HTML5

一、HTML5 新特性：表单新特性——新的input type

<input type="?">

**已经学过的input type：**

text、password、radio、checkbox、button、reset、submit、image、file、hidden

**HTML5新添的inputtype：**

(1)number：显示一个数字输入框

<input type="number" min="1" max="9" step="2"/>

(2)email：显示一个邮件地址输入框，提交时会进行格式验证，中间一个@符号，前后至少一个字符

<input type="email"/>

(3)url：显示一个URL地址输入框，提交时会进行格式验证，冒号前存在任意字符

<input type="url"/>

(4)tel：显示一个电话号码输入框，没有格式验证，仅在手机浏览器中弹出数字输入键盘

<input type="tel"/>

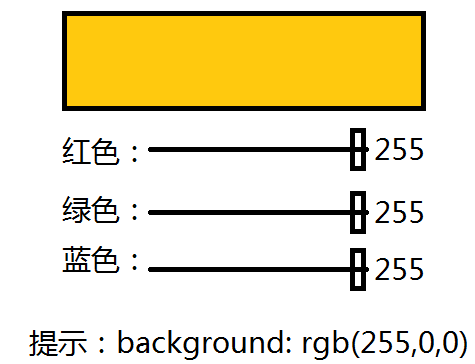
(5)search：显示搜索输入框，PC中Chrome下有X号；手机中弹出键盘右下角按钮显示文字“搜索”

<input type="search"/>

(6)range：显示一个范围选择滑块，让用户在特定范围内选择一个整数值

<input type="range" min="0" max="255" value="50" step="5"/>

练习：实现一个调色板，修改某个滑块的位置，可以改变背景颜色



(7)color：显示一个颜色选择窗口，浏览器会调用操作系统提供颜色选择窗口，提交的是颜色值

<input type="color">

(8)date：显示一个日期选择窗口，浏览器弹出一个日期选择窗口，不是DIV，不能定制样式

<input type="date" min="1949/10/01">

(9)month：显示一个月份选择器

<input type="month">

(10)week：显示一个“星期/周”选择器

<input type="week">

二、HTML5 新特性：表单新特性——新的表单元素

**HTML5之前已有的表单元素——可与用户交互并提交：**

input、select/option、textarea、button

**HTML5中新增的表单元素——不能与用户交互，不能提交，仅用于信息展示：**

(1)datalist：数据列表，为某个输入框提供输入建议

<datalist id="dish">

<option>京酱肉丝</option>

<option>鱼香肉丝</option>

</datalist>

<input list="dish">

(2)progress：进度条，显示前进的进度提示条

<progress value=""></progress>

练习：使用定时器，修改progress的进度，实现不断前进的效果

(3)meter：度量衡，刻度尺，使用不同的颜色标示出数据所处的区间

<meter min="0" low="30" value="0" high="70" optimum="50" max="100"></meter>

最优值在上下限： 黄 - 绿 - 黄

最优值小于下限：绿 - 黄 - 红

最优值大于上限：红 - 黄 - 绿

(4)output：输出值，用于表示数据的计算结果，是一个语义标签

<output for="price count"></output>

三、HTML5 新特性：表单新特性——表单元素新的属性——重点

**HTML5之前表单标签的常用属性：**

name、value、readonly、disabled、checked、for

**HTML5中表单标签的新属性：**

(1)placeholder：占位消息，为输入框中添加提示消息，注意：占位消息仅用作提示，与value属性根本不同，不能被提交

<input placeholder="提示性文字">

(2)autofocus：自动获得焦点，标识了该属性的输入框会自动的获得输入焦点

<inputautofocus>

(3)multiple：多项，用于email和file输入框，允许其中输入多个值，用逗号分隔

<input type="email/file" multiple>

(4)autocomplete：自动完成，可取值**on**/off，指定输入域是否记录上一次提交的输入，下次输入时给予提示

<input autocomplete="off">

(5)form：表单，指定当前输入域所属的表单的ID，可以将声明了form的输入框放到表单元素的外面，仍然可以一同被提交

<form id="formID"></form>

<input form="formID">

=========与输入验证相关新属性========

(6)required：必填项，若没有输入则无法提交

<input required>提交时才会验证

(7)min：最小值，若输入值小于该值则无法提交

<input type="number/date" min="18">

(8)max：最大值，若输入值大于该值则无法提交

<input type="number/date" max="60">

(9)minlength：最小长度，若输入的字符串长度小于该值无法提交，此属性不是H5标准属性，FF不支持，Chrome支持

<input minlength="6" required>

(10)maxlength：最大长度，若输入的字符串长度大于该值无法提交

<input maxlength="9">

(11)pattern：正则表达式样式，若输入的字符串不符合指定的正则表达式则无法提交

<input pattern="^1[3578]\d{9}$">

|  |
| --- |
| 归纳总结练习——表单新特性：  (1)新的input type——10个  number、email、url、tel、search、range、color、date、month、week  (2)新的表单元素——4个  datalist、progress、meter、output  (3)表单元素新的属性——11个  placeholder、autofocus、autocomplete、multiple、form  ----------------------  required、min、max、minlength、maxlength、pattern |

提示：上述红色的特性可能在提交表单时弹出错误提示。

6.如何自定义表单错误提示消息

HTML5为每个表单输入域添加了一个属性：validity（有效性）

ValidityState {

valid : true

badInput : false 无效的输入，number

customError: false 自定义错误，setCustomValidity('')参数字符串有内容，curstomError就变为true；参数值是空字符串，customError变为false

patternMismatch: false 样式不匹配，pattern

rangeOverflow: false 范围上溢出，max

rangeUnderflow: false 范围下溢出，min

stepMismatch: false 步长不匹配，step

tooLong: false 字符串太长，maxlength

tooShort : false 字符串太短，minlength

typeMismatch: false 类型不匹配，email/url

valueMissing: false 值缺失，required

}

该对象中的属性值随着输入域中值的改变而立即改变，不会等到表单提交。其中只有valid为true时，表示输入域中的值是有效的，才能被提交；只要其它的某个属性值为true，valid属性值就会变为false。

