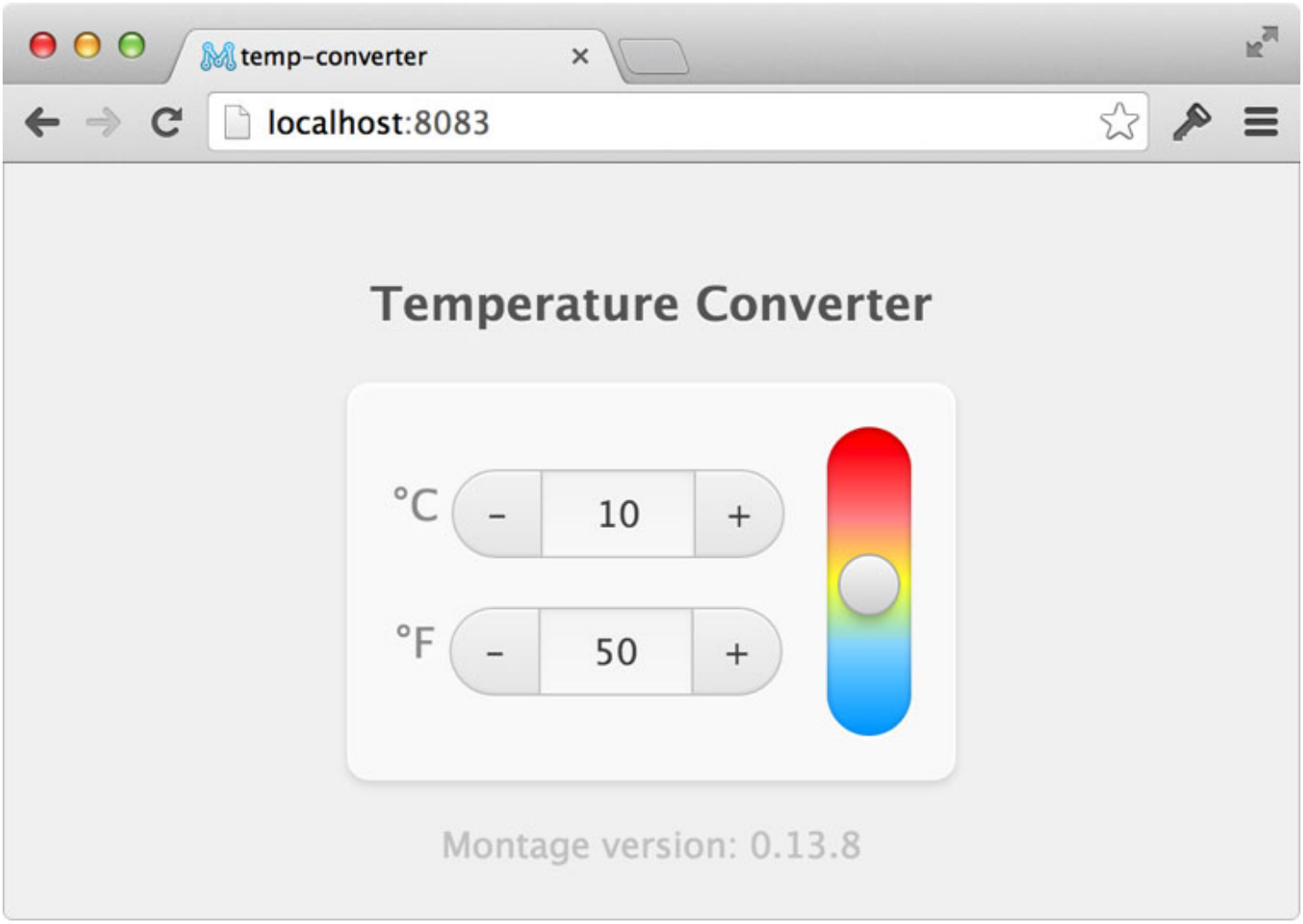
Hello MontageJS

你已经成功配置并且运行了第一个MontageJS应用，但是只有一个空白的页面：接下来我们做什么呢?

通过这个教程，你将实现一个简单的并能完美匹配移动设备的MontageJS应用：将摄氏温度转换成华氏温度，或将华氏温度转换为摄氏温度（如图1)。这个应用由三个元素构成，两个输入框，一个滑块，它们的值是互相绑定在一起的。 当你在一个输入框中输入一个数字之后，另外的一个输入框的值会自动改变，滑块也会移动到相应的位置。 同理，当你滑动滑块的时候，两个输入框的值也会改变成滑块元素当前对应的值。



图片 1. 你的最终目标是构建一个温度转换器

要完成这个教程，你必须掌握基本的HTML，CSS和JavaScript知识。

准备工作

确保你已经通过开始使用MontageJS配置好MontageJS开发环境。为了完成这个教程，node.js, npm, 和minit（MontageJS的初始化命令行工具）也是必须安装的。同时你还需要一个文本编辑器和一个最新稳定版本的浏览器Goolge Chrome, Safari, 或者Firefox。

