布局练习（响应式布局）

> 响应式网站开发是必备技能之一。响应式有它的很好的优点，也有它一定的缺点。这就需要我们在开发的时候做出取舍。对于内容较少、主要为展示类网站，故采用响应式；对于内容多，管理类的网站采用分开开发的方式，不同设备采用不同的一套代码。

一、响应式与自适应

很多人其实把**响应式**和**自适应**两种网站当成一回事，但事实上这两种网站的布局方式是有一定的区别的。我们可以来看看这两种方法的概念以及分别对应解决的问题。

1、什么是响应式布局?

响应式布局就是一个网站能够兼容多个终端，**可以根据屏幕的大小自动调整页面的的展示方式以及布局**，我们不用为每一个终端做一个特定的版本。

响应式网站的几个标志：

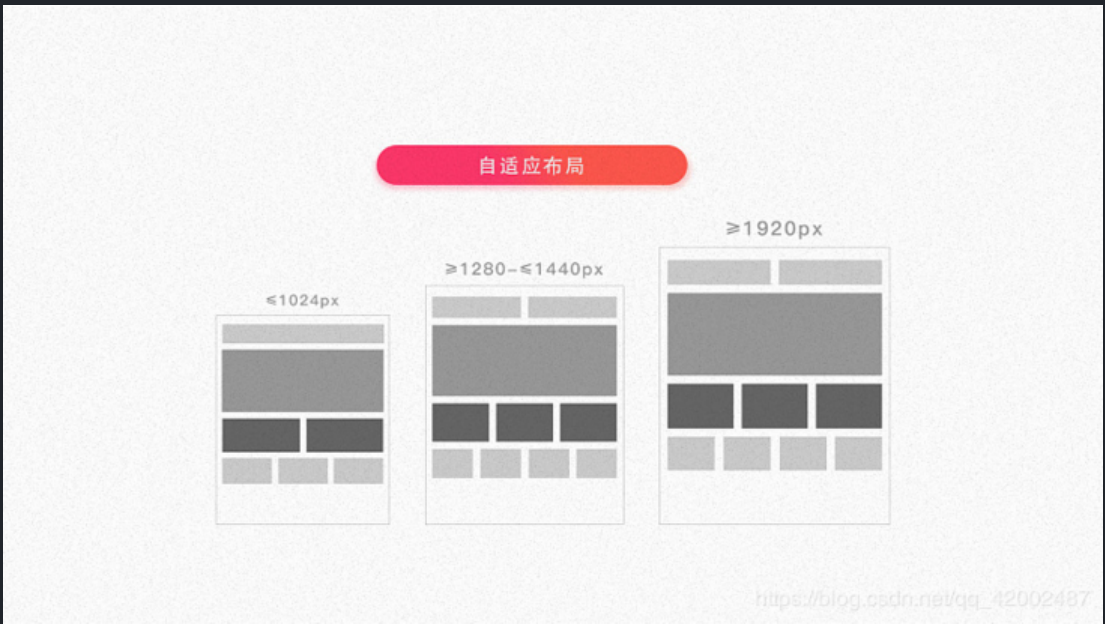
1. 同时适配PC + 平板 + 手机等；
2. 标签导航在接近手持终端设备时改变为经典的抽屉式导航；
3. 网站的布局会根据视口来调整模块的大小和位置；

2、什么是自适应布局?

自适应布局是指网页能够在不同大小的终端设备上自行适应显示，也就是让一个网站在不同大小的设备上显示同一样的页面，让同一个页面适应不同大小屏幕，**根据屏幕的大小，自动缩放**。

自适应布局的几个标志：

1. 大多只是适配单个终端的主流N个主流视口；
2. 当视口大小低于设置的最小视口时，界面会出现显示不全，并且出现横向滑动条；
3. 总体框架不变，横线布局的版块太多会有所减少。



3、如何选择

总的来说，这两种方式的原来是相似的，都是**先检测设备，根据不同的设备采用不同的CSS**。

那开发中我们该如何去选择？这就要结合响应式与自适应的优缺点来看。

响应式布局的优点：

- 灵活性强；
- 能够快捷解决多设备显示适用问题。

缺点：

- 效率较低，兼容各设备工作量大；
- 代码较为累赘，加载时间会加长；
- 在一定程度上改变了网站原有的布局结构。

自适应布局的优点：

- 对网站复杂程度兼容更大；
- 代码更高效；
- 测试和运营都相对容易和精准。

缺点：

- 同一个网站需要为不同的设备开发不同的页面，增加的开发的成本。

上面两种方法各有自己的优缺点，所以我们在开发的时候，要从实际的项目出发。

对于页面不是太复杂的情况下，我们可以利用响应式布局；

对于页面中信息较多，布局较为复杂的情况，我们可以采用自适应布局的方式。

二、响应式布局设计步骤

介绍完基本的概念，我们来看看响应式布局的基本步骤，主要分为下面几步：

1. 设置meta标签

```
<meta name="viewport" content="width=device-width,
initial-scale=1, maximum-scale=1, user-scalable=no">
```

2. 使用@Media查询来设置样式，这是响应式布局的核心

```
@media screen and (max-width: 1920px) { ... }
```

3. 设置布局分界点，即通过设置多种视图宽度样式来控制页面布局

```
@media screen and (max-width: 1920px) { ... }

@media screen and (max-width: 1700px) { ... }
```

其实还是很简单的，就是在适配的时候稍微有点点繁琐。

三、布局分界点

对于@Media查询的分界点，这个可以根据自己的项目来调整，设置合适自己项目的布局分界点。

在设置分界点时，要注意先后顺序，当**使用`max-width`数值大的在前面，数值小的在后面**；

当**使用`min-width`时，数值小的放前面，数值大的放后面**。

下面列出了在项目开发时常用设置的部分布局分界点：

```
@media screen and (max-width: 1920px) { ... }

@media screen and (max-width: 1700px) { ... }

@media screen and (max-width: 1600px) { ... }
```

```
@media screen and (max-width: 1440px) { ... }

@media screen and (max-width: 1280px) { ... }

@media screen and (min-width: 992px) and (max-width:
1200px) { ... }

@media screen and (min-width: 768px) and (max-width:
991px) { ... }

@media screen and (max-width: 767px) { ... }
```

那我们什么时候用`min-width`，什么时候用`max-width`呢?

