###day08 1.width：auto width：100% 2.响应式页面 定制化：修改bootstrap的源文件 将bootstrap入口less文件作为混合使用 3.attr&prop ---什么是attribute？ html的预定义和自定义属性 ---什么是property？ js对象身上的直接属性 ---什么是布尔值属性，什么是非布尔值属性？ property所对应的属性值是否布尔类型 ---attribute和property的同步关系 非布尔值属性 实时同步 布尔值属性 没有动过property attribute会同步property property不会同步attribute 动过property attribute不会同步property property不会同步attribute ---浏览器认谁，用户操作的是谁 property ---在jQuery中的体现 attr() prop() ---总结 布尔值属性最好使用prop方法 非布尔值属性attr方法 ---html5中有关的属性 classlist：相对于class的property(className) add remove toggle dataset:自定义属性（限制 data-x-y）的property

4.html5导学  
html5的意义，到底什么是html5，html5的优势  
h4与h5的对比  
编码  
渲染模式  
mine类型  
语义化标签  
header  
footer  
section  
nav

###day09 1.注意点 ---canvas图像的渲染有别于html图像的渲染， canvas的渲染极快，不会出现代码覆盖后延迟渲染的问题 写canvas代码一定要具有同步思想 ---在获取上下文时，一定要先判断 ---画布高宽的问题 画布默认高宽300\*150 切记一定要使用html的attribute的形式来定义画布的宽高 通过css形式定义会缩放画布内的图像 ---绘制矩形的问题 a.边框宽度的问题，边框宽度是在偏移量上下分别渲染一半，可能会出现小数边框， 一旦出现小数边框都会向上取整 b.canvas的api只支持一种图像的直接渲染：矩形 ---我们没法使用选择器来选到canvas中的图像 2.画布api oc.getContext("2d"); oc.width oc.height 3.上下文api ctx.fillRect(x,y,w,h):填充矩形 ctx.strokeRect(x,ymwmh):带边框的矩形 ctx.clearRect(0,0,oc.width,oc.height):清除整个画布 注意原点的位置 ctx.fillStyle ctx.strokeStyle ctx.lineWidth ctx.lineCap ctx.lineJoin ctx.moveTo(x,y):将画笔抬起点到x，y处 ctx.lineTo(x,y):将画笔移到x，y处 ctx.rect(x,y,w,h) ctx.arc(x,y,r,degS,degE,dir) ctx.arcTo(x1,y1,x2,y2,r):2个坐标，一个半径 结合moveTo(x,y)方法使用， x,y:起始点 x1,y1：控制点 x2,y2：结束点 ctx.quadraticCurveTo(x1,y1,x2,y2) 结合moveTo(x,y)方法使用， x,y:起始点 x1,y1：控制点 x2,y2：结束点 必须经过起点和终点 ctx.bezierCurveTo(x1, y1, x2, y2, x3, y3) 结合moveTo(x,y)方法使用， x,y:起始点 x1,y1：控制点 x2,y2：控制点 x3，y3：结束点 必须经过起点和终点 ctx.fill() ctx.stroke()

ctx.beginpath():清除路径容器  
ctx.closepath():闭合路径  
fill自动闭合  
stroke需要手动闭合  
ctx.save()  
将画布当前状态(样式相关 变换相关)压入到样式栈中  
ctx.restore()  
将样式栈中栈顶的元素弹到样式容器中  
图像最终渲染依赖于样式容器  
  
ctx.translate(x,y):将原点按当前坐标轴位移x，y个单位  
ctx.rotate(弧度):将坐标轴按顺时针方向进行旋转  
ctx.scale(因子):  
放大：放大画布，画布中的一个css像素所占据的物理面积变大，画布中包含的css像素的个数变少  
画布中图像所包含的css像素的个数不变  
缩小：缩小画布，画布中的一个css像素所占据的物理面积变小，画布中包含的css像素的个数变多  
画布中图像所包含的css像素的个数不变