新建一个工程

> **备注:** 如果你已经创建了一个新工程并且用minit成功运行了项目，你可以跳过以下步骤直接从“MontageJS基础知识”继续。

1. 打开终端或者命令行窗口输入

```
$ minit create:app -n temp-converter
```

2. 切换到temp-converter文件夹然后用minit运行你的项目

```
$ cd temp-converter
$ minit serve &
```

3. 打开浏览器转到http://localhost:8083/。

你将看到一个空白的网页，在网页的左上角有版本信息。

MontageJS 基础知识

MontageJS应用开发分为开发（构建应用）阶段和产品（优化开发版本）阶段。 在开发阶段我们使用一系列的组件来构建应用。这些组件存放在一个名为ui的文件夹中并且文件名以.reel结尾(如图2所示)。

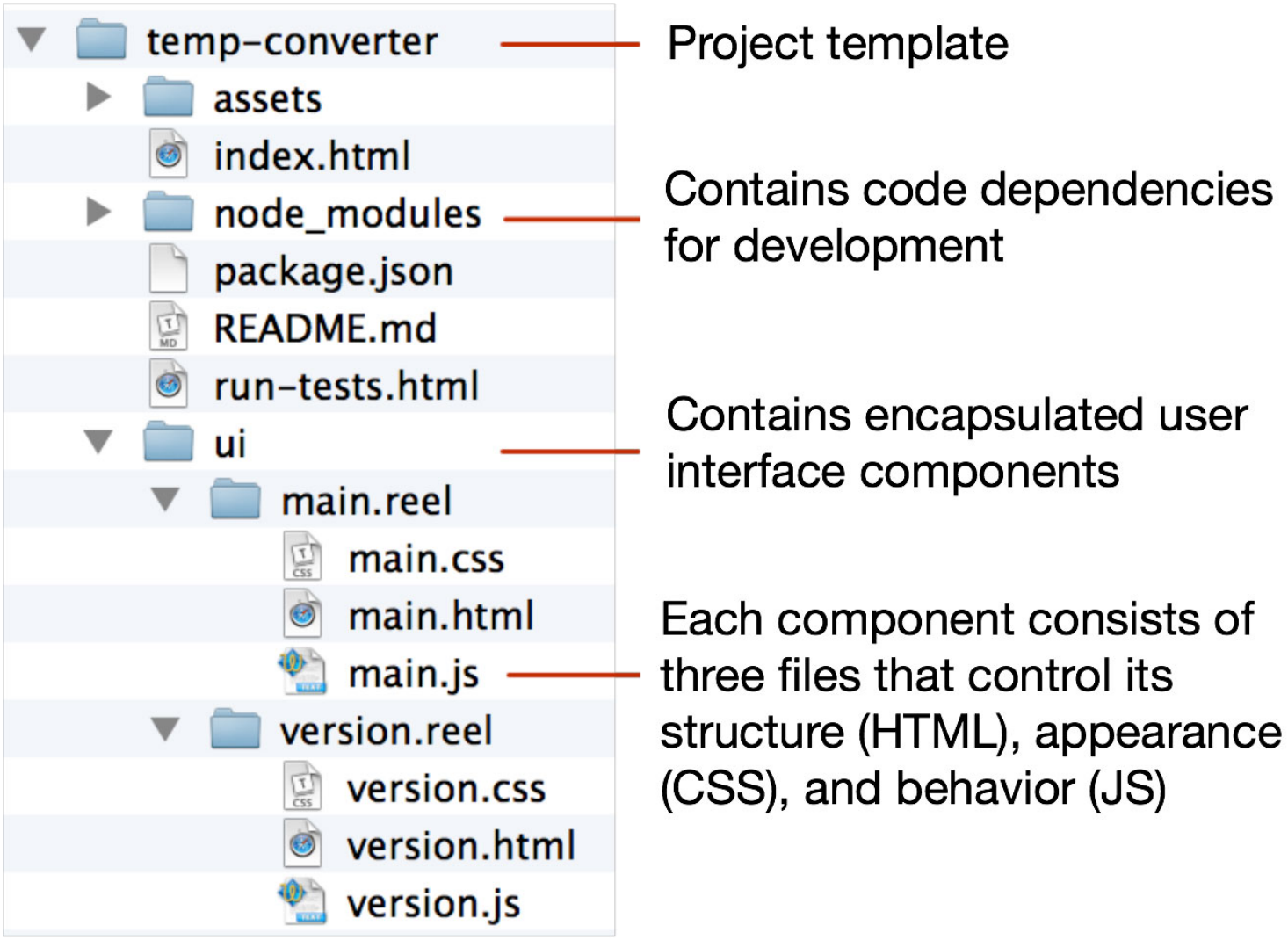


*图 **2.** 用户界面组件存放在项目的ui文件夹中。*

在你构建MontageJS应用的时候，通过修改ui文件夹中组件的HTML文件(在MontageJS中我们称为AKA模版)来改变布局，如果要改变组件的样式，你可以修改组件的CSS文件。

图3所示是组件在应用中对应的界面层次结构。 Main.reel是应用的主界面。 就像一个网站的首页一样，或者你也可以把它想象成一个单页面应用的入口: 它可以包含任意数量的子组件来构建应用。 Converter组件封装了应用的全部功能。底部的版本组件显示当前使用的MontageJS版本信息(可以很容易地从应用中移除)。

