网页开发——HTML

目录

练习1——HTML练习

练习要求

编写HTML代码，实现以下界面：

学生成绩表

姓名	学号	综合成绩	备注	操作1	操作2
张三	001	95	无	修改	删除
李四	002	59	该学生存在问题： 1. 早八迟到 2. 作业不交	修改	删除
小明	003	100	**学生表现优秀，** 开发过优秀的开源项目： 开源项目	修改	删除

练习参考代码

table的整体结构：

```
<body>
    <table border="2">
        <caption align="top">学生成绩表</caption>
        <tr>…
        </tr>
        <tr>…
        </tr>
        <tr>…
        </tr>
        <tr>…
        </tr>
    </table>
</body>
```

学生成绩表

姓名	学号	综合成绩	备注	操作1	操作2
张三	001	95	无	修改	删除
李四	002	59	该学生存在问题： 1. 早八迟到 2. 作业不交	修改	删除
小明	003	100	**学生表现优秀，** 开发过优秀的开源项目： 开源项目	修改	删除

练习参考代码

table表头

```
<tr>
    <th>姓名</th>
    <th>学号</th>
    <th>综合成绩</th>
    <th>备注</th>
    <th>操作1</th>
    <th>操作2</th>
</tr>
```

学生成绩表

姓名	学号	综合成绩	备注	操作1	操作2
张三	001	95	无	修改	删除
李四	002	59	该学生存在问题： 1. 早八迟到 2. 作业不交	修改	删除
小明	003	100	**学生表现优秀，** 开发过优秀的开源项目： 开源项目	修改	删除

练习参考代码

table内容——第1行

```
<tr>
    <td align="center">张三</td>
    <td align="center">001</td>
    <td align="center">95</td>
    <td align="left">无</td>
    <td align="center"><button>修改</button></td>
    <td align="center"><button>删除</button></td>
</tr>
```

学生成绩表

姓名	学号	综合成绩	备注	操作1	操作2
张三	001	95	无	修改	删除
李四	002	59	该学生存在问题： 1. 早八迟到 2. 作业不交	修改	删除
小明	003	100	**学生表现优秀，** 开发过优秀的开源项目： 开源项目	修改	删除

练习参考代码

table内容——第2行

```
<tr>
    <td align="center">李四</td>
    <td align="center">002</td>
    <td align="center" ><font color="red">59</font></td>
    <td align="left">
        <font color="red">该学生存在问题：</font>
        <ol>
            <li>早八迟到</li>
            <li>作业不交</li>
        </ol>
    </td>
    <td align="center"><button>修改</button></td>
    <td align="center"><button>删除</button></td>
</tr>
```

学生成绩表

姓名	学号	综合成绩	备注	操作1	操作2
张三	001	95	无	修改	删除
李四	002	59	该学生存在问题： 1. 早八迟到 2. 作业不交	修改	删除
小明	003	100	**学生表现优秀，** 开发过优秀的开源项目： 开源项目	修改	删除

练习参考代码

table内容——第3行

```
<td align="center">小明</td>
<td align="center">003</td>
<td align="center">100</td>
<td align="left">
    <b>学生表现优秀</b>, <br />
    开发过优秀的开源项目：<br />
    <a href="https://github.com">开源项目</a>
</td>
<td align="center"><button>修改</button></td>
<td align="center"><button>删除</button></td>
```

学生成绩表

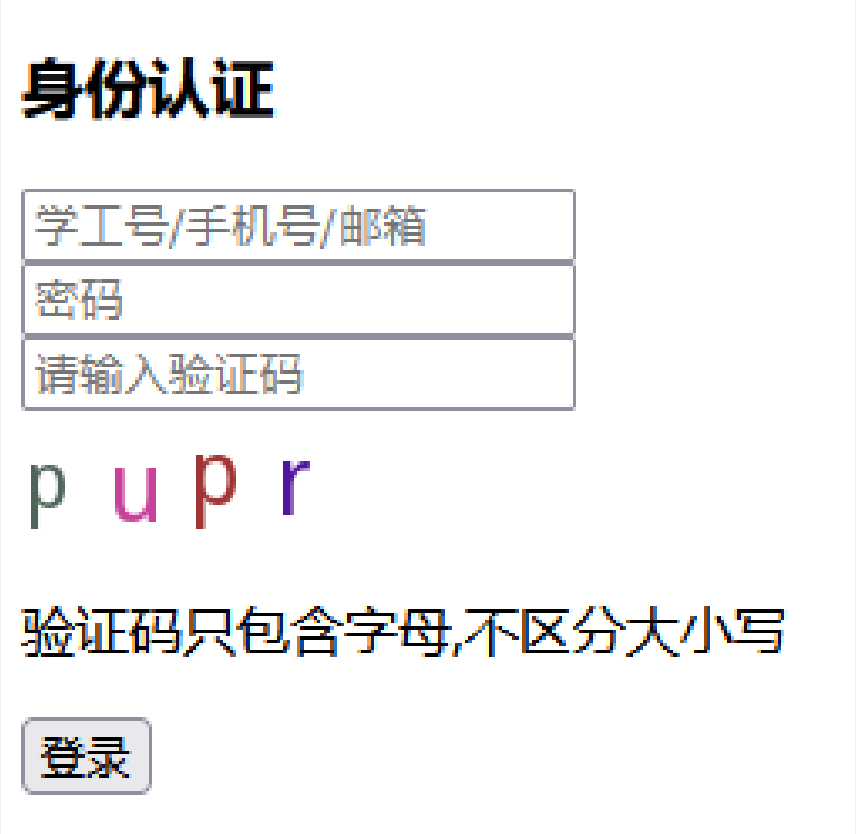
姓名	学号	综合成绩	备注	操作1	操作2
张三	001	95	无	修改	删除
李四	002	59	该学生存在问题： 1. 早八迟到 2. 作业不交	修改	删除
小明	003	100	**学生表现优秀，** 开发过优秀的开源项目： 开源项目	修改	删除

示例1——微人大登录界面

```
9   <body>
10    <div class="card-container">
11      <form>
12        <div id="login-title"><h3>身份认证</h3> </div>
13        <div class="input-group">
14          <input type="text" placeholder="学工号/手机号/邮箱" required>
15        </div>
16        <div class="input-group">
17          <input type="password" placeholder="密码" required>
18        </div>
19        <div class="input-group" id="captcha-input">
20          <input type="text" name="code" placeholder="请输入验证码">
21        </div>
22        <div class="input-group" id="captcha-img"></div>
23
24        <div class="alert"><p id="captcha-info">验证码只包含字母,不区分大小写</p></div>
25        <div><button class="button" type="submit"><span>登录</span></button></div>
26      </form>
27    </div>
28  </body>
```

微人大登录界面的HTML元素与实际界面元素的对应图

由于我们还没有写CSS文件来规定其样式，所以下方的图片就是纯HTML实现的页面

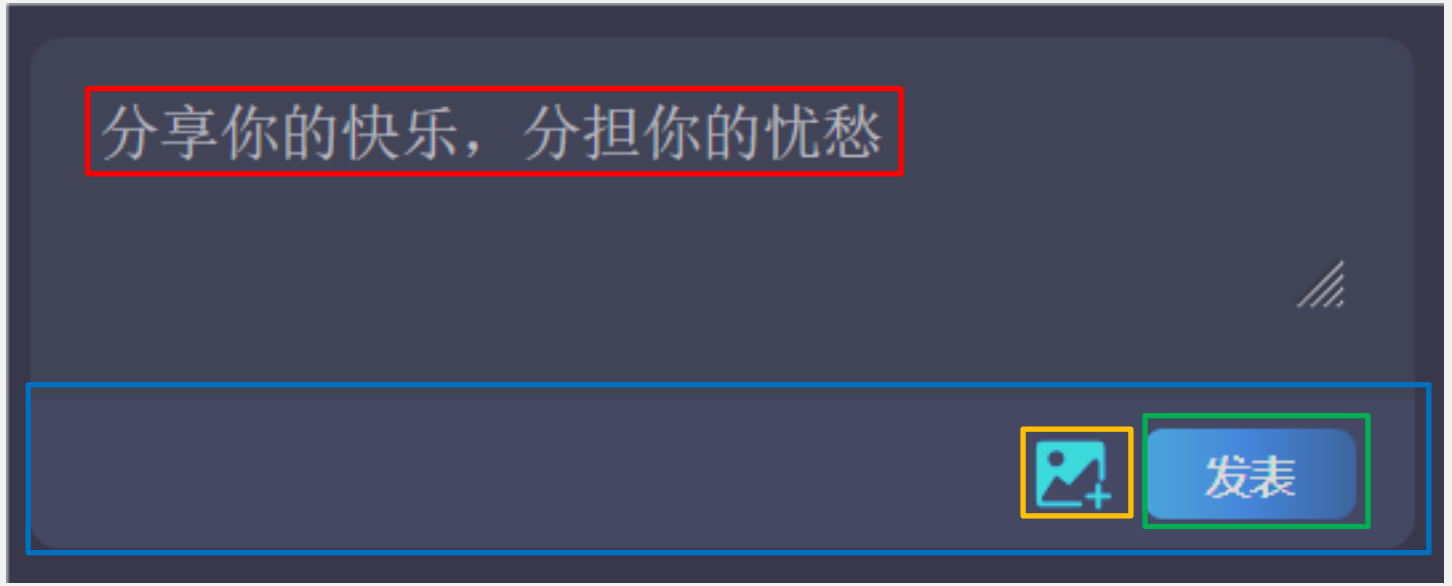


示例2——RUC小喇叭界面

```html
<div class="container">
  <h1>RUC小喇叭</h1>
  <div class="input_container">…
  </div>
  <div class="search_container">…
  </div>
  <div class="post_list">
    <div class="post_container">…
    </div>
    <div class="post_container">…
    </div>
  </div>
</div>
```

