26 Menu菜单样式的设计与开发

更新时间：2019-07-26 14:49:43

> 耐心和恒心总会得到报酬的。
>
> ——爱因斯坦

我们这一节的任务是来设计和实现一个菜单组件的样式。在移动端里，受到屏幕大小的限制，通常不会使用太复杂的菜单样式，不像 PC 端里有二级菜单，甚至三级菜单。移动端的菜单基本就是只有一级，如果有二级菜单也会再做一个二级菜单的页面来呈现。我们这一节要实现的菜单样式如下：

这个菜单中前四个菜单项都是比较基础的菜单样式，和简单列表很相似。后面两行就是带有开关样式的菜单项，这一块我们要重点讲这个开关的实现方法。下来我们先来聊一下对菜单组件的要求。

Menu菜单组件的需求

在 PC 端中，因为页面空间充足，菜单和操作可以处在同一个页面中，所以 PC 端的菜单通常只是充当页面或者功能的引导。但是在移动端，有些复杂的操作会被放进菜单项引导的二级页面来做，为了方便查看操作结果，通常会把操作信息也展示在菜单上。而对于简单的操作，也可以直接放在菜单上来完成，比如上面的开关功能。所以移动端的菜单除了基本的引导作用以外，还要承担信息展示和完成简单操作的作用。下面我们来分析一下移动端的菜单组件都会有什么需求：

1. 菜单项左右留出空间，不能紧贴边框。
2. 菜单名称占满一行的剩余空间，超长以后自动截断，不能折行。
3. 菜单信息部分可以是文本或者图标，要有最大宽度限制，超宽后自动截断，不能折行。
4. 需要进入下级页面的菜单项要有引导图标，引导图标位于菜单项的最右侧。
5. 有开关功能的菜单项，开关按钮位于菜单项的最右侧，且有开和关两种状态。
6. 开关的状态切换时，要有过渡效果。

有了这些需求，就可以进行菜单的设计和开发工作了。

Menu菜单组件的设计与开发

一、文件的建立

还是老规矩，我们先建立文件，首先是空的 /demo/menu.html 文件：

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="initial-scale=1, width=device-width, maximum-scale=1, user-scalable=no">
    <link rel="stylesheet" href="../src/tuitui-ui.css">
    <title>推推UI-菜单组件</title>
  </head>
  <body>
    <div class="tt-content">
      <h1 class="tt-panel-title">菜单组件</h1>
      <div class="tt-panel-body no-padding">
        <div class="tt-menu">
          <!-- 菜单组件内容 -->

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

这个文件里我们也只建立了一个 tt-menu 的容器。

然后是 CSS 文件，新建一个 /src/menu.css：

```
/*
 * @Author: Rosen
 * @Date: 2019-07-24 20:15:43
 * @Last Modified by: Rosen
 * @Last Modified time: 2019-07-25 16:05:41
 */

/* 菜单组件 */
```

这是一个空的 CSS 文件，里面只放了头部文件。

最后要把这个刚建好的 CSS 文件添加到 /src/tuitui-ui.css 里，在 /src/tuitui-ui.css 文件的最后追加上：

```
/* 菜单组件的样式 */
@import './menu.css';
```

这样就很快的把文件建出来了。

二、基础菜单的实现

根据前面的需求，基础菜单里的分为菜单名称、菜单信息、菜单引导图标这三个内容。我们先按着这个要求把基本的 html 结构做出来：

```
<div class="tt-menu">
    <a class="tt-menu-item">
        <p class="tt-menu-name">用户ID</p>
        <span class="tt-menu-value">33581893</span>
    </a>
    <a class="tt-menu-item">
        <p class="tt-menu-name">用户名</p>
        <span class="tt-menu-value">推推UI</span>
        <i class="fa fa-chevron-right tt-menu-icon"></i>
    </a>
    <a class="tt-menu-item">
        <p class="tt-menu-name">二维码</p>
        <i class="fa fa-qrcode tt-menu-value"></i>
        <i class="fa fa-chevron-right tt-menu-icon"></i>
    </a>
    <a class="tt-menu-item">
        <p class="tt-menu-name">隐私设置</p>
        <i class="fa fa-chevron-right tt-menu-icon"></i>
    </a>
</div>
```

