高等教育规划教材

免费提供
电子教案

Web 程序开发
案例教程

董祥和　编著

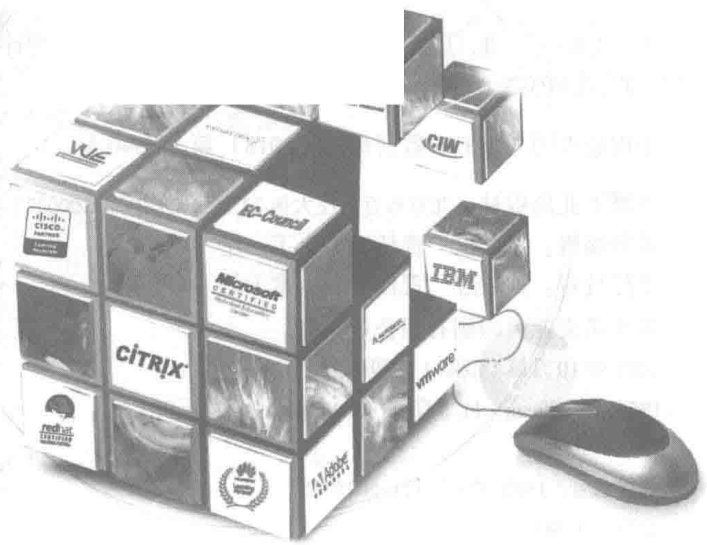
非外借

机械工业出版社
CHINA MACHINE PRESS

高等教育规划教材

Web 程序开发案例教程

董祥和　编著



机械工业出版社
CHINA MACHINE PRESS

本书以案例为驱动，侧重实践教学，主要讲解了 Web 程序开发中的关键技术以及常见问题的解决方法。全书分为 10 章，第 1 章为 Web 程序开发概述；第 2～6 章主要讲解 HTML5、CSS3、JavaScript 等前端技术常用相关知识，要求学生能运用相关知识动手实践完成网页布局并实现与用户的基本交互功能；第 7～9 章主要讲解 JSP 语法及内置对象、文件操作、数据库操作等相关知识，要求学生能动手设计动态网页和实现网站常用模块；第 10 章是综合案例，要求学生能设计实现信息发布系统。每章都会根据学生在实际开发中遇到的疑难问题给予相应的指导。通过本书的学习训练，能够夯实学生的理论基础，提高学生的动手编程能力和网站设计水平。

本书适合作为应用型本科院校和高等职业技术院校计算机、电子商务等相关专业学生学习 Web 程序开发课程的教材，也可作为 Web 程序开发人员的参考用书。

图书在版编目（CIP）数据

Web 程序开发案例教程/董祥和编著. —北京：机械工业出版社，2018.9
ISBN 978-7-111-60749-6

Ⅰ. ①W… Ⅱ. ①董… Ⅲ. ①网页制作工具-程序设计-高等学校-教材 Ⅳ. ①TP393.092

中国版本图书馆 CIP 数据核字（2018）第 195540 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策划编辑：丁 伦 责任编辑：丁 伦
责任校对：刘晓红 责任印制：常天培
北京圣夫亚美印刷有限公司印刷
2018 年 10 月第 1 版第 1 次印刷
185mm×260mm · 11.5 印张 · 281 千字
0001—3000 册
标准书号：ISBN 978-7-111-60749-6
定价：39.90 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
服务咨询热线：010-88379833
读者购书热线：010-88379649

网络服务
机 工 官 网：www.cmpbook.com
机 工 官 博：weibo.com/cmp1952
教育服务网：www.cmpedu.com
金 书 网：www.golden-book.com

封面无防伪标均为盗版

前　言

云计算、大数据、物联网等技术是目前IT行业发展的热点，这些技术大多将网站作为其应用接入方式。利用最新前端开发技术设计的响应式网站，可以根据设备的尺寸大小自动改变页面的布局，是集计算机、手机、平板电脑于一身的移动化网站。它不仅用于企业级站点，也用来做微站点、微商城等，还可以接入到微信、微博等应用程序。Java主要作为服务器端开发的语言，具有完美的开发平台，且有易适应和动态更新的能力。