输入框部分

```
<div class="input_container">
  <form>
    <textarea placeholder="分享你的快乐，分担你的忧愁" required></textarea>
    <div class="btn_row">
      <img id="add_img_btn" src="./img/add_images.svg" />
      <button type="submit"><span>发表</span></button>
    </div>
  </form>
</div>
```

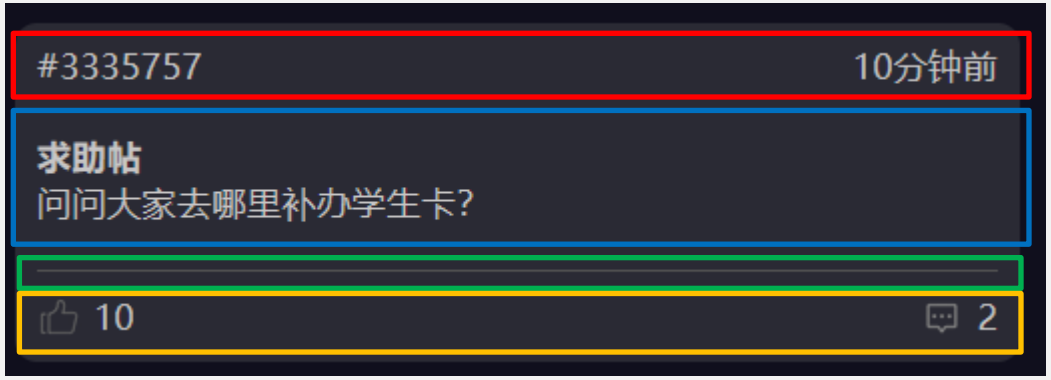


搜索框部分

```
<form>
  <div class="input_area"><input type="text" placeholder="分享你的快乐，分担你的忧愁" required></div>
  <div><button type="submit"><span>搜索</span></button></div>
</form>
```

帖子部分

```
<div class="post_container">
  <div class="post_header">
    <div class="post_id"><span>#3335757</span></div>
    <div class="post_time"><span>10分钟前</span></div>
  </div>
  <div class="post_content">
    <h4>求助帖</h4>
    <p>问问大家去哪里补办学生卡？</p>
  </div>
  <div class="banner"></div>
  <div class="post_bottom">
    <div class="like"><img src="like.png" /><span>10</span></div>
    <div class="message"><img src="message.png" /><span>2</span></div>
  </div>
</div>
```



由于我们还没有写CSS文件来规定其样式，所以下方的图片就是纯HTML实现的页面

网页开发——CSS+JS基础

目录

示例1——微人大登录界面

```
9   <body>
10    <div class="card-container">
11      <form>
12        <div id="login-title"><h3>身份认证</h3> </div>
13        <div class="input-group">
14          <input type="text" placeholder="学工号/手机号/邮箱" required>
15        </div>
16        <div class="input-group">
17          <input type="password" placeholder="密码" required>
18        </div>
19        <div class="input-group" id="captcha-input">
20          <input type="text" name="code" placeholder="请输入验证码">
21        </div>
22        <div class="input-group" id="captcha-img"></div>
23
24        <div class="alert"><p id="captcha-info">验证码只包含字母,不区分大小写</p></div>
25        <div><button class="button" type="submit"><span>登录</span></button></div>
26      </form>
27    </div>
28  </body>
```

微人大登录界面的HTML元素与实际界面元素的对应图

```css
/* 基础标题样式 - 三级标题 */
h3 {
  font-size: 17px;          /* 中等字号设置 */
  font-weight: 400;         /* 常规字重（非粗体） */
}
```

```html
<div id="login-title"><h3>身份认证</h3> </div>
```

```css
/* 主卡片容器 - 登录/注册表单容器 */
.card-container {
  max-width: 300px;                                  /* 最大宽度限制 */
  margin: 8vw auto;                                  /* 垂直间距8%视口宽度 + 水平居中 */
  background: #fff;                                  /* 纯白背景 */
  width: 100%;                                       /* 自适应父容器宽度 */
  box-shadow: 0 2px 2px 0 rgb(0 0 0 / 14%),          /* 三层阴影实现立体效果 */
    0 3px 1px -2px rgb(0 0 0 / 20%),
    0 1px 5px 0 rgb(0 0 0 / 12%);
  padding: 2rem 3rem;                                /* 内边距：上下2rem，左右3rem */
  border-radius: 1px;                                /* 极小的圆角 */
}
```

```css
/* 输入框组样式 */
.input-group > input {
  background-color: white;      /* 输入框背景 */
  border-radius: 3px;           /* 圆角效果 */
  box-sizing: border-box;       /* 盒模型计算方式 */
  border: 1px solid #ccc;       /* 灰色边框 */
  margin-bottom: -1px;          /* 负边距消除双边框重叠 */
  width: 100%;                  /* 占满父容器宽度 */
  height: 45px;                 /* 固定高度 */
  outline: none;                /* 移除默认聚焦轮廓 */
  padding: 0 16px 0 55px;       /* 左右内边距（预留图标空间） */
  position: relative;           /* 相对定位上下文 */
}
```

```html
<div class="input-group">
  <input type="password" placeholder="密码" required>
</div>
```

```css
/* 输入框聚焦状态 */
.input-group > input:focus {
  border: 1px solid #4DD2FF;    /* 天蓝色聚焦边框 */
  z-index: 100;                 /* 提升层级防止边框重叠 */
}
```

```html
<div><button class="button" type="submit"><span>登录</span></button></div>
```

```css
/* 按钮文字容器 - 确保文字显示层级 */
.button span {
  color: #fff;              /* 纯白文字保证对比度 */
  position: relative;       /* 维持相对定位上下文 */
  z-index: 1;               /* 提升层级确保文字在背景效果之上 */
}
```

```css
/* 通用按钮样式 - 用于主要操作按钮 */
.button {
  position: relative;       /* 创建定位上下文（用于可能的伪元素或动画效果） */
  font-size: 0.85rem;       /* 小型文本字号（约13.6px） */
  background: #d75757;          /* 品牌红色作为主背景色 */
  border: 1px solid #891524;  /* 深红色边框增强层次感 */
  padding: .3rem 2rem;      /* 内边距：垂直紧凑，水平宽松 */
  border-radius: 2px;       /* 轻微圆角保持现代感 */
  width: 100%;              /* 充满父容器宽度 */
  cursor: pointer;          /* 手型光标提示可点击 */
  overflow: hidden;         /* 隐藏溢出内容（常用于波纹动画效果） */
}
```

```css
/* 验证码输入区 */
#captcha-input {
  width: 45%;          /* 占容器45%宽度 */
  float: left;         /* 左浮动布局 */
  margin-top: 15px;    /* 上边距 */
  display: block;      /* 块级元素 */
}
```

```css
/* 验证码图片样式 - 包含边框和尺寸控制 */
#captcha-img img {
  border: 1px solid #ccc;  /* 浅灰边框（与输入框保持视觉统一） */
}
```

```css
/* 验证码信息提示区 - 用于辅助说明 */
#captcha-info {
  clear: both;                  /* 清除浮动确保独立成行 */
  color: rgb(102, 102, 102);  /* 中性灰文字颜色（降低视觉强度） */
  line-height: 20px;            /* 舒适的行高（1.5倍字体大小） */
  font-size: 13px;              /* 小号辅助文字 */
  display: block;               /* 块级元素特性 */
  margin-top: 80px;             /* 超大顶部间距（隔离表单主体内容） */
}
```

```html
<div class="input-group" id="captcha-img"></div>

<div class="alert"><p id="captcha-info">验证码只包含字母,不区分大小写</p></div>
```

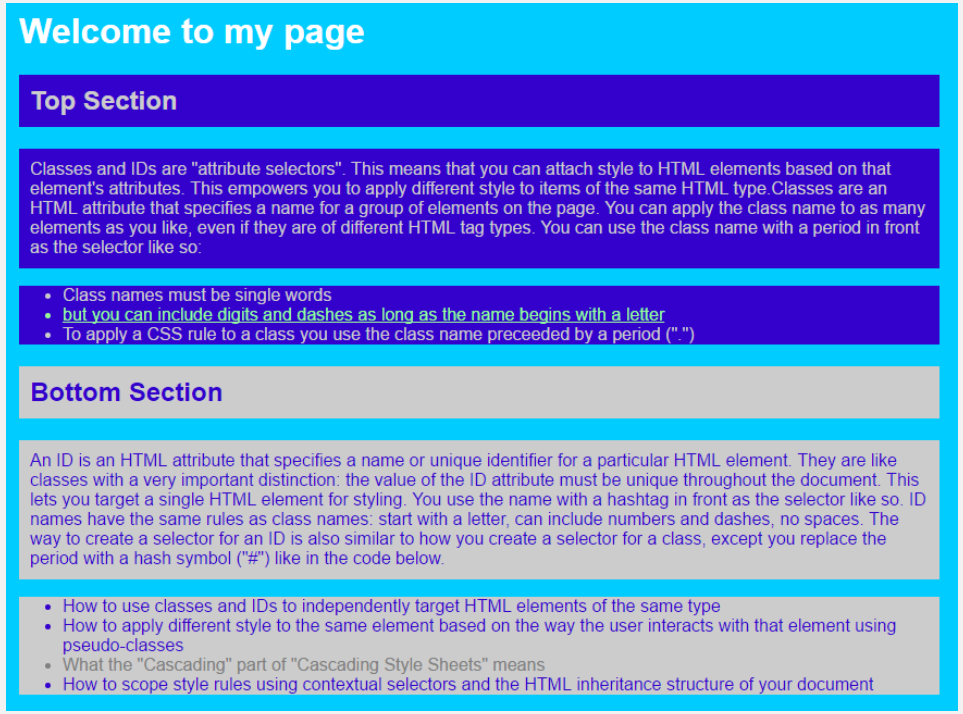

```html
<div class="input-group" id="captcha-input">
  <input type="text" name="code" placeholder="请输入验证码">
</div>
```

