23 Search搜索框样式的设计与开发

更新时间：2019-07-26 10:13:25

> 才能一旦让懒惰支配，它就一无可为。
>
> ——克雷洛夫

在页面中，搜索是使用频率很高的一个组件，我们这一节就来分析一下对搜索组件有哪些要求，然后再按着提出的需求实现出一个搜索组件。

搜索组件的需求

搜索框的功能比较简单，最基本的就是输入和提交两个逻辑。但是我们这个搜索框要额外的加一些细节，输入框分为输入状态和非输入状态，两种状态下输入框表现要有所不同。我们对这个搜索框组件会有以下要求：

1. 在搜索框输入后，可以直接点击键盘上的搜索按钮进行查询。这样整个搜索工作都可以在屏幕底部的键盘中完成，不用再去页面顶部点击提交按钮，可以减小操作范围。
2. 在输入状态中，当输入框有内容时，提供一键清空的功能。
3. 在输入状态中，要有输入提示，且整个搜索区域要遮盖住整个内容区，这样在搜索的时候不会受到内容部分的影响。
4. 在输入状态中，提供取消按钮，在非输入状态下隐藏取消按钮。

按着上面的要求我们将要实现下面这样一个输入框的样式：

在非输入状态下，只有一个输入框样式，在输入框中有一个搜索图标和“搜索”两个提示用的文本。

当在输入的状态时，显示取消按钮和搜索提示，并且在输入框不为空的时候提供一键清空的按钮。为了实现上面的效果，同学们可以先思考下要怎么做，然后再看下面的内容。

搜索组件的设计与开发

一、文件的建立

最开始先把这一个节需要的文件建立出来，和之前的组件类似，需要一个演示用的 html 文件和对应的样式文件。首先建立 /demo/search.html 文件：

```
<div class="tt-content">
    <!-- 带suggest的搜索框 -->
    <div class="tt-search">
        <!-- 搜索框 -->
        <form class="tt-search-form" action="#">
            <div class="tt-search-input-wrap">
                <i class="fa fa-search tt-search-icon"></i>
                <input type="text" class="tt-search-input" placeholder="搜索" autocomplete="off" required/>
                <i class="fa fa-close tt-search-clear"></i>
            </div>
            <span class="tt-search-cancel">取消</span>
        </form>
        <!-- 搜索建议 -->
        <ul class="tt-search-suggest">
            <li class="tt-suggest-item">手机</li>
            <li class="tt-suggest-item">iPhone XS Max</li>
            <li class="tt-suggest-item">华为P30</li>
            <li class="tt-suggest-item">小米 MIX3</li>
            <li class="tt-suggest-item">诺基亚1110</li>
        </ul>
    </div>
    <p class="content">内容区</p>
</div>
```

然后建立空的 /src/search.css 样式文件：

```
/*
 * @Author: Rosen
 * @Date: 2019-07-12 16:22:12
 * @Last Modified by: Rosen
 * @Last Modified time: 2019-07-17 16:40:59
 */
/* 搜索框 */
.tt-search{
    ...
}
```

最后在主文件把 /src/search.css 文件引入，在 /src/tuitui-ui.css 的最后追加：

```
/* 搜索栏的样式 */
@import './search.css';
```

由于篇幅关系，HTML 文件内容只放了 body 的部分，同学们在建立文件的时候，要把 head 部分按着以前的结构补充上。这个 html 结构里 .tt-search 是整个搜索框的容器，里面包含了 .tt-search-form 这个搜索表单和搜索建议用的容器 .tt-search-suggest，.tt-search 后面跟的就是内容区，可以放任意的内容。

现在我们按着这个结构，来分析下刚才的需求都应该怎么实现。

二、搜索组件容器的设计与开发

搜索组件的容器一共有两个状态，分为输入状态和非输入状态。我们使用“on-focus”这个类来区分搜索框是不是在输入状态。在非输入状态下，搜索组件就是文档流里一个普通的盒子，可以随着页面进行滚动，而在输入状态下搜索组件需要覆盖住整个页面，所以我们要给 .tt-search 如下的样式：

```
/* 搜索框 */
.tt-search{
    max-width: 640px;
    margin: 0 auto;
    background: #f8f8f8;
}
```

