**BOOTSTRAP第一天笔记**

**响应式页面**

**响应式页面概述**

**1. 什么是响应式页面**

响应式网页（RWD，Responsive Web Design），2010年5月由Ethan Marcotte提出。简而言之，就是一个网站能够针对多个终端提供不同的显示样式，而不是针对每个终端制作一个特定的版本。

由于智能手机和平板电脑的兴起，网站页面的制作不仅要满足PC端浏览器的显示，还要满足智能手机浏览器的显示或者平板电脑浏览器的显示。（**PC端、智能手机或平板电脑的浏览器之间的区别，主要集中在分辨率的不同。**）

**2. 为什么需要响应式页面**

想要同时满足PC端、智能手机和平板电脑浏览器的正常显示，实际上有以下两种实现方式：

a. 分别针对PC端、智能手机和平板电脑的浏览器制作特定的页面。例如 <https://www.tmall.com/>

* 优点：用户体验好，适合复杂页面。
* 缺点：工作量大，不够灵活。

b. 制作一个页面同时满足PC端、智能手机和平板电脑的浏览器显示。例如 <http://www.intel.cn/>

* 优点：比较灵活，一个页面适应不同终端。
* 缺点：页面加载时间长，效率降低。

如果PC端显示的页面与智能手机或平板电脑显示的页面不同时，建议选择 *a* 方案。比如智能手机端的可以利用 GPS 定位当前位置等不同于PC端的功能。但大多数网站的页面，无论是PC端还是智能手机或平板电脑显示的内容基本相同，建议选择 *b* 方案。

**3. 响应式页面的构成**

目前实现响应式页面可以通过以下两种方式：

a. 使用框架实现，例如 [Bootstrap](https://getbootstrap.com) 等。这种方式开发响应式页面更快速、简单，但由于CSS样式是集成式的，导致页面样式相似，没有特点。

b. 手写方式实现。这种方式开发响应式页面速度比较慢、代码多，但可以根据网站特点定制CSS样式。

手写方式需要使用到以下几种技术：

* 流式网格布局
* CSS3 的媒体查询
* 相对单位替换绝对单位

**测试响应式页面**

**1. 如何测试响应式页面**

测试响应式页面可以通过以下三种方式：

* 通过真实设备进行测试。这种方式进行测试效果最好，但也是最复杂，并且需要不同分辨率的设备，成本较高。
* 通过第三方模拟器进行测试。这种方式一般都是针对智能手机终端研发工具自带，需要在电脑安装不同开发工具，比较麻烦，并且性能较差。或者通过在线测试网站进行测试，测试周期较长。
* 通过浏览器自带的设备模拟器进行测试。这种方式是最方便的，主流浏览器都自带了设备模拟器，并且测试效果较好。

**2. 使用浏览器测试**

以 Chrome 浏览器为例。

打开 Chrome 浏览器，点击右上角的菜单，选择“更多工具”中的“开发者工具”。（**Windows操作系统版本的 Chrome 浏览器可以通过 F12 快捷键直接打开。**）

**值得注意的是：**

* 每次更换测试设备时，需要重新刷新页面显示。
* 有些功能无法正常测试，例如电话薄、摄像头等。

**3. 使用第三方工具测试**

**4. 使用真实设备测试**

**编写响应式页面**

**1. viewport元元素**

**1）什么是 viewport ？**

viewport 被翻译为“视区”，表示页面的可视区域。通过 HTML 页面的 meta 元元素进行设置，是响应式页面设计的必备内容。

移动端浏览器在一个通常比屏幕更宽的虚拟“窗口”（视口）中渲染页面，从而无序将所有页面都压缩进小屏幕里（那样会把很多没有针对移动端进行优化的站点打乱）。用户可以通过平移和缩放来浏览页面的不同区域。

通过设置 viewport 元元素可以允许开发者控制显示的尺寸及比例。如今大部分移动端浏览器都支持 viewport 元元素，尽管它**不是 web 标准的一部分**。

**2）设置 viewport**

一个典型的针对移动端优化的站点包含类似下面的内容：

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">

viewport 元元素的属性如下表：

| **属性名称** | **说明** | **可选值** |
| --- | --- | --- |
| width | 设置窗口的宽度 | 像素值或device-width（表示100%时屏幕宽度的像素值） |
| height | 设置窗口的高度 | 一般不指定 |
| initial-scale | 设置初始时的缩放倍率 |  |
| minimum-scale | 设置允许的最小缩放倍率 |  |
| maximum-scale | 设置允许的最大缩放倍率 |  |
| user-scalable | 是否允许用户手动缩放 | 1或yes表示允许；0或no表示不允许 |

meta 元元素建议设定在 head 元素中的最前面。

**2. 使用相对单位**

**1）使用相对宽度值**

通过学习 CSS 知道设置宽度可以使用 px 像素值和百分值。使用像素值设置宽度是固定值，不能随着屏幕大小变化而变化。实现响应式页面的宽度只能使用百分值或者auto。

**如何从固定布局修改为百分比布局**

在《无懈可击的 Web 设计》一书中，提供了一个简易可行的公式：

目标元素宽度 / 上下文元素宽度 = 百分比宽度

以下代码是使用像素值设置 HTML 页面元素：

div {

width : 500px;

height : auto;

border : 1px solid black;

}

p {

width : 480px;

padding-left: 10px;

}

将上述像素值修改为百分比：

div {

width : 60%;

height : auto;

border : 1px solid black;

}

p {

width : 96%; /\* 480/500=96% \*/

padding-left: 2%; /\* 10/500=2% \*/

}

**2）用 em 替换 px**

之前使用 em 替代 px 设置字体，主要是为了解决老版本的 IE 浏览器无法缩放以像素值为单位的文字。

目前使用 em 替代 px 设置字体，主要的目的具有以下两个：

* 实现老版本 IE 浏览器可以缩放文字。
* 更好地实现响应式页面，使页面效果更好。

em 的实际大小事相对于上下文的字体大小而言的，例如 1em 表示与上下文字体相同大小。em 单位设置字体同样可以通过以下公式进行计算：

目标元素尺寸 / 上下文元素尺寸 = 百分比尺寸

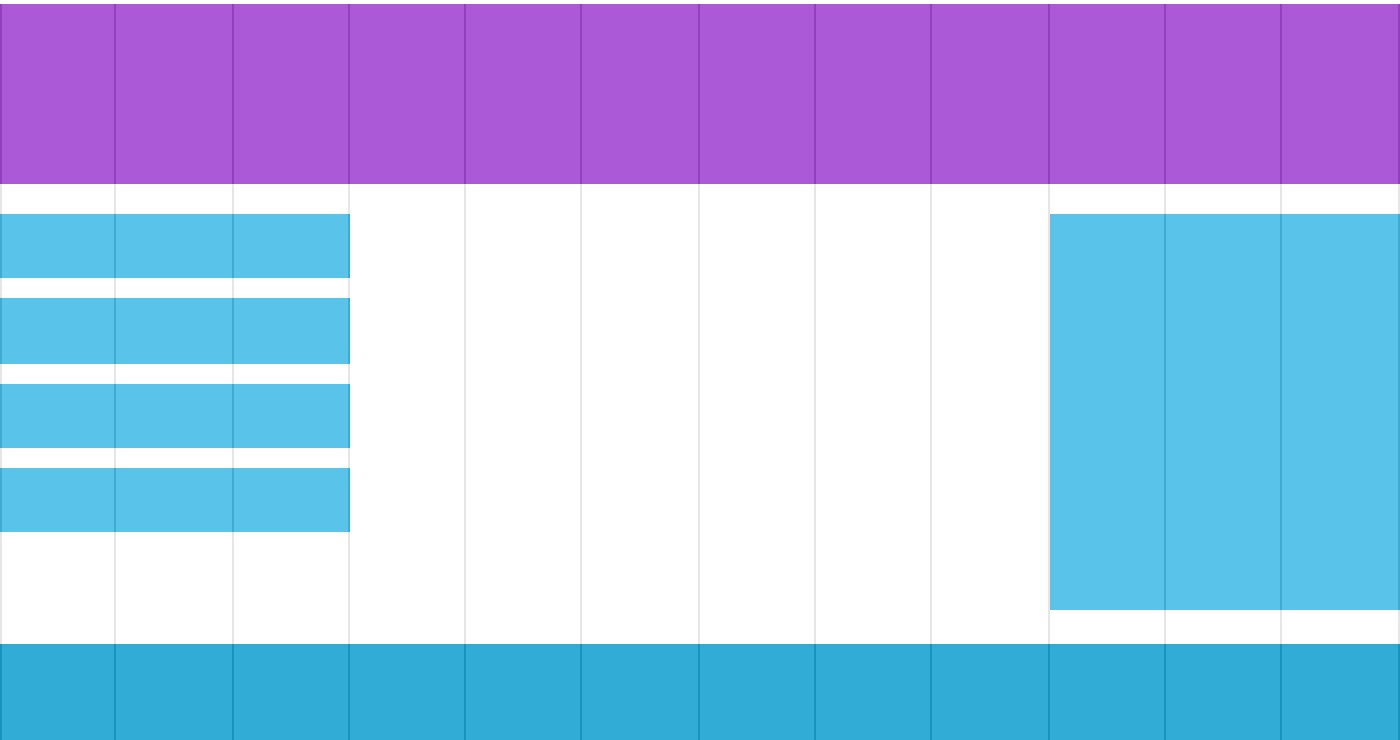
**值得注意的是：**浏览器显示默认文字大小都是 16 像素。

**3. 流式布局**

**1）什么是流式布局**

所谓流式布局，就是页面元素的宽度按照屏幕进行适配调整。简单来说，就是 HTML 页面中的元素会根据分辨率的不同而变化大小，但位置并不会有任何变化。

这种布局的主要问题就是，如果屏幕尺度跨度太大，那么在相对其原始设计而言过小或过大的屏幕上不能正常显示。



**2）创建流式布局**

流式布局通常被分为 12 列。将 HTML 页面的宽度设置为 100%，这样 HTML 页面会根据浏览器窗口大小的变化而自动伸缩。

* 计算每列所占的百分比：100% / 12 列 = 8.33%

根据以上的计算结果，设置 CSS 的 class：

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}

.col-4 {width: 33.33%;}

.col-5 {width: 41.66%;}

.col-6 {width: 50%;}

.col-7 {width: 58.33%;}

.col-8 {width: 66.66%;}

.col-9 {width: 75%;}

.col-10 {width: 83.33%;}

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

* 将所有列设置为浮动（一般为左浮动）

[class\*="col-"] {

float: left;

}

* 将 HTML 页面中所有的元素 box-sizing 设置为 border-box。
  + 当 box-sizing 的值为 content-box（默认值），标准盒模型。

尺寸计算公式：width = 内容的宽度，height = 内容的高度。

宽度和高度都不包含内容的边框（border）和内边距（padding）。

div {

width : 300px;

height : 200px;

border : 10px solid black;

background : deepskyblue;