通常来说，**如果是移动端优先，则用`min-width`；如果是PC端优先，则用`max-width`**。

四、布局单位

熟悉常用的布局单位还是很重要的，常用的布局单位包括像素（`px`），百分比（`%`），`em`，`rem`，`vw/vh`。

我们可以合理的运用这些布局解决方案，来提高我们开发时的效率和质量。

1、像素

像素是网页布局的基础，一个像素表示计算机屏幕所能显示的最小区域。像素分为两种类型：css像素和物理像素。两者区别如下：

> **css像素：为web开发者提供，在css中使用的一个抽象单位；**
> **物理像素：只与设备的硬件密度有关，任何设备的物理像素都是固定的。**

2、百分比

当浏览器的宽度或者高度发生变化时，通过百分比单位可以使得浏览器中的组件的宽和高随着浏览器的变化而变化，从而实现响应式的效果。一般我们的直观理解都会认为子元素的百分比完全相对于直接父元素，这种理解没问题，比如 `height` 和 `width` 属性。但是在css的盒模型中不止有 `height` 和 `width` 属性，还有 `padding` 、 `border` 、 `margin` 等属性，所以这就值得我们去具体分析一下。

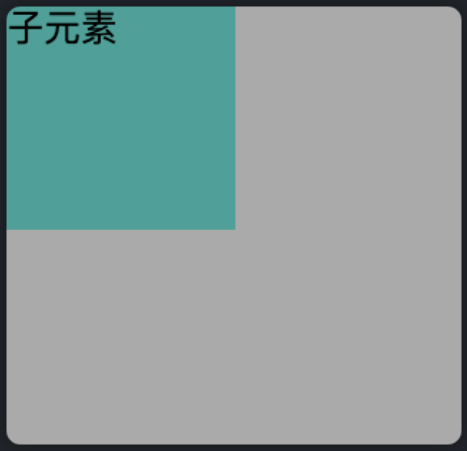
我们先写好html代码，然后通过不同的css代码来看看会呈现出什么样的情况：

```
<div class="parent">
    <div class="child">子元素</div>
</div>
```

1) 子元素的height和width

当子元素的 `height` 和 `width` 使用百分比时，是相对于直接父元素的 `height` 和 `width` 进行变化的。

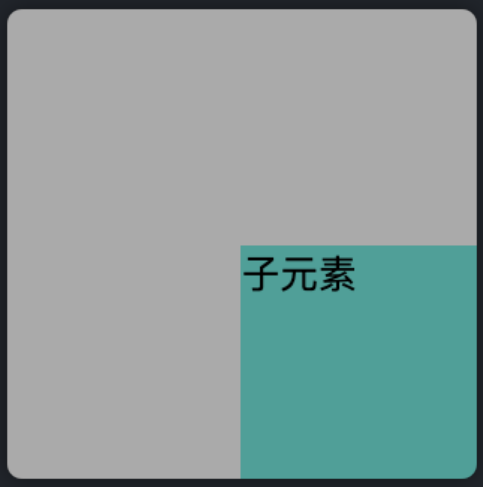
```
.parent{
    width: 200px;
    height: 200px;
    background: #aaa;
}
.child{
    width: 50%;
    height: 50%;
    background: #509F98;
}
```



2) 子元素的top、bottom 、left和right

子元素的 `top` 和 `bottom` 如果设置百分比，则相对于直接非static定位(默认定位)的父元素的高度；
子元素的 `left` 和 `right` 如果设置百分比，则相对于直接非static定位(默认定位的)父元素的宽度。

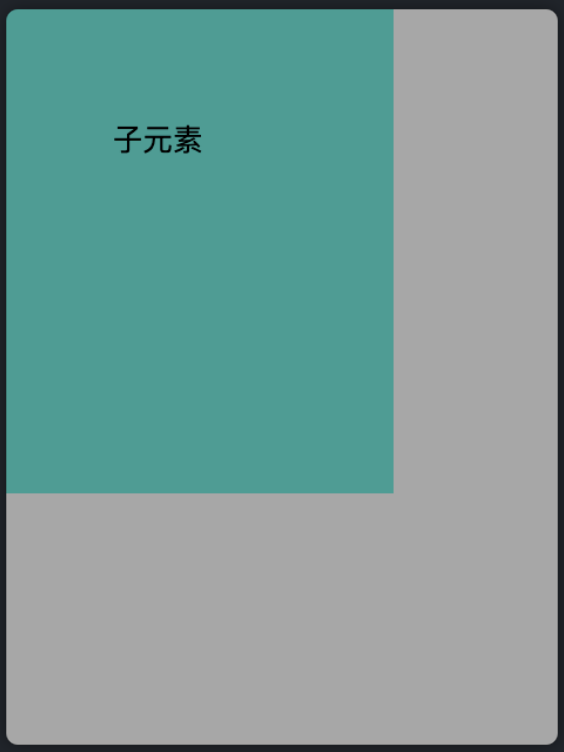
```
.parent{
    width: 200px;
    height: 200px;
    background: #aaa;
    position: relative;
}
.child{
    width: 50%;
    height: 50%;
    background: #509F98;
    position: absolute;
    top: 50%;
    left: 50%;
}
```