练习1——CSS练习

练习题目

练习1——属性选择器

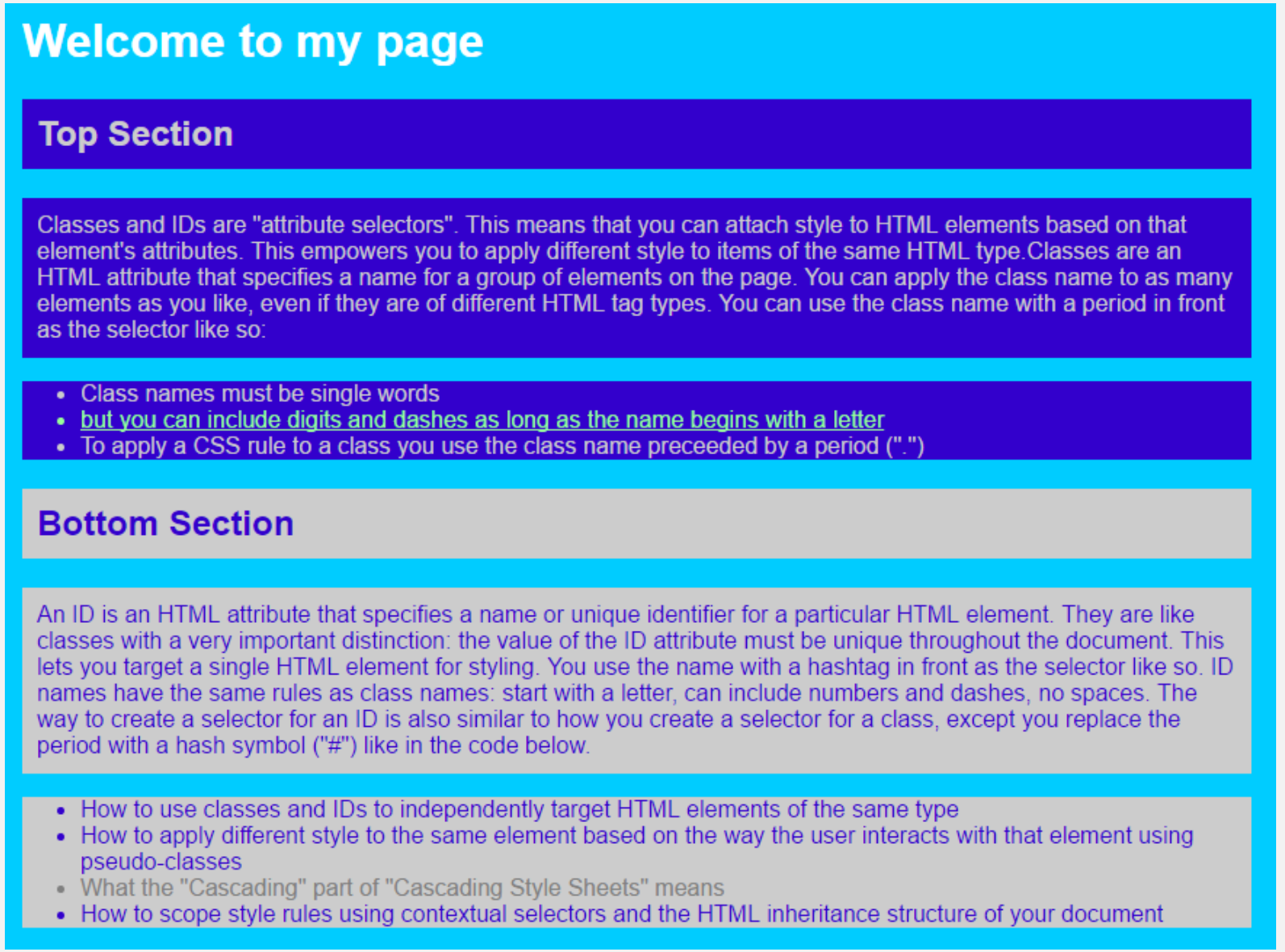
https://codepen.io/techie4good/pen/VKkJYd



练习2——上下文选择器+伪类/伪元素

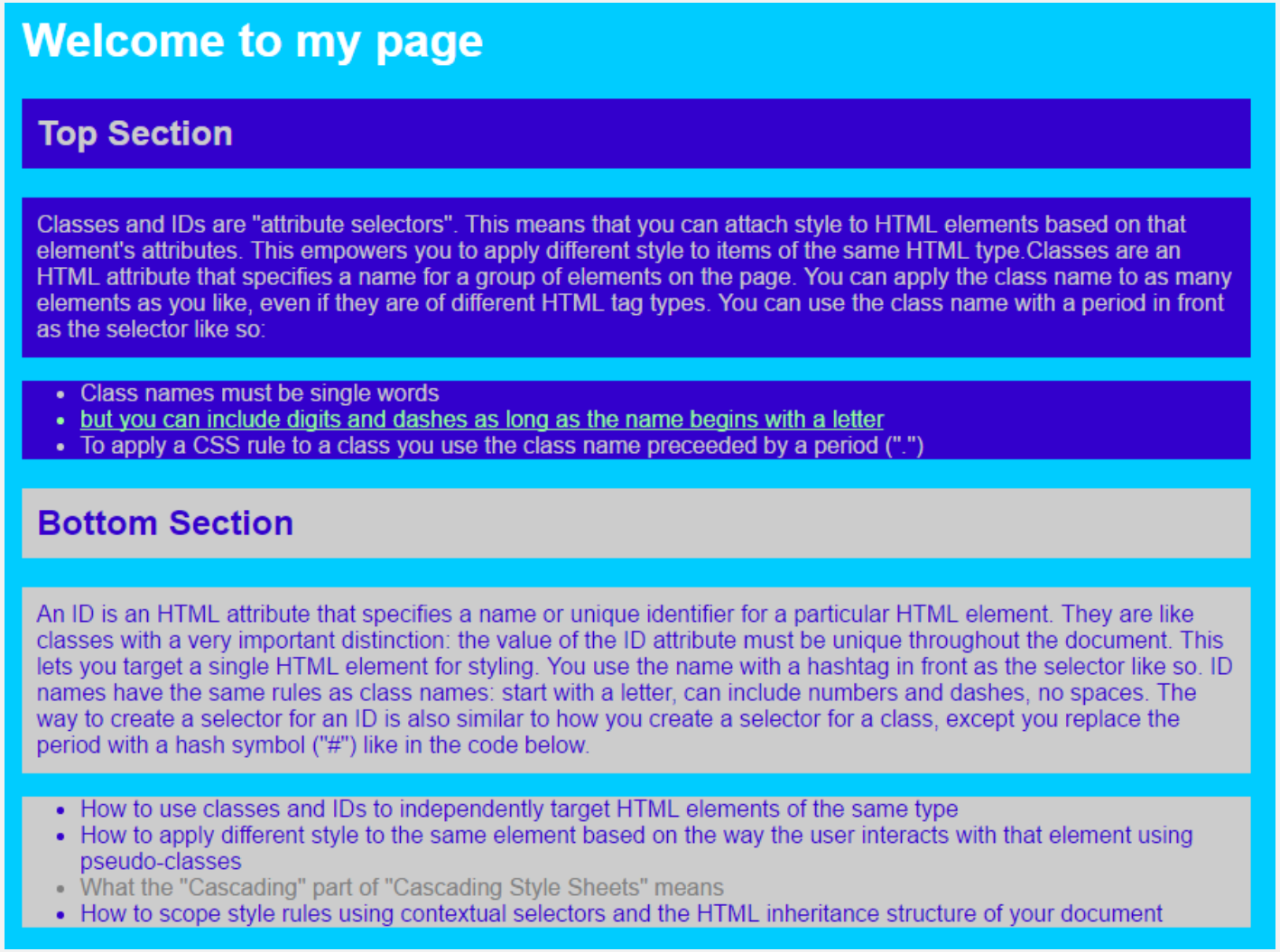
https://codepen.io/techie4good/pen/qaERNj