注意：只要自定义了错误消息，浏览器默认的错误消息就不再显示。

晚间练习：

在输入域失去焦点时，使用其validity属性的各个布尔类型的值，验证用户的输入是否合法。下面的效果图可以放大。





1.HTML5取代Flash体现在哪些方面

Flash-绘图、动画、游戏 —— HTML5- Canvas & SVG

Flash-视频、音频 —— HTML5 - Video & Audio

Flash-客户端存储 —— HTML5 -WebStorage

2.HTML5高级新特性——视频播放

<video src=""></video>在HTML5中专用于播放视频。

VIDEO标签默认是一个300\*150的inline-block；

Video标签/对象常用的成员：

|  |
| --- |
| 成员属性：  autoplay: false，是否自动播放  controls: false，是否显示播放控件  currentTime: 1.4901，当前播放到的时间  duration: 60.1012，总时长  defaultMuted: false, 默认是否静音  loop: false，是否自动循环播放  muted: false，当前是否静音  paused: false，当前是否处于暂停状态  poster：''，仅在影片开始播放之前显示的“电影海报”  volumn：1, 当前播放音量  preload: 'auto'，如何预加载影片内容，可取值有三个：  auto：自动预加载影片元数据，并缓冲一定的长度  metadata：仅预加载影片元数据（宽高、时长）  none：不预加载影片的任何数据 |
| 成员方法：  play() 让影片开始播放，paused属性变为false  pause() 让影片开始暂停，paused属性变为true |
| 成员事件：  onplay：fn 当调用了v.play()方法时触发  onpause: fn 当调用了v.pause()方法时触发 |

练习：禁用Video的默认播放控件，自定义播放和暂停按钮，悬停在影片的正上方，点击则播放，再点击则暂停——仿优酷的效果

<div>

<video></video>

<a><img></a>

</div>

练习：不使用Video的海报(poster)属性，自定义一个广告图片，当影片播放时候，广告要隐藏；影片暂停时，显示广告。

v.onplay = function(){}

v.onpause = function(){}

3.HTML5高级新特性——音频播放

<audio src=""></audio>在HTML5中专用于播放声音。

AUDIO标签默认是一个300\*30的inline-block；若不显示播放控件则display为none。

AUDIO标签/对象常用的成员：

|  |
| --- |
| 成员属性：  autoplay: false，是否自动播放  controls: false，是否显示播放控件  currentTime: 1.4901，当前播放到的时间  duration: 60.1012，总时长  defaultMuted: false, 默认是否静音  loop: false，是否自动循环播放  muted: false，当前是否静音  paused: false，当前是否处于暂停状态  volumn：1, 当前播放音量  preload: 'auto'，如何预加载音频内容，可取值有三个：  auto：自动预加载音频元数据，并缓冲一定的长度  metadata：仅预加载音频元数据（宽高、时长）  none：不预加载音频的任何数据 |
| 成员方法：  play() 让音频开始播放，paused属性变为false  pause() 让音频开始暂停，paused属性变为true |
| 成员事件：  onplay：fn 当调用了v.play()方法时触发  onpause: fn 当调用了v.pause()方法时触发 |

练习：根据复选框是否勾选，暂停/播放背景音乐 14:27



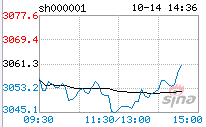
说明：

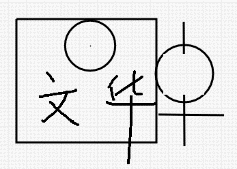
(1)早期的IE浏览器中，可以使用body的bgsound属性控制背景，但此属性不是HTML标准属性，且无法精确的控制播放和暂停：

<body bgsound="bg.mp3">

(2)iOS系统自带的Safari浏览器目前不支持AUDIO标签，只能使用隐藏的VIDEO标签来代替。

4.前端技术中可以实现绘图的技术





(1)WebGL技术 —— 当前还没有纳入HTML5标准

(2)Canvas技术 —— 在HTML5标准中提出的技术

(3)SVG技术 —— 早就存在的技术，纳入了HTML5标准

5.Canvas绘图技术 —— 比较难

提示：有两个难点

(1)属性/方法比较多，不好记忆

(2)小学数学计算(坐标轴，坐标点的位置计算)

Canvas：画布，在HTML5中默认是一个300\*150的inline-block。设定画布的宽和高不能使用CSS Style，只能使用width和height属性。

<canvas width="500" height="400">

您的浏览器不支持Canvas标签！

</canvas>

画布本身不能绘制内容，只用于承载被绘制的内容——使用画笔来往画布绘制内容。获得画笔使用原生JS：

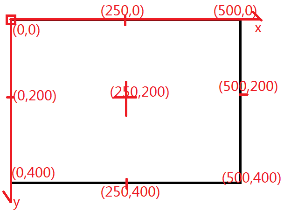
var ctx = canvas.getContext('2d'); //绘图上下文对象

画布上所有内容的绘制都要依靠“绘图上下文”对象。

Context对象常用的属性和方法：

|  |
| --- |
| 常用属性：  fillStyle:"#000000"填充样式  strokeStyle:"#000000"描边/轮廓样式  font:"10px sans-serif"  textAlign:"start"  textBaseline:"alphabetic"文本基线  globalAlpha:1全局不透明度  lineWidth:1线/描边的宽度  shadowOffsetX:0阴影在X轴上的偏移量  shadowOffsetY:0  shadowColor:"rgba(0, 0, 0, 0)"  shadowBlur:0阴影模糊半径 |
| 常用方法：  arc() 绘制一个拱形/圆形  beginPath()开始绘制一条路径——PS:钢笔工具  closePath() 闭合一条路径——PS  fill() 对路径进行填充  stroke() 对路径进行描边  moveTo() 移动到某一点  lineTo() 到另一个点绘制一条直线  fillRect() 填充一个矩形  strokeRect() 描边一个矩形  clearRect() 清空一个矩形范围内所有内容  fillText() 填充一个字符串  strokeText() 描边一个字符串  drawImage() 绘制图像 |

Content Context



6.使用绘图上下文——绘制矩形(rectangle)

提示：矩形的定位点在自己的左上角

ctx.fillStyle = '#000'/渐变对象 填充样式

ctx.strokeStyle="#000"/渐变对象 描边样式

ctx.lineWidth = 2; 描边的宽度

ctx.fillRect(x, y, w, h) 填充一个矩形

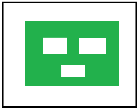
ctx.strokeRect(x,y,w,h) 描边一个矩形

ctx.clearRect(x,y,w,h) 清除一个矩形范围内的内容

|  |
| --- |
| 创建一个线性渐变对象：  var g = ctx.createLinearGradient(x1,y1, x2, y2);  g.addColorStop(offset1, color1);  g.addColorStop(offset2, color2); |

