

一、实验准备

课程主页: [课程主页\(gitee.com\)](#)

实验文档: [lab6文档](#)

初始资料: [项目初始文件夹压缩包](#)

参考资料: [参考资料](#)

二、实验目标

- 1、综合所学知识创建完整的推箱子游戏;
- 2、熟练掌握 canvas 和绘图 API。

三、实验步骤

1、项目创建和页面配置

基本流程见前两个lab, 在此不再赘述。

本项目一共需要 2 个页面, 即首页、游戏页。

1.1 首页功能需求

- (1) 首页需要包含标题和关卡列表。
- (2) 关卡至少要有 64个关卡选项, 每个关卡显示预览图片和第几关。
- (3) 点击关卡列表可以打开对应的游戏画面。

1.2 游戏页功能需求

- (1) 游戏页面需要显示游戏第几关、游戏画面、方向键和“重新开始”按钮。
- (2) 点击方向键可以使游戏主角自行移动或推动箱子前进。
- (3) 游戏画面由8*8的小方块组成, 主要包括地板、围墙、箱子、游戏主角和目的地。
- (4) 点击“重新开始”按钮可以将箱子和游戏主角回归初始位置并重新开始游戏。

2、全局设计

2.1 导航栏设计

编写App.json:

```

{
  "pages": [
    "pages/index/index",
    "pages/game/game"
  ],
  "window": {
    "backgroundTextStyle": "light",
    "navigationBarBackgroundColor": "#24A0ED",
    "navigationBarTitleText": "实验六：推箱子游戏",
    "navigationBarTextStyle": "white"
  },
  "sitemapLocation": "sitemap.json"
}

```

2.2 全局数据设计

编写utils / data.js:

```

//=====
//地图数据map1-map4
//地图数据：1墙，2路，3终点，4箱子，5人物，0墙的外围
//=====
//关卡1
var map1 = [
  [0, 1, 1, 1, 1, 1, 0, 0],
  [0, 1, 2, 2, 1, 1, 1, 0],
  [0, 1, 5, 4, 2, 2, 1, 0],
  [1, 1, 1, 2, 1, 2, 1, 1],
  [1, 3, 1, 2, 1, 2, 2, 1],
  [1, 3, 4, 2, 2, 1, 2, 1],
  [1, 3, 2, 2, 2, 4, 2, 1],
  [1, 1, 1, 1, 1, 1, 1, 1]
]
//关卡2
var map2 = [
  [0, 0, 1, 1, 1, 0, 0, 0],
  [0, 0, 1, 3, 1, 0, 0, 0],
  [0, 0, 1, 2, 1, 1, 1, 1],
  [1, 1, 1, 4, 2, 4, 3, 1],
  [1, 3, 2, 4, 5, 1, 1, 1],
  [1, 1, 1, 1, 4, 1, 0, 0],
  [0, 0, 0, 1, 3, 1, 0, 0],
  [0, 0, 0, 1, 1, 1, 0, 0]
]
//关卡3
var map3 = [
  [0, 0, 1, 1, 1, 1, 0, 0],
  [0, 0, 1, 3, 3, 1, 0, 0],
  [0, 1, 1, 2, 3, 1, 1, 0],
  [0, 1, 2, 2, 4, 3, 1, 0],
  [1, 1, 2, 2, 5, 4, 1, 1],
  [1, 2, 2, 1, 4, 4, 2, 1],
  [1, 2, 2, 2, 2, 2, 2, 1],
  [1, 1, 1, 1, 1, 1, 1, 1]
]

```

```

]
//关卡4
var map4 = [
  [0, 1, 1, 1, 1, 1, 1, 0],
  [0, 1, 3, 2, 3, 3, 1, 0],
  [0, 1, 3, 2, 4, 3, 1, 0],
  [1, 1, 1, 2, 2, 4, 1, 1],
  [1, 2, 4, 2, 2, 4, 2, 1],
  [1, 2, 1, 4, 1, 1, 2, 1],
  [1, 2, 2, 2, 5, 2, 2, 1],
  [1, 1, 1, 1, 1, 1, 1, 1]
]

module.exports = {
  maps: [map1, map2, map3, map4]
}