本书侧重实践教学，结合实际案例，主要介绍了HTML5、CSS3、JavaScript等前端技术和JSP服务器端知识。全书贯穿“案例驱动”的思想，围绕Web程序开发时常用的技术进行组织安排，将讲解的理论应用到案例实现中。通过本书的学习，学生可以学会如何利用前端知识进行网页布局，学会如何利用Java技术实现用户和服务器之间信息的基本交互。

本书第1章为Web程序开发概述，分析Web程序开发常用技术；第2章介绍HTML5常用标签和相关案例；第3章介绍网页布局基础；第4章介绍CSS3属性和相关案例；第5章介绍JavaScript语法和相关案例；第6章介绍DOM模型和相关案例；第7章介绍JSP语法及内置对象和相关案例；第8章介绍JSP文件操作和相关案例；第9章介绍JSP数据库操作和相关案例；第10章综合前面章节内容设计了一个完整的信息发布系统。

本书可以作为应用型本科院校和高等职业技术院校计算机、电子商务等相关专业学生学习Web程序开发课程的教材，也可作为Web应用开发人员的参考用书。

感谢读者选择使用本书，由于作者水平有限，时间仓促，书中难免有疏漏和不当之处，敬请广大读者提出意见和建议，作者将不胜感激。

作　者

目　录

第1章 概　述

Web 应用程序主要是指执行浏览器端和服务器端网页技术的程序，用于构成为用户提供信息、让用户完成某些特定任务的网站。

1.1 Web 程序开发技术

Web 应用程序开发包含前端开发、服务器端开发等技术。

Web 前端包含 HTML5、CSS3、JavaScript、Ajax 等核心技术，以及在此基础上衍生的响应式布局框架 Bootstrap、JavaScript 库 jQuery、开源的 JavaScript 框架 Vue. js 等技术。

在 HTML4 基础上，HTML5 新增了用于绘画的 canvas 元素、用于播放视频音频的 video 和 audio 元素、新的表单控件（calendar、date、time、email、url、search）等特性。在 CSS 基础上，CSS3 新增了 2D 和 3D 转换功能、动画属性、弹性盒子布局等特性。Bootstrap 基于 HTML5 和 CSS3 开发，设计了丰富的 Web 组件方便用户使用，包括下拉菜单、导航栏、警告对话框、进度条等。jQuery 是一个轻量级的 JavaScript 函数库，它封装了 JavaScript 常用的功能代码，提供了一种简便的 JavaScript 设计模式，优化了 HTML 文档操作、事件处理、动画设计和 Ajax 交互。Vue. js 是一套构建用户界面的渐进式框架，它只关注视图层，通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

Web 服务器端技术主要用于实现获取数据、处理数据、存储数据、响应等功能，包含 JSP、Spring MVC + Spring + Hibernate 框架、缓存应用等相关技术。

Web 应用程序在台式计算机或笔记本式计算机上显示的网站网页效果如图 1-1 所示。

图 1-1　商城计算机端主页

通过前端技术可以设计响应式 Web 应用，即指 Web 页面的设计与开发根据设备环境（屏幕尺寸、屏幕定向、系统平台等）以及用户行为（改变窗口大小等）进行相应的响应和调整，这样开发的网站既可以适应在计算机端浏览器上访问，也可以适应在手机、平板电脑等移动端浏览器上访问。手机上网页效果如图 1-2 所示。

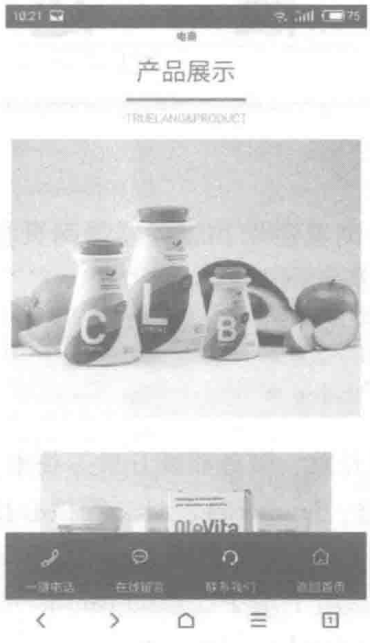


图 1-2　移动端企业产品页

1.2　Web App 的定义

App 是指运行在移动终端设备的第三方应用程序。在智能手机系统中有两种应用程序：一种是基于本地操作系统运行的 App；另一种是基于高端机的浏览器运行的 Web App。