这里我们做了四个菜单，分别展示了菜单的不同用法；

1. 第一个菜单项只展示信息，没有引导作用，所以只有菜单名称和菜单信息两个内容。
2. 第二个和第三个菜单项有展示带信息和引导到下级页面的作用，所以会有菜单名称、菜单信息和引导图标这三个内容。只不过第二个菜单项使用了文本，而第三个菜单项使用了一个二维码图标作为菜单信息。
3. 最后一个菜单项需要引导到下一级的菜单，没有具体的菜单信息，所以只有菜单名称和引导图标。

根据这几种菜单的类型我们可以知道，菜单名称是一定有的，后面的菜单信息和引导图标都不是必须的。根据需求，我们要求菜单名称靠左，菜单信息和菜单引导图标在最右侧，且都和页面的边界有一定的空间。这种需求一看就应该很熟悉了，与前面完成的列表和输入框的样式又很相似，直接用弹性布局解决。
首先是菜单项容器的样式：

```
/* 菜单项 */
.tt-menu > .tt-menu-item{
    display: flex;
    height: 2.3rem;
    padding: 0 1rem;
    align-items: center;
    border-bottom: 1px solid #eee;
    font-size: .8rem;
}
```

我们用“display: flex;”指定每个菜单项都是弹性盒子，然后通过高度和 align-items 属性限定容器里面的内容在竖直方向上居中，再然后是用 padding 属性给菜单项两边加上了边距，最后就是边框、字体这些基础样式的设置。

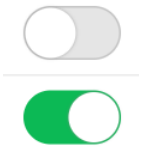
在下面来指定菜单项里三种元素的样式：

```
/* 菜单名称 */
.tt-menu > .tt-menu-item > .tt-menu-name{
    flex: 1;
    overflow: hidden;
    white-space: nowrap;
    text-overflow: ellipsis;
    color: #333;
}
/* 菜单信息 */
.tt-menu > .tt-menu-item > .tt-menu-value{
    max-width: 5rem;
    height: 1rem;
    line-height: 1rem;
    overflow: hidden;
    white-space: nowrap;
    text-overflow: ellipsis;
    color: #999;
}
/* 下级操作引导图标 */
.tt-menu > .tt-menu-item > .tt-menu-icon{
    margin-left: .3rem;
    color: #999;
}
```

在这三个元素中，菜单名称使用了“flex:1;”用来占满剩余空间，以便把其他元素挤到容器的最右侧，如果菜单名称过长的话会自动截断。菜单信息这个元素没有指定宽度，但指定了最大宽度和超出自动截断，这样就不会在信息太长的时候挤占菜单名称的位置。最后的引导图标就是简单的设置了一个左边距，用来和菜单信息隔开一段距离。这样基础菜单的样式就实现了，最终实现的效果就是下面这样：

三、开关的实现

这一节中比较有难度的，就是菜单中开关的实现方式了。根据前面的效果图，我们看到开关的样式如下：



我们如果实现这样两个样式的话，最直接的想法应该就是用两个元素分别充当开关的背景和开关的圆形按钮，再通过用不同的 class 来控制开关不同状态下的样式，甚至可以和菜单组件中实现单选和多选功能一样，在开关里再隐藏一个 input 元素来记录开关的状态值。刚才说的这种方式是完全可行的，但是我们这里要使用一种比较粗暴的方式，我们仅使用一个 input 元素来实现这个开关，HTML 的结构如下：

```
<div class="tt-menu">
    <a class="tt-menu-item">
        <p class="tt-menu-name">新消息提醒</p>
        <input class="tt-switch" type="checkbox">
    </a>
    <a class="tt-menu-item">
        <p class="tt-menu-name">仅在有WIFI网络下使用</p>
        <input class="tt-switch" type="checkbox" checked>
    </a>
</div>
```

当我们没有给 input 元素样式的时候，这两个菜单项的样式就像下面这样：

默认的 input 元素就是个选择框，我们如果想把它改造成一个开关的样式，那么就要去掉它默认的样式。这里可以使用下面这个样式来取消 webkit 内核中选择框的默认样式：

```
/* 去掉webkit内核里默认的样式 */
-webkit-appearance: none;
/* 去掉webkit内核里默认的点击效果 */
-webkit-tap-highlight-color: rgba(0, 0, 0, 0);
```

