# 重庆邮电大学实验报告

2022-2023 学年 2 学期

课程名称 《移动互联网应用开发 1》实验

姓　　名

学　　号

年　　级

专　　业 通信工程

# 实验二 电子时钟的设计与实现

## 一、实验项目

实验内容：设计简单的电子时钟。

功能要求：设计一款简单的电子时钟，要求实现显示当前的时分秒，并且可以每秒更新一次实现动态效果。

## 二、实验源代码

**HTML** 代码:

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>电子时钟的设计与实现</title>
    <link rel="stylesheet" type="text/css" href="css2.1.css">
</head>
<body onload = "getCurrentTime()">  <!--执行脚本文件-->
 <h3>简单电子时钟的设计与实现</h3>
 <hr>
 <p>
        <div id="clock">
            <div class="box1" id="h"></div>
            <div class="box2">:</div>
            <div class="box1" id="m"></div>
            <div class="box2">:</div>
            <div class="box1" id="s"></div>
        </div>
    <script src="2.1.js"></script>
</body>
</html>
```

**CSS** 代码：

```
body{
    margin: 0px auto; /*指定元素所有外边距*/
    max-width: 1500px;
    border: soild; /* 边框样式 */
}

*{
    padding: 0;/* 定义元素边框与元素内容之间的空间*/
```

```
        box-sizing: border-box;
    }

    .wrap{
      width: 800px;
      margin: 0 auto;
    }


        .clear{clear: both;}


    #clock{
        width:800px;
        font-size: 80px;
        font-weight: bold;
        color: #afeeee;
        text-align: center;
        margin: 20px;
    }

    .box1{
        margin-right: 10px;
        width: 120px;
        height: 115px;
        line-height: 100px;
        float: left;
        border: rgb(255, 170, 170) 10px solid;
    }
    /* 冒号区域的样式设置 */

    .box2{
        width: 40px;
        float: left;
        margin-right: 15px;
    }
```

**JavaScript 代码：**

```
    var hour=document.getElementById('h');//获取带有指定 ID 名称：h 的元
          素
    var minute=document.getElementById('m');
    var second=document.getElementById('s');
    //获取当前时间
```

```
5.    function getCurrentTime(){
6.        var date=new Date();    //得到最新的时间
7.        var h=date.getHours();  //拿到小时数
8.        var m=date.getMinutes();     //拿到分钟数
9.        var s=date.getSeconds();     //拿到秒数
10.
11.       if(h < 10) h='0'+h;//确保 0-9 时也显示成两位数
12.       if(m < 10) m='0'+m;//确保 0-9 分钟也显示成两位数
13.       if(s < 10) s='0'+s;//确保 0-9 秒也显示成两位数
14.
15.       hour.innerHTML=h; //设置 innerHTML 指定的元素内容
16.       minute.innerHTML=m;
17.       second.innerHTML=s;
18.   }
19.   //每秒更新一次时间
20.   setInterval('getCurrentTime()',1000);
```

## 三、实验结果


图 1 简单电子时钟示意图

如图 1 所示，本次实验利用了简单利用 Div 元素进行划分，划分 5 个 Div 元素，其中两个设置冒号，剩余 3 个设置数字。


**图 2 简单电子时钟时间对比示意图**

如图 2 所示，本次实验通过 js 调用时间，实现了与北京时间的完全同步。

## 四、主要知识点

- 整体设计：
  - 使用 标签划分区域
  - 使用外部 CSS 样式表 clock.css
- 时分秒显示框设计：
  - 采用 class 分割冒号与数字
  - 采用 id 分割小时、分钟、秒
- 时钟动态效果核心技术：
  - 利用定时器 setInterval()对 getCurrentTime()函数进行处理，设置 1000 毫秒进行一次函数
  - 采用 onload = "getCurrentTime()”实现当页面载入完毕后执行函数

## 五、心得体会（含意见建议）

在完成利用 HTML 完成电子时钟的设计与实现这一实验后，我对 Web 开发技术有了更深刻的理解和认识，同时也收获了一些经验和体会。这一实验让我更深入地了解了 HTML 标记语言的特点和使用方法，尤其是表格、样式、脚本等相关标记的应用。同时，通过查找相关资料和代码示例，我也学习到了一些实用的技巧和方法，例如利用 JavaScript 语言实现时钟的效果、使用 CSS 样式美化页面等。

意见建议：增加实验指导：虽然该实验提供了一份简单的实验指导，但仍然存在一些不够清晰和详细的地方，建议可以增加一些更具体的实验指导，例如给出代码示例、提供相关资料等。这样可以帮助学生更好地理解和掌握实验内容。

# 实验三 图片相框展示的设计与实现

## 一、实验项目

实验内容：目前市面上一些修图工具软件带有自动为图片添加不同款式的相框功能，用户可以选择本地图片文件然后为其添加相框效果。

功能要求：使用 HTML5 拖放 API 相关技术，在网页上实现为指定图片自动生成图片相框的效果。用户通过拖拽可以将本地的图片文件放置到页面上指定区域，即可在页面上自动生成带有相框效果的图片展示。

## 二、实验源代码

**HTML** 代码:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>HTML5 拖放 API 之图片相框效果</title>
  <link rel="stylesheet" href="css/photoframe.css">
    </head>
    <body>
        <h3>HTML5 拖放 API 之图片相框效果</h3>
        <hr />
 <!--可放置文件区-->
        <div id="recycle" ondragover="allowDrop(event)" ondrop="f
      ileDrop(event)">
            请将图片拖放至此处
        </div>
        <br />
 <!--带有相框的图片展示区-->
        <div id="output"></div>
        <script>
            //ondragover 事件回调函数
            function allowDrop(ev) {
                //解禁当前元素为可放置被拖拽元素的区域
                ev.preventDefault();
            }
  //ondrop 事件回调函数
            function fileDrop(e) {
  //解禁当前元素为可放置被拖拽元素的区域
                e.preventDefault();

  //获取图片展示区域对象 output
                var output = document.getElementById("output");
```

```
31.       //将图片展示区域的内容清空
32.                   output.innerHTML = "";
33.
34.                   //获取从本地拖拽放置的文件对象数组 files
35.       var files = e.dataTransfer.files;
36.
37.       //使用 for 循环遍历同时被拖拽并放置到指定区域的所有文件
38.                   for (var i = 0,f; f = files[i]; i++) {
39.        //(1)创建带有相框的图片
40.        //获取当前图片文件的 URL 来源
41.                       var imgURL = window.webkitURL.createObjectURL
           (f);
42.                       //创建图片对象 img
43.        var img = document.createElement("img");
44.        //设置图片对象 img 的 src 属性为当前图片文件 URL 地址
45.                       img.setAttribute("src", imgURL);
46.        //设置图片对象 img 的宽度为 330 像素
47.                       img.setAttribute("width", "330");
48.        //设置图片对象 img 的高度为 270 像素
49.                       img.setAttribute("height", "270");
50.
51.                       //设置相框对象 photo
52.        var photo = document.createElement("div");
53.        //为相框对象添加 class="photoframe",以加载相框背景图片
54.                       photo.setAttribute("class", "photoframe");
55.        //将图片添加相框对象中
56.                       photo.appendChild(img);
57.
58.        //创建图片对象 img2
59.        var img2 = document.createElement("img");
60.        //设置图片对象 img2 的 class="block"
61.        img2.setAttribute("class", "block");
62.        //将图片 2 也添加到相框元素中
63.                       photo.appendChild(img2);
64.
65.        //添加相框和相片效果
66.        output.appendChild(photo);
67.
68.                       //(2)创建图片下方的状态信息栏
69.        //使用 div 元素创建状态信息栏 status
70.        var status = document.createElement("div");
71.        //获取当前文件的最新修改日期
72.                       var lastModified = f.lastModifiedDate;
73.
```

```
74.
75.         //修改当前文件的最新修改日期的描述格式
76.                         var lastModifiedStr = lastModified ? lastModi
            fied.toLocaleDateString() + ' ' + lastModified.toLocaleTime
            String() : 'n/a';
77.         //设置图片下方状态信息栏描述内容
78.         status.innerHTML = '<strong>' + f.name + '</strong> (' + (f.
            type || 'n/a') + ')<br>大小: ' + f.size + '字节<br>修改时间:
            ' + lastModifiedStr;
79.
80.                         //添加文件描述
81.                         output.appendChild(status);
82.                     }
83.                 }
84.             </script>
85.         </body>
86.     </html>
```

