<!--autoplay属性表示资源加载完成后自动播放 controls表示根据浏览器的实现显示默认的播放器控件 loop属性表示循环 则音乐播放完毕后自动循环-->  
<audio autoplay controls loop>**您的浏览器太垃圾了 滚吧** <!--将多个资源放置audio标签内部 浏览器会自动查找支持的资源进行加载-->  
 <source src=**"./tree.mp3"** type=**"audio/mpeg"**>  
 <source src=**"horse.mp3"** type=**"audio/mpeg"**>  
</audio>  
<!--ie9之前的版本不支持该标签 可以在内部定义内容作为不支持的浏览器的提示-->  
  
  
<video src=**"https://gss3.baidu.com/6LZ0ej3k1Qd3ote6lo7D0j9wehsv/tieba-smallvideo-transcode/45133344\_5afa5db82b12a79a0a86f211b43668dd\_3.mp4?authorization=bce-auth-v1%2Fde94045c2e42438fad71ab8df47a6727%2F2017-05-09T13%3A09%3A25Z%2F1800%2F%2Ffd0a6e0cfe3f29a07729518520cd5c63cf81e0bd8ab74ad40ae49cbdc455c8b2"** controls poster=**"./timg.jpg"**></video>  
<!--video的src支持跨域 可以直接连接某个地址播放 autoplay controls loop 和上面一样 poster=url当视频播放前显示的图像-->  
<!--href不仅可以指向其他页面的url 也可以指向当前页面中定义的某个a标签的id或者name不过要加#号 表示转跳到该位置 a是内联标签-->  
<a href=**"#xxx"**>**toxxxx**</a>  
<a id=**"xxx"** style=**"**position: **absolute**;top:2000**px**; **"**>**xxx**</a>  
  
<!--nav是块级元素 用于作为导航 其实和div一样 不过是一种规范-->  
<nav>  
 <a href=**"http://www.baidu.com"**>**Home**</a>  
 <a href=**"./timg.jpg"**>**Previous**</a>  
</nav>  
  
<!--没有序列号的列表标签-->  
<ul>  
 <li >**1**</li>  
 <li >**2**</li>  
 <li >**3**</li>  
</ul>  
  
<!--有序列号的列表标签-->  
<ol>  
 <li>**4**</li>  
 <li>**5**</li>  
 <li>**6**</li>  
</ol>  
<!--dl也是列表标签 不过内部的dt标签可以作为一个标题 然后dd为条目 最常用的是ul-->  
<dl>  
 <dt>**一**</dt>  
 <dd>**1**</dd>  
 <dd>**2**</dd>  
 <dt>**二**</dt>  
<dd>**1**</dd>  
<dd>**2**</dd>  
</dl>  
<!--border属性是设置整个表格的所有边框粗细 cellspacing设置每个单元格之间的空隙 cellpadding设置单元格内的内间距 这三个属性每次建表格都要用 否许需要自己手动定义td的边框样式-->  
<table border=**"1"** cellspacing=**"0"** cellpadding=**"0"**>  
 <caption>**表格**</caption>  
 <!--表格的标题 一串文字 会显示在表格的顶部-->  
  
 <!--thead和tbody都是没有效果的标签 只是用来规范该位置是头和身体-->  
 <thead>  
 <tr>  
 <!--th与td一样 不过呈现粗体字 一般用来表示列名-->  
 <!--colspan表示该单元格横跨几个单位 td th都有该属性-->  
 <!--rowspan表示该单元格竖跨几个单位 td th都有该属性-->  
 <th colspan=**"2"**>**名字**</th>  
 <th>**年龄**</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr>  
 <td colspan=**"2"**>**name**</td>  
 <td>**age**</td>  
 </tr>  
 <tr>  
 <td >**Vijay**</td>  
 <td>**22**</td>  
 </tr>  
 </tbody>  
</table>

## Canvas api:

const *container*=*document*.getElementById("container");  
const *brush*=*container*["getContext"]("2d");  
// //fillStyle表示填充工具的颜色  
// brush.fillStyle="#0000ff";  
// //填充一个矩形 参数为x,y,width,height  
// brush.fillRect(20,20,150,100);  
  
// //strokeStyle是设置画笔颜色  
// brush.strokeStyle="#0000ff";  
// //画一个矩形 参数为x,y,width,height  
// brush.strokeRect(20,20,150,100);  
  
// const grd=brush.createLinearGradient(20,20,150,100);  
// //创建一个渐变的线性 x轴渐变为从20到150 y轴的渐变为从20到100 顺序也是从0到1  
// grd.addColorStop(0,"black");  
// grd.addColorStop(.5,"white");  
// grd.addColorStop(1,"blue");  
// //给画笔设置为渐变  
// brush.strokeStyle=grd;  
// brush.strokeRect(20,20,150,100);  
  
// const grd=brush.createRadialGradient(85,50,5,90,60,100);  
// //创建一个渐变的放射性 x轴渐变为从20到150 y轴的渐变为从20到100 顺序也是从0到1 第三和第六个参数是内部圆开始的半径到结束的半径的渐变  
// grd.addColorStop(0,"black");  
// grd.addColorStop(1,"white");  
// //给画笔设置为渐变  
// brush.fillStyle=grd;  
// brush.fillRect(20,20,150,100);  
  
