黄色】

progress：标示任务的进度 <progress value="25" max="100"> </progress>

**03 表单**

1. 新增input输入类型：**【注意：input标签一定要放在form里，否则按提交按钮会没有效果显示】**
2. <!--数值输入文本框-->

<input type="number" name="demoNum" min="1" max="100" step="2" />



<!--min是默认的长度，点个下三角会增加一个步长2，直到在100范围内最大的值-->

【注意】外观与text文本框相同，但不能输入数值以外的文字，否则提 交时将内容作为空白进行提交

1. <!-- 邮箱输入文本框-->

<input type="email" name="email" placeholder="请输入邮箱" />

<!--placeholder属性为提示信息，type为email专门用来输入email地址。当表单提交时，自动校验是不是符合email的格式【不用自己写正则表达式】，但是不会验证email是否存在。-->

<input type="submit" /><!--提交按钮-->

1. <!--url输入文本框-->

<input type="url" placeholder="输入正确的网址"/>

<input type="submit" />

1. <!-- 电话号码输入文本框-->

<input type="tel" placeholder="输入电话" />

<input type="submit" />

1. <!--滑动条输入文本框-->

<input type="range" min="0" max="50" step="5" name="rdemo" value="30" />

<!--value为最开始默认的位置-->

网址上，上传name的rdemo值为30

1. <!--日期时间输入文本框-->

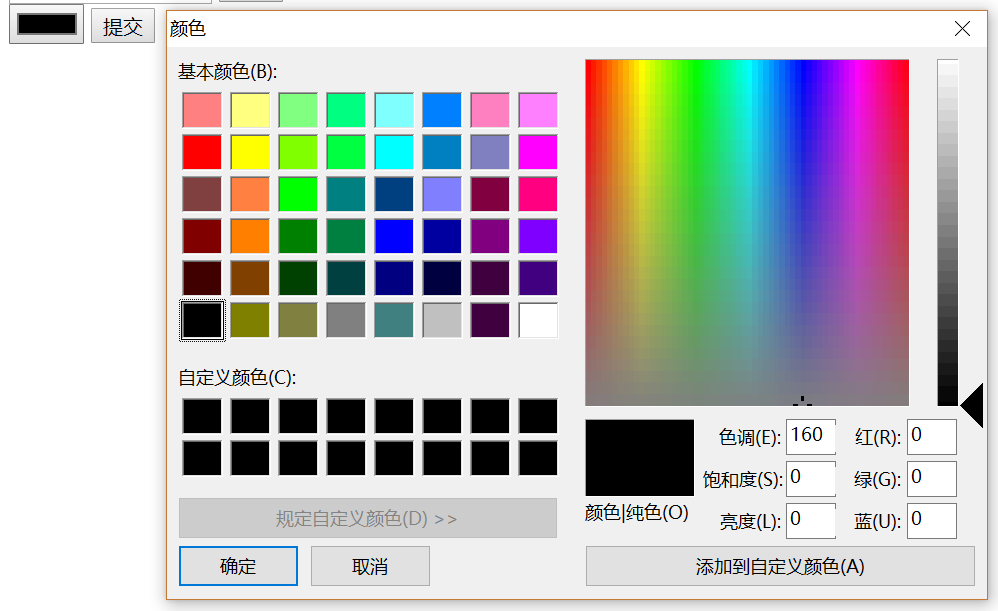
<input type="date" name="datedemo" /><!--默认的是当前日期-->

