17 页面布局的设计与开发

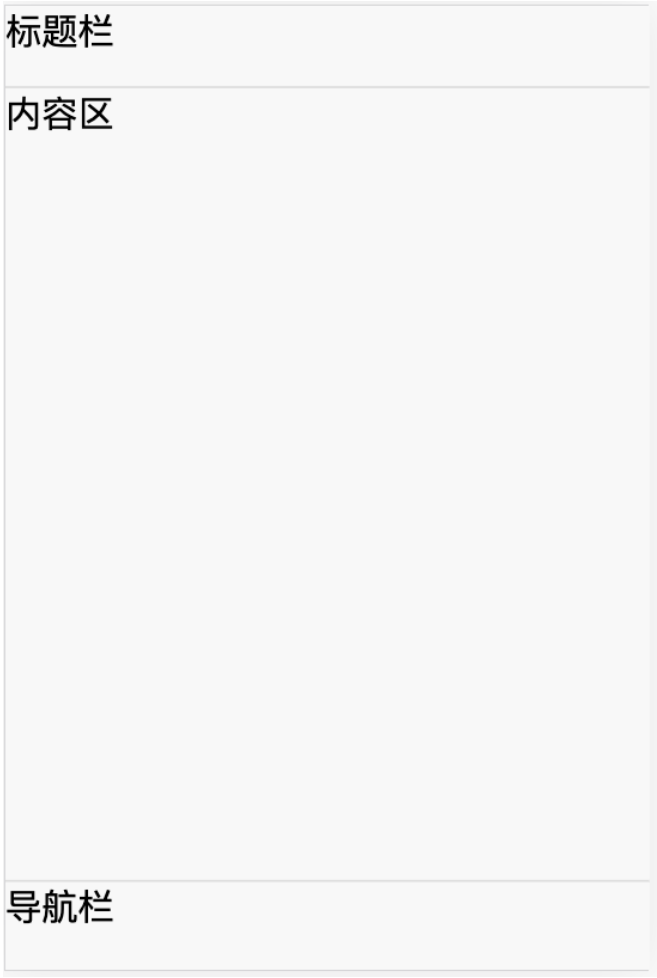
更新时间：2019-07-05 10:14:28

> 人不可有傲气，但不可无傲骨。
>
> ——徐悲鸿

这一节，我们来进行页面布局的开发。在移动端，因为屏幕空间有限，所以布局的方式也比较固定，一般就是上面一个标题栏用来放页面标题，中间的部分是内容区用来显示网页内容，最下面一般是导航栏。我们这一节就来拆解一下做页面的布局要做那些事情。

页面布局设计

移动端页面的基本结构就像下图这样：



在移动端布局中：

- **最上面**的头部标题栏区域通常是固定在最上面，不随着页面进行滚动；
- **内容区**是页面的主要部分，里面的内容可以在内容区内上下滚动；
- **最下面**是固定的导航条，也会固定到页面的最下面，也不随内容区滚动。

这三部分区域只有**内容区**是一定会有的，**头部标题栏**和下面的**导航栏**在有些情况下并不需要，比如在一些嵌入 App 的页面就不需要标题栏，有些二级三级页面就不需要导航栏。因此在考虑内容区的布局的时候，要把对另外两个区域适配的适配考虑进去，保证内容区的内容既不会被遮挡，也不会留下大片的空白区。

在移动端布局中还有另外一个问题，就是各个区域的层级关系。前面章节在讲 z-index 的时候说过，把内容区放在最底层，标题栏和导航条放在第二层，页面蒙版放在最上层。

设计上面所说的这样一个布局，就要同时要达到如下这些要求：

1. 头部标题栏固定在屏幕最上面，且和屏幕等宽。
2. 底部导航条固定在屏幕最下面，且和屏幕等宽。
3. 内容区可以滚动。
4. 内容区滚动的时候不被另外两个区域遮挡。
5. 当没有标题栏或者导航条的时候，内容区也能填满空白区域。
6. 蒙版层在最上层显示，铺满全屏，并且可以覆盖住标题栏、内容区和导航栏的所有内容。

