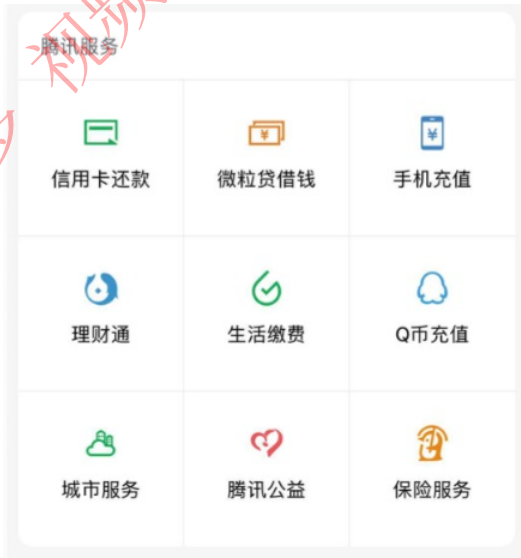
25 Grid网格样式的设计与开发

更新时间：2019-07-24 13:32:30

> 世界上最宽阔的是海洋，比海洋更宽阔的是天空，比天空更宽阔的是人的胸怀。
>
> ——雨果

这一节我们要来实现网格组件。在移动端中，网格组件通常用于页面中多个导航入口的展示。比如下面这两种：



这个是微信支付里的引导入口，使用的是九宫格形式。

这个是淘宝首页的引导入口，也可以使用网格系统来实现。

我们这一节要实现的就是这种网格组件。

网格组件的需求

在实现网格组件的时候，我们把它分成两部分来设计。一部分是最常用的标准九宫格，另一部分就是在九宫格的基础上再延伸出其他多列的网格。

一、九宫格

我们先来分析九宫格的需求，我们需要做的九宫格样式如下：

对于这个九宫格，我们有如下的要求：

1. 九宫格中多个盒子分组排列，每行三个，可以排列多行。
2. 九宫格每两个盒子中间都是用边框区分，整个容器上下有边框，左右不需要边框。
3. 每个盒子内容区水平竖直居中。
4. 每个格子之间的边框也可以去掉。

二、多列网格

在九宫格的基础上，我们要实现多列网格，样式如下：

对多列网格我们有如下要求；

1. 可以通过 class 直接指定网格的列数，列数可以从 2 列一直到 5 列。

2. 多列网格左右两侧也没有边框。
3. 格子的高度能根据列数的不同自行调整。

有了这些需求，我们就可以进行组件的设计和开发了。

网格组件的设计与开发

一、文件的建立

第一步还是建立文件，我们先建立空白的 HTML 文件 /demo/grid.html：

```
<!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>推推UI-网格组件</title>
  </head>
  <body>
    <div class="tt-content">
      <!-- 九宫格 -->
      <div class="tt-grid">

      </div>
      <br>
      <!-- 多列网格 -->
      <div class="tt-grid">

      </div>
    </div>
  </body>
</html>
```

然后是对应的 CSS 文件 /src/grid.css:

```
/*
 * @Author: Rosen
 * @Date: 2019-07-20 18:17:39
 * @Last Modified by: Rosen
 * @Last Modified time: 2019-07-20 22:14:47
 */

/* 网格组件 */
```

最后在 /src/tuitui-ui.css 主文件最后引入刚才添加的 grid.css 文件：

```
/* 网格组件的样式 */
@import './grid.css';
```

这样，这一节的文件就建好了。

二、九宫格的设计与实现

在实现九宫格的时候，我们要考虑的就是多个格子怎么分行排布。这种多行多列并且要在水平方向上平分空间的布局，和 table 元素的特点十分吻合。但是，如果使用 table 布局的话，那么网格的列数要固定下来的，这样在后面做多列表格的时候就需要再去改动 HTML 的结构，不是很方便。

接下来可以考虑使用 float 来实现这个效果，这样可以给格子定义好宽度，然后每行排满了就会自动从下一行开始排列。这种方案是可以的，唯一的不好就是定义盒子的宽度上，如果是 3 格或者 6 格这种非整数的百分比，就有可能出现不能占满全屏的情况，另外宽度的比值也要自己去计算。

最后还有一种选择，就是使用弹性布局。我们之前用的弹性布局的例子都是排布在同一行，但是弹性布局也是支持多行排列的，使用弹性布局的 flex-wrap 即可实现。

> **@ Tips:**
> flex-wrap 属性是弹性布局中用来指定弹性盒子里的元素换行方式的，它有以下三个可取的属性值：
> **nowrap**：这个属性表示弹性盒子里的元素在放不下的时候不换行，如果元素总宽度超过容器宽度，就会根据 flex-shrink 属性指定的方式进行压缩。
> **wrap**：这个属性表示弹性盒子里的元素在放不下的时候自动进行折行。
> **no-wrap**：这个属性也表示弹性盒子里的元素放不下时进行折行，但是这种折行方式比较特殊，折出来的多行是从下向上排列的。
>
> 关于 flex-wrap 属性要注意的是，它可以和 flex-grow 属性配合使用，能对盒子进行拉伸，保证每一行都是充满的。但是 flex-wrap 属性和 flex-shrink 属性是冲突的，当弹性盒子指定了可以换行的情况下，容器空间不足的情况下就会折行，而不会再去压缩盒子里的元素。

