**HTML5理论知识回顾**

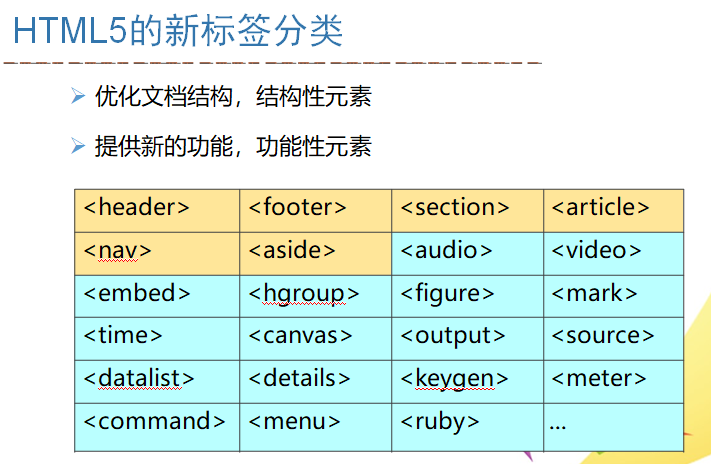
第1章-H5简介

1.结构标准：HTML

样式标准：CSS

行为标准：W3C DOM、ECMAScript

1. - HTML5标签及特性
2. HTML5的新标签



结构性元素：<header>、<footer>、<nav>、<article>…

功能性元素：<video>、<audio>※代码

新增的功能：HTML5中加入了对音频和视频的原生支持

优点：

·无需插件

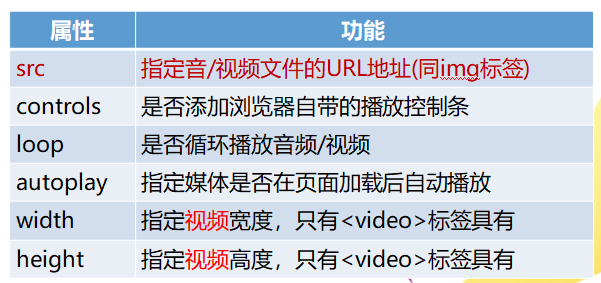
·提供JS脚本控制的API

缺点：

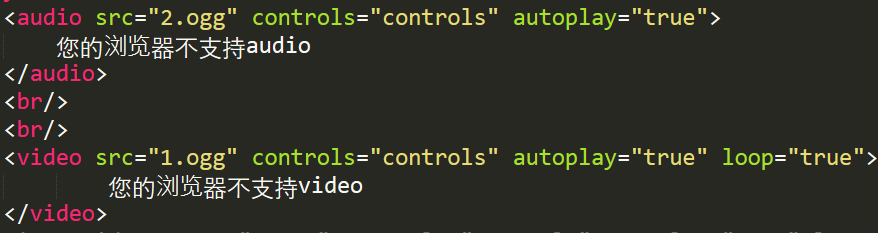
·音视频格式的支持有限