}

* + 当 box-sizing 的值为 border-box，IE 怪异模式（Quirks mode）使用的 盒模型 。

尺寸计算公式：width = border + padding + 内容的宽度，height = border + padding + 内容的高度。

**注意：**外边距和边框将会包括在盒子中。

div {

box-sizing: border-box;

width : 300px;

height : 200px;

border : 10px solid black;

background : deepskyblue;

}

根据上述流式布局的内容，实现以下案例：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

\* {

box-sizing: border-box;

}

[class\*="col-"] {

float: left;

padding: 15px;

}

.col-1 {

width: 8.33%;

}

.col-2 {

width: 16.66%;

}

.col-3 {

width: 25%;

}

.col-4 {

width: 33.33%;

}

.col-5 {

width: 41.66%;

}

.col-6 {

width: 50%;

}

.col-7 {

width: 58.33%;

}

.col-8 {

width: 66.66%;

}

.col-9 {

width: 75%;

}

.col-10 {

width: 83.33%;

}

.col-11 {

width: 91.66%;

}

.col-12 {

width: 100%;

}

html {

font-family: "Lucida Sans", sans-serif;

}

header {

background-color: #9933cc;

color: #ffffff;

padding: 15px;

}

aside ul {

list-style-type: none;

margin: 0;

padding: 0;

}

aside li {

padding: 8px;

margin-bottom: 7px;

background-color: #33b5e5;

color: #ffffff;

box-shadow: 0 1px 3px rgba(0, 0, 0, 0.12), 0 1px 2px rgba(0, 0, 0, 0.24);

}

aside li:hover {

background-color: #0099cc;

}

</style>

</head>

<body>

<header>

<h1>Chania</h1>

</header>

<aside class="col-3">

<ul>

<li>The Flight</li>

<li>The City</li>

<li>The Island</li>

<li>The Food</li>

</ul>

</aside>

<article class="col-9">

<h2>The City</h2>

<p>Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old

town and the modern city.</p>

<p>Resize the browser window to see how the content respond to the resizing.</p>

</article>

</body>

</html>

**4. 响应式图片**

实现图片的显示根据浏览器窗口大小的变化而变化，相对比较简单。只需要将图片的 width 设置为 100% 即可。

img {

width: 100%;

height: auto;

}

**但这样设置后，图片的显示可能会比实际大小更大。解决这个问题可以利用 max-width 替代 width 属性即可。**

img {

max-width: 100%;

height: auto;

}

这样设置后，图片的显示永远不会比实际大小更大了。

**5. CSS媒体查询**

想要满足更复杂的情况，需要使用 CSS3 中提供的媒体查询进行解决。

媒体查询 包含了一个媒体类型和至少一个使用如宽度、高度和颜色等媒体属性来限制样式表范围的表达式。CSS3加入的媒体查询使得无需修改内容便可以使样式应用于某些特定的设备范围。

**媒体查询语法**

媒体查询 包含 一个媒体类型、逻辑操作符和一个或多个媒体属性。如果媒体查询中的媒体类型与文档要展示的设备相符则查询结果为真，并且媒体查询中的所有表达式为真。

媒体查询具体的写法有以下两种方式：

* link 元素中的 CSS 媒体查询。

<link rel="stylesheet" media="mediatype and|not|only (media feature)" href="mystylesheet.css">

这种方式需要针对不同设备编写不同的 CSS 样式表。

* 样式表中的 CSS 媒体查询。

@media mediatype and|not|only (media feature) {

CSS-Code;

}

这种方式只需要编写一个 CSS 样式表，在样式表中根据不同的设备设置不同的样式。（**g更常用**）

**媒体类型**

| **值** | **描述** |
| --- | --- |
| all | 用于所有设备 |
| print | 用于打印机和打印预览 |
| screen | 用于电脑屏幕，平板电脑，智能手机等 |
| speech | 用于屏幕阅读器等发声设备 |

媒体类型更常用的值为 screen。

**逻辑操作符**

| **值** | **描述** |
| --- | --- |
| and | 用于把多个 媒体属性 组合起来，合并到同一条媒体查询中。只有当每个属性都为真时，这条查询的结果才为真. |
| not | 用于对一条媒体查询的结果进行取反。 |
| only | 表示仅在媒体查询匹配成功的情况下应用指定样式。可以通过它让选中的样式在老式浏览器中不被应用。 |

也可以将多个媒体查询以逗号分隔放在一起；只要其中任何一个为真，整个媒体语句就返回真。相当于 or 操作符。

| **值** | **描述** |
| --- | --- |
| width | 定义输出设备中的页面可见区域宽度 |
| max-width | 定义输出设备中的页面最大可见区域宽度 |
| max-device-width | 定义输出设备的屏幕最大可见宽度 |
| min-width | 定义输出设备中的页面最小可见区域宽度 |
| min-device-width | 定义输出设备的屏幕最小可见宽度 |

**媒体查询示例**

**link 元素中的 CSS 媒体查询**

* HTML 代码

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>LINK元素方式的媒体查询</title>

<meta name="viewport" content="width=device-width,initial-scale=1"/>

<link rel="stylesheet" href="css/main.css">

<link rel="stylesheet" media="screen and (min-width:768px)" href="css/pc.css">

<link rel="stylesheet" media="screen and (max-width:768px)" href="css/phone.css">

</head>

<body>

<header>

<h1>Chania</h1>

</header>

<aside class="col-3">

<ul>

<li>The Flight</li>

<li>The City</li>

<li>The Island</li>

<li>The Food</li>

</ul>

</aside>

<article class="col-9">

<h2>The City</h2>

<p>Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old

town and the modern city.</p>

<p>Resize the browser window to see how the content respond to the resizing.</p>

</article>

</body>

</html>

* 针对分辨率的宽度在 768px 之上的样式表

.col-1 {

width: 8.33%;

}

.col-2 {

width: 16.66%;

}

.col-3 {

width: 25%;

}

.col-4 {

width: 33.33%;

}

.col-5 {

width: 41.66%;

}

.col-6 {

width: 50%;

}

.col-7 {

width: 58.33%;

}

.col-8 {

width: 66.66%;

}

.col-9 {

width: 75%;

}

.col-10 {

width: 83.33%;

}

.col-11 {

width: 91.66%;

}

.col-12 {

width: 100%;

}

[class\*="col-"] {

float: left;

padding: 15px;

}

* 针对分辨率的宽度在 768px 之下的样式表

[class\*="col-"] {

width: 100%;

padding: 15px;

}

**样式表中的 CSS 媒体查询**

* HTML 代码

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>CSS样式表中的媒体查询</title>

<meta name="viewport" content="width=device-width,initial-scale=1"/>

<link rel="stylesheet" href="css/main.css">

<link rel="stylesheet" href="css/media.css">

</head>

<body>

<header>

<h1>Chania</h1>

</header>

<aside class="col-3">

<ul>

<li>The Flight</li>

<li>The City</li>

<li>The Island</li>

<li>The Food</li>

</ul>

</aside>

<article class="col-9">

<h2>The City</h2>

<p>Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old

town and the modern city.</p>

<p>Resize the browser window to see how the content respond to the resizing.</p>

</article>

</body>

</html>

* CSS 代码

/\* For desktop: \*/

.col-1 {

width: 8.33%;

}

.col-2 {

width: 16.66%;

}

.col-3 {

width: 25%;

}

.col-4 {

width: 33.33%;

}

.col-5 {

width: 41.66%;

}

.col-6 {

width: 50%;

}

.col-7 {

width: 58.33%;

}

.col-8 {

width: 66.66%;

}

.col-9 {

width: 75%;

}

.col-10 {

width: 83.33%;

}

.col-11 {

width: 91.66%;

}

.col-12 {

width: 100%;

}

[class\*="col-"] {

float: left;

padding: 15px;

}

@media only screen and (max-width: 768px) {

/\* For mobile phones: \*/

[class\*="col-"] {

width: 100%;

}

}

**BOOTSTRAP**

**起步**

**1. 什么是BOOTSTRAP**

**1）BOOTSTRAP概述**

Bootstrap 是 Twitter 公司开发的一个基于 HTML、CSS、JavaScript 的技术框架，符合 HTML、CSS 规范，且代码简介、视觉优美。

Bootstrap 集合 HTML、CSS 和 JavaScript，使用了最新的浏览器技术，为实现快速开发提供了一套前端工具包。使用 Bootstrap 不仅可以构建出非常优雅的 HTML 页面，而且占用资源非常少，使用 gzip 压缩后大小仅用 10KB。并且 Bootstrap 在跨浏览器兼容方面表示也很好。

**2）选择BOOTSTRAP的理由**

* Bootstrap 的 HTML 是基于 HTML5 的最新技术。
* Bootstrap 可以快速实现响应式页面。
* Bootstrap 集成了非常友好的 CSS 样式表，对于非设计人员也可以制作出很漂亮的网页。

**3）BOOTSTRAP提供的功能**

根据 Bootstrap 官网提供的文档，可以知道 Bootstrap 分为以下几个模块：

* 全局 CSS 样式：提供了格栅系统、表格、表单、按钮等集成样式。
* 组件：提供了下拉菜单、输入框、导航、列表组等组件。
* 插件：提供了模态框、滚动监听、警告框、弹出框等插件。

**4）BOOTSTRAP的版本变化**

* 2011年8月，Twitter 推出了 Bootstrap 1 版本，该工具由 Twitter 的设计师 Mark Otto 和 Jacob Thornton 合作完成。
* 2012年1月，Twitter 推出了 Bootstrap 2 版本。相比 Bootstrap 1 版本，Bootstrap 2 添加了响应式设计、采用 12 栏栅格布局，并且清晰地划分出 CSS 布局、组件和插件等功能。
* 2013年8月，Twitter 推出了 Bootstrap 3 版本。在这个版本中，采用移动优先和更好的盒子模型等。并且官方不再支持 IE 7及以下版本的浏览器。
* 目前，Bootstrap 4 只是开发者预览版。

**2. 下载BOOTSTRAP**

**1）Bootstrap 官网及下载地址**

* Bootstrap 官网：<http://getbootstrap.com>
* Bootstrap 下载地址：[http://getbootstrap.com/getting-started/#download](http://getbootstrap.com/getting-started/" \l "download)

**2）Bootstrap 使用的两种方式**

* 将 Bootstrap 提供的压缩包下载，导入到工程目录中使用。
* 使用 CDN 加速服务，例如以下内容：

<!-- 新 Bootstrap 核心 CSS 文件 -->

<link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.6/css/bootstrap.min.css">

<!-- 可选的Bootstrap主题文件（一般不用引入） -->

<link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.6/css/bootstrap-theme.min.css">

<!-- jQuery文件。务必在bootstrap.min.js 之前引入 -->

<script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></script>

<!-- 最新的 Bootstrap 核心 JavaScript 文件 -->

<script src="//cdn.bootcss.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>