有了 flex-wrap 属性，我们就可以很容易的实现九宫格组件了。我们先在 /demo/grid.html 里填充上九宫格的 HTML 结构：

```
<!-- 九宫格 -->
<div class="tt-grid">
  <div class="tt-grid-item">
    <i class="fa fa-area-chart tt-grid-icon"></i>
    <p class="tt-grid-label">格子1</p>
  </div>
  <div class="tt-grid-item">
    <i class="fa fa-area-chart tt-grid-icon"></i>
    <p class="tt-grid-label">格子2</p>
  </div>
  <div class="tt-grid-item">
    <i class="fa fa-bar-chart tt-grid-icon"></i>
    <p class="tt-grid-label">格子3</p>
  </div>
  <div class="tt-grid-item">
    <i class="fa fa-area-chart tt-grid-icon"></i>
    <p class="tt-grid-label">格子4</p>
  </div>
  <div class="tt-grid-item">
    <i class="fa fa-area-chart tt-grid-icon"></i>
    <p class="tt-grid-label">格子5</p>
  </div>
  <div class="tt-grid-item">
    <i class="fa fa-bar-chart tt-grid-icon"></i>
    <p class="tt-grid-label">格子6</p>
  </div>
  <div class="tt-grid-item">
    <i class="fa fa-area-chart tt-grid-icon"></i>
    <p class="tt-grid-label">格子7</p>
  </div>
  <div class="tt-grid-item">
    <i class="fa fa-area-chart tt-grid-icon"></i>
    <p class="tt-grid-label">格子8</p>
  </div>
  <div class="tt-grid-item">
    <i class="fa fa-bar-chart tt-grid-icon"></i>
    <p class="tt-grid-label">格子9</p>
  </div>
</div>
```

这里总共放上 9 个格子，每个格子里的包括了一个图标和一个格子的名称。在排布格子的时候，要注意的一个是刚才提到的 flex-wrap 的用法，另外一个就是盒子边框的设计。按着需求里对边框的要求，我们可以对边框进行如下设计：

首先在垂直方向上，每两个格子间的边框可以给每个格子下边框来实现，如图中红色边框所示的位置。然后在水平方向上格子间的边框可以给每个盒子右边框来实现。但是这样九宫格最右侧就会多出一格边框，我们再把最右侧一列格子的边框取消就可以达到图中的效果，也就如图中绿色边框所示。最后，就是给整个盒子容器加一个上边框，也就是黄色的边框部分，就可以实现需求里要求的边框样式。下来我们来实现一下，在 /src/grid.css 文件中添加如下代码：

```
/* 网格组件 */
.tt-grid{
    display: flex;
    flex-wrap: wrap;
    border-top: 1px solid #ddd;
}
/* 网格中的格子 */
.tt-grid > .tt-grid-item{
    position: relative;
    flex: 1 1 33%;
    box-sizing: border-box;
    padding: 1.2rem 0;
    text-align: center;
    border-right: 1px solid #ddd;
    border-bottom: 1px solid #ddd;
    background: #fff;
}
/* 默认是3列 */
.tt-grid .tt-grid-item:nth-child(3n){
    border-right: none;
}
```

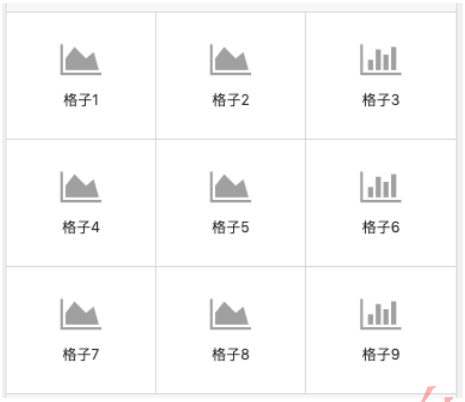
这里有几点要注意：

1. 这个格子的容器 .tt-grid 上加了“flex-wrap: wrap;”属性，然后在格子上加入“flex: 1 1 33%;”属性即可完成格子的排布要求，这里分三列所以我们选一个比 1/3 稍小的百分比，然后通过弹性让每行的三个格子撑满整行。
2. 每个格子要有边框，所以每个格子都是用“border-box”形式的盒模型。
3. 我们给每个格子右边框和下边框，然后再通过“:nth-child(3n)”这个伪类选择器来取消掉最右一排格子的右边框。
4. 通过垂直方向的 padding 值撑开了盒子的内容。

下面来处理里面图标和盒子名称的样式，这两个元素的样式也比较简单了：

```
/* grid内容区 */
.tt-grid > .tt-grid-item > .tt-grid-icon{
    font-size: 1.5rem;
    color: #aaa;
    margin-bottom: .5rem;
}
.tt-grid > .tt-grid-item > .tt-grid-label{
    font-size: .6rem;
    color: #333;
}
```