<video>播放视频

<audio>播放音频



只有不支持才显示标签的内容：

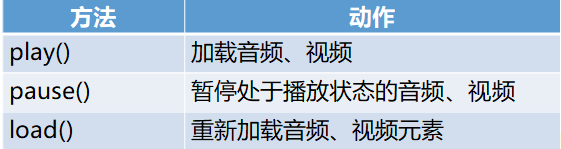


2. 新标签的使用方法

（1）音/视频格式兼容性解决方案：audio/video 组合source 元素。 ※代码



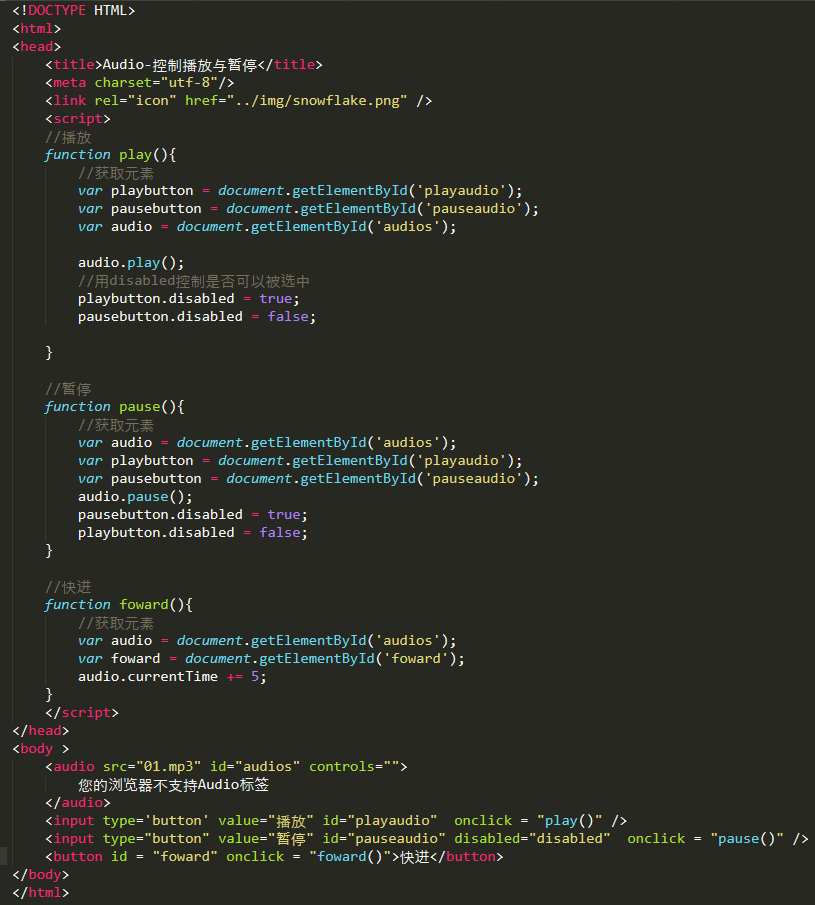
1. 音/视频操作相关API（操作音视频的标签）







※实例：通过 play( )、pause( )、 currentTime 实现对音频的播放、暂停、快进操作。



第3章 – 表单

1. 新增input输入类型(例： <input type="tel" placeholder="输入电话" name="p"/> )

