Bootstrap整体架构

# 栅格系统

## 实现原理

|  |
| --- |
| 1、数据行(.row)必须包含在容器（.container）中，以便为其赋予合适的对齐方式和内距(padding)。如：  <div class=“container”>  <div class=“row”></div>  </div>  2、在行(.row)中可以添加列(.column)，但列数之和不能超过平分的总列数，比如12。如：  <div class=“container”>  <div class=“row”>  <div class=“col-md-4”></div>  <div class=“col-md-8”></div>  </div>  3、具体内容应当放置在列容器（column）之内，而且只有列（column）才可以作为行容器(.row)的直接子元素  4、通过设置内距（padding）从而创建列与列之间的间距。然后通过为第一列和最后一列设置负值的外距（margin）来抵消内距(padding)的影响  5、如果一“行（row）”中包含了的“列（column）”大于 12，多余的“列（column）”所在的元素将被作为一个整体另起一行排列。  http://img.mukewang.com/53b0f9c000018b9305540282.jpg  简单对图解释一下：  1、最外边框，带有一大片白色区域，就是相当于浏览器的可视区域。在Bootstrap框架的网格系统中带有响应式效果，其带有四种类型的浏览器（超小屏，小屏，中屏和大屏），其断点（像素的分界点）是768px、992px和1220px。  2、第二个边框(1)相当于容器(.container)。针对不同的浏览器分辨率，其宽度也不一样：自动、750px、970px和1170px。  3、２号横条阐述的是，将容器的行（.row）平分了12等份，也就是列。每个列都有一个”padding-left:15px”(图中粉红色部分)和一个”padding-right:15px”(图中紫色部分)。这样也导致了第一个列的padding-left和最后一列的padding-right占据了总宽度的30px，从而致使页面不美观，当然，如果你需要留有一定的间距，这个做法是不错的。  4、３号横条就是行容器(.row),其定义了“margin-left”和”margin-right”值为”-15px”，用来抵消第一个列的左内距和最后一列的右内距。  5、将行与列给合在一起就能看到横条4的效果。也就是我们期望看到的效果，第一列和最后一列与容器（.container）之间没有间距。 |

## 基本用法

### 列组合

|  |
| --- |
| <div class="container">  <div class="row">  <div class="col-md-8">.col-md-8</div>  <div class="col-md-4">.col-md-4</div>  </div>  <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>  </div> |

### 列偏移

|  |
| --- |
| <div class="container">  <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>  // col-md-offset-4样式会先将列向右移动4格 |

### 列嵌套

|  |
| --- |
| 为了使用内置的栅格系统将内容再次嵌套，可以通过添加一个新的 .row 元素和一系列 .col-sm-\* 元素到已经存在的 .col-sm-\* 元素内。被嵌套的行（row）所包含的列（column）的个数不能超过12  <div class="container">  **<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> |

### 列排序

|  |
| --- |
| 通过使用 .col-md-push-\* 和 .col-md-pull-\* 类就可以很容易的改变列（column）的顺序。push向右移动，pull向左移动  <div class="container">  <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> |

## 响应式栅格

Bootstrap为不同的屏幕尺寸提供了不同的栅格样式，xs(超小),sm(小型),md(中型)， lg(大型)



### 跨设备组合定义

|  |
| --- |
| <div class="row">  <div class="col-xs-12 col-md-8">.col-xs-12 .col-md-8</div>  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>  </div>  <div class="row">  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>  </div>  <div class="row">  <div class="col-xs-6">.col-xs-6</div>  <div class="col-xs-6">.col-xs-6</div>  </div>    //如果只用min-width,则表示向大兼容。向小是不兼容的 |

### 清除浮动问题

|  |
| --- |
| <div class="container">  <div class="row">  <div class="col-xs-6 col-sm-3">  .col-xs-6 .col-sm-3 <br>  Resize your viewport or check it out on your phone for an example.  </div>  <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>    <div class="clearfix visible-xs-block"></div>    <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>  <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>  </div>  </div>  //由于col-默认都是左浮动的，当div1中的内容过多时候，导致div3在排列的时候直接浮动在div1后面。应该使用清除浮动：  **<div class="clearfix visible-xs-block"></div>** |

# CSS组件架构的设计思想

## 基础样式

|  |
| --- |
| 任何一个CSS组件在刚开始的时候都先定义了基本样式。例如：btn,alert |

## 颜色样式

|  |
| --- |
| Bootstrap为很多组件提供了5种颜色样式：primary(重点蓝)，success(成功绿)，info(信息蓝)，warning(警告橙)，danger(危险红)  例如：btn-info,btn-success |

## 尺寸样式

|  |
| --- |
| Bootstrap为大部分组件都提供了尺寸的快捷设置。一般组件的都有4个尺寸：超小(xm)，小型(sm)，普通，大型(lg)-->加大内边距，加大圆角  例如：btn-lg，btn-sm |

## 状态样式

|  |
| --- |
| // active,disabled  例如：active,disabled |

## 特殊元素样式

|  |
| --- |
| 所谓特殊元素样式，即特定类型的组件一般使用某一种或者几种固定的元素  例如：.alert .alert-link |

## 并列元素样式

|  |
| --- |
| 在很多情况，一个组件内部需要放置多个子元素，比如导航组件(nav)里可以放置多个li元素，按钮组里面可以放置多个buttun元素。如果有这种情况，就需要处理这样并列元素的间距问题。  例如：.alert >p |

## 嵌套子元素样式

|  |
| --- |
| 有的时候我们也需要将两个相同或不同的组件嵌套在一起，这是经常会出现一些特殊的情况。 例如：.btn-group .btn + .tbn |

## 动画样式

|  |
| --- |
| Bootstrap中动画样式应用的不是很多，只有进度条组件里会使用。  例如：active |

# JavaScript插件架构

HTML布局规则：基于元素自定义属性的布局规则，比如类似于data-target的自定义属性等。

JavaScript实现步骤：所有的插件都遵循jQuery插件开发的标准规范，所有时间保持统一标准。

插件调用方法：声明式，JavaScript调用式。