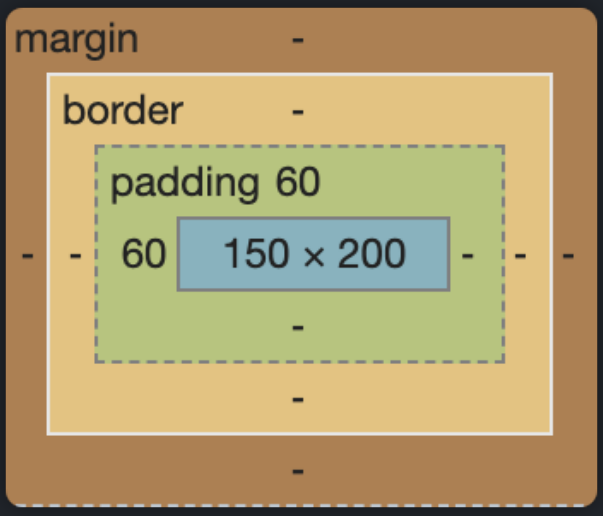
3) 子元素的padding

子元素的 `padding` 如果设置百分比，不论是垂直方向或者是水平方向，都相对于直接父亲元素的 `width` ，而与父元素的 `height` 无关。

```
.parent{
    width: 300px;
    height: 400px;
    background: #aaa;
}
.child{
    width: 50%;
    height: 50%;
    background: #509F98;
    padding-top: 20%;
    padding-left: 20%;
}
```



打开控制台，查看子元素，我们可以看见`padding-top`和`padding-left`都为父元素`width`的20% ——60px：



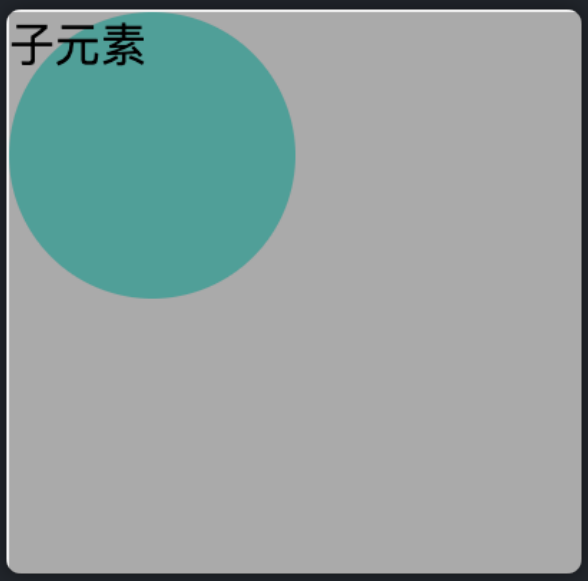
4) 子元素的margin

子元素的`margin`和`padding`是一样的，子元素的`margin`如果设置成百分比，不论是垂直方向还是水平方向，都相对于直接父元素的`width`。

5) 子元素的border-radius

`border-radius`设置为百分比则是相对于自身的宽度

```
.parent{
    width: 200px;
    height: 200px;
    background: #aaa;
}
.child{
    width: 50%;
    height: 50%;
    background: #509F98;
    border-radius: 50%;
}
```



3、em和rem

`em` 和 `rem` 相对于 `px` 更具灵活性，他们都是相对长度单位，而他们之间的区别可以用一句话来概括：**`em` 相对于父元素，`rem` 相对于根元素**。`em` 是相对于父元素的 `font-size`，`rem` 则是相对于html元素的 `font-size`。

4、vw/vh

`vw/vh` 是与视图窗口有关的单位，`vw` 表示相对于视图窗口的宽度，`vh` 表示相对于视图窗口高度，除了 `vw` 和 `vh` 外，还有 `vmin` 和 `vmax` 两个相关的单位。

单位	含义
vw	相对于视窗的宽度，视窗宽度是100vw
vh	相对于视窗的高度，视窗高度是100vh
vmin	vw和vh中的较小值
vmax	vw和vh中的较大值