number、email、url、tel、range、date、color、date…

注意：type为range时，

        <!-- 用于应该包含一定范围内数字值的输入域。range 类型显示为滑动条。能够设定对所接受的数字的限定。 -->

        <!-- step为移动步长 -->

        <input type="range" min="0" max="50" step="8" name="rdemo" value="0" />

其他都类比tel的例子书写。（每个必须写name=“xxx”）

2. 新增元素属性

required（自动校验）

<input type="email" name="email" required="required">

<input type="email" name="email" required>

placeholder、

Autofocus

<input autofocus="autofocus" />

multiple

<input type="file" name="img" multiple="multiple" />

第4章 – 地理位置定位

1.地理位置定位的作用（简答）

商业：根据用户位置，提供附近店铺的打折信息

社交：显示附近的其他用户；将个人位置信息共享给其他好友

地图：定位到用户所在位置，直接显示用户附近地图

工具：记录个人足迹

2.HTML5 地理位置定位

Geolocation 对象（此对象为 navigator 对象的一个属性）

Geolocation API

1. 能够获取到的数据信息：

latitude 纬度（十进制），例如：38.0441

longitude 经度（十进制），例如：114.51

accuracy 精确度，以米为单位

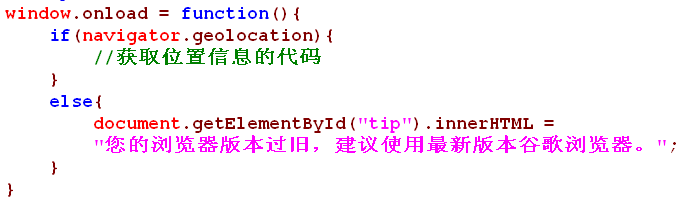
altitude 海拔，海平面以上以米计

heading 行驶方向，从正北开始以度计

speed 速度，以米/每秒计

timestamp 获取位置的时间

1. 检测兼容性：通过判断 geolocation 对象是否存在判断浏览器是否支持。



getCurrentPosition(onSuccess, onError, options) //**单次**获取当前地理位置

watchPosition(onSuccess, onError, options) //**持续**监视当前地理位置

clearWatch(watchId) //清除监视（传入该 ID 实现取消的目的。）

三个方法，期末不需要写代码，但是需要知道作用和名称

1. – 离线Web应用
2. HTML5离线应用的特点和应用（简答）

特点：

离线浏览 - 用户可在应用离线时使用它们、

加快速度 - 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

应用：

邮件、个人事务管理、博客发布平台

1. 创建离线应用程序

（1）创建缓存清单文件（manifest 文件）

（2）在 html 标记中指定使用缓存文件

<html manifest="cacheData.appcache">

1. Manifest文件

（1）CACHE MANIFEST （没网也可以获取）

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（必须有网）

在此标题下列出的文件**不会被缓存**，需要与服务器的连接，离线不可用

（3）FALLBACK（备用资源）

提供了获取不到缓存资源时的**备选资源路径**（比如 404 页面）

例：CACHE MANIFEST

theme.css

logo.gif

main.js

NETWORK:

login.php

FALLBACK:

html offline.html

1. 离线应用的更新（简答）

构建离线应用后，即使在线状态，用户也会访问缓存文件。及时更新用户的缓存文件非常重要。

HTML5离线缓存更新：

1. 用户清空浏览器缓存

2. manifest 文件被修改

3. 由程序来更新应用缓存

第6章 – Web存储

1.HTML5 Web存储的优点：

存储空间更大，浏览器至少提供 5M 的空间

数据仅仅存储在本地，不会自动向服务器发送

提供丰富的接口，方便的操作数据

独立的存储空间，每个域都有各自存储空间，不会造成混乱。

2.Web存储提供两种存储机制（优缺点、机制原理）

LocalStorage原理：将数据保存在客户端本地的硬件设备中，即使浏览器被关闭了，该数据仍然存在，下次打开浏览器访问网站时仍然可以继续使用。

Session Storage原理：将数据保存在session对象中。session 是指用户在浏览某个网站时，从进入网站到浏览器关闭所经过的这段时间。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。当用户关闭浏览器窗口后，数据会被删除。

sessionStorage为临时保存，而localStorage为永久保存。

3.LocalStorage使用方法

检测浏览器是否支持

window.onload = function(){

            //如果不存在localStorage

            if(!window.localStorage){

                alert("您的浏览器不支持localStorage");

            }

            //支持,开始读写操作

            else{

                localStorage.setItem("key0", 666);//存数据项最常用方法

                localStorage["key1"] = 777;//存数据项方法二

                localStorage.key2 = "第三个值";///存数据项方法三

                alert(localStorage.getItem("key0"));//666 读取数据项方法一

                alert(typeof(localStorage.getItem("key0")));//String

                alert(localStorage["key1"]);//读取数据项方法二

                alert(localStorage.key2);//读取数据项方法三

            }

        }

存储数据项

localStorage.setItem("key", value)

localStorage["key"] = value

localStorage.key = value

读取数据项

① localStorage.getItem(“key”)

② localStorage["key"]

③ localStorage.key

删除数据项

localStorage.removeItem(key)

清除全部数据

localStorage.clear( )

获得键名

localStorage.key(index)

获得数据项的个数

localStorage.length

4.SessionStorage使用方法

保存数据：sessionStorage.setItem(Key, value)

读取数据：sessionStorage.getItem(Key)

获得键名：sessionStorage.key(index)

移除数据：sessionStorage.removeItem(key)

清除所有数据：sessionStorage.clear()