练习1：画布500\*400，左上角填充一个红色的矩形100\*80；右上角描边一个绿色的矩形100\*80；描边+填充一个矩形100\*80；右下角描边+填充另一个颜色的矩形100\*80；正中央描边+填充与右下角完全一样的矩形。

练习2：绘制如下的图形

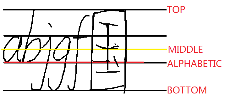


练习3：绘制一个可以在画布上左右晃动的矩形——最基础的动画模型

练习4：绘制一个上下晃动的矩形

练习5：练习一个斜向右下角30度晃动的矩形

7.使用绘图上下文——绘制文本(text)



提示：文本的定位点在整个字符串的文本基线的最左边

ctx.textBaseline = 'alphabetic'; //文本基线，可取值为top/middle/alphabetic/bottom

ctx.font = "10px sans-serif";

ctx.fillText( txt, x, y )

ctx.strokeText( txt, x, y )

ctx.measureText( txt ).width //测量文本的宽度

课后练习：

(1)使用视频做元素的背景



提示：Video自动播放、循环播放、静音，绝对定位到目标元素下面，z-index为负值即可

(2)假设使用AJAX从服务器端获取到如下的数据：

[

{"label": "部门1", "value":300},

{"label": "部门2", "value":500},

{"label": "部门3", "value":150},

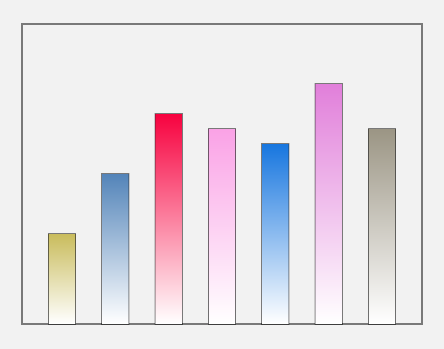
{"label": "部门4", "value":400},

{"label": "部门5", "value":600},

{"label": "部门6", "value":250}

]

根据这样的数据，绘制出如下图所示的统计图：



使用Canvas绘图——JS绘图

<canvas id="c1" width="500" height="400"></canvas>

c1.width = 500;

c1.height = 400;

提示：Canvas的尺寸不能使用CSS来设置

var ctx = c1.getContext('2d');

常用属性：

ctx.fillStyle = '#f00'/gradient;

ctx.strokeStyle = '#f00'/gradient;

ctx.lineWidth = 1;

ctx.font = '10px sanse-sarfi';

ctx.textBaseline = 'alphabetic';

ctx.shadowBlur = '';

常用方法：

绘制矩形

ctx.fillRect()

ctx.strokeRect()

ctx.clearRect()

绘制文本

ctx.fillText()

ctx.strokeText()

ctx.measureText(txt).width

绘制路径

绘制图像

今日目标：

(1)绘制路径 —— 麻烦

(2)绘制图像 —— 抽象

(3)Chart.js —— 掌握

1.使用Canvas进行绘图——绘制路径

路径：Path，类似于Photoshop中的钢笔工具，可以绘制直线、各种曲线；但路径本身是不可见的，有三种用途：

(1)创建选区(clip)，对画布内容进行裁剪

(2)进行描边(stroke)，绘制任意形状的折线

(3)进行填充(fill)，填充出任意形状的图形

相关函数：

ctx.beginPath() 开始一条新的路径

ctx.arc() 绘制一条圆形/椭圆/拱形路径

ctx.moveTo(x,y) 移动到指定点

ctx.lineTo(x,y) 从上一点到指定点绘制直线

ctx.busierCurve() 绘制贝塞尔曲线

ctx.closePath() 闭合路径

---------------------------------

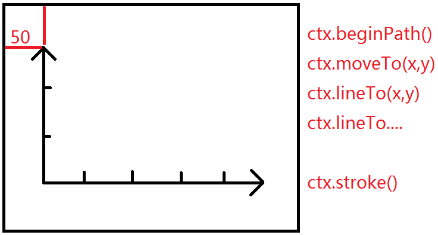
ctx.clip() 基于当前路径进行裁切

ctx.stroke() 基于当前路径进行描边

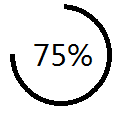
ctx.fill() 基于当前路径进行填充

练习：

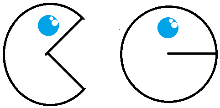
(1)使用路径绘制一个坐标轴



(2)使用定时器配合路径实现一个圆环状进度条



(3)创建两个函数，openMouth()和closeMouth()，分别绘制如下的两幅图形



2.使用Canvas绘图——绘制图像

提示：图像的定位点在自己的左上角

ctx.drawImage( img, x, y ) //原始大小绘制

ctx.drawImage( img, x, y, w, h ) //使用指定的宽高绘制图像——可能进行图像大小缩放

**注意：绘制图像时，必须等待图片异步加载完成**

var img = new Image();

img.src = "xx.png"; //向服务器异步请求图片

img.onload = function(){ //图片加载完成

ctx.drawImage( img, x, y )

}

练习：

(1)在画布的左上角、右上角、左下角、右下角绘制小星星；正中央绘制一个2倍大小的星星

(2)在画布上随机绘制大大小小随机出现的20个星星

(3)在画布上绘制一个从左上角到右下角移动的小星星

(4)在画布上原地旋转的小星星

图像的旋转问题：

(1)Canvas中的旋转不是画布旋转！而是“绘图上下文（画笔）”旋转。

ctx.rotate(deg)

(2)旋转必须有一个轴点——默认是坐标轴的原点；若想以某个固定的点旋转，就必须平移画布的坐标原点：

ctx.translate(x, y) //指定新的坐标轴原点

练习(比较难)1：绘制四个小飞机，各在一个角落以自己为轴点原地旋转

提示：绘制每个小飞机都要平移一次坐标轴原点，平移后再恢复到原来的位置供下一次使用。

练习(比较好玩)2：绘制一个可以随着鼠标移动的飞机

3.总结：Canvas绘图技术相关属性和方法——重点 17:18

ctx.fillStyle = 颜色/渐变色/样式;

ctx.strokeStyle =颜色/渐变色/样式;

ctx.font = "10px sanse-sarif";

ctx.textBaseline = '';

ctx.lineWidth = 1;