去掉默认样式以后，这个 input 元素就变成了一个标准的盒子了。并且可以通过这个 input 是不是 checked 状态来区分盒子的样式，可以用下面的代码来实现：

```
/* 切换开关 */
.tt-menu > .tt-menu-item > .tt-switch{
    position: relative;
    box-sizing: content-box;
    width: 2.6rem;
    height: 1.4rem;
    border: 1px solid #ccc;
    outline: 0;
    border-radius: .75rem;
    background-color: rgba(0, 0, 0, 0.1);
    /* 去掉webkit内核里默认的样式 */
    -webkit-appearance: none;
    /* 去掉webkit内核里默认的点击效果 */
    -webkit-tap-highlight-color: rgba(0, 0, 0, 0);
}
/* 选中状态下，改变开关边框和背景色 */
.tt-menu > .tt-menu-item > .tt-switch:checked{
    border-color: #07C160;
    background-color: #07C160;
}
```

在这段代码里，我们把 .tt-switch 这个 input 元素当成了容器来用。虽然这种用法不太常规，但 input 确实是可以这样使用的。这里还要注意下 checkbox 类型的 input 元素在有些浏览器里默认会是怪异盒模型，所以我们这里把它指定成了标准盒模型，以便一会计算圆形按钮的大小和位置。其他的样式，就是指定这个容器的样式了。最后我们又使用 .tt-switch:checked 选择器来区分按钮的背景和边框。这段代码最终的效果就有些接近我们的目标了：

接下来就要实现开关里的按钮了。我们的目的是用一个 input 元素就搞定整个开关的样式，但现在 input 元素已经被用作了开关容器。这时候就要想一下在第二章选择器部分讲过的一个东西，那就是伪元素选择器。我们是可以用 ::before 或者 ::after 来伪造出一个元素的，所以开关的那个小圆圈用伪元素选择器来做就可以实现我们的目标了。

我们只要给 .tt-switch::after 元素加上“content: " ";”属性，就可以把它当成一般的盒子来用了。我们可以用以下代码来制定这个 ::after 元素的样式：

```
/* 使用::after伪元素实现圆形按钮的样式 */
.tt-menu > .tt-menu-item > .tt-switch::after{
    content: " ";
    position: absolute;
    top: 0;
    left: 0;
    width: 1.4rem;
    height: 1.4rem;
    border-radius: .7rem;
    background-color: #FFFFFF;
    box-shadow: 0 0 2px #999;
}
/* 选中状态下，更改圆形按钮的位置 */
.tt-menu > .tt-menu-item > .tt-switch:checked::after{
    left: 1.2rem;
}
```

这段代码中要注意的是，我们使用 box-shadow 属性来替代 border 来区分元素的边界，是为了让边界有一种立体的效果。这样在计算宽度和高度的时候也变得简单了，不用考虑 border 带来的影响。

> **@ Tips：**
> box-shadow 属性是 CSS3 中规定的属性，是用来给盒子的边框添加阴影效果的。这个属性的语法是“box-shadow: px1 px2 px3 color;”，这个属性中几个值的作用如下：
>
> - px1 的值用来指定阴影的水平位移，可以是负值。
> - px2 用来指定阴影的竖直位移，也可以是负值。
> - px3 用来指定阴影的模糊半径，也就是阴影的模糊程度。
> - color 就是用来指定阴影的颜色。
>
> 在使用这个属性的时候有两点要注意：
>
> 1. box-shadow 只是盒模型的一种显示效果，不会影响盒子的尺寸和位置。
> 2. 这个属性的消耗比较大，不建议大批量的使用。

这个圆形按钮默认的情况下在开关的左侧，当选中以后位置就移到了开关的右侧。通过这样设置后，开关的样式就出来了：

我们现在可以点击这个开关，通过控制 checkbox 的选择情况就能直接控制开关的样式，不需要再手动的在元素上加什么 class。但是点击的时候就会发现，这个开关的效果非常生硬，所以我们需要给它加一些过渡效果。首先我们先来介绍下 CSS3 中的 transition 属性。