**3）Bootstrap 目录结构及说明**

下载压缩包之后，将其解压缩到任意目录即可看到以下（压缩版的）目录结构：

bootstrap/

├── css/ 样式库

│ ├── bootstrap.css 样式文件

│ ├── bootstrap.css.map

│ ├── bootstrap.min.css 压缩后的样式文件

│ ├── bootstrap-theme.css 主题文件

│ ├── bootstrap-theme.css.map

│ └── bootstrap-theme.min.css 压缩后的主题文件

├── js/ 核心 js 文件

│ ├── bootstrap.js 核心 js 文件

│ └── bootstrap.min.js 压缩后的核心 js 文件

└── fonts/ 字体库

├── glyphicons-halflings-regular.eot

├── glyphicons-halflings-regular.svg

├── glyphicons-halflings-regular.ttf

├── glyphicons-halflings-regular.woff

└── glyphicons-halflings-regular.woff2

**3. 对浏览器的支持情况**

**1）目前支持的浏览器**

|  | **Chrome** | **Firefox** | **Internet Explorer** | **Opera** | **Safari** |
| --- | --- | --- | --- | --- | --- |
| Android | 支持 | 支持 | N/A | 不支持 | N/A |
| IOS | 支持 | N/A | N/A | 不支持 | 支持 |
| Mac OS X | 支持 | 支持 | N/A | 支持 | 支持 |
| Windows | 支持 | 支持 | 支持 | 支持 | 不支持 |

**2）IE 8及之前版本的兼容**

* 对 HTML 5 提供的新元素，需要通过 html5shiv.js 库进行兼容。
* 对 CSS 3 提供的媒体查询，需要通过 respond.js 库进行支持。

需要在使用 Bootstrap 框架的页面 head 元素中，插入以下内容：

<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->

<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->

<!--[if lt IE 9]>

<script src="//cdn.bootcss.com/html5shiv/3.7.2/html5shiv.min.js"></script>

<script src="//cdn.bootcss.com/respond.js/1.4.2/respond.min.js"></script>

<![endif]-->

**3）IE 兼容模式**

Bootstrap 不支持 IE 古老的兼容模式。为了让 IE 浏览器运行最新的渲染模式下，建议将此 <meta> 元素加入到页面中：

<meta http-equiv="X-UA-Compatible" content="IE=edge">

**4）国产浏览器高速模式**

国内浏览器厂商一般都支持兼容模式（即 IE 内核）和高速模式（即 webkit 内核），不幸的是，所有国产浏览器都是默认使用兼容模式，这就造成由于低版本 IE （IE8 及以下）内核让基于 Bootstrap 构建的网站展现效果很糟糕的情况。

将下面的 <meta> 元素加入到页面中，可以让部分国产浏览器默认采用高速模式渲染页面：

<meta name="renderer" content="webkit">

**4. box-sizing**

Bootstrap 默认使用的盒子模型是 border-box，可能会和一些第三方组件冲突。

为了避免 Bootstrap 设置的全局盒模型所带来的影响，可以重置单个页面元素或覆盖整个区域的盒模型。

* 覆盖单个页面元素

/\* 通过 CSS 代码覆盖单个页面元素的盒模型 \*/

.element {

-webkit-box-sizing: content-box;

-moz-box-sizing: content-box;

box-sizing: content-box;

}

* 重置整个区域

/\* 通过 CSS 代码重置整个区域 \*/

.reset-box-sizing,

.reset-box-sizing \*,

.reset-box-sizing \*:before,

.reset-box-sizing \*:after {

-webkit-box-sizing: content-box;

-moz-box-sizing: content-box;

box-sizing: content-box;

}

**5. 基本模板**

Bootstrap 提供了一个最基本的 HTML 模板，基于这个模板开始实现 Bootstrap 响应式页面。

<!DOCTYPE html>

<html lang="zh-CN">

<head>

<!-- 设置当前 HTML 页面的编码格式为 UTF-8 -->

<meta charset="utf-8">

<!-- 设置 IE 的兼容模式 -->

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<!-- 设置移动优先 -->

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- 上述3个meta标签必须放在最前面，任何其他内容都必须跟随其后！ -->

<title>Bootstrap最基本的HTML模板</title>

<!-- 导入 Bootstrap 框架的 CSS 文件 -->

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

<!--

html5shiv.js 文件解决 IE8及之前版本的浏览器支持 HTML 5 元素的问题。

respond.js 文件解决 IE8及之前版本的浏览器支持 CSS 3 的媒体查询问题。

-->

<!--[if lt IE 9]>

<script src="bootstrap/js/html5shiv.min.js"></script>

<script src="bootstrap/js/respond.min.js"></script>

<![endif]-->

</head>

<body>

<h1>你好，世界！</h1>

<!--

由于 Bootstrap 是基于 jQuery 实现的核心 js 功能,

所以想要使用 Bootstrap 提供的插件功能,需要先导入 jQuery 文件。

-->

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<!-- 导入 Bootstrap 框架的 js 文件 -->

<script src="bootstrap/js/bootstrap.min.js"></script>

</body>

</html>

**全局CSS样式**

**1. 概述**

**1）HTML 5 文档类型**

Bootstrap 使用到的某些 HTML 元素和 CSS 属性需要将页面设置为 HTML5 文档类型。

<!DOCTYPE html>

<html lang="zh-CN">

...

</html>

**2）移动设备优先**

Bootstrap 是移动设备优先的。为了确保适当的绘制和触屏缩放，需要在 <head> 之中添加 viewport 元元素。

<meta name="viewport" content="width=device-width, initial-scale=1">

在移动设备浏览器上，通过为 viewport设置 meta 属性为 user-scalable=no 可以禁用其缩放功能。这样禁用缩放功能后，用户只能滚动屏幕，就能让你的网站看上去更像原生应用的感觉。

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">

**3）排版与链接**

Bootstrap 排版、链接样式设置了基本的全局样式。

* 为 body 元素设置 background-color: #fff;
* 使用 @font-family-base、@font-size-base 和 @line-height-base 变量作为排版的基本参数
* 为所有链接设置了基本颜色 @link-color ，并且当链接处于 :hover 状态时才添加下划线

**2. Normalize.css**

**1）CSS 重置样式库**

为了增强跨浏览器表现的一致性，Bootstrap 使用 Normalize.css（CSS 重置样式库）。

**2）什么是 Normalize.css**

Normalize.css 只是一个很小的 CSS 文件，但它在默认的 HTML 元素样式上提供了跨浏览器的高度一致性。相比于传统的 CSS reset，Normalize.css 是一种现代的、为 HTML5 准备的优质替代方案。

Normalize.css 的主要特点如下：

* 保护有用的浏览器默认样式而不是完全去掉它
* 一般化的样式：为大部分HTML元素提供
* 修复浏览器自身的bug并保证各浏览器的一致性
* 优化CSS可用性：用一些小技巧
* 解释代码：用注释和详细的文档来

**3）Normalize.css 与传统 Reset 的区别**

* Normalize.css 保护了有价值的默认值
* Normalize.css 修复了浏览器的bug
* Normalize.css 不会让你的调试工具变的杂乱
* Normalize.css 是模块化的
* Normalize.css 拥有详细的文档

**3. 布局容器**

Bootstrap 提供了以下两种布局容器：

* .container 类用于固定宽度并支持响应式布局的容器。

| **媒体查询** | **宽度值** |
| --- | --- |
| >1200px | 1170px |
| >992px | 970px |
| >768px | 750px |
| 小分辨率 | 100% |

* .container-fluid 类用于 100% 宽度，占据 viewport 的容器。

**值得注意的是：**这两种 容器类不能互相嵌套。

**4. 按钮**

**1）作为按钮的元素**

Bootstrap 利用 <a>、<button>和<input>元素作为按钮样式。

<div class="container">

<div class="row">

<a class="btn btn-default" href="#" role="button">Link</a>

<button class="btn btn-default" type="submit">Button</button>

<input class="btn btn-default" type="button" value="Input">

<input class="btn btn-default" type="submit" value="Submit">

</div>

</div>

**值得注意的是：**

1. 导航和导航条组件只支持 <button> 元素。
2. 如果 <a> 元素被作为按钮使用，务必为其设置 role="button" 属性。
3. 建议尽可能使用 <button> 元素。

**2）预定义样式**

Bootstrap 为按钮预定义了很多样式，具体样式说明如下：

| **样式名称** | **描述** |
| --- | --- |
| btn-default | 默认样式 |
| btn-primary | 首选项 |
| btn-success | 成功样式 |
| btn-info | 一般信息 |
| btn-warning | 警告样式 |
| btn-danger | 危险样式 |
| btn-link | 链接样式 |

<div class="container">

<div class="row">

<button type="button" class="btn btn-default">（默认样式）Default</button>

<button type="button" class="btn btn-primary">（首选项）Primary</button>

<button type="button" class="btn btn-success">（成功）Success</button>

<button type="button" class="btn btn-info">（一般信息）Info</button>

<button type="button" class="btn btn-warning">（警告）Warning</button>

<button type="button" class="btn btn-danger">（危险）Danger</button>

<button type="button" class="btn btn-link">（链接）Link</button>

</div>

</div>

**3）不同尺寸的按钮**

* Bootstrap 为按钮提供了大、默认、小和很小几个尺寸。

<div class="container">

<div class="row">

<button type="button" class="btn btn-default btn-lg">（大按钮）Large button</button>

<button type="button" class="btn btn-default">（默认尺寸）Default button</button>

<button type="button" class="btn btn-default">（默认尺寸）Default button</button>

<button type="button" class="btn btn-default btn-xs">（超小尺寸）Extra small button</button>

</div>

</div>

* 通过给按钮添加 .btn-block 类可以将其拉伸至父元素100%的宽度，而且按钮也变为了块级（block）元素。

<div class="container">

<div class="row">

<button type="button" class="btn btn-default btn-lg btn-block">（块级元素）Block level button</button>

</div>

</div>

**4）按钮激活状态**

为按钮元素添加 .active 类，设置其为激活状态。当按钮处于激活状态时，其表现为被按压下去（底色更深、边框夜色更深、向内投射阴影）。

<div class="container">

<div class="row">

<button type="button" class="btn btn-default btn-lg active">Button</button>

<a href="#" class="btn btn-default btn-lg active" role="button">Link</a>

</div>

</div>

**5）按钮禁用状态**

通过为按钮的背景设置 opacity 属性就可以呈现出无法点击的效果。

* 为 <button> 元素添加 disabled 属性，使其表现出禁用状态。

<button type="button" class="btn btn-default btn-lg" disabled="disabled">Button</button>