```

3、视图与逻辑设计

3.1 首页设计

首页主要包含两部分内容，即标题和关卡列表。

计划使用如下组件：

- 顶端标题：容器；
- 关卡列表：容器，内部使用数组循环。

编写index.wxml：

```

<!--pages/index/index.wxml-->
<view class="container">
  <!-- 标题 -->
  <view class="title">游戏选关</view>
  <!-- 关卡列表 -->
  <view class="levelBox">
    <view class="box" wx:for="{{levels}}" wx:key="index" bindtap="chooseLevel"
    data-level="{{index}}">
      <image src="/images/{{item}}"></image>
      <text>第{{index+1}}关</text>
    </view>
  </view>
</view>

```

编写index.wxss

```

/* pages/index/index.wxss */
.levelBox{
  width: 100%;
}
.box{
  width: 50%;
  float: left;
  margin: 20rpx 0;
}

```

```

display: flex;
flex-direction: column;
align-items: center;
}

image{
width: 300rpx;
height: 300rpx;
}

```

编写index.js:

```

// pages/index/index.js
Page({

  /**
   * 页面的初始数据
   */
  data: {
    levels: [
      "level01.png",
      "level02.png",
      "level03.png",
      "level04.png"
    ]
  },
  chooseLevel: function(e){
    let level=e.currentTarget.dataset.level
    wx.navigateTo({
      url: '../game/game?level='+level
    })
  },
  /**
   * 生命周期函数--监听页面加载
   */
  onLoad: function (options) {

  },

  /**
   * 生命周期函数--监听页面初次渲染完成
   */
  onReady: function () {

  },

  /**
   * 生命周期函数--监听页面显示
   */
  onShow: function () {

  },

  /**
   * 生命周期函数--监听页面隐藏

```

```

    */
    onHide: function () {

    },

    /**
     * 生命周期函数--监听页面卸载
     */
    onUnload: function () {

    },

    /**
     * 页面相关事件处理函数--监听用户下拉动作
     */
    onPullDownRefresh: function () {

    },

    /**
     * 页面上拉触底事件的处理函数
     */
    onReachBottom: function () {

    },

    /**
     * 用户点击右上角分享
     */
    onShareAppMessage: function () {

    }
  })

```

3.2 游戏页设计

游戏页面需要用户点击首页的关卡列表；

然后在新窗口中打开该页面。游戏页面包括游戏标题、游戏画面、方向键和“重新开始”按钮；

计划使用如下组件：

- view：整体容器和顶端标题；
- button：四个方向键和一个“重新开始”按钮，一个“上一关”按钮和一个“下一关按钮”；
- canvas：表示游戏画布。

编写game.wxml：

```

<!--pages/game/game.wxml-->
<view class="container">
  <!-- 关卡提示 -->
  <view class="title">第{{level}}关</view>
  <!-- 游戏画布 -->
  <canvas canvas-id="myCanvas"></canvas>

  <!-- 方向键 -->

```

```

<view class="btnBox">
  <button class="custom-btn" bindtap="up">↑</button>
  <view>
    <button class="custom-btn" bindtap='left'>←</button>
    <button class="custom-btn" bindtap='down'>↓</button>
    <button class="custom-btn" bindtap='right'>→</button>
  </view>
</view>

<!-- 按钮容器 -->
<view class="button-container">
  <!-- 上一关 -->
  <button class="custom-btn {{level == 1 ? 'disabled-btn' : ''}}"
bindtap="prevLevel" disabled="{{level == 1}}">上一关</button>
  <!-- 重新开始 -->
  <button class="custom-btn" bindtap="restartGame">重新开始</button>
  <!-- 下一关 -->
  <button class="custom-btn {{level == 4 ? 'disabled-btn' : ''}}"
bindtap="nextLevel" disabled="{{level == 4}}">下一关</button>
</view>
</view>