> **@ Tips：**
> transition 属性也是 CSS3 中的属性，是用来指定元素的过渡效果的。比如某元素从样式1变成了样式2，如果没有过渡就是直接改变；如果使用了过渡效果的话，浏览器会计算这两个状态之间的所有状态，然后显示出变化过程。 transition 属性其实是四个属性的缩写，它们依次是：
>
> - **transition-property**，这个属性用来指定对哪个样式属性添加过渡效果。这个属性可以指定的通常是有数值的属性，比如 height、width、top、bottom、left、right、color等。如果希望对所有属性都添加过渡效果，那这个属性还可以使用“all”这个属性值，但不建议这么用。
> - **transition-duration**，这个属性用来指定完成过渡效果的时间。这个属性值的形式是“数字+s”，比如 .5s 就是 0.5 秒完成过渡效果。
> - **transition-timing-function**，这个属性用来指定过渡效果的变化速度的，它的取值可以是 linear、ease、ease-in、ease-out 和 ease-in-out 这几个具体的名称，也可以使用“cubic-bezier(n,n,n,n)”这种三阶贝塞尔函数。这里几种具体名称的取值实际上就是对几种固定参数的贝塞尔函数做了个别名，如果对贝塞尔函数感兴趣的话，可以自己去了解一下。
> - **transition-delay**，这个属性用来指定过渡效果的延时时间的，有了这个属性就可以允许过渡效果过一段时间后再开始。这个属性的取值也是一个时间格式的值，同 transition-duration 用法相同。
>
> 在使用 transition 属性的时候，我们要知道这也是一个高消耗的属性，不要大面积使用。

了解了 transition 属性以后，再做过渡动画就容易了，我们只需要给容器和按钮加上过渡效果，这个按钮的过渡效果就完成了。容器里我们针对背景色和边框颜色来添加，而按钮中只需要对 left 属性添加过渡，所以我们会对刚才写过的代码做改动，分别给 .tt-switch 元素和 .tt-switch::after 伪元素加上 transition 属性：

```
/* 切换开关 */
.tt-menu > .tt-menu-item > .tt-switch{
    position: relative;
    box-sizing: content-box;
    width: 2.6rem;
    height: 1.4rem;
    border: 1px solid #ccc;
    outline: 0;
    border-radius: .75rem;
    background-color: rgba(0, 0, 0, 0.1);
+   transition: background-color .3s, border .3s;
    /* 去掉webkit内核里默认的样式 */
    -webkit-appearance: none;
    /* 去掉webkit内核里默认的点击效果 */
    -webkit-tap-highlight-color: rgba(0, 0, 0, 0);
}
```

```
/* 使用::after伪元素实现圆形按钮的样式 */
.tt-menu > .tt-menu-item > .tt-switch::after{
    content: " ";
    position: absolute;
    top: 0;
    left: 0;
    width: 1.4rem;
    height: 1.4rem;
    border-radius: .7rem;
    background-color: #FFFFFF;
    box-shadow: 0 0 2px #999;
+   transition: left .3s;
}
```

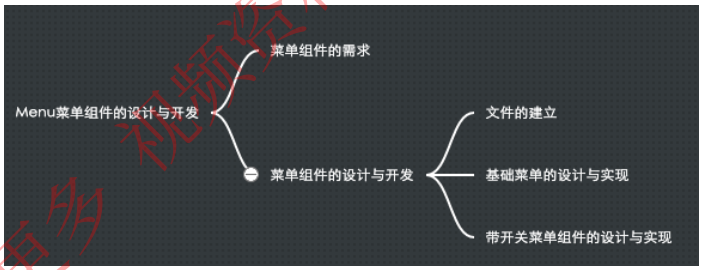
这里我们没有给 transition 指定速度曲线，用的是默认值ease。曾经调整过很多次这个贝塞尔曲线，但最后还是觉得默认值是最干爽的。这样我们这个开关的切换效果也完成了，最终的效果如下：

总结

到这里我们整个菜单组件就实现完成了。这一节有下面几个比较重要的知识点：

1. 如何去除元素的默认样式？
2. 伪元素选择器 ::after 的实际用法。
3. box-shadow 属性的用法，以及需要注意的地方。
4. 使用 transition 属性来实现过渡效果的方法。

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【菜单组件在线预览】来查看这一节的演示效果，下一节将进行模态框组件的开发。

25 Grid网格样式的设计与开发
27 Modal模态框样式的设计与开发