-------------------------------

(1)绘制矩形

ctx.fillRect() ctx.strokeRect() ctx.clearRect()

(2)绘制文本

ctx.fillText() ctx.strokeText() ctx.measureText()

(3)绘制路径

ctx.beginPath()

ctx.moveTo() ctx.lineTo() ctx.arc() ctx.rect()

ctx.stroke() ctx.fill() ctx.clip()

(4)绘制图像

ctx.drawImage(img,x,y,[w],[h])

ctx.rotate( deg )

ctx.translate( x, y )

4.基于Canvas的图表绘制框架/工具库

(1)Chart.js —— 免费开源，有九类图表可供使用

(2)Echart.js —— 免费的，百度提供，中文手册易上手

(3)FusionCharts.js —— 收费的，功能强大

5.第三方图表绘制工具——Chart.js的使用

(1)寻找官网：http://www.chartjs.org

(2)看官网介绍：

Simple yet flexible JavaScript charting for developers

(3)看官网提供的使用文档(API Document)

http://www.chartjs.org/docs/

(4)根据官网上的实例编写代码：

<canvas id="c2" width="600" height="400"></canvas>

<script src="js/Chart.js"></script>

<script>

new Chart(c2, {

type: 'bar', //图表类型

data: { }, //图表数据

options: { } //可选参数

});

</script>

课后练习：

1. 使用Canvas绘制一个随机改变的验证码图片——注意：真正项目中验证码图片都是由后台程序，如PHP生成的，此例子仅仅是为了练习Canvas的使用。



var str = 'ABCDEFGHJKLMNPQRSTWXY3456789';

var char = str[ 0~字符串长度间的随机数 ];

要求：

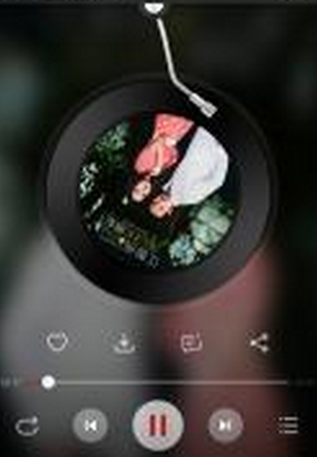
画布背景颜色随机(浅色) ctx.fillRect()

文字内容随机、大小随机，颜色随机(深色)、旋转角度随机。

5条随机干扰线（每个颜色随机深色），处于文字上方。

100个杂色点（半径为1为圆），处于文字上方。

2.使用Canvas仿写网易云音乐的播放界面



点击播放按钮，碟片开始旋转，背景音乐开始播放；

再次点击播放按钮，碟片停止旋转，背景音乐停止播放。

Canvas绘图使用方法：

<canvas id="c2" width="" height=""></canvas>

var ctx = c2.getContext('2d');

ctx.fillStyle = 颜色/渐变/样式;

ctx.strokeStyle =颜色/渐变/样式;

ctx.lineWidth = 1;

ctx.font = '10px sans-serif';

ctx.textBaseline = 'alphabetic';

ctx.showdowXxx =

---------------------------

ctx.fillRect() ctx.strokeRect() ctx.clearRect()

ctx.fillText() ctx.strokeText() ctx.measureText()

ctx.beginPath()

ctx.moveTo() ctx.lineTo() ctx.arc()

ctx.closePath

ctx.clip() ctx.fill() ctx.stroke()

ctx.drawImage()

ctx.rotate() ctx.translate()

1.补充知识点：

若Canvas绘图中需要绘制多张图片，必须等待所有图片加载完成才能开始绘制；而每张图片的加载都是异步请求，彼此没有先后顺序，哪一张先加载完成完全无法预测。

var progress = 0; //全局加载进度

var img = new Image();

img.src = 'xx.jpg';

img.onload = function(){

progress += 10; //该图片权重

if(progress===100){

startDraw();

}

}

2.补充知识点：如何为Canvas中的某个图形/图像添加事件监听

HTML中，只有标签/元素才能添加事件监听。

Canvas绘图技术，只有一个标签——Canvas。

为某个部分中的图形/图像添加事件监听，只能委托给Canvas，获取事件发生的坐标，是否处于目标图像/图形范围内。

结论：为Canvas的某个图形/图像添加事件监听非常麻烦！

今日目标：

(1)SVG绘图 —— 重点

(2)Two.js第三方绘图工具库

1.SVG绘图概述

**矢量图：** Flash/AI，每个线条/色块有独立的颜色，可以无限缩放，不善于描述颜色细节。—— SVG

**位图：** PS每个点都是一个独立的颜色，善于表现丰富的颜色细节，放大后会失真。—— Canvas

SVG：Scalable Vector Graphics，可缩放的矢量图

SVG和Canvas的比较

|  |  |
| --- | --- |
| **SVG** | **Canvas** |
| 绘制的是矢量图 | 绘制的是位图 |
| 每个图形/图像都是独立的标签 | 只有画布是标签 |
| 容易为每个图形添加事件监听 | 不能为每个图形添加独立的事件监听 |
| 可以无限缩放，适合地图 | 不能无限放大，适合表现颜色细节，例如游戏 |

2.SVG的使用方法

SVG标准诞生于2001年，远早于HTML5，后来纳入H5标准，去除了一些原有的特性。

SVG图形的使用方法

**(1)HTML5之前：**

把SVG标签写在独立的XML文档中；

在HTML文档中使用IMG、IFRAME、EMBED、OBJECT等标签引入svg文件

**(2)HTML5之后(SVG标签直接纳入H5标签库)：**

可以直接在HTML文档中使用SVG标签——HTML解释器已经可以直接解析SVG标签了。

3.使用SVG绘图——绘制矩形

注意：SVG图形的属性可以使用HTML标签属性来声明，也可以使用类似CSS的形式来声明——SVG标签专用样式；但这两种方式本质不一样。

<rect></rect>

可用属性：

width 矩形的宽

height 矩形的高

x 定位点的X坐标

y 定位点的Y坐标

fill 填充颜色

fill-opacity 填充颜色透明度

stroke 描边颜色

stroke-width 描边宽度

**提示：上述属性不属于HTMLDOM范畴，不能直接rect.x读写。但可以使用核心DOM的setAttribute()方法来操作。**

练习：

(1)点击一个矩形，它就隐藏

(2)一个矩形，从左向右移动

(3)随机生成20个大大小小的矩形

结论：使用JS创建新SVG元素有两种方法：

1)svg.innerHTML = '<rect></rect>';