这个单位和百分比很类似，但是还是有区别的：

单位	含义
%	大部分相对于祖先元素，也有相对于自身的情况比如（border-radius、translate等)
vw/vm	相对于视窗的尺寸

五、响应式布局实践

我们现在结合上面的理论知识，来完成一个响应式布局的demo。

1、初始布局

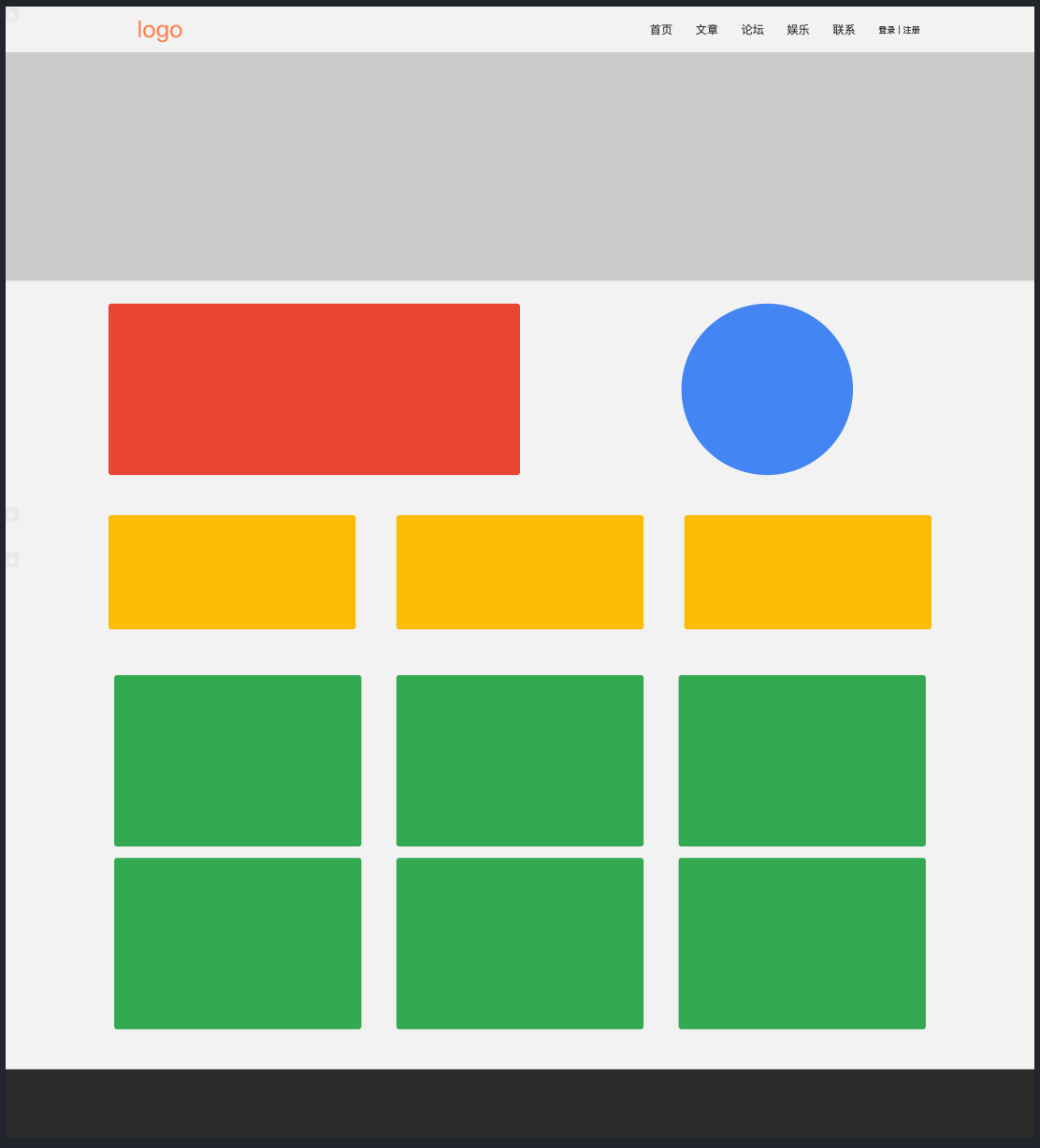
首先，我们进行整体的布局，这个demo主要分为三部分：头部、内容、底部，类似于简单的企业官网。

```
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge"
/>
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <title>响应式布局实践</title>
    <link rel="stylesheet" href="./css/index.css" />
  </head>
  <body>
    <div id="root">
      <!-- 头部 -->
      <header>
        <div class="header-box">
          <div class="logo">logo</div>
          <nav class="main-nav">
            <ul class="nav-list">
              <li id="nav-item" class="main-nav-item">
                <ul class="nav-main-list">
                  <li class="list-item">首页</li>
                  <li class="list-item">文章</li>
```

```
            <li class="list-item">论坛</li>
            <li class="list-item">娱乐</li>
            <li class="list-item">联系</li>
          </ul>
        </li>
        <li class="main-nav-login">
          <div class="login-box">登录 | 注册</div>
        </li>
      </ul>
    </nav>
  </div>
</header>

<!-- 内容 -->
<div id="container">
  <div class="container-header"></div>
  <div class="content">
    <!-- 内容一-->
    <div class="content-box content-box1">
      <div class="content-box1-text"></div>
      <div class="content-box1-img">
        <div class="img-radius"></div>
      </div>
    </div>
    <!-- 内容二-->
    <div class="content-box content-box2">
      <div class="content-box2-text"></div>
      <div class="content-box2-text"></div>
      <div class="content-box2-text"></div>
    </div>
    <!-- 内容三-->
    <div class="content-box content-box3">
      <div class="content-box3-text"></div>
      <div class="content-box3-text"></div>
      <div class="content-box3-text"></div>
      <div class="content-box3-text"></div>
      <div class="content-box3-text"></div>
      <div class="content-box3-text"></div>
    </div>
  </div>
</div>
```

```
        <!-- 底部 -->
        <footer></footer>
      </div>
    </body>
</html>
```

在这个demo中，最初的css代码是在`1920px`下进行编写的，这个就是我们的初始样式。
初始样式代码比较长，就不贴在文章里面了， 看打卡附件
我们先来看看初始的效果：



