响应式布局
（Responsive Web Design）

概述

在网页设计的早期，页面是针对固定宽度屏幕尺寸构建的。随着终端设备的多样化发展，不同尺寸的屏幕越来越多，响应式布局的概念出现了。响应式布局是指网页布局不是固定的，而是根据不同终端设备或者不同浏览器窗口大小而呈现出不同的布局。响应式布局不仅可应用于整个页面，也可以应用于页面的一部分。

传统的开发方式是PC端开发一套页面，手机端再开发一套页面。但是这样做非常麻烦，随着不同的终端越来越多，你需要开发多个不同版本的页面。采用响应式设计时，网站仅有一个版本，能浏览全部的内容，且网站会自己重新排列以完美地适配任何尺寸的屏幕，并具有全尺寸文本，这样无需进行缩放操作。EthanMarcotte在2010年5月份提出了响应式布局的概念，简而言之，就是一个网站能够兼容多个不同的终端设备。

响应式开发的原理

使用CSS3中的**Media Query（媒体查询）**针对不同宽度的设备设置不同的布局和样式，从而适配不同的设备。

媒体查询的语法格式如下：

```
@media media-type and (media-feature-rule) {
  /* CSS rules go here */
}
```

① 媒体类型：告诉浏览器页面显示在什么类型的设备上，常见的设备包括屏幕（screen）、打印机（print）等。

② 媒体特征规则：设备宽高、方向（横屏或竖屏）、分辨率等特征。

③ CSS规则：会在媒体特征通过且媒体类型正确的时候应用。

响应式开发

响应式开发的原理是使用CSS3中的Media Query（媒体查询）针对不同宽度的设备设置不同的布局和样式，从而适配不同的设备。

媒体类型

媒体类型是指设备类型，常用的有如下几种：

① all：所有类型设备；

② screen：计算机、平板电脑、智能手机等；

③ print：打印机和打印预览；

④ speech：语音或音频合成器；

例如，可分别设置屏幕设备和打印机的显示样式，屏幕设备上h1标题显示蓝色，所有设备包括屏幕设备或者打印机上段落文字加粗显示，代码如下：

```
<h1>响应式开发</h1>
<p>响应式开发的原理是使用CSS3中的Media Query（媒体查询）针对不同宽度的设备设置不同的布局和样式，从而适配不同的设备。</p>
```

```
@media all{
    p{
      font-weight: bold;
    }
}
```

```
@media screen{
    h1{
      color:blue;
    }
}
```

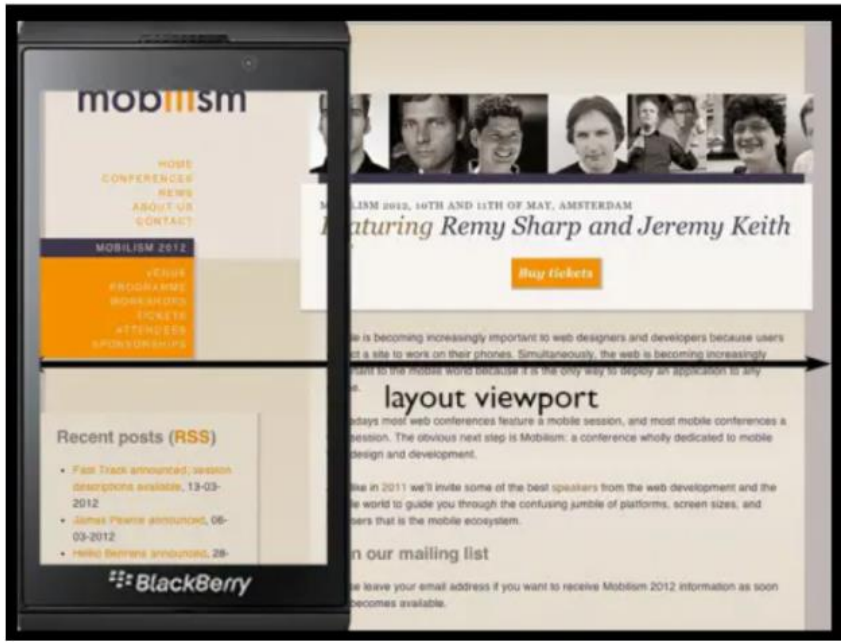
媒体特征规则

媒体特征规则就是设备宽高、横屏或竖屏、屏幕分辨率等相关特征。媒体特征涉及的一个重要概念是视口，视口是与设备相关的一个矩形区域，可以采用相对单位描述其尺寸，并规定其最大或最小宽度。