**跨浏览器兼容性**

Internet Explorer 9 及更低版本的浏览器将会把按钮中的文本绘制为灰色，并带有恶心的阴影，目前还没有解决办法。

* 为基于 <a> 元素创建的按钮添加 .disabled 类。

<a href="#" class="btn btn-default btn-lg disabled" role="button">Link</a>

**5. 图片**

**1）响应式图片**

通过为图片添加 .img-responsive 类可以让图片支持响应式布局。

如果需要让使用了 .img-responsive 类的图片水平居中，请使用 .center-block 类，不要用 .text-center。

<div class="container">

<div class="row">

<img src="imgs/1.jpg" class="img-responsive" alt="Responsive image">

</div>

<div class="row">

<img src="imgs/1.jpg" class="img-responsive center-block" alt="Responsive image">

</div>

</div>

**2）图片形状**

通过为  元素添加以下相应的类，可以让图片呈现不同的形状。

| **类名** | **描述** |
| --- | --- |
| img-rounded | 圆角矩形 |
| img-circle | 圆形 |
| img-thumbnail | 矩形 |

IE 8 不支持 CSS3 中的圆角属性。

<div class="container">

<div class="row">

<img src="imgs/1.jpg" alt="..." class="img-rounded">

<img src="imgs/1.jpg" alt="..." class="img-circle">

<img src="imgs/1.jpg" alt="..." class="img-thumbnail">

</div>

</div>

**6. 文本**

**1）文本对齐**

通过以下文本对齐类，可以简单方便的将文字重新对齐。

| **类名** | **描述** |
| --- | --- |
| text-left | 左对齐 |
| text-center | 居中对齐 |
| text-right | 右对齐 |
| text-justify | 两端对齐，段落中超出屏幕部分文字自动换行 |
| text-nowrap | 段落中超出屏幕部分不换行 |

<div class="container">

<div class="row">

<p class="text-left">Left aligned text.</p>

<p class="text-center">Center aligned text.</p>

<p class="text-right">Right aligned text.</p>

<p class="text-justify">Justified text.</p>

<p class="text-nowrap">No wrap text.</p>

</div>

</div>

**2）改变大小写**

通过以下这几个类可以改变文本的大小写。

| **类名** | **描述** |
| --- | --- |
| text-lowercase | 小写 |
| text-uppercase | 大写 |
| text-capitalize | 首字母大写 |

<div class="container">

<div class="row">

<p class="text-lowercase">Lowercased text.</p>

<p class="text-uppercase">Uppercased text.</p>

<p class="text-capitalize">Capitalized text.</p>

</div>

</div>

**3）文本颜色**

Bootstrap 允许为文本设置不同颜色表示不同含义，具体如下：

| **类名** | **描述** |
| --- | --- |
| text-muted | 提示，使用浅灰色 |
| text-primary | 首选项，使用蓝色 |
| text-success | 成功，使用浅绿色 |
| text-info | 信息，使用浅蓝色 |
| text-warning | 警告，使用黄色 |
| text-danger | 危险，使用褐色 |

<div class="container">

<div class="row">

<p class="text-muted">提示信息</p>

<p class="text-primary">首选项</p>

<p class="text-success">成功</p>

<p class="text-info">信息</p>

<p class="text-warning">警告</p>

<p class="text-danger">危险</p>

</div>

</div>

**本笔记由 *金云龙* 编写**

作者博客地址：[龙哥有话说](http://www.longestory.com)

如转载或发布，请注明作者名称

**BOOTSTRAP第二天笔记**

**全局 CSS 样式**

**1. 列表**

**1）无序列表**

<div class="container">

<div class="row">

<ul>

<li>苹果</li>

<li>香蕉</li>

<li>西瓜

<ul>

<li>大兴西瓜</li>

<li>东北西瓜</li>

</ul>

</li>

<li>芒果</li>

<li>樱桃</li>

</ul>

</div>

</div>

**2）有序列表**

<div class="container">

<div class="row">

<ol>

<li>苹果</li>

<li>香蕉</li>

<li>西瓜

<ol>

<li>大兴西瓜</li>

<li>东北西瓜</li>

</ol>

</li>

<li>芒果</li>

<li>樱桃</li>

</ol>

</div>

</div>

**3）无样式列表**

移除了默认的 list-style 样式和左侧外边距的一组元素（只针对直接子元素）。

<div class="container">

<div class="row">

<ul class="list-unstyled">

<li>苹果</li>

<li>香蕉</li>

<li>西瓜

<ul>

<li>大兴西瓜</li>

<li>东北西瓜</li>

</ul>

</li>

<li>芒果</li>

<li>樱桃</li>

</ul>

</div>

</div>

**4）内联列表**

通过设置 display: inline-block; 并添加少量的内补（padding），将所有元素放置于同一行。

<div class="container">

<div class="row">

<ul class="list-inline">

<li>苹果</li>

<li>香蕉</li>

<li>西瓜

<ul>

<li>大兴西瓜</li>

<li>东北西瓜</li>

</ul>

</li>

<li>芒果</li>

<li>樱桃</li>

</ul>

</div>

</div>

**5）描述**

**a. 垂直排列的描述**

带有描述的短语列表。

<div class="container">

<div class="row">

<dl>

<dt>Description lists</dt>

<dd>A description list is perfect for defining terms.</dd>

<dt>Euismod</dt>

<dd>Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit.<br>

Donec id elit non mi porta gravida at eget metus.</dd>

<dt>Malesuada porta</dt>

<dd>Etiam porta sem malesuada magna mollis euismod.</dd>

</dl>

</div>

</div>

**b. 水平排列的描述**

.dl-horizontal 可以让

内的短语及其描述排在一行。

<div class="container">

<div class="row">

<dl class="dl-horizontal">

<dt>Description lists</dt>

<dd>A description list is perfect for defining terms.</dd>

<dt>Euismod</dt>

<dd>Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit.<br>

Donec id elit non mi porta gravida at eget metus.</dd>

<dt>Malesuada porta</dt>

<dd>Etiam porta sem malesuada magna mollis euismod.</dd>

</dl>

</div>

</div>

**2. 表格**

**1）基本表格**

为任意 <table> 标签添加 .table 类可以为其赋予基本的样式。

<div class="container">

<div class="row">

<table class="table">

<thead>

<tr>

<th>ID</th>

<th>名称</th>

<th>价格</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td>书籍</td>

<td>12.50</td>

</tr>

<tr>

<td>2</td>

<td>电视</td>

<td>1000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

</tbody>

</table>

</div>

</div>

**2）条纹状表格**

通过 .table-striped 类可以给 <tbody> 之内的每一行增加斑马条纹样式。

<div class="container">

<div class="row">

<table class="table table-striped">

<thead>

<tr>

<th>ID</th>

<th>名称</th>

<th>价格</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td>书籍</td>

<td>12.50</td>

</tr>

<tr>

<td>2</td>

<td>电视</td>

<td>1000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

</tbody>

</table>

</div>

</div>

条纹状表格是依赖 :nth-child CSS 选择器实现的，而这一功能不被 IE 8 支持。

**3）带边框表格**

添加 .table-bordered 类为表格和其中的每个单元格增加边框。

<div class="container">

<div class="row">

<table class="table table-bordered">

<thead>

<tr>

<th>ID</th>

<th>名称</th>

<th>价格</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td>书籍</td>

<td>12.50</td>

</tr>

<tr>

<td>2</td>

<td>电视</td>

<td>1000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

</tbody>

</table>

</div>

</div>

**4）鼠标悬停**

通过添加 .table-hover 类可以让 <tbody> 中的每一行对鼠标悬停状态作出响应。

<div class="container">

<div class="row">

<table class="table table-hover">

<thead>

<tr>

<th>ID</th>

<th>名称</th>

<th>价格</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td>书籍</td>

<td>12.50</td>

</tr>

<tr>

<td>2</td>

<td>电视</td>

<td>1000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

</tbody>

</table>

</div>

</div>

**5）紧缩表格**

通过添加 .table-condensed 类可以让表格更加紧凑，单元格中的 padding 均会减半。

<div class="container">

<div class="row">

<table class="table table-condensed">

<thead>

<tr>

<th>ID</th>

<th>名称</th>

<th>价格</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td>书籍</td>

<td>12.50</td>

</tr>

<tr>

<td>2</td>

<td>电视</td>

<td>1000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

<tr>

<td>3</td>

<td>笔记本</td>

<td>2000</td>

</tr>

</tbody>

</table>

</div>

</div>

**6）状态类**

通过这些状态类可以为行或单元格设置颜色。

| 类名 | 描述 | | active | 鼠标悬停在行或单元格上时所设置的颜色 | | success | 标识成功或积极的动作 | | info | 标识普通的提示信息或动作 | | warning | 标识警告或需要用户注意 | | danger | 标识危险或潜在的带来负面影响的动作 |

<div class="container">

<div class="row">

<table class="table">

<thead>

<tr>

<th>ID</th>

<th>名称</th>

<th>价格</th>

</tr>

</thead>

<tbody>

<tr class="active">

<td>1</td>

<td>书籍</td>

<td>12.50</td>

</tr>

<tr class="success">

<td>2</td>

<td>电视</td>

<td>1000</td>

</tr>

<tr>

<td class="info">3</td>

<td>笔记本</td>

<td class="warning">2000</td>

</tr>

<tr>

<td>3</td>

<td class="danger">笔记本</td>

<td>2000</td>

</tr>

</tbody>

</table>

</div>

</div>

**值得注意的是：**

这些状态类不仅可以作用于<tr>元素，还可以作用于<td>元素。

**7）响应式表格**

将任何 .table 元素包裹在 .table-responsive 元素内，即可创建响应式表格。其会在小屏幕设备上（小于768px）水平滚动。当屏幕大于 768px 宽度时，水平滚动条消失。

**3. 辅助样式**

**1）背景颜色**

和文本颜色一样，Bootstrap 允许添加不同的背景颜色表示不同含义，具体如下：

| **类名** | **描述** |
| --- | --- |
| bg-primary | 首选项，使用蓝色 |
| bg-success | 成功，使用浅绿色 |
| bg-info | 信息，使用浅蓝色 |
| bg-warning | 警告，使用黄色 |
| bg-danger | 危险，使用褐色 |

<div class="container">

<div class="row">

<p class="bg-muted">提示信息</p>

<p class="bg-primary">首选项</p>

<p class="bg-success">成功</p>

<p class="bg-info">信息</p>

<p class="bg-warning">警告</p>

<p class="bg-danger">危险</p>

</div>

</div>

**2）浮动**

**a. 浮动**

通过添加 .pull-left 或 .pull-right 类，可以将任意元素向左或向右浮动。

* CSS 代码

.block1 {

width : 300px;

height : 200px;

border : 1px solid black;

background : dodgerblue;

}

.block2 {

width : 500px;

height : 300px;

border : 1px solid black;

background : yellowgreen;

}

* HTML 代码

<div class="container">

<div>

<div class="block1 pull-left"></div>

<div class="block1 pull-right"></div>

</div>

<div class="block2"></div>

</div>

**b. 清除浮动**

通过为父元素添加 .clearfix 类可以很容易地清除浮动。

* CSS 代码

.block1 {

width : 300px;

height : 200px;

border : 1px solid black;

background : dodgerblue;