都是只需要方法名及作用。

第7章 – 数据通信

1.WebSocket 原理：WebSocket是HTML5提供的一种浏览器与服务器间进行全双工通讯的网络通讯协议。浏览器和服务器只需要做一个握手的动作，便形成了一条快速通道。两者之间就直接可以数据互相传送。（WebSocket和HTTP属于应用层协议，都是通过TCP协议传输数据。 WebSocket是全双工通信协议，HTTP是单向的通信协议。对于WebSocket来说，它必须依赖HTTP协议进行一次握手，握手成功后，数据就直接从TCP通道传输，此后就与HTTP无关了。）

2.WebSocket 特点：

（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，握手阶段采用 HTTP 协议，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。

不需要记具体方法，理解原理，握手后的操作相应、特点。

第8章 – 文件操作

1.FileList对象与file对象

File对象指用户选择的文件

Filelist对象指用户选择的文件列表

1. Blob对象表示二进制原始数据，属性：

slice( ) 可以访问到字节内部的原始数据块。

size属性 表示一个Blob对象的字节长度

type属性 表示Blob的MIME类型，如果是未知类型，则返回一个空字符串。

※通过类型过滤文件（type属性）

image/开头的，后面紧跟图像的类型。

audio/\* 表示音频文件 HTML5

video/\* 表示视频文件 HTML5

image/\* 表示图片文件

1. FileReader对象对象用来把文件读入内存，并且读取文件中的数据。
2. 实现图片预览的例子※代码及demo

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>fileReader方法示例</title>

    <link rel="icon" href="img/snowflake.png" />

</head>

<body>

    <script>

    var result=document.getElementById("result");

    var file=document.getElementById("file");

    if (typeof FileReader == 'undefined' )

    {

     result.innerHTML = "<p>抱歉，你的浏览器不支持 FileReader</p>";

     file.setAttribute( 'disabled','disabled' );

    }

    //将文件以Data URL形式进行读入页面

    function readAsDataURL()

    {

     //检查是否为图像文件

     var file = document.getElementById("file").files[0];

     if(!/image\/\w+/.test(file.type))

     {

     alert("请确保文件为图像类型");

     return false;

     }

     var reader = new FileReader();

     reader.readAsDataURL(file);

     reader.onload = function(e)

     {

     var img\_result=document.getElementById("result");

     img\_result.innerHTML = '<img src="'+this.result+'" alt=""/>'

     }

    }

    //将文件以二进制形式进行读入页面

    function readAsBinaryString()

    {

     var file = document.getElementById("file").files[0];

     var reader = new FileReader();

     reader.readAsBinaryString(file);

     reader.onload = function(f)

     {

     var result=document.getElementById("result");

     result.innerHTML=this.result;

     }

    }

    //将文件以文本形式进行读入页面

    function readAsText()

    {

     var file = document.getElementById("file").files[0];

     var reader = new FileReader();

     reader.readAsText(file);

     reader.onload = function(f)

     {

     var result=document.getElementById("result");

     result.innerHTML=this.result;

     }

    }

    </script>

    <p>

     <label>请选择一个文件：</label>

     <input type="file" id="file" />

     <input type="button" value="读取图像" onclick="readAsDataURL()"/>

     <input type="button" value="读取二进制数据" onclick="readAsBinaryString()"/>

     <input type="button" value="读取文本文件" onclick="readAsText()"/>

    </p>

    <div name="result" id="result">

     <!-- 这里用来显示读取结果 -->

    </div>

</body>

</html>

第9章 – 拖放

1.DataTransfer对象

（1）源对象与目标对象

（2）方法

①设置数据（向DataTransfer对象存入数据）

setData(format, data)

Format:格式

Data：数据

②获取数据（从DataTransfer对象读数据）

getData(format)

注：现在支持拖动处理的MIME的类型有：“text/plain（文本文字）”、“text/html”、“text/xml”、“text/uri-list（URL列表，每个URL为一行）