## 三、实验结果

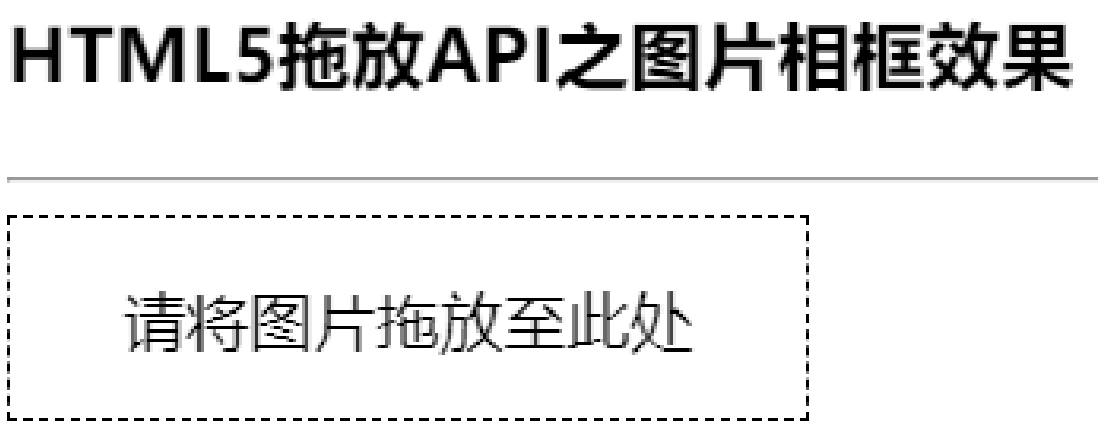


**图 3 未拖拽图片示意图**

如图 3 所示，本次实验即将外部的图片拖拽到途中的方框内，即可实现图片的居中相框效果实现。

如图 4 所示，本次实验即将外部的图片拖拽到途中的方框内，即可实现图片的居中相框效果实现，此时在此拖动另一张图片，即可实现图片替换。


**图 4 拖拽单张图片示意图**

如图 5 所示，本次实验即将多张外部的图片拖拽到途中的方框内，即可实现多张图片的居中相框效果实现，此时图片呈上至下依次排列完成。

**HTML5拖放API之图片相框效果**

请将图片拖放至此处

**回收站.jpg** (image/jpeg)
大小：11742字节
修改时间：2023/3/26 22:06:19

**star.jpg** (image/jpeg)
大小：1785字节

图 5 拖拽多张图片示意图

## 四、主要知识点

- 界面设计：
  - 使用 标签划分区域
    - 本地文件放置区域
    - 带有相框图片的展示区域
  - CSS 文件辅助渲染样式
- 相框自动生成功能的实现：
  - 采用 ondragover 事件回调函数，解禁当前元素为可放置被拖拽元素的区域
  - 采用 ondrop 生成带有相框的图片效果，其中使相框居中采用了一种以块元素向下挤动图片实现居中
  - 显示图片文件信息：通过图片文件利用 js 的函数获取当前图片的最新信息