<input type="submit" />

1. <!--颜色选择文本框-->

<input type="color" name="col" />

<input type="submit" />【默认为黑色】



【网址上有选择颜色的值】

1. <!--搜索功能文本框-->

<input type="search" name="sdemo" />

<input type="submit" />

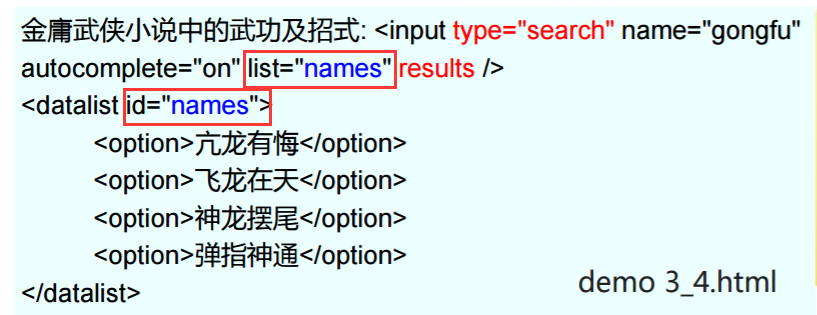
<!--现在的功能不是特别强大，只有后面文本框中带有一个小×-->



1. 新增表单元素：
2. datalist元素（为输入框提供可选的列表）->搜索功能文本框

【input的type=”search”与datalist公用】

【list的属性值，要和datalist中的属性值一样（进行绑定）】



冯家栋是不是猪：<input type="search" name="love" autocomplete="on" list="feng" results />

<datalist id="feng">

<option>是啊，当然是</option>

<option>开玩笑，怎么不是</option>

<option>恭喜你，答对了这个臭猪猪</option>

<option>他是个无敌的赖皮猪</option>

</datalist>

<input type="submit" />

autocomplete="on"启动自动完成功能

1. <!--output-->

<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>



1. 新增表单属性：
2. Form属性：将表单内的从属元素卸载页面的任意位置，然后为该元素指定form属性，属性值为该表单的id。如需引用多个表单，使用空格分隔的列表

<form id="testform" method="get" action="form.asp">

<!--action将表单提交到页面，method将表单元素分别指定不同的提交方法-->

姓名：<input type="text" name="name1" />

<input type="submit" />

</form>

自我介绍：<textarea name="area1" form="testform"></textarea>

1. formaction、formmethod属性:

<!--在HTML5中使用formaction属性实现将表单提交到不同的页面，使用formmethod属性对每个表单元素分别指定不同的提交方法-->

<form id="testform" method="get" action="sever.jsp">

<input type="submit"name="s1"value="v1"formaction="s1.jsp" />提交到s1

<input type="submit" name="s2" value="v2"formmethod="post" />提交到s2

<input type="submit"name="s3"value="v3"formmethod="get" />提交到s3

</form>

1. required,placeholder属性：

<!--required规定输入域在提交之前必须填写-->

<form>

姓名：\*<input type="text" required="required" />

<input type="submit" />

</form>



1. list属性：

input元素的list属性与datalist元素的id属性绑定

1. antofocus【布尔属性，加载的时候表单域自动聚焦】、multiple【允许上传多个文件】、max/min/step属性【设置最大最小值，数值或日期时间的增量】

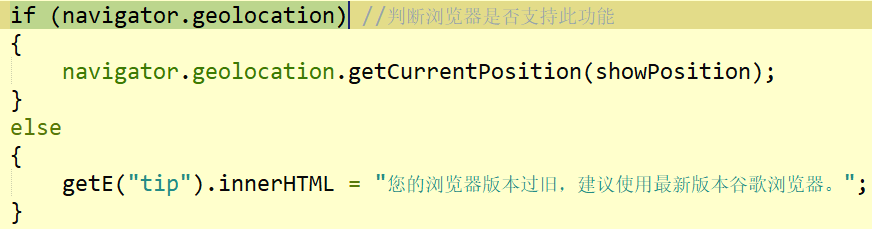
**04 地理位置定位**

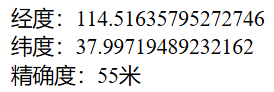
1. 地理位置定位的作用
2. HTML5获取当前地理位置
3. 地理位置定位示例
4. 为了获取用户的地理位置信息，需要使用多个资源
5. 浏览器提供geolocation对象，此对象为navigator对象的一个属性
6. getCurrentPosition (onSuccess, onError, options)【单词获取位置数据】