// const grd=brush.createLinearGradient(20,20,150,100);  
// //创建一个渐变的线性 x轴渐变为从20到150 y轴的渐变为从20到100 顺序也是从0到1  
// grd.addColorStop(0,"black");  
// grd.addColorStop(.5,"white");  
// grd.addColorStop(1,"blue");  
// //给画笔设置字体  
// brush.font="30px Verdana";  
// //给画笔设置为渐变  
// brush.strokeStyle=grd;  
// //设置画出来的图像的阴影颜色  
// brush.shadowColor="black";  
// //设置画出来图像的模糊程度  
// brush.shadowBlur=10;  
// //设置模糊背景的偏移量  
// brush.shadowOffsetX=10;  
// brush.shadowOffsetY=10;  
// //用画笔画出字体（空心的）如果需要画实心的字则用fill 方法是一样的  
// brush.strokeText("Hello",100,100,1000);  
  
// const img=new Image();  
// img.src="./timg.jpg";  
// img.onload=()=>{  
// const pat=brush.createPattern(img,"repeat");  
// //给画笔设置图片和对应图片的重复策略  
// brush.fillStyle=pat;  
// brush.fillRect(20,20,1500,1000);}  
  
// //开始定义路径  
// brush.beginPath();  
// //设置线条末尾的线帽"butt|round|square"  
// brush.lineCap="round";  
// //设置折线的样式"bevel|round|miter";  
// brush.lineJoin="round";  
// //设置线条宽度  
// brush.lineWidth=20;  
// //往前移动x,y  
// brush.moveTo(20,20);  
// //往前画到x,y  
// brush.lineTo(200,50);  
// //再从上一个画到的点往下画 从x为200的地方画到x为20的地方 y为50画到y为300  
// brush.lineTo(20,300);  
// //开始绘制路径  
// brush.stroke();  
  
// brush.beginPath();  
// brush.lineCap="round";  
// brush.lineJoin="round";  
// brush.lineWidth=10;  
// brush.moveTo(50,50);  
// printY(150,function () {  
// printY(125,function () {  
// printX(50,function () {  
//  
// })  
// },false);  
// },true);  
//  
//  
// function printY(y,cb,async){  
// if(async){  
// for(var i=50;i<y+50;i++){  
// (function (size) {  
// setTimeout(()=>{  
// brush.lineTo(50,size);  
// brush.stroke();  
// if(size===y+50-1){  
// cb();  
// }  
// },10\*size);  
// })(i)  
// }}else{  
// for(var i=50;i<y+50;i++){  
// (function (size) {  
// setTimeout(()=>{  
// brush.lineTo(50,size);  
// brush.stroke();  
// if(size===y+50-1){  
// cb();  
// }  
// },0);  
// })(i)  
// }}  
//  
// }  
// function printX(x,cb){  
// for(var i=50;i<x+50;i++){  
// (function (size) {  
// setTimeout(()=>{  
// brush.lineTo(size,125);  
// brush.stroke();  
// if(size===x+50-1){  
// cb();  
// }  
// },10\*size);  
// })(i)  
// }  
//  
// }  
  
*brush*.beginPath();  
*brush*.moveTo(20,20);  
*brush*.lineWidth=5;  
*brush*.lineCap="round";  
*brush*.lineJoin="round";  
*brush*.lineTo(20,100);  
*brush*.lineTo(70,100);  
//返回到路径的起始点  
*brush*.closePath();  
//把当前上下文进行旋转  
*brush*.rotate(10\**Math*.PI/180);  
*brush*.scale(2,2);  
*brush*.globalAlpha=0.6; //设置当前上下文对象的透明度  
//移动当前上下文环境  
*brush*.stroke();  
  
//设置当前上下文放大两倍 之后的所有操作尺寸都放大两倍 如果继续放大的话 是在这个基础上放大的  
*brush*.moveTo(20,20);  
*brush*.lineTo(20,100);  
*brush*.lineTo(70,100);  
//返回到路径的起始点  
*brush*.closePath();  
*brush*.globalAlpha=0.6; //设置当前上下文对象的透明度  
*brush*.stroke();  
  
const *gdt*=*brush*.createRadialGradient(85,50,5,90,60,100);  
*gdt*.addColorStop(0,"black");  
*gdt*.addColorStop(.5,"white");  
*gdt*.addColorStop(1,"blue");  
//判断当前画笔所在的坐标  
*console*.info(*brush*.isPointInPath(20,20));  
*brush*.fillStyle=*gdt*;  
*brush*.fill();  
  
  
  
// //画图片  
// const img=new Image();  
// img.src="./timg.jpg";  
// img.onload=function () {  
// brush.drawImage(img,20,20);  
// };  
  
  
// //参数为width,height  
// const imageData=brush.createImageData(100,100);  
// //给每个像素点赋值rgba颜色 0123为第一个像素点 4567为第二个像素点 以此类推 直到布满整个长宽  
// for (var i=0;i<imageData.data.length;i+=4)  
// {  
// imageData.data[i+0]=255;  
// imageData.data[i+1]=2;  
// imageData.data[i+2]=255;  
// imageData.data[i+3]=255;  
// }  
// brush.putImageData(imageData,0,0);  
// console.info(brush.getImageData(20,20,1,1)); //可以获取到画布上的某个坐标的像素点的颜色