1、视口

视口分为两类：一类是设备视口，一类是布局视口。

设备本身的宽度



页面的宽度

媒体特征规则

一般最常探测的媒体特征是视口宽度，而且可以使用min-width、max-width和width媒体特征，在视口宽度大于或者小于某个大小——或者是恰好处于某个大小——的时候，应用 CSS样式规则。想在视口正好是 600px的时候，让 body 的文本变为红色。

```
@media screen and (width: 600px) {
    body {
        color: red;
    }
}
```

媒体特征规则

max-width和min-width属性

① max-width：是指视口的最大宽度，当视口宽度小于或等于指定的宽度时，样式生效。当PC端优先（指开发时优先考虑PC用户体验）、适配移动端时使用max-width。

```
@media screen and (max-width: 400px) {
    body {
        color: blue;
    }
}
```

② min-width：是指视口的最小宽度，当视口宽度大于或等于指定的宽度时，样式生效。当移动端优先、适配PC端时使用min-width。

媒体特征规则

可以应用meta元素对响应式布局视口做出规定，例如：

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

① width=device-width表示布局视口宽度要与设备视口宽度缩放到一致

② initial-scale=1.0表示页面最初加载时的缩放比例，即device-width/width=1.0

③ initial-scale和width可以只设置其中一个，另一个会被推算出来；如果initial-scale和width同时设置，会取计算后大的那一个，保证页面宽度刚好容纳进设备视口。

设备视口的宽度分类：（手机->平板->桌面显示器->大桌面显示器）

设备屏幕	尺寸/px
超小屏extra small	<768
小屏small	>=768
中等medium	>=992
大屏large	>=1200

复杂媒体查询

可以应用and、not等逻辑运算符或者逗号连接多个条件从而创建复杂查询。

1、and运算符

可以使用and连接多个媒体特征，表示同时满足若干条件时样式生效。

```
@media screen and (min-width: 400px) and (orientation: landscape) {
    body {
        color: blue;
    }
}
```

2、not运算符

not让整个媒体查询失效，直接反转了整个媒体查询的含义。

```
@media not all and (max-width:900px) {
    body {
        color: brown;
    }
}
```

复杂媒体查询

3、逗号

逗号分隔多个媒体查询条件时，相当于or，表示满足其中任意一个条件都会应用对应的样式。

```
@media screen and (min-width: 400px), screen and (orientation: landscape) {
    body {
        color: blue;
    }
}
```