Web App 是指主要在移动端浏览器上访问的网站。它是针对 Android、iPhone 优化后的 Web 站点，仍然是使用 Web 开发技术完成 Web App 的开发，即使用的技术还是 HTML5、CSS3、JavaScript 等前端技术和 Java、PHP 等服务器端技术。

Web App 和一般的 Web 应用程序一样，维护升级不需要通知用户，在服务器端更新文件即可，用户完全察觉不出来。Android、iPhone 手机内置浏览器都是基于 WebKit 内核的，所以在开发 Web App 时，多数都是使用 HTML5 和 CSS3 技术做用户界面布局，比如京东 Web App 搜索页效果如图 1-3 所示。

图 1-3　京东 Web App 搜索页

相比于 Web App，Native App 是一种基于智能手机本地操作系统（如 iOS、Android）并使用原生程序编写运行的第三方应用程序，也叫本地 App，使用的开发语言多为 Java、Objective-C 等。

Native App 支持在线或离线、消息推送或本地资源访问、摄像拨号功能的调取。由于设备碎片化，Native App 的开发成本高，维持多个版本的更新升级比较麻烦。

Hybrid App（混合模式移动应用）是指介于 Web-App、Native-App 这两者之间的 App，兼具“Native App 良好用户交互体验的优势”和“Web App 跨平台开发的优势”。

1.3 JSP 技术

JSP、PHP、ASP. NET 等是运行在服务器端的语言，本书主要探讨 JSP。JSP（Java Server Pages）是一种使软件开发者可以响应客户端请求，而动态生成 HTML、XML 或其他格式文档的 Web 页面的技术标准。JSP 技术以 Java 语言作为脚本语言，可以与处理业务逻辑的 Servlet 一起使用，是 Java EE 的一部分。

1.3.1 JSP 运行环境搭建

首先需要下载安装 Java SDK，然后设置 path 环境变量指明包含 javac 和 java 等命令所在的文件夹路径，以及设置 java_home 环境变量指明所安装的 Java SDK 所在路径。

Web 服务器 Tomcat 是一个开源软件，可作为独立的服务器运行 JSP 和 Servlet，也可以集成在 Apache Web Server 中。下载并解压 Tomcat 压缩文件，其中目录 bin 放置二进制执行文件，里面最常用的文件是 startup. bat，如果是 Linux 或 Mac 系统则启动文件为 startup. sh；目录 conf 里面最核心的文件是 server. xml，可以配置应用程序目录和修改端口号等；lib 存放库文件，连接 MySQL 或 SQL Server 数据库时的驱动程序包放在此处；webapps 下可以直接存放 web 应用程序，在浏览器中可以直接访问；work 目录下编译以后的 class 文件。

执行 bin 文件夹中的 startup. bat 可以启动 Tomcat 服务器，Tomcat 服务器使用 java_home 环境变量指定的 Java SDK。

1.3.2 网页运行原理

JSP 动态网页文件的扩展名为 . jsp，运行在服务器端。动态网页重要的特征之一是交互性，可以根据用户的选择执行不同的代码、显示不同的内容。

JSP 本质上就是把 Java 代码嵌入到 HTML 中，然后经过 JSP 容器编译执行，再根据这些动态代码的运行结果生成对应的 HTML 代码，从而可以在客户端的浏览器中正常显示。如果 JSP 页面是第一次被请求运行，则服务器的 JSP 编译器会生成 JSP 页面对应的 Java 代码，并且编译成类文件。当服务器再次收到 JSP 页面请求的时候，会判断该 JSP 页面是否被修改过，如果被修改过就会重新生成 Java 代码并且重新编译，而且服务器中的垃圾回收方法会把没用的类文件删除；如果没有修改过，服务器就会直接调用以前已经编译过的类文件。

JSP 页面在服务器中都会被编译成对应的 Servlet，JSP 程序无须改动就可以方便地迁移到其他操作系统平台，拥有 Java 跨平台的优势。Java 代码可以使用 JavaBean 封装，实现逻辑功能代码的重用，从而提高系统的可重用性和程序的开发效率。