2)document.createElementNS('http://www.w3.org/2000/svg', 'rect');

4.使用SVG绘图——绘制圆形

<circle></circle>

可用属性：

r 半径

cx 圆心的X坐标

cy 圆心的Y坐标

fill 填充颜色，默认为#000

fill-opacity 填充颜色透明度

stroke 描边颜色，默认为transparent

stroke-width 描边宽度

练习：

(1)使用HTML在SVG四个角各创建一个圆形

(2)使用JS创建20个随机大小、位置、颜色随机的圆形，点击某一个变大变淡，直至消失，从DOM上删除它

(3)点击SVG的某处，生成一个圆形，变大变大直至消失

5.使用SVG绘图——绘制椭圆

<ellipse></ellipse>

可用属性：

rx 横向半径

ry 纵向半径

cx 圆心的X坐标

cy 圆心的Y坐标

fill 填充颜色，默认为#000

fill-opacity 填充颜色透明度

stroke 描边颜色，默认为transparent

stroke-width 描边宽度

6.使用SVG绘图——绘制直线

<line></line>

可用属性：

x1 起点的横坐标

y1 起点的纵坐标

x2 终点的横坐标

y2 终点的纵坐标

stroke 描边颜色，默认为transparent，必须进行修改

stroke-width 描边宽度

练习：使用line元素实现下述小图标



提示：可以使用“元素组”<g></g>把多个元素包含起来，这样每个组员可以共享<g>元素声明的属性。 16:21

7.使用SVG绘图——绘制折线

<polyline></polyline>

可用属性：

points 折线上的点，值形如："0,0 10,0 50,100 ..."

fill 填充颜色，默认为#000

fill-opacity 填充颜色透明度，必须设置为0，否则会自动填充

stroke 描边颜色，默认为transparent，必须设置

stroke-width 描边宽度

练习：根据下列数据，绘制折线图，鼠标悬停时，线条的颜色改变

var data = [ [ 260, 180, 280, 150, 270 ], [130,180,220,210, 260] ];



8.使用SVG绘图——绘制多边形

<polygon></polygon>

可用属性：

points 多边形上的点，值形如："0,0 10,0 50,100 ..."

fill 填充颜色，默认为#000

fill-opacity 填充颜色透明度

stroke 描边颜色，默认为transparent

stroke-width 描边宽度

练习：使用polygon绘制下述图标



提示：真正项目中，如果需要这样的SVG图标，可以使用AI进行手绘，再使用插件导出为svg文件即可。

9.使用SVG绘图——绘制文本

<text>文本内容</text>

可用属性：

x

y

font-size

font-family

fill 填充颜色，默认为#000

fill-opacity 填充颜色透明度

stroke 描边颜色，默认为transparent

stroke-width 描边宽度

10.使用SVG绘图——绘制图像

提示：若绘制了位图在SVG上，这个SVG图放大也会产生失真。

<image></image>

可用属性：

x

y

xlink:href 指定图片的URL

width 图片的宽，默认为0，不显示

height 图片的高，默认为0，不显示

|  |
| --- |
| 矩形、圆形、椭圆、直线、折线、多边形、文本、图像 |

11.在绘制图形时使用渐变色

渐变对象是SVG中的特效对象——特效对象都必须定义在<defs></defs>标签内：

<svg id="s1" width="500" height="400">

<defs> <!--定义特效元素-->

<linearGradient id="rainbow" x1="0" y1="0" x2="100%" y2="0">

<stop offset="0" stop-color="red"></stop>

<stop offset="1" stop-color="purple"></stop>

</linearGradient>

</defs>

<rect x="50" y="100" width="400" height="200" fill="url(#rainbow)"></rect>

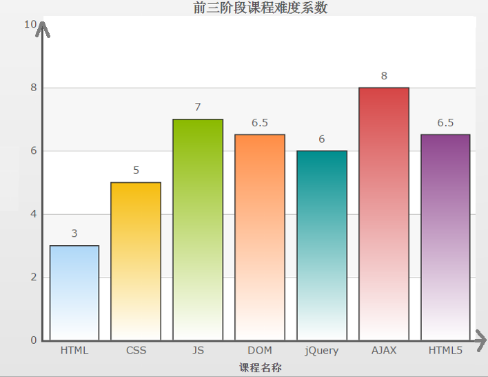
</svg>

课后练习：

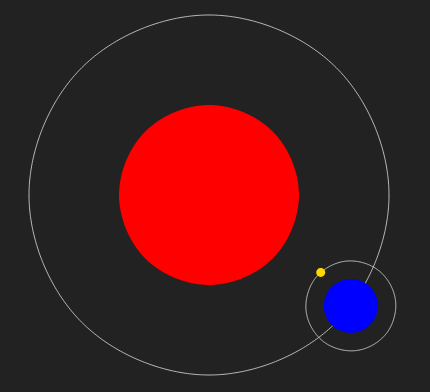
1. 用折线绘制五角星
2. 假设前端页面获取到后台返回了如下的JSON数据：

'[{"label":"HTML",value:3}, {"label":"CSS",value:5},....]'