在实际开发的时候，有些模块中包含的内容，使得这些模块不能无限制的进行缩小。

所以在index.css的初始布局中，我们对一些模块的大小进行了限制，设置了`min-width`。

2、中大屏适配

前面我们提到，布局分界点是要根据实际的项目来进行划分的，针对这个项目我们进行以下的划分。

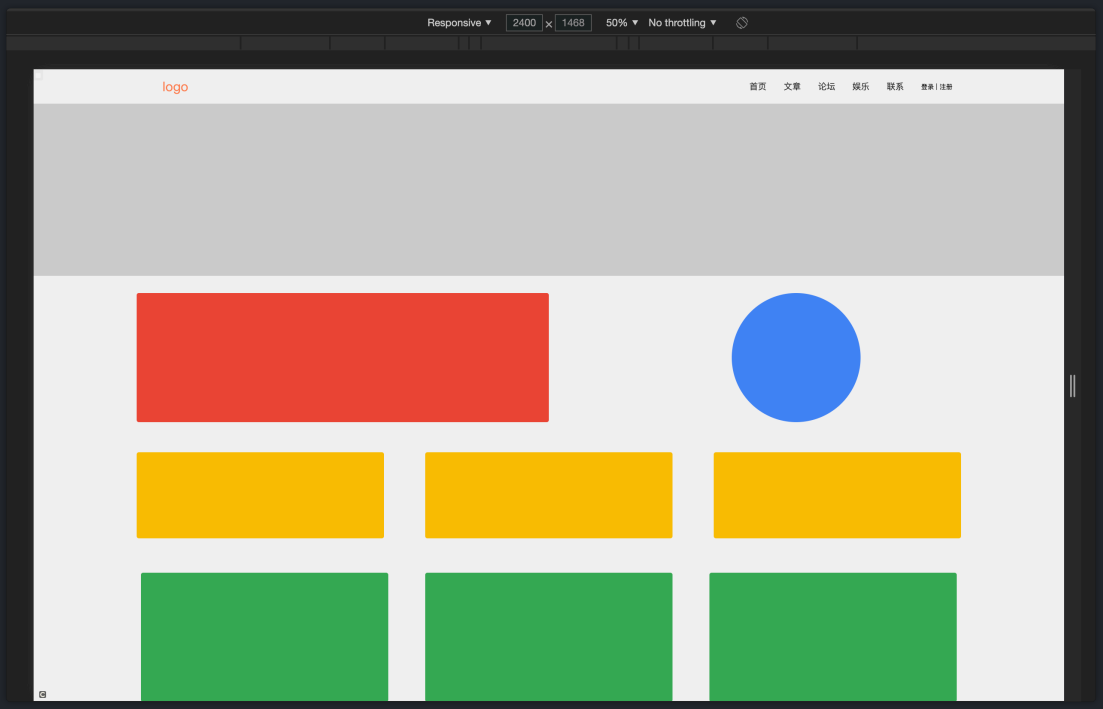
可以在chrome调试界面改变尺寸，看效果

```
/* 响应式布局 */
/* 中大屏样式 */
@media screen and (max-width: 1280px) {
  /* 内容一  */
  .content-box1 {
    flex-wrap: wrap;
    height: 650px;
  }
  .content-box1-text {
    width: 90%;
    margin: 0 auto;
  }
  .content-box1-img {
    display: flex;
    align-items: center;
    margin: 0 auto;
  }
}

@media screen and (min-width: 992px) and (max-width:
1200px) {
  /* 内容二 */
  .content-box2 {
    height: 400px;
  }
  .content-box2-text {
    height: 300px;
    min-width: 200px;
```

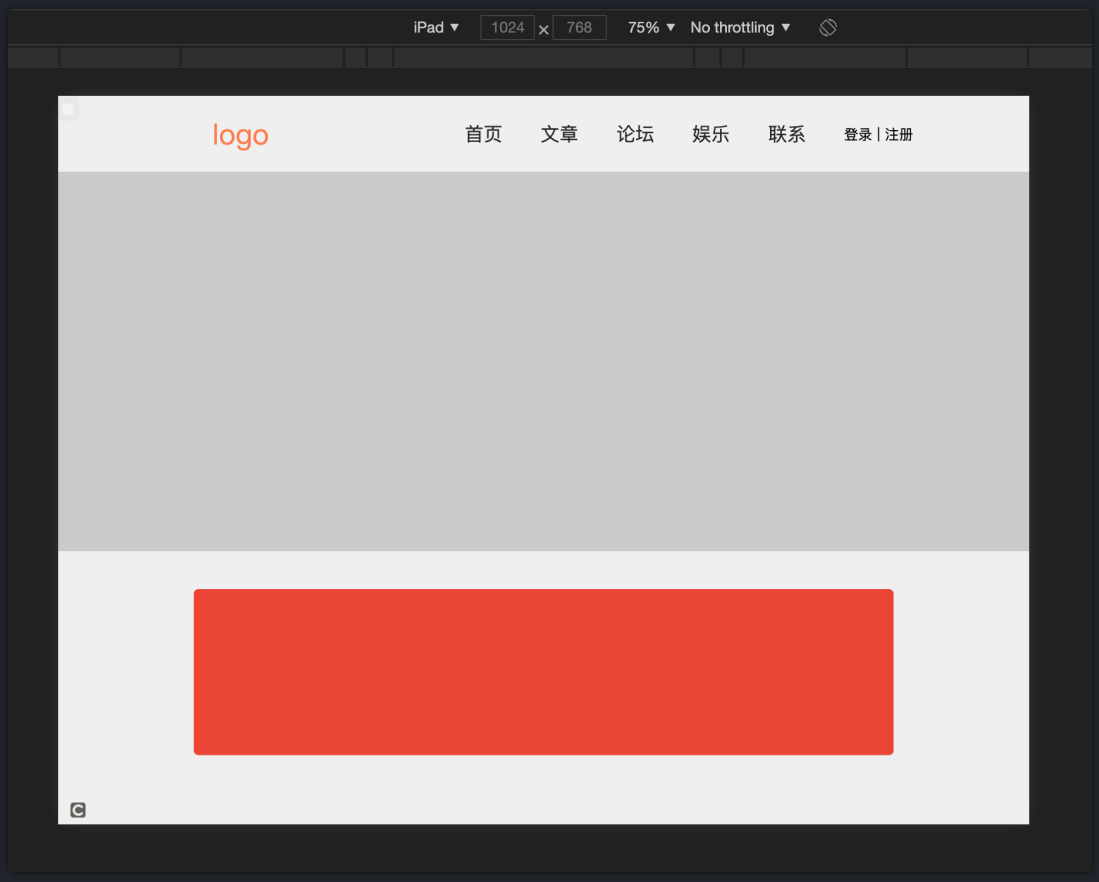
```
  }

  /* 内容三 */
  .content-box3 {
    justify-content: space-around;
  }
}

@media screen and (min-width: 768px) and (max-width:
991px) {
  .header-box {
    width: 90%;
  }
  .logo {
    min-width: 160px;
    margin-right: 2rem;
  }
  .login-box {
    padding: 0 2rem;
  }
  .content {
    width: 90%;
  }
  .list-item {
    padding: 0 1.5rem;
  }
  /* 内容二 */
  .content-box2 {
    height: 900px;
    flex-wrap: wrap;
    width: 100%;
  }
  .content-box2-text {
    width: 90%;
    margin: 0 auto;
  }
  /* 内容三 */
  .content-box3 {
    justify-content: space-around;
  }
}
```

```css
@media screen and (max-width: 767px) {
  .header-box {
    width: 90%;
    position: relative;
  }
  .logo {
    min-width: 160px;
    margin-right: 2rem;
  }
  .login-box {
    padding: 0 2rem;
    font-size: 1.5rem;
  }
  .content {
    width: 90%;
  }
  .list-item {
    padding: 0 1.5rem;
  }
  /* 内容一 */
  .content-box1-text {
    min-width: 90%;
  }
  /* 内容二 */
  .content-box2 {
    height: 900px;
    flex-wrap: wrap;
    width: 100%;
  }
  .content-box2-text {
    width: 90%;
    margin: 0 auto;
  }
  /* 内容三 */
  .content-box3 {
    justify-content: space-around;
  }
}
```