练习1题目要求：无需修改CSS，需要修改HTML，为DOM加上class或id。



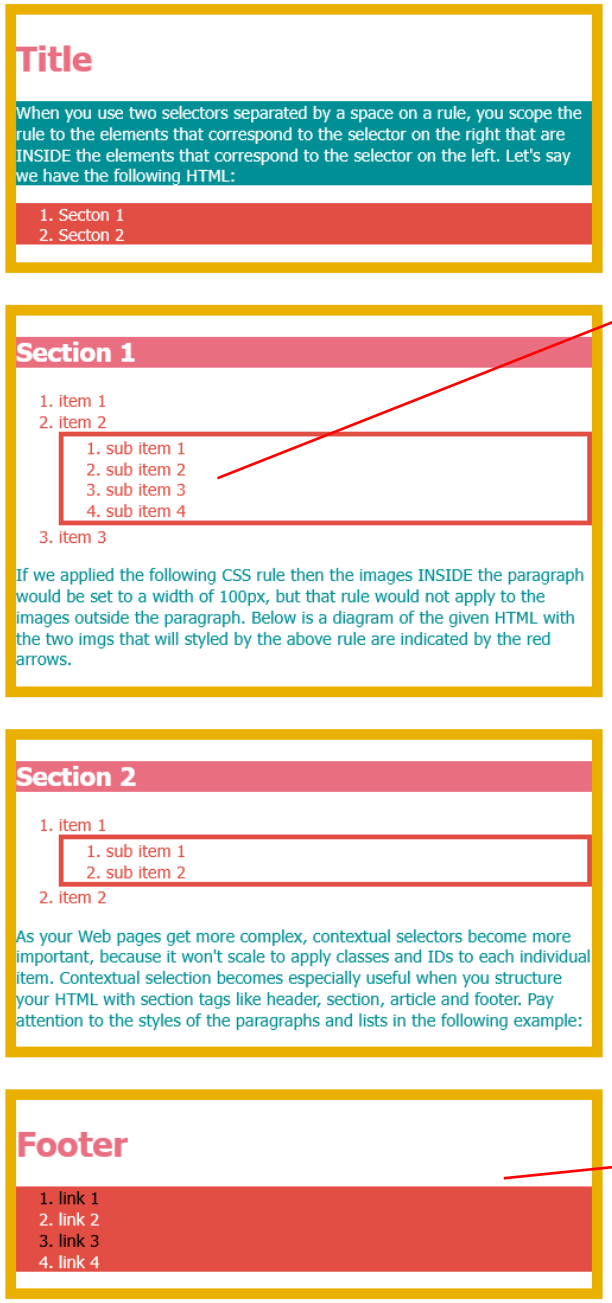
参考代码如下：

```
    <h1>Welcome to my page</h1>
    <h2 class="topSection">Top Section</h2>
    <p class="topSection">Classes and IDs are
"attribute selectors". ...</p>
    <ul class="topSection">
      <li>Class names must be single words</li>
      <li id="importantItem">but you can include
digits and dashes as long as the name begins with a
letter</li>
      <li>To apply a CSS rule to a class you use the
class name preceeded by a period (".")</li>
    </ul>
    <h2 class="bottomSection">Bottom Section</h2>
    <p class="bottomSection">An ID is an HTML
attribute...    </p>
    <ul class="bottomSection">
      <li>How to use classes and IDs to
independently target HTML elements of the same
type</li>
      <li>How to apply different style to the same
element based on the way the user interacts with
that element using pseudo-classes</li>
      <li id="unimportantItem">What the "Cascading"
part of "Cascading Style Sheets" means</li>
      <li>How to scope style rules using contextual
selectors and the HTML inheritance structure of your
document</li>
    </ul>
```



练习2
题目要求：

无需修改HTML，只需要修改CSS



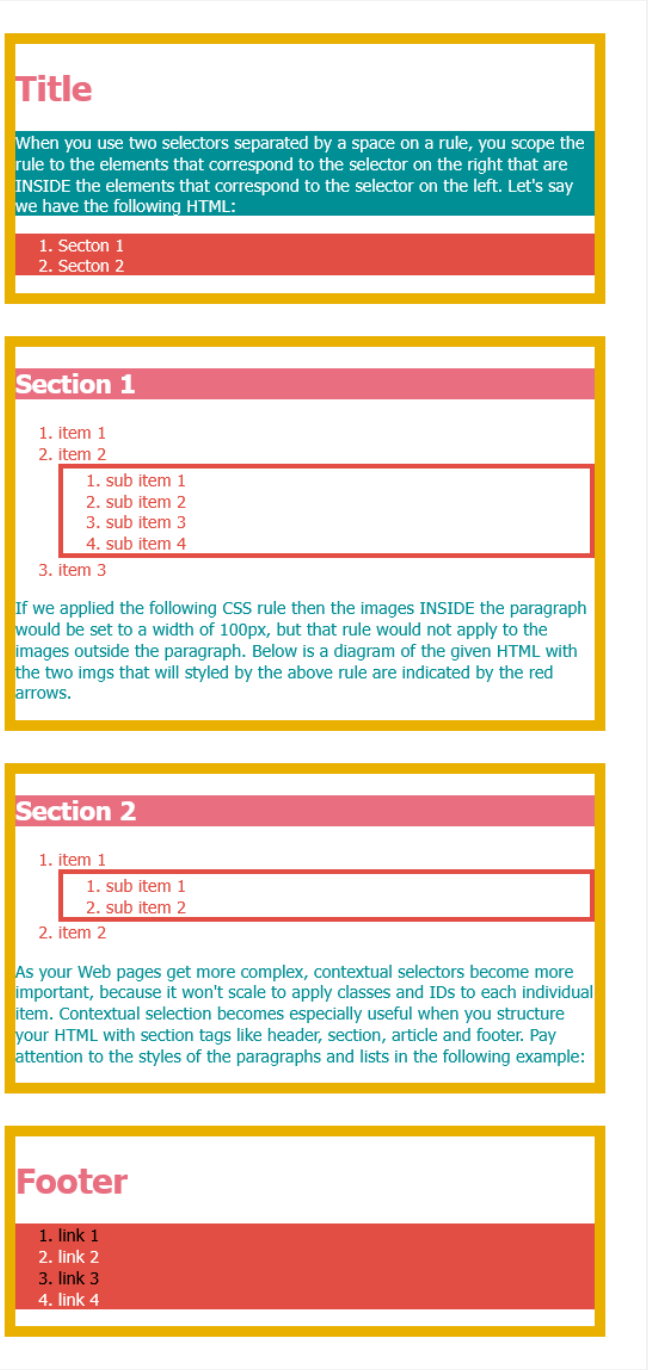
当光标悬浮到元素上方时，background变为blue

列表的奇数行文字颜色为黑色（#000）
偶数行文字颜色为白色（#FFF）

参考代码如下：

```
body {
  font-family: Tahoma, sans-serif;
}

header, article, footer {
  border: 10px #E9B000 solid;
  margin: 30px;
}

footer h1, header h1 {
  color: #E86E80;
}

article h1 {
  background-color: #E86E80;
  color: #FFFFFF;
}

header p {
  background-color: #008F95;
  color: #FFFFFF;
}

article p {
  background-color: #FFFFFF;
  color: #008F95;
}
```

```
header ol,
footer ol {
    background-color: #E24E43;
    color: #FFFFFF;
}

footer ol li:nth-child(odd) {
    color: #000;
}

footer ol li:nth-child(even) {
    color: #FFFFFF;
}

article ol {
    background-color: #FFFFFF;
    color: #E24E43;
}

article ol ol {
    border: 5px solid #E24E43;
}

article ol ol>li:hover {
    background-color: blue;
}
```



CSS知识扩展——FLEX布局

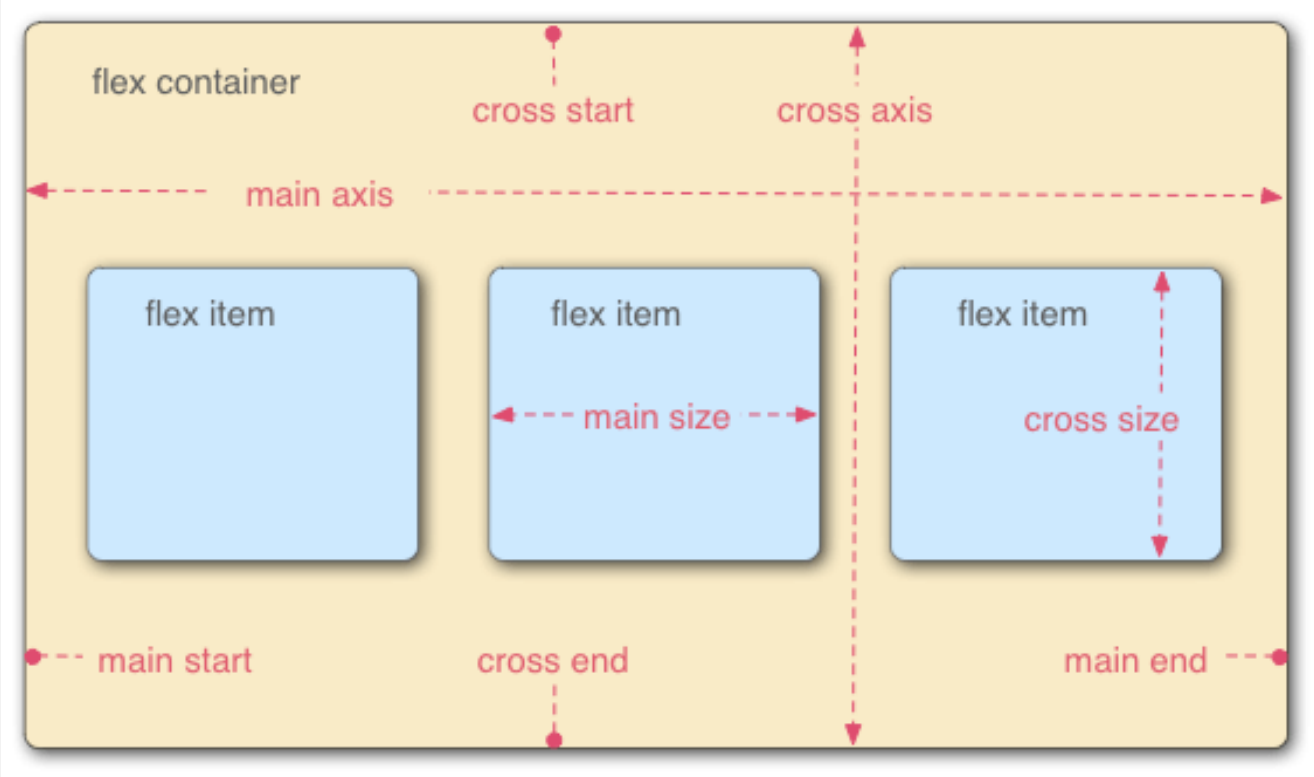
教程参考:
https://www.runoob.com/w3cnote/flex-grammar.html

