1. 说明
   1. 改写策略

**1.策略说明**

（1）pc端只参考bootstrap，移动端参考ydui，不参考其他。

（2）去掉一切多余、复杂样式，只保留最基本的样式和组件。

（3）分为基础组件样式和扩展组件样式。基础组件样式尽可能的简洁。扩展组件一般不存在相互依赖，只依赖基础组件样式。

（4）图标使用font-awesome，原因是实际工作中并不会用这个图标库，而是会打包用到的少量图标。

**2.如何适应PC端和移动端？**

（1）PC端和移动端使用不同的样式。

媒体查询响应式只能适应简单的样式。PC端更多考虑兼容性，尽可能地兼容ie6。移动端尽可能使用CSS3语法，力求简洁高效。

（2）框架仍然考虑简单的响应式。

通过考虑响应式，满足简单的移动端需求。结构为：共用样式、PC端、移动端。

（3）PC端的样式也分2部分。

基本样式部分只兼容现代浏览器（还分出来响应式样式部分），另外1部分专门兼容ie低版本。正如同bootstrap一样，但基本部分如果可能仍要做些一些必要的兼容。

**3.响应式**

（1）只保留最简单的响应式，如果需要其他复杂的响应式，单独添加使用。

（2）保留的响应式有：1）container，2）

**4.和bootstrap的不同**

（1）去掉input-group。

* 1. 兼容性

（1）使用HTML5标签，引入html5shi.js兼容低版本ie。

（2）如果需要兼容ie8以下使用媒体查询，需要引入response.min.js。

（3）ie8中，当box-sizing:border-box;与min-width、max-width、min-height或max-height一同使用时，IE8不能完全支持box-sizing属性（没有解决办法）。

（4）插件selectivizr可以让ie8以下浏览器支持css3选择器。需要引入jQuery。

http://www.zhangxinxu.com/study/201009/selectivizr-test-demo.html

（5）引入DD\_belatedPNG\_0.0.8a-min.js，兼容ie6png图片透明问题。

* 1. 框架整体风格
     1. 尺寸

**1.bootstrap尺寸特点**

（1）button和表单元素的高度都是34px，即20px的行高+6×2px的padding+1×2px的border。table单元格的高度也差不多，也是20px的行高+8px×2的padding+共用的2px的border。

（2）段落和它们不一样，就只有行高。

**2.尺寸基本考虑**

（1）不同部分之间显然应该用margin分开。一篇文章内部，各段落、内容成员之间margin-bottom为1/2个line-height。组件之间、组件与文章之间，以及文章中非内容成员如落款address与内容部分之间，为1个line-height。

（2）垂直方向上，元素之间由margin-bottom分开，而不用margin-top（因为折叠的原因，要确保后面有空间）。

（3）margin-top用到的地方：（1）h\*。（2）垂直方向btn/btn-group/input/p之间，这样的好处是最后没有margin-bottom，bottom由外面的包裹元素决定。

（4）涉及到1个section内部的各部分之间的，才会设置margin-bottom为line-height，如navbar/breadcrumbs/list-group/wells/tables（未直接指明为line-height，但为20，就是1行）。部分中的段只有1/2个line-height，即10px，如p/h\*。段内单元之间为5px。其他通常不设置margin。

（5）inline元素padding通常为5px，更小的为3px。通常为6/12px，大的为10/15px。

（6）line-height都统一，除了label等为1外。

* + 1. 颜色

（1）一律不管是否是安全色。

（2）border颜色比bg颜色深10%。button、a等标签，深色系时，:hover颜色浅10%，:active颜色不用设置；浅色系时，:hover颜色深10%。current颜色貌似应该使用相反色系，hover也对应变化。

1. 底层
   1. normalize

使用https://github.com/necolas/normalize.css，将所有浏览器显示差异规范到一致，没有对样式风格的调整（包括也没有设置a标签的颜色等）。

* 1. reset

在bootstrap4中，在reboot文件中，既包含normalize，还包含bootstrap一些样式重置、调整。但reboot文件中还包括了一些样式设置，这些样式又在type等文件中重置，因此去掉了这些（这些都将会在type中设置），只保留了除normalize外，在显示上的最基础的设置。

* 1. scaffolding

（1）设置html字体为10px（通常默认为16px）。

（2）body的字体、行高、颜色。

如果html/body设置为100%，那么就是屏幕的尺寸，如果内部内容大于它们，它们仍然只是这个尺寸不变，只是里面的内容会显示出来。因此，设置html/body为100%的情况，只有1种，那就是需要设置内部某个元素为屏幕100%。鉴于此，要不要设置为100%，需要自己在使用中另行设置。

另外，去掉了原来的sr-only和sr-only-focusable样式。

1. 基础组件
   1. 布局
      1. container

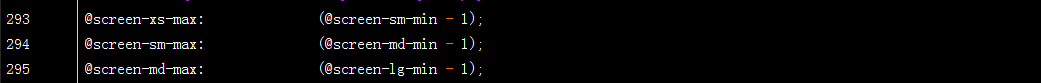
**1.bootstrap中的屏幕宽度**

（1）定义了4种屏幕宽度：480px、768px、992px、1200px。与此相对应，也定义了4种宽度的container，分别100%（小于768）、750px（小于992）、970px（小于1200）、1170px（大于等于1200）。

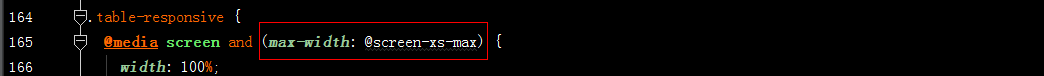
（2）以上屏幕宽度只是1种假设，并没有特别含义。只要小于480px就认为是手机（6plus也就412），不一定真是手机，只是按手机样式显示。同样，481-767就当作平板或手机横屏。大于等于768就当作pc屏。

（3）实际处理中，小于768px都是100%显示。即，手机屏和平板通常是一样的布局。

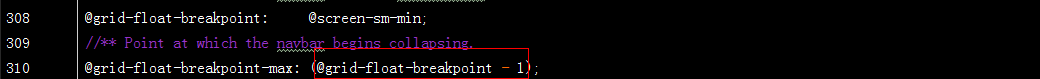
（4）min-width和max-width都包含临界值。因此，平板屏最大为768-1，以此类推。在原bootstrap中有：







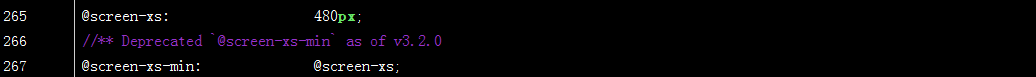
即，只要是屏幕就可以使用响应式表格。又如：

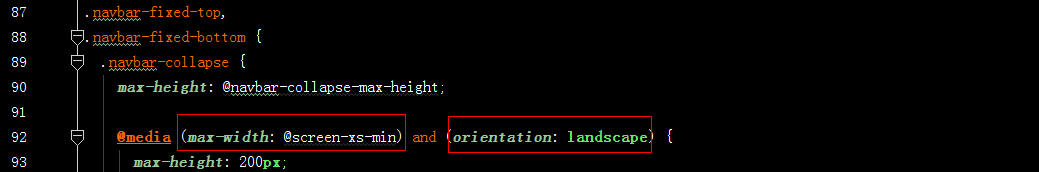




用到-max的都需要屏幕宽度-1（改写的时候去掉grid-float-breakpoint）。

（5）手机屏宽度跟其他有区别。整个bootstrap中直接用到@screen-xs只有1个地方：





1）只有这1个地方定义的原因是，bootstrap移动优先，首先定义的样式就是移动样式，后面再定义的其他样式（min）。

2）由这里可见移动端没有-1，说明480也按手机屏显示，除非后面定义了@screen-sm。

**2.改写说明**

鉴于480px屏幕宽度没有太大意义，因此去掉。新定义的屏幕尺寸，分别是：0-767px，768-991px，992-1199px，1200px以上，定义了4种对应宽度的container，分别100%（小于768）、750px（小于992）、970px（小于1200）、1170px（大于等于1200）。



**3.container的作用**

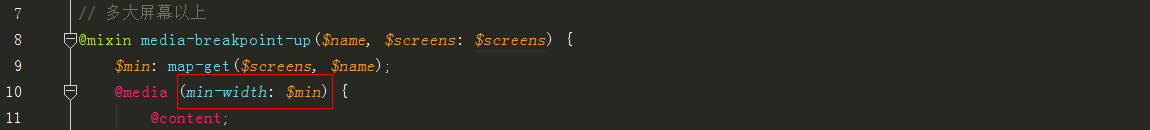
（1）container用于被嵌套在页面各个部分中，如header、section等。目的是这些页面部分仍然占满屏幕宽度，可以看到各个部分一致的满屏背景色、背景图像，而其中的内容部分仍然居中显示。

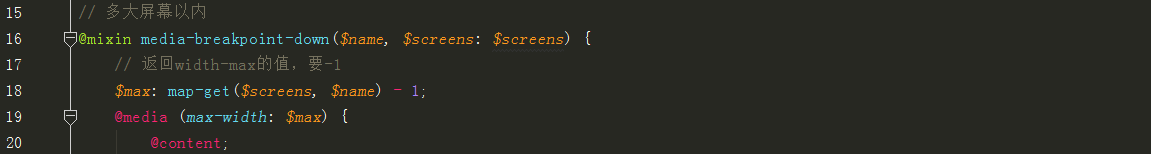
（2）pc端没有必要设置padding，没有必要，因为如果需要和屏幕进一步区分开，应该将上面的width减小即可，没必要再设padding。但移动端因width为100%宽度，所以应设左右15px的padding。

（3）手机屏container有15px的padding。通常情况没有任何问题。但在下拉菜单等情况下，因为弹出的部分也应占满屏幕宽度，因此，需要设置container的position为static，所在页面部分如section为relative。这样弹出部分width=100%就可以相对组成部分。不需要使用row，语义不对。

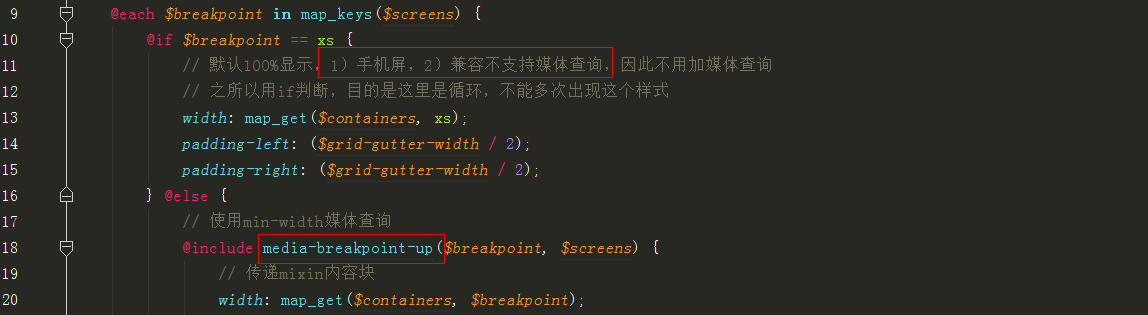
**4.mixin**

（1）获取媒体查询max-width和min-width。





（2）将媒体查询写入container中，使用了mixin的内容块语法。



**5.样式**

（1）container-fluid

没有设定宽度；通过margin设定水平居中；清理浮动。

（2）container

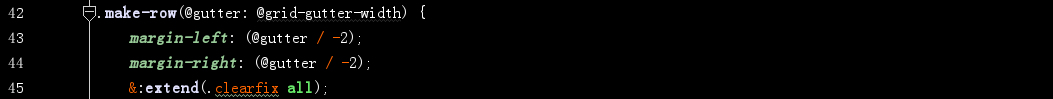
1）因混合container-fluid，所以能够水平居中。

2）默认响应式，定义了width。

* + 1. 格栅

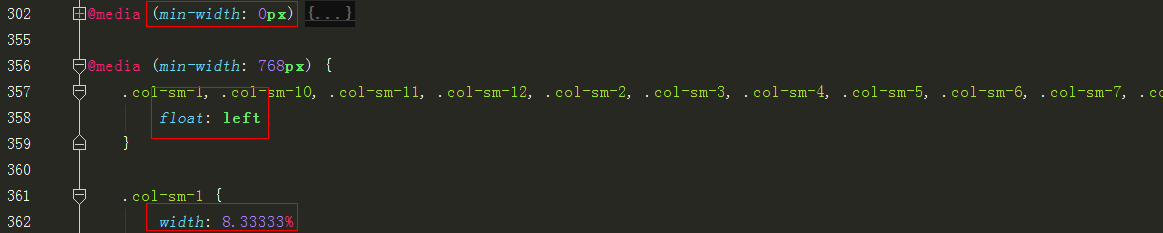
**1.row**

row应该和grid结合使用。唯一的作用是去掉首尾grid的padding（和container的没有关系）。



**3.grid**

（1）使用min-width进行媒体查询。想一想：目的是大于多少屏幕才如何布局不是？



（2）xs也使用了min-width：0%媒体查询。原因是即使移动优先，移动端显然都支持媒体查询。而桌面端不支持的，那么就是自身属性决定，如果是div，就是100%宽度。

* + 1. 其他响应式样式

layout-responsive-utilities文件中提供针对元素在不同屏幕下显示、隐藏的样式。需要时引入。

* 1. 排版样式——基础
     1. 基础标签

**1.标题h\***

（1）重新定义h1～h6的标准样式，新增.h1～.h6样式类（与标签完全一样），以使在所有浏览器呈现一致效果。

（2）字体大小：h6-h4按4px递增，h4-h1按6px递增。

（3）外边距：h6-h4上下均为普通行距1/2，h3-h1上边距为1个行距，下边距为1/2行距（说明上下紧密度不同）。

（4）设置line-height为1.1（1个单独定义的行距变量），不是base的1.4285（既然使用外边距间隔，就没有必要使用过高行距）。

（5）标题中可以使用small标签。使用百分比定义字体大小，还定义了颜色。





（6）并没有设置对齐，对齐的话使用text-left\*实现。

**2.段落p**

（1）去掉浏览器默认的上边距，只保留下外边距，为行高1/2。由此可见，段内行之间有line-height，段之间多1/2的line-height。

（2）定义强调段落.lead样式类，有更大字体和下外边距。





**3.a标签样式**

a标签保留hover时下划线。

* + 1. 列表样式

**1.重新定义ul/ol标签样式**

（1）去掉默认的margin-top，定义ul/ol的margin-bottom为1/2行高。

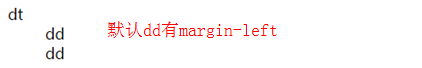
（2）如果有ul/ol内部有嵌套，去掉内部的ul/ol的margin-bottom。

（3）其他默认样式保留，如ul/ol的数字或标点，显示他们的padding-left。

**2.重新定义dl标签样式**

（1）去掉默认的margin-top，重新定义dl的margin-bottom为1/2行高（原来为1倍行高）。

（2）默认dd有margin-left，去掉。定义dt为粗体来区分。



**3.去点列表样式.list-unstyled**

* 去掉padding-left和list-style。

**4.内联列表**

* 引入去点列表样式去掉默认点样式。
* 将内部li设为inline-block，并设置左右padding为5px（一共就是10px，小于字体14px）。
  + 1. 语义标签

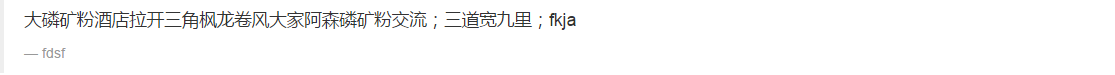
（1）重新定义small（font-size=85%）、cite（去掉斜体，仅作为语义标签）、mark（将默认黄色背景色改为红色）3个强调标签样式（去掉默认样式）。