PC宽屏效果


Responsive
2400
1468
50%
No throttling
logo
首页
文章
论坛
娱乐
联系
登录 | 注册

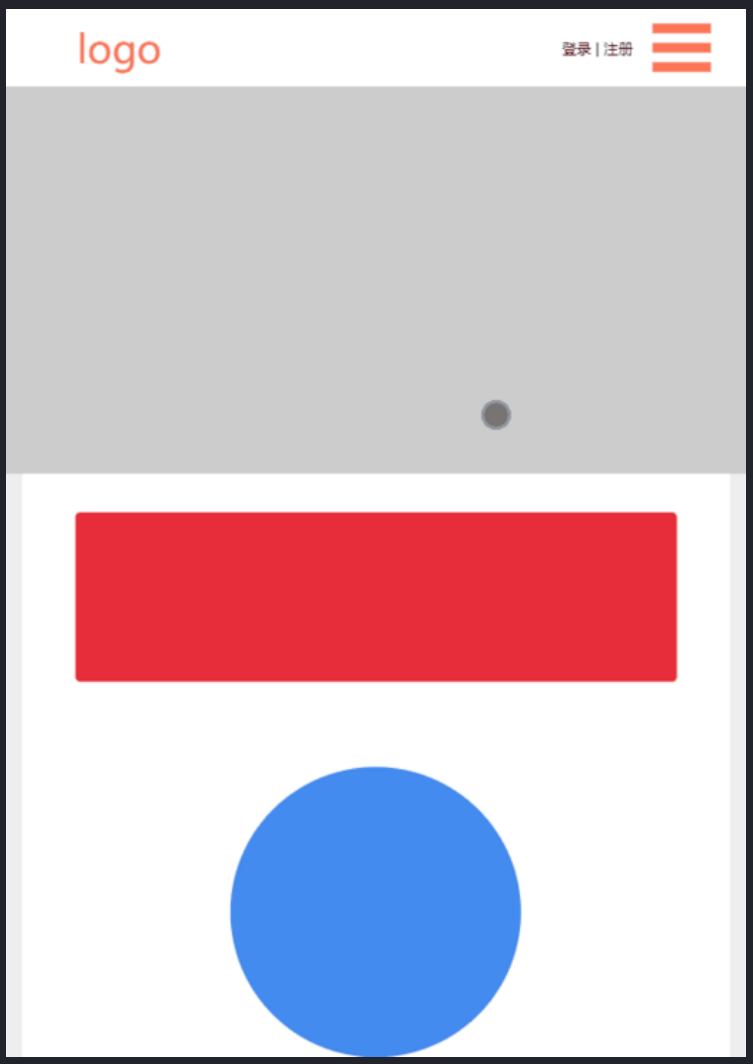
iPad横屏效果：


iPad
1024
768
75%
No throttling
logo
首页
文章
论坛
娱乐
联系
登录 | 注册

3、手机端适配

在boostrap的栅格系统中，把`小于768px`的屏幕归为手机。我们也可以以此作为参考。

此时布局调整不仅仅是内容的调整，我们的导航栏也要发生改变——**由横向导航栏变为点击出现的纵向导航栏**，如下图所示：

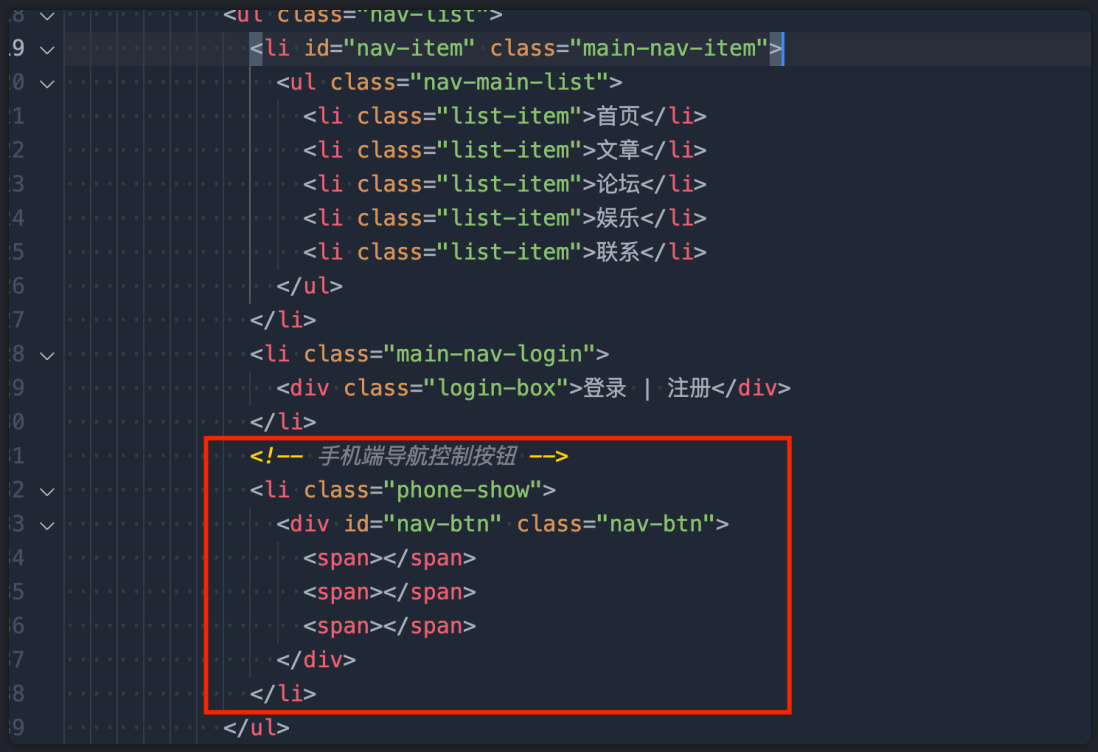


要想实现上图中的效果我们需要做哪些事情呢?

1. 写一个按钮，如右图右上角的按钮，样子是不是很熟悉；
2. 导航条样式重写，由横向导航条变为纵向导航条；
3. 利用js实现点击按钮，导航条显示与隐藏的切换。

1) 按钮

我们先在htm中给类名为 `nav-list` 的末尾加入以下HTML代码：



```
<li class="phone-show">
    <div id="nav-btn" class="nav-btn">
        <span></span>
        <span></span>
        <span></span>
    </div>
</li>
```

然后对按钮样式进行美化，在样式中，加入以下代码：

- 监听屏幕尺寸变化，来使用手机端样式
- 导航样式修改，我们可以巧妙的运用一个相对定位和绝对定位的关系实现导航条样式和位置的修改：

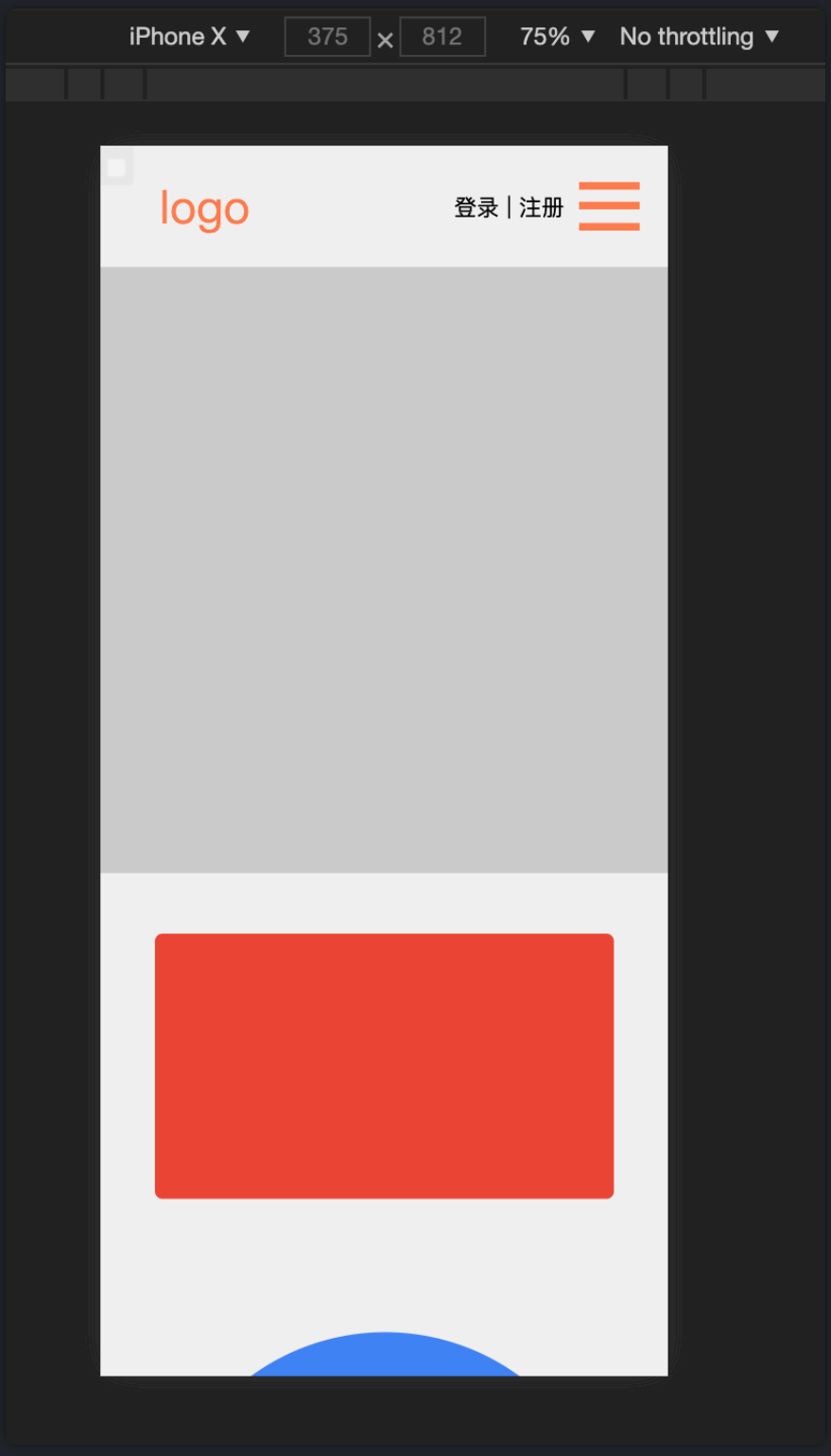
```
/* 响应式布局 */
/* 手机端样式 */
@media screen and (max-width: 767px) {
  .header-box {
    width: 90%;
    position: relative;
  }
```

```css
.logo {
  min-width: 160px;
  margin-right: 2rem;
}
.login-box {
  padding: 0 2rem;
  font-size: 1.5rem;
}
.content {
  width: 90%;
}
.list-item {
  padding: 0 1.5rem;
}
/* 内容一 */
.content-box1-text {
  min-width: 90%;
}
/* 内容二 */
.content-box2 {
  height: 900px;
  flex-wrap: wrap;
  width: 100%;
}
.content-box2-text {
  width: 90%;
  margin: 0 auto;
}
/* 内容三 */