flex布局简介

Flex是Flexible Box的缩写，意为”弹性布局”，用来为盒状模型提供最大的灵活性。任何一个容器都可以指定为Flex布局。

采用Flex布局的元素，称为Flex容器（flex container），简称”容器”。它的所有子元素自动成为容器成员，称为Flex项目（flex item），简称”项目”。

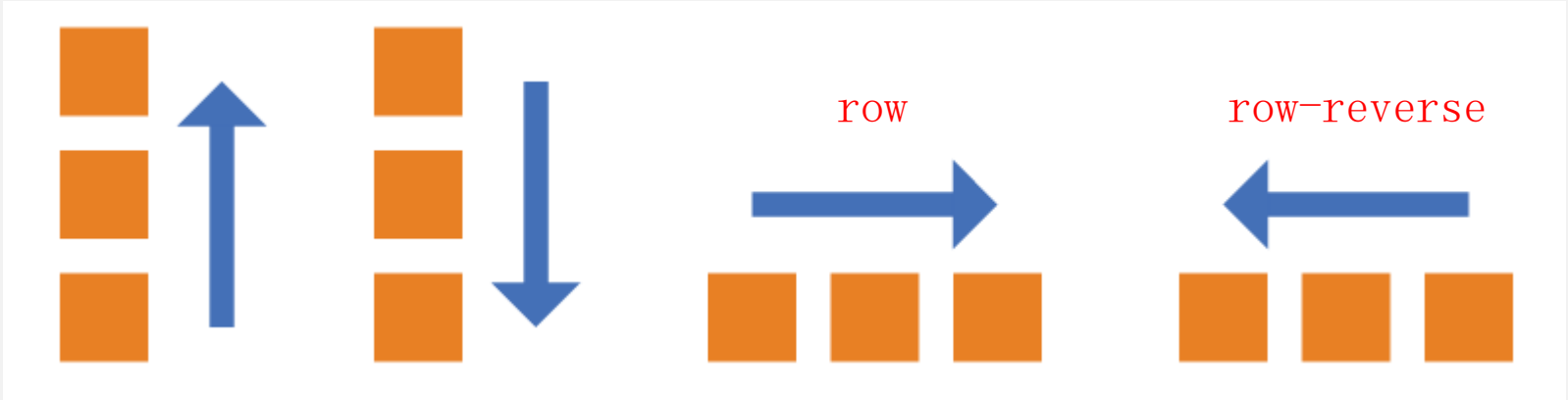
- 容器默认有两个轴：水平主轴（main axis）和垂直的交叉轴（cross axis）
- 主轴的开始位置（与边框的交叉点）叫做main start，结束位置叫做main end。
- 交叉轴的开始位置叫做cross start，结束位置叫做cross end。
- 项目默认沿主轴排列。单个项目占据的主轴空间叫做main size，占据的交叉轴空间叫做cross size。



flex-direction属性

- flex-direction属性决定主轴的方向（即项目的排列方向）

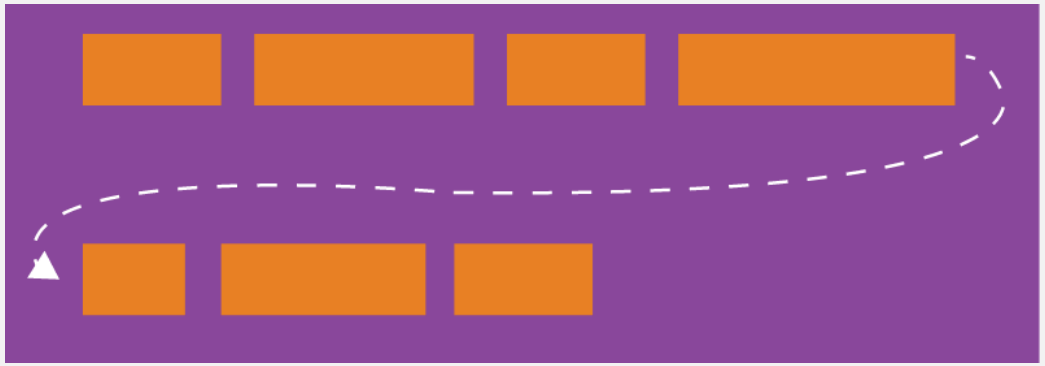
```
.box {
  flex-direction: row | row-reverse | column | column-reverse;
}
```



- row（默认值）：主轴为水平方向，起点在左端。
- row-reverse：主轴为水平方向，起点在右端。
- column：主轴为垂直方向，起点在上沿。
- column-reverse：主轴为垂直方向，起点在下沿。

flex-wrap属性

- 默认情况下，项目都排在一条线（又称”轴线”）上。flex-wrap属性定义，如果一条轴线排不下，如何换行

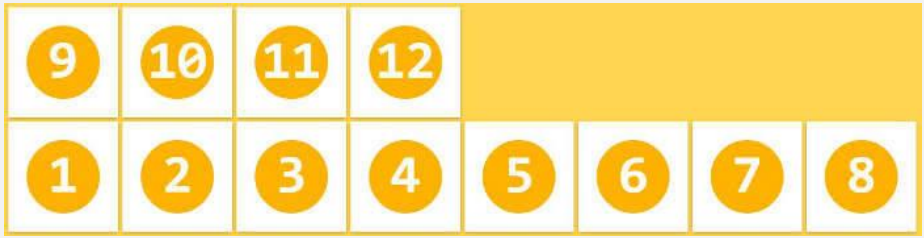


```
.box{
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

- nowrap（默认）：不换行

- wrap：换行，第一行在上方

- wrap-reverse：换行，第一行在下方



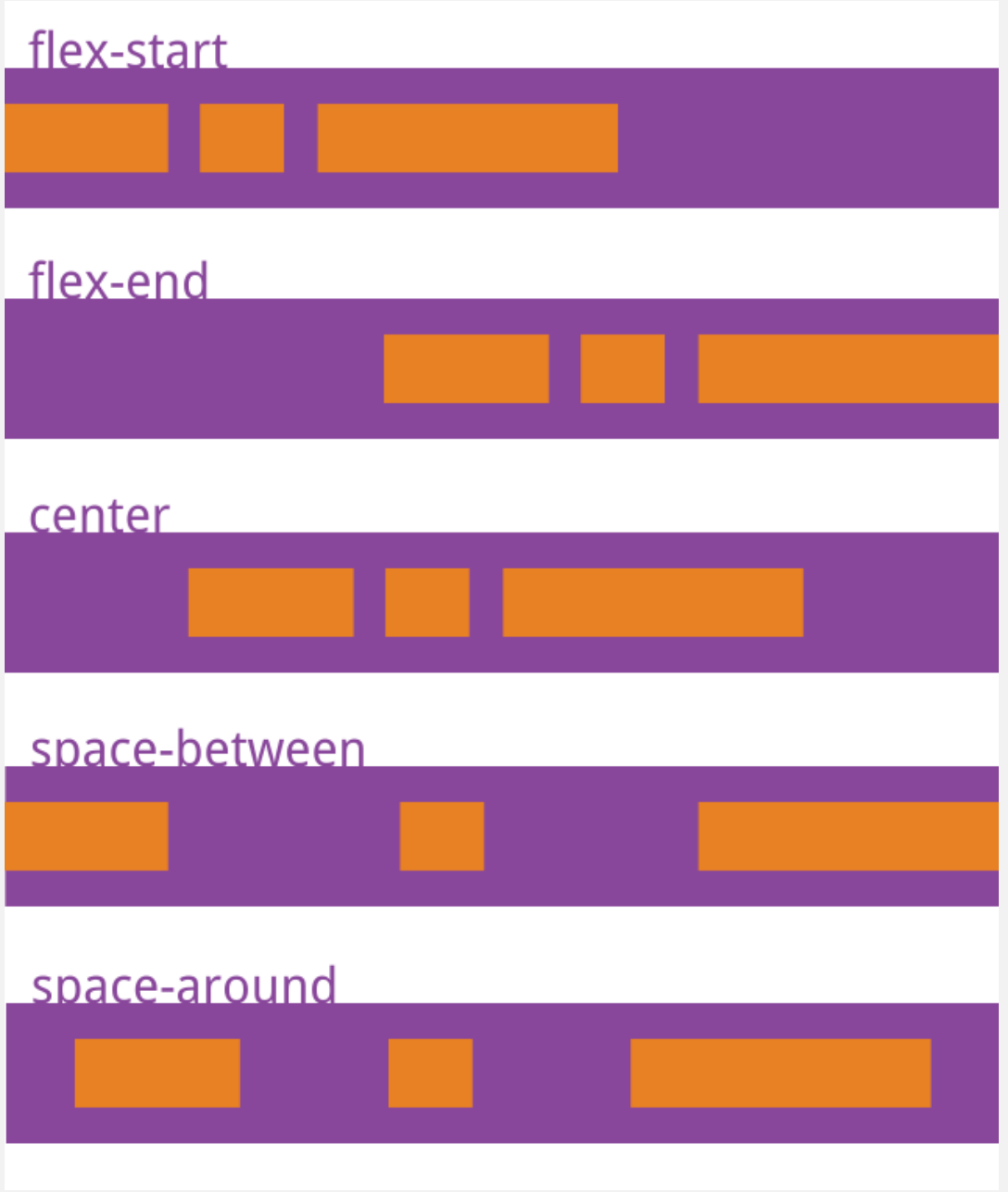
justify-content属性

- justify-content属性定义了项目在主轴上的对齐方式