根据这段JSON字符串，绘制出如下的统计图，要求每个柱在鼠标悬停时都会加粗显示边框：



3)提前自学 Two.js工具的使用，理解其作用，仿写官方示例代码，学会使用方法，实现如下效果——月亮绕着地球转，地球绕着太阳转：



SVG绘图：

方法1： 1.svg <img src="1.svg">

方法2： <body><svg></svg></body>

绘制矩形：

<rect x="" y="" width="" height=""></rect>

绘制圆形：

<circle cx="" cy="" r=""></circle>

绘制椭圆：

<ellipse cx="" cy="" rx="" ry=""></ellipse>

绘制直线：

<line x1="" y1="" x2="" y2=""></line>

绘制折线：

<polyline points="x1,y1 x2,y2 ..."></polyline>

绘制多边形：

<polygon points="x1,y1 x2,y2 ..."></polygon>

绘制文本：

<text x="" y="" font-size="">XXXX</text>

绘制图像：

<image xlink:href=""></image>

使用渐变：

<svg>

<defs>

<linearGradient id="g1" x1="" y1="" x2="" y2="">

<stop offset="0" stop-color="" stop-opacity="">

</linearGradient>

</defs>

<rect fill="url(#g1)" stroke="url(#g1)"></rect>

</svg>

今日目标：

(1)补充：SVG中使用滤镜

(2)Two.js第三方工具

(3)地理定位

(4)拖放API —— 比较重要

1.SVG补充知识点

在SVG中使用滤镜(filter)——对图像图像进行像素化处理。使用方法：

<defs>

<filter id="f1">

<feGaussianBlur stdDeviation="5">

</feGaussianBlur>

</filter>

</defs>

<rect filter="url(#f1)"></rect>

2.三种绘图技术

(1)WebGL —— 3D渲染

(2)Canvas —— 2D动画游戏

(3)SVG —— 矢量图无限缩放

上述三个技术都可以用于绘制统计图——线、柱、圆...。

各自的使用方法不同，增加学习和使用难度。

two.js 2D绘图函数库

three.js 3D绘图函数库

3.第三方绘图工具库——Two.js

**官网：**https://two.js.org/

**官方介绍：**

Two.js is a two-dimensional drawing api geared towards modern web browsers. It is renderer agnostic enabling the same API to draw in multiple contexts: svg, canvas, and webgl.

一个2D绘图函数库，可以使用完全一样的API实现在不同的环境(svg/canvas/webgl)下绘制图形

**使用方法：**

<div id="container"></div>

<script src="js/two.js"></script>

<script>

var two = new Two({}).appendTo(container);

two.makeCircle(.....);

two.makeRectangle(....);

//two.update();

//two.play();

</script>

4.HTML5新特性——地理定位

window.navigator.geolocation：获取当前浏览器所在的地理位置，如

经度longitude

纬度latitude

海拔altitude

速度speed

浏览器如何定位：

(1)手机浏览器：使用内置GPS模块或通过信号基站

(2)PC浏览器：使用IP地址反向解析出对应的地址

浏览器地理定位涉及到个人隐私，所以浏览器都会询问用户，是否给予某个网页读取的权限。

navigator.geolocation {

getCurrentPosition: fn, //一次性的获取定位信息

watchPosition: fn, //周期性的监视定位信息

clearWatch: fn //清除定位监视器

}

使用方法：

navigator.geolocation.getCurrentPosition(

function(pos){ //获取成功

console.log(pos.coords.longtude); 经度

console.log(pos.coords.latitude); 纬度

console.log(pos.coords.altitue); 海拔

console.log(pos.coords.speed); 速度

},

function(err){ //获取失败

console.log(err.code);

console.log(err.message);

}

);

5.扩展知识点：如何在网页中使用百度地图

(1)注册一个百度开发者账号

http://lbsyun.baidu.com/

(2)登录百度账号

(3)查看使用JS调用百度地图说明文档：

http://lbsyun.baidu.com/index.php?title=jspopular

(4)申请使用密钥



(5)创建HTML页面，调用百度地图API

查看使用手册——非常通俗易懂。 16:10



6.HTML5新特性——拖放API——今日重点

Drag & Drop：HTML5为拖放事件定义了7个事件

**被拖动的对象——源对象(source)可以触发的事件：**

(1)ondragstart 拖动开始

(2)ondrag 拖动中

(3)ondragend 拖动结束

**可以拖动着进入上方并松手的对象——目标对象(target)可以触发的事件：**

(1)ondragenter 拖动着进入上方

(2)ondragover 拖动着在上方悬停

(3)ondrop 松开

(4)ondragleave 拖动着离开

提示：ondragover事件后续的默认行为是ondragleave，即ondragover后默认必然触发ondragleave，必须阻止浏览器此事件默认行为！！

练习：

(1)创建一个可以随着拖动而移动的飞机图片

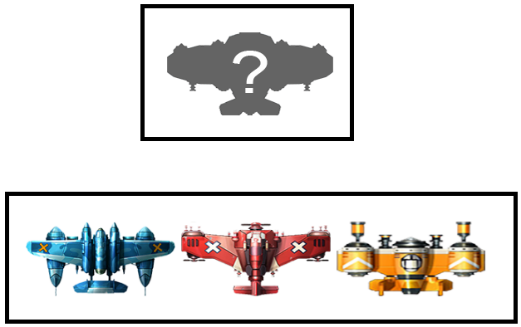
提示：IMG必须定位！拖动过程中读取事件发生的坐标，及时修改IMG的left和top。

(2)垃圾箱默认是半透明，拖动小飞机到垃圾箱上方时浏览器完全不透明；拖动着离开浏览器继续变为半透明；

若飞机在垃圾箱上方释放，则从DOM树上删除飞机；垃圾箱继续变为半透明

(3)把练习1 和 练习2 组合在一起

课后练习： 英雄选择



要求：

(1)拖动某架飞机到目标区域(随鼠标移动)，则在上方显示出该飞机；下方没有改飞机了；

(2)拖动另一架飞机到目标区域，则显示出该飞机，之前选中的飞机重回待选区域；

(3)从目标区域将某架飞机拖走，则该飞机重回待选区域；目标区域变回？飞机。

1.拖放API补充知识点

如何在源对象的事件和目标对象的事件间传递数据？

**方法1：创建一个全局变量——**污染全局对象

var 全局变量= null;

src.ondragstart = function(){

全局变量= 数据值;

}

target.ondrop = function(){

console.log(全局变量);

}

**方法2：使用H5拖放API提供的dataTransfer对象**

提示：源对象事件的dataTransfer与目标对象事件的dataTransfer不是同一个对象！但二者之间可以传递数据。

src.ondragstart = function(e){

//利用数据传递对象保存数据

e.dataTransfer.setData('key', 'value');

}

target.ondrop = function(e){

//读取数据传递对象中的得到的数据

var data = e.dataTransfer.getData('key');

}

|  |
| --- |
| 提示：H5标准中dataTransfer对象可以保存的数据必须有key，key只能叫'text/html'、'text/uri-list'、'Files'三者之一。但经过实测，该key名称可以为任意名称。 |