## 五、心得体会（含意见建议）

在进行“图片相框展示的设计与实现”的 html 实验时，我深刻地体会到了实践的重要性。通过实验，我深入了解了 html 标签的使用方法，包括如何嵌套标签、如何设置样式等等。在实验过程中，我也遇到了很多问题，但通过查找相关资料、向同学请教等途径，我最终解决了这些问题，也更加深刻地理解了 html 的知识点。通过实验，我还学会了如何将所学知识应用到实际项目中，这对我今后的学习和工作都具有很大的帮助。

意见建议：可以增加一些案例分析，让学生们能够更加清晰地了解 html 在实际项目中的应用。同时，也可以增加一些互动环节，比如学生们可以自己设计一个网页，并分享给其他同学进行评价和交流，这样能够更加激发学生们的学习兴趣。

# 实验四 问卷调查页面的设计与实现

## 一、实验项目

实验内容：随着移动终端的普及，手机移动支付业务成为了现在市场支付手段的发展趋势。以手机移动支付业务为例，使用 HTML5 表单技术实现相关业务的问卷调查页面。

功能要求：

1. 用户可以根据调查问题进行单选、多选以及在结尾处填写姓名、职位和联系电话等信息。
2. 每个输入栏目的文本框均需要显示提示信息。
3. 用户在点击按钮提交注册信息时可以验证所有栏目均为必填项以及电子邮箱的有效性。

## 二、实验源代码

**HTML** 代码:

```
<!doctype html>
<html>

<head>
 <meta charset="utf-8">
 <title>手机移动支付调查问卷</title>
 <style>
  body {
   background: #ffffff;
  }

  div {
   background: #FFF;
   color: rgb(5, 5, 5);

   padding: 15px;
   margin: 60px auto 0px;
   width: 600px;
   font-family: "微软雅黑";
   box-shadow: 10px 10px 15px black;
  }

  h2 {
   text-align: center;
   color: #09F;
  }

  hr {
   width: 80%;
   border: 1.05px solid #09F;
   margin-top: -5px;
  }

  input {
   margin: 10px
  }

  li {
   margin: 10px;
  }
```

```
41.
42.     input[type="text"],
43.     input[type="tel"] {
44.      width: 100px;
45.      border: 0px;
46.      border-bottom: 1px solid;
47.     }
48.
49.     button {
50.      background: #09F;
51.      display: block;
52.      margin: 0 auto;
53.      color: white;
54.      width: 100px;
55.      height: 40px;
56.      font-size: 16px;
57.      font-weight: bold;
58.      font-family: "微软雅黑";
59.      margin: 5% 39%;
60.     }
61.
62.     button:hover {
63.      background: #0CF;
64.     }
65.    </style>
66.   </head>
67.
68.   <body>
69.    <div>
70.     <h2>手机移动支付业务问卷调查</h2><br />
71.     <hr /><br /><!-- 水平线标签 -->
72.     <!--get：提交的数据量要小于1024字节，表单提交时表单域数值（表单请求
          的信息：账号、密码…）将在地址栏显示。
73.     post:传递的数据量不受限制，表单提交时表单的域值（表单请求的信息：账
          号、密码…）不会在地址栏显示，安全性能较高，对信息进行了隐藏，一般
          在开发中采用post。-->
74.     <!--若onsubmit事件返回false，则将阻止表单的提交。如果不返回值，则
          默认为true-->
75.     <form method="post" action="目前没有服务器，随便编一个
          " onSubmit="return check()">
76.      <ol><!-- 有序表 -->
77.       <li>您的教育程度是？</li>
78.       <!-- 单选type -->
79.       <!-- required 非空属性 -->
```

```
80.       <label> <input type="radio" name="1" required>高中 </label>
81.       <label> <input type="radio" name="1" required>大专 </label>
82.       <label> <input type="radio" name="1" required>本科 </label>
83.       <label> <input type="radio" name="1" required>硕士研究
            生 </label>
84.       <label> <input type="radio" name="1" required>博士及以
            上 </label>
85.
86.       <li>您的年龄段是？</li>
87.       <label> <input type="radio" name="2" required>18 岁以
            下 </label>
88.       <label> <input type="radio" name="2" required>18-25
            岁 </label>
89.       <label> <input type="radio" name="2" required>26-30
            岁 </label>
90.       <label> <input type="radio" name="2" required>31-35
            岁 </label>
91.       <label> <input type="radio" name="2" required>35 岁以
            上 </label>
92.       <li>您了解以下哪些手机移动支付业务？（可多选）</li>
93.       <label> <input type="checkbox" name="q3">支付宝 </label>
94.       <label> <input type="checkbox" name="q3">微信支付 </label>
95.       <label> <input type="checkbox" name="q3">Apple Pay </label>
96.       <label> <input type="checkbox" name="q3">电信翼支付 </label>
97.       <label> <input type="checkbox" name="q3">以上均不了解 </label>
98.       <li>您看重以下哪些支付功能？（可多选）</li>
99.       <label> <input type="checkbox" name="q4">话费充
            值 </label><br />
100.      <label> <input type="checkbox" name="q4">刷手机加
            油 </label><br />
101.      <label> <input type="checkbox" name="q4">刷手机购
            物 </label><br />
102.      <label> <input type="checkbox" name="q4">乘坐公交/地
            铁 </label><br />
103.      <label> <input type="checkbox" name="q4">以上均不感兴
            趣 </label><br />
104.    </ol>
105.    <label>您的姓名<input type="text" name="name" required></label>
106.    <label>您的职业
            <input type="text" name="position" required></label>
107.    <label>联系电话<input type="tel" name="tel" required></label>
108.    <button type="submit">提交问卷</button>
109.
110.  </form>
```

```
111.    <script>
112.     function checkBox(name) {
113.      var j = 0;
114.      var flag = 0;
115.      var checkbox = document.getElementsByName(name);
116.      for (var i = 0; i < checkbox.length; i++) {
117.       if (checkbox[i].checked) {
118.        if (i == checkbox.length - 1) {
119.         flag++;
120.        }
121.        j++;
122.
123.       }
124.      }
125.      if (j == 0 || ((j > flag) && flag)) { return false; }
126.      return true;
127.     }
128.     function check() {
129.      var q3 = checkBox("q3");
130.      if (q3 == false) {
131.       alert("第 3 题至少要选择一个选项");
132.       return false;
133.      }
134.      var q4 = checkBox("q4");
135.      if (q4 == false) {
136.       alert("第 4 题至少要选择一个选项且 5 选项不能与其他选项兼容");
137.       return false;
138.      }
139.     }
140.
141.    </script>
142.   </div>
143. </body>
144.
145. </html>
```