格子里这两个元素就是简单的文本样式，这样整个九宫格的排布和边框的处理就完成了，效果如下图：



接下来我们处理一下不需要边框的情况，也就是我们需要一个类把容器和盒子上的边框都去掉。这里我们使用一个 no-border 的类来控制网格的边框：

```
/* 不需要边框时取消容器上的border */
.tt-grid.no-border{
    border: none;
}
/* 不需要边框时取消格子上所以的border */
.tt-grid.no-border > .tt-grid-item{
    border: none;
}
```

有了这两个样式，只要在 .tt-grid 上添加上 no-border 的类，整个九宫格里的边框就会被干掉了，效果如下：

这样整个九宫格的开发就完成了。

三、多列网格的设计与实现

在实现多列网格的时候，只需要在刚才的九宫格上进行改动就可以。这里我们要变动的地方有：

1. 改变折行的位置。假如要制作两列网格的时候，就可以把格子宽度设置成 50%，这样每行就正好能放下 2 个格

子

2. 改变边框的情况。假如要制作两列网格的时候，就要把默认情况下所有第 3n 个格子的右边框恢复，然后再把所有 2n个格子的右边框去掉。
3. 改变 .tt-grid-item 的内边距，来调整整个格子的高度。

根据这些要求，我们就可以写出多列网格的样式了。2 列网格就可以用如下代码来实现：

```
/* 两列网格 */
.tt-grid.tt-grid-2 .tt-grid-item{
    border-right: 1px solid #ddd;
    flex-basis: 50%;
    padding: 1.8rem 0;
}
.tt-grid.tt-grid-2 .tt-grid-item:nth-child(2n){
    border-right: none;
}
```

使用这个样式就可以实现 2 列网格的样式了，效果如下：

同理，如果再实现剩下列数的网格，下面是 4 列网格和 5 列网格的样式：

```
/* 四列网格 */
.tt-grid.tt-grid-4 .tt-grid-item{
    border-right: 1px solid #ddd;
    flex-basis: 25%;
    padding: .9rem 0;
}
.tt-grid.tt-grid-4 .tt-grid-item:nth-child(4n){
    border-right: none;
}
/* 五列网格 */
.tt-grid.tt-grid-5 .tt-grid-item{
    border-right: 1px solid #ddd;
    flex-basis: 20%;
    padding: .6rem 0;
}
.tt-grid.tt-grid-5 .tt-grid-item:nth-child(5n){
    border-right: none;
}
```

这里 4 列网格我们把格子宽度设置成 25%，5 列网格的格子宽度就是 20%。在移动端通常就到 5 列，我们也就实现到这么多列的。如果有特殊需求需要更多列的网格，到时按着这个原理再添加就可以了。这两种网格的效果如下。

四列网格样式：

五列网格样式：

到这里我们对多列网格的开发也完成了。但经过测试我们会发现一个小问题，当给其他列数的网格添加 no-border 的时候，会有如下的现象：

这是因为“.tt-grid.no-border > .tt-grid-item”和“.tt-grid.tt-grid-4 .tt-grid-item”这两个选择器的优先级相同了，而 no-border 样式的语句在前面，无法覆盖后面 .tt-grid-4 里面 border 的样式。为了解决这个问题，我们可以在，no-border 的样式里加上“!important”来解决这个问题。所以前面写的“.tt-grid.no-border > .tt-grid-item”的样式要修改为：

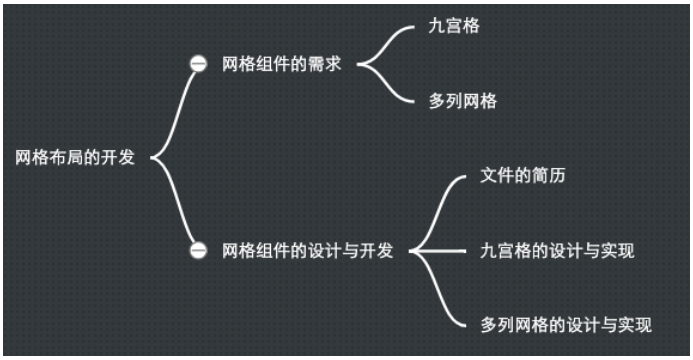
```
/* 不需要边框时取消格子上所以的border */
.tt-grid.no-border > .tt-grid-item{
    border: none !important;
}
```

这样再测试多列网格的样式就变得正常了：

结语

到这里网格的开发就完成了，我们这一节考察的内容就是 flex-wrap 的用法，还有对“:nth-child()”这个伪类选择器的应用。这一节我们使用 flex-wrap 来实现网格会很方便，但是这种实现方式也有一个问题，就是当格子的个数不是列数的整数倍时，最后一行的格子就会被抻拉的和前面不对齐。如果有非整行排列的需求的话，就要精确的指定每个格子的宽度，然后再取消格子的拉伸属性。

我们这一节的内容结构如下：



我们这一节的内容就到这里，同学们可以访问【Grid网格组件在线预览】来查看这一节的演示效果，下一节将进行菜单组件的开发。

24 List列表样式的设计与开发
26 Menu菜单样式的设计与开发