```

编写game.wxss

```

/* pages/game/game.wxss */
/* 游戏画布样式 */
canvas {
  border: 1rpx solid;
  width: 320px;
  height: 320px;
}

/* 方向键按钮整体区域 */
.btnBox {
  display: flex;
  flex-direction: column;
  align-items: center;
  color: #24A0ED;
}

/* 方向键按钮第二行 */
.btnBox view {
  display: flex;
  flex-direction: row;
}

/* 所有方向键按钮 */
.btnBox button {
  width: 90rpx;
  height: 90rpx;
}

/* 所有按钮样式 */
button {
  margin: 10rpx;
}

```

```

}

/* 按钮容器样式 */
.button-container {
  display: flex;
  justify-content: space-around;
  margin-top: 20px;
}

.custom-btn {
  background-color: #24A0ED; /* 设置背景颜色 */
  color: white; /* 设置文字颜色 */
  border-radius: 10rpx; /* 设置圆角 */
  padding: 10rpx 20rpx; /* 设置内边距 */
}

/* 禁用按钮样式 */
.disabled-btn {
  background-color: transparent;
  color: #ccc;
  border: 1px solid #ccc;
}

```

编写game.js:

```

// pages/game/game.js
var data = require('../../utils/data.js')
//地图图层数据
var map = [
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0]
]
//箱子图层数据
var box = [
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0, 0]
]
//方块的宽度
var w = 40
//初始化小鸟的行与列
var row = 0
var col = 0

```

```

Page({

  /**
   * 页面的初始数据
   */
  data: {
    level: 1
  },

  /**
   * 自定义函数--初始化地图数据
   */
  initMap: function(level) {
    // 读取原始的游戏地图数据
    let mapData = data.maps[level]
    //使用双重for循环记录地图数据
    for (var i = 0; i < 8; i++) {
      for (var j = 0; j < 8; j++) {
        box[i][j] = 0
        map[i][j] = mapData[i][j]

        if (mapData[i][j] == 4) {
          box[i][j] = 4
          map[i][j] = 2
        } else if (mapData[i][j] == 5) {
          map[i][j] = 2
          //记录小鸟的当前行和列
          row = i
          col = j
        }
      }
    }
  },

  /**
   * 自定义函数--绘制地图
   */
  drawCanvas: function() {
    let ctx = this.ctx
    //清空画布
    ctx.clearRect(0, 0, 320, 320)
    //使用双重for循环绘制8x8的地图
    for (var i = 0; i < 8; i++) {
      for (var j = 0; j < 8; j++) {
        //默认是道路
        let img = 'ice'
        if (map[i][j] == 1) {
          img = 'stone'
        } else if (map[i][j] == 3) {
          img = 'pig'
        }

        //绘制地图
        ctx.drawImage('/images/icons/' + img + '.png', j * w, i * w, w, w)

        if (box[i][j] == 4) {
          //叠加绘制箱子
          ctx.drawImage('/images/icons/box.png', j * w, i * w, w, w)
        }
      }
    }
  }
})

```



```

    }
}

//叠加绘制小鸟
ctx.drawImage('/images/icons/bird.png', col * w, row * w, w, w)

ctx.draw()
},

/**
 * 自定义函数--方向键：上
 */
up: function() {
    //如果不在最顶端才考虑上移
    if (row > 0) {
        //如果上方不是墙或箱子,可以移动小鸟
        if (map[row - 1][col] != 1 && box[row - 1][col] != 4) {
            //更新当前小鸟坐标
            row = row - 1
        }
        //如果上方是箱子
        else if (box[row - 1][col] == 4) {
            //如果箱子不在最顶端才能考虑推动
            if (row - 1 > 0) {
                //如果箱子上方不是墙或箱子
                if (map[row - 2][col] != 1 && box[row - 2][col] != 4) {
                    box[row - 2][col] = 4
                    box[row - 1][col] = 0
                    //更新当前小鸟坐标
                    row = row - 1
                }
            }
        }
    }
    //重新绘制地图
    this.drawCanvas()
    //检查游戏是否成功
    this.checkwin()
}
},

/**
 * 自定义函数--方向键：下
 */
down: function() {
    //如果不在最底端才考虑下移
    if (row < 7) {
        //如果下方不是墙或箱子,可以移动小鸟
        if (map[row + 1][col] != 1 && box[row + 1][col] != 4) {
            //更新当前小鸟坐标
            row = row + 1
        }
        //如果下方是箱子
        else if (box[row + 1][col] == 4) {
            //如果箱子不在最底端才能考虑推动
            if (row + 1 < 7) {
                //如果箱子下方不是墙或箱子
                if (map[row + 2][col] != 1 && box[row + 2][col] != 4) {
                    box[row + 2][col] = 4
                    box[row + 1][col] = 0
                }
            }
        }
    }
}
}
}