（3）属性

①files ：拖拽文件列表

file.length可以表示文件长度

②dropEffect ：表示拖放操作的视觉效果，即鼠标形状。

③types

④effectAllowed

2.步骤

（1）将需要拖放的对象元素的draggable属性设为true ，即 draggable="true"

注： img元素与a元素默认允许拖放。

设置拖动什么 ondragstart 和 setData()

放置元素到何处 ondragover

进行放置 ondrop 和 getData()

第10章 ~ 第12章 画布

1.创建Canvas元素

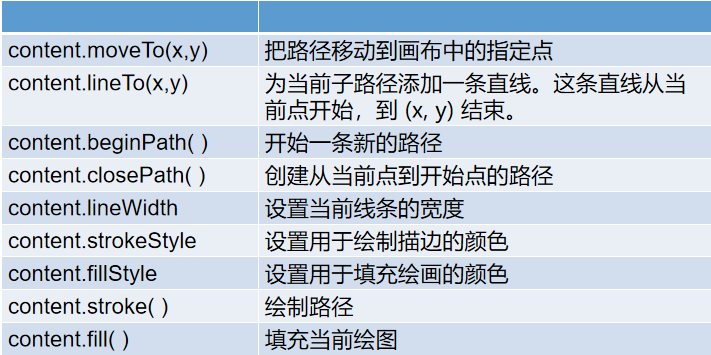
<canvas id="canvas1" width="200" height="100">

您的浏览器不支持 canvas 元素，请更换或更新浏览器

</canvas>

2.Canvas绘制直线、多边形、填充、曲线、矩形、文字、图像的方法

不需要代码实现，需要理解具体的方法、作用。



路径封闭（canvas 之中只能有一条路径存在，称之为“当前路径” (current path) 。）

（1）beginPath点( )

开始一条新的路径（路径开始点），写在moveTo之前。

（2）closePath( ) //自动变成密闭图形

创建从当前点到开始点的路径（路径结束）。

关闭一条打开的子路径。写在stroke之前。

写在fill之前或之后填充效果相同，因为fill也会自动密闭。

CSS3理论知识回顾

1. – CSS3概述及选择器
2. 属性选择器

（1）E[att ^= "val"]属性选择器

[id ^= sub]{

background-color: rgb(129,26,91);

}//标记定义了att属性，att属性值是以val**开头**的字符串

（2）E[att $= "val"]属性选择器att属性值是以val结尾的字符串

（3）E[att \*= "val"]属性选择器att属性值是包含val的字符串

1. 伪元素选择器

|  |  |
| --- | --- |
| **选择器名称** | **说明** |
| 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。 |

第14章 – 文本、字体与颜色

1.文本样式

文本阴影

text-shadow: 3px 3px 5px black;

//水平向右，垂直向下，模糊距离为5像素的黑阴影。

文本自动换行：

（1）设置当前行超过指定容器的边界时是否断开转行，默认值为normal。

word-wrap: normal | break-word;



1. 可以让浏览器实现半角空格或连字符后面的换行，而且可以实现任意位置的换行。默认值为normal。 word-break: normal | break-all | keep-all



2.颜色样式

① rgba(red, green, blue, <opacity>)

② hsl (hue, saturation, lightness)

色相（Hue）、饱和度（ Saturation ）和亮度（ Lightness ）

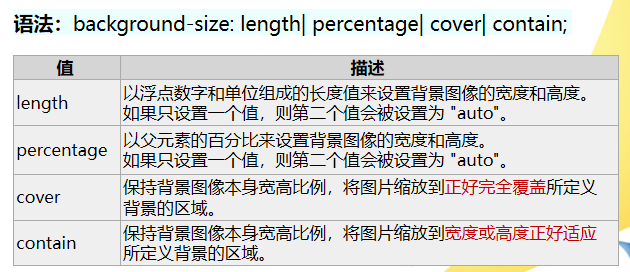
③HSLA 颜色

hsla(hue, saturation, lightness, opacity)

