28 Loading加载提示样式的设计与开发

更新时间：2019-07-31 16:20:16

> 衡量一个人的真正品格，是看他在知道没人看见的时候干些什么。
>
> ——孟德斯鸠

我们这一节来实现移动端常用的加载提示组件。移动端中需要用到加载提示工具的地方一般就有三种场景，我们分别来介绍一下。

第一种场景是整个页面的加载时，这时候需要一个加载中或者加载失败的提示信息。这种场景下整个页面都是空白的，所以需要一个大一些的提示信息来填充空白的页面，就可以使用下面这种大的加载提示组件：

第二种场景是列表在拉到头的时候，会去加载下一页的内容，这个时候需要在列表的尾部添加一个加载的状态，也就是下面这类提示组件：

第三种需要加载提示的组件就是在做某些重要操作的时候，比如订单的确认、支付的操作等。由于这类操作耗时会比较长，所以需要给出一个正在操作的提示信息，就会像下面这种 Toast 形式的信息提示组件：

我们这一节要实现的加载提示组件会包含前两种，而第三种 Toast 形式的提示组件我们放到下一节，作为 Toast 提示组件的内容来完成。

加载提示组件的需求

接下来，我们聊一下对这一节要完成的两种加载提示组件都有什么要求。

一、页面加载提示工具

加载提示组件并不是只有加载中的状态下才用的到，在页面加载出现异常情况下，也可以使用这个组件进行提示。页面如果加载成功了就会显示页面的内容，因此不需要有加载成功的信息的提示。

对于页面加载提示工具，我们有如下要求：

1. 提示工具分为图标和文本两部分，两种内容水平居中。
2. 如果是载入中的提示，图标使用旋转的loading图标。
3. 如果是其他类型的提示，使用静态的图标即可。

4. 图标和文字的颜色要浅一些，不会抢占用户的视线。

二、列表加载提示工具

对于列表加载提示工具，也是会有几种状态。最常见的就是加载中的状态。此外，还有当拉到页面底部，可能会用到提示操作的上拉加载更多的提示，还有在没有更多数据的时候显示已经到底的信息等。

对于列表中使用的单行加载提示工具，会有以下要求：

1. 提示工具中的文本水平居中。
2. 文本的左右有两个线条，作为提示文本的装饰。
3. 如果是正在加载的提示，左侧添加旋转的加载图标。
4. 其他情况下，提示信息只包含纯文本。
5. 如果是提示上拉加载更多的提示，在右侧添加上下振动的上箭头图标作为引导。

有了需求，下面就进入设计和开发的阶段。

加载提示的组件的设计与开发

一、文件的建立

老规矩，第一步我们先把这节需要的文件建立出来。首先是HTML文件/demo/loading.html:

```
<!DOCTYPE html>
<html lang="en">
  <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-Loading加载提示组件</title>
  </head>
  <body>
    <div class="tt-content">
      <!-- 页面加载提示组件 -->
      <h1 class="tt-panel-title">页面加载提示组件</h1>
      <div class="tt-panel-body">
        <!-- ... -->
      </div>
      <!-- 单行加载提示组件 -->
      <h1 class="tt-panel-title">单行加载提示组件</h1>
      <div class="tt-panel-body">
        <!-- ... -->
      </div>
    </div>
  </body>
</html>
```

然后是对应的CSS文件/src/loading.css：

```
/*
 * @Author: Rosen
 * @Date: 2019-07-29 21:21:29
 * @Last Modified by: Rosen
 * @Last Modified time: 2019-07-30 15:46:29
 */

/* 页面加载提示组件 */
.tt-loading{

}
```

最后把刚建好的CSS文件加入到主文件/src/tuitui-ui.css中：

```
/* 加载提示组件 */
@import './loading.css';
```

这样所有需要的文件就建好了。

二、页面加载提示工具的设计与开发

我们先来开发页面使用的加载提示工具。根据需求，这个组件会很容易实现，其实就是一个图标和一个说明文字。我们先来完成HTML的部分：

```
<!-- 页面加载提示组件 -->
<h1 class="tt-panel-title">页面加载提示组件</h1>
<div class="tt-panel-body">
    <div class="tt-loading">
        <i class="fa fa-circle-o-notch fa-spin tt-loading-icon"></i>
        <span class="tt-loading-info">用力载入中...</span>
    </div>
    <div class="tt-loading">
        <i class="fa fa-refresh tt-loading-icon"></i>
        <span class="tt-loading-info">加载出错，点我重新加载</span>
    </div>
</div>
```

从 HTML 结构来看，这个组件确实很简单，主要的结构就是一个容器套住一个图标和一行文本。这里注意前面的需求中要求如果是加载中的状态，需要图标进行旋转。我们这里直接在图标元素上加了“fa-spin”这个 class，就能达到图标旋转的效果。