## 三、实验结果

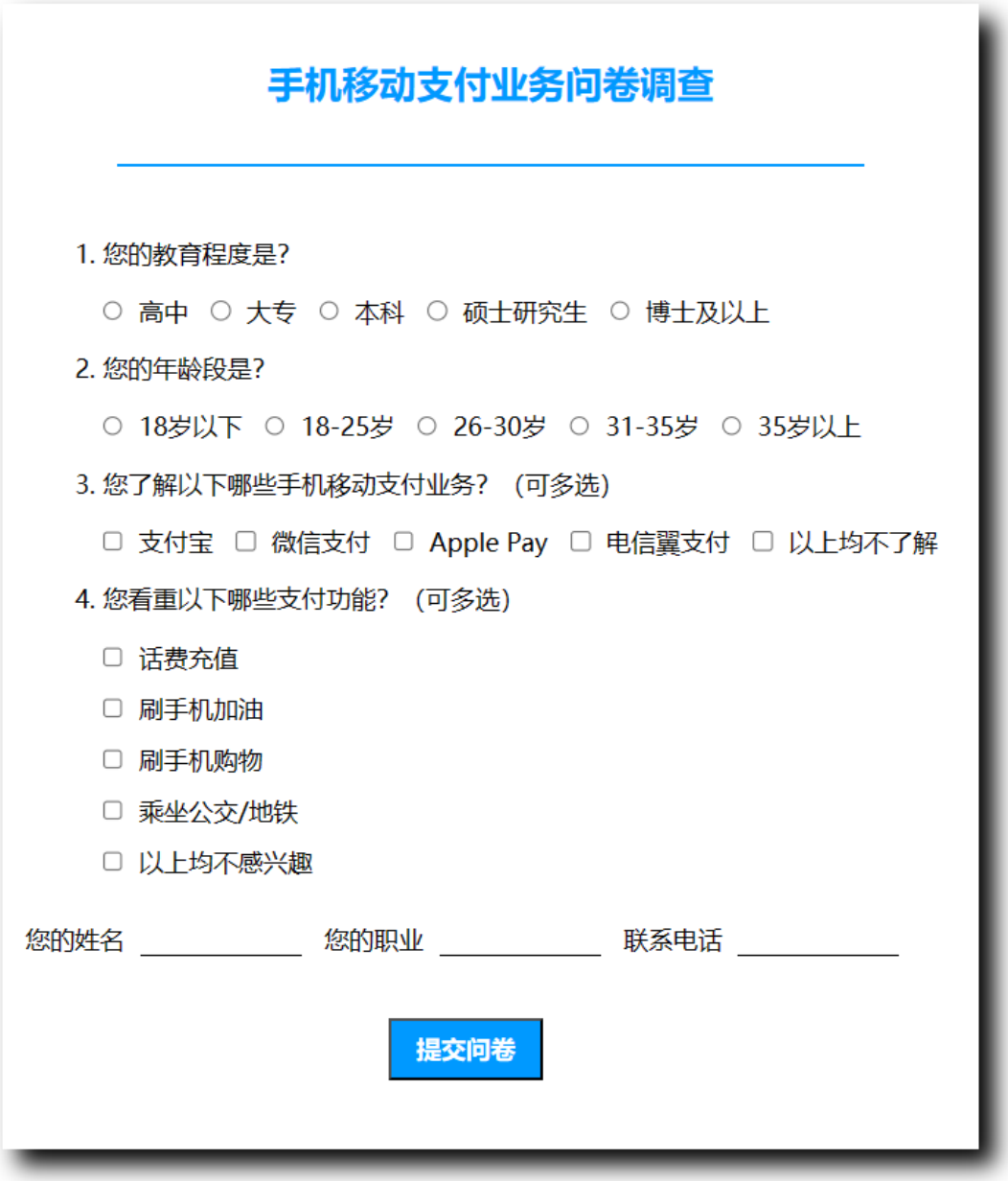


图 6 手机移动支付业务问卷调查示意图

如图 6 所示，本次实验利用 ol 有序表，划分为 4 个问题，利用 input 进行调查问卷问题类型的修改，最终利用 css 代码完成最终样式的修改使最终整洁美观。


**图 7 异常情况示意图**

如图 7 所示，本次实验若不填写或者不符合要求则会产生提示问卷提交必须填写该字段。

## 四、主要知识点

- 整体设计：
  - 使用 划分区域；
  - 使用 CSS 外部样式表定义样式；
- 表单设计：
  - 使用<form>形成表单区域
  - 使用有序列表标签<ol>设计问题样式
  - 使用<input>标签设计问题选项 type=radio：单选题 type=checkbox：多选题
  - 使用<input>标签设计个人信息填写栏目 type=text：姓名、职位 type=tel：联系电话
  - 使用<button>标签设计问卷提交按钮 type=submit
- 验证功能的实现：
  - 使用<input>标签的 required 属性实现单选框的非空验证
  - 使用<input>标签的 required 属性实现个人信息栏的非空验证
  - 使用 JavaScript 自定义函数实现多选框的非空验证 监听提交表单的动作