第一个参数：获取数据成功后执行的回调函数

第二个参数：（可选，但是还是经常写出来）获取数据失败后执行的回调函数

第三个参数：（可以选择不写）设定最长可接受的定位返回时间等





1. 纬度：coords.latitude

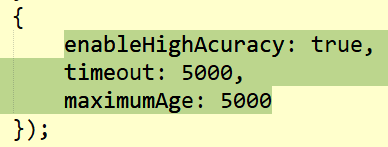
经度：coords.longitude

精确度：coords.accuracy

1. 失败回调函数：【定位信息失败时返回一个error对象给失败回调函数】

属性：

1. Code错误编码（0,1,2,3）
2. Message错误信息
3. Options可选参数列表：

精确模式、等待响应的最大时间、使用最近缓存的位置数据

1. watchPosition(onSuccess, onError, options)持续获取位置数据方法 ->只针对移动设备

【只要用户位置发生改变的时候，他就会调用成功回调函数，返回一个ID】

1. 百度地图的应用【了解】

**05 离线web应用**

1. HTML5提供离线web应用的实现机制。使web应用可以在用户离线的状态下进行访问。
2. 本地缓存、浏览器网页缓存
3. 创建离线应用程序：

**第一步**：创建缓存清单文件（manifest文件）

文件扩展名使用appcache

Manifest文件需要在web服务器上配置正确的MIME类型为 text/cache-manifest

**第二步**：在html标记中指定使用缓存文件



Manifest文件分为三部分：

1. CACHE MANIFEST【在此标题下列出的文件将在首次下载后进行缓存，写在第一行，必须有该部分】
2. 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】
3. FALLBACK【提供了获取不到缓存资源时的备选资源路径（比如 404 页面）】
4. 构建离线应用后，即使在线状态，用户也会访问缓存文件

**06 Web存储【重点掌握】**

1. 因为将数据存储在客户端（浏览器）上，所以记录了浏览历史……
2. 存储在浏览器上的数据 => cookie

【手机上：您的手机是否想要使用cookie在保存您的浏览记录…】

1. Cookie：是HTML4中客户端存储简单用户信息的一种方式，使用文本来存储信息

保存用户信息、用户设置、密码记忆

1. Cookie的缺点：
2. 存储空间是有限的
3. 安全性差【使用简单的文本存储数据，保存在客户端浏览器，容易被黑客窃取】
4. 浪费带宽【每次访问服务器，都会来回传送cookie】
5. 客户端操作cookie，非常复杂
6. HTML5：Web存储【本地存储】
7. Web存储提供两种存储机制
8. LocalStorage【永久保存】
9. Session Storage【临时保存】
10. Web Storage

存储是基于键/值对的形式存储的，每个键值对称为一个项。存储和检索数据都是通过指定的键名。

键名的类型是字符串类型

值可以是包括字符串、布尔值、数值在内的任意 JavaScript 支持的类型。

【在控制台中application-storage查看键值对】

LocalStorage对象：【操作数据的方法】【6-1/6-2】

1. Web Storage用法

存储数据的数据类型：仅能存储字符串类型的数据

对象存储：【转换为字符串-JSON API】->JSON.stringify() JSON.parse()

1. SessionStorage【数据有效期：会话周期（浏览器关闭则失效）】

用户浏览这个网站所花费的时间就是session的生命周期

Session对象可以用来保存在这段时间内所要求保存的任何数据。它的作用域是窗口级别的

SessionStorage方法针对一个session进行数据存储。当用户关闭浏览器窗口后，数据会被删除

1. 计数器【localstorage/sessionstorage】

LocalStorage可以作为浏览计数器，SessionStorage可以作为刷新计数器

作业：

页面换肤【要求：当用户关闭之后，浏览器仍然能保持原来的设置 设置个默认颜色】

**07 数据通信 【作为了解】**

1. Web应用的数据通信方式：

HTTP协议：【通信只能有客户端发起，服务器总是被动的】

Ajax：

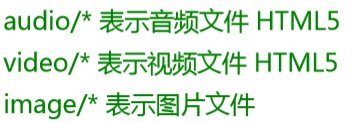
1. WebSocket【全双工通讯的网络通讯协议】（HTTP是单向的通信协议） 但他们两个都是通过TCP协议传输数据
2. WebSocket服务器