> **备注:** 虽然你可以只通过Main一个组件来完成你的应用，但是我们建议你还是拆分为单独的几个组件(就像你把一个网页拆分为几个独立的页面一样) - 这样我们就可以充分地利用MontageJS提供的功能，包括模块化的结构，封装和可重用的组件。

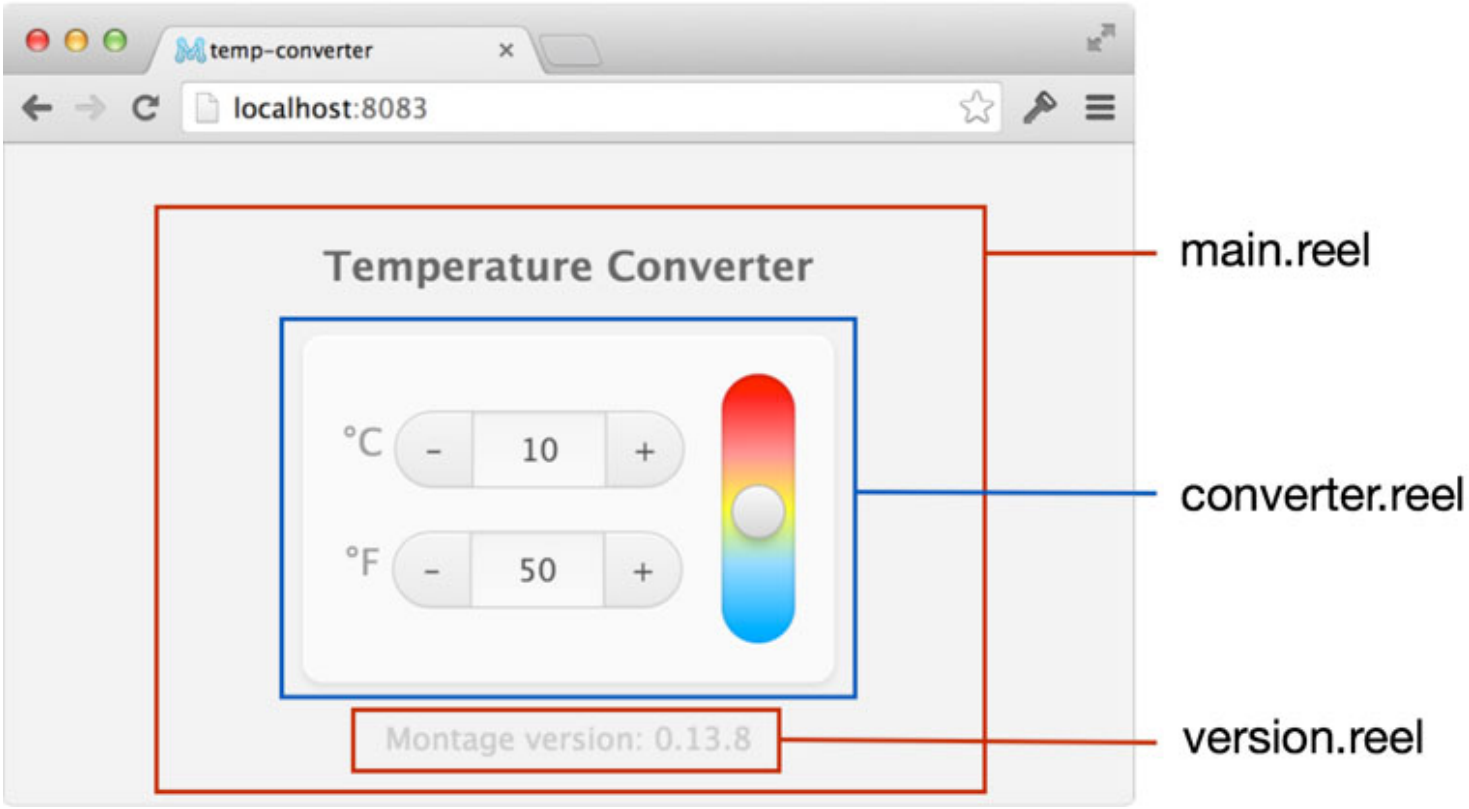


*图 **3.** 用来构建温度转换器的组件。*

新建一个converter组件

按照以下步骤在项目中添加一个新的组件:

1. 在命令行窗口，输入:

   ```
   $ minit create:component -n converter
   ```

   以上命令将在ui文件夹中新建一个converter.reel组件。 要在应用中使用这个组件，你需要在Main组件中声明它。

2. 在项目目录中，打开ui/main.reel/main.html。

3. 在 `<body>` 标签中， `<div data-montage-id="montageVersion"></div>` 之后，添加下面的代码片段:

   ```
   <h1>Temperature Converter</h1>
   <div data-montage-id="tempConverter"></div>
   ```

data-montage-id 自定义数据属性是用来标识你想控制DOM中哪个元素的功能。 控制对象定义在HTML文档头部

的script标签中。

1. 在 `<script>` 标签中，owner对象之后(montageVersion对象之前)，添加下面的代码片段(注意结尾的逗号; 你需要它来分开每个对象否则应用不能正常运行)

```
"tempConverter": {
    "prototype": "ui/converter.reel",
    "properties": {
        "element": {"#": "tempConverter"}
    }
},
```

以上定义了一个名称为tempConverter的Converter组件实例(`"prototype": "ui/converter.reel"`)。它的element属性对应你在上一个步骤中添加的HTML元素
(`<div data-montage-id="tempConverter"></div>`)。

1. 保存修改然后刷新你的浏览器。

如果一切顺利，你将看到应用的标题和Montage版本信息(如图4)。因为你还没有在Converter组件中指定任何的内容，所以现在它还是不可见的。(如果你只看到一个空白的页面，请检查 `<script>` 中的对象是否用逗号正确分开。)

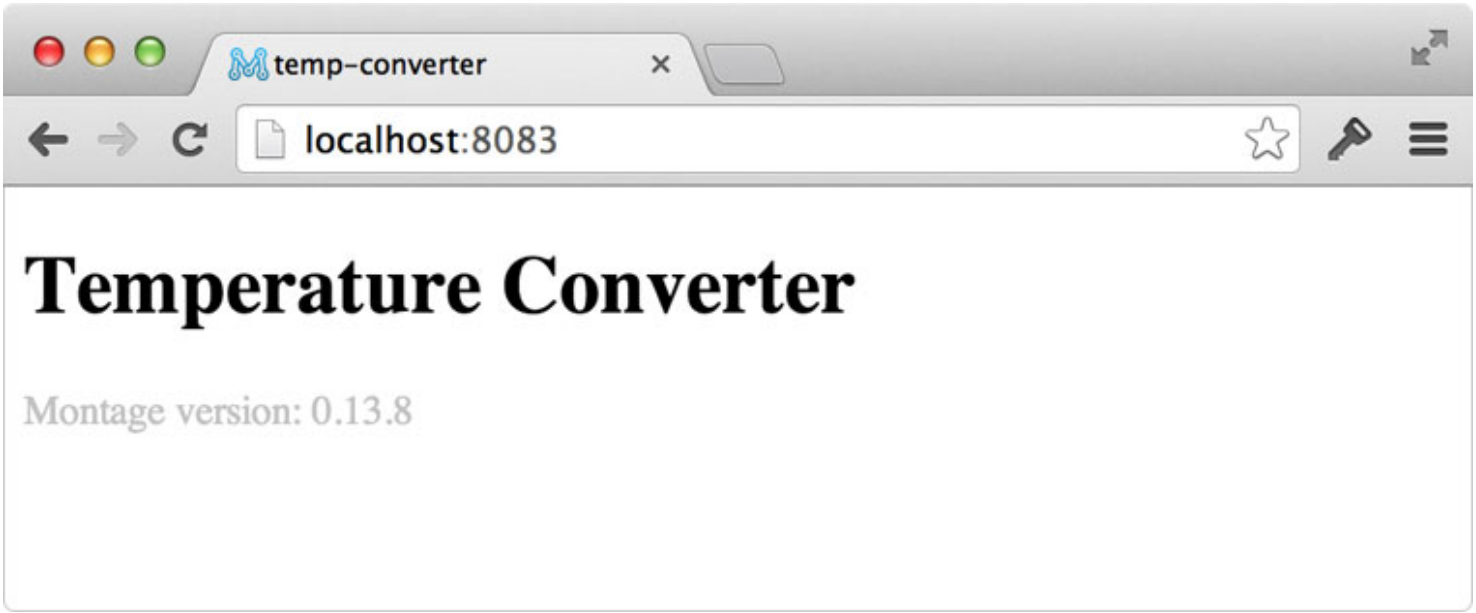


*图 **4.** 应用的Main组件，Main包含title和MontageJS版本信息*

添加标记

你将要构建的应用包含四个元素-一个标题，两个数字输入框，和一个滑块，这些你都需要在HTML中声明。你已

经在main.html中声明了标题，接下来你需要在converter.html中声明输入框和滑块。

1. 打开ui/converter.reel/converter.html。
2. 把标签中的HTML替换为以下的代码片段:

```
<div data-montage-id="converter" class="Converter">
    <div>
    <fieldset>
        <div>&deg;C
           <input type="number"/>
        </div>
        <div>&deg;F
           <input type="number"/>
        </div>
    </fieldset>
    </div>
    <fieldset>
       <input type="range"/>
    </fieldset>
</div>
```

