**企业网站**

讲此章内容, 以第二章为基础, 第三章掌握不住的学生不让其掌握了.

讲此章节, 给学生分析需求, 之前需求文档.

**网页内容基本构造部分**

* **页头区**：这一部分包含Logo、带下拉菜单的主导航、二级和实用链接导航，以及登录和注册选项。
* **主内容区**：这一部分布局复杂，至少三栏。
* **页脚区**：包含多栏链接和信息。

这样，我们需要做以下这些事。

1. 完成复杂的页头区，包括Logo、导航以及右上角的实用导航（桌面视口）。

2. 在较窄的视口中，实用导航只显示为图标，与折叠后的响应式导航条并列。

3. 要实现企业风格的配色方案。

4. 调整桌面版和响应式导航条。

5. 为主内容区和页脚区设置复杂的多栏布局。

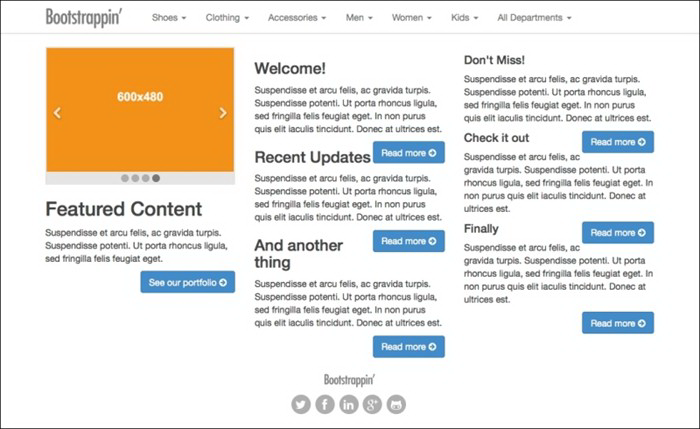
**4.1　准备启动文件**

* Bootstrap LESS和JavaScript文件，分别位于下列文件夹：
  + less/bootstrap：Bootstrap的LESS文件
  + js/bootstrap：Bootstrap的个别插件
  + js/plugins.js：Bootstrap压缩后的插件
* HTML5 Boilerplate及下列文件：
  + 基本的结构化标记文件index.html
  + js/vendor/modernizr-2.6.2.min.js
  + js/vendor/query-1.10.2.min.js
* 保证兼容Internet Explorer 8的repond.js：
  + js/vendor/respond.js
* Font Awesome字体图标，包括：
  + fonts文件夹中的图标字体
  + less/font-awesome文件夹中的LESS文件

除了以上重要的资源之外，我们还在构造第2章的网站时添加过一些LESS文件，可以在less文件夹中找到它们。

* \_\_main.less：基于bootstrap.less,，导入了位于less/bootstrap中的Bootstrap的LESS文件、Font Awesome字体图标和我们自定义的LESS文件。
* \_carousel.less：基于Bootstrap的carousel.less，自定义了传送带的内边距、背景和指示图标。
* \_footer.less：包含Logo及社交媒体图标的布局和设计样式。
* \_navbar.less：基于Bootstrap的navbar.less，调整了.navbar-brand类的样式，以使导航条中的Logo位置合适。
* \_page-contents.less：其中的样式确保了每一栏中的浮动按钮在呈现为单栏的情况下相互清除。
* \_variables.less：基于Bootstrap和variables.less，针对导航条和传送带自定义了灰颜色、新增了变量。

index.html文件提供的内容，



下面说明一下相关的特性。

* 导航条很复杂，有7项，每一项都有下拉菜单。
* 三栏中的第一栏开头是一个传送带，后面是一个标题、一个段落和一个按钮。
* 第二和第三栏同样都包含标题和段落，以及“Read more ->”按钮。
* 页脚包含Logo和社交媒体图标。

使用了一个JavaScript插件holder.js，目的是为传送带动态生成占位图片。查看源代码，就会发现在页面底部plugins.js插件之前，包含了这个holder.js脚本：

<!-- Holder.js for project development only -->

<script src="js/vendor/holder.js"></script>

加载了holder.js之后，就可以方便地把holder.js作为任意图片的来源。然后使用伪URL指定大小、颜色和填充文本，比如：

<img src="holder.js/600x480/auto/vine/textmode:literal" alt="Holder Image">

更多信息可以参考holder.js的文档：<https://github.com/imsky/holder>。

有了这些元素，以及Bootstrap内置的样式和行为，我们的起点就非常高了。下面我们就来实现所有特性。

首先，我们重新定位导航条，实现页头区的设计。

**4.2　页头区**

从上到下，先来实现复杂的页头区，包括如下特性。

* 在桌面浏览器及较大视口中，让站点Logo显示在导航条之上。
* 包含菜单项的导航条，每个菜单项又都包含下拉菜单。
* 实用导航区。
* 带用户名和密码字段的登录表单。
* 注册选项。

以下是桌面浏览器中的目标结果：



窄视口中的目标结果如下：

{%}

先从放置站点Logo开始。

**4.2.1　把Logo放到导航条上方**

在这个设计方案里， Logo可能出现在两个地方，视情况而定：

* 在桌面和宽屏幕中，显示在导航条上方；
* 在平板和手机屏幕中，显示在响应式导航条内部。

利用Bootstrap的响应式实用类，这两点我们都可以做到！方法如下。

1. 在编辑器中打开index.html。

2. 找到导航条，复制navbar-brand的链接与图片：

<a class="navbar-brand" href="index.html"><img src="img/logo.png" alt="Bootstrappin'" width="120"></a>

3. 然后粘贴到导航条上方，在<header role="banner">标签和<nav role= "navigation" class="navbar navbar-default">标签之间。

4. 把这个Logo用<div class="container">...</div>包装起来，使其被限制在Bootstrap居中的网格内部。

5. 编辑Logo的链接，将其类名由navbar-brand改为banner-brand。然后把图片宽度改为180。

　Logo图片其实很大，有900px宽。为了让它在视网膜屏上显示清晰，我们已经使用width属性（使用CSS规则也行）把它缩小到了120px宽，以便提供足够的像素密度。

结果代码如下所示：

<header role="banner">

<div class="container">

<a class="**banner-brand**" href="index.html"><img src="img/logo.png" alt=

"Bootstrappin'" width="**180**"></a>

</div><!-- /.container -->

<nav role="navigation" class="navbar navbar-default">

保存修改，然后在浏览器中刷新页面。应该在导航条上面看到新的Logo。



来调整Logo，让它只在必要的时候显示。

在\_variables.less中，确认变量@grid-float-breakpoint的值。可以通过搜索找到这个变量，less/bootstrap/variables.less文件中这个变量的默认值如下：

// Point at which the navbar stops collapsing

@grid-float-breakpoint: @screen-sm-min;

这个变量决定了导航条在窄视口中折叠，在宽视口中展开。在设计中，考虑到导航的复杂性，需要在接近的下一个较宽的断点折叠导航条。因此，需要把变量的值设置为@screen-md-min。

// Point at which the navbar stops collapsing

@grid-float-breakpoint: **@screen-md-min;**

设置完这个变量后，要考虑让banner-brand只在中、大型视口中显示，而让navbar-brand只在小和超小型视口中显示。Bootstrap为此提供了一组响应式实用类，具体可以参考在线文档：[http://getbootstrap.com/css/#responsive-utilities](http://getbootstrap.com/css/" \l "responsive-utilities" \t "_blank)。