一会我们就来按着这样的要求来做出一个合格的页面布局。

固定定位

在做布局的时候，因为标题栏和导航栏都要固定在屏幕的固定位置，且不跟随内容区进行滚动。为了实现这个效果，最先能想到的就是使用固定定位（position: fixed）。所以我们在这把固定定位讲解一下。

固定定位是将元素固定在屏幕可视区（viewport）的固定位置，位置不会随着其他内容的滚动而变化。之前讲过使用固定定位会使元素脱离文档流，并且会晚于正常文档流里的元素渲染的特点。除此之前，固定定位的元素还有一个比较重要的特性，就是它的宽和高都是相对于可视区的，和它的父级元素祖先元素都没关系。

概念性的东西不太多，我们这里主要是给出几种固定定位的常用用法。

一、固定的横栏

在实际项目中有很多需要固定的横栏，比如前面所说的标题栏。这种元素和屏幕等宽，有一定的高度，然后固定在某个位置。在实现这类元素样式的时候，我们以标题栏为例，就可以用下面的样式：

```
<!-- 标题栏 -->
<div class="header"></div>

/* 标题栏 */
.header{
    position: fixed;
    width: 100%;
    height: 50px;
    left: 0;
    top: 0;
}
```

上面这个样式里，一定要注意给元素设置 left 和 top 属性，这样才能以屏幕左上角为起始点。如果不设置定位的位置，这个元素的起始位置会是它在脱离文档流之前所占的位置。

二、页面蒙版

在开发中，还有一类比较常见，就是我们用的页面蒙版。在做一些操作时，为了避免干扰会用蒙版把没用的内容都遮住。根据固定横栏的经验，全屏覆盖想起来就很简单了，把height 也换成 100% 不就可以了。但是这里我们要注意，这种功能靠百分比做的全屏蒙版不太灵活。假如需要蒙版把标题栏的部分留出来，这时候靠百分比就不灵了。所以在做蒙版的时候通常使用下面这种方式：

```
<!-- 页面蒙版 -->
<div class="mask"></div>

/* 页面蒙版 */
.mask{
    position: fixed;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    background: rgba(0,0,0,.5);
}
```

这其中给每个方向上定位的值都设置为 0，就会把蒙版撑开到全屏。这样做的好处是假如需要留出来标题栏，就可以调整 top 值来实现，需要留下面导航栏就调整 bottom 值来实现，非常方便。

三、水平竖直居中

有些固定元素需要在屏幕最中间显示，比如登录框这些。这种需求就可以用下面的方式来实现：

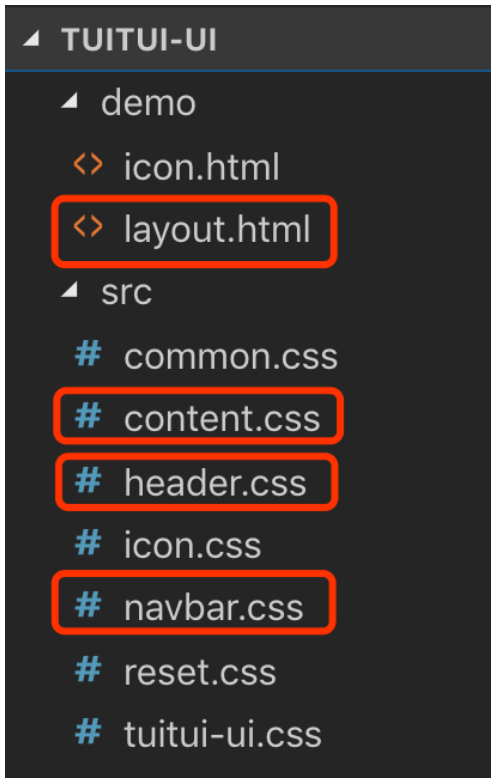
```
<!-- 登录框 -->
<div class="login-wrap"></div>

/* 登录框 */
.login-wrap{
    position: fixed;
    width: 300px;
    height: 400px;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    margin: auto;
}
```