因为这个组件的容器可能会有使用固定定位的情况，所以和之前实现标题栏一样，我们给它限制住最大宽度，并且使用“margin: 0 auto;”让它水平居中。

在输入状态下，希望 .tt-search 容器遮盖住整个内容区（ .tt-content 覆盖的区域）。这里我们可以使用固定定位来处理输入状态下的遮盖问题，和我们之前讲全屏遮罩时用到的方法是一样的：

```
/* 搜索状态中，覆盖内容区 */
.tt-search.on-focus{
    position: fixed;
    left: 0;
    right: 0;
    top: 0;
    bottom: 0;
    overflow-y: auto;
}
/* 处理有标题栏的情况 */
.tt-header ~ .tt-content .tt-search.on-focus{
    top: 2.3rem;
}
/* 处理有导航栏的情况 */
.tt-navbar ~ .tt-content .tt-search.on-focus{
    bottom: 2.5rem;
}
```

上面的代码就会使搜索容器覆盖住全屏，并且通过兄弟选择器处理了有标题栏和导航栏的情况。这样整个输入框的容器就处理好了。

三、搜索框的设计与开发

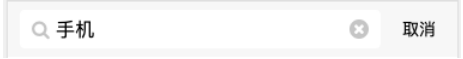
搜索框里，我们使用了 form 做了最外层的容器，这是因为之前提到想在键盘上直接提交搜索操作，如果想达到这种效果，就必须使用下面这种结构：

```
<form action="#">
    <input type="text" />
</form>
```

也就是带有 action 属性的表单里包含的 input 元素，这样的结构就会被输入法认定为表单的功能，会在键盘上显示出“前往”的按钮，如下：

> **@ Tips:**
> 如果 input 的类型是 H5 中的搜索类型，也就是 <input type=“search” />，那么在有些输入法中表单的提交按钮的文本就可以变成“搜索”，但是这种用法会有问题，同学们下去自己尝试一下，并试着解决一下遇到的问题。

下来我们回到样式的开发上，我们要实现下面这样的样式：



首先来做 form 这个容器，我们给这个容器固定 2.3rem 的高度，给它设置“display: flex;”。此外，这个容器里的各个元素准备放在竖直居中的位置，我们就可以用弹性布局里的“align-items”属性来搞定。这个 form 容器最终的样式就是：

```
/* 搜索栏中的表单 */
.tt-search > .tt-search-form{
    display: flex;
    height: 2.3rem;
    align-items:center;
}
```

> **@ Tips:**
> align-items 属性就是用来标记弹性布局里的元素在侧轴（也就是和 flex-direction 方向相垂直的轴）上的对齐方式，form 里所有元素都是水平排列的，所以想竖直居中，就可以直接使用“align-items:center;”来实现。
> align-items 有以下可取的值：
>
> 1. **stretch**（默认），元素进行抻拉来填满，如果盒子有指定大小，那么以指定的大小为准。
> 2. **center**，元素会布局在容器侧轴的中间位置。
> 3. **flex-start**，元素位于容器的开头。
> 4. **flex-end**，元素位于容器的结尾。
> 5. **baseline**，元素位于第一行文本的基准线位置，这里要注意是第一行的文本，而不是第一个元素。

接下来是 form 容器里的 4 个元素，因为搜索和一键清空的图标都是和 input 关联的，所以我们做了个 .tt-search-input-wrap 容器把这三个放了进去，最后的取消按钮和这个容器在同一层。我们先来处理 .tt-search-input-wrap 和取消按钮的关系：

```
/* 输入框的容器 */
.tt-search > .tt-search-form > .tt-search-input-wrap{
    flex: 1;
    position: relative;
    padding: 0 .5rem;
}
/* 取消按钮 */
.tt-search > .tt-search-form > .tt-search-cancel{
    flex: 0 0 2.2rem;
    padding-right: .5rem;
    text-align: center;
    font-size: .7rem;
    display: none;
}
/* 对取消按钮的控制 */
.tt-search.on-focus > .tt-search-form > .tt-search-cancel{
    display: block;
}
```