下面就按照需求来应用这些类。

1. 把visible-md visible-lg添加到banner-brand类后面：

<a class="banner-brand visible-md visible-lg"

href="index.html"><img src="img/logo.png"

alt="Bootstrappin'" width="180"></a>

2. 把visible-xs visible-sm添加到navbar-brand类后面：

<a class="navbar-brand visible-xs visible-sm"

href="index.html"><img src="img/logo.png"

alt="Bootstrappin'" width="120"></a>

保存修改，刷新网页。在中、大型视口中，只会显示banner-brand中的Logo：



在小型和超小型视口中，只会显示navbar-brand中的Logo：

{%}

Bootstrap太棒啦！

**4.2.2　调整导航条**

现在的导航条包含7项，每项又各有子菜单，体现了一个大型复杂网站的需求。

其中下拉菜单的标记直接取自Bootstrap的导航条文档：[http://getbootstrap.com/components/#navbar](http://getbootstrap.com/components/" \l "navbar" \t "_blank)。

如果你查看结果标记，会发现以下特殊的类和属性：

* 父级li元素中的class="dropdown"；
* 链接中的class="dropdown-toggle"；
* 链接中的attribute="data-toggle"；
* 子菜单ul中的class="dropdown-menu"。

以下是结果标记：

<li **class="dropdown**">

<a href="#" **class="dropdown-toggle" data-toggle="dropdown**">Shoes

<b class="caret"></b></a>

<ul **class="dropdown-menu**">

<li><a href="#">Action</a></li>

<li><a href="#">Another action</a></li>

<li><a href="#">Something else here</a></li>

...

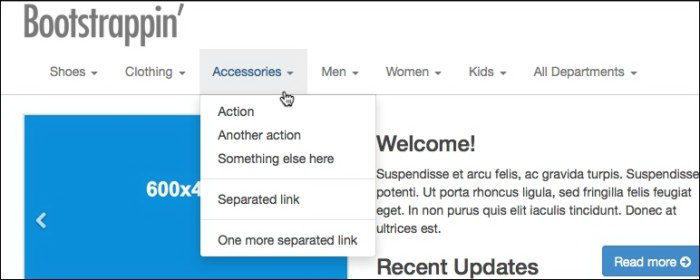
</ul>

</li>

另外还有一个带特殊类的特殊标签，用于显示下拉菜单指示图标：<b class="caret"> </b>。（创建这个指示图标的CSS在less/bootstrap/dropdowns.less中。）

检查plugins.js，确保其中包含了bootstrap- dropdown.js插件。

在LESS、JavaScript和标记就位的情况下，导航条及其下拉菜单应该像下面屏幕截图中一样。（注意，Bootstrap的所有下拉菜单都要通过单击激活。）



熟悉了标记结构，并且确认菜单工作正常之后，接下来要把All Departments菜单挪到导航条的最右端，让它与其他菜单项保持最大距离。

为此，需要把相应的列表项嵌套在它自己的无序列表中。

1. 在All Departments列表项之前，关闭ul class="nav"这个ul标签，用于包含之前的所有列表项。

2. 在All Departments列表项之前，再新建一个ul标签，类名为nav和navbar-nav。添加了这个开始标签后，这个独立的列表项就具备了标准的导航菜单结构。

3. 除了nav和navbar-nav类之外，再添加一个pull-right类，这是Bootstrap的一个实用类，用于把元素浮动到右侧。

以下代码中加粗的部分是新添加的代码，紧随其后的是原来的列表项和链接：

**</ul>**

**<ul class="nav navbar-nav pull-right">**

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown">All

Departments <b class="caret"></b></a>

保存修改并刷新页面，应该可以看到All Departments下拉菜单已经浮动到了导航条的最右端：



一切顺利！下面来添加实用导航。

**4.3　添加实用导航**

我们的设计需要提供几个实用的导航链接，让用户可以登录、注册和查看购物车。

在中大型的视口中，我们把它们放到页头区的右上角，如下图所示：



在较小的屏幕中，则把对应的链接图标显示在折叠后的导航条的最右端，如下图所示：



还是在index.html中，要在页头区添加实用导航项的标记，放在banner-brand元素后面。以下是完整的标记，开头是header标签，新增的代码加粗了：

<header role="banner">

<div class="container">

<a class="banner-brand visible-md visible-lg" href=

"index.html"><img src="img/logo.png" alt="Bootstrappin'"

width="180"></a>

**<div class="utility-nav">**

**<ul>**

**<li><a href="#" title="Login or Register"><i class="icon fa fa-user fa-lg"></i> Log In or Register</a></li>**

**<li><a href="#" title="View Cart"><i class="icon fa fa-shopping-cart fa-lg"></i> View Cart</a></li>**

**</ul>**

**</div><!-- /.utility-nav -->**

</div><!-- /.container -->

关于以上标记，还要说明两点。

* 类utility-nav只是为了方便使用，它不是Bootstrap特有的类，也没有什么样式。
* 这里已经通过fa-user和fa-shopping-cart类添加了Font Awesome的用户和购物车图标，并通过fa-lg类把它们的尺寸增大了33%。关于增大Font Awesome图标的详细说明，请参见它的文档：[http://fontawesome.io/examples/#larger](http://fontawesome.io/examples/" \l "larger" \t "_blank)。

保存修改并刷新页面，应该看到新添加的utility-nav出现在了banner-brand Logo下方，如下图所示：



接下来，对布局进行相对位置的调整，也就是要应用一些自定义的样式。为此，要针对页头区新建一个文件，步骤如下。

1. 创建一个新文件并命名为\_banner.less，保存到less文件夹中，与其他自定义的LESS文件放在一起。

2. 把\_banner.less添加到\_\_main.less的导入文件列表中。

// Other custom files

**@import "\_banner.less"; // added**

3. 在\_banner.less中，先给出一个帮助性的注释。把.utility-nav设置为绝对定位到右上角，而且是在header[role="banner"]的上下文中应用样式：

//// Banner Area Styles

//

header[role="banner"] {

.utility-nav {

position: absolute;

top: 0;

right: 0;

}

}

4. 下面我们要调整一些细节。

* 为.banner-brand类添加上内边距，以增加页头区的高度。
* 将页头区container的定位方式设置为relative，以使它包含绝对定位的utility-nav元素。
* 删除无序列表的项目符号。
* 向左浮动列表项。
* 将链接显示为inline-block并添加内边距。
* 删除悬停时的下划线。

完成上述调整的样式规则如下：

header[role="banner"] {

.banner-brand {

padding-top: 40px;

}

> .container {

position: relative;

}

.utility-nav {

position: absolute;

top: 0;

right: 0;

> ul {

list-style: none;

> li {

float: left;

> a {

display: inline-block;

padding: 8px 12px;

&:hover {

text-decoration: none;

}

}

}

}

}

}

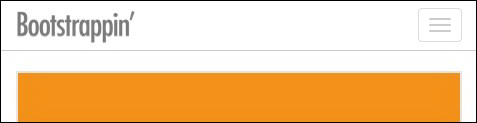
保存修改并编译。把浏览器窗口调整到桌面窗口大小，然后刷新。应该能看到utility-nav元素出现在了页头区的右上角位置。



这些调整适合中大型的视口。下面我们针对折叠后的响应式导航条来添加样式。

**4.4　调整响应式导航**

在小屏幕中，导航条折叠后utility-nav会出现问题。最明显的问题是它会消失不见：



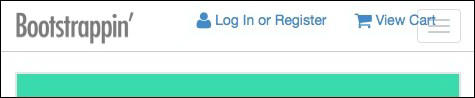
要想让utility-nav重见天日，必须给它设置一个比导航条更大的z-index，前者在\_variables.less中被设置为1000。我们可以在\_banner.less中，把.utility-nav的z-index设置为1999。