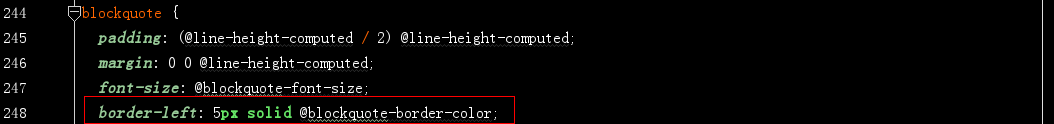
（2）重新定义address、abbr。



（3）引用

去掉反向引用，只保留footer作为引用脚注。





* + 1. 基础文本样式

（1）定义4种对齐样式：text-left/text-right/text-center/text-justify。

（2）定义6种文本前景色和背景色。使用text-emphasis-variant和bg-variant的mixin，目的是设置a标签的hover颜色。

* 1. 排版样式——图片

（1）在reset的时候，img标签已经去掉了border。这里设置了图片垂直居中对齐。



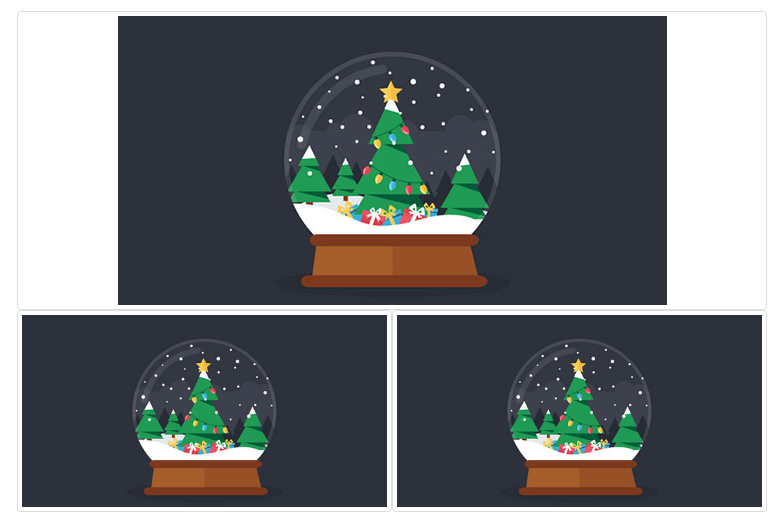
（2）定义图片img-responsive样式，主要用于移动端，宽度根据自身决定，最大100%，height为auto。

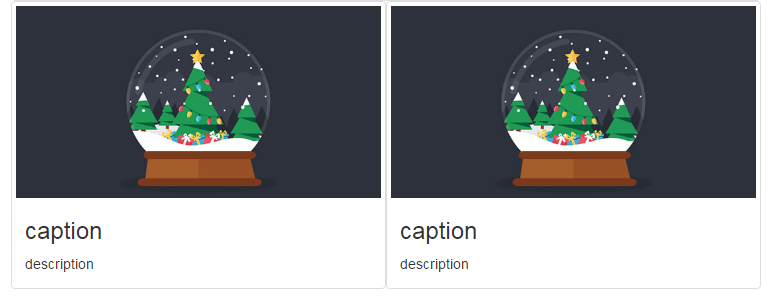
（3）定义img-rounded、img-circle、img-thumbnail三类样式。





（4）微缩图thumbnail样式





主要处理：

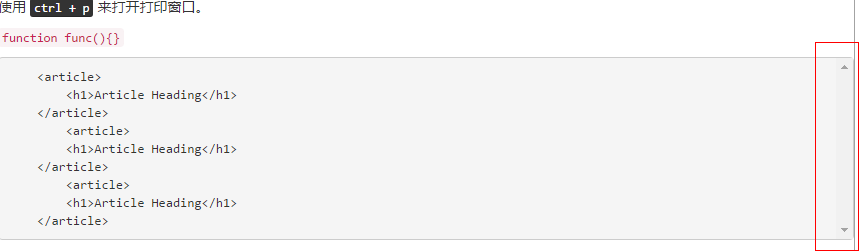
1）用1个边框包裹1个图片，图片要自适应。

2）说明文字的左右padding应略大于图片。

* 1. 排版样式——代码

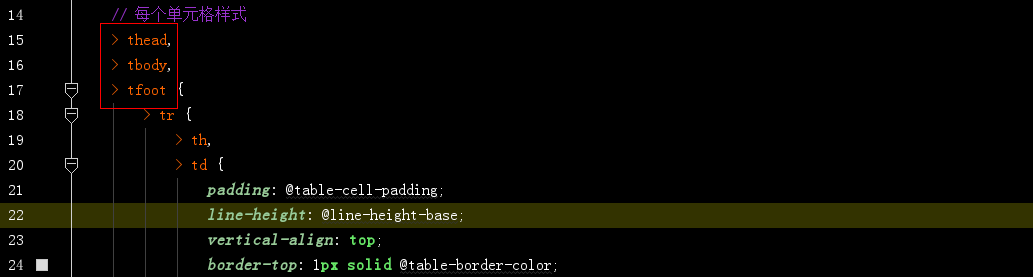
重新定义code、kbd（行内）、pre（块）样式：

* code和kbd仍然是inline，虽然没有border，但设置了背景色，仍然可以使用border-radius，显示出圆角背景。
* pre可以使用pre-scrollable样式添加滚动条。



* 1. tables.less
     1. table样式设置原则

（1）只对table和th/td设置样式。



（2）行的边框、单元格的边框都是通过设置th/td边框实现。只要设置border-collapse为collapse，即使四边都有边框也会重合不会多显示。

* + 1. 基础样式

**1.table**

（1）使用上，对table运用.table样式即可。

（2）单元格内容有基础的line-height，另外还设置了padding。

**2.table-condensed**

仅减小padding大小。

**3.table-borderd**

（1）设置所有th/td有1px的border。

（2）设置thead中的th/td为2px的border以覆盖上面的设置。

**4.border-striped**

使用属性选择器odd。

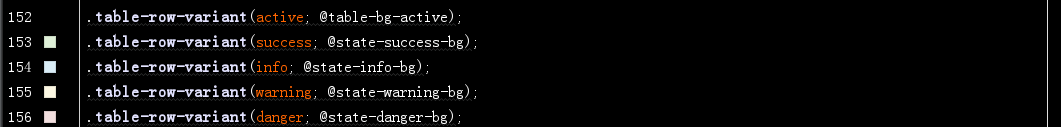


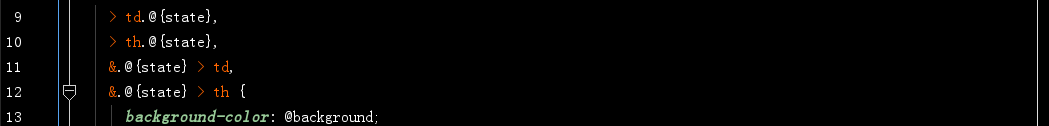
**5.table-hover**

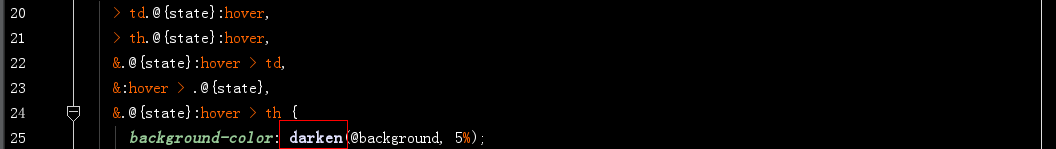
使用伪类选择器tr:hover>td/th。



* + 1. 响应式table-responsive
* 响应式就是在table外面嵌套1个div.table-response，其width为100%，并设置overflow-x。
* 设置表格td/th内容white-stage:nowrap。
  + 1. table-row-variant/mixin
* 定义active等5种单元格背景样式，可用于行/单元格（还是通过对每个th/td设置实现）。





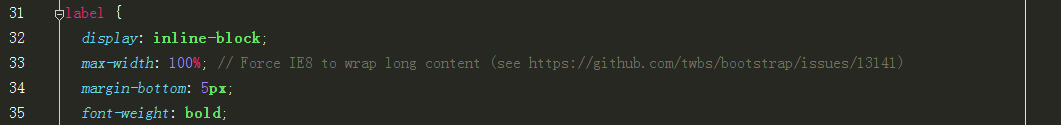


* 1. form
     1. 重置样式

所有的表单元素默认均为inline。

**1.label**

重置为inline-block。对于radio/checkbox，可以使用<label><input/>说明文字</label>方式。对于其他input，还是使用label+input方式。只所以设置margin-bottom有5px，目的是在和form-control结合使用时需要。在和其他样式结合使用时都去掉了。



**2.input[type="file"]**

重置为block，独占1行。



**3.radio和checkbox**

radio和checkbox存在与旁边文字对不起情况。bootstrap的样式重置不起作用，使用如下样式替换。



* + 1. 基础样式

基础样式不对form添加样式，只对表单控件设置。都是独占1行（此时匹配的label也是独占1行），在输入内容较多和手机屏时比较有用。

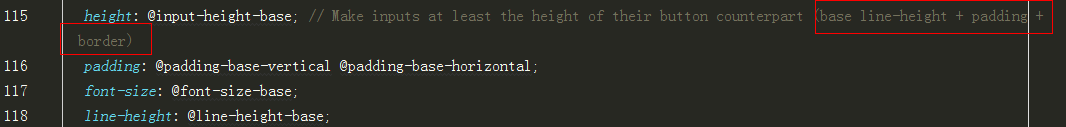
**1.form-control**

用于除file/hidden/radio/checkbox/submit/button/reset外的input，以及select、textarea。使用时因独占1行，不能被label嵌套，和lable平行（label有margin-bottom=5px）。有如下样式：

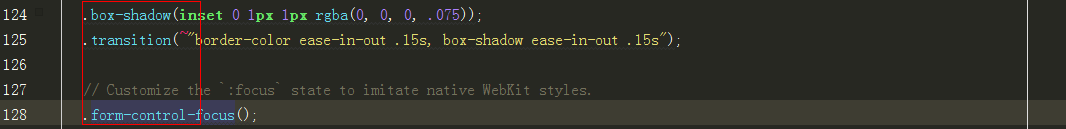
* 独占1行显示。

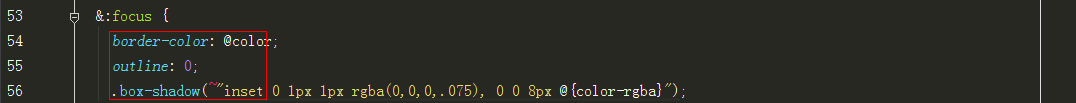


* 内容部分普通行高，外加普通padding。

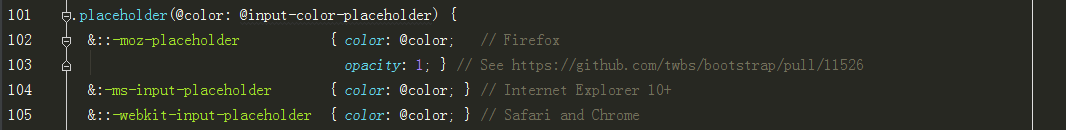


* 没有设outline，而是设置的内阴影。普通情况下就有内阴影，focus下加大并变色。



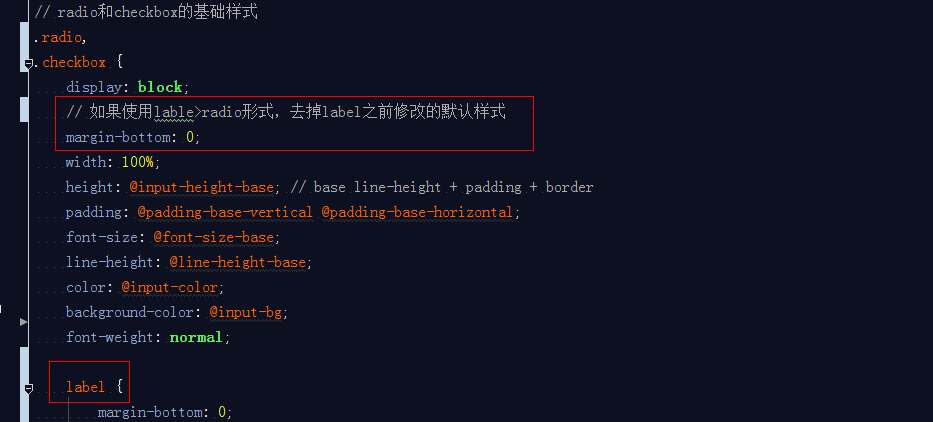


* placeholder颜色较浅。



**2.checkbox和radio特殊设置**

既然.form-control为block，那么包裹checkbox和radio的label或div也应该为block。高度（line-height、padding等）应该一样，区别在于不能有边框。





**3.分组**

不是用来分组多个input控件，而是分组1对input+label。只设置了margin-bottom，form-control没有设置控件之间的间距。



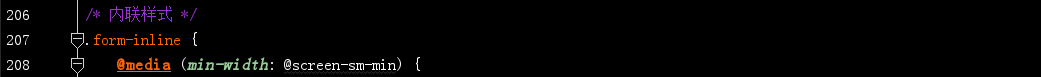


* + 1. 内联表单

所有控件1行显示。和基础样式一样，input和label对应该放在form-group内。



（1）对form加.form-inline，只在非手机屏幕中可使用。



（2）内部并没有特别设置，主要是将form-control、radio、checkbox样式中的block改为inline-block，width改为auto（浏览器默认宽度），label的vertica-align设置为middle即可。

（3）由于radio和checkbox毕竟内容较少，不用独占1行。因此单独有radio-inline和checkbox-inline两个样式，可以不在form-inline下使用，即在其他都是form-control样式下，它们单独为内联样式。