## 五、心得体会（含意见建议）

对 HTML 和 CSS 的掌握情况：通过实验，我们可以进一步掌握 HTML 和 CSS 的知识，了解如何使用它们来设计和实现网页。

对表单元素的熟悉程度：在实现问卷调查页面时，我们需要使用 HTML 表单元素，因此需要对表单元素有一定的熟悉程度，包括如何设置、如何获取用户输入等。同时通过这次实验让我发现自己的漏洞，明白了<label>标签的作用即可以电子文字选择内容。

意见建议：强化交互性：课程应该更加注重交互性，增加学生之间的互动和交流，提高学生的学习兴趣和积极性。

# 实验十 基于 HTML5 的贪吃蛇游戏的设计与实现

## 一、实验项目

实验内容：综合实验设计 html 贪吃蛇游戏。

功能要求：贪吃蛇游戏是一款经典的单机休闲游戏，玩家通过上下左右按键控制蛇头的移动方向使其向指定方向前进，并吃掉随机位置上产生食物来获得分数。每吃掉一次食物，贪吃蛇的蛇身都会变长，并且会继续在随机位置上产生下一个食物。如果蛇头撞到墙壁或蛇身，则判定游戏失败。

## 二、实验源代码

**HTML** 代码:

```
<!DOCTYPE html>
<html>

<head>
    <!--标题-->
    <title>Snakey</title>
    <h3>基于 HTML5 的贪吃蛇小游戏</h3>
    <style>
        body {
            text-align: center;
            margin: 100px;
            background-color: #ffffff"

        }

        #container {
            text-align: center;
            width: 600px;
            margin: auto;
            padding: 10px;
            background-color: white;
            box-shadow: 10px 10px 15px gray;
        }

        /* 状态栏样式 */
        #status {
            padding: 10px;
```

```
            width: 400px;
            height: 20px;
            margin: auto;
        }

        .box {
            float: left;
            width: 200px;
        }

        button {
            width: 200px;
            height: 50px;
            margin: 10px 0;
            border: 0;
            outline: none;
            font-size: 25px;
            font-weight: bold;
            color: white;
            background-color: lightcoral;
        }

        button:hover {
            background-color: coral;
        }
    </style>
    <hr>
    <!--状态信息栏-->
    <div id="status">
        <!--历史最高分-->
        <div class="box">
            历史最高分：<span id="bestScore">0</span>
        </div>
        <!--当前分数-->
        <div class="box">
            当前分数：<span id="currentScore">0</span>
        </div>
    </div>
</head>
<!--设置 body 中元素居中，外边距 100 像素，背景颜色为#aaaaaa-->

<body>
    <!--设置 id，宽 400 像素，高 400 像素，背景颜色设置为灰色-->
```

```
71.     <canvas id="canv" width="400" height="400" style="background-
          color:gray">
72.     </canvas>
73.     <script>
74.         //声明变量
75.         //设置 canvas 画布的绘图的环境，当前唯一支持的参数是 2d
76.         var box = document.getElementById('canv').getContext('2d'
          );
77.         //声明一个变量表示蛇
78.         var snake;
79.         //声明键盘事件的变量。1 表示向右，-1 向左，20 向下，-20 向上
80.         var direction;
81.         //下次移动相关
82.         var n;
83.         //声明食物变量
84.         var food;
85.         //声明标志变量
86.         var flag;
87.         //当前得分
88.         var score;
89.         //历史最高分纪录
90.         var bestScore;
91.
92.         function showBestScore() {
93.             //从本地存储数据中读取历史最高分
94.             bestScore = localStorage.getItem("bestScore");
95.             //如果尚未记录最高分，则重置为0
96.             if (bestScore == null)
97.                 bestScore = 0;
98.             //将历史最高分更新到状态栏中
99.             var best = document.getElementById("bestScore");
100.            best.innerHTML = bestScore;
101.        }
102.
103.        //小蛇初始化函数
104.        function initialization_snake() {
105.            snake = [];
106.            flag = Math.floor(Math.random() * 4);
107.            if (flag == 0) {
108.                snake[0] = Math.floor((Math.random() * 400));
109.                snake[1] = snake[0] - 1;
110.                direction = 1;
111.                n = snake[0] + direction;
112.                detectCollision()
```

```
113.             }
114.             else if (flag == 1) {
115.                 snake[0] = Math.floor((Math.random() * 400));
116.                 snake[1] = snake[0] + 1;
117.                 direction = -1;
118.                 n = snake[0] + direction;
119.                 detectCollision()
120.             }
121.             else if (flag == 2) {
122.                 snake[0] = Math.floor((Math.random() * 400));
123.                 snake[1] = snake[0] - 20;
124.                 direction = 20;
125.                 n = snake[0] + direction;
126.                 detectCollision()
127.             }
128.             else if (flag == 3) {
129.                 snake[0] = Math.floor((Math.random() * 400));
130.                 snake[1] = snake[0] + 20;
131.                 direction = -20;
132.                 n = snake[0] + direction;
133.                 detectCollision()
134.             }
135.         }
136.
137.         function initialization_food() {
138.             while (snake.indexOf(food = ~~(Math.random() * 400))
        > 0);
139.         }
140.
141.         //碰撞检测函数
142.         function detectCollision() {
143.             while (n < 0 || n > 399 || (direction == -
        1 && n % 20 == 18) || (direction == 1 && n % 20 == 1) || (d
        irection == -
        20 && n + 40 > 399) || (direction == 20 && n - 40 < 0) || (
        direction == -
        1 && n % 20 == 19) || (direction == 1 && n % 20 == 0)) {
144.                 snake[0] = Math.floor((Math.random() * 400));
145.                 snake[1] = snake[0] - direction;
146.                 n = snake[0] + direction;
147.             }
148.         }
149.
150.         //绘制地图
```

```
151.          function draw(point, color) {
152.              //fillStyle()方法为设置用于填充绘画的颜色
153.              box.fillStyle = color;
154.              //fillRect()方法用于绘制“被填充”的矩形