```

```

        //更新当前小鸟坐标
        row = row + 1
    }
}
}
//重新绘制地图
this.drawCanvas()
//检查游戏是否成功
this.checkwin()
}
},
/**
 * 自定义函数--方向键：左
 */
left: function() {
    //如果不在最左侧才考虑左移
    if (col > 0) {
        //如果左侧不是墙或箱子,可以移动小鸟
        if (map[row][col - 1] != 1 && box[row][col - 1] != 4) {
            //更新当前小鸟坐标
            col = col - 1
        }
        //如果左侧是箱子
        else if (box[row][col - 1] == 4) {
            //如果箱子不在最左侧才能考虑推动
            if (col - 1 > 0) {
                //如果箱子左侧不是墙或箱子
                if (map[row][col - 2] != 1 && box[row][col - 2] != 4) {
                    box[row][col - 2] = 4
                    box[row][col - 1] = 0
                    //更新当前小鸟坐标
                    col = col - 1
                }
            }
        }
    }
    //重新绘制地图
    this.drawCanvas()
    //检查游戏是否成功
    this.checkwin()
}

},
/**
 * 自定义函数--方向键：右
 */
right: function() {
    //如果不在最右侧才考虑右移
    if (col < 7) {
        //如果右侧不是墙或箱子,可以移动小鸟
        if (map[row][col + 1] != 1 && box[row][col + 1] != 4) {
            //更新当前小鸟坐标
            col = col + 1
        }
        //如果右侧是箱子
        else if (box[row][col + 1] == 4) {
            //如果箱子不在最右侧才能考虑推动
            if (col + 1 < 7) {

```

```

        //如果箱子右侧不是墙或箱子
        if (map[row][col + 2] !== 1 && box[row][col + 2] !== 4) {
            box[row][col + 2] = 4
            box[row][col + 1] = 0
            //更新当前小鸟坐标
            col = col + 1
        }
    }
}
//重新绘制地图
this.drawCanvas()
//检查游戏是否成功
this.checkwin()
}
},

```

```

/**
 * 自定义函数--判断游戏成功
 */
iswin: function() {
    //使用双重for循环遍历整个数组
    for (var i = 0; i < 8; i++) {
        for (var j = 0; j < 8; j++) {
            //如果有箱子没在终点
            if (box[i][j] === 4 && map[i][j] !== 3) {
                //返回假，游戏尚未成功
                return false
            }
        }
    }
    //返回真，游戏成功
    return true
},

```

```

/**
 * 自定义函数--游戏成功处理
 */
checkwin: function() {
    if (this.iswin()) {
        wx.showModal({
            title: '恭喜',
            content: '游戏成功!',
            showCancel: false,
            success: (res) => {
                if (res.confirm) {
                    wx.redirectTo({
                        url: '/pages/index/index'
                    })
                }
            }
        })
    }
}
},

```

```

/**
 * 自定义函数--重新开始游戏

```

```

    */
    restartGame: function() {
        //初始化地图数据
        this.initMap(this.data.level - 1)
        //绘制画布内容
        this.drawCanvas()
    },

    // 上一关
    prevLevel: function() {
        if (this.data.level > 1) {
            wx.redirectTo({
                url: '/pages/game/game?level=' + (this.data.level - 2)
            });
        }
    },

    // 下一关
    nextLevel: function() {
        if (this.data.level < 4) {
            wx.redirectTo({
                url: '/pages/game/game?level=' + this.data.level
            });
        }
    },

    /**
     * 生命周期函数--监听页面加载
     */
    onLoad: function(options) {
        //获取关卡
        let level= options.level
        //更新页面关卡标题
        this.setData({
            level: parseInt(level)+1
        })
        //创建画布上下文
        this.ctx = wx.createCanvasContext('myCanvas')
        //初始化地图数据
        this.initMap(level)
        //绘制画布内容
        this.drawCanvas()
    },

    })