```
.box {
  justify-content: flex-start | flex-end | center | space-between | space-around;
}
```

它可能取5个值，具体对齐方式与轴的方向有关。下面假设主轴为从左到右（轴为其他方向时类似）

- flex-start（默认值）：左对齐
- flex-end：右对齐
- center： 居中
- space-between：两端对齐，项目之间的间隔都相等。
- space-around：每个项目两侧的间隔相等。所以，项目之间的间隔比项目与边框的间隔大一倍。



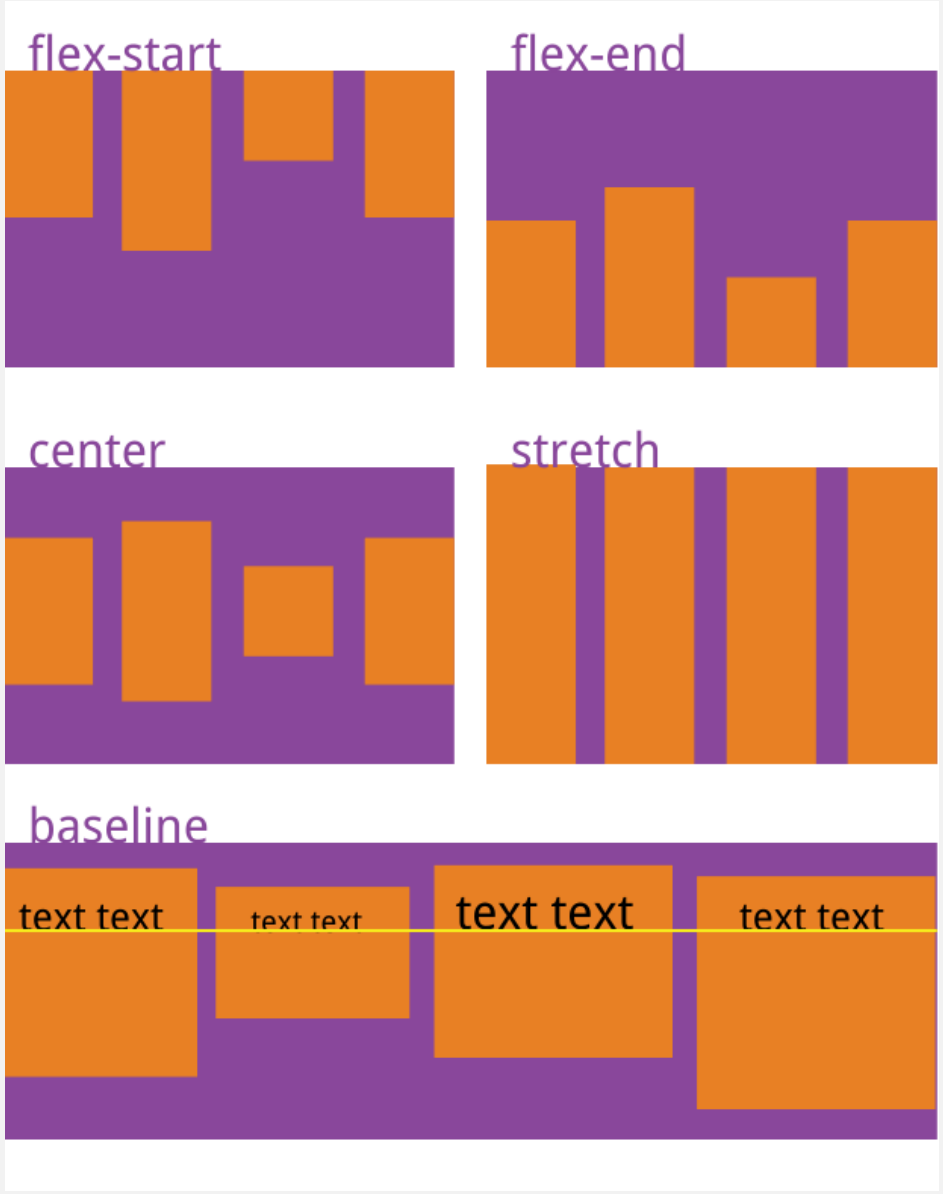
align-items属性

- align-items属性定义项目在交叉轴上如何对齐

```
.box {
  align-items: flex-start | flex-end | center | baseline | stretch;
}
```

它可能取5个值。具体的对齐方式与交叉轴的方向有关，下面假设交叉轴从上到下。（轴为其他方向时类似）

- flex-start：交叉轴的起点对齐。
- flex-end：交叉轴的终点对齐。
- center：交叉轴的中点对齐。
- baseline：项目的第一行文字的基线对齐。
- stretch（默认值）：如果项目未设置高度或设为auto，将占满整个容器的高度。



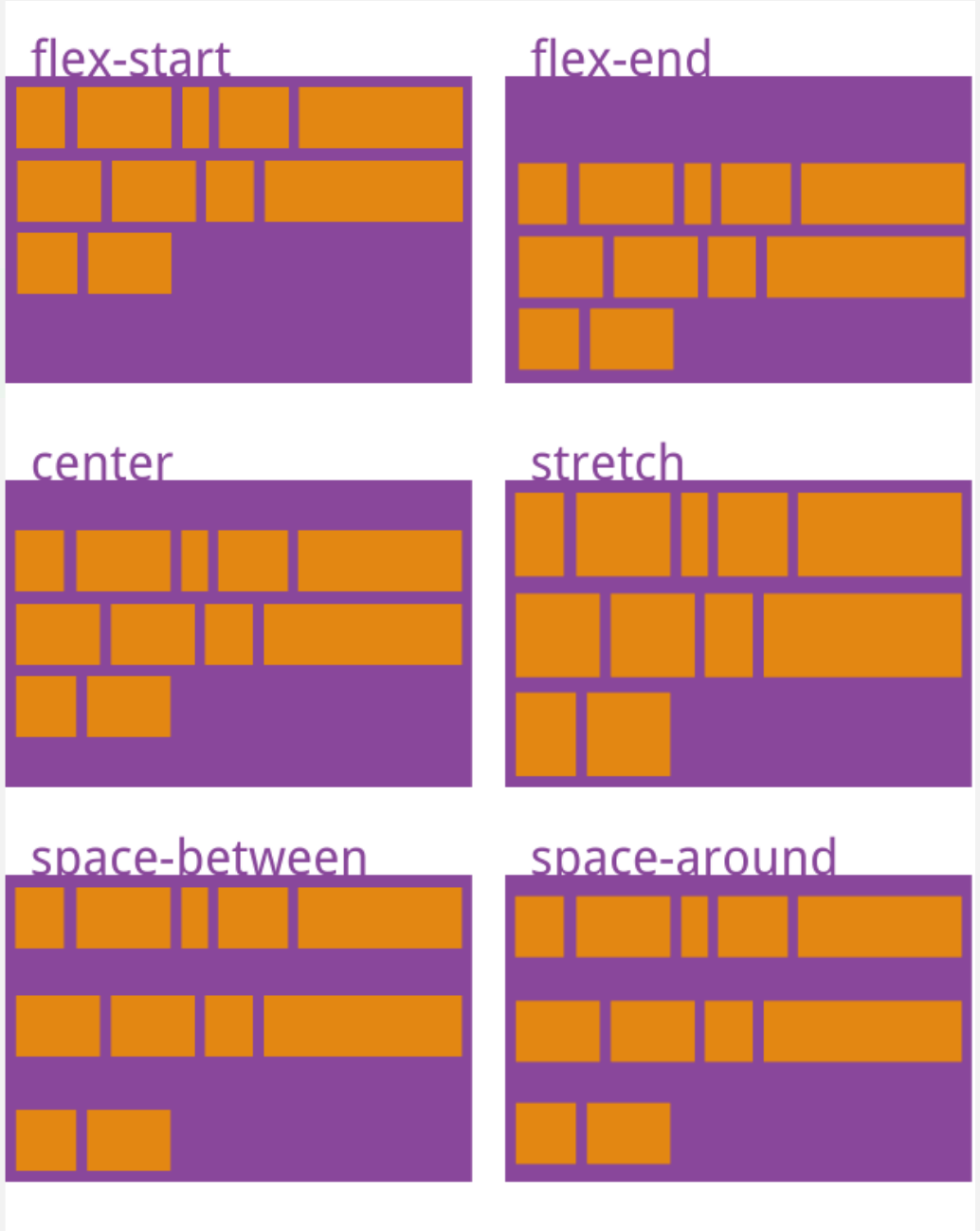
align-content属性

- align-content属性定义了多根轴线的对齐方式。如果项目只有一根轴线，该属性不起作用。