}

.block2 {

width : 500px;

height : 300px;

border : 1px solid black;

background : yellowgreen;

}

* HTML 代码

<div class="container">

<div class="clearfix">

<div class="block1 pull-left"></div>

<div class="block1 pull-right"></div>

</div>

<div class="block2"></div>

</div>

**3）居中**

为任意元素设置 .center-block 类让其中的内容居中。

<div class="center-block"></div>

**4）显示或隐藏**

.show 和 .hidden 类可以强制任意元素显示或隐藏(对于屏幕阅读器也能起效)，这些类只对块级元素起作用。

另外，.invisible 类可以被用来仅仅影响元素的可见性，也就是说，元素的 display 属性不被改变，并且这个元素仍然能够影响布局。

<div class="show"></div>

<div class="hidden"></div>

<div class="invisible"></div>

<div></div>

**4. 栅格系统**

**1）什么是栅格系统**

栅格系统用于通过一系列的行（row）与列（column）的组合来创建页面布局，内容就可以放入这些创建好的布局中。

Bootstrap 提供的栅格系统，主要特点如下：

* 行的宽度，可以是固定宽度，也可以是相对宽度（100%）。
* 一行中最多可以包含 12 个列。如果一行中包含的列大于 12，多余的列则被作为一个整体另起一行排列。
* Bootstrap 根据 4 种分辨率提供不同的栅格系统预定义样式。

**2）栅格系统工作原理**

Bootstrap 的栅格系统的工作原理如下：

* 栅格系统必须指定一个容器元素，该元素的 class 必须为 .container（固定宽度）或 .container-fluid（100%宽度）。
* 作为“行”的元素必须作为容器元素的直接子元素，并且 class 设置为 .row。
* 作为“列”的元素必须是“行”元素的直接子元素（**“行”元素的子元素不能直接包含内容，内容应该被包含在“列”元素中。**），并且一行最多允许创建 12 列。
* “列”元素的预定义样式，具体内容请参考 *“3）栅格系统的参数”* 内容。

**3）栅格系统的参数**

Bootstrap 根据 4 种不同的分辨率提供了不同的预定义样式（栅格类）。栅格类适用于与屏幕宽度大于或等于分界点大小的设备 ， 并且针对小屏幕设备覆盖栅格类。

|  | **超小屏幕 手机（<768px）** | **小屏幕 平板（>=768px）** | **中等屏幕 桌面显示器（>=992px）** | **大屏幕 大桌面显示器（>=1200px）** |
| --- | --- | --- | --- | --- |
| 栅格系统行为 | 总是水平排列 | 以折叠开始，大于断点成水平方式 | 以折叠开始，大于断点成水平方式 | 以折叠开始，大于断点成水平方式 |
| 容器宽度 | None（自动） | 750px | 970px | 1170px |
| class 前缀 | .col-xs- | .col-sm- | .col-md- | .col-lg- |
| 列数 | 12 | 12 | 12 | 12 |
| 列宽 | 自动 | 60px | 78px | 95px |
| 槽宽 | 30px（每列左右均为 15px） | 30px（每列左右均为 15px） | 30px（每列左右均为 15px） | 30px（每列左右均为 15px） |
| 可嵌套 | 是 | 是 | 是 | 是 |
| 偏移 | 是 | 是 | 是 | 是 |
| 排序 | 是 | 是 | 是 | 是 |

**4）栅格系统示例**

* CSS 代码

body {

margin-top : 100px;

}

[class\*=col-] {

border : 1px solid black;

padding-top: 15px;

padding-bottom: 15px;

}

* HTML 代码

<div class="container">

<h3>三等分列</h3>

<div class="row">

<div class="col-md-4">.col-md-4</div>

<div class="col-md-4">.col-md-4</div>

<div class="col-md-4">.col-md-4</div>

</div>

<h3>三不等分列</h3>

<div class="row">

<div class="col-md-3">.col-md-3</div>

<div class="col-md-6">.col-md-6</div>

<div class="col-md-3">.col-md-3</div>

</div>

<h3>手机、平板电脑和PC桌面不同布局</h3>

<div class="row">

<div class="col-md-3 col-sm-6 col-xs-12">col-md-3 col-sm-6 col-xs-12</div>

<div class="col-md-3 col-sm-6 col-xs-12">col-md-3 col-sm-6 col-xs-12</div>

<div class="col-md-3 col-sm-6 col-xs-12">col-md-3 col-sm-6 col-xs-12</div>

<div class="col-md-3 col-sm-6 col-xs-12">col-md-3 col-sm-6 col-xs-12</div>

</div>

</div>

**5）调整列的顺序**

**a. 列偏移**

使用 .col-md-offset-\* 类可以将列向右侧偏移。（\* 表示向右偏移的列数）

<div class="container">

<h3>列偏移</h3>

<div class="row">

<div class="col-md-4">.col-md-4</div>

<div class="col-md-4 col-md-offset-4">.col-md-4 .col-md-offset-4</div>

</div>

<div class="row">

<div class="col-md-3 col-md-offset-3">.col-md-3 .col-md-offset-3</div>

<div class="col-md-3 col-md-offset-3">.col-md-3 .col-md-offset-3</div>

</div>

<div class="row">

<div class="col-md-6 col-md-offset-3">.col-md-6 .col-md-offset-3</div>

</div>

</div>

**b. 嵌套列**

在 Bootstrap 的栅格系统允许进行嵌套。所谓**栅格系统嵌套**，就是指在 Bootstrap 栅格系统中的每列中允许包含一个栅格系统。

被嵌套的“行”中所包含的列的个数依旧不能超过 12 列。

<div class="container">

<h3>嵌套列</h3>

<div class="row">

<div class="col-sm-9">

Level 1: .col-sm-9

<div class="row">

<div class="col-xs-8 col-sm-6">Level 2: .col-xs-8 .col-sm-6</div>

<div class="col-xs-4 col-sm-6">Level 2: .col-xs-4 .col-sm-6</div>

</div>

</div>

</div>

</div>

**c. 列排序**

通过将栅格系统中的列向右移动（push）或向左移动（pull）来改变列的顺序。

* .col-md-push-\*表示向右移动
* .col-md-pull-\*表示向左移动

<div class="container">

<h3>列排序</h3>

<div class="row">

<div class="col-md-9 col-md-push-3">.col-md-9 .col-md-push-3</div>

<div class="col-md-3 col-md-pull-9">.col-md-3 .col-md-pull-9</div>

</div>

</div>

**6）响应式工具**

通过单独或联合使用以下列出的 class，可以针对不同屏幕尺寸隐藏或显示页面内容。

|  | **超小屏幕 手机（<768px）** | **小屏幕 平板（>=768px）** | **中等屏幕 桌面（>=992px）** | **大屏幕 大桌面（>=1200px）** |
| --- | --- | --- | --- | --- |
| .hidden-xs | 隐藏 | 可见 | 可见 | 可见 |
| .hidden-sm | 可见 | 隐藏 | 可见 | 可见 |
| .hidden-md | 可见 | 可见 | 隐藏 | 可见 |
| .hidden-lg | 可见 | 可见 | 可见 | 隐藏 |

<div class="container">

<h3>响应式工具</h3>

<div class="row">

<div class="col-md-3 col-sm-4 hidden-xs">.col-md-3</div>

<div class="col-md-6 col-sm-8 col-xs-12">.col-md-6</div>

<div class="col-md-3 hidden-sm hidden-xs">.col-md-3</div>

</div>

</div>

**7）响应式栅格系统示例**

<!DOCTYPE html>

<html lang="zh-CN">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Bootstrap栅格系统案例</title>

<!-- Bootstrap -->

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

<!--[if lt IE 9]>

<script src="bootstrap/js/html5shiv.min.js"></script>

<script src="bootstrap/js/respond.min.js"></script>

<![endif]-->

</head>

<body>

<div class="jumbotron">

<div class="container">

<h1>Hello, world!</h1>

<p>This is a template for a simple marketing or informational website. It includes a large callout called a

jumbotron and three supporting pieces of content. Use it as a starting point to create something more

unique.</p>

</div>

</div>

<div class="container">

<!-- Example row of columns -->

<div class="row">

<div class="col-md-4">

<h2>Heading</h2>

<p>Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris

condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis

euismod. Donec sed odio dui. </p>

<p><a class="btn btn-default" href="#" role="button">View details &raquo;</a></p>

</div>

<div class="col-md-4">

<h2>Heading</h2>

<p>Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris

condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis

euismod. Donec sed odio dui. </p>

<p><a class="btn btn-default" href="#" role="button">View details &raquo;</a></p>

</div>

<div class="col-md-4">

<h2>Heading</h2>

<p>Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula

porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut

fermentum massa justo sit amet risus.</p>

<p><a class="btn btn-default" href="#" role="button">View details &raquo;</a></p>

</div>

</div>

<hr>

<footer>

<p>&copy; 2015 Company, Inc.</p>

</footer>

</div>

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

</body>

</html>

**5. 表单**

**1）基本表单**

Bootstrap 为单独的表单元素预定义了全局样式，表单内元素的具体使用如下：

* 设置了 .form-control 类的 <input>、<textarea> 和 <select> 元素都将被默认设置宽度属性为 width: 100%;。
* 将 label 元素和前面提到的控件包裹在 .form-group 中可以获得最好的排列。

<div class="container">

<div class="row">

<form>

<div class="form-group">

<label for="exampleInputEmail1">Email address</label>

<input type="email" class="form-control" id="exampleInputEmail1" placeholder="Email">

</div>

<div class="form-group">

<label for="exampleInputPassword1">Password</label>

<input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password">

</div>

<button type="submit" class="btn btn-default">Submit</button>

</form>

</div>

</div>

**2）内联表单**

为 <form> 元素添加 .form-inline 类可使其内容左对齐并且表现为 inline-block 级别的控件。

只适用于 viewport 至少在 768px 宽度时（视口宽度再小的话就会使表单折叠）。

<div class="container">

<div class="row">

<form class="form-inline">

<div class="form-group">

<label for="exampleInputName2">Name</label>

<input type="text" class="form-control" id="exampleInputName2" placeholder="Jane Doe">

</div>

<div class="form-group">

<label for="exampleInputEmail2">Email</label>

<input type="email" class="form-control" id="exampleInputEmail2" placeholder="jane.doe@example.com">

</div>

<button type="submit" class="btn btn-default">Send invitation</button>

</form>

</div>

</div>

**需要注意的是：**

在内联表单，我们将这些元素的宽度设置为 width: auto;，因此，多个控件可以排列在同一行。

如果你没有为每个输入控件设置 label 标签，屏幕阅读器将无法正确识别。对于这些内联表单，你可以通过为 label 设置 .sr-only 类将其隐藏。

<div class="container">

<div class="row">

<form class="form-inline">

<div class="form-group">

<label class="sr-only" for="exampleInputEmail3">Email address</label>

<input type="email" class="form-control" id="exampleInputEmail3" placeholder="Email">