这组样式看起来和上面的差不多，但当给元素设置好固定的长和宽的时候，它不会出现在屏幕的正中间了。只有给加上“margin: auto” 样式，它才能处在页面的正中心。

关于固定常用的情况就先说这几种。这里给同学们留一个思考题，假如我想让这个高 400px、宽 300px 的登录框再向上提高页面高度 10% 的距离，应该怎么做？

页面布局的开发

补充完基础知识，我们就可以进入布局的开发了。这里我们先建立几个文件：



这里面 /demo/layout.html 文件是一开始就建立好的，用来测试页面布局的样式；src 目录下圈出来的三个文件是新建出来的，分别用来存放标题栏、内容区和导航条的样式；还有一些其他的样式我们放在 common.css 里。在移动端的布局中，最容易想到的就是让内容区是正常的流式布局，然后把上下两个盒子使用固定定位来固定在页面的头部或尾部。下来我们就按着这种思路来进行布局。

首先我们先填充 layout.html 文件：

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0,user-scalable=no">
    <link rel="stylesheet" href="../src/tuitui-ui.css">
    <title>页面布局</title>
  </head>
  <body>
    <div class="tt-header">
      标题栏
    </div>
    <div class="tt-navbar">
      导航栏
    </div>
    <div class="tt-content">
      内容区
    </div>
    <!-- 页面蒙版 -->
    <div class="tt-mask"></div>
  </body>
</html>
```

这个文件中我们在 body 里放了四部分内容，分别是标题栏、导航栏、内容区和页面蒙版。细心的同学会发现，这个HTML 文件里导航栏在内容区的上面了。这是因为后面我们要通过判断是不是有标题栏和导航栏来控制内容区的样式，所以把内容区放在后面就可以使用兄弟选择器来操作内容区的样式了，省去了 JS 的工作。

下来我们来开发 CSS 文件。首先要在 /src/tuitui-ui.css 文件里把刚才新建好的三个文件加进去，直接在文件末尾追加即可。追加后的 tuitui-ui.css 文件内容如下；

```
/*
 * @Author: Rosen
 * @Date: 2019-06-20 23:08:44
 * @Last Modified by: Rosen
 * @Last Modified time: 2019-06-23 22:22:59
 */

/* css reset */
@import './reset.css';
/* 公共样式 */
@import './common.css';
/* 字体图标 */
@import './icon.css';

/* 标题栏样式 */
@import './header.css';
/* 内容区样式 */
@import './content.css';
/* 导航条样式 */
@import './navbar.css';
```

接下来我们还要设置一下 body 的基础样式。我们先在 /src/common.css 文件中设置 body 的样式，我们直接在文件末尾直接追加：

```
/* body默认样式 */
body{
  max-width: 640px;
  margin: 0 auto;
  background: #f8f8f8;
  overflow-x: hidden;
  -webkit-overflow-scrolling: touch;
}
```

关于body的这几个样式，它们的作用是：

- max-width: 640px; 设置 body 最宽只能 640px，这是为了在一些超宽的屏幕（如：横屏的ipad）上，页面不至于被撑得太宽，我们这里限定屏幕像素大于 640px 的时候，只显示到 640px 这么宽。
- margin: 0 auto; 当屏幕超宽时，保证内容区能水平居中。
- background: #f8f8f8; 给页面设置一个浅灰色背景。
- overflow-x: hidden; 横轴方向如果出现了超宽的元素，设置为隐藏，这样就能防止在X轴方向上出现滚动条。
- -webkit-overflow-scrolling: touch; 在 ios 系统上，可以让滚动元素带有弹性，滚动的更顺滑。注意这条样式只是 webkit 内核提供的，只在基于 webkit 内核的浏览器上有效。

下来我们就能开始填充另外几个 CSS 文件了。

/src/header.css :