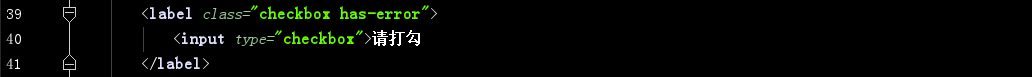
* + 1. 其他样式

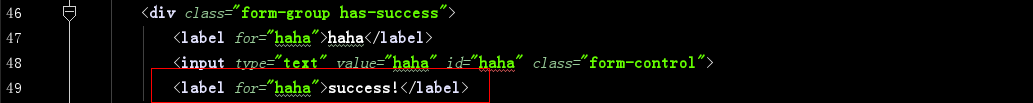
**1.控件尺寸**

提供input-sm和input-lg两类样式，使用input-size这个mixin，包括设置一些尺寸和字体大小。

**2.验证提示状态**





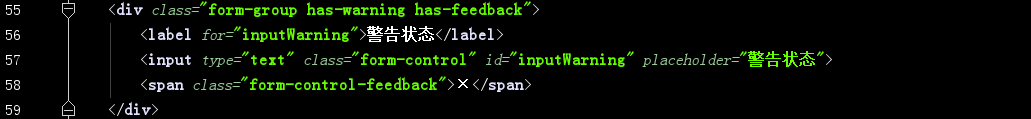


（1）使用form-control-validation这个mixin定义，只涉及颜色。

（2）提示语用lable。

**3.图标提示**





（1）使用relative+absolute结构，需要计算出absolute的位置：

1）设置absolute的高度和input高度一样。

2）都是独占1行情况下，absolute的top位置=label的高度+label的margin-bottom。

2）内联状态下，absolute的top位置应该为0。



（2）使用label作为提示。

* 1. button

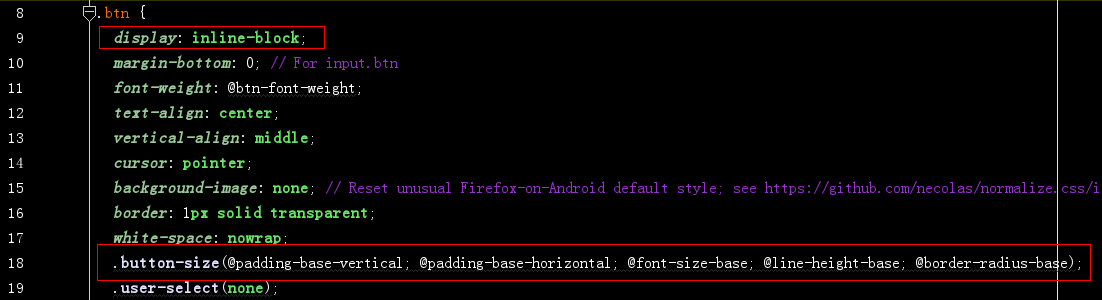


**1.默认设置**

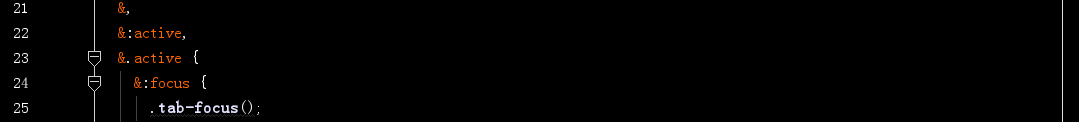
（1）默认为display=inline-block。

（2）尺寸同table单元格，都是正常line-height+padding+border。

（3）组合样式，没有设置颜色。



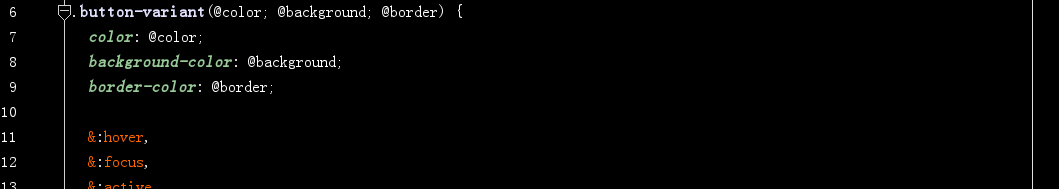
（4）设置focus/active/disabled等情况的outline、decoration、opacity等。



**2.颜色样式**

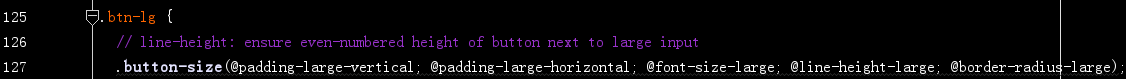
默认样式.btn没法单独使用，必须和颜色样式一块使用。有6组btn颜色变量，通过传入button-variant这个mixin，设置自身及各种伪类颜色（只有颜色）。





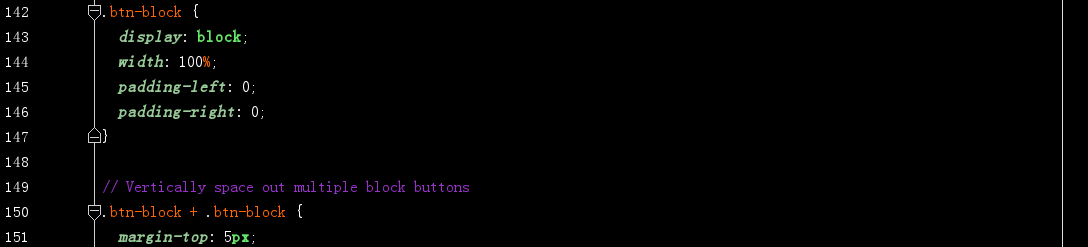
**3.大小样式**

默认样式中已有尺寸，还可附加尺寸改变。



**4.块样式**

默认样式为inline-block，可以设为块样式独占1行。



**5.链接样式**

在应用.btn基础上，再应用.btn-link。设置样式为透明，前景色和普通a一样。

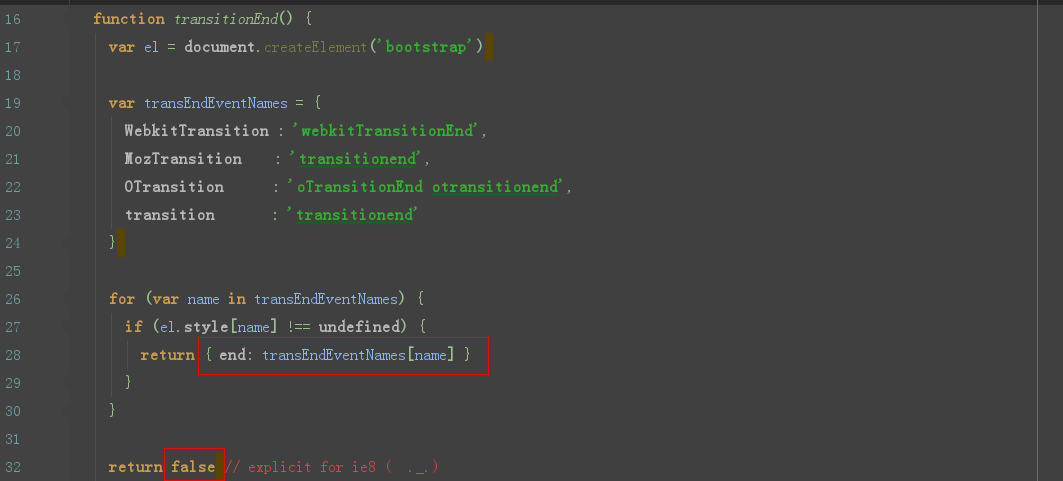
* 1. transition.js
     1. transitionend事件兼容问题

（1）当css动画结束会触发transitionend事件。不同浏览器transitionend事件有不同前缀。另外，当有多个属性触发css动画，将多次触发transitionend事件，而实际只需要所有属性动画结束后的一次触发，因此，jQuery中使用one绑定事件。

（2）当元素display为none，动画时间为0等情况不会触发transitionend事件，为此需要手动触发。

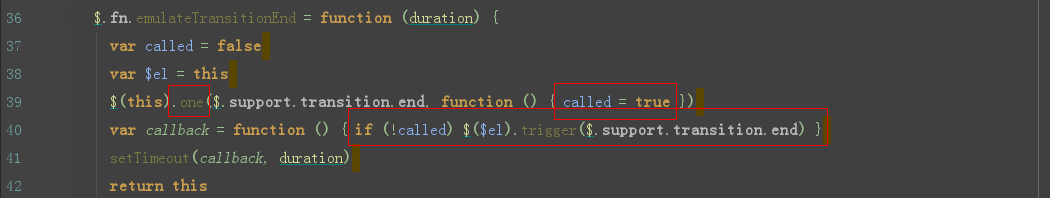
* + 1. 源码
* 使用$.support.transitiond形式添加transitionend事件兼容性测试。





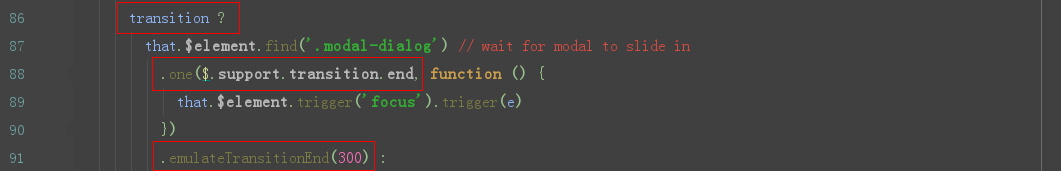
如果支持，返回的是{end:\*}形式，否则返回false。

* 添加$.fn.emulateTransitionEnd原型方法，用于手动触发transitionen事件。



called的作用就是只执行trigger一次。虽然one也行，但这样可能效率更高。

* + 1. 使用



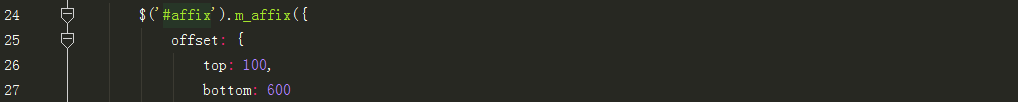
当不支持时，$.support.transition.end为undefined，并不会绑定事件。$.fn.emulateTransitionEnd会手动触发1次事件。

1. 扩展组件
   1. affix

粘条，跟随滚动条滚动或在屏幕上静止不动的一种效果。应用如购物车等。

从上向下滚动时，滚动条距离window距离达到top值以后，affix元素就固定不动了；继续滚动，当到滚动条底部距离window底部达bottom值时，浮标与window底部距离保持不变，于是又开始继续滚动，以免和底部重叠。因此，浮标的样式在3个阶段分别为：（1）不添加浮标样式；（2）fixed；（3）通过jquery的offset方法，设置top值，自动换算为浮标的相对定位。





**1.算法逻辑**

（1）参数含义

top：浮标顶部距离window（非document）顶部的距离。到达这个距离时，浮标固定不动。

bottom：浮标底部距离window（非document）底部的距离。到达这个距离时，浮标保持与底部的距离不变，从而当滚动条继续往下运动时，跟随着滚动。

（2）满足top条件的情形

当浮标是正常文档流元素时，为浮标顶部文档坐标-文档滚动条距离<=top时：

$element.offset().top-$(document).scrollTop()<=top

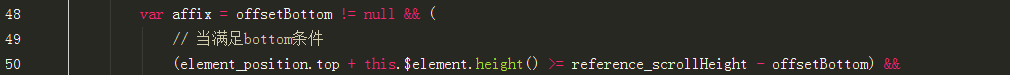
注意这里有=，因为一旦满足条件，浮标就转为fixed定位，上面的等式两边相等。



（3）满足bottom条件的情形，为浮标底部距文档顶部距离>=文档高度-bottom值：

$element.offset().top+$element.height()>=$(document).scrollHeight-bottom

注意这里也有=，也是因为满足条件后，会设定浮标的坐标，从而使它们始终相等。



（4）满足bottom后，向上滚动，何时复位？

需要考虑一个非常重要的事情，那就是什么时候退出bottom：即，一旦满足bottom，设置为relative，当向上滚动时，如果不调整回去，永远都固定到bottom的那个地方了（上面的=判断）。判断方法为：

$(document).scrollTop()+top+$element.height()<=$(document).scrollHeight-bottom

即，滚动条位置+设定的top+浮标高度，不再满足bottom条件，应该回位。考虑以下几种情况：

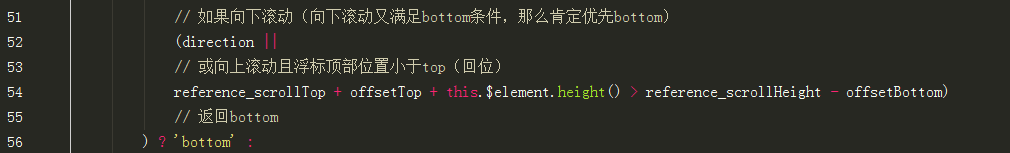
* 如果window高度特别小，bottom值特别大，导致此时浮标上面部分还隐藏时，说明bottom处的位置top值+浮标高度值>$(document).scrollHeight，要不就不会隐藏，因此有：

$(document).scrollHeight-bottom-$element.height()<$(document).scrollTop()

即：

$(document).scrollHeight-bottom<$(document).scrollTop()+$element.height()

不满足条件，只有当全部显示出来且有了top视觉高度才会回位。

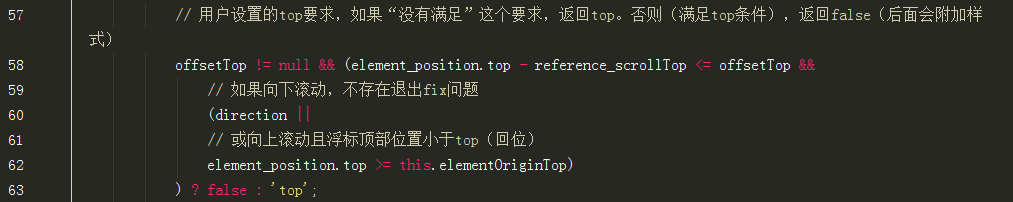


* 其他情况

应该和上面类似。

（5）向上滚动，何时去掉fixed样式，复位到最初的样式？

在初始化插件的时候，就获取浮标最初的top值，当继续向上滚动，浮标坐标top小于最初的坐标top时，复位。



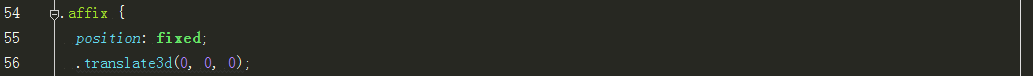
【说明】

参照物之所以只为document。因为其他参照物都包含在document内，top/bottom值可以为function，函数内可以获取这些参照物的位置。此时，top/bottom值必须为1个function，如果直接用jquery获取参照物位置，则只是1个固定的值，如果是function，则可以每次检查时，通过执行function从而获取到1个实时的值。

不考虑浮标包含在1个滚动容器内情形。

**2.样式**

Affix浮标元素只有1个样式affix，设置其为fixed。



但在js中还有2个CSS样式类名affix-top和affix-bottom。affix-top是在正常文档流中的状态定义时的状态。affix-bottom是与文档底部固定距离的状态。这两个样式应该是用来提供给用户自定义。

**3.js**

原来的js代码有些不合逻辑：如果设置了bottom，当滚动到bottom之后，浮标元素将回归最初的位置而不是设置为与文档底部距离固定状态。因此进行了改写。

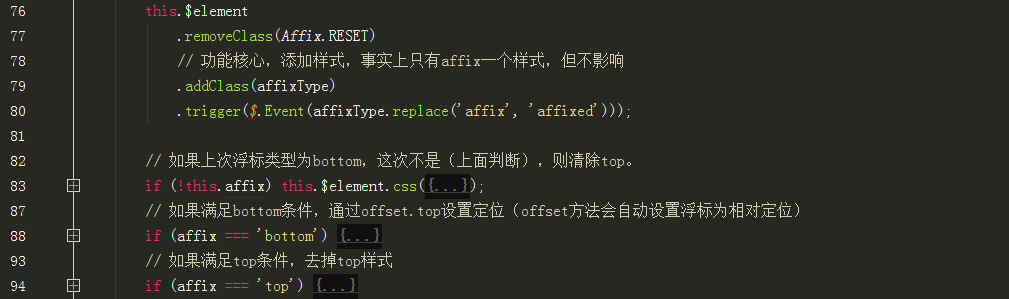
（1）位置判断

需要同时考虑浮标top和bottom（当然设置只能设置top）。bottom判断优于top判断，因为满足bottom都满足top条件。由于bottom条件设置的relative为内联样式，因此在top条件成立时应该将该内联样式去掉。