下来就是实现这三个元素的样式了，在 /src/loading.css 添加如下样式：

```
/* 页面加载提示组件 */
.tt-loading{
    padding: 1rem 0;
    text-align: center;
}
/* 页面加载提示组件的图标 */
.tt-loading > .tt-loading-icon{
    font-size: 4.5rem;
    color: rgba(0, 0, 0, .05);
}
/* 页面加载提示组件的提示信息 */
.tt-loading > .tt-loading-info{
    display: block;
    margin-top: .6rem;
    font-size: .8rem;
    color: #ccc;
}
```

这里就是很基础的样式，唯一值得注意的是在使用图标颜色的时候用了 rgba 的形式，这是因为通过 rgba 设置透明度的方式实现的淡化效果比纯灰色看起来会更舒服。这样页面加载提示组件就完成了。

三、列表加载提示工具

单行的列表加载提示工具的结构也很简单，也是由文本和图标组成。这里面唯一要注意的就是文本两侧两条横线的实现方式。最直接的做法就是做两个盒子分别放在文本的左右两侧，但是还有更简单的方法，就是直接使用当前容器的边框来实现，再使用文本的背景色把实线中间的部分遮盖住。

我们先来定义这种类型提示工具的HTML结构：

```
<div class="tt-loading-inline">
    <span class="tt-loading-info">
        <i class="fa fa-circle-o-notch fa-spin tt-loading-icon"></i>
        用力加载中
    </span>
</div>
```

这个结构和我们刚才用的有所区别，外层容器用了一个新的类名 tt-loading-inlie，这是因为这个加载组件的样式和刚才的页面加载工具结构不太一样，所以直接用了一个新域名，而没有采用样式覆盖的方式。另外一个不同点是这个组件的图标 tt-loading-icon 被包在了 tt-loading-info 容器里，而不是和页面加载提示工具里那样放在了同一级。

对于这个结构，我们就可以采用下面的样式来完成它的要求：

```
/* 单行加载提示组件 */
.tt-loading-inline{
    margin: 1.5rem auto 1rem;
    width: 12.5rem;
    position: relative;
    box-sizing: border-box;
    text-align: center;
    color: #999;
    height: 1rem;
    border-top: 1px solid rgba(0, 0, 0, .1);
}
/* 单行加载提示组件的文本信息 */
.tt-loading-inline > .tt-loading-info{
    display: inline-block;
    padding: 0 .5rem;
    position: relative;
    top: -.7rem;
    height: 1rem;
    line-height: 1rem;
    font-size: .7rem;
    background: #fff;
}
/* 单行加载提示组件的图标 */
.tt-loading-inline > .tt-loading-info > .tt-loading-icon{
    color: rgba(0, 0, 0, .2);
}
```

这里面唯一需要注意的就是文本两边横线的实现方式，我们对 tt-loading-inline 容器加了个上边框，用它来充当那条横实线。然后再把内容使用“position: relative; top: -.7rem;”这两个属性向上移动了 .7rem 的距离，这样容器的边框就正好在文本的中间了。最后给文本加上白色背景和左右的内边距，就把背后的横线遮住了一段，看起来就是下图这样的效果了：

根据需求，在提示信息是上拉加载更多时，需要一个一直在上下振动的上箭头。这个箭头也是图标库里有现成的，我们只需要让它冻起来就可以了。在实现效果之前，我们先来介绍一下 CSS3 中制作动画用的 animation 属性。

> **@Tips：**
> CSS3 中定义的 animation 属性，可以允许用户自己定义一些简单的动画效果。animation 的语法如下：
>
> ```
> animation: name duration timing-function delay iteration-count direction;
> ```

animation 里有很多个属性，如果同学们仔细观察的话，动画里的参数和 transition 属性很相似。实际上，我们就可以把 animation 理解成更复杂的渐变效果。下面我们先来介绍下这几个参数：

animation-name（动画的名称），这个参数是用来指定使用哪个效果的动画，这里的名称是需要我们使用 @keyframes 定义出来的。我们先把 @keyframes 放一边，等介绍完 animation 的属性再来介绍这个东西。

animation-duration（动画持续的时间），这个就和渐变属性 transition 里的 duration 一样了，就是一轮动画持续的时间。

animation-timing-function（时间函数），定义了动画的播放速度曲线。

animation-delay（延迟时间），定义了动画播放的延迟时间，表示多长时间以后再开始播放。

animation-iteration-count（循环次数），这个参数式之前没见过的，它定义了动画执行几遍。这个值可以取正整数n,表示动画会播放n遍后停止。如果需要动画一直重复播放的话，把这个参数值设置成“infinite”就可以了。

animation-direction（动画播放方向），这个参数定义了动画播放的模式，它的取值可以是“normal”、“reverse”、“alternate”和“alternate-reverse”这四个值，默认的“normal”表示正常从前到后的播放；“reverse”表示动画从后往前播放；“alternate”表示先从前往后播放，然后再从后往前到播放；“alternate-reverse”表示的就是先从后往前播放，再从前往后播放。