```
/*
 * @Author: Rosen
 * @Date: 2019-06-23 22:38:24
 * @Last Modified by: Rosen
 * @Last Modified time: 2019-06-23 22:46:30
 */

/* 头部导航条 */
.tt-header{
    position: fixed;
    box-sizing: border-box;
    width: 100%;
    max-width: 640px;
    height: 2.3rem;
    top: 0;
    z-index: 200;
    border-bottom: 1px solid #ddd;
}
```

这个头部导航和在固定定位里讲的横栏的开发很相似。但是这个导航条的样式和刚才讲的还不太一样。

- 这里面没有使用“left: 0;”来把标题栏放在窗口最左面，这是因为如果不给它限定水平方向位置，那么它起始的位置就时它脱离文档流之前的位置，也就是现在 body 的起始位置。这样在超宽屏幕下也能保证标题栏也在 body 的左上角，而不是屏幕的左上角。
- **box-sizing: border-box;** 使用了怪异盒模型，是因为使用了border来分隔标题栏和内容区，这样标题栏的高度就是“height: 2.3rem;”所指定的高度了。
- **max-width: 640px;** 和 **body** 统一，限定了最大宽度就是 640px，不限定的话导航条的宽度就变成了整个屏幕的宽度，在超宽屏幕下会造成错乱。

这就是 **/src/header.css** 文件的内容及说明。

/src/navbar.css :

```
/*
 * @Author: Rosen
 * @Date: 2019-06-23 23:00:22
 * @Last Modified by: Rosen
 * @Last Modified time: 2019-06-23 23:01:08
 */

/* 底部导航栏 */
.tt-navbar{
    position: fixed;
    box-sizing: border-box;
    bottom: 0;
    width: 100%;
    max-width: 640px;
    height: 2.5rem;
    border-top: 1px solid #ddd;
    z-index: 200;
}
```

这个文件的布局样式和导航条十分相似，只不过定位的时候使用的是bottom代替了导航条的top，把元素固定在了页面的底部。

/src/content.css :

```
/*
 * @Author: Rosen
 * @Date: 2019-06-23 22:56:56
 * @Last Modified by: Rosen
 * @Last Modified time: 2019-06-23 23:08:27
 */

/* 内容区 */
.tt-content{
    box-sizing: border-box;
    position: relative;
    overflow-y: auto;
    padding-top: 2.3rem;
    padding-bottom: 2.5rem;
}
```

上面就是内容区的样式，我们来逐条解释下:

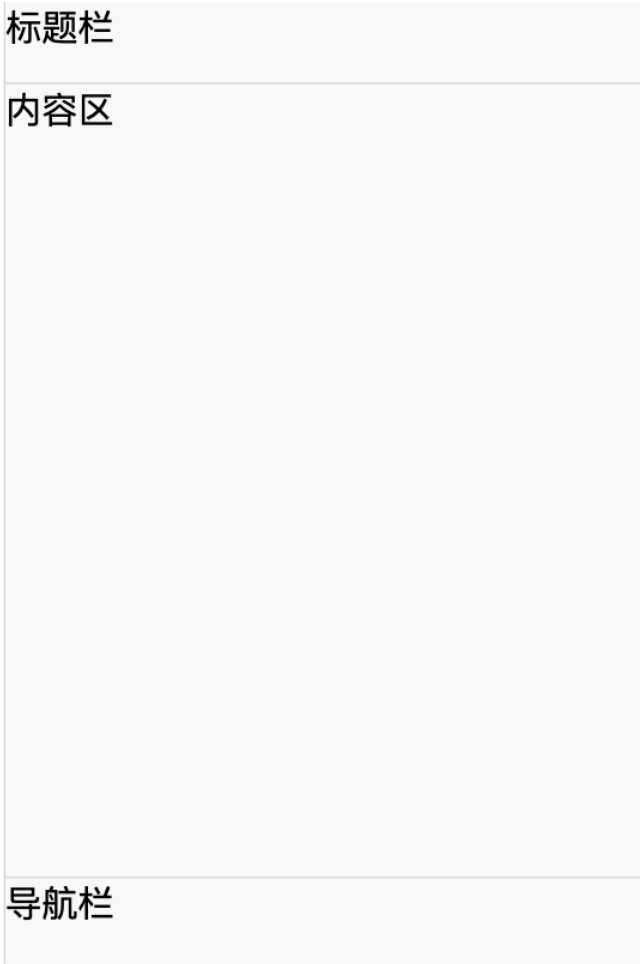
- **box-sizing: border-box;** 使用怪异盒模型，是为了能正确处理后面的padding。
- **position: relative;** 内容区设置为相对定位，是为了让子元素在做绝对定位的时候以它为参照物。
- **overflow-y: auto;** 内容区高度超出屏幕高度的时候，自动出现滚动条。
- **padding-top: 2.3rem;** 内容区上方用内边距让出标题栏的高度，这样内容显示的位置正好在标题栏的下方，不会被标题栏遮住。
- **padding-bottom: 2.5rem;** 内容区下方用内边距让出导航栏的高度，滚动条拉到最下面时正好能把内容显示完整。

我们刚才提到的要求里，有一条是当标题栏和导航条不显示时，内容区能自动适配。如果按着刚才的样式，在没有标题栏或导航条的时候，响应的位置就是一片由 padding 撑出来的空白。为了自适应，我们可以把刚才的代码做个调整，改成如下的样子：