```
.box {
  align-content: flex-start | flex-end | center | space-between | space-around | stretch;
}
```

该属性可能取6个值

- flex-start：与交叉轴的起点对齐。
- flex-end：与交叉轴的终点对齐。
- center：与交叉轴的中点对齐。
- space-between：与交叉轴两端对齐，轴线之间的间隔平均分布。
- space-around：每根轴线两侧的间隔都相等。所以，轴线之间的间隔比轴线与边框的间隔大一倍。
- stretch（默认值）：轴线占满整个交叉轴。



示例2——RUC小喇叭界面（静态）

```html
<div class="container">
  <h1>RUC小喇叭</h1>
  <div class="input_container">…
  </div>
  <div class="search_container">…
  </div>
  <div class="post_list">
    <div class="post_container">…
    </div>
    <div class="post_container">…
    </div>
  </div>
</div>
```

```css
/* 主内容容器 - 包裹所有页面元素 */
.container {
    width: 500px;                  /* 固定宽度布局 */
    display: flex;                 /* 启用flex布局 */
    flex-direction: column;        /* 垂直排列子元素 */
    background-color: #100f1d;     /* 深蓝黑背景 */
    color: #c8c6c6;                /* 浅灰文字颜色 */
}
```

```css
/* 主标题特殊样式 - 覆盖容器文字颜色 */
.container>h1 {
    margin-top: 20px;    /* 顶部间距 */
    color: #fff;         /* 纯白文字增强对比度 */
}
```

```html
<div class="container">
  <h1>RUC小喇叭</h1>
  <div class="input_container">···
  </div>
  <div class="search_container">···
  </div>
  <div class="post_list">
    <div class="post_container">···
    </div>
    <div class="post_container">···
    </div>
  </div>
</div>
```

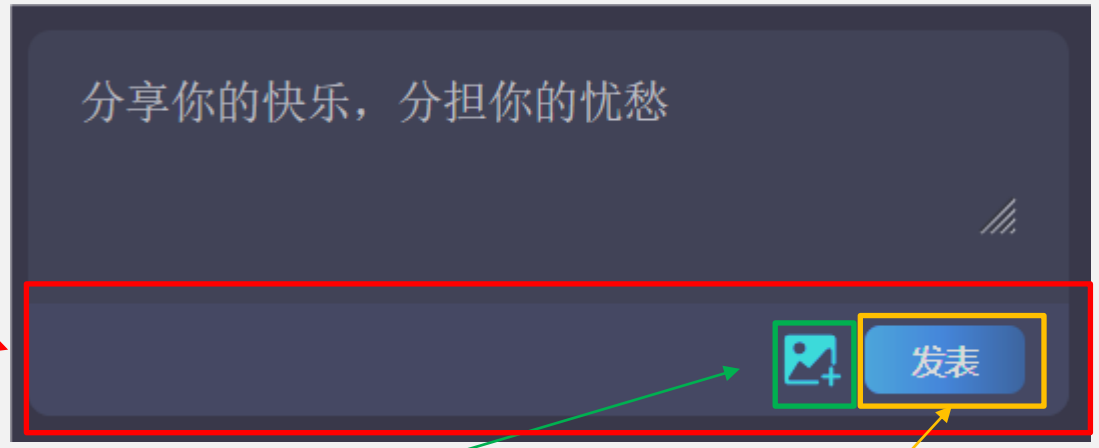
```css
/* 输入区域容器 - 包含文本域和按钮 */
.input_container {
    margin-top: 50px;        /* 与上方元素间距 */
    border-radius: 10px;     /* 圆角效果 */
    margin-right: 20px;      /* 右侧留白 */
    margin-left: 20px;       /* 左侧留白 */
    display: flex;           /* flex布局 */
    flex-direction: column;  /* 垂直排列 */
    background: #161827;     /* 深色背景 */
}
```


```css
/* 文本输入域样式 */
.input_container textarea {
    background: #161827;     /* 继承容器背景色 */
    border: 0;               /* 无边框 */
    width: 90%;              /* 相对宽度 */
    margin-top: 20px;        /* 顶部间距 */
    min-height: 70px;        /* 最小高度 */
    color: #c8c6c6;          /* 文字颜色 */
    font-size: 20px;         /* 大号字体 */
    border-radius: 10px;     /* 匹配容器圆角 */
    margin-bottom: 30px;     /* 底部间距 */
    resize: vertical;        /* 允许垂直调整大小 */
}
```

```html
<div class="input_container">
  <form>
    <textarea placeholder="分享你的快乐，分担你的忧愁" required></textarea>
    <div class="btn_row">
      <img id="add_img_btn" src="./img/add_images.svg" />
      <button type="submit"><span>发表</span></button>
    </div>
  </form>
</div>
```

```css
/* 按钮行容器 - 包含图标和提交按钮 */
.input_container .btn_row {
    display: flex;                /* flex布局 */
    flex-direction: row;          /* 水平排列 */
    justify-content: end;         /* 右侧对齐 */
    align-items: center;          /* 垂直居中 */
    border-radius: 0 0 10px 10px; /* 底部圆角 */
    padding: 10px 20px 10px 0;  /* 内边距（右对齐） */
    background-color: #191c31;  /* 深蓝背景 */
}
```

```css
/* 按钮图标样式 */
.input_container .btn_row img {
    height: 27px;         /* 固定高度 */
    width: 27px;          /* 固定宽度（保持方形） */
    margin-right: 10px; /* 图标与按钮间距 */
}
```




```html
<div class="input_container">
  <form>
    <textarea placeholder="分享你的快乐，分担你的忧愁" required></textarea>
    <div class="btn_row">
      <img id="add_img_btn" src="./img/add_images.svg" />
      <button type="submit"><span>发表</span></button>
    </div>
  </form>
</div>
```

```css
/* 主要操作按钮样式 */
.input_container .btn_row button {
    border: 0;            /* 无边框 */
    color: #fff;          /* 白色文字 */
    font-size: 15px;      /* 标准字号 */
    padding: 5px 20px 5px 20px;  /* 内边距 */
    border-radius: 7px;  /* 圆角效果 */
    background-image: linear-gradient(to left, #133260, #1f65ba); /* 蓝色渐变 */
}
```

```css
/* 搜索区域容器 - 与输入区样式统一 */
.search_container {
    margin-top: 50px;        /* 与上方元素间距 */
    border-radius: 10px;     /* 统一圆角 */
    margin-right: 20px;      /* 右侧留白 */
    margin-left: 20px;       /* 左侧留白 */
    background: #22222f;     /* 中灰背景 */
}
```

```css
/* 搜索表单布局 */
.search_container>form {
    display: flex;           /* flex布局 */
    flex-direction: row;     /* 水平排列 */
    justify-content: end;    /* 内容右对齐 */
    align-items: center;     /* 垂直居中 */
    width: 100%;             /* 撑满容器 */
}
```

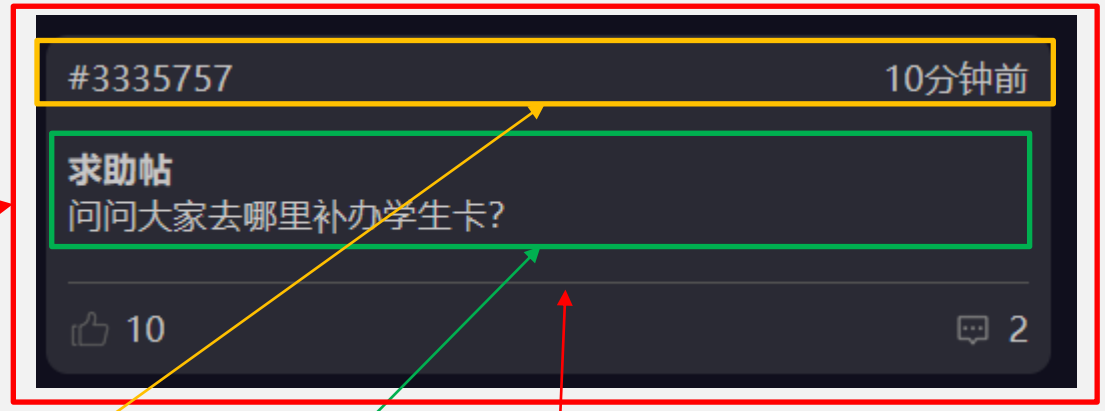

```css
/* 搜索按钮样式 */
.search_container button {
    border: 0;               /* 无边框 */
    color: #fff;             /* 白色文字 */
    font-size: 15px;         /* 标准字号 */
    padding: 5px 20px 5px 20px;  /* 内边距 */
    border-radius: 0 10px 10px 0; /* 右侧圆角 */
    background-image: linear-gradient(to left, #133260, #3372c0); /* 浅蓝渐变 */
    padding: 10px;           /* 加大内边距 */
}
```

```css
/* 搜索输入框样式 */
.search_container input {
    border: 0;                    /* 无边框 */
    background: #ffffff00;        /* 透明背景 */
    color: #c8c6c6;               /* 文字颜色 */
    font-size: 18px;              /* 中等字号 */
    padding: 10px 0 10px 0;       /* 垂直内边距 */
    width: 80%;                   /* 输入框占比 */
}
```

