



西安电子科技大学

# 课程报告

名称：Java Web 程序设计第一次作业

小组成员	
姓名	学号
谢锦源	14030130100
王鹏	14030130101

# 1. 设计概述

本次设计的是一个具有多页面的静态网站，在设计过程中，使用了 HTML（超文本标记语言）以及 CSS（层叠样式表）。

页面中主要采用了以下标记：

标记	作用
<!DOCTYPE>	定义文档的类型
<head>	定义文档头部
<html>	定义 HTML 文档
<link>	链接外部样式
<title>	设置文档标题
<body>	定义文档主要内容
<nav>	定义导航菜单
<img>	向网页嵌入图片
<div>	文档块元素
<article>	定义文章
<p>	定义段落
<a>	定义超链接
<h1> - <h6>	定义标题
<pre>	定义预格式化文本
<code>	程序代码文本
<ul>	无序 HTML 列表
<li>	列表项目
<strong>	文本加粗
<em>	将文本定义为强调内容

# 2. 主要代码结构

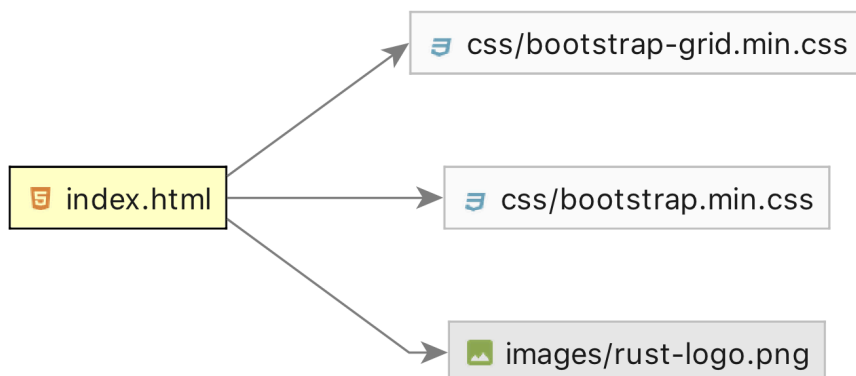
HTML Web 页面为树形结构。本次设计中页内代码的基本结构如下：

```
<!DOCTYPE html>  
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <title>...</title>
</head>
<body class="container">
  <!-- 导航栏开始 -->
  <nav>
    ...
  </nav>
  <!-- 正文开始 -->
  <article>
    ...
  </article>
</body>
</html>
```

第一行定义文档的类型为 HTML，紧接着定义 HTML 文档内容。HTML 文档中，最外层包含了两个对象 `<head>` 以及 `<body>`。`<head>` 块用于引用脚本、提供元信息等，最后设置页面的标题。`<body>` 块中，包含了 `<nav>`（导航栏的定义）以及正文部分 `<article>`。


同时，网站使用了 Bootstrap 用于页面布局与排版，下图为其中一个页面的外部资源引用关系：



# 3. 运行结果

网站一共设计了 4 个页面，以下为运行结果：

## 3.1. 页面 1：index.html

 简介 第一个 Rust 程序 学习资源

### Rust 程序设计语言

安全、高效、并发

开始学习

#### 安全

Rust 通过所有权系统来保证内存安全。所有的这一切都在编译期完成检查，这意味着你不需要付出运行时开销。

[了解 Rust 的安全机制](#)

#### 高效

Rust 没有垃圾回收机制，依赖于其所有权系统，内存分配器就确定了应该在何时分配和释放，这使得程序更加高效。


[了解更多](#)

#### 并发

Rust 提供了语言层面上的并发支持，这使得我们更容易编写正确的并发（并行）程序，充分利用多核处理器的优势。

[了解更多](#)

## 3.2. 页面 2：intro.html

 简介 第一个 Rust 程序 学习资源

Rust是一个由 Mozilla 主导开发的通用、编译型编程语言。它的设计准则为“安全，并发，实用”，支持函数式、并发式、过程式以及面向对象的编程风格。Rust 语言原本是 Mozilla 员工 Graydon Hoare 的私人项目，而 Mozilla 于 2009 年开始赞助这个项目，并且在 2010 年首次揭露了它的存在。也在同一年，它的编译器源代码开始由原本的 OCaml语言转移到用 Rust 语言，进行 bootstrapping 工作，称做 **rustc**，并于 2011 年实际完成。这个可自我编译的编译器在架构上采用了 LLVM 做为它的后端。