一个简单的 JSP 页面 1. jsp 代码如下。

```
<%@ page contentType="text/html;charset=UTF-8" %>
<! DOCTYPE html >
<html>
<head>
<meta charset="utf-8">
<title>脚本识别</title>
<script language=javascript>
   alert("客户端 javascript 脚本!");
</script>
</head>
<body>
<p>这是一个简单的 JSP 页面
<%
  int i, sum=0;
  for(i=1;i<=100;i++)
  {
     sum=sum+i;
  }
%>
<p>1 到 100 的连续和是:<%=sum %>
</body>
</html>
```

执行结果如图 1-4 所示。

从上面代码可见，一个 JSP 页面除了普通的 HTMl 标记符外，再使用标记符号"<%"和"%>"加入 Java 程序片。用<script language=javascript>... </script>标记括起来的内容是 JavaScript 程序代码，称为客户端脚本。<%...%>包含的 Java 代码由服务器计算处理，由服务器处理解释的脚本称为服务器端脚本。通过单击查看浏览器菜单的源文件选项，可以看到浏览器收到的由服务器处理完后发送过来的 HTML 文件，不包含 Java 程序代码。

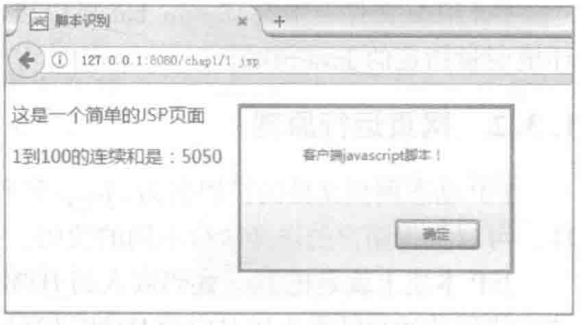


图 1-4　JSP 页面测试

1.4　网页运行测试案例

案例的目的是让学生掌握服务器的配置和简单网页文件的编写。

1.4.1　案例——识别客户端和服务器端

编写网页 testDate. jsp，测试服务器时间和浏览器时间，效果要求如图 1-5 所示。

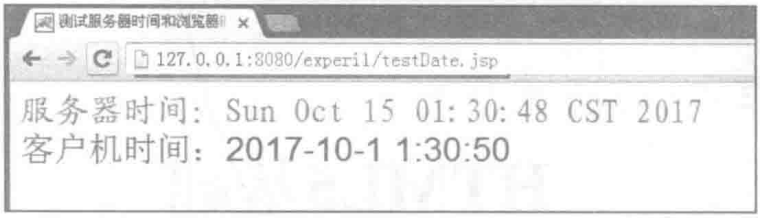


图 1-5 时间获取对比

1.4.2 案例——设置 Web 服务目录

在 C 盘下创建一个名字为 shop 的目录，并将该目录设置为一个 Web 应用程序目录，然后编写一个简单的 JSP 页面，保存到该目录中，让用户使用虚拟目录 company 访问该 JSP 页面。

1.5 网页运行测试案例分析

配置服务器时需要分析 server. xml 文件和设置 Web 服务目录。访问 Web 服务目录方法如下。

Tomcat 的根目录是指 webapps 下的 root，可以在 root 下直接放置网页，如 testDate. jsp。在访问该网页时，可以在浏览器网址中输入 Tomcat 服务器的 IP 地址（或域名）、端口号和 JSP 网页名字。调试网址为

```
http://127.0.0.1:8080/testDate.jsp
```

可以在 root 下创建目录 company，将 testDate. jsp 放在 company 下。此时访问网址为

```
http://127.0.0.1:8080/company/testDate.jsp
```

可以将 Tomcat 服务器所在计算机的某个目录设置成一个 Web 服务目录，并为该 Web 服务目录指定虚拟目录，需要修改 Tomcat 服务器安装目录下 conf 文件夹中的 server. xml 文件来设置新的 Web 服务目录。假如要将 C:\ shop 作为 Web 服务目录，并让用户使用 company 虚拟目录访问 C:\ shop 下的 JSP 页面，在 server. xml 文件的 </Host > 前面加入如下语句：

```
    <Context path="/company"  docBase="C:\shop" debug="0" reloadable="
true" />
```

配置文件 server. xml 修改后，必须重新启动 Tomcat 服务器。重启后就可以访问 C:\ shop 下的 JSP 页面，在浏览器地址栏访问：

```
http://127.0.0.1:8080/company/testDate.jsp
```