WebSocket构造函数：var Socket = new WebSocket(url, [protocol] );

**08 文件**

1. HTML5文件操作：可以完成选中多个文件、读取文件内容等【以前都是规定只能上传一个文件】
2. 沙箱：【相当于一个保护系统】是一个虚拟系统程序，允许你在沙盘环境中运行浏览器或其他程序，因此运行所产生的变化可以随后删除。
3. File对象指用户选择的文件，FileList对象指用户选择的文件列表
4. Multiple属性 【在file控件内允许一次放置多个文件】
5. files有length属性和item方法，files[index] files.item(index)获取选择的file对象
6. Blob对象 表示二进制原始数据【使用blob对象，获取一些文件信息】
7. slice() 访问字节内部的原始数据块
8. size属性【可以获取文件大小（文件字节长度）】
9. type属性 【作用：只上传符合我要求的文件（判断文件类型）】

可以通过type属性过滤文件【上传某种特定类型文件】

Type属性： 

1. 停止 break 跳过 continue【8-3.html】
2. FileReader对象 用来把文件读入内存，并且读取文件中的数据【可以用来实现预览功能】

需要检测浏览器是否支持FileReader【检测的是这个接口的浏览器是否有一个位于Window对象下的FileReader构造函数】

// if ( typeof FileReader == 'undefined' )

if ( ！window.FileReader ) {

alert( " 您的浏览器未实现 FileReader 接口 " );

} else {

var reader = new FileReader();

}

FileReader对象的方法：

1. abort
2. readAsBinaryString
3. readAsDataURL
4. readAsText

**09 拖放**

1. 源对象 目标对象
2. DataTransfer对象 拖拽数据传递对象

为事件对象的一个属性【event.dataTransfer】用于在源对象和目标对象的事件间传递字符串格式的数据 只能在拖放事件的事件处理程序中访问，可以使用此对象的属性和方法来完善拖放功能

1. DataTransfer对象的方法

* setData(format, data) 向DataTransfer对象存入数据 【参数：类型，内容】



* getData(format) 从DataTransfer对象读数据



1. DataTransfer对象的属性

types属性 存入数据的种类

dropEffect属性 表示拖放操作的视觉效果

1. 拖放：
2. 设置元素为可拖放

【源对象】

1. 设置拖动什么 ondragstart 和 setData()
2. 放置元素到何处 ondragover
3. 进行放置 ondrop 和 getData()



**10 画布**

1. Canvas 只是图形容器 需使用脚本绘制图形
2. <canvas id = “myCanvas” width = “200” height = “100”>

您的浏览器不支持canvas元素，请更换或更新浏览器

</canvas>

【设置id是便于以后绘制图形，设置长宽同时改变canvas元素的大小和元素绘图表面的大小（通过css只会改变canvas的大小）=>所以建议在canvas标签里设定长宽】

设置长宽的时候不适用px，直接写数值

1. getContext()方法

getContext()方法可返回一个对象，该对象提供了用于在画布上绘图的方法和属性



【可以接受连个参数值，2d和3d，分别表示二维和三维（现在二维的应用比较多）】

1. canvas是一个二维网格

坐标系原点位于（0,0）【canvas的左上角】，x轴水平向右，y轴垂直向下

1. 路径和描绘
2. moveTo(x,y)【用于创建新的子路径，并规定其起始点为(x,y)】
3. lineTo(x,y) 【为当前子路径添加一条直线，这条直线从当前点开始，到(x,y)结束】
4. stroke() 【实际画出路径，默认为黑色】
5. 线条样式设置
6. lineWidth【线条宽度】context.lineWidth = 5;
7. strokeStyle【绘制描边的颜色】context.strokeStyle = "red";
8. 填充