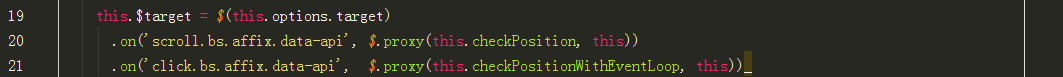
（2）如果和上次是同一位置，不做处理。



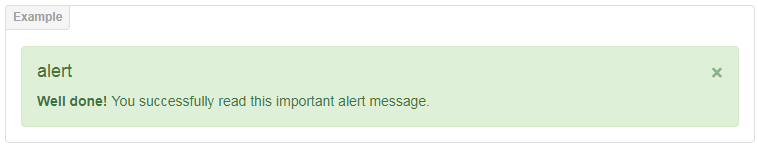
（3）核心还是设置样式。



（4）最终是有target的滚动事件调用。



* 1. alert



1.需要解决的问题

（1）close定位。

使用close-dismissable的padding-right和absolute实现close。

（2）内部h4/p/ul之间的边距问题。

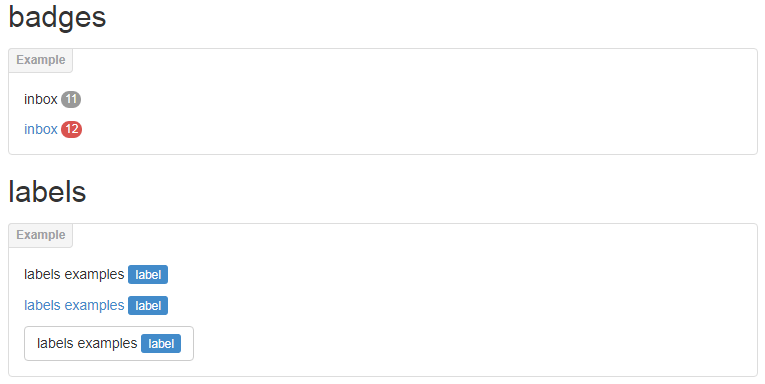
需要让里面的元素更紧凑，去掉默认的margin。

2.js

只有close这1个方法，没有其他方法。

* 1. 徽章和标签

对原bootstrap中的徽章和标签样式进行了统一。两者只在padding和圆角有所不同，其余均相同。



（1）要有那些样式？

字体稍小一些。和本行文字一样能垂直居中对齐。badges更圆滑一些，padding更小一些。

（2）如何实现

1）因为涉及到大小和间距，要设置为inline-block。

2）通常情况inline-block都是baseline对齐，要实现垂直居中，因为大小和字体不一样，且字体不能为0，需要用relative直接调整。

* 1. 面包屑



（1）使用ul>li>.divide+a结构，.divide为1个span，人为添加。

（2）需要前后保持相同的间距。

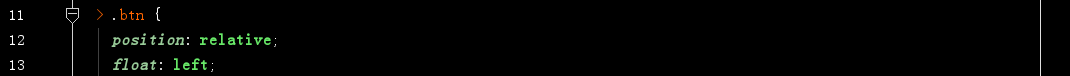
如果使用inline-block，会有1px的间距，干脆使用float，然后设置相等间距。

* 1. 按钮组
     1. 基础样式

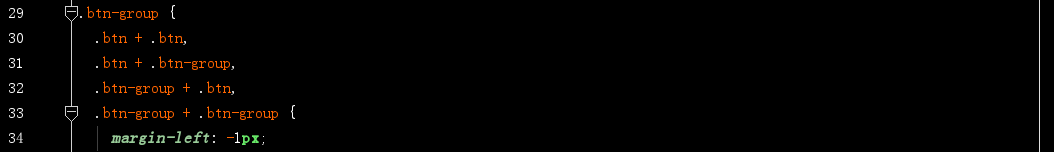


需要解决的问题：

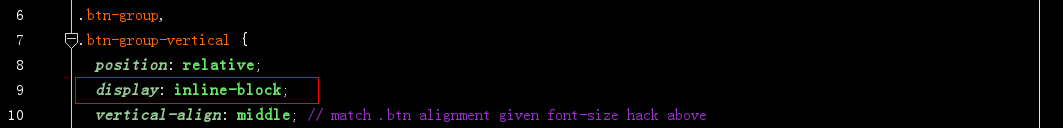
（1）基于移动优先的原则，如果btn-group内的btn较多，多于1行时情况，因此设置btn为左浮动。这里.btn没有使用inline-block，而是使用浮动，从而避免1px问题。



（2）2个btn之间的border有2px了，因此要去掉1px，这个时候就用到了-1px的margin。

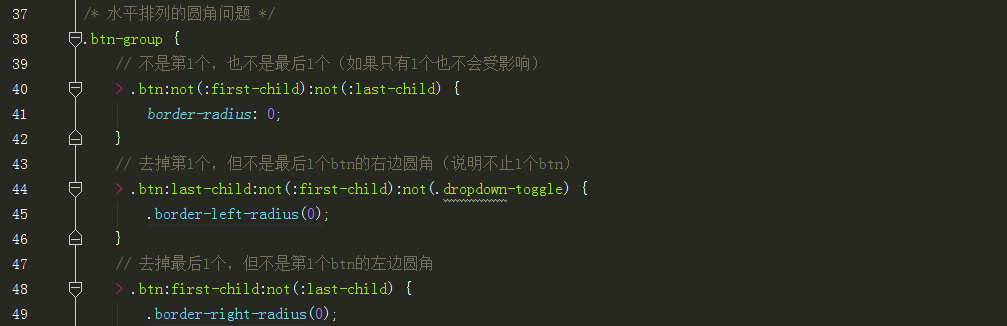


（3）btn-group也应该设置为inline-block，btn-group的嵌套也应该浮动，这样可以与其他btn、btn-group并排且会有1px的间隔。





（4）去掉非第1个btn左边的圆角，去掉非最后1个btn右边的圆角，且要考虑btn-group中只有1个btn的情况。分2步处理：1）先去掉非首尾btn的圆角，2）然后考虑第1个btn是否为最后1个，已经最后1个是否也是第1个。

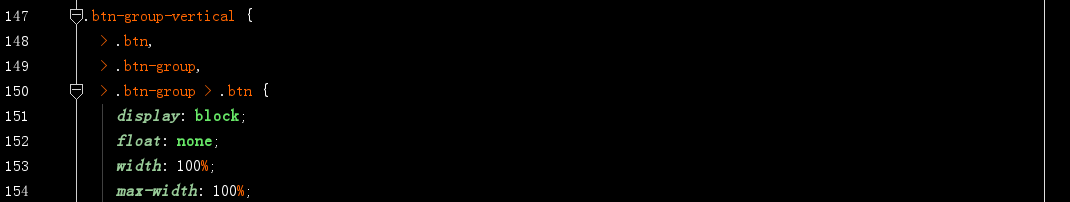


* + 1. 垂直分组

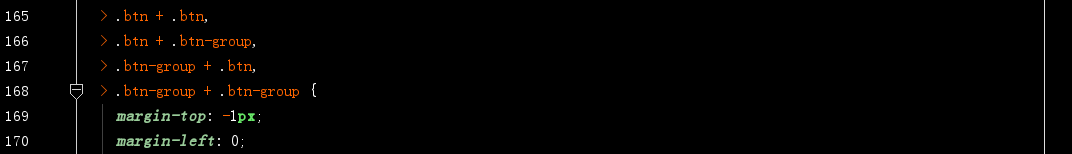


需要考虑的情况：

（1）内部的btn、btn-group都必须是block，因此还要去掉浮动。



（2）因为是上下排列，因此要去掉上下排列的边框重叠，设置margin-top=-1px，去掉之前设置的margin-left。



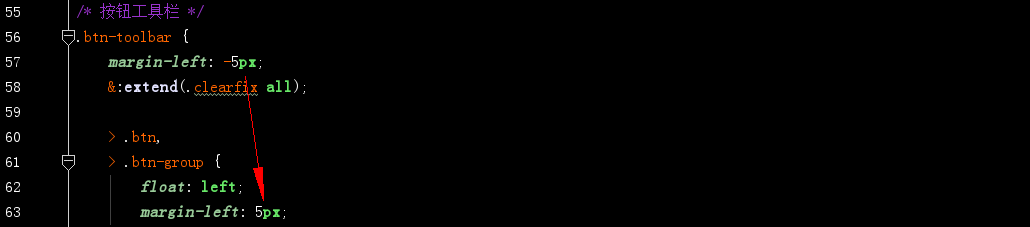
（3）圆角问题

因为btn-group-vertical和btn-group并列，进行样式组合，可以在修订样式中直接合并到基础样式圆角处理样式一起。

* + 1. 按钮工具栏



由.btn-toolbar包裹.btn-group或btn。没有其他需要考虑的，只需要设置margin-left。然后通过设置包裹元素的负margin抵消整体偏移。



* + 1. 嵌套分组

太复杂，只保留嵌套下拉列表。这里只留有嵌套分组的基本样式，需要在其他组件内设置。

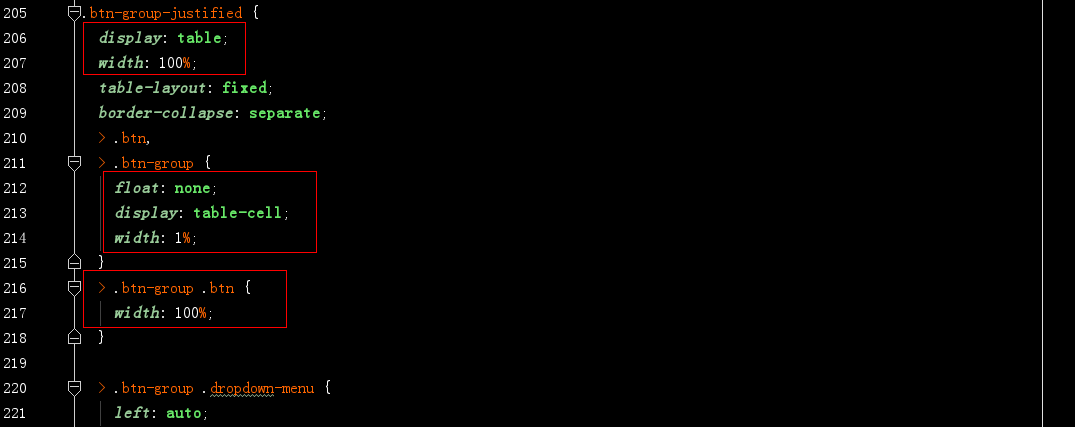
* + 1. 自适应分组

使用外层table样式和内层table-cell样式实现自适应。需要：

（1）最外层的btn-group-justified设置为table，并且width为100%。

（2）内层btn和btn-group为table-cell，并都要去掉浮动。

（3）.btn-group内的.btn（dropdown-toggle）要100%宽度（仍然是浮动，只是100%占据宽度，而之前则是有dropdown-toggle撑开父元素宽度）。

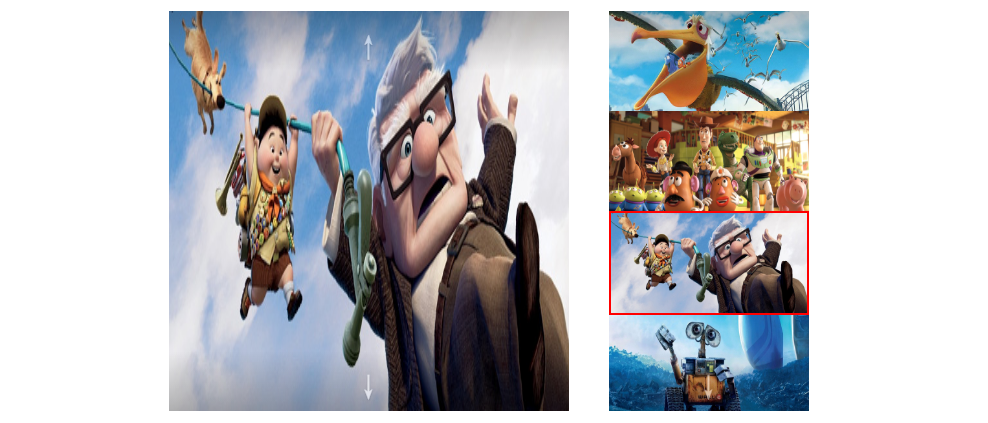


* 1. carousel

轮播。有如下几种形式：

（1）单纯轮播图。可以自动无限循环滚动，有指示栏，也可以手动左右滚动，可以水平也可以垂直滚动等。

（2）相册。本质是两个轮播图。需要考虑连个轮播图之间的同步。要求只能是相册可以自动滚动，滚动时对应的轮播图高亮，点击轮播图时相册移动到对应的位置等。



（3）能缩放的轮播图。



以上都是几乎相同的dom结构。

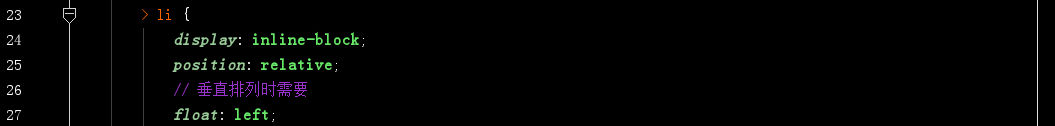


* + 1. 单纯轮播

**1.less**

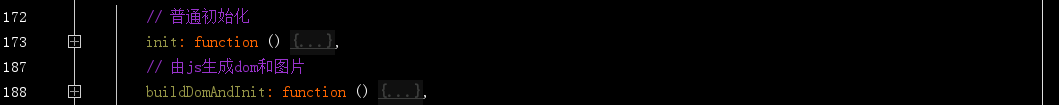
（1）ul为绝对定位或相对定位，通过移动ul来显示当前图片。

（2）ul中的li应该为float，目的是垂直移动时能够换行（inline-block也可以，设为float没有1px间距）。

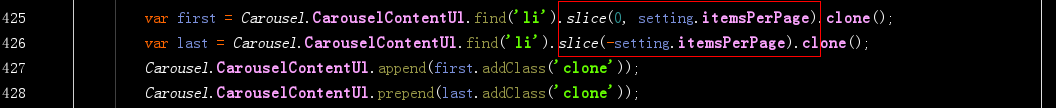


**2.js**

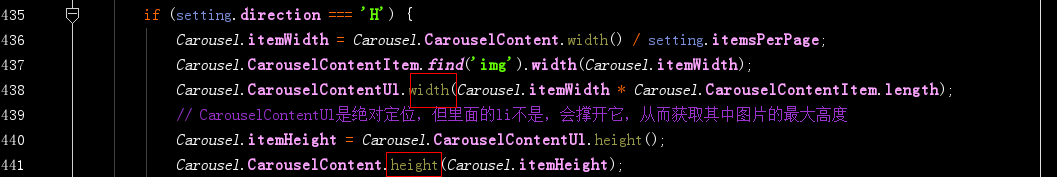
（1）提供两种初始化方法：从dom中获取元素和直接用js生成dom。



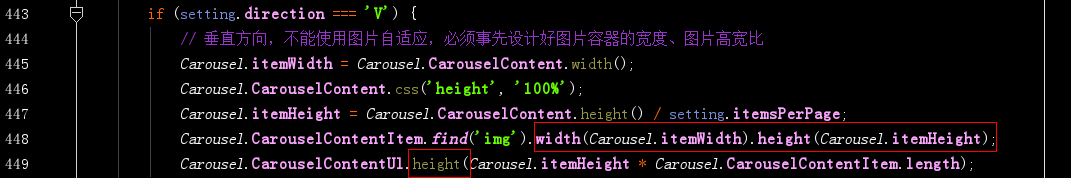
（2）根据1页显示的个数复制前后元素，以实现无限循环（必须clone，注意在clone后才能append）。