###day09 1.注意点 ---canvas图像的渲染有别于html图像的渲染， canvas的渲染极快，不会出现代码覆盖后延迟渲染的问题 写canvas代码一定要具有同步思想 ---在获取上下文时，一定要先判断 ---画布高宽的问题 画布默认高宽300\*150 切记一定要使用html的attribute的形式来定义画布的宽高 通过css形式定义会缩放画布内的图像 ---绘制矩形的问题 a.边框宽度的问题，边框宽度是在偏移量上下分别渲染一半，可能会出现小数边框， 一旦出现小数边框都会向上取整 b.canvas的api只支持一种图像的直接渲染：矩形 ---我们没法使用选择器来选到canvas中的图像 2.画布api oc.getContext("2d"):获取画布的2d上下文 oc.width:画布在横向上css像素的个数 oc.height:画布在纵向上css像素的个数 oc.toDataUrl():拿到画布的图片地址 3.上下文api ctx.fillRect(x,y,w,h):填充矩形 ctx.strokeRect(x,ymwmh):带边框的矩形 ctx.clearRect(0,0,oc.width,oc.height):清除整个画布 注意原点的位置 ctx.fillStyle: 填充颜色 背景fillStyle的值可以是createPattern(image, repetition)返回的对象 线性渐变fillStyle的值可以是createLinearGradient(x1, y1, x2, y2))返回的对象 addColorStop(position, color) 径向渐变fillStyle的值可以是createRadialGradient(x1, y1, r1, x2, y2, r2)返回的对象 addColorStop(position, color) ctx.strokeStyle:线条颜色 ctx.lineWidth：线条宽度 ctx.lineCap：线条两端的展现形式 ctx.lineJoin：线条连接处的展现形式 ctx.moveTo(x,y):将画笔抬起点到x，y处 ctx.lineTo(x,y):将画笔移到x，y处 ctx.rect(x,y,w,h) ctx.arc(x,y,r,degS,degE,dir) ctx.arcTo(x1,y1,x2,y2,r):2个坐标，一个半径 结合moveTo(x,y)方法使用， x,y:起始点 x1,y1：控制点 x2,y2：结束点 ctx.quadraticCurveTo(x1,y1,x2,y2) 结合moveTo(x,y)方法使用， x,y:起始点 x1,y1：控制点 x2,y2：结束点 必须经过起点和终点 ctx.bezierCurveTo(x1, y1, x2, y2, x3, y3) 结合moveTo(x,y)方法使用， x,y:起始点 x1,y1：控制点 x2,y2：控制点 x3，y3：结束点 必须经过起点和终点 ctx.fill() ctx.stroke()

ctx.beginpath():清除路径容器  
ctx.closepath():闭合路径  
fill自动闭合  
stroke需要手动闭合  
ctx.save()  
将画布当前状态(样式相关 变换相关)压入到样式栈中  
ctx.restore()  
将样式栈中栈顶的元素弹到样式容器中  
图像最终渲染依赖于样式容器  
  
ctx.translate(x,y):将原点按当前坐标轴位移x，y个单位  
ctx.rotate(弧度):将坐标轴按顺时针方向进行旋转  
ctx.scale(因子):  
放大：放大画布，画布中的一个css像素所占据的物理面积变大，画布中包含的css像素的个数变少  
画布中图像所包含的css像素的个数不变  
缩小：缩小画布，画布中的一个css像素所占据的物理面积变小，画布中包含的css像素的个数变多  
画布中图像所包含的css像素的个数不变  
  
ctx.drawImage(img,x,y,w,h)  
:在canvas中引入图片一定在图片加载完成之后再去操作  
ctx.measureText("文本")  
:返回一个持有文本渲染宽度的对象  
ctx.fillText()  
ctx.strokeText()  
ctx.font  
ctx.textAlign  
ctx.textBaseline  
shadowOffsetX = float  
shadowOffsetY = float  
shadowBlur = float  
shadowColor = color(必需项)  
ctx.getImageData(x,y,w,h)  
ImageData对象  
width：选中区域在横向上css像素的个数  
height：选中区域在纵向上css像素的个数  
data:数组  
选中区域所有像素点的rgba信息，rgba的取值从0到255  
ctx.putImageData(imgdata,x,y)  
ctx.createImageData(w,h)  
返回的是imgdata对象 默认像素点的信息是rgba(0,0,0,0)  
ctx.globalAlpha  
取值为0到1  
ctx.globalCompositeOperation  
source-over(默认值):源在上面,新的图像层级比较高  
source-in :只留下源与目标的重叠部分(源的那一部分)  
source-out :只留下源超过目标的部分  
source-atop:砍掉源溢出的部分  
  
destination-over:目标在上面,旧的图像层级比较高  
destination-in:只留下源与目标的重叠部分(目标的那一部分)  
destination-out:只留下目标超过源的部分  
destination-atop:砍掉目标溢出的部分  
ctx.ispointinpath(x,y)  
x,y这个点是否在路径上  
4.实例  
时钟动画：结合了所有基础api  
飞鸟动画：结合图片创建动画  
马赛克：像素操作  
刮刮卡：合成+像素操作

###day11复习 1.音视频标签 兼容（音视频的转码） attribute src width hieght controls autoplay loop muted preload property 音量和静音属性不回同步，静音属性优先级更高 load函数： 重新加载（结合source标签的时候才有用）