.utility-nav {

...

z-index: 1999;

于是，实用导航就会出现了：



接下来的问题就是它会遮挡navbar-toggle按钮。为此，我们要把这个切换按钮转移到导航条的左侧，步骤如下。

1. 在编辑器中打开less/\_navbar.less。

2. 搜索注释// Navbar toggle，编辑这条注释正下方的.navbar-toggle选择符，把浮动值由right改为left，把margin-right改为margin-left：

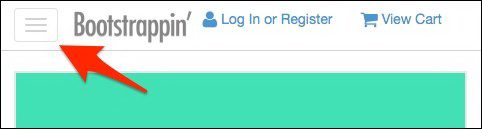
.navbar-toggle {

position: relative;

**float: left; // edited**

**margin-left: @navbar-padding-horizontal; // edited**

保存并编辑上面的修改，可以看到切换按钮已经转移到了折叠后导航条的左端，如下图所示。



现在解决过分拥挤的问题，也就是要对除屏幕阅读器之外的其他设备隐藏链接文本。在折叠后的导航条中，图标本身就足以传达意图了，何况还可以把图标弄得更大一些。

1. 在index.html中，用span标签包围utility-nav中每个链接的文本：

<li><a href="#" title="Login or Register"><i class="icon fa

fa-user fa-lg"></i><span>Log In or

Register</span></a></li>

<li><a href="#" title="View Cart"><i class="icon fa

fa-shopping-cart fa-lg"></i><span>

View Cart</span></a></li>

这样可以为后面进一步调整样式提供抓手。

2. 在\_banner.less中添加针对这些span标签的媒体查询。在使用LESS的情况下，可以精确地嵌套媒体查询。在此要使用@grid-float-breakpoint变量，把max-width查询设置为@grid-float-breakpoint - 1，因为这个变量的值意味着在它那么宽时，导航条就会从折叠变成扩展状态。在这个媒体查询中，使用实用类sr-only作为混入，对除屏幕阅读器之外的所有设备隐藏文本。（参见这个类的文档：[http://getbootstrap.com/css/#helper-classes-screen-readers](http://getbootstrap.com/css/" \l "helper-classes-screen-readers" \t "_blank)。）代码片段如下：

.utility-nav {

...

...

> a {

...

**@media (max-width: (@grid-float-breakpoint - 1)) {**

**span {**

**.sr-only();**

}

}

}

}

这样就隐藏了span标签中的文本，屏幕上将只剩图标！

3. 再增大图标尺寸，并在垂直方向增加一些行高。同样还在这个媒体查询中写样式：

@media (max-width: @grid-float-breakpoint - 1) {

span {

.sr-only();

}

**.icon {**

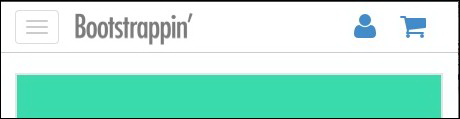
**font-size: 2em;**

**line-height: 1.2;**

}

}

保存，编译，刷新。应该看到下图所示的结果：



再放大、缩小浏览器窗口，反复经过断点宽度，看看整个页头区来回变换的效果是不是很平滑。

就我而言，能有这么一个框架让我如此高效地实现这种流畅的响应式界面，我真要谢天谢地了。

**4.5　调整配色**

我们网站现在的配色是标准的企业网站颜色：蓝、红、灰。下面我们把这些颜色放到变量里。(再让陈金波分享一下企业网站配色)

1. 在编辑器中打开\_variables.less，从一开始的变量开始。

2. 先看一下目前灰色变量覆盖的范围。

// Grays

// -------------------------

@gray-darker: #222; // edited

@gray-dark: #454545; // edited

@gray: #777; // edited

@gray-light: #aeaeae; // edited

@gray-lighter: #ccc; // edited

@gray-lightest: #ededed; // edited

@off-white: #fafafa; // edited

3. 在灰色变量下方，再添加品牌色。修改@brand-primary，新增红色的@brand-feature：

@brand-primary: #3e7dbd; // edited blue

@brand-feature: #c60004; // added new red

4. 下面调整链接的悬停颜色，使其比@brand-primary稍浅（而不是稍深），否则就太深了：

// Links

// -------------------------

@link-color: @brand-primary;

@link-hover-color: lighten(@link-color, 15%);

设置好这些颜色变量后，就可以着手调整导航条的配色了。

**4.6　调整折叠后的导航条配色**

还在\_variables.less中，搜索// Navbar，在这里可以看到导航条用到的变量。这里指定的大多数标准值既对折叠后的响应式导航条有效，也对宽屏幕下扩展的导航条有效。

我们希望折叠后响应式导航条的背景、文本和链接颜色与默认值基本一致，但在中大型视口中变成蓝色背景、浅色文本。

为此要调整一些变量的默认值，然后再创建一些新变量，只应用给扩展后的导航条。

1. 把@navbar-height的值减小为44px，然后把前面设置的变量应用到合适的地方。再把@navbar-default-color设置为@text-color，把@navbar-default-bg设置为@white。

// Basics of a navbar

@navbar-height: 44px;

...

@navbar-default-color: @text-color;

@navbar-default-bg: #fff;

2. 向下找到导航条链接区，调整颜色让链接与导航条文本颜色一致，并给活动链接添加一点背景色：

// Navbar links

@navbar-default-link-color: @navbar-default-color;

@navbar-default-link-hover-color: @navbar-default-color;

@navbar-default-link-hover-bg: darken(@navbar-default-bg, 5%);

@navbar-default-link-active-color: @navbar-default-color;

@navbar-default-link-active-bg: @navbar-default-link-hover-bg;

3. 再调整navbar-toggle的样式，删除边框和背景，调深导航条：

// Navbar toggle

@navbar-default-toggle-hover-bg: transparent;

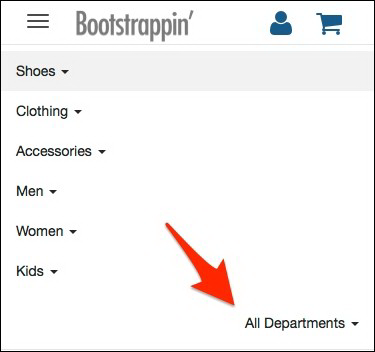
@navbar-default-toggle-icon-bar-bg: @gray;

@navbar-default-toggle-border-color: transparent;

保存，编译，然后刷新浏览器，在窄视口中应该看到下图所示的效果：

{%}

折叠导航条还有两个地方要调整。点击打开下拉导航菜单，会看到浮动到右侧的All Departments链接。



还记得我们把pull-right类放到了All Departments菜单项中了吗？那就是为了让它浮动到扩展后的导航条右侧。但此时此地，我们希望它还在左侧，怎么办？Bootstrap专门有一个类就是为了这个目的！