【fill()和stroke()只有宽度会影响（根据所需样式设置）】

1. fill()

填充当前绘图，默认黑色

如果路径未关闭，那么 fill() 方法会自动从路径结束点到开始点之间添加一条线，以关闭该路径，然后填充该路径

1. fillStyle

用于填充绘画的颜色 context.fillStyle="red";

1. 路径封闭

beginPath()开始一条新的路径

closePath()关闭一条打开的子路径

【canvas之中只能有一条路径存在，称之为当前路径(current path)】

1. 绘制弧和圆
2. arc()创建弧/曲线【用于创建圆或部分圆】

context.arc(x, y, r, sAngle, eAngle, anticlockwise)【默认为顺时针】

【例，同心圆、八卦图】

1. 绘制矩形

context.rect(x, y, width, height)

x,y指定矩形左上角的位置

rect(x，y，width，height); 创建矩形 

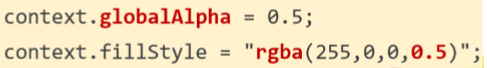
fillRect(x，y，width，height) ; 绘制 "已填色" 的矩形（实心） 

strokeRect(x，y，width，height); 绘制不填色的矩形（空心） 

clearRect(x，y，width，height); 清空给定矩形内的指定像素

1. 透明度

globalAlpha 属性【全局的透明度设置】

【两种方法】

【掌握随机填充色的写法（rgba…）】

rgba是字符串引出来的，里面并不是纯字符串，就得用**字符串拼接**

**例**context.fillStyle = "rgba("+random(0,255)+","+random(0,255)+","+random(0,255)+",0.5)";

**11 画布（二）**

1. 线条样式
2. lineWidth 线条的宽度
3. lineCap 线条末端线帽的样式 【butt(默认) round square】

round和square会使线条略微变长

1. lineJoin 两条线交汇时所创建边角的类型

context.lineJoin="bevel |round |miter";

1. miterLimit 设置最大斜接长度（默认值为10）【只有当lineJoin属性为“miter”，miterLimit才有效】

斜接长度指的是在两条线交汇处内角和外角之间的距离

如果斜接长度超过 miterLimit 的值，边角会以 lineJoin 的"bevel" 类型来显示

1. 平移 translate(dx, dy) 【两次translate 会累加平移】
2. 旋转 rotate(angle) 【正值表示顺时针方向，负值表示逆时针方向】

angle = degrees/180\*Math.PI【角度转换为弧度公式】

【同样会累加旋转】

1. 缩放 scale(swidth, sheight) 【x放大的倍数，y放大的倍数】【**定位也会缩放**】

【同样会累加缩放】

1. 画布坐标状态保存好恢复
2. save() 保存当前canvas绘图环境的所有属性、坐标变换信息等
3. restore() 将绘图环境状态恢复为保存值
4. 可以嵌套式的调用 save( )、restore( ) 方法
5. 绘制填充文字
6. fillText( text, x, y, maxWidth ) 在画布上绘制填色的文本【默认为黑色】
7. font 上文本内容的当前字体属性
8. 绘制描边文字 strokeText( text, x, y, maxWidth ) 在画布上绘制文本
9. 其他
10. measureText(text) 指定字体宽度
11. textAlign 对齐方式
12. textBaseline 文本基线
13. 阴影
14. shadowColor
15. shadowBlur 模糊级别
16. shadowOffsetX 形状与阴影的水平距离【默认0】
17. shadowOffsetY 形状与阴影的垂直距离【默认0】