第2章 HTML5基础

HTML5 是 HTML 最新的修订版本，其设计目的是为了在移动设备上支持多媒体。<!DOCTYPE>声明必须位于 HTML5 文档中的第一行，即 <! DOCTYPE html>。中文网页需要使用 <meta charset="utf-8"> 声明编码，否则会出现乱码。有些 HTML4.01 的标签，如 <font> <strike> 等已经被一些 CSS 属性替代，而 <applet> <frame> 等均被 HTML5 新标签和 JavaScript 代码替代。

2.1 HTML5 图形绘制

HTML5 中的 <canvas> 标签结合 JavaScript 可以完成图形的绘制。<canvas> 标签是图形容器，使用脚本来绘制路径、盒子、圆、字符等图形。

一个画布在网页中是一个矩形框，通过 <canvas> 标签来绘制。<canvas> 标签默认没有边框和内容，需要使用 style 属性来添加边框。<canvas> 标签通常需要指定一个 id 属性(脚本中需要引用)，width 和 height 属性定义画布的大小。可以在 HTML 页面中使用多个 <canvas> 标签。示例代码如下：

```
<! DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>canvas 标签</title>
<style>
body{margin:0;padding:0}
canvas{margin:10px;padding:0px}
</style>
</head>
<body>
<canvas id="myCanvas" width="200" height="200" style="border:1px solid #900;"></canvas>
</body>
</html>
```

运行效果如图 2-1 所示。

图 2-1 <canvas>标签使用效果

JavaScript 在画布上的绘图需要首先创建画布，然后创建 context 对象，最后调用相关属性和方法完成绘图。示例代码如下：

```
<! DOCTYPE html >
<html >
<head >
<meta charset = "utf-8" >
<title > JavaScript 结合 canvas </title >
</head >
<body >
<canvas id = "myCanvas" width = "200" height = "200" style = "border:1px
solid #900;" >
</canvas >
<script >
var c = document. getElementById("myCanvas");
//找到 <canvas > 元素

var ctx = c. getContext("2d");
//创建 context 对象
//getContext("2d")是内建的 HTML5 对象,拥有多种绘制路径、矩形、圆形、字符以及添加
图像的方法

ctx. fillStyle = "#FF0000";
//设置 fillStyle 属性可以是 CSS 颜色、渐变或图案
//fillStyle 默认设置是#000000(黑色)

ctx. fillRect(0,0,150,75);
//fillRect(x,y,width,height)方法定义了矩形当前的填充方式
```

```
</script>
</body>
</html>
```

运行效果如图 2-2 所示。

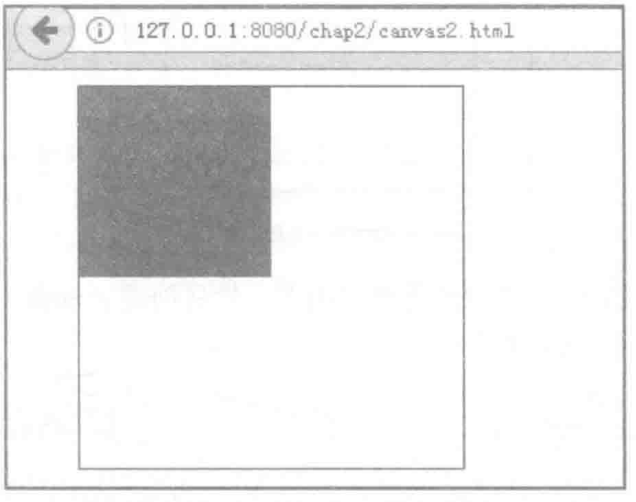


图 2-2　JavaScript 结合 <canvas> 标签

canvas 是一个二维网格，左上角坐标为（0，0）。fillRect（0，0，150，100）是指在画布上绘制 150 像素×100 像素的矩形，从左上角开始（0，0）。画布上的 X 和 Y 坐标用于在画布上对绘画进行定位，鼠标移动的矩形框上，显示定位坐标。

在 canvas 上绘制路径，需要利用 moveTo（x，y）和 lineTo（x，y）分别定义路径开始坐标和结束坐标，利用 stroke()方法绘制出通过 moveTo（x，y）和 lineTo（x，y）方法定义的路径，默认颜色是黑色，可以使用 strokeStyle 属性设置或返回用于笔触的颜色、渐变或模式。示例代码如下：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>路径</title>
</head>
<body>
<canvas id="myCanvas" width="200" height="200" style="border:1px
solid #d3d3d3;"></canvas>
<script>
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.moveTo(0,0);
```