</div>

<div class="form-group">

<label class="sr-only" for="exampleInputPassword3">Password</label>

<input type="password" class="form-control" id="exampleInputPassword3" placeholder="Password">

</div>

<button type="submit" class="btn btn-default">Sign in</button>

</form>

</div>

</div>

**3）水平排列的表单**

通过为表单添加 .form-horizontal 类，并联合使用 Bootstrap 预置的栅格类，可以将 label 标签和控件组水平并排布局。

这样做将改变 .form-group 的行为，使其表现为栅格系统中的行（row），因此就无需再额外添加 .row 了。

<div class="container">

<div class="row">

<form class="form-horizontal">

<div class="form-group">

<label for="inputEmail3" class="col-sm-2 control-label">Email</label>

<div class="col-sm-10">

<input type="email" class="form-control" id="inputEmail3" placeholder="Email">

</div>

</div>

<div class="form-group">

<label for="inputPassword3" class="col-sm-2 control-label">Password</label>

<div class="col-sm-10">

<input type="password" class="form-control" id="inputPassword3" placeholder="Password">

</div>

</div>

<div class="form-group">

<div class="col-sm-offset-2 col-sm-10">

<button type="submit" class="btn btn-default">Sign in</button>

</div>

</div>

</form>

</div>

</div>

**4）单选框和多选框**

**a. 默认外观**

* 单选框使用 class 为 .radio
* 多选框使用 class 为 .checkbox

单选框或多选框的样式要定义在 div 元素中（该元素作为容器元素，包含文本和单选或多选框元素）。

<div class="container">

<div class="row">

<form>

<div class="checkbox">

<label>

<input type="checkbox" value="">

Option one is this and that&mdash;be sure to include why it's great

</label>

</div>

<div class="checkbox disabled">

<label>

<input type="checkbox" value="" disabled>

Option two is disabled

</label>

</div>

<div class="radio">

<label>

<input type="radio" name="optionsRadios" id="optionsRadios1" value="option1" checked>

Option one is this and that&mdash;be sure to include why it's great

</label>

</div>

<div class="radio">

<label>

<input type="radio" name="optionsRadios" id="optionsRadios2" value="option2">

Option two can be something else and selecting it will deselect option one

</label>

</div>

<div class="radio disabled">

<label>

<input type="radio" name="optionsRadios" id="optionsRadios3" value="option3" disabled>

Option three is disabled

</label>

</div>

</form>

</div>

</div>

**b. 内联单选和多选框**

通过将 .checkbox-inline 或 .radio-inline 类应用到一系列的多选框（checkbox）或单选框（radio）控件上，可以使这些控件排列在一行。

<div class="container">

<div class="row">

<form>

<label class="checkbox-inline">

<input type="checkbox" id="inlineCheckbox1" value="option1"> 1

</label>

<label class="checkbox-inline">

<input type="checkbox" id="inlineCheckbox2" value="option2"> 2

</label>

<label class="checkbox-inline">

<input type="checkbox" id="inlineCheckbox3" value="option3"> 3

</label>

<label class="radio-inline">

<input type="radio" name="inlineRadioOptions" id="inlineRadio1" value="option1"> 1

</label>

<label class="radio-inline">

<input type="radio" name="inlineRadioOptions" id="inlineRadio2" value="option2"> 2

</label>

<label class="radio-inline">

<input type="radio" name="inlineRadioOptions" id="inlineRadio3" value="option3"> 3

</label>

</form>

</div>

</div>

**组件**

**1. 字体图标**

Bootstrap 使用的字体图标是来自 Glyphicions Halflings 提供的。**在使用的时候，需要遵守以下几点规则：**

* 图标使用的 class 不能与其他组件的 class 混合使用。换言之，图标的 class 必须被定义在 span 元素上。
* 图标使用的 class 具有一个基类和一个修饰类。基类（class）统一为 .glyphicon，修饰类（class）为 .glyphicon-\*-\*或.glyphicon-\*。
* 只对内容为空的元素起作用（图标使用的 class 只能应用在不包含任何文本内容或子元素的元素上）。

<div class="container">

<p class="text-danger">

<span class="glyphicon glyphicon-warning-sign"></span>

<span class="glyphicon glyphicon-alert"></span>

警告：您的浏览器版本太低了!

</p>

<button class="btn btn-success" type="button">

<span class="glyphicon glyphicon-apple"></span>

删除

</button>

<button class="btn btn-danger" type="button">

<span class="glyphicon glyphicon-arrow-left"></span>

</button>

<button class="btn btn-default" type="button">

<span class="glyphicon glyphicon-star"></span>

</button>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-play"></span>

</a>

</div>

**2. 下拉菜单**

**1）基本样式**

Bootstrap 提供的下拉菜单并不是 HTML 默认的 <select> 元素，而且利用一组 HTML 元素组合而成。

* <div>元素作为容器元素
* <button>或<a>元素作为下拉菜单的提示项。

作为下拉菜单的提示项，Bootstrap 默认显示的是按钮样式（也就是需要设置 class 为 btn btn-\*）。

按钮在被点击后，会提供不同的样式效果。如果想改变这样的样式，需要添加样式dropdown-toggle。

通过为下拉菜单的提示项设置属性 data-toggle=dropdown ，实现下拉菜单的动态显示。（此内容会在 **组件** 详细讲解）

* 无序列表作为下拉菜单的列表项。

**a. 向下弹出**

为作为下拉菜单的容器元素设置样式dropdown即可。

<div class="container">

<div class="dropdown">

<button class="btn btn-default" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<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>

<li><a href="#">Separated link</a></li>

</ul>

</div>

</div>

**b. 向上弹出**

为作为下拉菜单的容器元素设置样式dropup即可。

<div class="container">

<div class="dropup">

<button class="btn btn-default" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<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>

<li><a href="#">Separated link</a></li>

</ul>

</div>

</div>

**2）标题**

可以为作为下拉菜单的列表项中，添加 class 为 dropdown-header 的 <li> 元素作为标题，表示一组动作。

<div class="container">

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li class="dropdown-header">Dropdown header</li>

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

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

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

<li class="dropdown-header">Dropdown header</li>

<li><a href="#">Separated link</a></li>

</ul>

</div>

</div>

**3）分割线**

可以为作为下来菜单的列表项中，添加 class 为 divider 的 <li> 元素作为分割线，表示将多个选项进行分组。

<div class="container">

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<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>

<li class="divider"></li>

<li><a href="#">Separated link</a></li>

</ul>

</div>

</div>

**4）禁用项**

为下拉菜单中的 <li> 元素添加 .disabled 类，从而禁用相应的菜单项。

<div class="container">

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li><a href="#">Regular link</a></li>

<li class="disabled"><a href="#">Disabled link</a></li>

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

</ul>

</div>

</div>

**3. 按钮组**

**1）基本按钮组**

通过按钮组容器把一组按钮放在同一组里，为按钮组容器设置 class 为 btn-group。

<div class="container">

<div class="btn-group">

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-ok-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-remove-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-pencil"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-check"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-question-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-edit"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-search"></span>

</a>

</div>

</div>

**2）不同布局**

**a. 两端对齐**

让一组按钮拉长为相同的尺寸，填满父元素的宽度。为按钮组容器添加样式 btn-group-justified。

<div class="container">

<div class="btn-group btn-group-justified">

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-ok-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-remove-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-pencil"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-check"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-question-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-edit"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-search"></span>

</a>

</div>

</div>

**b. 垂直排列**

让一组按钮垂直堆叠排列显示而不是水平排列，为按钮组容器设置 class 为 btn-group-vertical。

<div class="container">

<div class="btn-group-vertical">

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-ok-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-remove-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-pencil"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-check"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-question-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-edit"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-search"></span>

</a>

</div>

</div>

**3）尺寸**

只要给 .btn-group 加上 .btn-group-\* 类，就省去为按钮组中的每个按钮都赋予尺寸类。

Bootstrap 提供了以下几种尺寸：

| **class 名称** | **描述** |
| --- | --- |
| btn-group-lg | 大 |
| btn-group-sm | 小 |
| btn-group-xs | 很小 |

除上述尺寸外，Bootstrap 还提供一种默认尺寸（无需设置）。

<div class="container">

<div class="btn-group btn-group-lg">

<a class="btn btn-info" href="#">Left</a>

<a class="btn btn-info" href="#">Middle</a>

<a class="btn btn-info" href="#">Right</a>

</div>

<div class="btn-group">

<a class="btn btn-info" href="#">Left</a>

<a class="btn btn-info" href="#">Middle</a>

<a class="btn btn-info" href="#">Right</a>

</div>

<div class="btn-group btn-group-sm">

<a class="btn btn-info" href="#">Left</a>

<a class="btn btn-info" href="#">Middle</a>

<a class="btn btn-info" href="#">Right</a>

</div>

<div class="btn-group btn-group-xs">

<a class="btn btn-info" href="#">Left</a>

<a class="btn btn-info" href="#">Middle</a>

<a class="btn btn-info" href="#">Right</a>

</div>

</div>

**4）按钮工具栏**

把一组 <div class="btn-group"> 组合包裹在一个 <div class="btn-toolbar"> 中就可以做成更复杂的组件。

<div class="container">

<div class="btn-toolbar">

<div class="btn-group">

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-ok-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-remove-sign"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-pencil"></span>

</a>

</div>

<div class="btn-group">

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-check"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-question-sign"></span>

</a>

</div>

<div class="btn-group">

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-edit"></span>

</a>

<a class="btn btn-info" href="#">

<span class="glyphicon glyphicon-search"></span>

</a>

</div>

</div>

</div>

**5）嵌套下拉菜单**

想要把下拉菜单混合到一系列按钮中，只须把 .btn-group 放入另一个 .btn-group 中。

<div class="container">

<div class="btn-group">

<button type="button" class="btn btn-default">1</button>

<button type="button" class="btn btn-default">2</button>

<div class="btn-group" role="group">

<button class="btn btn-default dropdown-toggle" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li><a href="#">Dropdown link</a></li>

<li><a href="#">Dropdown link</a></li>

</ul>

</div>

</div>

</div>

**4. 警告框**

警告框组件通过提供一些灵活的预定义消息，为常见的用户动作提供反馈消息。

**1）基本警告框**

Bootstrap 的警告框样式由 .alert 和 4 个修饰class组成。4 个修饰class表示不同的警告信息。

| **class 名称** | **描述** |
| --- | --- |
| alert-success | 成功 |
| alert-info | 信息 |
| alert-warning | 警告 |
| alert-danger | 危险 |

<div class="container">

<div class="alert alert-danger">

<span class="glyphicon glyphicon-alert"></span>

您的浏览器禁用了Cookie！本站的部分功能无法启用！

</div>

<div class="alert alert-info">

<span class="glyphicon glyphicon-alert"></span>

没有更多数据可供加载了！

</div>

<div class="alert alert-warning">

<span class="glyphicon glyphicon-alert"></span>

您的浏览器OUT啦！请使用更新版本的浏览器!