（3）水平滚动时，由于ul内的li是inline-block，必须设置ul的width，保证不换行。ul可能是绝对定位，因此要设置content的height。



（4）垂直滚动时，必须固定每张图片的height/width，因为height用来计算移动距离，不能自动。宽度用于定位，否则格式会乱。

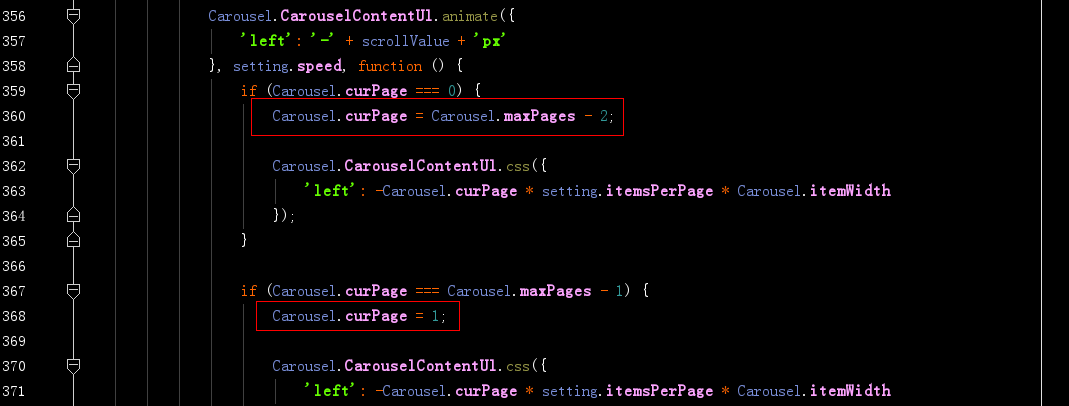


（5）使用moving属性禁止动画时添加新的滚动，且停止定时器，只能动画结束后才可以。这样，就不用考虑快速点击导致页码错误。





（6）动画结束后，判断页面以实现无线滚动。如果是第1页往前滚动到clone的最后1页，需要调整到倒数第2页（倒数第1页是clone的第1页）。如果是最后1页往后滚动到clone的第1页，需要调整到第2页（0才是第1页）。



由于设置了moving属性，所以不用在prev/next/to等方法中考虑页面调整。

* + 1. 相册

**1.less**

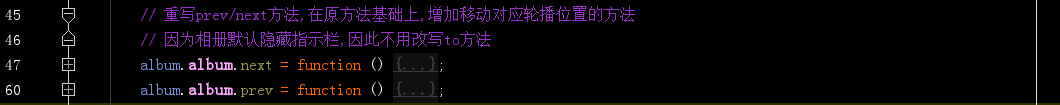
没有特别的设置，如果是相册，需要人为设置好相册和轮播图之间的位置和大小。另外，垂直滚动不需要显示控制栏。

**2.js**

（1）实例化2个轮播图。

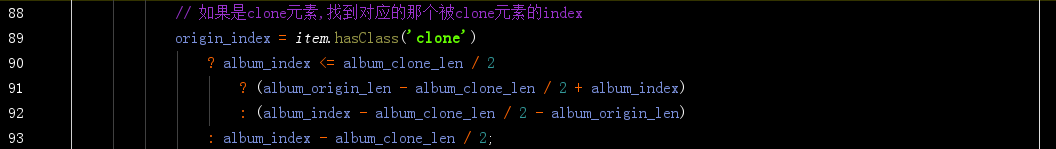


（2）重新相册这个轮播图的pre和next方法（没有to这个方法了，因为没有控制栏）。



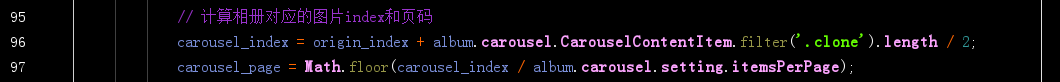
和原来的方法区别是在里面增加控制对应的轮播图高亮为位置。分为2步：

1）计算当前显示相册的第1个li在没有clone的列表中的序号。



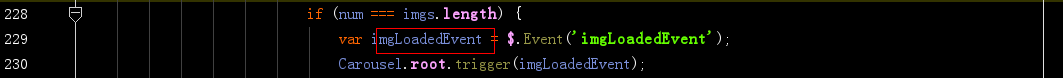
即，如果是clone，那么是前面的clone还是最后面的clone，由此计算出自己的origin位置。

2）计算轮播图的含有clone元素的列表的位置和应该显示的页面。

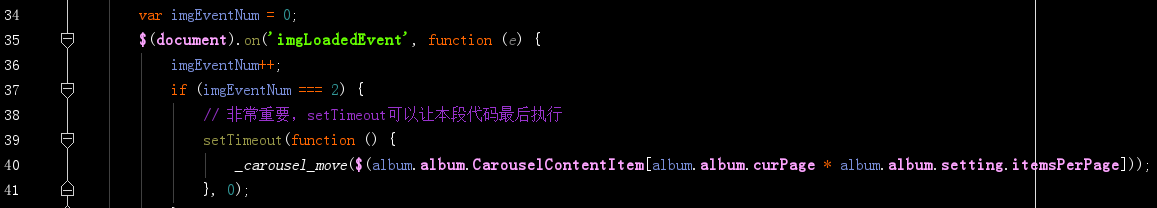


（3）给轮播图每个li添加1个click事件，显示相册对应的图片。和上面类似，也是先求出origin的位置，然后找到相册的页码（只要页面即可）。

（4）初次加载完成后，需要高亮显示轮播图对应的元素和页面，这个功能单个轮播是没有的，可以实现上面的\_carousel\_move方法。但需要等两个轮播img都加载完成之后，且\_init方法之后才能使用。因此给图片加载完成之后添加自定义事件。



当2个轮播加载完，执行。另外，通过setTimeout方法，实现在\_init之后执行该代码。

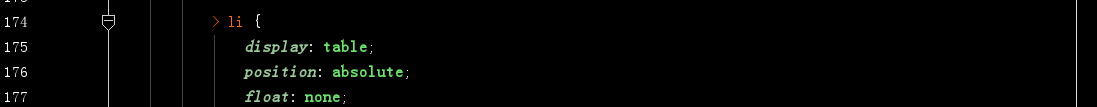


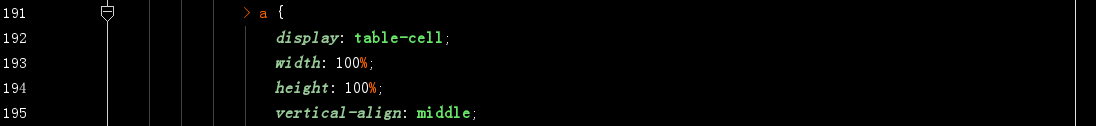
* + 1. 可缩放的轮播

**1.less**

（1）只能水平滚动。

（2）因为有缩放功能，图片没有占据整个li的高度，因此要垂直居中。这里还不能用js，因为缩放动画之前不知道图片高度。只能用table和table-cell实现。

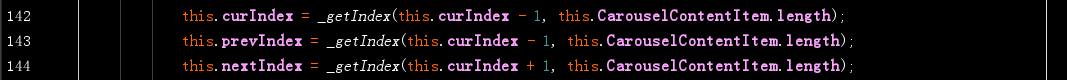




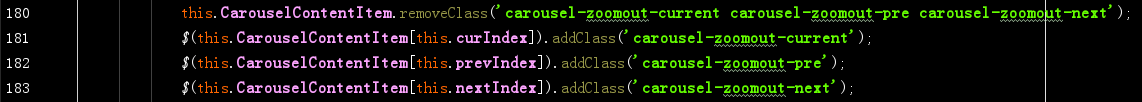
（3）实现原理是先求出当前中间显示的li的index，然后再计算它前后li的index，分别添加样式。样式带有动画。

**2.js**

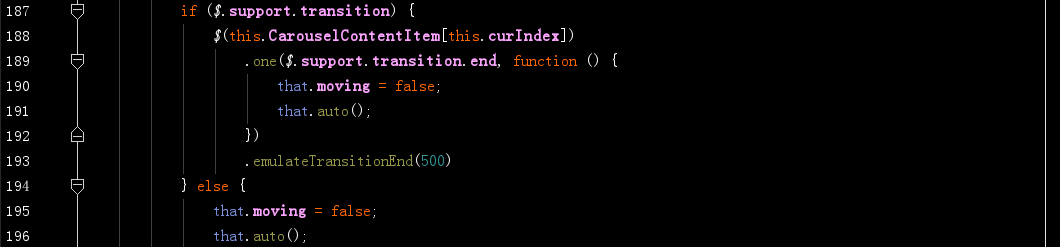
实现非常简单，就是先计算index。



添加样式。



动画结束后设置moving属性为false。



* 1. collapse

**1.展开**

（1）折叠时的样式类为collapse。



（2）因为display为none无法触发动画，需要首先去掉，然后添加动画样式collapsing（内有transition），并设置对应的折叠动画维度为0。





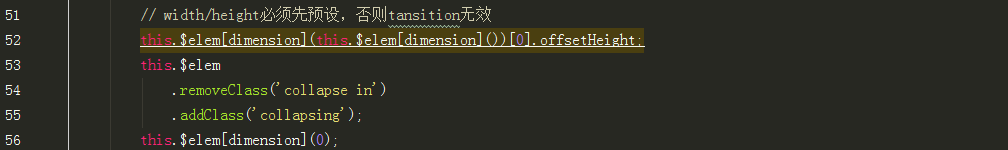
（3）此时，元素的width/height为0，但仍然有scrollWidth/Height（注意，如果里面没有内容，scrollWidth/Height仍然也是为0，除非有内容，把它撑开。对于没有内容直接设置width/height是无效的），获取，设置，从而产生动画。动画结束，添加in，修改元素display。

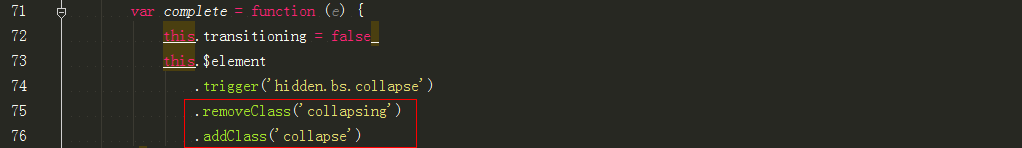
**2.折叠**

（1）特别注意，对于width/height，需要先预设当前值，否则动画无效

（2）添加动画样式类collapsing。

（3）设置动画终点值为0。动画结束，添加样式collapse，去掉样式collapsing。





* 1. dropdown
     1. 基础样式

dropdown不是1个具有定位功能的组件，不能单独使用，需要结合navbar使用。其结构为：



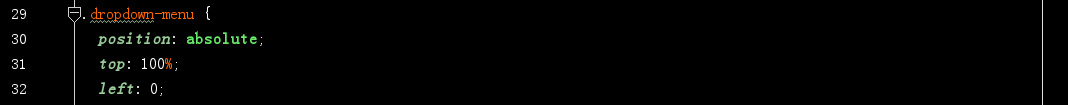
即：.dropdown中包裹>ul.dropdown-menu和button.dropdown-toggle。最外层的.dropdown只有1个样式position=relative，所以默认占1行。如下，如果不将.dropdown设置宽度，就会独占1行，从而使dropdown-menu跑到最右边去。



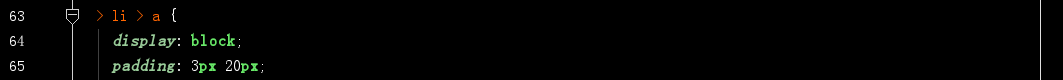
**1.less**

需要考虑的事项有：

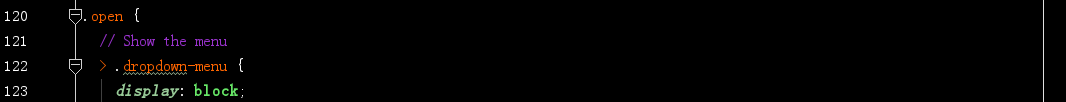
（1）外面.dropdown设为relative。.dropdown-menu设为absolute，且top为100%。默认左对齐，因此设置left为0。默认.dropdown-menu在下部，因此设置top=100%。



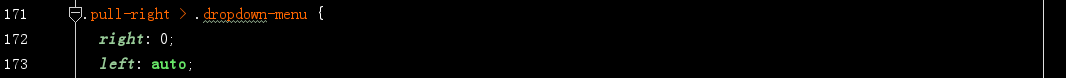
（2）所有列表中的li标签中的a都是block。



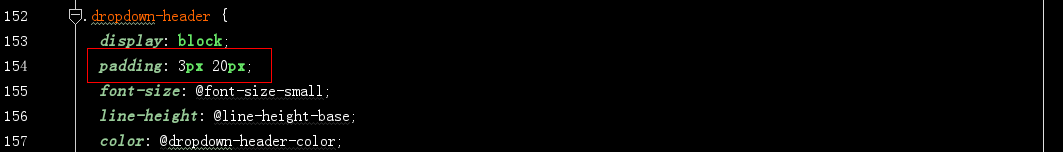
（3）与.open合用设其display:block。



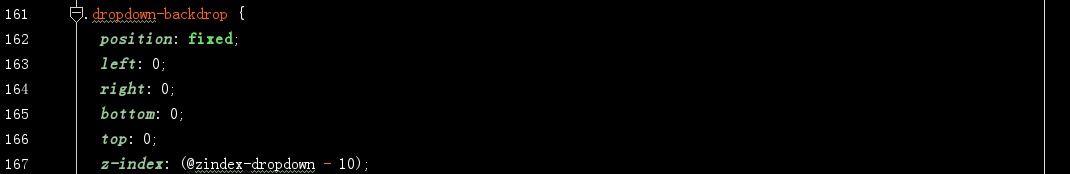
（4）默认为left:0，可以和.pull-right合用，其设right:0;left:auto。



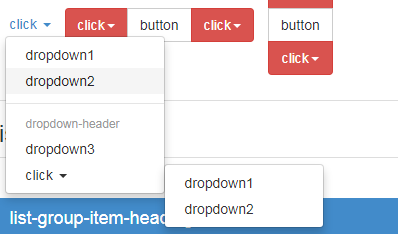
（5）分组标题.dropdown-header（直接添加到li上），li里面不再有a。



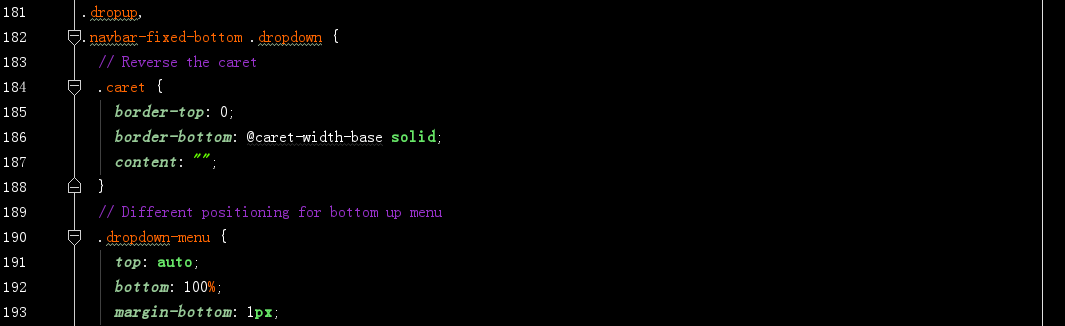
（6）背景，要通过js设置。



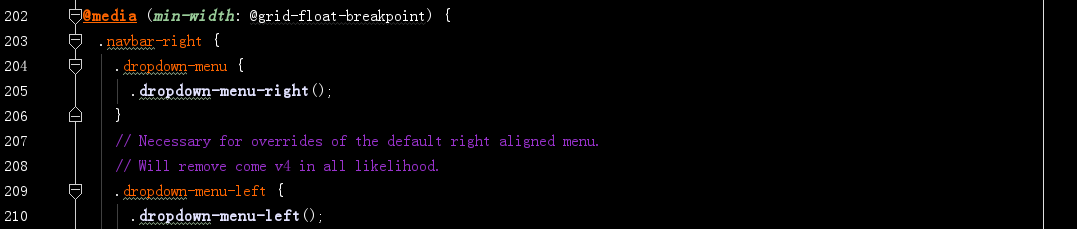
（7）多级嵌套：可在li中添加ul.dropdown-menu（bootstrap原来不行）。但li并没有加relative定位，需要手动添加dropdown样式，因为js在向上查找时，需要这个样式类。当嵌套dropdown显示时，默认显示在右边，可以使用dropdown-menu-left显示在左边。