```
    ctx.lineTo(200,200);
    ctx.strokeStyle="#900";
    ctx.stroke();
    </script>
    </body>
    </html>
```

运行效果如图2-3所示。

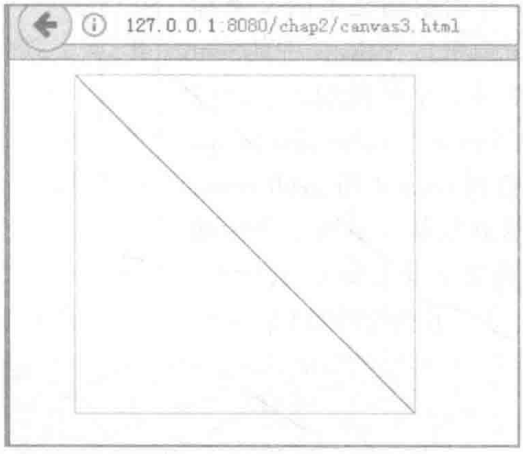


图2-3 在canvas上绘制路径

在canvas上绘制圆形使用方法arc(x,y,r,start,stop)，其中x是指圆心在x轴上的坐标，y是指圆心在y轴上的坐标，r是指半径长度，start是指起始角度（圆心平行的右端为0度），stop是指结束角度，Math.PI表示180°，画圆的方向是顺时针。示例代码如下：

```
    <!DOCTYPE html>
    <html>
    <head>
    <meta charset="utf-8">
    <title>arc</title>
    </head>
    <body>
    <canvas id="myCanvas" width="200" height="200" style="border:1px
solid #900;"></canvas>
    <script>
    var c=document.getElementById("myCanvas");
    var ctx=c.getContext("2d");
    ctx.beginPath();
    //beginPath()方法表示开始一条路径,或重置当前的路径。
    ctx.arc(100,100,100,0,2* Math.PI);
    ctx.stroke();
```

```
    </script>
    </body>
    </html>
```

运行效果如图 2-4 所示。

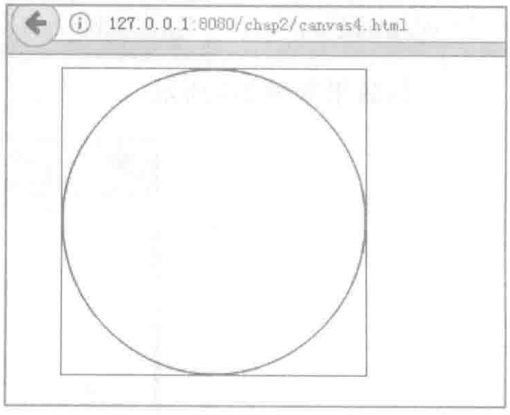


图 2-4　在 canvas 上绘制圆形

使用 fillText(text,x,y)方法可以在 canvas 上绘制实心文本，使用 strokeText(text,x,y)方法可以在 canvas 上绘制空心文本，其中 text 是指在画布上输出的文本，x 和 y 分别表示开始绘制文本的 x 坐标位置和 y 坐标位置（相对于画布）。canvas 中的 font 属性规定字体样式、字体变体、字体粗细、字号像素、字体系列，如 context.font = " italic small-caps bold 20px arial"。字体变体值有 normal 和 small-caps，其中 normal 是默认值，表明浏览器会显示一个标准的字体，而 small-caps 是指英文字母小型大写，大小跟小写字母一样，样式是大写。示例代码如下：

```
    <! DOCTYPE html>
    <html>
    <head>
    <meta charset = "utf-8">
    <title>绘制文本</title>
    </head>
    <body>
    <canvas id = "myCanvas" width = "400" height = "300" style = "border:1px
solid #000000;"></canvas>
    <script>
    var c = document.getElementById("myCanvas");
    var context = c.getContext("2d");
    context.fillText('Web 前端',100,40);
    //绘制文本

    context.font = '20px Arial';
    context.fontStyle = 'italic';
    //修改字体
    context.fillText('Web 移动端',100,100);

    context.font = '36px 隶书';
    context.strokeStyle = "#900"
    context.strokeText('Web 服务器端',100,200);
```