【偏移量是可正可负的】

**12 画布（三）**

1. 渐变色
2. 颜色
3. fillStyle【设置用于填充绘画的颜色】
4. strokeStyle【设置用于绘制描边的颜色】
5. 线性渐变

A．createLinearGradient(x0,y0,x1,y1) 创建线性渐变对象

【var grd = context. createLinearGradient(0,0,200,0) 创建对象，水平向右的渐变】

B．addColorStop(stop,color)规定渐变对象中的颜色和停止位置

C．context.fillStyle = grd;【填充】

D．context.fillRect（0,0,200,100）

1. 小于最小stop的部分会按最小stop的color来渲染，大于最大stop的部分会按最大stop的color来渲染

【若矩形超过渐变对象的长，超过的那个部分就以渐变对象最大的颜色来渲染，反之…】

1. 放射状/环形渐变

createRadialGradient(x0,y0,r0,x1,y1,r1)【两个圆心位置都是一样的】

1. 图片填充、描绘

createPattern() ( image, type)方法

//图片填充、描绘

var image = new Image();

image.src = "img/taidi.jpg";

**image.onload** = function(){

var img = context.createPattern(image,"repeat");

context.fillStyle = img;

context.strokeStyle = "#FFC0CB";

context.lineWidth = 3;

context.fillRect(0,0,1200,600);

context.strokeRect(0,0,1200,600);

}

1. 合成操作globalCompositeOperation 【默认：source-over（源图像可能会把一部分的目标图像遮住）】

源图像 —— 打算放置到画布上的绘图

目标图像 —— 已经放置在画布上的绘图

1. 绘制图像drawImage( ) 方法在画布上绘制图像或视频

（1）context.drawImage(img, x, y) 【在画布上定位图像】

（2）context.drawImage(img, x, y, width, height) 【在画布上定位图像，并规定图像的宽度和高度（缩放）】

（3）context.drawImage(img, sx, sy, sw, sh, dx, dy, dwidth, dheight)【剪切图像，并在画布上定位被剪切的部分。（剪切、缩放）】【先截图，再缩小和放大（前面是抠图，后面是显示的位置）】

【注意：需要在图片加载完成之后才能绘制图片】

【注意：drawImage()该方法的参数只能为 3 5 9】

**img.onload**=function ( ){

context.drawImage(img,50,100,300,300);

}

1. 绘制视频 drawImage()

drawImage(video,x,y)

drawImage(video,x,y,width,height)

【注意：仅对当前正在播放视频的帧进行描绘（同步的）】

1. ImageData该对象为画布上指定的矩形像素数据

创建ImageData对象

返回ImageData对象【了解】

1. 动画循环【在播放动画时持续更新并绘制动画， 即动画循环】

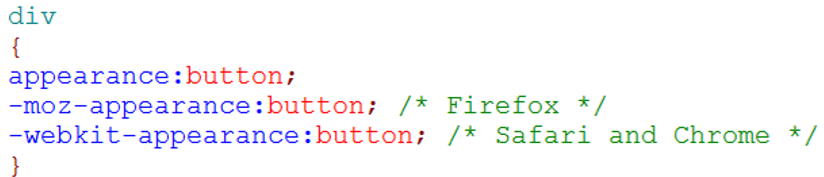
**13 CSS3概述及选择器**

1. CSS3是最新的CSS标准
2. CSS3的新特性
3. CSS3使用浏览器前缀

-moz- Firefox 的替代用法

-webkit- Safari 和 Chrome的替代用法

-ms- IE 的替代用法



1. CSS3选择器：

属性选择器

兄弟选择器

伪类选择器

1. 属性选择器：
2. E[att ^= "val"] 【选择名称为E 的标记，且该标记定义了att属性，att属性值是**以val 开头**的字符串】

[id ^= sub]{ background-color: rgb(129,26,91); }【选择id以sub开头的字符串】

1. E[att $= "val"]【选择名称为E 的标记，且该标记定义了att属性，att属性值是**以val 结尾**的字符串】
2. E[att \*= "val"]【选择名称为E 的标记，且该标记定义了att属性，att属性值是**包含 val**的字符串】
3. 属性选择器的应用：通过属性选择器进行属性的分类筛选

a[href $= "/"]:after, a[href$="htm"]:after,a[href$="html"]:after{

content:" Web网页";

color: red;