文本会在视口至少为 400 像素宽的时候或者设备处于横放状态的时候变为蓝色。

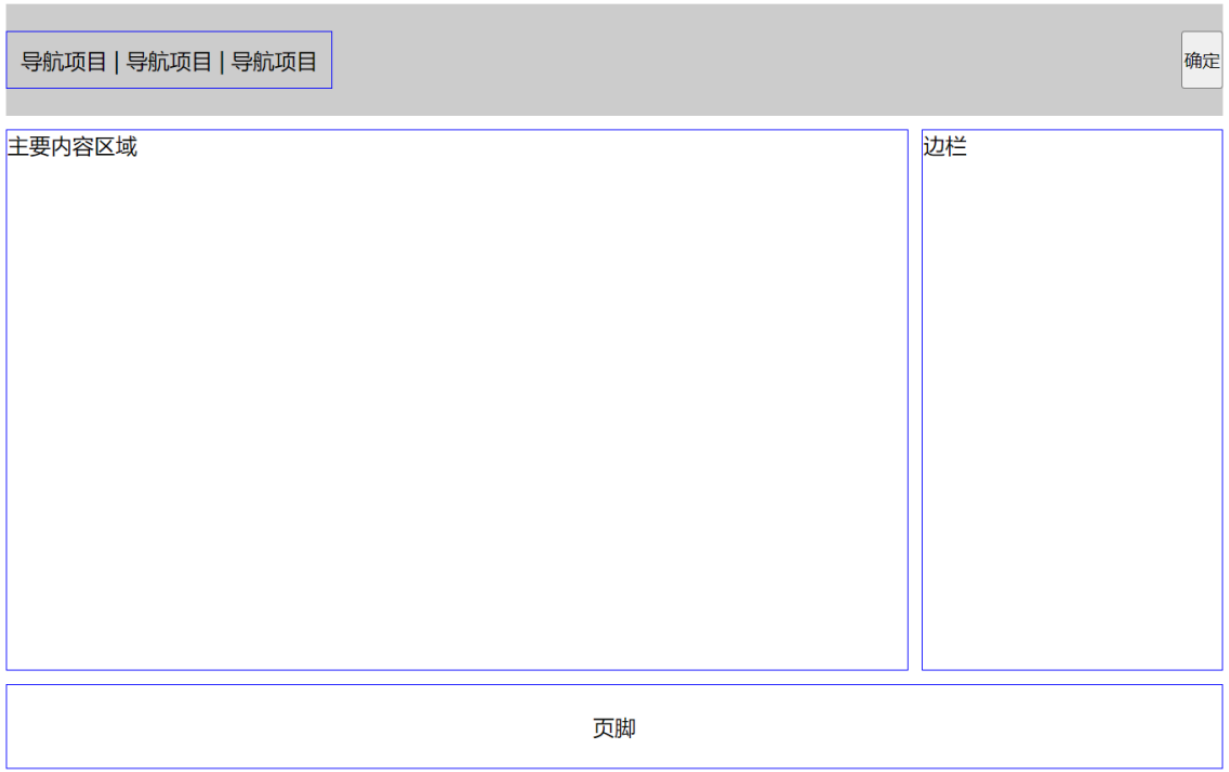
link方式添加媒体查询方式

可以使用link标签的media属性完成特定样式的呈现，例如下面代码可以在宽度小于或等于800px时应用middle.css样式。

<link rel=“stylesheet” media=“max-width:800px” href=“middle.css” />

综合案例

1、应用媒体查询和网格布局，完成响应式布局，当宽度大于600px时，呈现如左图所示的布局，当宽度小于或等于600px时，呈现如右图所示的布局。



综合案例

html代码

```
<div class="container">
    <header>
      <nav>导航项目 | 导航项目 | 导航项目</nav>
      <button>确定</button>
    </header>
    <main>主要内容区域</main>
    <aside>边栏</aside>
    <footer>页脚</footer>
  </div>
```

综合案例

style.css代码

```
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
.container{
  max-width:900px;
  margin: 10px auto;
  display: grid;
  /* 总体布局分为3行2列 */
  /* 2列，宽度3:1 */
  grid-template-columns: 3fr 1fr;
  /* 网格区域命名 */
  grid-template-areas:
  "header header"
  "main aside"
  "footer footer";
  /* 网格间隙10px */
  grid-gap:10px;
}
header{
  grid-area: header;
  display: grid;
  grid-template-columns: 1fr 1fr;
  background-color: #ccc;
  padding: 20px 0;
}
header nav{
  justify-self: start;
  border: 1px solid blue;
  padding: 10px;
}
header button{
  justify-self: end;
}
/* 主要内容区域 */
main{
  grid-area: main;
  height: 400px;
  border: 1px solid blue;
}
aside{
  grid-area:aside;
  border: 1px solid blue;
}
footer{
  grid-area: footer;
  border: 1px solid blue;
  text-align:center;
  padding: 20px 0;
}
/* 媒体查询实现响应式布局 */
/* 宽度小于或等于600px，变为单列布局 */
@media(max-width:600px){
  .container{
    grid-template-columns: 1fr;
    grid-template-areas:
    "header"
    "main"
    "aside"
    "footer";
  }
}
```