（8）底部上拉列表。两种情况下使用：1）嵌套层为dropup（不用dropdown）；2）底部固定导航条navbar-fixed-bottom中的dropdown。没有其他设置，就是箭头朝上，top改为bottom为100%。



（9）对于导航条中的左右2部分导航栏，navbar-right下的下拉列表应该在左边，dropdown-menu-left的应该在右边。



**2.dropdown.js**

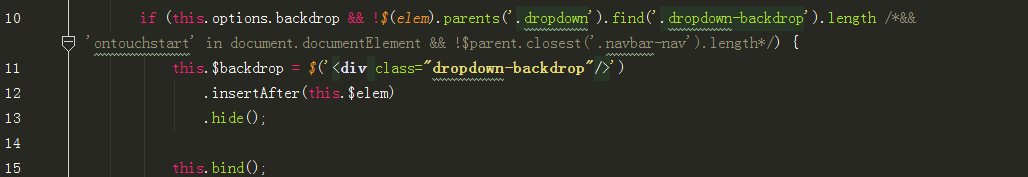
* 检测点击标签是否disabled。



* toggle下拉菜单的open类样式。



* dropdown也有backdrop（跟原bootstrap不同），默认不显示，可以通过配置控制。



特别需要注意的是，因为存在多个dropdown，所以就需要多个backdrop，否则，点击backdrop时，无法得知生成它的dropdown，导致所以dropdown-menu都toggle。这个与modal不同，modal理论上只会有1个地方显示。

* + 1. 作为按钮组的一部分

按钮组中嵌套下拉列表可以分为水平和垂直嵌套两种。





需要考虑的情况：

（1）dropdown-toggle和dropdown-menu必然包裹在btn-group中。因此，要对btn-group设置relative定位。

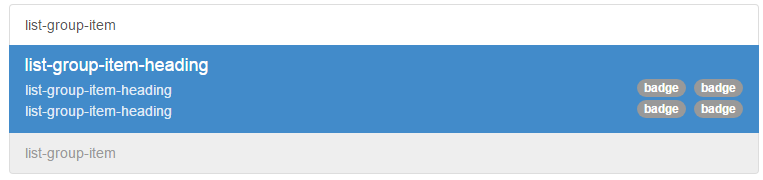
（2）鉴于btn内会有图标，因此btn也应该设置为relative定位。

以上两个问题，按钮组样式已包含了。

（3）圆角问题。只需考虑被包裹的btn-group是否是第1个和最后1个元素。和按钮组区别在于需要考虑补回在按钮组中去掉的圆角。



* 1. 列表组



1.需要处理什么问题？

（1）边框。相邻两个li直接的边框重叠和第1个/最后1个边框问题。

（2）圆角，只有第1个和最后1个才应该有圆角。

（3）内部的标题和p应该更紧凑一些，所以要减小margin-bottom。

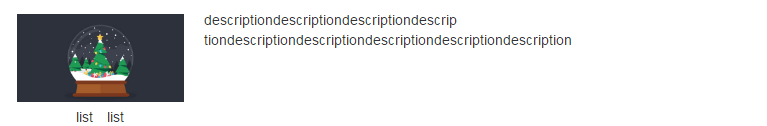
（4）badge应该右浮动。

2.样式处理

（1）所有item均设上下边框，然后margin-top=-1px遮住重叠的。

（2）内部h的margin-bottom减半，p行高增加1点，但margin取消。

* 1. 媒体对象



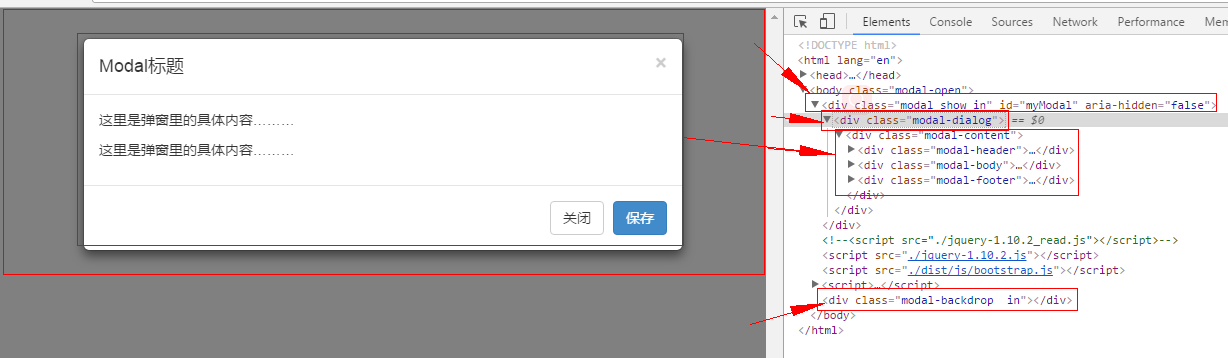
1.要解决的问题

左右布局，两者宽度分配比例、间距。

2.实现

使用浮动布局，直接使用pull-left和pull-right，两者按25-75比例分配宽度，通过设置padding区分开。

* 1. 模态框
     1. 基本样式



**1.结构样式**

有3层嵌套：.modal>.modal-dialog>.modal-content。

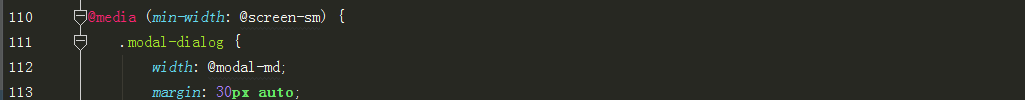
（1）.modal作用

背景容器，100%铺满屏幕，当模态框内容多于1屏时，能够滚动。

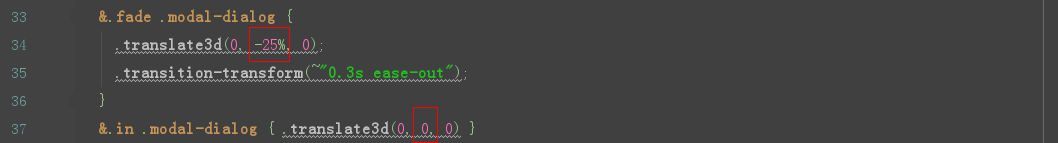


（2）.modal-dialog作用（貌似也可以直接加到.modal-content上）

包裹.modal-content，设置了宽度并水平居中，并在显示时添加了动画样式。

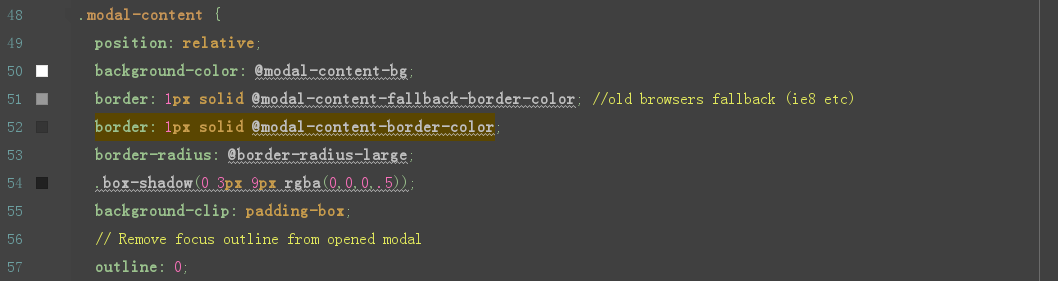


另外，.modal-dialog在模态框进入时有动画效果：从顶部-25%到0%。

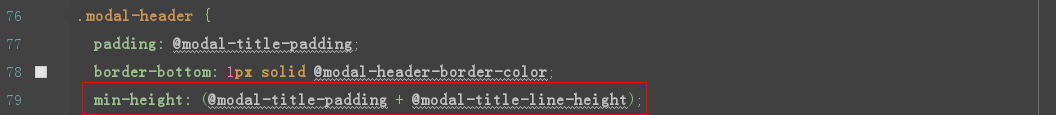


（3）.modal-content作用

真正内容载体，样式包括显示背景色、圆角、边框、阴影等，内部有.modal-header、.modal-body、.modal-footer。



* .modal-header设置了最小高度为padding+基本行高。内部的文字标题样式.modal-title的margin设置为0。





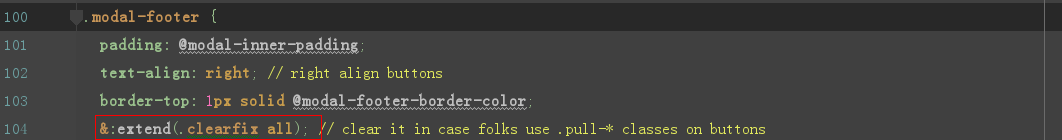


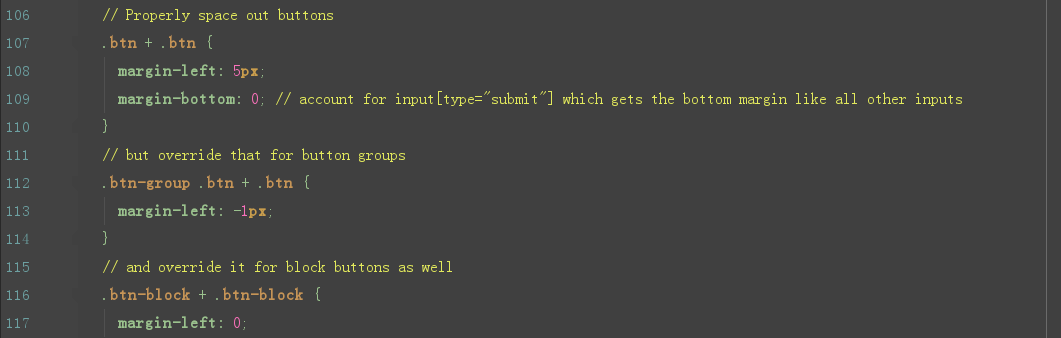
.modal-header中可能使用close，因此混入.close-dismissable。



* .modal-body和.modal-footer主要设置了padding，.modal-footer对内部的.btn做了单独设置。



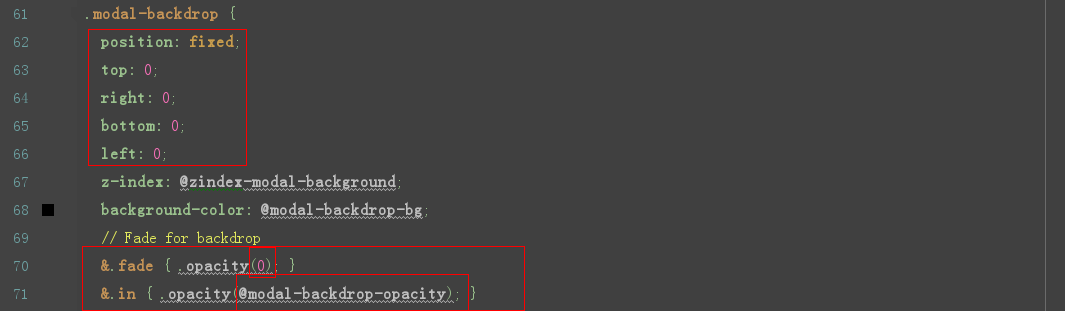




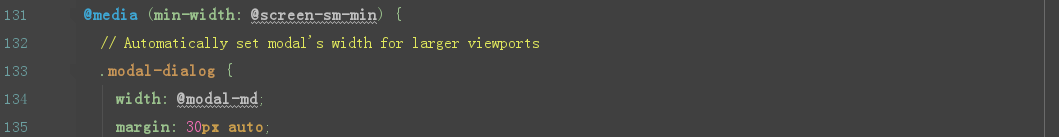
* + 1. 其他样式
* 对body设置.modal-open，保证不重复出现滚动条。

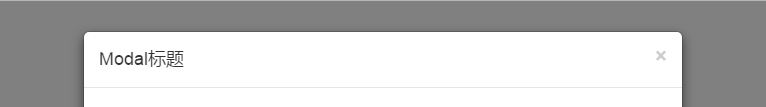


* 设置全屏.modal-backdrop，设置灰色大背景。



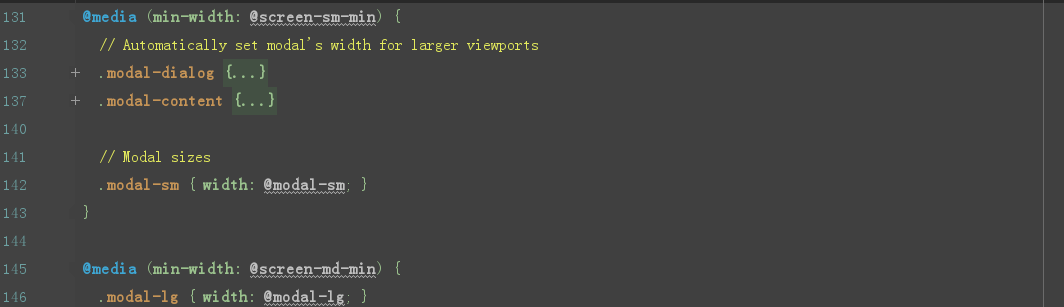
* + 1. 自适应样式
* 普通屏幕，设置.modal-dialog有宽度，保证居中，上面有30px边距。手机屏，全部宽度，顶部显示。







* 在普通屏幕，还提供.modal-sm和.modal-lg两个样式，定义.modal-dialog宽度。



* + 1. js

**1.用法**

去掉了声明式用法，只保留js用法。初始化时，可以传入3个参数：

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数类型** | **默认值** | **描述** |
| backdrop | boolean | true | True：出现灰色背景，点击该背景关闭弹窗。  Static：出现灰色背景，点击不会关闭弹窗。  False：不出现灰色背景。 |
| keyboard | boolean | true | 按ESC是否关闭弹窗。 |
| show | boolean | true | 初始化时是否显示。 |

因代码内部将字符串型入参当作调用方法名，因此，初始化之后，可以使用show、hide、toggle作为参数。

**2.事件**

支持4种类型的事件订阅，分别对应模态弹窗的弹出前、弹出后、关闭前、关闭后。

|  |  |
| --- | --- |
| **事件名称** | **描述** |
| show.bs.modal | show方法调用时触发。如果单击了1个元素，该元素将作为事件的relatedTarget属性。 |
| shown.bs.modal | modal展示之后触发。如果单击了1个元素，该元素将作为事件的relatedTarget属性。 |
| hide.bs.modal | 在hide方法调用时触发。 |
| hidden.bs.modal | 在modal隐藏之后触发。 |

**3.实现要点**

modal的样式并没什么，主要是js。js里面主要做了这么几件事：

（1）操作的主体就是最外层的.modal（即里面的element）。需要注意的是，backdrop的zindex小于modal，是无法点击到的，就是通过点击modal来模拟点击backdrop。

（2）内部绑定的事件只是点击modal是否focus和消失，无法toggle。

（3）需要手动给close等绑定插件的show/hide/toggle方法控制modal。

（4）代码里面主要是body滚动条显示问题，真正代码很少，就是给modal加上/去掉in。

**4.代码**

无论是show还是hide方法，主要作用都是显示/隐藏modal（设置display=block/none，添加/移除fade/in），添加/移除backdrop元素到body中，显示/隐藏（仍然是fade/in），触发一些事件等。