</div>

<div class="alert alert-success">

<span class="glyphicon glyphicon-alert"></span>

您已经登录了！无需再次登录！

</div>

</div>

**2）可关闭的警告框**

为警告框添加一个可选的 .alert-dismissible 类和一个关闭按钮。

<div class="container">

<div class="alert alert-success alert-dismissible">

<!--.close元素必须是alert的第一个子元素-->

<span data-dismiss="alert" class="close">&times;</span>

<span class="glyphicon glyphicon-alert"></span>

您已经登录了！无需再次登录！

</div>

</div>

**3）警告框中的链接**

用 .alert-link 工具类，可以为链接设置与当前警告框相符的颜色。

<div class="container">

<div class="alert alert-warning">

<span class="glyphicon glyphicon-alert"></span>

您的浏览器OUT啦！请使用更新版本的<a href="#" class="alert-link">浏览器</a>!

</div>

</div>

**本笔记由 *金云龙* 编写**

作者博客地址：[龙哥有话说](http://www.longestory.com)

如转载或发布，请注明作者名称

**BOOTSTRAP第三天笔记**

**组件**

**1. 导航**

Bootstrap 中的导航组件都依赖同一个 .nav 类。

**1）两种导航样式**

Bootstrap 提供标签页主要利用的是 **无序列表** 元素。

**a. 标签页式导航**

标签页式导航，就是为作为标签页的<ul>元素添加 class 为 .nav-tab 的样式。

**需要注意的是：**

* .nav-tabs 类依赖 .nav 基类。
* 默认被点击的样式为 active。
* 实现动态切换效果，需要为作为导航项的元素设置 data-toggle 为 tab。（具体内容会在 **组件** 详细讲解）

<div class="container">

<h3>标签页式导航</h3>

<ul class="nav nav-tabs">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</div>

**b. 胶囊式标导航**

胶囊式导航，就是为作为标签页的<ul>元素添加 class 为 .nav-pills 的样式。

**水平方向**

胶囊式导航的默认方向，就是水平方向。

<div class="container">

<h3>胶囊式导航</h3>

<ul class="nav nav-pills">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</div>

**垂直方向**

想要将胶囊式导航设置为垂直方向，只需为作为标签页的<ul>元素添加 class 为 .nav-stacked 的样式。

<div class="container">

<h3>垂直方向的胶囊式导航</h3>

<ul class="nav nav-pills nav-stacked">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</div>

**2）两端对齐**

在大于 768px 的屏幕上，通过 .nav-justified 类可以很容易的让标签页或胶囊式导航呈现出同等宽度。在小屏幕上，导航链接以垂直方向呈现。

<div class="container">

<ul class="nav nav-tabs nav-justified">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

<ul class="nav nav-pills nav-justified">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</div>

**3）嵌套下拉菜单**

导航中也可以嵌入下拉菜单。

<div class="container">

<ul class="nav nav-pills">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Help</a></li>

<li data-toggle="tab" class="dropdown">

<a class="dropdown-toggle" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</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>

<li><a href="#">Separated link</a></li>

</ul>

</li>

</ul>

</div>

**2. 导航条**

导航条是在应用或网站中作为导航页头的响应式基础组件。

**1）导航条样式**

Bootstrap 提供的导航条使用 HTML5 的语义化元素 <nav> 作为容器元素，并为该元素设置 class 为 .navbar。

**a. 默认样式的导航条**

默认样式的导航条，需要为作为容器元素的 <nav> 元素，添加 .navbar-default 样式。

<div class="container">

<nav class="navbar navbar-default">

<!-- 导航条组件 -->

</nav>

</div>

**b. 反色的导航条**

通过添加 .navbar-inverse 类可以改变导航条的外观。

<div class="container">

<nav class="navbar navbar-inverse">

<!-- 导航条组件 -->

</nav>

</div>

**2）导航条组件**

**a. 品牌图标**

一般在导航条添加网站的名称或图标，使用 <a> 元素，并且为其设置 class 为 .navbar-brand。

<div class="container">

<nav class="navbar navbar-default">

<a class="navbar-brand" href="#">

<img src="imgs/logo.png" alt="Brand" width="20">

</a>

</nav>

</div>

**需要注意的是：**

* .navbar-brand样式设置的高度为 50px，并且内边距为 15px。所以，如果使用图标的话，一定设置图标的高度为 20px。

**b. 表单**

在导航条添加表单的话，需要为 <form> 元素设置 class 为 navbar-form。表单内组件可以呈现很好的垂直对齐，并且较窄的宽度中呈现折叠状态。

<div class="container">

<nav class="navbar navbar-default">

<form class="navbar-form">

<div class="form-group">

<input type="text" class="form-control" placeholder="Search">

</div>

<button type="submit" class="btn btn-default">Submit</button>

</form>

</nav>

</div>

**c. 按钮**

如果要在导航条中添加非表单内的按钮时，需要为其添加 navbar-btn 样式，使之在导航条中垂直居中。

<div class="container">

<nav class="navbar navbar-default">

<button class="btn btn-default navbar-btn">Sign in</button>

</nav>

</div>

**d. 文本**

在导航条中包含普通的文本内容，需要使用 <p> 元素，并且为其设置 class 为 navbar-text，使其具有正确的行距和颜色。

<div class="container">

<nav class="navbar navbar-default">

<p class="navbar-text">Signed in as Mark Otto</p>

</nav>

</div>

**e. 导航**

在导航条中最主要的功能就是集成导航，需要为作为导航的 <ul> 元素设置 class 为 navbar-nav。

<div class="container">

<nav class="navbar navbar-default">

<ul class="nav navbar-nav">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</nav>

</div>

**f. 组件排列**

通过添加 .navbar-left 和 .navbar-right 工具类让导航链接、表单、按钮或文本对齐。

<div class="container">

<nav class="navbar navbar-default">

<ul class="nav navbar-nav navbar-left">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

<form class="navbar-form navbar-right">

<div class="form-group">

<input type="text" class="form-control" placeholder="Search">

</div>

<button type="submit" class="btn btn-default">Submit</button>

</form>

</nav>

</div>

**3）固定的导航条**

固定的导航条不会随着页面滚动而消失。

**a. 固定在顶部**

为 <nav> 元素添加 .navbar-fixed-top 类可以让导航条固定在顶部，并且导航条的宽度与页面宽度一致。

<div class="container">

<nav class="navbar navbar-default navbar-fixed-top">

<!-- 导航条组件 -->

</nav>

</div>

**b. 固定在底部**

为 <nav> 元素添加 .navbar-fixed-bottom 类可以让导航条固定在底部，并且导航条的宽度与页面宽度一致。

<div class="container">

<nav class="navbar navbar-default navbar-fixed-bottom">

<!-- 导航条组件 -->

</nav>

</div>

**4）响应式导航条**

Bootstrap 以 768px 宽度为分界点，分别进行设置导航条的内容及样式。

| **class 名称** | **描述** |
| --- | --- |
| navbar-header | 针对宽度小于 768px 的屏幕 |
| navbar-collapse | 针对宽度大于等于 768px 的屏幕 |

**针对小于 768px 屏幕的导航条**

* 使用 <div> 作为容器元素，并为其设置 class 为 navbar-header。
* 一般使用 <button> 或 <a> 元素，并设置 class 为 navbar-toggle。
* 通过为 <button> 或 <a> 元素设置属性 data-toggle 为 collapse，实现点击交互效果。
* 通过为 <button> 或 <a> 元素设置属性 data-target 为 对应菜单容器元素的 id 属性值。
* 在 <button> 或 <a> 元素内添加三个 <span class="icon-bar"></span> 元素（汉堡包按钮样式）。

<div class="container">

<nav class="nav navbar-default">

<div class="navbar-header">

<button class="navbar-toggle" data-toggle="collapse" data-target="#navbar-collapse">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

</div>

<!-- 导航条组件 -->

</nav>

</div>

**针对大于等于 768px 屏幕的导航条**

使用 <div> 作为容器元素，并为其设置 class 为 collapse navbar-collapse 和 id 属性（**用于与针对小于 768px 屏幕的button元素的data-target属性值对应。**）。

<div class="container">

<nav class="nav navbar-default">

<!-- 针对宽度小于 768px 的汉堡包按钮 -->

<div class="collapse navbar-collapse" id="navbar-collapse">

<ul class="nav navbar-nav">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</div>

</nav>

</div>

**3. 媒体对象**

Bootstrap 媒体对象常用于用户的评论、帖子或商品列表。

* 使用 <div> 作为媒体对象的容器元素，并设置 class 为 .media 。
* 媒体对象提供左（media-left）、主体（media-body）和右（media-right）三种结构。
* 设置垂直方向的上（media-top）、中（media-middle）和下（media-bottom）三种。
* 如果使用图片的话，需要设置为media-object。

<div class="container">

<h3>左中结构</h3>

<div class="media">

<div class="media-left">

<a href="#">

<img src="imgs/1.jpg" class="media-object">

</a>

</div>

<div class="media-body">

<h4>Media heading</h4>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Beatae debitis dicta repellat repellendus vel? Ab assumenda, doloribus eaque enim iure mollitia, non officia recusandae repellat reprehenderit repudiandae sunt velit, veritatis?</p>

</div>

</div>

<hr>

<h3>中右结构</h3>

<div class="media">

<div class="media-body media-middle">

<h4>Media heading</h4>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Beatae debitis dicta repellat repellendus vel? Ab assumenda, doloribus eaque enim iure mollitia, non officia recusandae repellat reprehenderit repudiandae sunt velit, veritatis?</p>

</div>

<div class="media-right">

<a href="#">

<img src="imgs/1.jpg" class="media-object">

</a>

</div>

</div>

<hr>

<h3>左中右结构</h3>

<div class="media">

<div class="media-left">

<a href="#">

<img src="imgs/1.jpg" class="media-object">

</a>

</div>

<div class="media-body media-bottom">

<h4>Media heading</h4>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Beatae debitis dicta repellat repellendus vel? Ab assumenda, doloribus eaque enim iure mollitia, non officia recusandae repellat reprehenderit repudiandae sunt velit, veritatis?</p>

</div>

<div class="media-right">

<a href="#">

<img src="imgs/1.jpg" class="media-object">

</a>

</div>

</div>

</div>

**4. 列表组**

**1）基本列表组**

默认的列表组使用无序列表实现。

<ul> 元素作为列表组的容器元素，并为其设置 class 为 list-group。<li> 元素作为列表组的列表项，并为其设置 class 为list-group-item。