.content-box3 {
  justify-content: space-around;
}

.main-nav-item {
  display: none;
  position: absolute;
  width: 10rem;
  right: 0;
  top: 8rem;
  border-radius: 0 0 0.5rem 0.5rem;
}
.nav-main-list {
```

```
  width: 100%;
  background: #ffffff;
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  border-radius: 0 0 0.5rem 0.5rem;
}
.list-item {
  width: 100%;
  margin: 0 1rem;
  border-bottom: 1px solid #eeeeee;
}
.phone-show {
  display: block;
  height: 60px;
  width: 60px;
}
.nav-btn {
  height: 100%;
  width: 100%;
  display: flex;
  flex-direction: column;
  flex-wrap: wrap;
  justify-content: space-around;
}
.nav-btn span {
  display: inline-block;
  height: 10px;
  width: 60px;
  background: coral;
}

@media screen and (max-width: 414px) {
  .content-box1-img {
    min-width: 100%;
  }
}
@media screen and (max-width: 375px) {
  .logo {
    min-width: 100px;
    min-height: 40px;
    margin-right: 1rem;
```

```
      font-size: 3rem;
    }
    .login-box {
      padding: 0 1rem;
    }
    .phone-show {
      height: 40px;
      width: 40px;
    }
    .nav-btn span {
      width: 40px;
      height: 5px;
    }
    .list-item {
      font-size: 1.5rem;
    }
    .content {
      padding: 4rem 0;
    }
    .content-box2-text {
      min-width: 100%;
    }
  }
}
```