2.播放器  
less的运用  
拖拽

###Html5 HTML5 是定义 HTML 标准的最新的版本。 该术语表示两个不同的概念： 它是一个新版本的HTML语言，具有新的元素，属性和行为， 它有更大的技术集，允许更多样化和强大的网站和应用程序。 这个集合有时称为HTML5和朋友，通常缩写为HTML5。

HTML5     约等于     HTML + CSS + JS

###Html5优势 跨平台:唯一一个通吃PC MAC Iphone Android等主流平台的跨平台语言 快速迭代 降低成本 导流入口多 分发效率高

###H5 1.DOCTYPE和浏览器渲染模式 DOCTYPE，或者称为 Document Type Declaration（文档类型声明，缩写 DTD） 通常情况下，DOCTYPE 位于一个 HTML 文档的最前面的位置，位于根元素 HTML 的起始标签之前。 因为浏览器必须在解析 HTML 文档正文之前就确定当前文档的类型，以决定其需要采用的渲染模式， 不同的渲染模式会影响到浏览器对于 CSS 代码甚至 JavaScript 脚本的解析

到目前为止，各浏览器主要包含了三种模式。在 HTML5 草案中，更加明确的规定了模式的定义：  
传统名称   HTML5 草案名称 document.compatMode 返回值  
standards mode(strict mode) no-quirks mode CSS1Compat  
almost standards mode limited-quirks mode CSS1Compat  
quirks mode quirks mode BackCompat

document.compatMode  
document.compatMode 属性最初由微软在 IE 中创造出来，这是一个只读的属性，返回一个字符串，  
只可能存在两种返回值：  
BackCompat：标准兼容模式未开启（怪异模式）  
CSS1Compat：标准兼容模式已开启（标准模式）  
​  
在现代主流浏览器中，其实怪异模式的渲染和标准与几乎标准间没有太大的差别（ie9+ 谷歌 火狐 ...）  
  ie5.5之前都是ie自己的渲染模式，怪异模式  
  ie6才开始慢慢支持标准，标准模式，在ie6 中怪异和标准模式的区别最大  
  ie7 8 9都是基于标准模式升级的，他们理论上存在怪异模式  
​  
HTML5提供的<DOCTYPE html>是标准模式，向后兼容的,等同于开启了标准模式，  
那么浏览器就得老老实实的按照W3C的 标准解析渲染页面  
一个不含任何 DOCTYPE 的网页将会以 怪异(quirks) 模式渲染。

2.DTD  
<!DOCTYPE html>  
当 doctype 信息如上时，表明该页面是遵守了 HTML5 规范的，浏览器会选择 Standards Mode，这种 doctype 是最推荐的一种，  
我们平时设计页面都应该加上这一个 doctype。  
HTML5 不基于 SGML,所以不需要引用 DTD,浏览器内部本身有对标签进行解析渲染验证的模块  
  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  
当 doctype 如上时，浏览器同样会选择 Standards Mode，虽然和第一种 doctype 有一些区别，但是几乎可以认为是一样的。  
​  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  
当 doctype 如上时，浏览器会选择 Almost Standards Mode，渲染时和标准会有一些区别  
​  
当 doctype 缺失的时候，浏览器会选择 Quirks Mode，这是非常不推荐的方式，  
当 doctype上面有注释，标签或者空行时，某些浏览器都会认为该页面不具有doctype。  
我们应该尽量避免 Quirks Mode，这对一个 web 应用是非常不利的地方。

4.每种doctype在浏览器中对应的渲染模式  
https://en.wikipedia.org/wiki/Quirks\_mode

###根元素 H4中的根元素: <html xmlns="http://www.w3.org/1999/xhtml">

首先这个标记没有任何问题，你喜欢的话,那就背下来继续用。它是有效的。  
但这个标记中的很多字节在Html5中我们都可以省略了  
  
xmlns:这是XHTML1.0的东西，  
它的意思是在这个页面上的元素都位于http://www.w3.org/1999/xhtml这个命名空间内  
但是HTML5中的每个元素都具有这个命名空间，不需要在页面上再显示指出

H5中的根元素  
<html></html>

###head元素 MIME类型: 每当浏览器请求一个页面时，web服务器会在发送实际页面内容之前，先发送一些头信息。 浏览器需要这些信息来决定如何解析随后的页面内容。最重要的是Content-Type