第一个有版本号的 Rust 编译器于 2012 年 1 月发布。Rust 1.0 是第一个稳定版本，于 2015 年 5 月 15 日发布。

Rust 是在完全开放的情况下进行开发，并且相当欢迎社区的回馈。在 1.0 稳定版之前，语言设计也因为通过撰写 Servo 网页浏览器排版引擎和 rustc 编译器本身，而有进一步的改善。虽然它由 Mozilla资助，但它其实是一个共有项目，有很大部分的代码是来自于社区的贡献者。

Rust 是一个着重于安全、速度和并发的编程语言。它的设计不仅可以使程序获得性能和对底层语言的控制，并且能够享受高级语言强大的抽象能力。这些特性使得 Rust 适合那些有类似 C 语言经验并正在寻找一个更安全的替代者的程序员，同时也适合那些来自类似 Python 语言背景，正在探索在不牺牲表现力的情况下编写更好性能代码的开发者。

Rust在编译时进行其绝大多数的安全检查和内存管理决策，因此程序运行时期的性能没有受到影响。这让其在许多其他语言不擅长的应用场景中得以大显身手：存在可预测空间和时间要求的程序，嵌入到其他语言中，以及编写底层代码，如设备驱动和操作系统。Rust 也很擅长 web 程序：它驱动着 Rust 包注册网站：[crates.io!](#)

[开始学习 Rust 语言!](#)

### 3.3. 页面 3: first.html

[简介](#) [第一个 Rust 程序](#) [学习资源](#)

#### 第一个 Rust 程序

当学习一门新语言的时候，编写一个在屏幕上打印 "Hello, world!" 文本的小程序是一个传统，下面我们将讲述如何利用 Rust 编写这个小程序。

首先，我们创建一个 Rust 源文件，**hello.rs**，正如你所看见的，Rust 的源代码文件以 **.rs** 后缀结束。

现在打开刚创建的 **hello.rs** 文件，并输入如下代码：

Filename: hello.rs

```
fn main() {  
    println!("Hello, world!");  
}
```

保存文件，并回到终端窗口。在 Linux 或 OSX 上，输入如下命令：

```
$ rustc hello.rs  
$ ./hello  
Hello, world!
```

在 Windows 上，运行 **.\hello.exe** 而不是 **./hello**。不管使用何种系统，你应该在终端看到 **Hello, world!** 字符串。如果你做到了，那么恭喜你！你已经正式编写了一个 Rust 程序。

#### 分析 Rust 程序

现在，让我们回过头来仔细看看 "Hello, world!" 程序到底发生了什么。下面是第一部分：

```
fn main() {  
}
```

这几行定义了一个 Rust **函数**。**main** 函数是特殊的：这是每一个可执行的 Rust 程序首先运行的函数（译者注：入口点）。第一行表示“定义一个名叫 **main** 的函数，没有参数也没有返回值。”如果有参数的话，它们应该出现在括号中，**(和)**。

同时注意函数体被包裹在大括号中，**(和)**。Rust 要求所有函数体都位于大括号中。将前一个大括号与函数声明置于一行，并留有一个空格被认为是一个好的代码风格。

在 **main()** 函数中：


```
    println!("Hello, world!");
```

这行代码做了这个小程序的所有工作：它在屏幕上打印文本。这里有很多需要注意的细节。第一个是 Rust 代码风格使用 4 个空格缩进，而不是 1 个制表符。第二个重要的部分是 **println!()**。这叫做 Rust **宏**，是如何进行 Rust 元编程的关键所在。相反如果是调用一个函数的话，它应该看起来像这样：**println**（没有 **!**）。你只需记住当看到符号 **!** 的时候，意味着你在调用一个宏而不是一个普通的函数。

接下来，**"Hello, world!"** 是一个 **字符串**。我们把这个字符串作为一个参数传递给 **println!**，它负责在屏幕上打印这个字符串。