第15章 – 背景与边框

1.背景样式（了解属性的使用）

背景图片的尺寸



背景图片的定位区域

background-origin: border-box | padding-box | content-box;

多个背景图像

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

背景图像的绘制区域



2.边框样式

圆角边框：

语法 border-radius: 1-4 length | %;

第一个值表示圆角的水平半径，第二个值表示圆角的垂直半径，两个参数值通过斜线分隔。如果仅包含一个参数值，表示两个值相同，即1/4圆角。

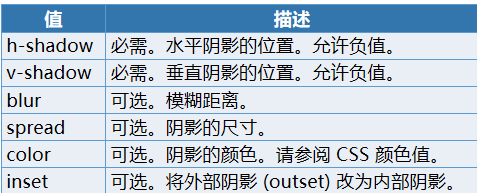
border-width：上右下左；

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。

1. 元素阴影



1. – 渐变与变形处理

1.transform属性用于实现平移、缩放、旋转和倾斜等 2D 变换，向元素应用 2D 或 3D 转换。通过转换能够对元素进行旋转、缩放、移动、倾斜或拉伸。（**作用※**）

2.移动—translate()能够重新定位元素的坐标。**实例※**

translateX(x)：元素仅在水平方向移动（X轴移动）；

translateY(y)：元素仅在垂直方向移动（Y轴移动）；

translate(x , y)：元素在水平方向和垂直方向同时移动

实例：使class为test的划入变红，整体向下5px

        .test a:hover{

            font-weight:bold;

            background:url(images/icon2.jpg) red no-repeat 5px 12px;

            transform: translateY(-5px);

            color:seashell;

        }

1. 旋转—rotate()

rotate( ) 方法能够相对中心原点旋转指定的元素。

transform: rotate(angle )

正角度为顺时针旋转元素

负角度为逆时针旋转元素

1. 缩放—scale()

scale( ) 方法能够实现文字或图像根据中心原点进行缩放。

scaleX(x)：元素仅水平方向缩放（X轴缩放）；

scaleY(y)：元素仅垂直方向缩放（Y轴缩放）；

scale(x，y)：元素水平方向和垂直方向同时缩放；

参数 x，y 为自然数数值（可以为正、负、小数）

1. 倾斜—skew()

skew( ) 方法能够倾斜指定的元素。

skewX(x)：元素仅水平方向倾斜（X轴倾斜）；

skewY(y)：元素仅垂直方向倾斜（Y轴倾斜）；

skew(x，y)：元素水平方向和垂直方向同时倾斜；如果第二个参数为空，则默认为0，参数为负表示向相反方向倾斜。

没有写实例的只了解方法及属性、参数、作用

第17章 - CSS3动画

1.过渡transition

分写方式：

//规定设置过渡效果的 CSS 属性的名称。

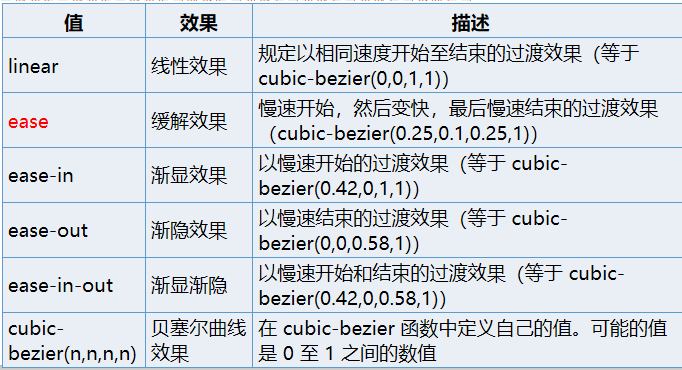
transition-property: all;

//规定完成过渡效果需要的时间。默认是 0。

transition-duration: .5s;

//规定过渡效果的时间曲线。默认是ease

transition-timing-function: ease-in;



//定义过渡效果何时开始。默认是 0。

transition-delay: .1s;