在index.html中找到包含All Departments的标记，把类pull-right改为navbar-right，如下所示：

<ul class="nav navbar-nav **navbar-right**">

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown">All

Departments <b class="caret"></b></a>

想知道背后都发生了什么，就打开\_navbar.less，搜索.navbar-right。你会看到在一些注释下面有以下代码：

@media (min-width: @grid-float-breakpoint) {

.navbar-left { .pull-left(); }

.navbar-right { .pull-right(); }

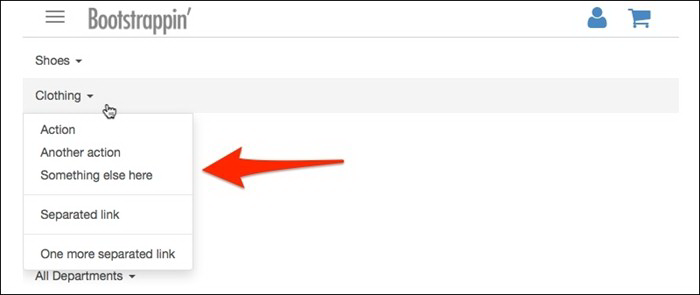
}

这些规则嵌套在一个媒体查询中，只适用于扩展后的导航条，而这些类恰好能满足我们的需要。在应用新的navbar-right类之后，保存index.html，刷新，会发现All Departments在折叠的导航条中会浮动到左侧，而在扩展的导航条中仍然会浮动到右侧。



太厉害了！现在，我们来调整导航条中下拉菜单的行为。从Bootstrap 3.0.2起，下拉菜单项被配置为与折叠后的导航条同宽，但前提是导航条是在最初的@screen-sm-min值为@grid-float-breakpoint的情况下发生折叠。我们的下拉菜单就不是全宽了。要测试，可以把浏览器窗口拖动到@screen-sm range（768～991px），再试试打开下拉菜单。

你会看到下面的结果：



实际上，只要调整一个媒体查询，就可以解决这个问题。

1. 打开\_navbar.less搜索.open.dropdown-menu，会发现这条规则被嵌套在一个媒体查询中：

@media (max-width: @screen-xs-max) {

// Dropdowns get custom display when collapsed

.open.dropdown-menu {

2.要调整这个媒体查询，让它的max-width值等于@grid-float-breakpoint。实际上，就简单地把这个变量放到那儿就行：

@media (max-width: **@grid-float-breakpoint)** {

保存修改，编译文件，然后刷新页面。这次就会看到下拉菜单与屏幕一样宽了：



**调整水平导航条**

在中大型屏幕中，导航条水平排列在Logo下面。我们希望此时的导航条呈现@brand-primary变量中设置的蓝色背景。为此，必须要反转链接和文本的颜色，即由深变浅。我们要使用Bootstrap的inverted-navbar变量和样式。

1. 在\_variables.less中，搜索注释// Inverted navbar。找到后，会发现一些与默认导航条所用类似的变量。我们就要通过它们来给扩展后的导航条应用颜色。

2. 按照如下所示调整变量：

// Inverted navbar

//

// Reset inverted navbar basics

@navbar-inverse-color: **@gray-lightest;**

@navbar-inverse-bg: **@brand-primary;**

@navbar-inverse-border: darken(@navbar-inverse-bg, 10%);

// Inverted navbar links

@navbar-inverse-link-color: **@navbar-inverse-color;**

@navbar-inverse-link-hover-color: **#fff;**

@navbar-inverse-link-hover-bg: **darken(@navbar-inverse-bg, 5%);**

@navbar-inverse-link-active-color:@navbar-inverse-link-hover-color;

@navbar-inverse-link-active-bg: darken(@navbar-inverse-bg, 10%);

调整好这些变量后，只要把它们应用给扩展导航即可。为此得写几行自定义的LESS代码。考虑到这种颜色切换属于页头匹配色的变化，所以我们就把代码写到\_banner.less中。

3. 打开\_banner.less并添加一个新的带注释的区块：

// Apply .navbar-inverse styles to the expanded navbar

@media (min-width: @grid-float-breakpoint) {

.navbar-default {

.navbar-inverse();

}

}

这个媒体查询使用@grid-float- breakpoint变量确定了应用新规则的最小视口宽度。因为我们已经在导航条中添加了navbar-default类，所以可以直接使用这个类作为选择符。混入.navbar-inverse()则把在\_navbar.less中定义的样式.navbar-inverse应用给了这个媒体查询中的导航条。

保存以上修改，编译文件，刷新浏览器。在中大型屏幕中，可以看到导航条的蓝色背景和浅色文本了。



此时此刻，可以看到导航条两端的圆角。我们得把这些样式去掉。为此，打开\_variables.less，搜索变量@navbar-border-radius，然后将其值设置为0：

@navbar-border-radius: 0;

最后，我们把文本转换为大写形式，稍微缩小一点，再加粗。

在\_banner.less中，把如下代码添加到.navbar-inverse()混入之后：

@media (min-width: @grid-float-breakpoint) {

.navbar-default {

.navbar-inverse();