这里面有几个点要注意下：

1. 没有取消按钮的时候，.tt-search-input-wrap 容器是撑开全宽，然后使用左右的 padding 来保证里面的输入框可以在左右留下一点空闲。
2. 有取消按钮的时候，为了保证取消按钮在空闲区域是居中的，所以会给取消按钮一个“padding-right: .5rem;”来和 .tt-search-input-wrap 右边留下的空间对称，这样取消按钮只要是文本居中就可以了。
3. .tt-search-cancel 在默认情况下是隐藏的，只有在搜索状态中才会显示。
4. 取消按钮使用固定的 2.2rem 的宽度，不做拉伸和收缩。

这样，这一层容器就处理好了，最后就是要处理 .tt-search-input-wrap 里面的三个元素了。

接下来我们先来处理搜索图标，这个图标只需要在 .tt-search-input-wrap 里进行绝对定位就可以了，前面给 .tt-search-input-wrap 加过“position: relative;”属性，就是为了在这里使用。下面是搜索图标的样式：

```
/* 搜索图标 */
.tt-search .tt-search-icon{
    position: absolute;
    height: .8rem;
    line-height: .8rem;
    font-size: .7rem;
    left: 1rem;
    top: 0;
    bottom: 0;
    margin: auto;
    color: #ccc;
}
```

这个图标的样式我们已经用过很多次了，没有什么难度。这里有个小的注意点，就是因为选择器的层级有些长了，所以这里直接使用后代选择器来处理最里层元素的样式。下来是输入框的样式，这个输入框只需要撑满 .tt-search-input-wrap 容器就可以了：

```
/* 输入框的样式 */
.tt-search .tt-search-input{
    box-sizing: border-box;
    width: 100%;
    height: 1.6rem;
    border: none;
    font-size: .8rem;
    padding-left: 1.5rem;
    background: #fff;
    border-radius: .2rem;
}
```

这个输入框左侧需要有 1.5rem 的内边距，用来让出搜索图标的位置。然后因为它宽度是 100% 的情况下还需要有内边距，所以直接使用了“box-sizing: border-box;”来处理盒子的模型就可以了。

最后是清空按钮的样式，这个也是一个简单图标的应用，在默认情况下它是不可见的，只有在搜索状态下并且输入框有内容，才让它出现。我们使用如下的样式来处理这个一键清空的图标：

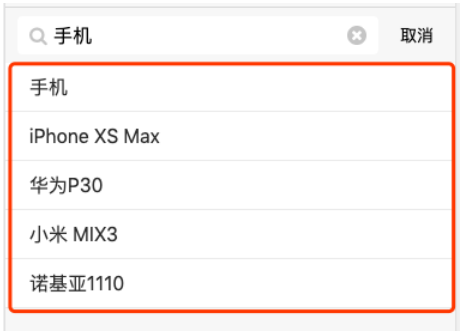
```
/* 清空按钮的样式 */
.tt-search .tt-search-clear{
    position: absolute;
    height: .8rem;
    line-height: .8rem;
    width: .8rem;
    font-size: .6rem;
    top: 0;
    bottom: 0;
    margin: auto 0;
    right: 1rem;
    border-radius: 50%;
    color: #fff;
    background: #ccc;
    display: none;
}
/* 对清空按钮的控制 */
.tt-search.on-focus .tt-search-input:valid + .tt-search-clear{
    display: block;
}
```

上面的代码中，图标的默认样式是没什么问题的，就是前面搜索图标的用法，只不过把位置放在了右边。这里面要注意的是第二条样式，我们刚才说过，只有在搜索状态下并且输入框有内容的时候才让这个清空按钮出现，所以使用了一个 :valid 选择器来判断和它相邻的输入框的状态。这个 :valid是和 input 中的“required”属性对应的，input 元素有内容时“required”验证条件就会通过，这时候 :valid 选择器就会选中这个 input，从而后面的兄弟选择器才会选中 .tt-search-clear 元素。这种用法就可以直接使用 CSS 来控制清空按钮的显示了，省去了 JS 的工作。

到这里我们整个输入框的样式就实现好了，最后我们实现一下搜索建议的部分。

四、搜索建议的设计与实现

最后，我们来实现一下搜索建议，也就是下面这部分的样式：



搜索建议部分只有在搜索状态中才会出现，并且按着默认的布局排列就好，所以我们只需要控制 .tt-search-suggest 这个容器的显示逻辑，代码如下：

```
/* 搜索建议 */
.tt-search > .tt-search-suggest{
    display: none;
}
.tt-search.on-focus > .tt-search-suggest{
    display: block;
}
```