}

1. 兄弟选择器 element1~element2

例，p~ul 【寻找p后面能匹配到ul的兄弟节点 (也就是p后面的ul)】

1. 伪元素选择器：针对CSS中已经定义好的伪元素使用的选择器

E:first-line 设置对象内的第一行的样式。

E:first-letter 设置对象内的第一个字符的样式

E:before 设置在对象前（依据对象树的逻辑结构）发生的内容。用来和content属性一起使用，并且必须定义content属性

E:after 设置在对象后发生的内容。用来和content属性一起使用，并且必须定义content属性。

例，li:before{ content: "△"; }

1. 结构性伪元素选择器

E:root 匹配文档的根元素。在HTML中，根元素永远是HTML。

E:not(s) 匹配不含有s选择符的元素E。

E:empty 匹配没有任何子元素（包括text节点）的元素E。

E:target 匹配**锚点链接**中当前活动的目标元素。

【点击某个链接，链接的背景颜色变为黄色】

:target{

background-color: yellow;

}

<p id="menu">

<a href="#text1">示例文字1</a> |

<a href="#text2">示例文字2</a>

</p>

<div id="text1"> //锚点链接

<h2>示例文字1</h2>

<p>区块1内容</p>

<p>区块1内容</p>

<p>...此处略去</p>

</div>

<div id="text2">

<h2>示例文字2</h2>

<p>区块2内容</p>

<p>区块2内容</p>

<p>...此处略去</p>

</div>

1. E:first-child 匹配父元素的第一个子元素E。

E:last-child 匹配父元素的最后一个子元素E。

E:only-child 匹配子元素E，该子元素属于其父元素的唯一子元素。

/\*从父元素里面挑选第一个子元素为li\*/

li:first-child{

background-color: yellow;

}

/\*从父元素里面挑选最后一个子元素为li\*/

li:last-child{

background-color: skyblue;

}

<ul>

<li>列表项目1</li>

<li>列表项目2</li>

<li>列表项目3</li>

<li>列表项目4</li>

<li>列表项目5</li>

<p>1123</p>

</ul>

**【只有列表项目1的样式会改变，1123不会改变（最后一个元素为p不为li）】**

1. E:nth-child(n) 匹配父元素的第n个子元素E 【E:nth-child(2n)；】

E:nth-last-child(n) 匹配父元素的倒数第n个子元素E

1. E:nth-of-type(n) 匹配属于父元素的特定类型的第 n 个子元素的每个元素。

E:nth-last-of-type(n) 匹配属于父元素的特定类型的第 n 个子元素的每个元素， 从最后一个子元素开始计数。

E:only-of-type 匹配属于其父元素的特定类型的唯一子元素的每个元素。

【\*】E:nth-child(n) & E:nth-of-type(n)

E:nth-child(n)匹配的是父元素的第二个为E的，如果不为E则不成功

E:nth-of-type(n)匹配的是父元素特定的E的第二个元素

li:nth-child(2){

background-color: gold;

}

<ul>

<li>1</li>

<div>2</div>

<li>2</li>

</ul> 【不改变，因为第二个为div不是li】

li:nth-of-type(2){

background-color: gold;

}

<ul>

<li>1</li>

<div>2</div>

<li>2</li>

</ul> 【改变第二个li】

**14 文本、字体与颜色**

1. 阴影：

（1）text-shadow给文本添加阴影【默认为none】

text-shadow : **h-shadow v-shadow** blur color;【h-shadow v-shadow对于text的水平距离和垂直距离】

（2）外发光（霓虹灯）文字效果 => 阴影不发生位移，同时定义阴影模糊显示

（3）可以指定多个阴影，使用逗号分隔 => 实例，火焰字、立体字、描边

1. 文本自动换行【word-wrap】

word-wrap: normal | break-word;【默认为normal】

normal会有溢出 break-word不会

1. 文本自动换行【word-break】

word-break: normal | break-all | keep-all;