155.              box.fillRect(point % 20 * 20 + 1, ~~(point / 20) * 20
          + 1, 18, 18);
156.          }
157.
158.          // 获得历史最高分记录
159.          showBestScore();
160.          //自动运行函数
161.          (function () { ready(); }());
162.
163.          //各种变量初始化，背景初始化
164.          function ready() {
165.              //循环绘制地图块
166.              clearInterval()
167.              score = 0;
168.              for (var i = 0; i < 400; i++) {
169.                  draw(i, "#313131");
170.              }
171.              //对蛇和食物进行静态赋值并绘制到地图上
172.              initialization_snake()
173.              initialization_food()
174.              draw(food, "yellow");
175.              draw(snake[0], "#00b7ee");
176.              draw(snake[1], "#00b7ee");
177.              var currentScore = document.getElementById("currentSc
          ore");
178.              currentScore.innerHTML = score;
179.          }
180.
181.          //核心算法
182.          function run() {
183.              //在游戏开始以后“开始游戏”按钮变为不可点击状态
184.              document.getElementById("butn").setAttribute("disable
          d", true);
185.              //用unshift()方法向数组的开头添加一个新元素，新元素为蛇下
          一步所移动的坐标
186.              snake.unshift(n = snake[0] + direction);
187.              //边界判断，如果蛇头碰到上下左右四个边或者碰到自己的身子
188.              //游戏结束，初始化游戏，将“开始游戏”按钮设置为可以点击，点
          击开始按钮可以重新进行游戏
```

```
189.             //条件：snake.indexOf(n,1)>0 判断撞蛇尾若首次出现泽返回-
         1     n<0||n>399 超出边界      (direction==-
         1&&n%20==19)||(direction==1&&n%20==0)碰壁
190.             if (snake.indexOf(n, 1) > 0 || n < 0 || n > 399 || (d
         irection == -
         1 && n % 20 == 19) || (direction == 1 && n % 20 == 0)) {
191.                 showBestScore();
192.                 ready();
193.                 document.getElementById("butn").removeAttribute("
         disabled");
194.                 return alert("游戏结束");
195.                 window.location.reload();
196.             }
197.             if (bestScore < score) {
198.                 localStorage.setItem("bestScore", score);
199.                 var best = document.getElementById("bestScore");
200.                 best.innerHTML = score;
201.             }
202.             //如果蛇头没有碰到边界或蛇身，在地图上绘制蛇头
203.             draw(n, "#00b7ee");
204.             //如果蛇头吃到了食物（坐标相同）
205.             if (n == food) {
206.                 //随机在地图上产生一个新食物，新事物不能于蛇身子重合
207.                 while (snake.indexOf(food = ~~(Math.random() * 40
         0)) > 0);
208.                 //绘制食物
209.                 draw(food, "yellow");
210.                 score += 10;
211.                 //更新状态栏中的当前分数
212.                 var currentScore = document.getElementById("curre
         ntScore");
213.                 currentScore.innerHTML = score;
214.             }
215.             //如果蛇头没有吃到食物
216.             else {
217.                 //将蛇尾元素删除并且根据删除的坐标绘制为地图块的颜色
218.                 draw(snake.pop(), "#313131");
219.             }
220.             //每 200 毫秒重复执行一次该函数
221.             setTimeout(arguments.callee, 200);
222.         }
223.
224.         //添加键盘事件
225.         document.onkeydown = function (e) {
```

```
226.                 //如果蛇当前横着走，当键盘输入 up 时，将蛇头方向改为向上，
          输入 down 时将蛇头方向改为向下
227.                 if ((direction == 1 || direction == -
          1) && (snake[0] - snake[1] == 1 || snake[0] - snake[1] == -
          1)) {
228.                     if (e.keyCode == 38) {
229.                         direction = -20;
230.                     }
231.                     if (e.keyCode == 40) {
232.                         direction = 20;
233.                     }
234.                 }
235.                 //如果蛇当前竖着走，当键盘输入 left 时向左，right 时向右
236.                 if ((direction == 20 || direction == -
          20) && (snake[0] - snake[1] == 20 || snake[0] - snake[1] ==
           -20)) {
237.                     if (e.keyCode == 39) {
238.                         direction = 1;
239.                     }
240.                     if (e.keyCode == 37) {
241.                         direction = -1;
242.                     }
243.                 }
244.             }
245.     </script>
246.     <div>
247.         <!--设置游戏开始按钮-->
248.         <button id="butn" type="button" onclick="run()">开始游戏
          </button>
249.     </div>
250. </body>
251.
252. </html>
```

## 三、实验结果

针对于本次贪吃蛇项目，如下所示为整个项目以及逻辑的伪代码。

**Variable Definitions:** *currentScore*: the current score *highScore*: the highest score *snake*: the snake instance *food*: the food instance *gameOver*: a flag indicating whether the game is over
**Initialization:** *currentScore* ← 0 *highScore* ← retrieve the highest score from local storage *snake* ← create an initial snake instance *food* ← create an initial food instance *gameOver* ← false
**Game Loop: while** *not gameOver* **do**
  retrieve user input update the snake's position **if** *the snake and the food overlap* **then**
    increase the score update the current score display generate a new food instance update the high score
  **end**
  **if** *the snake's head hits a boundary or its own body* **then**
    *gameOver* ← true if the current score is higher than the high score, store the current score as the new high score display the game over screen
  **end**
**end**