我们通过 .tt-search 上是不是有 on-focus 类来确定搜索建议的内容要不要显示。再下面实现搜索建议里每一条的样式就可以了，这和我们表单元素里面用的样式很相似，就直接放代码了：

```
/* 搜索建议的内容 */
.tt-search > .tt-search-suggest > .tt-suggest-item{
    padding-left: 1rem;
    height: 2rem;
    line-height: 2rem;
    font-size: .8rem;
    background: #fff;
    border-bottom: 1px solid #eee;
    color: #333;
}
```

到这里我们整个搜索框的样式就实现完成了，同学们就可以通过在 .tt-search 上切换 on-focus 类来预览组件的两种状态了，在实际项目中会通过 JS 代码来控制状态的切换。但是经过一番测试，会发现在 iPhone 中会有问题，就像下面这样：

这个问题是因为 .tt-content 有“overflow-y: auto;”属性，而现在 .tt-content 的高度只有里面 p 标签撑开的那么高。在一般的浏览器里，“overflow: auto”属性对固定定位元素默认是 visible 的，但在 iPhone 的浏览器里，auto 对固定定位元素的表现是hidden的，所以就会遮住 .tt-search 一部分内容。

为了解决这个问题，我们只要把 .tt-content 内容区的高度撑开就可以了，所以我们要在 html、body 和 .tt-content 上面都加上“height:100%”样式即可。这里我们要在 /src/common.css 里的 html 和 body 选择器的样式里加上“height:100%”，然后再在 /src/content.css 里 .tt-content 选择器的样式里也加上“height:100%”，这样问题就解决了。

结语

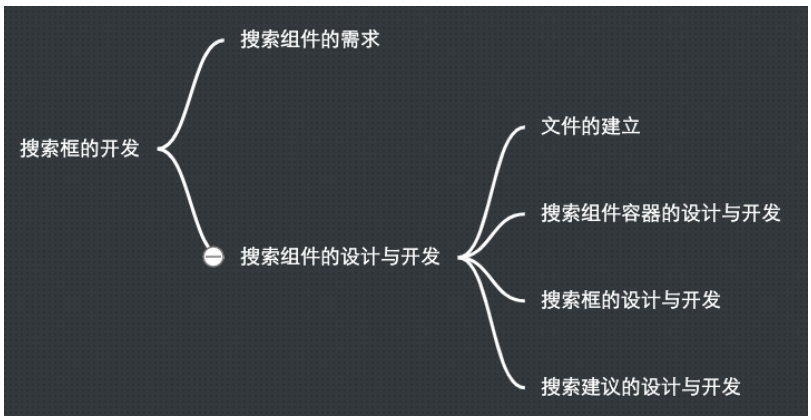
经过一番努力，我们完成了这个搜索栏的样式。这一节涉及到组件两种状态的切换方式，还有组件里各种元素的控制和配合的方式。在基础知识部分，要记住这几点：

1. 什么样的 html 结构可以让输入法中出现提交表单的按钮？
2. 弹性布局中，了解 align-items 属性的用法，以及几种可取的属性值。
3. 了解伪类选择器 input:valid 的用法。

下面给同学们留几个思考题：

1. 我们这个搜索栏在非输入状态下是随着页面内容滚动的，如果我们需要把它固定在页面的顶部不随页面滚动，应该怎么修改？
2. 搜索框里如果使用 type=“search” 类型的 input 元素，会遇到什么问题？怎么解决？
3. 刚解决 iPhone 兼容性问题的时候，如果不考虑兼容问题，使用 vh 单位来实现应该怎么做？是不是比百分比的方式要简单一点？

这两个思考题留给同学们自己去实现，最后附上这一节内容的结构：



我们这一节的内容就到这里，我们这一节的内容就到这里，同学们可以访问【Search搜索组件在线预览】来查看这一节的演示效果，下一节将进入列表组件的开发。

22 常用样式组件的设计与开发
24 List列表样式的设计与开发