3. 保存修改然后刷新你的浏览器。

在页面中你应该可以看到两个输入框和一个滑块组件(如图5)。

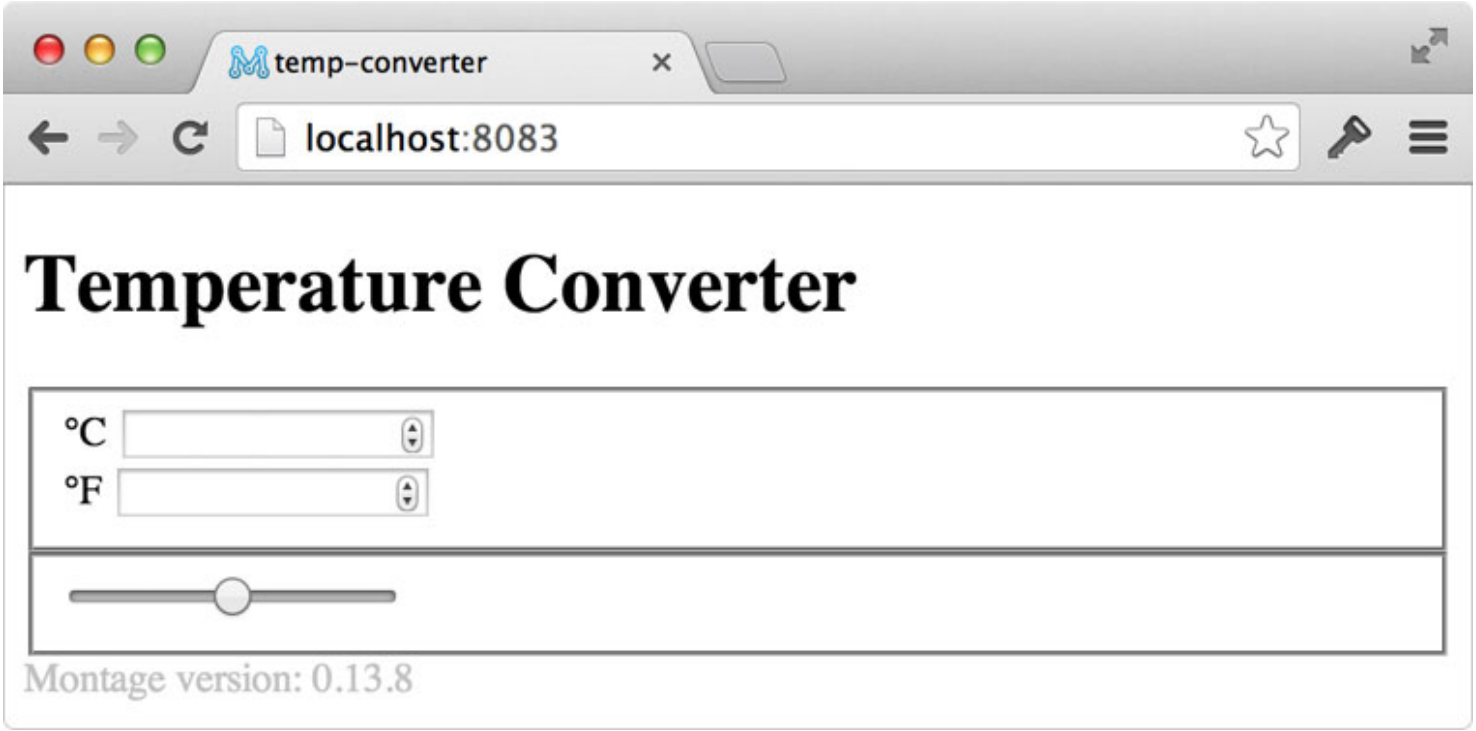


*图 **5.** 使用浏览器默认样式绘制的应用元素。*

下一步，你将使用MontageJS来修改这些HTML元素的功能和布局。

定义模板

首先，修改你想控制的HTML元素的data-montage-id属性。

1. 在ui/converter.reel/converter.html的标签中，把原来的内容替换成下面的代码片段:

```
<div data-montage-id="converter" class="Converter">
    <div>
        <fieldset>
            <div>&deg;C
               <input data-montage-id="celsius"/>
            </div>
            <div>&deg;F
               <input data-montage-id="fahrenheit"/>
            </div>
        </fieldset>
    </div>
    <fieldset>
       <input data-montage-id="thermometer" type="range" />
    </fieldset>
</div>
```

下一步，添加一个控制这个HTML元素的组件。

1. 把 `<script>` 标签中内的内容替换为下面的代码片段:

```
{
    "owner": {
        "properties": {
            "element": {"#": "converter"}
        }
    },

    "celsiusNumberfield": {
        "prototype": "digit/ui/number-field.reel",
        "properties": {
            "element": {"#": "celsius"}
         }
    },

    "fahrenheitNumberfield": {
        "prototype": "digit/ui/number-field.reel",
        "properties": {
            "element": {"#": "fahrenheit"}
        }
    },

    "thermometer": {
        "prototype": "digit/ui/slider.reel",
        "properties": {
            "element": {"#": "thermometer"},
            "axis": "vertical"
        }
    }
}
```

提示:

- 标签 `celsiusNumberfield`，`fahrenheitNumberfield` 和 `thermometer` 是控制HTML元素对应的组件的标识符。
- `prototype` 定义组件对应的目录位置（比如这里使用的就是Montage内置的Digit组件库里面的组件）
- `properties` 定义传给当前组件的参数。
- `element` 属性定义需要控制的DOM元素，通过DOM元素的 `data-montage-id` 属性连接起来，比如这里的 `celsius`，`fahrenheit` 和 `thermometer`.
- `axis` 属性指定滑块组件是纵向布局的。