这一行以一个分号结尾（**;**）。**;** 代表这个表达式的结束和下一个表达式的开始。大部分 Rust 代码行以 **;** 结尾。

## 4. 页面 4: learn.html

[简介](#) [第一个 Rust 程序](#) [学习资源](#)

Rust 中文资源

[中文文档](#)  
[Rust By Example 中文翻译](#)  
[Rust Primer](#)  
[Rust 中文社区](#)  
[The Little Book of Rust Macros 中文翻译](#)

Rust 官方资源

[Rust 官方网站](#)  
[Rust 代码仓库](#)  
[官方参考文档](#)

## 5. 附录：完整代码

### 5.1. index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="assets/css/bootstrap.min.css">
  <link rel="stylesheet" href="assets/css/bootstrap-grid.min.css">
  <title>Rust 程序设计语言</title>
</head>
<body class="container">
<!-- 导航栏开始 -->
<nav class="navbar navbar-toggleable-md navbar-light bg-faded">
  <a class="navbar-brand" href="index.html">
    
  </a>
  <ul class="navbar-nav mr-auto">
    <li class="nav-item"><a class="nav-link" href="intro.html">简介</a></li>
    <li class="nav-item"><a class="nav-link" href="first.html">第一个 Rust 程序</a></li>
    <li class="nav-item"><a class="nav-link" href="learn.html">学习资源</a></li>
  </ul>
</nav>
<!-- 导航栏结束 -->
<!-- 中间大屏开始 -->
<div class="jumbotron">
  <div class="container" align="center">
```

```
<div class="row">
  <div class="col-md-8 center-block offset-md-2">
    <br><br><br>
    <h3 class="text-info">Rust 程序设计语言</h3>
    <br>
    <div class="text-muted">安全、高效、并发</div>
    <br>
    <a class="btn btn-primary" href="learn.html" role="button">开
始学习</a>
  </div>
</div>
</div>
</div>
<!-- 中部大屏结束 -->
<div class="row">
  <div class="card col-md-4">
    <div class="card-block">
      <h4 class="card-title">安全</h4>
      <p class="card-text">Rust 通过所有权系统来保证内存安全。所
有的这一切都在编译期完成检查，这意味着你不需要付出运行时开销。
</p>
      <a href="#" class="card-link">了解 Rust 的安全机制</a>
    </div>
  </div>
  <div class="card col-md-4">
    <div class="card-block">
      <h4 class="card-title">高效</h4>
      <p class="card-text">Rust 没有垃圾回收机制，依赖于其所有权
系统，内存存在编译器就确定了应该在何时分配和释放，这使得程序更加高
效。</p>
      <a href="#" class="card-link">了解更多</a>
    </div>
  </div>
</div>
```

```
    </div>
</div>
<div class="card col-md-4">
    <div class="card-block">
        <h4 class="card-title">并发</h4>
        <p class="card-text">Rust 提供了语言层面的并发支持，这使得我们更容易编写正确的并发（并行）程序，充分利用多核处理器的优势。</p>
        <a href="#" class="card-link">了解更多</a>
    </div>
</div>
</div>
</body>
</html>
```

## 5.2. intro.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="assets/css/bootstrap.min.css">
    <link rel="stylesheet" href="assets/css/bootstrap-grid.min.css">
    <link rel="stylesheet" href="assets/css/typo.css">
    <title>Rust 程序设计语言</title>
</head>
<body class="container">

<!-- 导航栏开始 -->
<nav class="navbar navbar-toggleable-md navbar-light bg-faded">
    <a class="navbar-brand" href="index.html">
```



```

</a>
<ul class="navbar-nav mr-auto">
  <li class="nav-item"><a class="nav-link" href="intro.html">简介</a></li>
  <li class="nav-item"><a class="nav-link" href="first.html">第一个 Rust 程序</a></li>
  <li class="nav-item"><a class="nav-link" href="learn.html">学习资源</a></li>
</ul>
</nav>
<!-- 导航栏结束 -->
<!-- 正文开始 -->
<article style="margin-top: 20px">
<p>Rust 是一个由 Mozilla 主导开发的通用、编译型编程语言。它的设计准则为“安全，并发，实用”，支持函数式、并发式、过程式以及面向对象的编程风格。</p>
<p>Rust 语言原本是 Mozilla 员工 Graydon Hoare 的私人项目，而 Mozilla 于 2009 年开始赞助这个项目，并且在 2010 年首次揭露了它的存在。
  也在同一年，它的编译器源代码开始由原本的 OCaml 语言转移到用 Rust 语言，进行 bootstrapping 工作，称做 rustc，并于 2011 年实际完成。这个可自我编译的编译器在架构上采用了 LLVM 做为它的后端。</p>
<p>第一个有版本号的 Rust 编译器于 2012 年 1 月发布。Rust 1.0 是第一个稳定版本，于 2015 年 5 月 15 日发布。</p>
<p>Rust 是在完全开放的情况下进行开发，并且相当欢迎社区的回馈。在 1.0 稳定版之前，语言设计也因为通过撰写 Servo 网页浏览器排版引擎和 rustc 编译器本身，而有进一步的改善。
  虽然它由 Mozilla 资助，但它其实是一个共有项目，有很大部分的代码
```