```
/*
 * @Author: Rosen
 * @Date: 2019-06-23 22:56:56
 * @Last Modified by: Rosen
 * @Last Modified time: 2019-06-23 23:13:39
 */

/* 内容区 */
.tt-content{
    box-sizing: border-box;
    position: relative;
    overflow-y: auto;
}
/* 根据header和navbar自动适应内容区高度 */
.tt-header ~ .tt-content{
    padding-top: 2.3rem;
}
.tt-navbar ~ .tt-content{
    padding-bottom: 2.5rem;
}
```

这样在默认的情况下内容区是没有上下内边距的，当页面有了标题栏的时候，再用**兄弟选择器**给 tt-content 元素设置上“padding-top: 2.3rem;”。同理，在有导航栏的时候，也用兄弟选择器给内容区设置上“padding-bottom: 2.5rem;”。这样就达到了**内容区自适应**的需求。

经过这几个步骤，我们的页面布局基本就成型了，我们可以看下现在的效果：

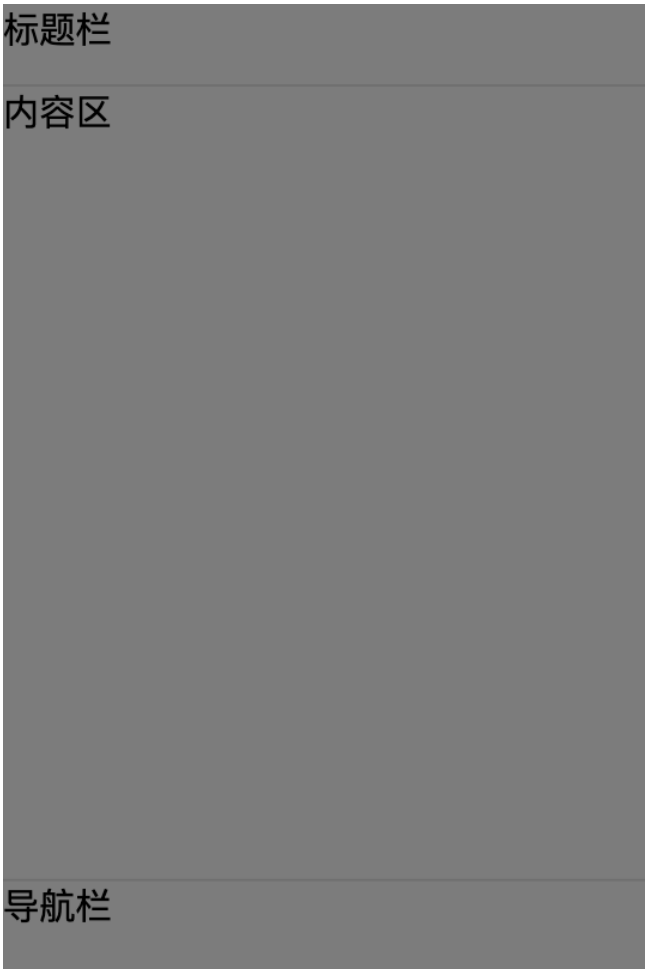


有了这个布局我们就可以再往里面填充内容了。

这一节的最后，我们还有一个蒙版层要实现。这个蒙版层会在很多地方用到，比如提示信息、modal 弹窗等，所以我们把它的样式放在 **/src/common.css** 文件里。我们只要在文件的末尾追加一个蒙版的样式即可：