合写方式：

transition: all .5s ease-in .1s,

**实例※**

**如需向多个样式添加过渡效果，则添加多个属性，由逗号隔开**

div{

    background-color: #ffff00;

    color: #000000;

    width: 300px;

    transition: background-color 1s linear, color 1s linear, width 1s linear;

}

div:hover{

    background-color: #003366;

    color: #ffffff;

    width: 400px;

}

2.动画animation

animations 功能与 transitions 功能的相似点与不同点？

相似点：都是通过改变元素的属性值来实现动画效果。

不同点：transitions 过渡只是通过指定属性的开始值与结束值，然后以在这两个属性之间进行平滑过渡的方式来实现简单动画效果。animations 动画则是通过定义多个关键帧以及定义每个关键帧中元素的属性值来实现更为复杂的动画效果。

步骤1制作动画关键帧

步骤2调用动画

        div{

         background-color: red;

        }

        @keyframes mycolors{

         0%{

         background-color: red;

         }

         40%{

         background-color: blue;

         }

         70%{

         background-color: yellow;

         }

         100%{

         background-color: red;

         }

        }

        div:hover{

         animation-name: mycolors;//选中需要绑定到选择器的 keyframe 名称

         animation-duration: 5s;//完成动画所花费的时间

animation-itration-count:1;//规定播放次数。

        }

**3.@keyframes 规则※考**

@keyframes 动画名{

0%{

    background-color: red;

}

70%{

background-color: yellow;

}

100%{

    background-color: red;

}

}

第18章 – CSS3多列布局

1.CSS3设置多列布局的相关属性、方法



1. 创建多列布局

        #div1{

         width:1200px;

            /\* 最多三列 \*/

            column-count: 5;

            /\* 列宽，列宽过大将不会分为最多的列数 \*/

            column-width: 200px;

            /\* 列与列之间的间距 \*/

            column-gap: 50px;

        }

第19章 – CSS3盒布局

1.弹性盒布局

#container{

display: flex;

}

行内：

#container{

display: inline-flex;

}

2.弹性盒模型

第20章 – CSS3媒体查询※

1.理解自适应网页、响应式布局.

自适应网页设计（Adaptive Web Design）指能自动识别屏幕宽度、并做出相应调整的网页设计。

响应式 Web 设计(Responsive Web Design,RWD)根据设备环境以及用户行为调整布局。对交互设计和前端实现提出了更高的要求，需要考虑清楚不同分辨率下页面的布局变化、内容的缩放等。

1. CSS3使用媒体查询实现响应式布局的步骤

(1 添加媒体查询media query表达式来指定媒体类型；

(2 针对不同的浏览器窗口尺寸编写不同的样式；

(3 浏览器根据不同的窗口尺寸来选择使用不同的样式表。

3**.响应式布局实例**

@media screen and (min-width:1000px){...}

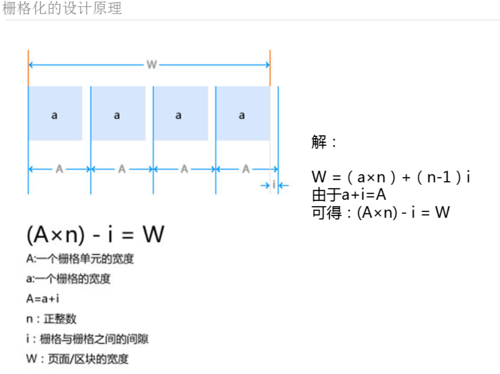
@media screen and (min-width:650px) and (max-width:1000px){...}