**Algorithm 3:** Snake Game

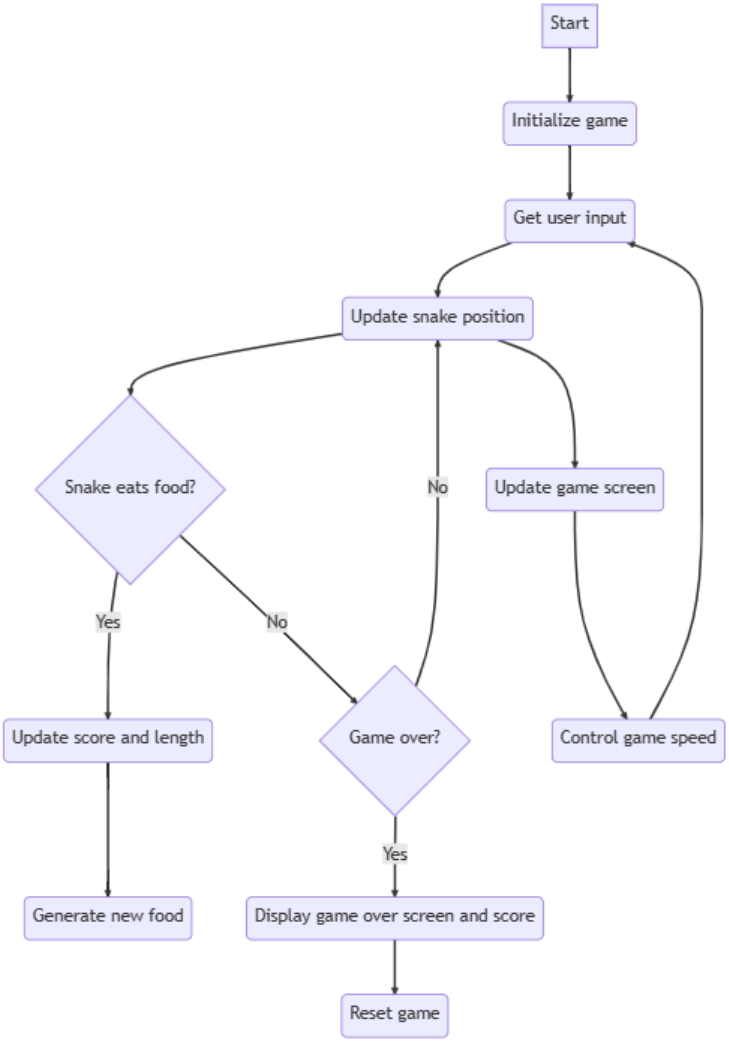


图 8 贪吃蛇项目流程图

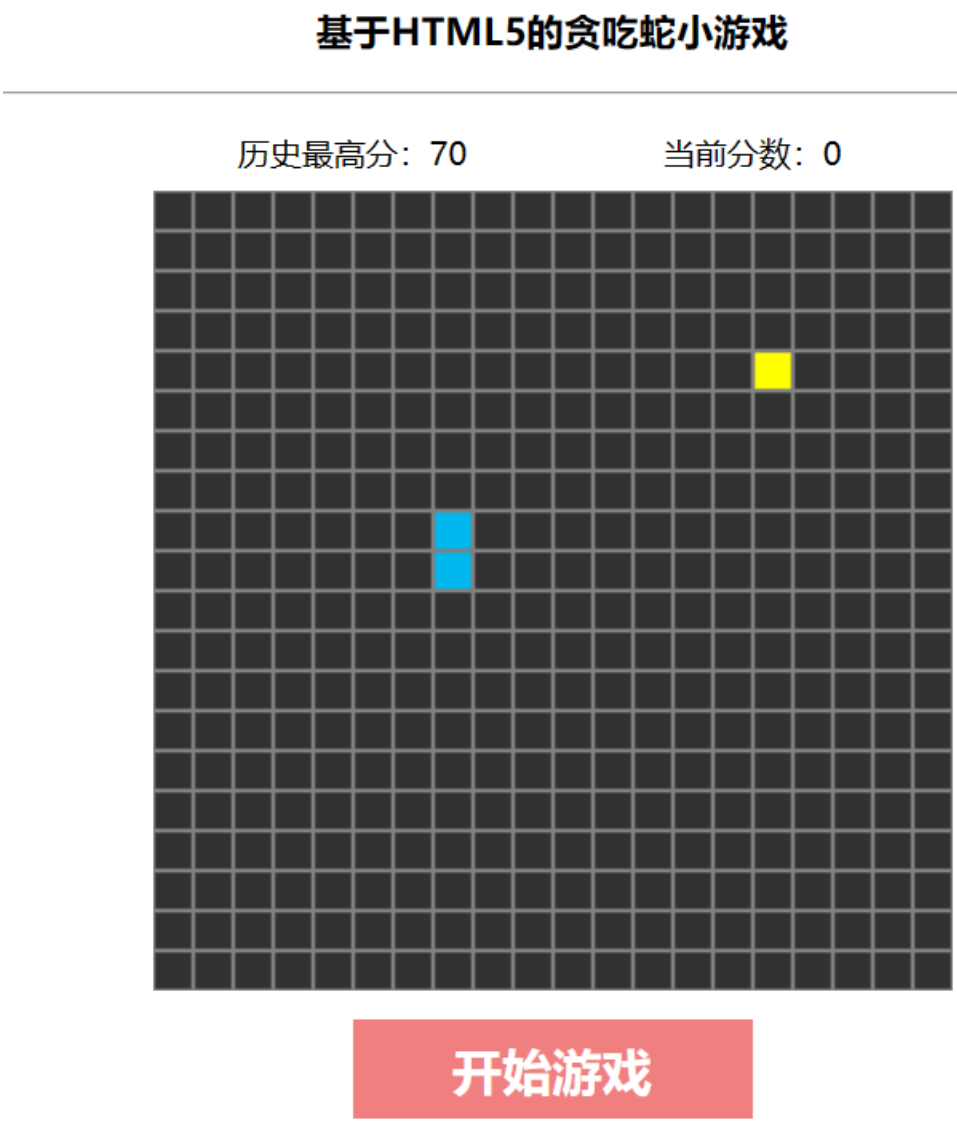


**图 9 贪吃蛇游戏设计图**

通过利用随机数加入碰撞条件以及重合条件，每次随即生成小蛇与食物，并且通过碰撞检测以及自我碰撞来检测是否游戏结束，与此同时每次吃到食物几分 10 分，并显示与更新当前得分与最高分数。

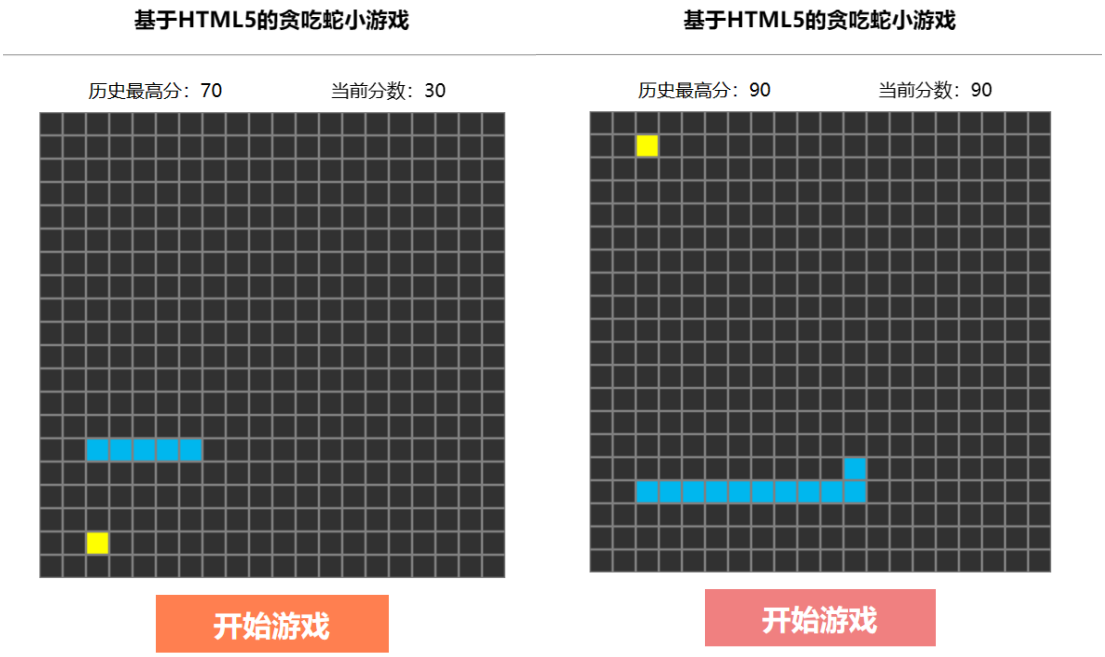


图 10 贪吃蛇分数更新图

如图 10 所示，本次实验贪吃蛇可以做到历史最高分数与当前分数的显示以及实时更新。

## 四、主要知识点

- 整体设计：
  - 使用 标签划分区域
  - 使用 CSS 样式
- 信息展示区设计：
  - 使用 标签划分区域
- 主游戏界面设计：
  - 使用<canvas>元素制作游戏画面
  - 使用<button>元素制作按钮
- 数据模型设计
  - 采用数组创建贪吃蛇模型
  - 利用方向参数设计蛇身移动模型
  - 采用数组创建食物模型
- 游戏逻辑实现
  - 设置贪吃蛇的初始状态，包括蛇身长度、首次出现的位置和移动方向
  - 蛇吃食物模型增加分数并重新随机生成一个不与小蛇重合的食物数组
  - 自定义 drawSnake()专门用于绘制贪吃蛇