是来自于社区的贡献者。</p>

<p>Rust 是一个着重于安全、速度和并发的编程语言。它的设计不仅可以使程序获得性能和对底层语言的控制，并且能够享受高级语言强大的抽象能力。

这些特性使得 Rust 适合那些有类似 C 语言经验并正在寻找一个更安全的替代者的程序员，同时也适合那些来自类似 Python 语言背景，正在探索在不牺牲表现力的情况下编写更好性能代码的开发者。</p>

<p>Rust 在编译时进行其绝大多数的安全检查和内存管理决策，因此程序运行时期的性能没有受到影响。这使其在许多其他语言不擅长的应用场景中得以大显身手：

存在可预测空间和时间要求的程序，嵌入到其他语言中，以及编写底层代码，如设备驱动和操作系统。Rust 也很擅长 web 程序：它驱动着 Rust 包注册网站：<a href="https://crates.io/">crates.io</a>!</p>

<a href="https://rust-lang.org">开始学习 Rust 语言! </a>  
</article>  
<!-- 正文结束 -->  
</body>  
</html>

### 5.3. first.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="assets/css/bootstrap.min.css">
  <link rel="stylesheet" href="assets/css/bootstrap-grid.min.css">
  <link rel="stylesheet" href="assets/css/typo.css">
  <title>Rust 程序设计语言</title>
```

```
</head>
<body class="container">

<!-- 导航栏开始 -->
<nav class="navbar navbar-toggleable-md navbar-light bg-faded">
  <a class="navbar-brand" href="index.html">
    
  </a>
  <ul class="navbar-nav mr-auto">
    <li class="nav-item"><a class="nav-link" href="intro.html">简介</a></li>
    <li class="nav-item"><a class="nav-link" href="first.html">第一个 Rust 程序</a></li>
    <li class="nav-item"><a class="nav-link" href="learn.html">学习资源</a></li>
  </ul>
</nav>

<!-- 导航栏结束 -->
<!-- 正文开始 -->
<article style="margin-top: 20px">
  <h2> 第一个 Rust 程序</h2>
  <p>当学习一门新语言的时候，编写一个在屏幕上打印 “Hello, world!” 文本的小程序是一个传统，下面我们将讲述如何利用 Rust 编写这个小程序。</p>
  <p>首先，我们创建一个 Rust 源文件，<em>hello.rs</em>，正如你所看见的，Rust 的源代码文件以 <em>.rs</em> 后缀结束。</p>
  <p>现在打开刚创建的 <em>hello.rs</em> 文件，并输入如下代码：</p>
  <p><span class="filename">Filename: hello.rs</span></p>
  <pre><code class="language-rust">fn main() {
```