看看效果：



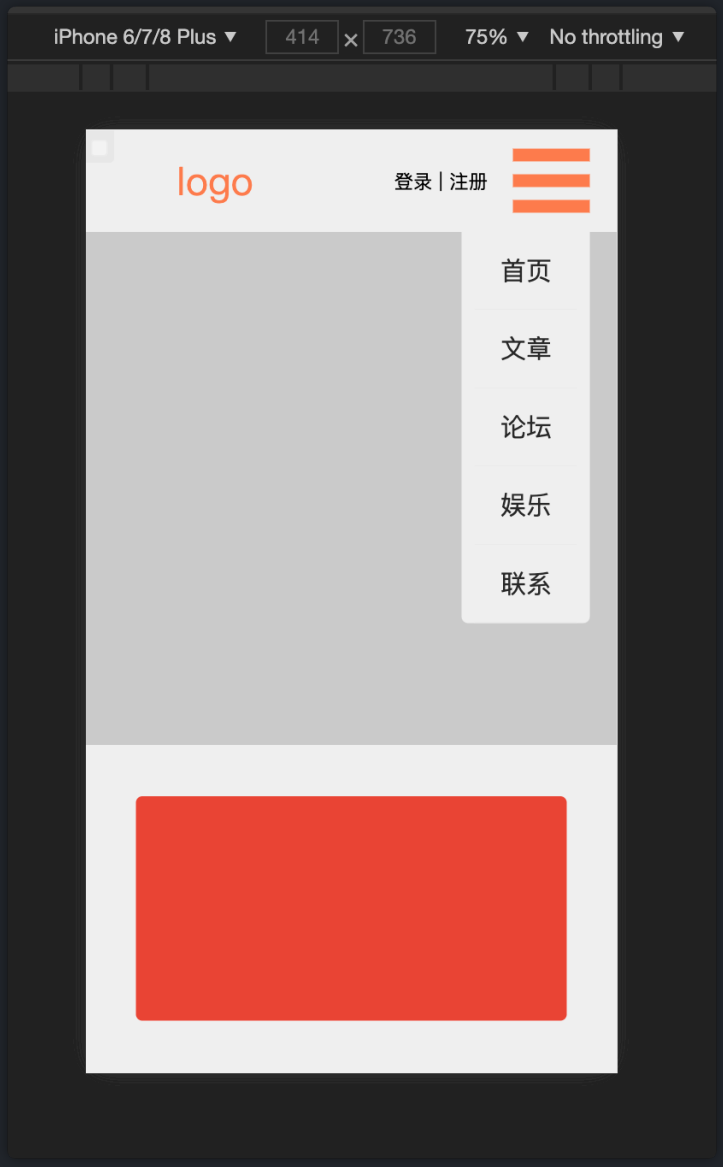
2) 导航条显示和隐藏切换

利用js来控制切换的思路很简单，我们在网页打开时，先把导航隐藏掉，然后定义一个变量 `hide`，初始值为 `true`，点击按钮时进行判断：如果 `hide` 值为 `true`，则让导航显示；反之隐藏。

在页面底部加入script脚本代码：

```
    <script>
        let btn = document.getElementById('nav-btn')
        let nav = document.getElementById('nav-item')
        let hide = true
        btn.addEventListener('click', function () {
            if (hide) {
                nav.style.display = 'block'
                hide = false
            } else {
                nav.style.display = 'none'
                hide = true
            }
        })

    </script>
```

实现效果，点击右上角的按钮可以实现导航的隐现，在平板和PC宽度下，该按钮不出现



最后再进行一些细节方面的调整，我们的响应式布局demo就初步完成了。

在日常的开发的时候，也要根据实际情况去进行调整，以满足实际的需求。