```

四、程序运行结果

(1) 首页

21:10

80%

实验六：推箱子游戏



游戏选关



第1关



第2关





第3关




第4关



CSDN @归零者007

(2) 第一关详情。每一关可以重新开始，也可以跳转到上一关或下一关（如果有的话）。

21:11

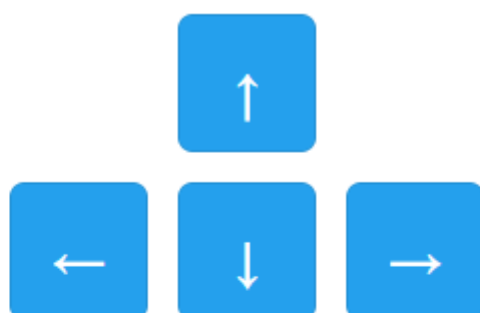
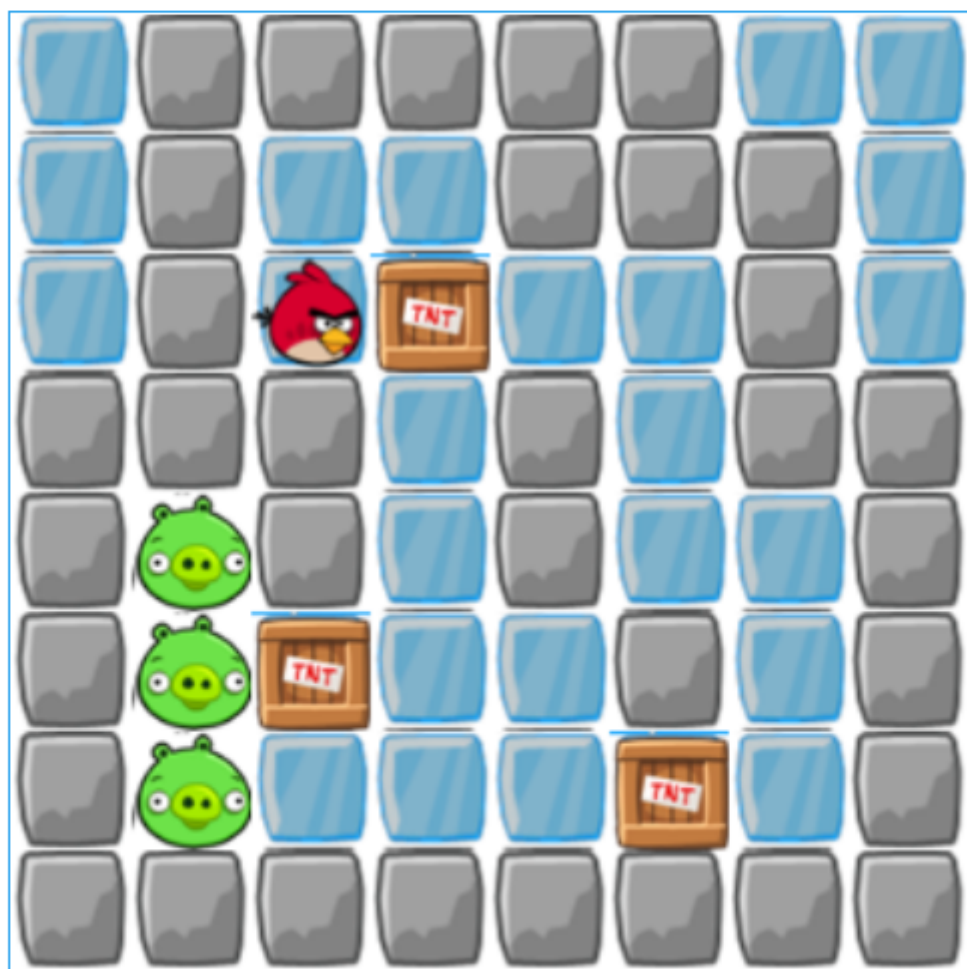
80% 