**.navbar-nav > li > a {**

**text-transform: uppercase;**

**font-size: 82%;**

**font-weight: bold;**

**}**

}

}

这样就得到了我们想要的结果：



下图是悬停在一个导航项上的特写：

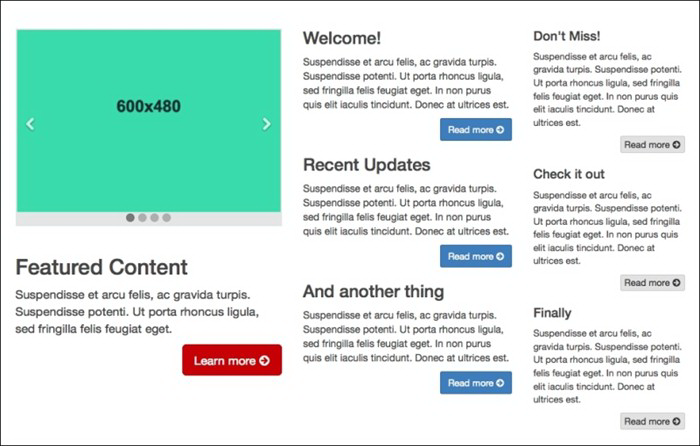


页头和导航条完工了！接下来该轮到页面的主内容区了。

**4.7　设计复杂的响应式布局**

在刚刚结束的客户会面中作出了一个承诺，要把主页内容分成三层，按重要程度排序。

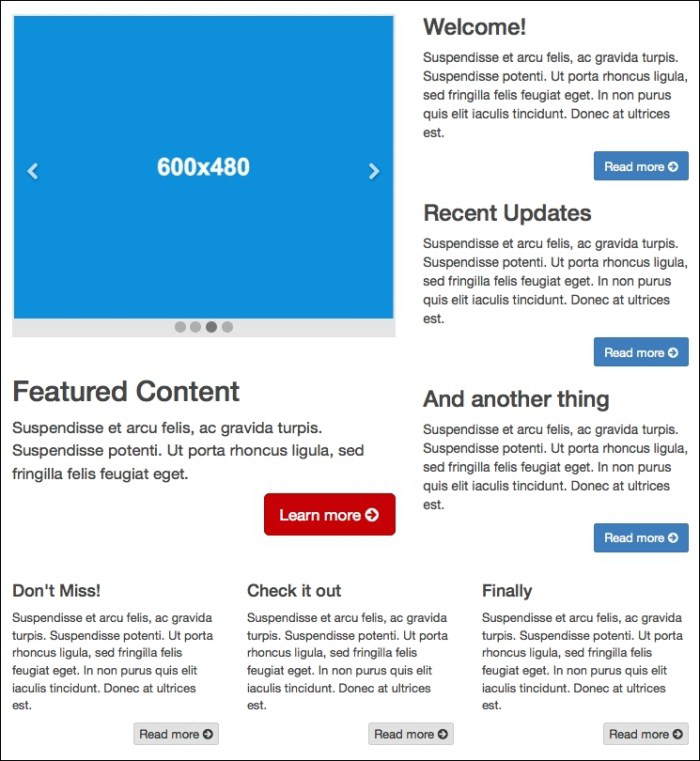
在中大型视口中，所有内容将分布在三栏中，如下图所示：



在较窄的视口中，这些栏将从上到下排成一栏：



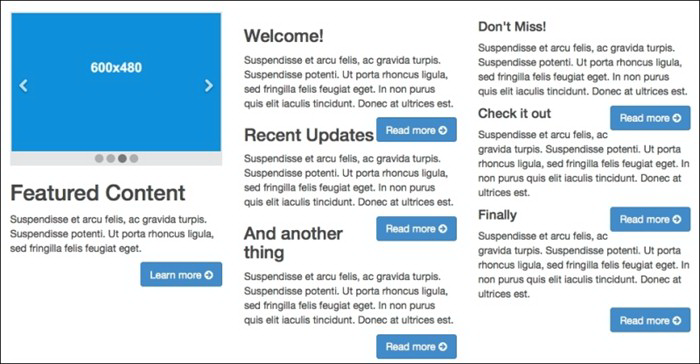
而在平板电脑的视口中，并排的只有两栏，第三栏水平放到它们下面，如下图所示：



作为起点，我们已经有了三个等宽栏的标记。下面就来看一下这些标记，考虑一下怎么实现我们的设计意图。先从中大型屏幕的三栏布局开始。

**4.7.1　调整中、宽布局**

当前，在中宽视口中，三栏是等宽的，而且字体大小、按钮大小，还有颜色都一样。结果就是没有层次感，如下图所示：



要实现内容从视觉上的分层，可以调整栏宽、字体大小、按钮大小，还有颜色。我们先从调整栏宽开始。

1. 在index.html中，搜索包含内容的section标签：

<section class="content-primary col-sm-4">

这里的类col-sm-4表示当前栏是父元素宽度的三分之一，从小视口（764px）及以上宽度开始。

我们想在中大视口（992px及以上）内保留三栏，而且希望第一栏比另两栏宽。

2. 把col-sm-4修改为col-md-5，如下所示：

<section class="content-primary **col-md-5**">

这样就把栏宽设置为了父元素的5/12，并且从中型视口开始应用。

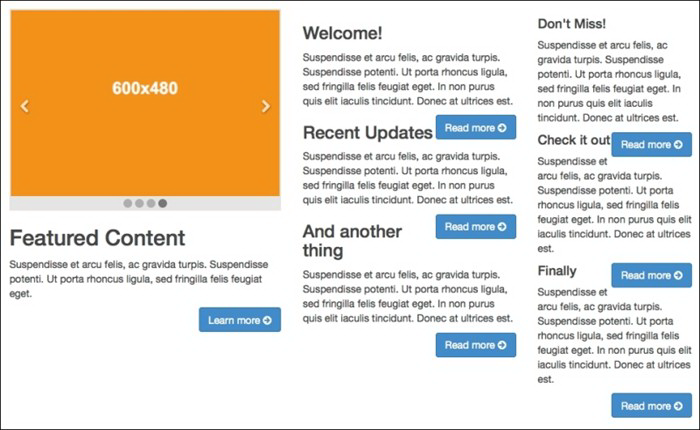
3. 再搜索到后面两栏的开始section标签，将它们的类分别改为col-md-4和col-md-3：

<section class="content-secondary **col-md-4**">

...

<section class="content-tertiary **col-md-3**">

保存，刷新，可以看到以宽度分层的三栏：



中间和第三栏的按钮并没有清除。下一步就来调整这些按钮，还有字体大小。

**4.7.2　调整标题、字体大小和按钮**

我们先来调整标题，以便它们清除自己上方的按钮，目前这些按钮都浮动到了右侧。为此，要用到之前新建的用于管理页面内容细节的文件：\_page-contents.less。

以下是调整步骤。

1. 在\_page-contents.less中，写一个选择符选择嵌套在Bootstrap的分栏类中的h1到h4。这里可以使用CSS2的属性选择符，同时只针对嵌套在类以col-开头的元素内的这些标题：

这样，就可以选中所有可能用到的标题标签，以便清除它们的浮动，再给它们添加一些内边距。

main {

...

[class\*="col-"] {

**h1, h2, h3, h4 {**

**clear: both;**

**padding-top: 20px;**

}

}

}

这样标题之间有了必要的分隔，也让按钮浮动到了相应位置。但这样也在第二和第三栏上方增加了不必要的上内边距。

在下面的屏幕截图中，下方箭头指出的是改进，上方箭头指出的是内边距造成的两栏顶部不齐的问题。



2. 下面来删除每栏最顶部标题的上内边距和上外边距。为此，要使用:first-child选择符，嵌套在标题选择符内。这里使用的&组合符，用于选择这些标题的第一个实例：

h1, h2, h3, h4 {

...

**&:first-child {**

**margin-top: 0;**

**padding-top: 0;**

**}**

}

3. 结果就是去掉了后两栏顶部多余的内外边距：



4. 不过，我们只想在小或较大视口中删除上内、外边距，这时候主页呈现为多栏。显然，应该把上面的样式嵌套在与相应断点对应的媒体查询中。这个断点就是从单栏布局切换到多栏布局的断点。

换句话说，需要把上面的样式嵌套在小及更大的视口中：

**@media (min-width: @screen-sm-min) {**

&:first-child {

margin-top: 0;

padding-top: 0;

}

}

通过把新样式嵌套在上面的媒体查询中，可以保留单栏布局下元素间适当的间距，如下图所示：



完成了上述调整，接下来可以调整按钮和字体大小，从而反映内容层次了。下面就来增大主内容区字体和按钮大小，还有修改颜色。

**4.7.3　增大主栏**

首先来增大主栏内容的字体大小。

1. 在\_variables.less中，搜索@font-size-large变量，将其值修改为：

ceil(@font-size-base \* 1.15);

2. 在\_page-contents.less中，添加如下代码，以利用上一步中设定的字体大小：

.content-primary {

font-size: @font-size-large;

}

保存修改，编译文件，刷新浏览器。应该看到主栏文本的字体增大了！

接下来调整按钮的颜色，这要用到红色的@brand-feature变量，这个变量是在\_variables.less中设定的：

@brand-feature: #c60004;

还需要利用Bootstrap在mixins.less中提供的方便的混入。希望大家抽点时间看看这个文件。打开bootstrap/mixins.less，搜索// Button，可以看到以下面代码开头的混入：

.button-variant(@color; @background; @border) {

这个混入的作用如下。

* 指定按钮字体、背景和边框颜色（分别对应于混入接受的三个参数）。
* 生成悬停、焦点和禁用状态的按钮，调整字体颜色、背景颜色和边框。

如果你有兴趣，还可以看一看Bootstrap在bootstrap/buttons.less中如何使用这个混入，就在注释// Alternate buttons的下面。以下是为默认主按钮生成的样式：

// Alternate buttons

// --------------------------------------------------

.btn-default {

.button-variant(@btn-default-color; @btn-default-bg; @btn-default-border);

}

.btn-primary {

.button-variant(@btn-primary-color; @btn-primary-bg; @btn-primary-border);