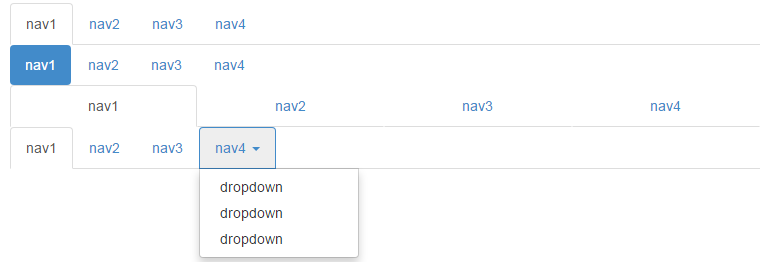
因为backdrop也有动画，因此，show/hide逻辑是在backdrop动画时候执行，因此单独定义backdrop方法，然后以回调形式传入。



添加backdrop背景需要考虑：（1）创建div.modal-backdrop标签，添加到body中。（2）根据.modal是否有fade类判断是否有动画，如果有，那么div.modal-backdrop也添加fade类。（3）添加in类显示。最后执行传入的回调。

显示.modal差不多，先将.modal添加到body中（如果还没添加，添加之后不会移除）。设置fade+in显示。

* 1. 导航元素



要解决的问题：

（1）多种样式复用？

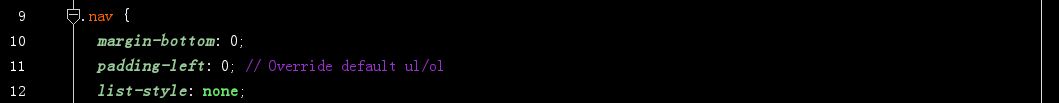
样式复用用组合样式，基础样式定义在nav中，tabs的边框自己定义，pills的背景色自己定义。

（2）对于tabs样式，如何使整个底部有边框但avtive下没有边框？

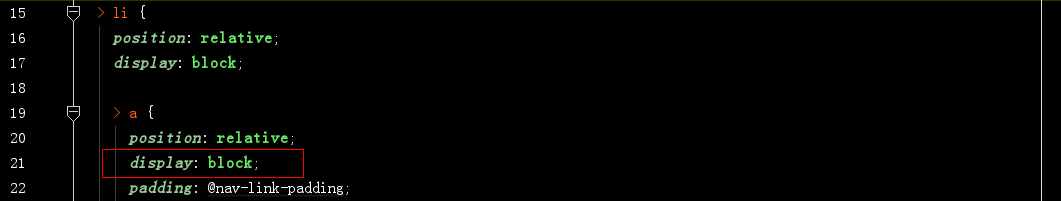
tabs边框处理：要让整个底部有边框，必须对ul上使用border-bottom，但这个边框在li/a标签外，没法覆盖，因此需要将li的margin-bottom设为-1px，这样就可以通过a来覆盖它下面的ul的边框。

**1.基础样式**

（1）在ul上应用nav样式。因为nav只是导航元素，不是定位样式，即需要包裹在定位样式内，因此应去掉定位样式margin，并去掉其默认样式。



（2）记住，所有导航样式li中的a标签都是block。



（3）设置hover/active时背景色。

**2.选项卡导航样式**



主要涉及：

（1）所有的li都必须是左浮动。

（2）边框处理：

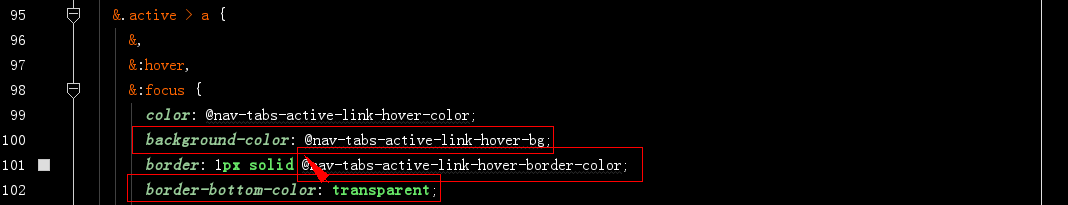
1）定义最外层nav的底部边框，这样整个nav下面都有border（否则不会铺满）。

2）因为li中的a是block，所以主要样式设置都是在a上而不是li上。

3）定义li的margin-bottom为-1px，以便a可以覆盖ul的border（内部元素在外部元素之上）。定义a四周border，颜色为transparent。因a没有background-color，此时a的border看起来就是下面ul的border颜色（特别要说明的是，li设置-1px的margin-bottom不是覆盖a的border，而只是让ul的border与a的border重叠）。

4）设置在active下的a四周边框的颜色，再设置底部颜色为transparent。因为还设置了a的background-color，因此，此时border-bottom的颜色就是a的background-color。





之所以要使用这么复杂的边框样式，是因为a设置了margin-right，否则就不要对nav设置底部边框。

**3.胶囊选项卡式导航**

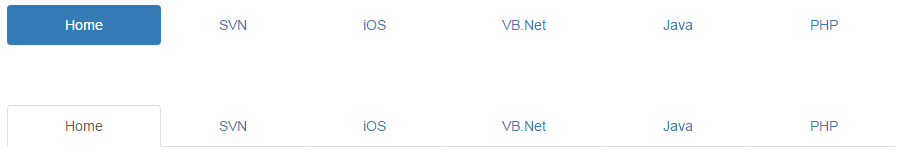


更简单一些，就是设置li为float，并设置active下a的背景色即可。

**3.堆叠式胶囊式导航**

将nav-stacked和nav-pills一块使用，也比较简单，因为要垂直排列，所以要去掉li的浮动，然后设置垂直方向上的外边距即可。

**4.分散对齐导航**



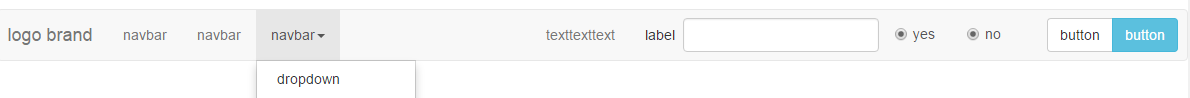
（1）手机屏下，独占1行，因此去掉浮动、设置垂直方向margin（胶囊式不考虑边框问题）。

（2）非手机屏，使用table-cell实现。需要清理nav-pills/nav-stacked设置的li的float，并对每个li设置width=1%以保证平分宽度。

（3）使用nav-tabs-justified单独设置选项卡式导航的边框问题。手机屏，显示全部边框。非手机屏，因为没有了margin-right，去掉nav的底部边框，直接在a上设置/隐藏边框。

* 1. 导航条

导航条navbar是定位元素，内包裹有导航条的标题、导航元素（上一节）、搜索框、登录等。导航元素没有使用上一节设置的风格，而是使用这里定义的navbar-nav（因为它不是tab式，也不是胶囊式）。导航条navbar基础样式为navbar，有navbar-default和navbar-inverse两种风格可选。



****

* + 1. 要解决的问题

**1.导航条的高度**

（1）导航条高度固定，但内部的button有lg、xs等尺寸，以及text等高度也并不一致。实现的核心是不改变这些组件的高度，而是通过navbar-vertical-align这个mixin，计算组件高度和导航条高度之差，设置垂直方向margin。

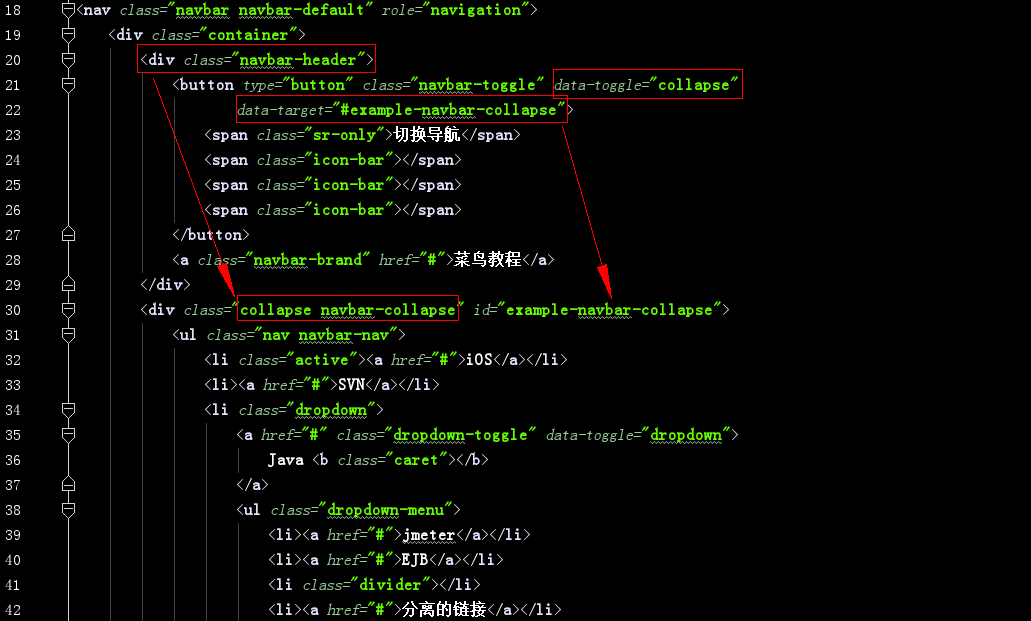


（2）导航条min-height为50px，如果折行，那么是以50px倍数增加。

**2.排列**

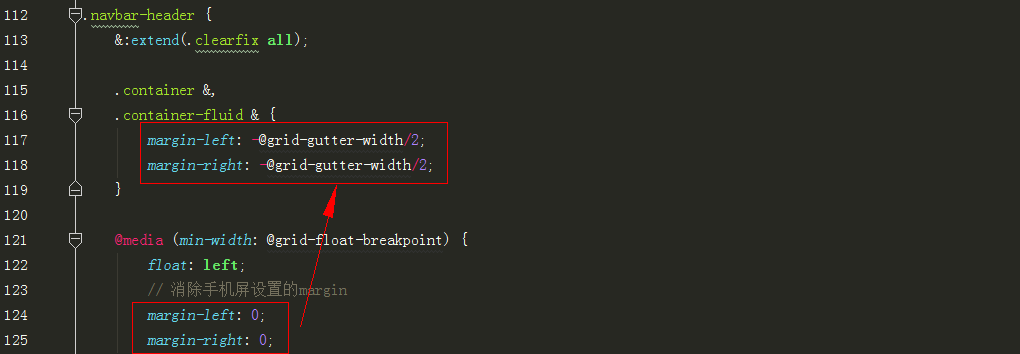
普通屏下，默认navbar-brand、navbar-header、navbar-nav都是左浮动。其他并没有设置浮动，需要使用pull-left/right。

**3.响应式**



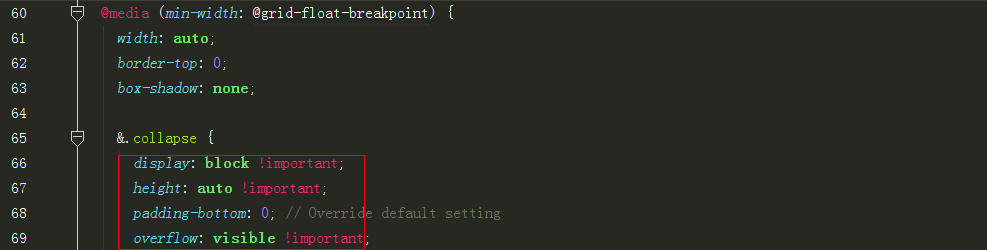
（1）navbar只包裹navbar-header和navbar-collapse两个直接子元素。

（2）普通屏下，navbar-header和navbar-collapse只是包裹元素，如同普通导航元素float外，没有特别样式。



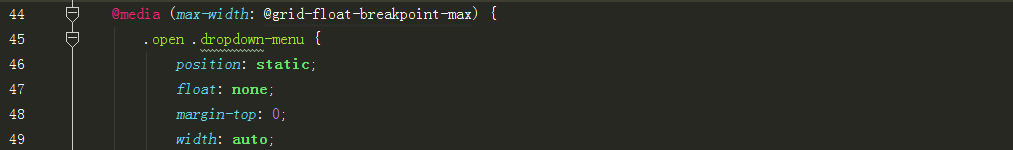
但手机屏下，navbar-header为非浮动元素，独占1行。

navbar-collapse和collapse、in配合使用，控制手机屏下的显示。在PC端不会受到影响。



（3）普通屏下，button.navbar-toggle不显示。

（4）手机屏下，.navbar下的每个子元素都会单独1行显示。



**4.手机屏的margin和padding**

（1）使用navbar-header和navbar-collapse

无论是手机屏，还是普通屏，都需要套container。普通屏没有padding，但手机屏有padding，因此必须使用负margin将container的padding去掉，否则即便navbar设置有背景，但不会包裹到navbar-collapse，列表背景显示没有占满1行。

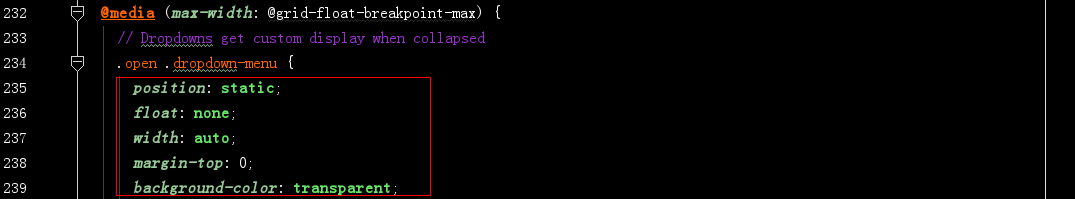
（2）没有使用响应式

不存在padding问题，因为背景色在navbar上，此时会完全包裹内部元素。

**5.导航条中的下拉菜单**

在普通屏幕中没有特别设置，主要是去掉了原来与包含块之间的1px的margin。

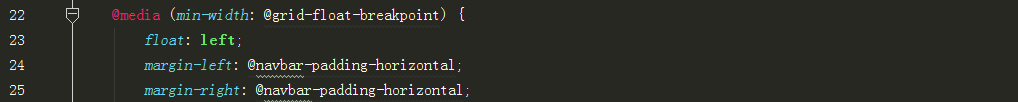
有一个地方可以学习，就是在手机端时，导航条不能显示到包含块外，而是显示到包含块内。因此不能设置为absolute，而是static。



* + 1. 样式

（1）.navbar是定位元素，只有一些定位相关样式。

（2）.navbar-nav只有尺寸样式，没有颜色样式，样式样式由风格样式决定。只有在PC端时，才会有左右margin。



手机屏则不需要，因为内部的li>a是由padding的，而且背景色就在navbar-nav上，如果有margin就显示不全。

（3）.navbar-brand已经设置好了高度。

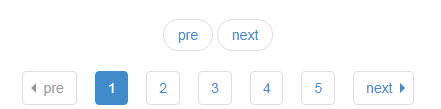
（3）.navbar-form、.navbar-btn、.navbar-text都没有其他样式，均是通过.navbar-vertical-align控制好垂直居中。

（4）对于navbar-header和navbar-collapse，两个并没有其他样式，只是1个包裹，唯一要做的就是用margin抵消掉container的padding。原因是：在手机屏时，即便navbar设置有背景，但不会包裹到navbar-collapse。