这些就是关于 animation 的属性的介绍下来我们回到刚才的 @keyframes 的用法。如果有细心的同学翻看过 Font Awesome 的源码，看过 fa-spin 类的实现方法的话，就应该看过 @keyframes 的用法：

```
.fa-spin {
    -webkit-animation: fa-spin 2s infinite linear;
    animation: fa-spin 2s infinite linear
}
```

```
@keyframes fa-spin {
    0% {
        -webkit-transform: rotate(0deg);
        transform: rotate(0deg)
    }

    100% {
        -webkit-transform: rotate(359deg);
        transform: rotate(359deg)
    }
}
```

Font Awesome里实现图标旋转的效果，其实就是使用了这两部分代码。前面一部分是定义 fa-spin 的元素样式，就是加了一个动画效果。后面这段代码，就是定义了一个名为 fa-spin 的动画。要主意这个动画的名称和 .fa-spin 选择器没什么关系，只不过用了相同的名字。这个名为 fa-spin 的动画定义了在一个动画周期中，开始位置是旋转 0 度，动画最后旋转到 359 度，看起来就是转了一圈。然后在 animation属性中使用“infinite”值指定了动画一直循环播放，所以我们就看见了一直旋转的加载图标。

这就是 @keyframes 的主要用法，只需要在里面指定几个关键节点，浏览器就会计算出对应的动画效果。这里起始位置是 0 %，当然还可以加很多个中间状态，最后动画结束的位置就是 100% 时指定的样式。

如果像刚才截图的那样，动画简单到只有起始和结束状态，那么还可以把“0%”替换成“from”，把“100%”替换成“to”，像下面这样：

```
@keyframes fa-spin {
    from {
        -webkit-transform: rotate(0deg);
        transform: rotate(0deg)
    }

    to {
        -webkit-transform: rotate(359deg);
        transform: rotate(359deg)
    }
}
```

上面这种写法和之前那种百分比的写法效果是完全一样的，只有在有多个中间状态时，百分比形式的写法才有意义。

了解了 animation 这个属性后，就可以直接在图标的文件中加入需求中要求的动画效果了。我们把这个动画效果归到图标的样式里，就可以在 /src/icon.css 中加入下面的代码：

```
/* 垂直方向上振动 */
.fa-vibrate-y{
  animation: fa-vibrate-y 1.5s infinite ease-in;
}
/* 振动轨迹 */
@keyframes fa-vibrate-y{
  0% {
    transform: translateY(-10%);
  }
  50% {
    transform: translateY(10%);
  }
  100% {
    transform: translateY(-10%);
  }
}
```

这里我们给图标定义了一个 fa-vibrate-y 的类，它和用作图标旋转的 fa-spin 类是同一类作用，只不过使用的是上下振动的动画效果。这个动画就是在图标高度的正负 10% 的位置不停的上下振动。有了这个类，我们再回到加载提示组件的 HMTL 文件中，在 tt-panel-body 中加入如下结构：

```
<div class="tt-loading-inline">
  <span class="tt-loading-info">
    上拉加载更多
    <i class="fa fa-long-arrow-up fa-vibrate-y tt-loading-icon"></i>
  </span>
</div>
```

通过给 tt-loading-icon 加上“fa-vibrate-y”这个 class，就达到了需要的动画效果：

最后还剩下两种静态的信息提示就可以直接用了，不再需要额外的样式：

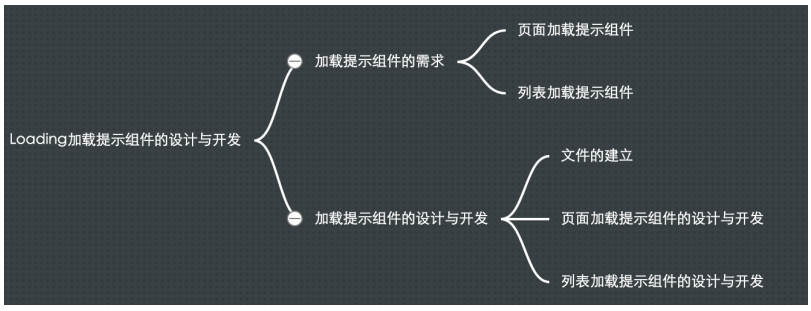
```
<div class="tt-loading-inline">
    <span class="tt-loading-info">
        我可是有底线的
    </span>
</div>
<div class="tt-loading-inline">
    <span class="tt-loading-info">
        .
    </span>
</div>
```

把这四种单行加载组件放在一起，就是下面这样最终的效果了：

这样，这两种加载提示组件的设计和开发就完成了。

结语

这一节我们讲了两种加载提示组件的设计和开发，主要用到的技巧就是实现文本两侧添加水平实线的方式，还有使用 animation 属性实现动画效果的用法。我们这一节的结构如下：



我们这一节的内容到这，同学们可以访问【加载提示组件在线预览】来查看这一节的演示效果，下一节将进行加载组件的开发。

27 Modal模态框样式的设计与开发
29 Toast提示工具的样式与开发

精选留言 0

欢迎在这里发表留言，作者筛选后可公开显示

目前暂无任何讨论