```
println!("Hello, world!");
}
```

保存文件，并回到终端窗口。在 Linux 或 OSX 上，输入如下命令：

```
$ rustc hello.rs
$ ./hello
Hello, world!
```

在 Windows 上，运行 `.\hello.exe` 而不是 `./hello`。不管使用何种系统，你应该在终端看到 `Hello, world!` 字符串。如果你做到了，那么恭喜你！你已经正式编写了一个

Rust 程序。

#### 分析 Rust 程序

现在，让我们回过头来仔细看看 “Hello, world!” 程序到底发生了什么。下面是第一部分：

```
fn main() {
}
```

这几行定义了一个 Rust **函数**。`main` 函数是特殊的：这是每一个可执行的 Rust 程序首先运行的函数（译者注：入口点）。第一行表示“定义一个叫

`main` 的函数，没有参数也没有返回值。”如果有参数的话，它们应该出现在括号中，`( )` 和 `( )`。

同时注意函数体被包裹在大括号中，`{ }`。Rust 要求所有函数体都位于大括号中。将前一个大括号与函数声明置于一行，并留有一个空格被认为是一个好的代码风格。

在 `main()` 函数中：

```
println!("Hello, world!");
}
```

这行代码做了这个小程序的所有工作：它在屏幕上打印文本。这里有很多需要注意的细节。第一个是 Rust 代码风格使用 4 个空格缩进，而不是 1 个制表符。

第二个重要的部分是 `println!()`。这叫做 Rust 宏，是如何进行 Rust

元编程的关键所在。相反如果是调用一个函数的话，它应该看起来像这样： `println`（没有 `!`）。你只需记住当看到符号 `!` 的时候，意味着你在调用一个宏而不是一个普通的函数。

接下来， `&quot;Hello, world!&quot;` 是一个 `字符串`。我们把这个字符串作为一个参数传递给 `println!`，它负责在屏幕上打印这个字符串。

这一行以一个分号结尾（`;`）。 `;` 代表这个表达式的结束和下一个表达式的开始。大部分 Rust 代码行以 `;` 结尾。

<!-- 正文结束 -->

## 5.4. learn.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <link rel="stylesheet" href="assets/css/bootstrap.min.css">
```

```
  <link rel="stylesheet" href="assets/css/bootstrap-grid.min.css">
```

```
  <link rel="stylesheet" href="assets/css/typo.css">
```

```
<title>Rust 程序设计语言</title>
```

```
</head>
<body class="container">
<!-- 导航栏开始 -->
<nav class="navbar navbar-toggleable-md navbar-light bg-faded">
  <a class="navbar-brand" href="index.html">
    
  </a>
  <ul class="navbar-nav mr-auto">
    <li class="nav-item"><a class="nav-link" href="intro.html">简介</a></li>
    <li class="nav-item"><a class="nav-link" href="first.html">第一个 Rust 程序</a></li>
    <li class="nav-item"><a class="nav-link" href="learn.html">学习资源</a></li>
  </ul>
</nav>
<!-- 导航栏结束 -->
<!-- 正文开始 -->
<article class="row" style="margin-top: 20px">
  <div class="col-md-6">
    <h4>Rust 中文资源</h4>
    <ul>
      <li>
        <a href="https://github.com/KaiserY/rust-book-chinese">
          中文文档
        </a>
      </li>
      <li>
        <a href="https://github.com/rust-lang-cn/rust-by-example-cn">
          Rust By Example 中文翻译
        </a>
      </li>
    </ul>
  </div>
</article>
```

```
        </a>
    </li>
    <li>
        <a href="https://github.com/rustcc/RustPrimer">
            Rust Primer
        </a>
    </li>
    <li><a href="https://rust-china.org/">Rust 中文社
区</a></li>
    <li>
        <a href="https://github.com/DaseinPhaos/tlborm-chinese">
            The Little Book of Rust Macros 中文翻译
        </a>
    </li>
</ul>
</div>
<div class="col-md-6"></div>
<h4>Rust 官方资源</h4>
<ul>
    <li><a href="https://rust-lang.org"> Rust 官方网
站</a></li>
    <li><a href="https://github.com/rust-lang/rust">Rust 代码
仓库</a></li>
    <li><a href="https://doc.rust-lang.org/nightly/book/">官
方参考文档</a></li>
</ul>
</div>
</article>
<!-- 正文结束 -->
</body>
</html>
```