（5）.navbar-fixed-top和.navbar-fixed-bottom没其他样式，就是定位而已。

（6）.navbar-default和.navbar-inverse设置了内部元素的颜色。

* 1. 分页



包括pager和pagination两个组件，需要处理的有：

（1）颜色

正常情况，active情况，hover等情况。

（2）居中

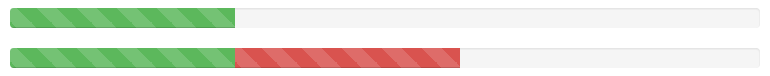
内部均为inline-block，使用文本居中方式居中。

* 1. panel

结构分为heading、body、footer，因此主要问题就是处理圆角、相邻边框问题。

可以对panel-body使用collapse。

* 1. 进度条



1.要处理的问题

（1）外层包裹内层，内层和外层高度一致。

（2）进度条能够添加。

2.处理方法

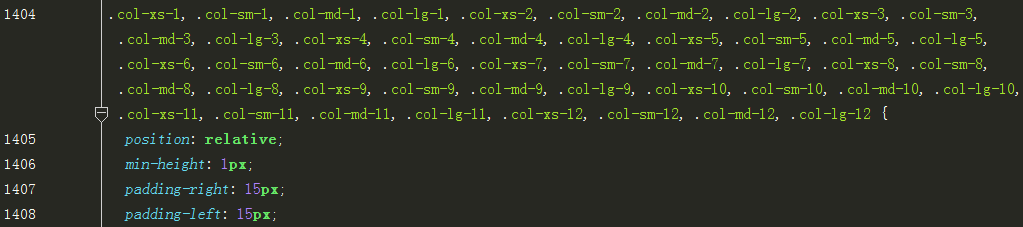
（1）外层定义高度，内层设置为height=100%（外层没有设置就无效，这里会有效果因为外层显示设置了height）。

（2）内层定义为float，这样就能够叠加。

1. 框架源码分析
   1. bootstrap
      1. 如何生成响应式格栅

**1.步骤**

（1）首先由make-grid-columns混合生成64个公共样式。

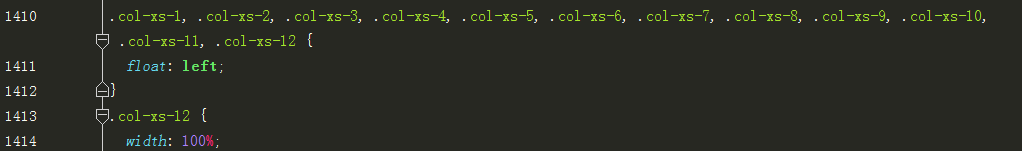


（2）由make-grid(xs)生成没有媒体查询的xs级别的columns样式。然后生成其他3类媒体查询下的columns样式。

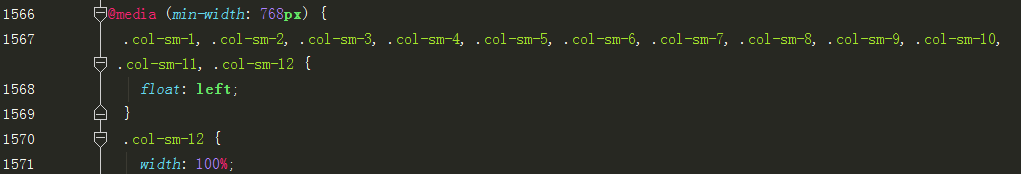




生成的xs级别的columns样式如下：



媒体查询下的样式则在媒体查询下：

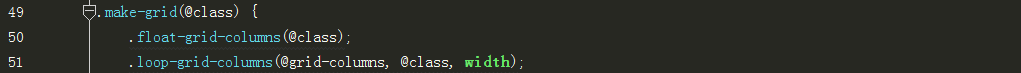


【说明】

之所以要将浮动放在每个媒体查询中单独生成，原因是如screen为800px时，如果只使用了col-lg-3。按理，col-lg-3就不应该起作用了，但如果和col-xs-\*放在一起，那么就还有浮动，所以不行。

**2.mixin说明**

以make-grid为例，make-grid会在其中引入其他mixin。



以.float-grid-columns为例，里面的步骤为：



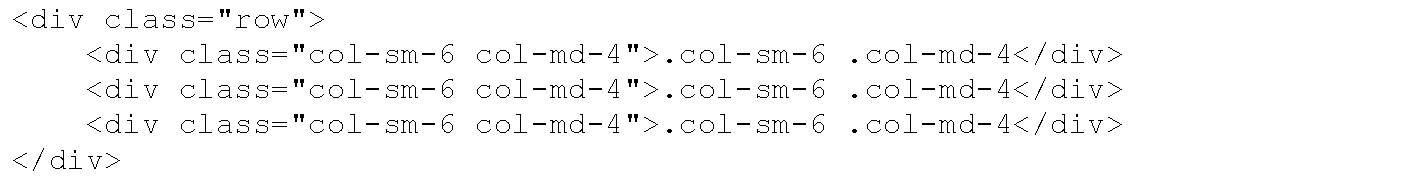
* + 1. 如何使用格栅系统

（1）样式定义上，因为xs在sm、md、lg之后，因此如果在使用上同时引用它们且满足后面的媒体查询条件，xs的样式会被覆盖。

（2）使用上，例如只应用了样式col-lg-2，当屏幕小于1200px时，相当于只有make-grid-columns()这个mixin定义的公共样式，没有make-grid(lg)定义的浮动、宽度等，导致均为块元素，都是单独一行排列。

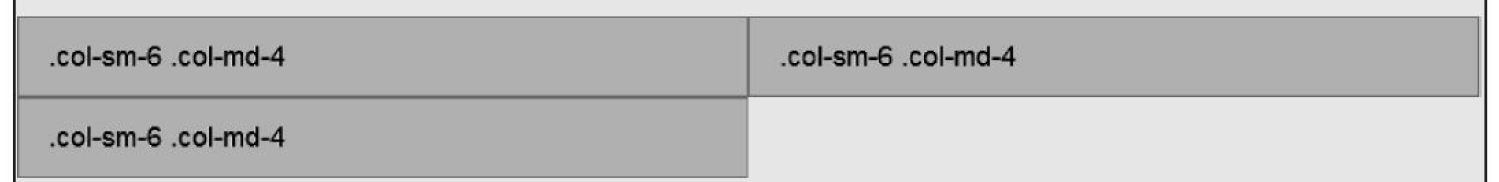


因此，要实现真正媒体查询的响应式，需要同时应用上述多个样式，如

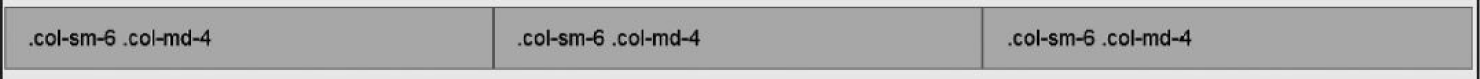


当屏幕宽度在480px以内时，单独1行显示。

屏幕宽度在768px-992px之间时，1行展示2个，并导致另外1个另起1行。



当大于992px时，1行展示3个。



【所以，响应式不仅要考虑样式，还要考虑实际的标签数量】

* + 1. bootstrap中的页面标题样式说明



page-header是页面标题区域，是1个区域，里面除了h\*之外，还可以有如浏览量等信息。

页面标题呈现的最终上下边距和均为2×line-height。.page-header和h\*的margin-top折叠，共2×line-height。下面边距分别为h\*的margin-bottom（1/2）+.page-header的padding-bottom和border-bottom合计（1/2）+.page-header的margin-bottom（1）。

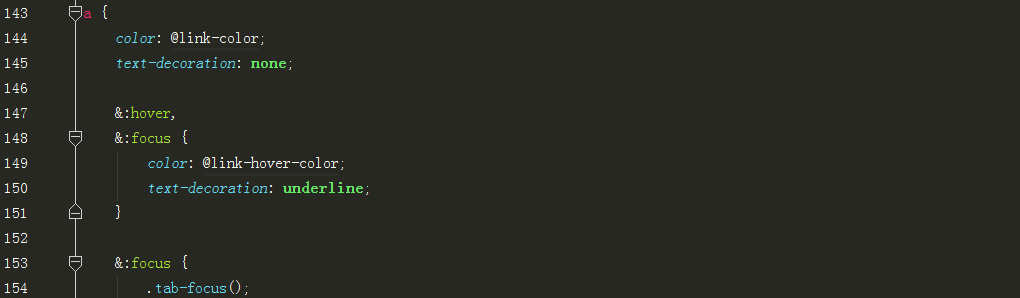
* + 1. a标签样式

（1）基础样式

a标签还有4个伪类：:link（未访问）、:visited、:hover、:active，固定顺序就是LVHA（爱恨原则LoVe/HAte）。四个伪类默认都有下划线text-decoration:underline。a标签的默认样式为：



bootstrap基础样式中只设置了link和hover的样式，没有去掉visited和active的颜色。



1. 移动端
   1. 底层

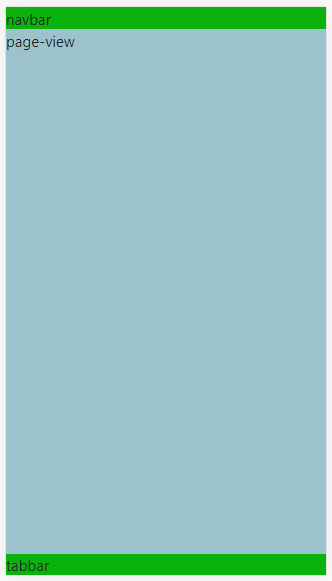
**1.m-boot.flexible.js**

（1）使用手淘移动端解决方案，将屏幕宽度分为100等分，rem为10等分。

（2）由于html设置的font-size只是用来作为基准尺寸参考的，因此还要在body上设置font-size用作基准字体大小。

（3）简单起见，不考虑多倍屏。

* 1. 基础组件
     1. page



（1）使用flex的纵向布局。navbar和tabbar高度有自己设定，page-view的flex为1，填满整个区域。





（2）为什么要使用固定高度的page-view，让滚动条出现在它上面；而不是设置navbar和tabbar为fixed定位，page-view上下设置margin，让滚动条出现在body上？

美观，上下导航栏不应该旁边有滚动条。另外还和下拉刷新可能有关。

（3）是否需要设置page的width和height为100%？

必须，不然page-view没法布满剩余空间。

* + 1. grids

（1）层级结构为.grids-title和.grids-N并列，.grids-N内部嵌套.grids-item。



（2）.grids-N使用flex横向布局，内部元素等分





（3）添加.grids-border后，.grids-N会有下边框，内部嵌套.grids-item除最后1列的元素外，会有右边框，均有下边框。



* + 1. typography

和pc版基本一致。

* + 1. table

和pc版基本一致。

* + 1. forms

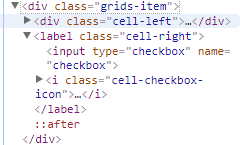
（1）和pc版不同，需要嵌套在grids中使用。使用cell-left和cell-right分开说明文字和表单。



因grids为flex布局，cell-left没有设置width，cell-right布满剩余空间。

（2）cell-right中的表单元素样式进行了优化，input、select等为占满剩余空间，没有其他样式。

（3）radio和checkbox需要嵌套在label中，label为布满剩余空间，radio和checkbox绝对定位隐藏，有cell-radio-icon和cell-checkbox-icon两个图标，根据radio和checkbox是否checked而显示不同图标。



（4）有switch样式，也是通过before和after实现。

* + 1. button

和pc版基本相同。

* 1. 扩展组件
     1. actionsheet



（1）布局actionsheet为fixed定位，默认transform: translate(0, 100%)，附加actionsheet-toggle时，transform: translate(0, 0)。

（2）内部上面部分为actionsheet-item，就是底部有边框，没其他样式。

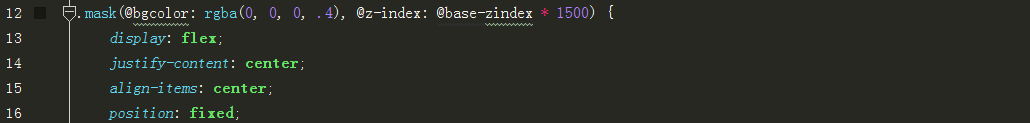
（3）下面的按钮样式为actionsheet-action，有margin-top，没其它样式。

* + 1. dialog
       1. confirm/alert





（1）在屏幕中居中，原因为它们是mask中的元素。mask为flex，设置了内部元素水平和垂直居中。



（2）主要是底部样式dialog-confirm-ft，为flex，只要不是最后1个元素都有right-border。

* + - 1. toast



（1）同样在mask中。

（2）内部可以有success、error、loading三种图表可选。

* + - 1. notify



顶部通知，样式为fixed，动画进入。

* + 1. keyboard



（1）外面布局和actionsheet类似，fixed。

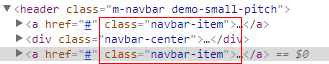
（2）keyboard-head有border-bottom；密码框keyboard-password为flex布局，内部的li的flex属性值为1，只要不是最后1个都有有边框，里面的元素为1个黑点；然后是keyboard-numbers，类似九宫格，分为4行，只要不是最后1个元素都有右边框、上边框。

* + 1. navbar



（1）布局容器navbar并没有定位，需要加上navbar-fixed才会是绝对定位。

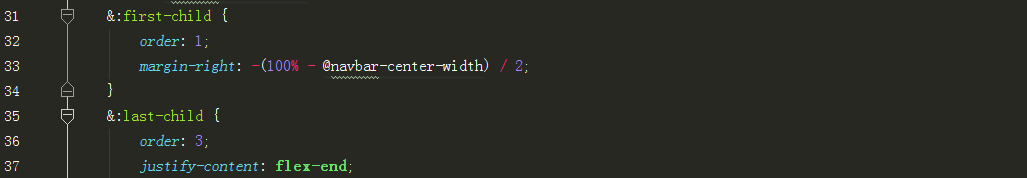
（2）navbar为flex，左、右为navbar-item，中间为navbar-center。



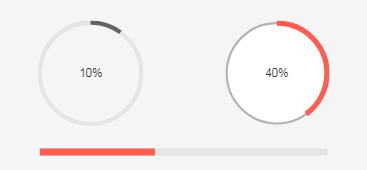
（3）定义了navbar-center的宽度，然后根据这个宽度计算navbar-item的宽度。



（4）通过伪类选择权，设置navbar-item左右margin以及order。



* + 1. progress-bar



（1）通过svg实现。

（2）主要是js实现，这里只是定义容器progress-bar为相对定位，内部progressbar-content说明文字居中。

* + 1. spinner

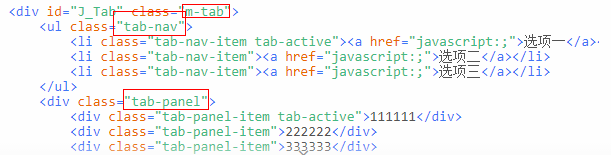


（1）外面的spinner为inline-block，内部左右为a标签，中间为input，均为float。

（2）内部元素没有边框，只要spinner有边框。

* + 1. tab

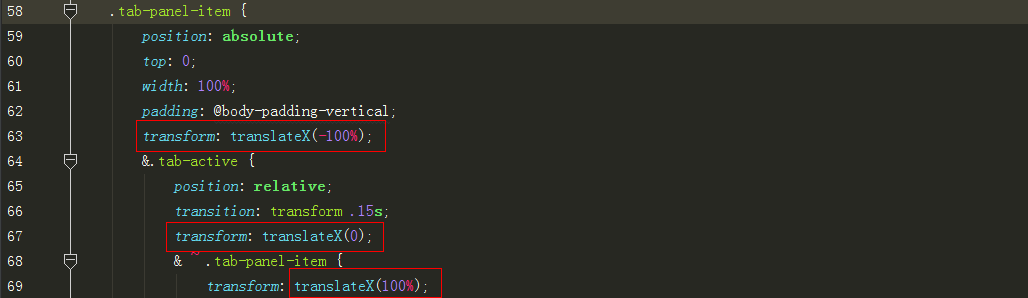




（1）外面的tab只是包裹，没有任何样式。

（2）tab-nav为flex，内部tab-nav-item等分。

（3）tab-panel相对定位，内部tab-panel-item为absolute，通过translateX控制动画方向。



* + 1. tabbar



（1）tabbar为flex，内部tabbar-item等分。如果需要底部固定，需要加tabbar-fixed样式。