@media screen and (max-width:650px){...}

Bootstrap理论知识回顾

1. – Bootstrap概览及栅格系统
2. 理解栅格系统的设计原理

在网页设计中，把宽度为“W”的页面分割成n个网格单元“a”，每个单元与单元之间的间隙设为“i”，把“a+i”定义“A”。



Bootstrap提供了一套响应式、移动设备优先的流式栅格系统，随着屏幕或视窗（viewport）尺寸的增加，系统会自动分为最多12列。

1. 栅格系统的标准用法
2. 使用Bootstrap栅格系统进行网页布局的**实例※※**

        <div class="container">

            <div class="row">

             <div class="col-xs-8" style="background:red">这是第一列</div>

             <div class="col-xs-4" style="background:yellow">这是第二列</div>

            </div>

            <div class="row">

                <div class="col-md-6" style="background: pink;">hahaha</div>

                <div class="col-md-6" style="background: orange;">hahaha</div>

            </div>

        </div>

第22章 ~第23章 全局CSS

详见笔记

第24章 组件

1.缩略图，实例※.thumbnail

        <div class="container">

            <div class="row">

             <div class="col-sm-6 col-md-4">

                    <div class="thumbnail">

                        <!-- 设置圆角图片预定义类rounded -->

                        <img src="images/pic.jpg" class="img-rounded" alt="">

                        <!-- 额外插入一对div标签 -->

                        <div class="caption">

                            <h3>响应式的缩略图</h3>

                            <p>一个段落</p>

                            <a href="#" class="btn btn-success" role="button">按钮</a>

                            <a href="#" class="btn btn-warning" role="button">提交</a>

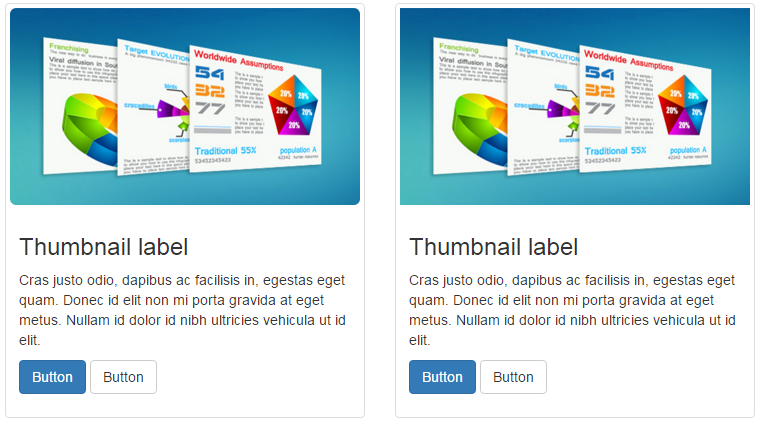
                        </div>

                    </div>

             </div>

            </div>

        </div>



1. 响应式导航条，实例※

<div class="container">

<p></p>

<!-- 给nav添加预定义类 -->

<nav class="navbar navbar-default ">

<div class="navbar-header">

<!-- navbar-toggle表示切换显示的内容 -->

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#col-1">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#">logo</a>

</div>

<div class="collapse navbar-collapse" id = "col-1">

<!-- 将无序列表增强为导航样式 -->

<ul class="nav navbar-nav">

<li><a href="#">首页</a></li>

<li><a href="#">图书馆</a></li>

<li><a href="#">AR实验室</a></li>

<li><a href="#">教务处</a></li>

</ul>

<form class="navbar-form navbar-right">

<div class="form-group">

<input type="text" class="form-control" placeholder="搜索" id=""/>

</div>

<button type="submit" class="btn btn-default">提交</button>

</form>

</div>

</nav>

</div>

1. 分页（简答）

       <ul class="pagination">//可选大小 "pagination pagination-lg"

          <li class="disabled"><a href="#">&laquo;</a></li> //禁用

          <li class="active"><a href="#">1</a></li> //激活状态

          <li><a href="#">2</a></li>

          <li><a href="#">3</a></li>

          <li><a href="#">4</a></li>

          <li><a href="#">&raquo;</a></li>

        </ul>

.pager 类



1. 插件
2. 站点引用 Bootstrap 插件的方式的两种方法

插件可以单个引入（使用 Bootstrap提供的单个\*.js文件），或一次性全部引入（使用 bootstrap.js 或压缩版的 bootstrap.min.js）。

2.模态框（理解）

模态框（Modal）是覆盖在父窗体上的子窗体。