也可以将 <ul> 元素替换为 <div> 元素，将 <li> 元素替换为 <button> 或 <a> 元素。

<div class="container">

<ul class="list-group">

<li class="list-group-item">Cras justo odio</li>

<li class="list-group-item">Dapibus ac facilisis in</li>

<li class="list-group-item">Morbi leo risus</li>

<li class="list-group-item">Porta ac consectetur ac</li>

<li class="list-group-item">Vestibulum at eros</li>

</ul>

</div>

**2）徽章**

给链接、导航等元素嵌套 <span class="badge"> 元素，可以很醒目的展示新的或未读的信息条目。

<div class="container">

<ul class="list-group">

<li class="list-group-item">

<span class="badge">14</span>

Cras justo odio

</li>

<li class="list-group-item">

<span class="badge">2</span>

Dapibus ac facilisis in

</li>

<li class="list-group-item">Morbi leo risus</li>

</ul>

</div>

**5. 路径导航**

在一个带有层次的导航结构中标明当前页面的位置。

<div class="container">

<ol class="breadcrumb">

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

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

<li class="active">Data</li>

</ol>

</div>

**6. 标签**

为具有特定含义的文本设置为标签样式。标签具有一个基类（label）和一个修饰类（label-\*），修饰类预定义以下几种 class 即可改变标签的外观。

| **class 名称** | **描述** |
| --- | --- |
| label-default | 默认样式 |
| label-primary | 首选项样式 |
| label-success | 成功样式 |
| label-info | 信息样式 |
| label-warning | 警告样式 |
| label-danger | 危险样式 |

<div class="container">

<span class="label label-default">Default</span>

<span class="label label-primary">Primary</span>

<span class="label label-success">Success</span>

<span class="label label-info">Info</span>

<span class="label label-warning">Warning</span>

<span class="label label-danger">Danger</span>

</div>

**7. Well**

把 Well 用在元素上，能有嵌入（inset）的简单效果。一般用于文章的导读等功能。

Well 由一个基类（well）和一个修饰类（well-\*）组成。

| **class 名称** | **描述** |
| --- | --- |
| well-lg | 大边距样式 |
| well-sm | 小边距样式 |

<div class="container">

<div class="well">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Animi commodi, corporis debitis doloremque harum illo ipsam ipsum itaque labore minima non obcaecati qui quis quos recusandae reiciendis sunt unde ut.</div>

<div class="well well-lg">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus ad assumenda atque beatae debitis ducimus esse explicabo fuga, id ipsum maiores nobis non odio pariatur quae quos rerum sequi voluptas.</div>

</div>

**8. 巨幕**

这是一个轻量、灵活的组件，它能延伸至整个浏览器视口来展示网站上的关键内容。

<div class="container">

<div class="jumbotron">

<h1>Hello, world!</h1>

<p>This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.</p>

<p><a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a></p>

</div>

</div>

**9. 页头**

页头组件能够为 h1 标签增加适当的空间，并且与页面的其他部分形成一定的分隔。

<div class="page-header">

<h1>Example page header <small>Subtext for header</small></h1>

</div>

**10. 分页**

**1）分页**

Bootstrap 提供的分页由以下两组元素组成：

* <nav>元素作为分页的容器元素。
* 使用无序列表作为分页的成员元素，并为 <ul> 元素设置 class 为 pagination。

<div class="container">

<nav>

<ul class="pagination">

<li>

<a href="#" aria-label="Previous">

<span aria-hidden="true">&laquo;</span>

</a>

</li>

<li><a href="#">1</a></li>

<li><a href="#">2</a></li>

<li><a href="#">3</a></li>

<li><a href="#">4</a></li>

<li><a href="#">5</a></li>

<li>

<a href="#" aria-label="Next">

<span aria-hidden="true">&raquo;</span>

</a>

</li>

</ul>

</nav>

</div>

另外，通过为 <ul> 元素添加 .pagination-lg 或 .pagination-sm 类提供了额外可供选择的尺寸。

**2）翻页**

用简单的标记和样式，就能做个上一页和下一页的简单翻页。用在像博客和杂志这样的简单站点上效果会很好。

<div class="container">

<nav>

<ul class="pager">

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

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

</ul>

</nav>

</div>

**插件**

**1. 概览**

Bootstrap 提供相应插件在 HTML 页面中实现动态交互效果。

**1）前提条件**

由于 Bootstrap 提供的插件都是依赖于 jQuery 的，所以必须要先引入 jQuery 文件。

Bootstrap 的每个插件都对应具有一个 JavaScript 文件，允许单独引入到 HTML 页面。也提供了一个完整版本（Bootstrap.js 或 Bootstrap.min.js 文件），允许一次性将所有插件全部引入到 HTML 页面中。

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

**2）两种使用方式**

**a. data 属性**

通过 data 属性 API 就能使用所有的 Bootstrap 插件，无需写一行 JavaScript 代码。

<div class="container">

<div class="dropdown">

<a data-toggle="dropdown" class="btn btn-default" href="#">产品大全</a>

<ul class="dropdown-menu">

<li><a href="#">冰箱</a></li>

<li><a href="#">洗衣机</a></li>

<li><a href="#">电视</a></li>

</ul>

</div>

</div>

如果在某些情况下可能需要将此功能关闭，提供了关闭 data 属性 API 的方法（**即解除以 data-api 为命名空间并绑定在文档上的事件**）。

$(document).off('.data-api')

**b. JavaScript API**

所有 Bootstrap 插件提供了纯 JavaScript 方式的 API。所有公开的 API 都是支持单独或链式调用方式，并且返回其所操作的元素集合(**和jQuery的调用形式一致**)。

<div class="container">

<div class="dropdown">

<button id="btn2" class="btn btn-default">产品大全</button>

<ul class="dropdown-menu">

<li><a href="">冰箱</a></li>

<li><a href="">洗衣机</a></li>

<li><a href="">电视</a></li>

</ul>

</div>

</div>

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

<script>

//var num = 1;

$('#btn2').dropdown();

</script>

**2. 下拉菜单**

**1）通过 data-\* 属性方式实现**

<div class="container">

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" data-toggle="dropdown">

Dropdown

<span class="caret"></span>

</button>

<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>

<li><a href="#">Separated link</a></li>

</ul>

</div>

</div>

**2）通过 JavaScript 编程方式实现**

<div class="container">

<div class="dropdown">

<button class="btn btn-default dropdown-toggle">

Dropdown

<span class="caret"></span>

</button>

<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>

<li><a href="#">Separated link</a></li>

</ul>

</div>

</div>

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

<script>

$('button[class\*=btn]').dropdown();

</script>

**3. 警告框**

**1）通过 data-\* 属性方式实现**

<div class="container">

<div class="alert alert-danger alert-dismissible">

<span data-dismiss="alert" class="close">&times;</span>

<h4><span class="glyphicon glyphicon-alert"></span>警告！</h4>

<p>您的浏览器版本太老了！请更新到最新版本的浏览器！</p>

</div>

</div>

**2）通过 JavaScript 编程方式实现**

<div class="container">

<div class="alert alert-danger alert-dismissible">

<span id="btn\_close" class="close">&times;</span>

<h4><span class="glyphicon glyphicon-alert"></span>警告！</h4>

<p>您的浏览器版本太老了！请更新到最新版本的浏览器！</p>

</div>

</div>

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

<script>

//$('#btn\_close').alert(); //如此书写，将直接消失

$('#btn\_close').click(function(){

$(this).alert('close');

});

</script>

**4. 标签页**

**1）通过 data-\* 属性方式实现**

<div class="container">

<ul class="nav nav-tabs">

<li data-toggle="tab" class="active"><a href="#">Home</a></li>

<li data-toggle="tab"><a href="#">Profile</a></li>

<li data-toggle="tab"><a href="#">Messages</a></li>

</ul>

</div>

**2）通过 JavaScript 编程方式实现**

<div class="container">

<ul class="nav nav-tabs" id="mytab">

<li class="active"><a href="#">Home</a></li>

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

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

</ul>

</div>

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

<script>

$('#mytab a').click(function(event){

event.preventDefault();

$(this).tab('show');

})

</script>

**5. 折叠框**

**1）通过 data-\* 属性方式实现**

<div class="container">

<a class="btn btn-primary" role="button" data-toggle="collapse" href="#collapseExample">

Link with href

</a>

<button class="btn btn-primary" data-toggle="collapse" data-target="#collapseExample">

Button with data-target

</button>

<div class="collapse" id="collapseExample">

<div class="well">

...

</div>

</div>

</div>

**2）通过 JavaScript 编程方式实现**

<div class="container">

<a id="href" class="btn btn-primary" href="#collapse2">

Link with href

</a>

<button id="btn" class="btn btn-primary" data-target="#collapse2">

Button with data-target

</button>

<div class="collapse" id="collapse2">

<div class="well">

...

</div>

</div>

</div>

<script src="bootstrap/js/jquery-1.11.3.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

<script>

$('#href,#btn').click(function(){

$("#collapse2").collapse('toggle');

});

</script>

**3）实现手风琴组件效果**

**a. panel 组件**

* 通过 <div> 元素作为 panel 组件的容器元素，并设置 class 为 panel，以及一个修饰类。
* panel 组件是由一个标题（panel-heading）和一个主体（panel-body）组成。

<div class="container">

<div class="panel panel-default">

<div class="panel-heading">

<h4 class="panel-title">Collapsible Group</h4>

</div>

<div class="panel-body">

...

</div>

</div>

</div>

**b. 动态的 panel 组件**

想要实现动态的 panel 组件，需要 .collapse 和 .panel 两种组件组合而成。

<div class="container">

<div class="panel panel-default">

<div class="panel-heading" id="headingOne">

<h4 class="panel-title">

<a role="button" data-toggle="collapse" href="#collapseOne">

Collapsible Group Item #1

</a>

</h4>

</div>

<div id="collapseOne" class="panel-collapse collapse in">

<div class="panel-body">

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

</div>

</div>

</div>

</div>

**c. panel-group 组成手风琴**

将 .panel-group 元素作为手风琴的容器元素，与 .panel 组件相关联。

<div class="container">

<div class="panel-group" id="accordion">

<div class="panel panel-default">

<div class="panel-heading" id="headingOne">

<h4 class="panel-title">

<a role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseOne">

Collapsible Group Item #1

</a>

</h4>

</div>

<div id="collapseOne" class="panel-collapse collapse in">

<div class="panel-body">

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" id="headingTwo">

<h4 class="panel-title">

<a class="collapsed" data-toggle="collapse" data-parent="#accordion" href="#collapseTwo">

Collapsible Group Item #2

</a>

</h4>

</div>

<div id="collapseTwo" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingTwo">

<div class="panel-body">

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" id="headingThree">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseThree">

Collapsible Group Item #3

</a>

</h4>

</div>

<div id="collapseThree" class="panel-collapse collapse">

<div class="panel-body">

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

</div>

</div>

</div>

</div>

</div>

**本笔记由 *金云龙* 编写**

作者博客地址：[龙哥有话说](http://www.longestory.com)

如转载或发布，请注明作者名称