}

　以@btn-default-和@btn-primary-开头的变量是在variables.less中定义的。

按照同样的模式，只需简单四步即可生成我们自己的自定义功能按钮。

1. 首先，准备一组新的按钮变量。在\_variables.less中，// Buttons下面，复制三个@btn-primary-变量，将-primary-改为-feature-，并使用@brand-feature作为背景颜色：

@btn-feature-color: #fff;

@btn-feature-bg: @brand-feature;

@btn-feature-border: darken

(@btn-feature-bg, 5%);

2. 然后，创建一个文件来保存自定义按钮的样式。新建\_buttons-custom.less并根据bootstrap/buttons.less中的混入写一个下面这样的混入调用：

.btn-feature {

.button-variant(@btn-feature-color; @btn-feature-bg; @btn-feature-border);

}

3. 保存文件，并将其添加到\_\_main.less中的导入列表：

@import "bootstrap/buttons.less";

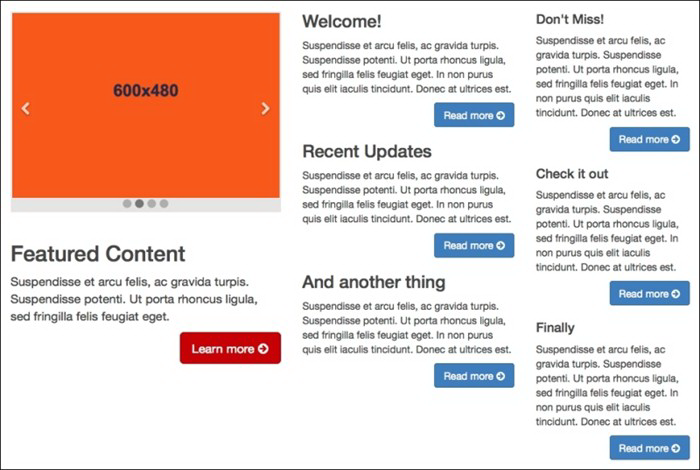
**@import "\_buttons-custom.less";** // added

4. 在index.html中，将类名btn-primary改为btn-feature。完事之后，还要把按钮变大一些，因此再添加类btn-lg：

<a class="btn btn-feature btn-lg pull-right" href="#">

Learn more

保存并刷新浏览器，应该看到如下结果。左侧主栏的字体大小和按钮都变大了，而且使用了brand-feature颜色。



与此同时，第二（中）栏的字体大小和按钮颜色正是我们想要的。接下来需要修改的是降低第三栏内容的重要程度。

**4.7.4　调整第三栏**

对第三栏要做的很简单，就是缩小字体大小，同时让按钮不那么突出。

1. 首先调整字体大小。在\_variables.less中，调整@font-size-small变量：

**@font-size-small: ceil(@font-size-base \* 0.90);**

2. 接下来把下面这行添加到\_page-contents.less中：

.content-tertiary {

font-size: @font-size-small;

}

3. 保存，编译，刷新，字号变小了。

4. 接下在，要编辑index.html中的按钮类，把btn-primary改为btn-default，并使用btn-xs减小其尺寸：

<a class="btn btn-default btn-xs pull-right" href="#">Read more ...

这样就减小了按钮并把背景变成了白色。

5. 再把按钮的背景颜色改为浅灰色，同时调整字体颜色和边框。在\_variables.less中，像下面这样调整三个@btn-default-变量的值：

@btn-default-color: @gray;

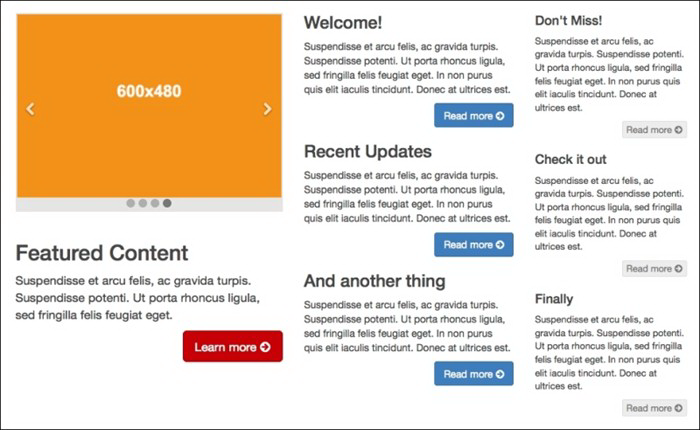
@btn-default-bg: @gray-lightest;

@btn-default-border: darken

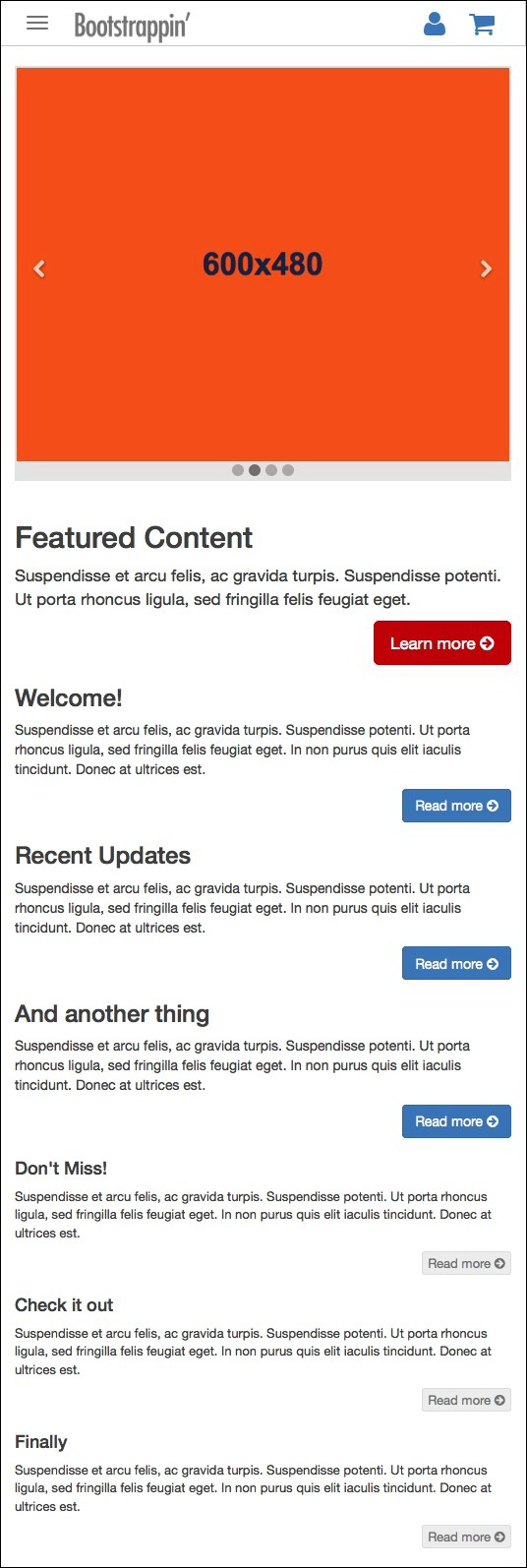
(@btn-default-bg, 5%);

保存修改，编译文件，刷新浏览器。

现在页面的层次已经很清晰了，从左到右依次是主内容、次内容和第三栏。



再看看我们的设计在小屏幕单栏布局时的样子：



在窄视口中，三栏布局变成了垂直排列的一栏，主内容在上，然后是次内容和第三栏。

那么剩下所要做的，就是对设计精雕细琢，以便让它在不同设备和视口中体验更佳。

**4.7.5　针对多个视口进行微调**

无论在什么视口中，通常都应该在页面中提供一些留白。另外，每个区块的边框最好也有所标示。

1. 首先，在内容上下各添加一些内边距。给main元素添加一些上内边距，这个内边距适用于所有视口，所以不必使用媒体查询：

main {

padding-top: 20px;

padding-bottom: 40px;