实验六：推箱子游戏



第1关





(3) 第二关详情

21:12

80% 



实验六：推箱子游戏



第2关



上一关

重新开始

下一关

CSDN @归零者007

(4) 第三关详情与第二关类似

(5) 第四关详情

21:12

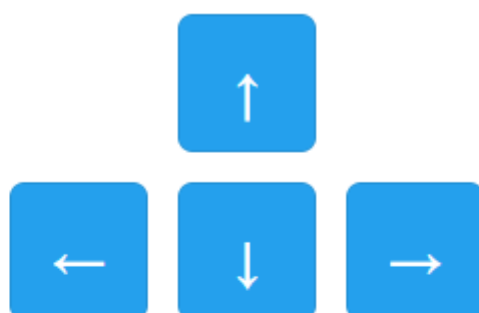
80%



实验六：推箱子游戏



第4关





(6) 第二关通关。点击确定后跳转回首页。

21:13

80%



实验六：推箱子游戏



第2关





五、问题总结与体会

1. 新增上下关按钮

- 增加了上下关按钮的功能，用户点击“上一关”按钮时，会跳转到上一关，第一关没有上一关，按钮镂空；“下一关”按钮类似。

```
<!--pages/game/game.wxml-->
.....
<!-- 按钮容器 -->
<view class="button-container">
  <!-- 上一关 -->
  <button class="custom-btn {{level == 1 ? 'disabled-btn' : ''}}"
bindtap="prevLevel" disabled="{{level == 1}}">上一关</button>
  <!-- 重新开始 -->
  <button class="custom-btn" bindtap="restartGame">重新开始</button>
  <!-- 下一关 -->
  <button class="custom-btn {{level == 4 ? 'disabled-btn' : ''}}"
bindtap="nextLevel" disabled="{{level == 4}}">下一关</button>
</view>
```

```
/* pages/game/game.wxss */
.....
/* 按钮容器样式 */
.button-container {
  display: flex;
  justify-content: space-around;
  margin-top: 20px;
}

/* 禁用按钮样式 */
.disabled-btn {
  background-color: transparent;
  color: #ccc;
  border: 1px solid #ccc;
}
```

```
// pages/game/game.js
.....
// 上一关
prevLevel: function() {
  if (this.data.level > 1) {
    wx.redirectTo({
      url: '/pages/game/game?level=' + (this.data.level - 2)
    });
  }
},

// 下一关
nextLevel: function() {
  if (this.data.level < 4) {
    wx.redirectTo({
      url: '/pages/game/game?level=' + this.data.level
    });
  }
},

```

2. 优化游戏成功处理逻辑

- 游戏成功弹出提示弹窗之后，点击确定可以返回小程序首页。

```
// pages/game/game.js
.....
/**
 * 自定义函数--游戏成功处理
 */
checkwin: function() {
  if (this.iswin()) {
    wx.showModal({
      title: '恭喜',
      content: '游戏成功!',
      showCancel: false,
      success: (res) => {
        if (res.confirm) {
          wx.redirectTo({
            url: '/pages/index/index'
          })
        }
      }
    })
  }
},

```

收获与体会：

- 掌握组件使用与样式设计：
 - 学会了使用各种组件及其属性，如 `wx:for`、`wx:key` 和 `wx:for-item` 等，实现了高效的循环处理。
 - 熟练使用 `id` 和 `class` 等样式选择器，理解了样式的作用域，避免了代码冗余。
- 强化小程序设计流程：

- 通过实际项目，强化了从页面布局、样式设计到逻辑实现的完整设计流程思维。
- 设计并实现了一个有趣的推箱子小游戏，提升了对相关代码和组件的理解。
- **综合应用与技能提升：**
 - 综合所学知识，成功创建了完整的推箱子游戏，熟练掌握了 `canvas` 和绘图 API。
 - 通过小游戏的开发，进一步理解了 JavaScript 及相关技术，收获满满。