1. 字体样式

实现网页中插入特殊字体的过程

1. 颜色样式：

三种颜色值定义模式：RGBA颜色值、HSL颜色值和 HSLA颜色值

1. Rgba
2. HSL

饱和度为100时，颜色最鲜亮

1. HSLA

是带有透明度的

**15 背景与边框**

1. background-size

【语法：background-size: length| percentage| cover| contain;】

1. background-origin

【语法 background-origin: border-box | padding-box | content-box;】

1. 允许设置多个背景图像

background-image:url(flwr.gif)**,** url(tree.gif);

【先写的那个是会在整体背景的最上层】=> 注意图像的叠放顺序

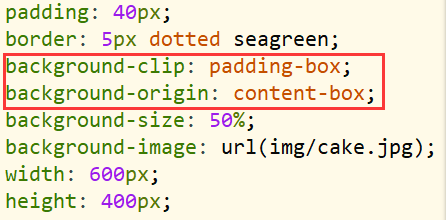
1. 规定背景的绘制区域=>决定背景在哪些区域显示

【语法 background-clip: border-box| padding-box| content-box;】

1. background-clip & background-origin

background-clip默认的是border，背景在元素的边框、补白和内容区域都会显示

background-origin是从哪个区域开始绘制背景 background-clip真正作用是决定背景在哪些区域显示





1. 圆角边框【border-radius】
2. border-radius: 50px/30px【圆角的水平半径为50px，圆角的垂直半径为30px】=>有椭圆的感觉（若其中一个设置成为0px，则不改变）
3. 圆角边框四个角的半径各不相同=>左上角、 右上角、右下角、左下角的顺序设置每个方向圆角半径的值
4. 图像边框

border-image: url(图像文件的路径) A B C D【ABCD四个参数表示浏览器自动把图像分隔时的上边距、右边距、 下边距以及左边距】

border-image: url(border.png) 18 18 18 18;【背景大小不够的时候可能会有拉伸】

border-image: url(border.png) 18 18 18 18 round;【会铺满，不会拉伸】

1. 元素阴影

box-shadow: **h-shadow v-shadow** blur spread color inset;

**16 渐变与变形处理**

1. 线性渐变：

background: linear-gradient(direction, color-stop1, color-stop2, ...);

【linear-gradient()函数创建了一个img图像 没有内在的尺寸 会随着匹配的元素而改变】

默认的是从上到下进行渐变

1. 重复的线性渐变

repeating-linear-gradient()

【background: repeating-linear-gradient(red, yellow 10%, green 20%);】

若改成background: linear-gradient(red, yellow 10%, green 20%); 则剩下的会以最后的颜色填充

1. Linear-gradient(to bottom,#fff,#999) 第一个参数渐变方向【默认为to bottom】

第二、三个参数，表示颜色的起始点和结束点，可有多个颜色值

background: linear-gradient(45deg,transparent 20px,lightgreen 0px);

transparent 20px 透明色 lightgreen 0px 是切角

lightgreen可以取值0-20px 因为在其中是不过渡的

1. 径向渐变

background: radial-gradient (shape at position, color1 stop, color2 stop,...)

shape=>circle[圆]/ellipse[椭圆 默认]

1. Transform

通过转换能够对元素进行旋转、缩放、移动、倾斜或拉伸。

translateX(x) translateY(y) translate(x , y)【可以为负值】

transform: translate(-2px,-2px);

1. 旋转：

transform: rotate(angle ) 正角度为顺时针 负角度为逆时针

1. 缩放：

scaleX(x) scaleY(y) scale(x,y) 【当值为负数时，对象反转】

1. 倾斜：

skewX(x) skewY(y) skew(x,y)

1. Transform-origin 更改变换的基点位置

transform-origin: x-axis y-axis; //设置基准点



1. 多重变形 【对同一元素可添加多种变形效果】

transform: translate(150px,200px) rotate(120deg) scale(1.5,1.5);