2. 保存修改然后刷新浏览器。

你已经成功配置了data-montage-id属性，现在就会按照Digit组件库的默认样式绘制页面(如图6)

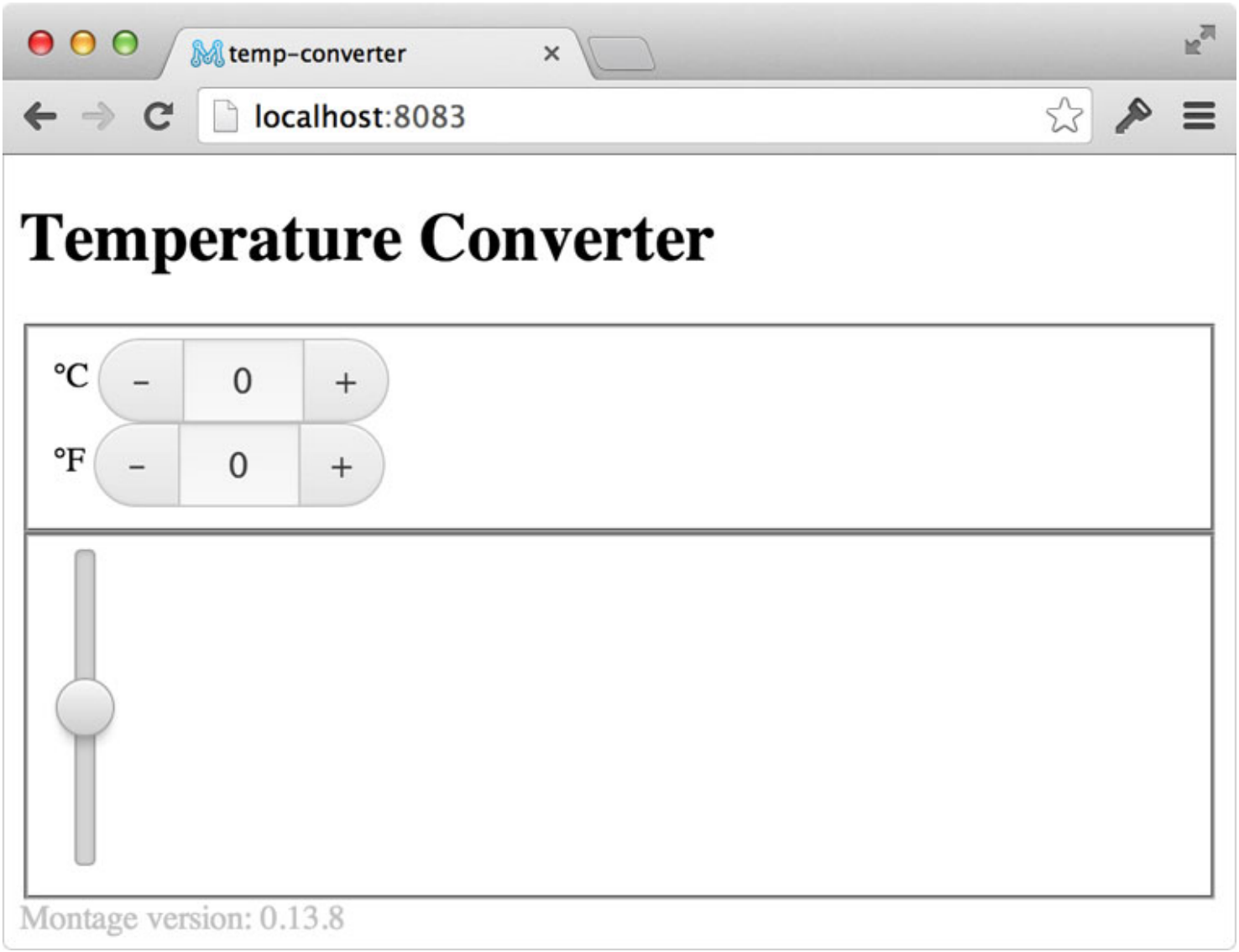


*图 **6.** 继承Digit组件库的元素。*

Next, you will bind together the properties of the input fields and slider.

下一步，你需要把输入框和滑块的值绑定在一起。

添加绑定

MontageJS使用FRB实现绑定，通过一些定义把对象互相绑定在一起。

1. 为了快速的完成教程，把 `<script>` 标签中的内容替换为下面的代码片段。

```
{
    "owner": {
        "properties": {
            "element": {"#": "converter"}
        }
    },

    "celsiusNumberfield": {
        "prototype": "digit/ui/number-field.reel",
        "properties": {
            "element": {"#": "celsius"}
        },
        "bindings": {
            "value": {"<->": "(+@fahrenheitNumberfield.value - 32) / 1.8"}
        }
    },

    "fahrenheitNumberfield": {
        "prototype": "digit/ui/number-field.reel",
        "properties": {
            "element": {"#": "fahrenheit"},
            "value": "32"
        }
    },

    "thermometer": {
        "prototype": "digit/ui/slider.reel",
        "properties": {
            "element": {"#": "thermometer"},
            "axis": "vertical"
        },
        "bindings": {
            "value": {"<->": "@fahrenheitNumberfield.value"}
        }
    }
}
```