}

2. 然后，设置分栏在单栏布局时清除上方的浮动元素。如果不设置，第二和第三栏可能会覆盖紧上方的按钮。这些样式要写在媒体查询中，以便限制它只应用到窄视口：

//Make columns clear floats in narrow viewport single-column layout

@media (max-width: @screen-sm-min) {

[class\*="col-"] {

clear: both;

}

}

就这样了，主内容区收工。最后是复杂的页脚。

**4.8　复杂的页脚**

接下来我们要实现一个复杂的多用途的页脚，页脚包括：指向网站三个重要栏目的三组链接、About Us文本、社交媒体图标，还有Logo。

**4.8.1　准备标记**

我们先从准备标记着手。页脚的目的是对用户尽可能有用，其标记可以按如下步骤来准备。

1. 用编辑器打开项目文件夹04\_Code\_BEGIN中的footer-content.html，复制其中的全部内容。

2. 再回到index.html，找到粘贴它的位置。这个位置就在footer role="contentinfo"中，位于div class="container"后面，ul class="social"前面。（我已经在页脚的位置放了注释。）

3. 粘贴内容之前，我们再准备一下利用Bootstrap的网格系统。为此，像下面这样把页脚区包含在div class="row"中：

<footer role="contentinfo">

<div class="container">

<div class="row">

**<!-- INSERT ADDITIONAL FOOTER CONTENT HERE -->**

</div><!-- /.row -->

<ul class="social">

4. 下面把内容粘贴到位。

5. 接下来把三组链接及各自的标题包含在类为col-md-2的div中。这样就保证在中大视口中每一组链接的宽度是父元素总宽度的六分之一。加在一块，三组链接占视口宽度的一半。

**<div class="col-md-2">**

<h3>Categories</h3>

6. 继续把这一行内容做完，把About Us标题及其段落包含在一个类为col-md-6的div中，这样它就占据了剩下的一半宽度。

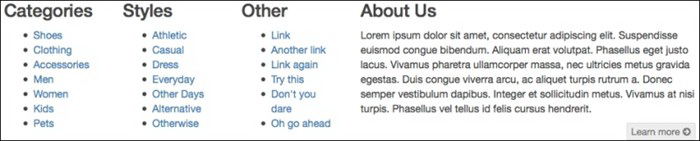
**<div class="about col-md-6">**

<h3>About Us</h3>

　别忘了给每个div补全结束标签。

7. 保存，刷新，检查结果。

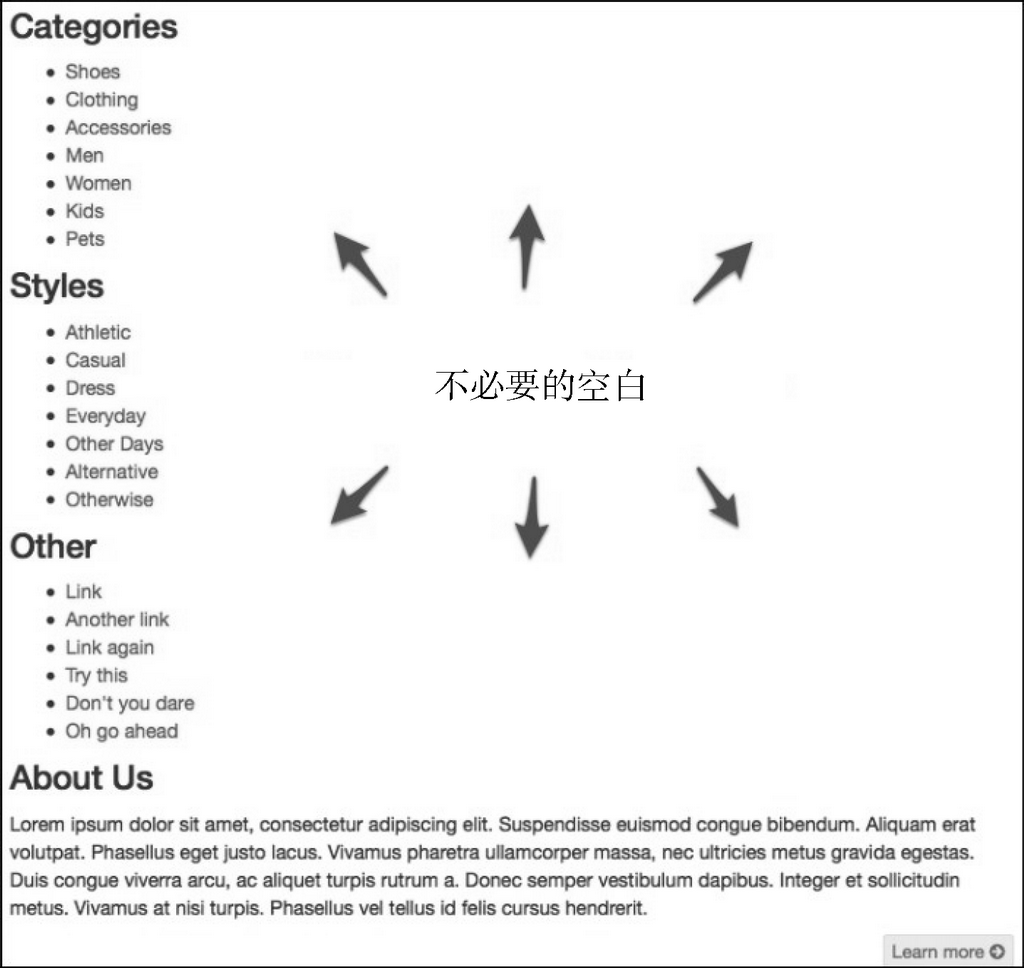
在980px及更大的视口中，页脚中的栏如下所示：



这是中大型视口中的布局。在超小的屏幕中，呈现出的则是单栏布局。但对于768px到980px之间的平板电脑，我们的布局还是需要调整一下。开始吧。

**4.8.2　调整布局适应平板**

测试一下视口在768px到980px之间时的布局。Bootstrap把这个区间界定为小断点，对应变量@screen-sm和col-sm-网格类。在这个宽度内，单栏布局会导致不必要的空白，如下图所示：



要改进这个布局，可以让三组链接浮动起来。使用Bootstrap的类col-sm-4，可以将每一栏设置为三分之一宽，使用ol-sm-12将About Us设置为全宽：

<div class="**col-sm-4** col-md-2">

...

<div class="**col-sm-4** col-md-2">

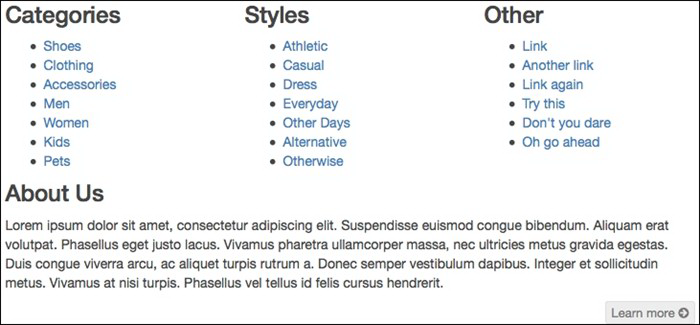
...