```
/* 页面蒙版 */
.tt-mask{
    position: fixed;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    background: rgba(0,0,0,.5);
    z-index: 200;
}
```

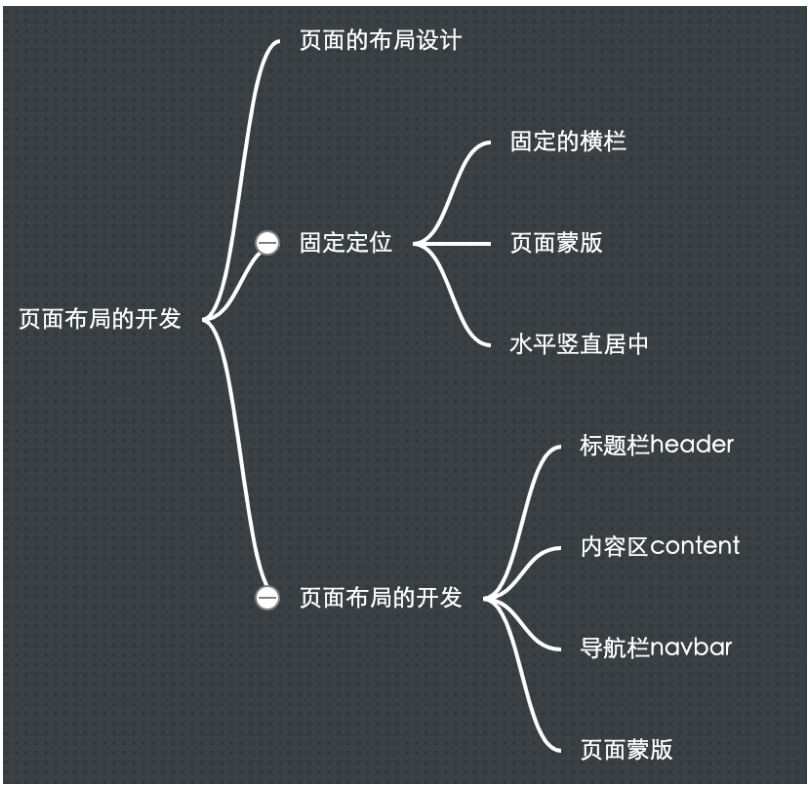
这个蒙版可以直接使用刚才讲的用固定定位制作蒙版的方式，这条样式加上以后，页面上就多了一个全屏的黑色半透明的蒙版：



这样我们这一节最后一项工作也完成了。

结语

到这里我们这一节的内容就完成了，这一节我们分析了移动端页面是怎么布局的，然后又介绍了固定定位的常见用法，最后完成了项目的页面布局。这一节的内容大纲如下：



经过这一节的学习，我们可能会发现，进入开发以后课程进行的速度反而快了，这都怪那些磨人的概念。

这一节我们使用的是固定定位的方式去完成页面的布局的，同学们下去可以思考一下还有没有别的方式去完成这个定位？提示一下，可以使用绝对定位。

到这里我们这一节的任务就完成了，同学们可以访问【页面布局在线预览】来查看这一节的演示效果，下一节我们进入布局组件的开发，把这节里做的几个区域填充起来。

16 字体图标的使用
18 布局组件的开发