提示:

- Fahrenheit的默认值设置为32；这是温度转换器在浏览器中加载之后的状态。
- 双向绑定(<->)的建立是相互之间的:
  - 对象celsiusNumberfield的值(`(+@fahrenheitTextfield.value - 32) / 1.8`) 绑定到对象fahrenheitNumberfield的值.
  - thermometer绑定到fahrenheitNumberfield 的值

现在当你修改任何一个组件的值之后其它也会相应的做出改变。你可以试一下。在Celsius输入一个值，用鼠标的滚轮增加或者减少Fahrenheit的值，或者左右拖动滑块。 ##美化应用

现在为止，应用已经按照预想的工作了但是跟设计(如图1)看起来还不太一样。我们可以轻松地通过添加一下CSS规则来实现这种样式。

美化Converter组件

首先，你需要在组件的HTML文档中指定CSS的类名称。

1. 打开ui/converter.reel/converter.html，把标签中的内容替换为下面的代码片段：

```
<div data-montage-id="converter" class="Converter">
    <fieldset class="Numbers">
        <div class="Label">&deg;C
           <input data-montage-id="celsius">
        </div>
        <div class="Label">&deg;F
           <input data-montage-id="fahrenheit">
        </div>
    </fieldset>
    <fieldset class="Slider">
       <input data-montage-id="thermometer" class="Slider-handle" type="range" mi
n="-13" max="122">
    </fieldset>
</div>
```

   提示：

   - 为输入框(Numbers)，标签 (Label)，滑块（Slider），和滑块按钮（Slider-handle）添加控制布局和显示的样式名称。
   - 设置Fahrenheit允许的最小（-13）和最大值（122）- 滑块组件和华氏温度组件的值是双向绑定的，所以这个范围设置也会被华氏温度组件应用。

   下一步你需要在组件的样式表中添加CSS规则。

2. 打开MontageJS项目的ui/converter.reel/converter.css文件。

3. 把下面的规则复制到这个文件中：

```
.Converter {
    margin: 20px auto;
    padding: 20px;
```

```
    width: 274px;
    border-radius: 10px;
    background-color: hsl(0,0%,98%);
    box-shadow: inset 0px 1px 2px 1px hsla(0,0%,100%,1), 0px 2px 5px hsla(0,0%,0%
,.1);
}

.Converter:after {
    content: "";
    display: block;
    clear: both;
}

.Numbers {
    float: left;
    border: none;
    margin: 0;
    padding: 0;
}

.Label {
    margin: 15px 0;
    line-height: 40px;
    font-size: 1.2em;
}

.Label .digit-NumberField-input {
    width: 70px;
    vertical-align: middle;
}

.Slider {
    float: right;
    margin: 0;
    padding: 8px 4px;
    border-radius: 100px;
    border: none;
    box-shadow: inset 0px 1px 3px hsla(0,0%,0%,.3),
                0 2px 0 hsla(0,0%,100%,1);
    background: -webkit-linear-gradient(bottom,
                hsl(200,100%,50%),
                hsl(200,100%,80%) 30%,
                hsl(60,100%,65%) 50%,
                hsl(0,100%,80%) 70%,
                hsl(0,100%,50%) );
```

```
    background: linear-gradient(to top,
                hsl(200,100%,50%),
                hsl(200,100%,80%) 30%,
                hsl(60,100%,65%) 50%,
                hsl(0,100%,80%) 70%,
                hsl(0,100%,50%) );
}

.Slider-handle.digit-Slider.montage-Slider--vertical {
    height: 120px;
}

.Slider-handle.digit-Slider {
    background-color: transparent;
    border-color: transparent;
    box-shadow: none;
}
```