<div class="**col-sm-4** col-md-2">

...

<div class="about **col-sm-12** col-md-6">

保存并在小视口中测试一下，应该能看到下图所示的结果：



改进很多啊！不过离完成还远着呢。尝试单击上方三栏中的链接，恐怕点击不了。检查元素会发现包含About Us栏的div元素并没有清除上方的浮动栏。虽然About Us标题及文本会出现在三个浮动栏下方，但那个div元素却仍然覆盖在三栏内容上面。

**4.8.3　针对性地清除**

在Bootstrap标准的布局方案中，应该使用类为row的div元素清除上方的浮动栏。但我们在此要使用另一个方案，因为我们只希望这个内容块在特定的断点范围清除浮动。

为此，可以在LESS文件中写一些自定义的样式。不过，也可以直接在标记中使用Bootstrap的响应式实用类提供的针对性的clearfix。因为我们已经在标记中指定了网格类，那就干脆使用第二个方案。

关于我们使用的这个方案，大家可以参考Bootstrap的文档，地址在： <http://getbootstrap.com/css/#grid-responsive-resets>。如此一来，就要创建一个类为clearfix的div，并添加一个Bootstrap的响应式实用类，使其只在小屏幕中可见。就把这个div放到About Us栏的前面吧：

<div class="**clearfix visible-sm**"></div>

<div class="about col-sm-12 col-md-6">

这个clearfix类会强制当前元素清除上方的浮动。而visible-sm类则控制这个div仅在小屏幕，也就是我们指定的断点范围内可见。在其他断点区间，这个div元素就像不存在一样。

保存以上修改，刷新浏览器。这次就会看到About Us栏清除了它上方的浮动，而链接也可以点击了。

大功告成。最后再稍微修整几处。

**4.8.4　修整细节**

对于页脚，我们还想再修整几个地方，包括：

* 修整三组链接的外观；
* 调整内外边距；
* 反转配色方案，与导航条保持一致。

要完成以上工作，得写一些自定义的样式。我们遵照层叠原理，先写一些针对页脚的通用规则，然后再过渡到特殊规则。

1. 在编辑器中打开\_footer.less以添加针对页脚的自定义样式。

在这个文件里，可以看到第2章中一些样式的修改版本。包括给页脚添加的内边距，以及针对社交媒体图标，还有页脚中Logo的样式规则。

2. 现在开始添加针对复杂页脚的样式。首先，缩小页脚字体大小，反转配色与导航条对应——蓝色背景，浅色文本。我们先设置成这样的颜色，然后再把它们稍微调暗一点。为此，就要使用\_variables.less中适当的变量，包括 @font-size-small、@navbar- inverse-bg和@navbar-inverse-color：

footer[role="contentinfo"] {

padding-top: 24px;

padding-bottom: 24px;

**font-size: @font-size-small;**

**background-color: darken(@navbar-inverse-bg, 18%);**

**color: darken(@navbar-inverse-color, 18%);**

　这里及后面的规则，都要放到footer[role="contentinfo"]选择符下。

3. 接下来调整链接和按钮，以适应新的配色。同样要把规则放在footer[role="contentinfo"]选择符下：

a {

color: @navbar-inverse-color;

&:focus,

&:hover,

&:active {

color: @navbar-inverse-link-hover-color;

}

}

.btn-default {

color: darken(@navbar-inverse-bg, 18%) !important;

}

4. 下面是四个h3标题，调整它们的字号，去掉下外边距，并把文本转换成大写：

h3 {

font-size: 120%;

margin-bottom: 4px;

text-transform: uppercase;

}

5. 然后，再去掉链接列表前的项目符号，调整它们的内外边距：

ul {

list-style: none;

padding: 0;

margin: 0;

}

　在这个设计中，这里规则针对的是站点页脚中的所有无序列表。如果你自己网站中包含其他无序列表，可能要考虑添加特定的类，比如footer-nav。

6. 最后，调整社交媒体图标。就是添加一些上内边距，调整一下颜色，以便与新配色方案协调一致。因为图标使用的是Font Awesome字体，所以只要调整颜色和背景颜色的值即可：

ul.social {

...

padding: 24px 0 0;

...

> li {

...

> a {

...

**background-color: darken(@navbar-inverse-bg, 27%);**

**color: darken(@navbar-inverse-color, 18%);**

...

&:hover {

...

**background-color: darken(@navbar-inverse-bg, 32%);**

**color: @navbar-inverse-link-hover-color;**

}

}

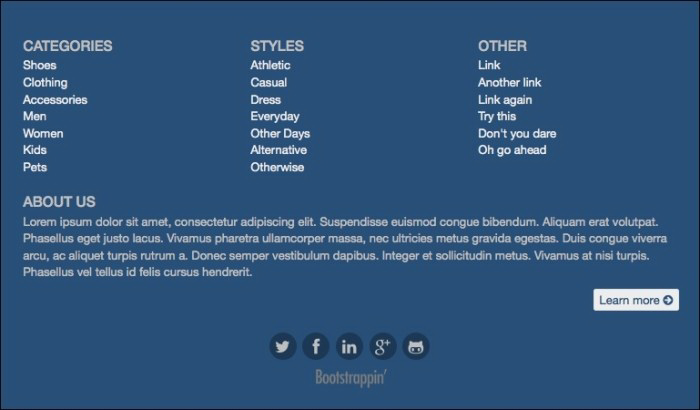
}

}

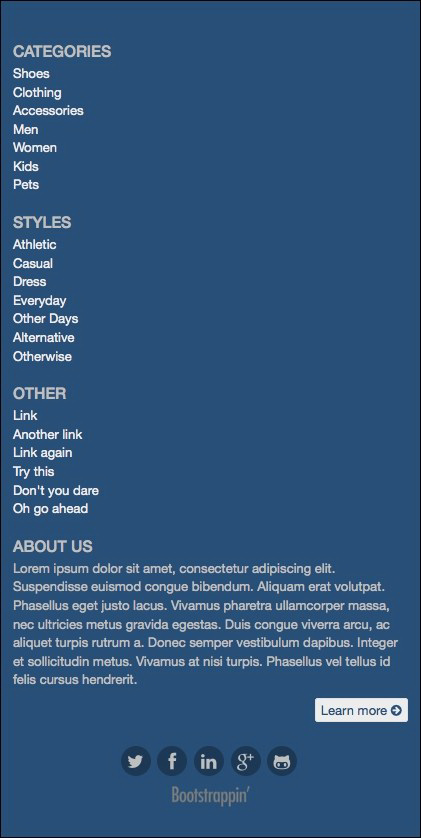
就这样了。保存，编译，刷新，好好欣赏吧！以下是页脚在中大型屏幕中的结果：



在小屏幕中的效果如下：



在超小屏幕中的效果如下：



不错嘛！我们的页脚其实挺复杂的，内容多，又适配了超小、小、中、大型视口。

**4.9　小结**

通过本章的项目，我们又掌握了一些利用Bootstrap的新技术。可以简单总结如下。

* 为复杂的响应式导航条添加样式，使其在中大视口中出现在Logo下方，而在小屏幕中又能折叠起来。
* 构建了自定义的响应式实用导航条，文本和图标都能创造性地适应较大和较小的屏幕。
* 为页面的主内容设计了响应式布局，使三栏内容主次分明。
* 构建了一个复杂的页脚，有效地组织了多个链接块，还有跨视口的文本段落。
* 以导航条配色为基础增强了页脚的配色。