2.扩展知识点：如何拖放客户端本地的图片到网页中显示

一般来讲，网页中的图片都应该来自于服务器；一般情况下，网页中不能使用客户端的图片。

可以使用拖放的方式把客户端的图片在网页中显示出来：

客户端的图片文件是拖放事件的源对象；

网页中的元素是拖放事件的目标对象；

container.ondrop = function(e){

var f0 =e.dataTransfer.files[0]; //File对象

var fr = new FileReader();//读取文件中的内容

fr.readAsDataURL(f0); //把图片作为dataURL来读取

fr.onload = function(){ //文件读取完成

var img = new Image(); //创建IMAGE元素

img.src = fr.result; //dataURL

container.appendChild(img);

}

}

|  |
| --- |
| HTML5新增的文件读取相关的对象：  File 代表一个文件  FileList 代表一个文件列表  FileReader 用于读取一个文件中的内容  FileWriter 用于向一个文件中写出内容 |

3.面试题：请描述一下浏览器中的线程模型。

Program：程序，编写好的代码经过处理，可以在计算机上执行，放在文件系统(磁盘)上

Process：进程/任务，程序从磁盘中调入内存，分配必须可执行代码空间、数据空间，随时准备被CPU执行

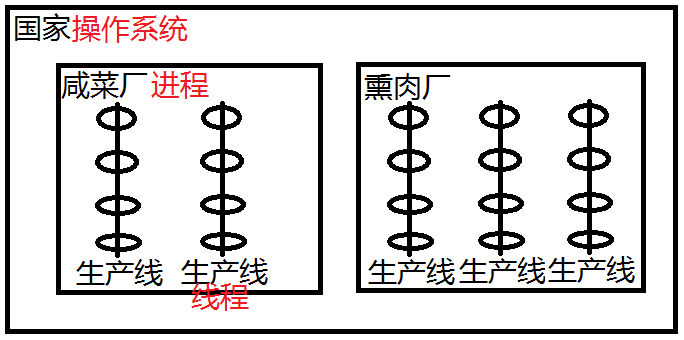
Thread：线程，是进程内部执行代码的基本单位

**进程和线程的关系：**

进程是操作系统分配内存基本单位；

线程是执行代码(JS/HTML/CSS)的基本单位；

线程处于进程内部，一个进程内部必须至少有一个线程，也可以有多个线程，这些线程间彼此可以没有影响——并发执行（宏观上看是同时执行，微观上看是在CPU上交替执行）



Chrome.exe进程中至少有6个线程，可以并发的向Web服务器发起请求（有的请求HTML、有的请求CSS、有的请求JS）。但是最终负责内容渲染(HTML解析/JS执行)的只有1个线程——UI主线程。

4.HTML5新特性——Web Worker

由于浏览器中负责渲染/监听只有一个UI主线程，所有的HTML/CSS/JS的执行都在这一个线程内，若页面中加载了非常耗时（算法复杂）的JS操作，会阻塞后续的HTML/CSS/JS的渲染和事件监听。

如何解决？—— 创建一个新的并发线程来执行这样的耗时操作。

JS中没有创建新线程的方法，替代方案——Web Worker

var w = new Work( 'xx.js' );

含义：在当前UI主线程中创建并启动一个新的并发的工作线程，该线程执行耗时操作，可能阻塞；但不会对当前UI主线程产生影响。

注意：Worker线程的致命问题——不能执行任何DOM操作，不能使用任何DOM&BOM元素——浏览器中只允许UI主线程修改DOM树。jQuery之类的JS文件决不能使用Worker来执行！！

**(1)如何让UI主线程给Worker线程传递数据：**

UI主线程：

var w = new Work('xx.js');

w.postMessage('data');

Worker线程：

onmessage = function(event){

var data = event.data;

}

**(2)如何让Worker线程把运算结果传递给UI主线程：**

Worker线程：

postMessage('data');

UI主线程：

var w = new Worker('xx.js');

w.onmessage = function(event){

var data = event.data;

}

练习：UI主线程中读取用户输入，发送给Worker线程；Worker线程开始运算，完毕后，把运算结果发送回给UI主线程，显示在DIV元素中。

**总结：Worker用于执行耗时的JS任务，在一个独立的线程中，可以避免UI主线程的阻塞问题。**

5.HTML5新特性——WebStorage——重点

项目中的数据如何保存？

**方式1：保存在服务器端——商品信息、用户信息、帖子**

**方式2：保存在客户端——浏览记录、登录信息、内容定制**

Web的客户端端存储技术有哪些？

(1)Cookie

优势：兼容性好

不足：操作繁琐，数据长度有限制（如4KB）

(2)Flash

优势：大小没有限制

不足：依赖于Flash允许环境

(3)Web Storage

优势：大小可达到8MB，操作简单

不足：HTML5新特性

(4)IndexedDB —— 课下自学

优势：大小没有限制，使用JS操作的一种客户端的数据库

不足：操作稍显复杂

HTML5中的Web Storage技术涉及两个新对象：

window.sessionStorage——会话级存储，其中的数据可以在一次会话中的多个页面中共享——数据存储在浏览器进程内存中。

sessionStorage.setItem(key, value)

var value = sessionStorage.getItem(key)

sessionStorage.removeItem(key)

sessionStorage.clear()

sessionStorage.key(i)

sessionStorage.length

window.localStorage——跨会话级存储/本地存储，其中的数据即使关闭浏览器/电脑，下次仍然可以访问——数据存储在文件系统的磁盘文件中。

localStorage.setItem(key, value)

var value = localStorage.getItem(key)

localStorage.removeItem(key)

localStorage.clear()

localStorage.key(i)

localStorage.length

注意：如果localStorage中的数据发生了改变，所有已打开的当前网站的浏览器窗口，都会自动触发window.onstorage事件，从而可以得到本地存储的数据已被修改。

|  |
| --- |
| Session：一个会话，指客户端连接到服务器后，在一段时间内的先后发起的多个请求，即一个会话中可能包含多个页面内容。只要浏览器不关闭，此次会话就一直存在；反之浏览器一关闭，会话就结束了。 |

练习：

(1)创建productlist.html页面，显示登录输入框，登录成功后可以点击“查看购物车”跳转到shoppingcart.html，此页面中显示出“欢迎回来：xxx”。还有一个超链接“退出登录”，删除登录的用户名。

(2)创建index.html页面，包含一个下拉列表，让用户选择自己喜欢网页风格，如：蔚蓝天空、芭比公主、杀马特等；接下来跳转到usercenter.html，该页面也呈现上一页面所选的风格；即使重启浏览器，再访问这些页面仍然是用户之前所选的风格。