  - 使用 document 对象的 onkeydown 方法监听并获取用户按键
  - 创建 detectCollision()函数用于进行蛇与障碍物的碰撞检测
  - 显示历史最高分：使用 HTML5 Web 存储 API 中的 localStorage 进行历史最高分记录的读取；声明 showBestScore()方法用于获取并在状态栏展示

历史最高分

- 当蛇碰撞到墙壁或者自身导致游戏失败时会自动重新开始游戏
- 控制游戏速度

## 五、心得体会（含意见建议）

代码结构和模块化设计：在编写代码时，应该采用模块化设计，以便于维护和扩展。例如，可以将游戏逻辑和界面渲染分离为不同的模块，或者将不同功能的代码分离为不同的文件。贪吃蛇游戏的速度和移动方向对于游戏体验至关重要。应该确保游戏速度适中，不会过快或过慢，同时确保贪吃蛇能够按照玩家的操作进行移动。在游戏过程中，应该时刻记录贪吃蛇的状态和当前游戏状态，以便于处理游戏结束的情况。当贪吃蛇碰到边界或自身时，游戏应该结束，并展示游戏结束的界面。在编写代码时，应该注意测试和调试。可以使用各种工具和技术进行测试和调试，以确保游戏功能和性能的稳定和优化。

意见建议：加强对 HTML/CSS/JavaScript 基础知识的讲解，让学生对语法和语义有更深入的理解；对于游戏逻辑和算法的讲解，可以适当提高难度，让学生有挑战性；强调代码规范和注重代码质量，例如使用代码检查工具、进行代码评审等。