4. 保存修改然后刷新浏览器。

现在为止你的应用看起来应该如图7。

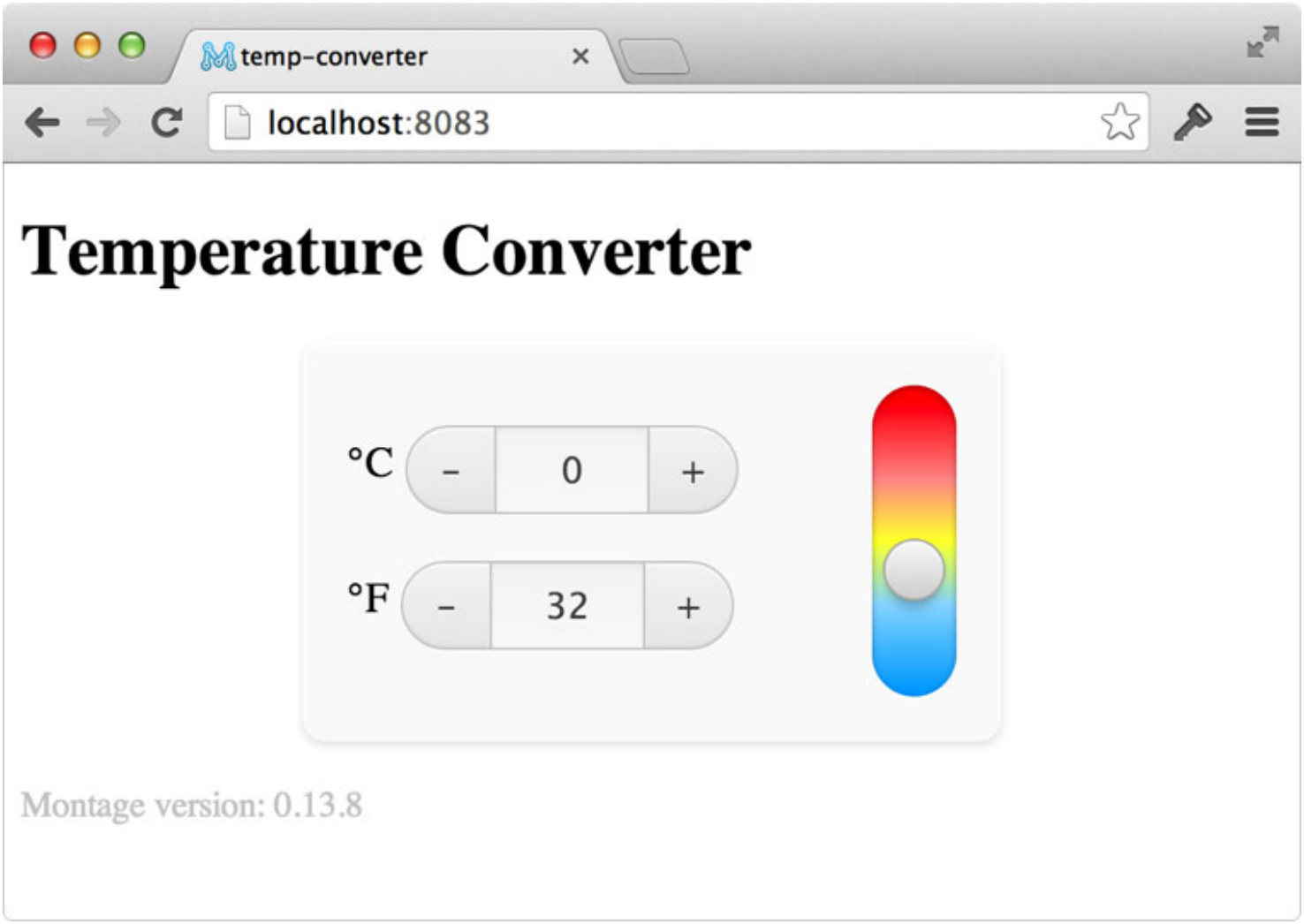


图 7. 已经添加样式的温度转换器应用。

整个应用差不多已经全部完成了；仅剩最后一个步骤。

准备部署

在前面我们已经提到过，MontageJS应用分为开发阶段和产品阶段。当你的应用已经完成之后，你需要部署它。这个时候就需要使用MontageJS的优化工具mop。

Mop是一个使用简单的命令行工具，使用它可以把一个臃肿的开发阶段应用转换成一个优化的可发布版本。在mop的处理过程中也会最小化你的代码，这将有助于应用的加载速度。

Mop不包含在MontageJS默认安装中；你需要单独安装它。下面就是具体步骤：

1. 打开命令行窗口输入：

```
$ sudo npm install -g mop
```

Mop的安装是全局的，所以你可以在当前的项目中使用也可以在以后的项目中使用

> **注意:** 需要使用sudo执行npm命令,这样才能够让map命令行工具全局可用。

2. 切换 (`cd`) 到temp-converter的目录。
3. 在命令行窗口输入mop然后回车。

   Mop分析代码的依赖，标识哪些模块被项目引用，然后会在当前项目的目录中添加一个builds文件夹，在这个builds文件夹中包含你应用的最小化版本。
4. 双击builds打开链接的文件夹，然后双击index.html。

   温度转换器应用将会在浏览器中被打开，这个时候我们就不再需要minit作为web服务器了（注意一下浏览器地址栏的URL）。

要部署最终的项目，你只需要把builds链接到的目录中的内容全部复制到你的web服务器。

总结。你已经构建了一个简单的MontageJS应用，然后还通过部署优化了源代码。在这个过程中你应该已经发现使用MontageJS开发的很多好处；通过声明式的风格让你只需要几行代码就可以完成复杂的界面要求； 模块化可以让一个复杂应用的代码和组件容易管理；清晰的功能分割能够让开发者和设计者在一个项目中轻松协同工作。

Yet, you've barely scratched the surface of what you can do with MontageJS.

不过，目前为止你也只是使用了MontageJS提供的很小一部分功能而已。

下一阶段

- 学习更多的关于如何使用MontageJS，查看更多的教程和示例代码文档。
- 查看MontageJS的组件库，打开主题。
- 查看MontageJS推荐的命名规范，打开命名规范。
- 了解更多关于如何使用mop和builds中的内容，打开使用64Bit mop。