比如: Content-Type:text/html  
  
text/html:即这个页面的"内容类型",或者称为MIME类型。这个头信息将唯一确定某个资源的本质是什么  
也决定了它应该如何被呈现。  
  
图片也有自己的MIME类型   
jpg:image/jpeg     
png:image/png  
  
js也有自己的MIME类型，css也有自己的MIME类型，  
任何资源都有自己的MIME类型，整个web都依靠MIME类型来运作

<meta charset="UTF-8">:  
告诉浏览器你应该使用哪种编码来解析网页

###语义化标签 在HTML 5出来之前，我们用div来表示页面头部，章节，页脚等。但是这些div都没有实际意义。 各大浏览器厂商分析了上百万的页面，从中发现了DIV名称的通用id名称大量重复。例如，很多开发人员喜欢使用 div id="footer"来标记页脚内容，所以Html5元素引入了语义化标签（一组新的片段类元素）

https://dev.opera.com/blog/presentation-html5-and-accessibility-sitting-in-a-tree-4/idlist-url.htm  
https://dev.opera.com/blog/presentation-html5-and-accessibility-sitting-in-a-tree-4/classlist-url.htm  
​  
<hgroup></hgroup>  
<header></header>  
<nav></nav>  
<section></section>  
<footer></footer>  
<article></article>  
<aside></aside>

语义化的好处  
HTML5可以让很多更语义化结构化的代码标签代替大量的无意义的div标签  
这种语义化的特性提升了网页的质量和语义  
对搜索引擎更加的友好  
他们这些标签功能就是代替<div>功能中的一部分，他们没有任何的默认样式，除了会让文本另起一行外；  
https://gsnedders.html5.org/outliner/

hgroup元素代表 网页 或 section 的标题，当元素有多个层级时，该元素可以将h1到h6元素放在其内，譬如文章的主标题和副标题的组合  
​  
<hgroup>  
  <h1>HTML 5</h1>  
  <h2>这是一篇介绍HTML 5语义化标签和更简洁的结构</h2>  
</hgroup>  
​  
hgroup使用注意：  
如果只需要一个h1-h6标签就不用hgroup  
如果有连续多个h1-h6标签就用hgroup  
如果有连续多个标题和其他文章数据，h1-h6标签就用hgroup包住，和其他文章元数据一起放入header标签

header 元素代表 网页 或 section 的页眉。  
通常包含h1-h6元素或hgroup  
​  
<header>  
  <hgroup>  
      <h1>网站标题</h1>  
      <h2>网站副标题</h2>  
  </hgroup>  
</header>  
  
header使用注意：  
可以是“网页”或任意“section”的头部部分  
没有个数限制。  
如果hgroup或h1-h6自己就能工作的很好，那就不要用header。

nav元素代表页面的导航链接区域。用于定义页面的主要导航部分。  
​  
<nav>  
  <ul>  
      <li>HTML 5</li>  
      <li>CSS3</li>  
      <li>JavaScript</li>  
  </ul>  
</nav>  
  
nav使用注意：  
用在整个页面主要导航部分上，不合适就不要用nav元素；

section元素代表文档中的 节 或 段，段可以是指一篇文章里按照主题的分段；节可以是指一个页面里的分组。  
​  
<section>  
  <h1>section是啥？</h1>  
  <article>  
      <h2>关于section</h1>  
      <p>section的介绍</p>  
      <section>  
          <h3>关于其他</h3>  
          <p>关于其他section的介绍</p>  
      </section>  
  </article>  
</section>  
  
section使用注意：  
section不是一般意义上的容器元素，如果想作为样式展示和脚本的便利，可以用div。  
article、nav、aside可以理解为特殊的section，  
所以如果可以用article、nav、aside就不要用section，没实际意义的就用div

article元素最容易跟section和div容易混淆，其实article代表一个在文档，页面或者网站中自成一体的内容  
​  
<article>  
  <h1>一篇文章</h1>  
  <p>文章内容..</p>  
  <footer>  
      <p><small>版权：html5jscss网所属，作者：damu</small></p>  
  </footer>  
</article>  
  
article使用注意：  
独立文章：用article  
单独的模块：用section  
没有语义的：用div

aside元素被包含在article元素中作为主要内容的附属信息部分，其中的内容可以是与当前文章有关的相关资料、标签、名次解释等  
  
在article元素之外使用作为页面或站点全局的附属信息部分。最典型的是侧边栏，其中的内容可以是日志串连，其他组的导航，甚至广告，这些内容相关的页面。  
  
<article>  
  <p>内容</p>  
  <aside>  
      <h1>作者简介</h1>  
      <p>小北，前端一枚</p>  
  </aside>  
</article>  
  
aside使用总结：  
aside在article内表示主要内容的附属信息，  
在article之外则可做侧边栏  
如果是广告，其他日志链接或者其他分类导航也可以用

footer元素代表 网页 或 section 的页脚，通常含有该节的一些基本信息，譬如：作者，相关文档链接，版权资料。  
​  
<footer>  
  COPYRIGHT@damu  
</footer>  
  
footer使用注意：  
可以是 网页 或任意 section 的底部部分；  
没有个数限制，除了包裹的内容不一样，其他跟header类似。