提示：在客户端存储选中的主题的className

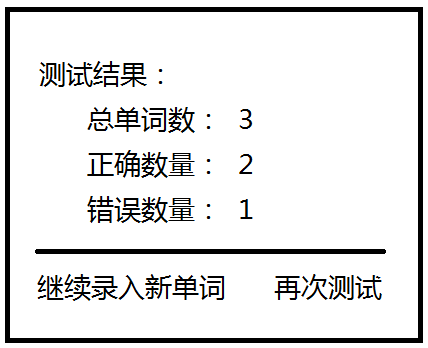
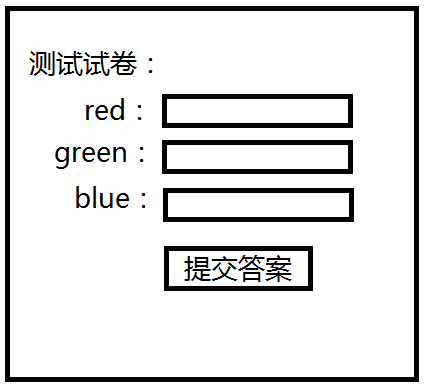
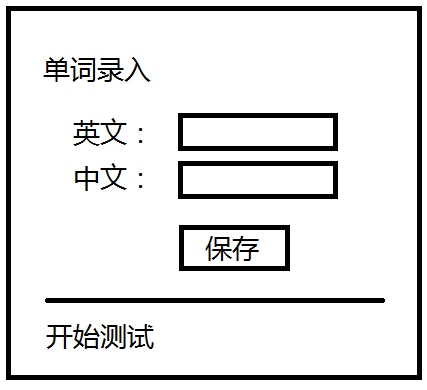
课后练习： 单词测试系统

(1)用户可以在save.html中不停的录入新单词；

(2)进入test.html开始测试，需要对之前录入过的所有单词进行测试；

(3)提交答案后，在result.html中显示出测试成绩。

提示：录入的单词需要永久保存；而此次测试结果只需要在当前会话中保存。



//遍历客户端存储的数据

for(var i=0; i<localStorage.length; i++){

var key = localStorage.key(i) //获取第i个key

}

课下任务：

(1)挑战性任务：不使用传统的表单，使用AJAX+拖放API实现文件上传。

1.HTML5新特性——WebSocket

**HTTP协议的不足**：基于“请求-响应”模型，只有客户端发了请求，服务器才会给响应，即没有请求就没有响应；一次请求也只能得到一次响应。在特殊应用场景有不足：

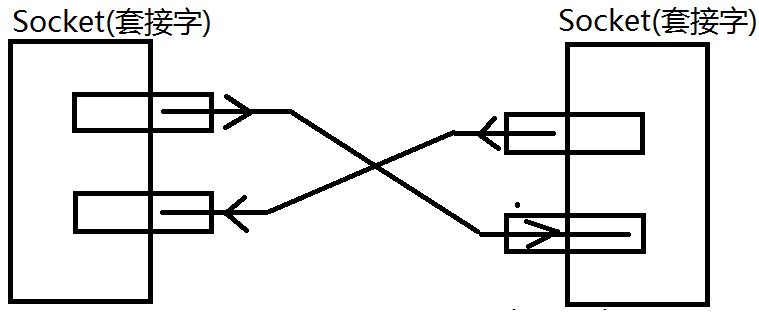


使用HTTP协议的解决方案： setInterval+AJAX =>心跳请求

注意：心跳过于频繁服务器压力太大；不频繁客户端获取消息有延迟。

最佳解决方案：——改用其它协议：

**WebSocket协议**：基于“广播-收听”模型，客户端连接到服务器上就不再断开，服务器有了消息可以随时发送给客户端，同时客户端也可以不停的给服务器发消息，服务器可以没有一次应答。即一方可以连续发多个消息，对方不停的接收。不足：客户端和服务器是永久连接——服务器端可以同时容纳的连接数有限制的。适用于“聊天室”、“实时走势图”等应用场景。



WebSocket应用需要服务器端和客户端都要使用WebSocket协议。传统的ApacheWeb不再适用。可以使用PHP或Node.js编写这样的服务器。

**运行PHP编写的Socket服务端程序：**

c:/xampp/php/php.exe e:/socket\_server.php

2.HTML5阶段项目

所用技术：

MySQLDB + PHP + AJAX + HTML5-Video + HTML5-Canvas + HTML5-SVG + HTML5-WebStorage

实现功能：

(1)用户注册

(2)确定下单

(3)用户中心- 我的订单

(4)用户中心- 消费统计

(5)用户中心- 幸运抽奖

实现步骤：

(1)修改SQL，为jd\_user表添加新的字段：

uid,uname,upwd,email,homepage,age,birthday(bigint)

(2)创建PHP，user\_add.php，接收客户端提交的注册信息，保存入数据库，返回{"msg": "succ", "uid": 3} 或 {"msg":"err", "sql":"INSERT...."}

(3)创建HTML，register.html（需要编写register.css），使用视频做背景，所有的输入域都要有输入验证，点击“注册”，使用AJAX异步提交注册信息，成功则跳转到产品列表页面（productlist.html）

--------------------------------------------

(4)修改SQL，添加jd\_order（订单信息表），包含字段：oid, orderNum, shopName, rcvName, price, payment(付款方式，INT), orderTime(下单时间,BIGINT), status(订单状态,INT), userId(下单用户的ID号)

(5)修改SQL，添加jd\_order\_detail（订单详情表），包含字段： did, orderId, productId, count(购买数量)

(6)创建PHP，order\_add.php，接收客户端提交的订单信息，添加到订单表和订单详情表，返回{"msg":"succ","oid",12, "orderNum": 1234567890 }或 {"msg":"err", "sql":"INSERT ...."}

SQL1: INSERT INTO jd\_order VALUES...获得oid

SQL2: INSERT INTO jd\_order\_detail VALUES... 循环执行

(7)创建HTML，addorder.html，根据当前购物车中的内容，生成订单，点击“下单购买”异步提交给服务器，进行订单生成，成功后在一个新的页面(addorder\_result.html)中显示出订单编号——sessionStorage

(8)创建HTML，usercenter.html，页头和页尾异步加载，主体内容分为左侧的“附加导航”和“右侧主体”