```css
/* 单个帖子容器 */
.post_container {
    margin: 15px 20px 15px 20px;  /* 四周留白 */
    background-color: #2a2a34;    /* 深灰背景 */
    border-radius: 10px;          /* 圆角效果 */
    display: flex;                /* flex布局 */
    flex-direction: column;       /* 垂直排列 */
    padding: 10px;                /* 内边距 */
}
```

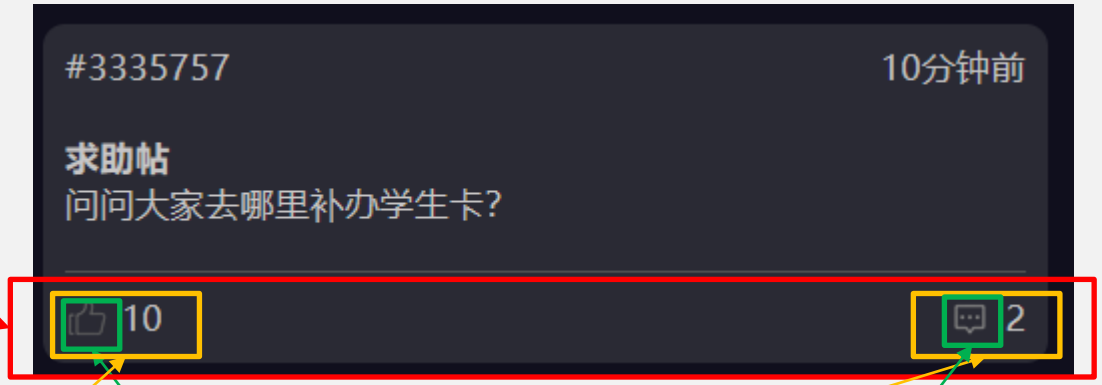
```css
/* 帖子头部布局 - 作者与时间 */
.post_container .post_header {
    display: flex;                /* flex布局 */
    justify-content: space-between; /* 两端对齐 */
}
```

```css
/* 帖子内容区域 */
.post_container .post_content {
    margin: 20px 0 20px 0;  /* 上下边距 */
    display: flex;          /* flex布局 */
    flex-direction: column; /* 垂直排列 */
    align-items: start;     /* 左对齐 */
}
```




```css
/* 分隔线样式 */
.banner {
    height: 1px;              /* 细线高度 */
    width: 100%;              /* 撑满容器 */
    background-color: #535353; /* 中灰色 */
}
```

```css
/* 帖子底部操作区 */
.post_container .post_bottom {
    display: flex;                /* flex布局 */
    flex-direction: row;          /* 水平排列 */
    justify-content: space-between; /* 两端对齐 */
    margin-top: 10px;             /* 顶部间距 */
}
```




```css
/* 点赞/评论按钮通用样式 */
.post_bottom .like,
.message {
    display: flex;                /* flex布局 */
    flex-direction: row;          /* 水平排列 */
    justify-content: center;      /* 居中显示 */
    align-items: center;          /* 垂直居中 */
}
```

```css
/* 操作按钮图标 */
.post_bottom img {
    width: 20px;      /* 图标尺寸 */
    height: 20px;     /* 保持方形 */
    margin-right: 6px; /* 图标与文字间距 */
}
```

JS基础练习

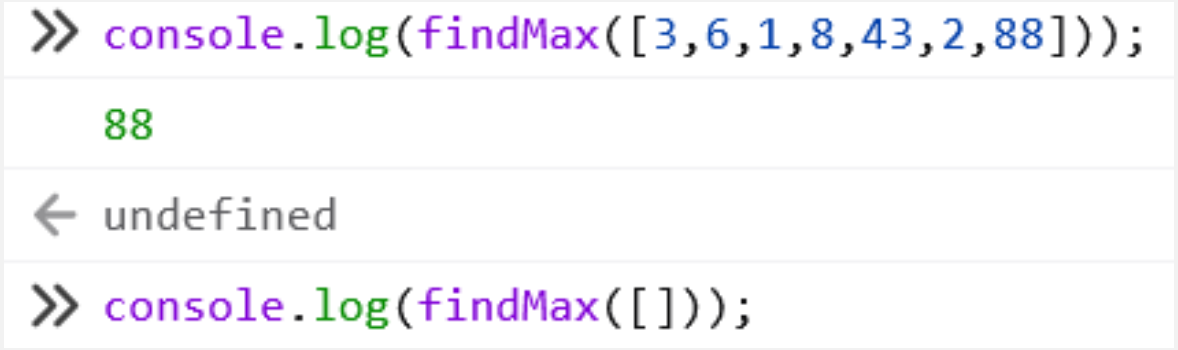
练习1——两数相加

- 请你实现函数addNumbers，该函数有2个参数，整数a和整数b，返回他们的和

```
» function addNumbers(a, b){
      return a+b;
  }
← undefined
» console.log(addNumbers(5, 3))
  8
```

练习2——找出数组中最大值

- 请你实现函数findMax，该函数有1个参数，整数数组arr，返回该数组中最大的元素。如果arr为空，则返回-Infinity



练习2——找出数组中最大值

- 请你实现函数findMax，该函数有1个参数，整数数组arr，返回该数组中最大的元素。如果arr为空，则返回-Infinity

版本1

```
function findMax(arr){
    let tmp = -Infinity;
    let length = arr.length;
      for(let i = 0; i < length; i++){
          if(arr[i] > tmp) tmp = arr[i];
      }
      return tmp;
  }
```

版本2

```
function findMax(arr){
    let tmp = -Infinity;
      arr.forEach(item => {
          if(item > tmp){
              tmp = item;
          }
      });
      return tmp;
  }
```

版本3

```
function findMax(arr){
    return Math.max(...arr);
}
```

arr = [1, 2, 3, 4]

Math.max(...arr)

=> Math.max(1, 2, 3, 4)

用于传参

练习3——回文判断

- 请你实现函数isPalindrome，该函数有1个参数，字符串str，请你判断str是否是回文（正读反读都一样），如果是则返回true，不是则返回false

```
>> console.log(isPalindrome('level'));
   true
<- undefined
>> console.log(isPalindrome('hello'));
   false
```

练习3——回文判断

- 请你实现函数isPalindrome，该函数有1个参数，字符串str，请你判断str是否是回文（正读反读都一样），如果是则返回true，不是则返回false

版本1

```
function isPalindrome(str) {
  let left = 0;
  let right = str.length - 1;
  while(left < right) {
    if(str[left] !== str[right]){
      return false;
    }
    left += 1;
    right -= 1;
  }
  return true;
}
```

版本2

```
function isPalindrome(str) {
    return str === str.split('').reverse().join('');
}
```

```
'hello'.split('')
▶ Array(5) [ "h", "e", "l", "l", "o" ]
'hello'.split('').reverse()
▶ Array(5) [ "o", "l", "l", "e", "h" ]
'hello'.split('').reverse().join('')
"olleh"
```

网页开发——JS进阶

DOM操作+事件响应

目录

JS 事件绑定的几种写法

JS 事件绑定的几种写法（“茴”字的四种写法）

如何学会绑定JS事件很重要，在HTML中使用JavaScript绑定点击事件有以下几种方法:

1. HTML属性方式（内联事件）

```
<button onclick="onClickBtn()">点击我</button>
```

2. DOM属性方式

```
document.getElementById('按钮ID').onclick = function() {
    // 处理点击事件
};
```

3. addEventListener方法

```
let btn = document.getElementById('test-button')
btn.addEventListener('click', function() {
    // 处理点击事件
    alert('Hello, world!');
});
```

JS 事件绑定的几种写法

如何学会绑定JS事件很重要，在HTML中使用JavaScript绑定点击事件有以下几种方法:

4. 委托事件（事件代理）

```
document.body.addEventListener('click', function(e) {
 if(e.target.id === '按钮ID') {
     // 处理点击事件
 }
```

5. JQuery或者是别的库 or 框架

```
$('#按钮ID').click(function() {
    // 处理点击事件
});
```

练习1——JS事件绑定

练习描述

给你一个html代码，请实现相应的JS代码，实现点击按钮弹出对话框，显示“Hello, world!”

```
<body>
    <h1>委托事件（事件代理）</h1>
    <p>点击按钮，会弹出对话框，显示“Hello, world!”</p>
    <button id="test-button">点击我</button>
</body>
<script>
    // TODO: 实现委托事件

</script>
```

练习参考代码

```
<body>
    <h1>HTML属性方式</h1>
    <p>点击按钮，会弹出对话框，显示“Hello, world!”</p>
    <button id="test-button" onclick="onClickBtn()">点击我</button>
</body>
<script>
    function onClickBtn() {
        alert('Hello, world!');
    }
</script>
```

练习参考代码

```
<body>
    <h1>DOM属性方式</h1>
    <p>点击按钮，会弹出对话框，显示“Hello, world!”</p>
    <button id="test-button">点击我</button>
</body>
<script>
    let button = document.getElementById('test-button');
    button.onclick = function () {
        alert('Hello, world!');
    }
</script>
```

练习参考代码

```
<body>
    <h1>addEventListener方法</h1>
    <p>点击按钮，会弹出对话框，显示“Hello, world!”</p>
    <button id="test-button">点击我</button>
</body>
<script>
    let button = document.getElementById('test-button');
    button.addEventListener('click', () => {
        alert('Hello, world!');
    })
</script>
```

练习参考代码

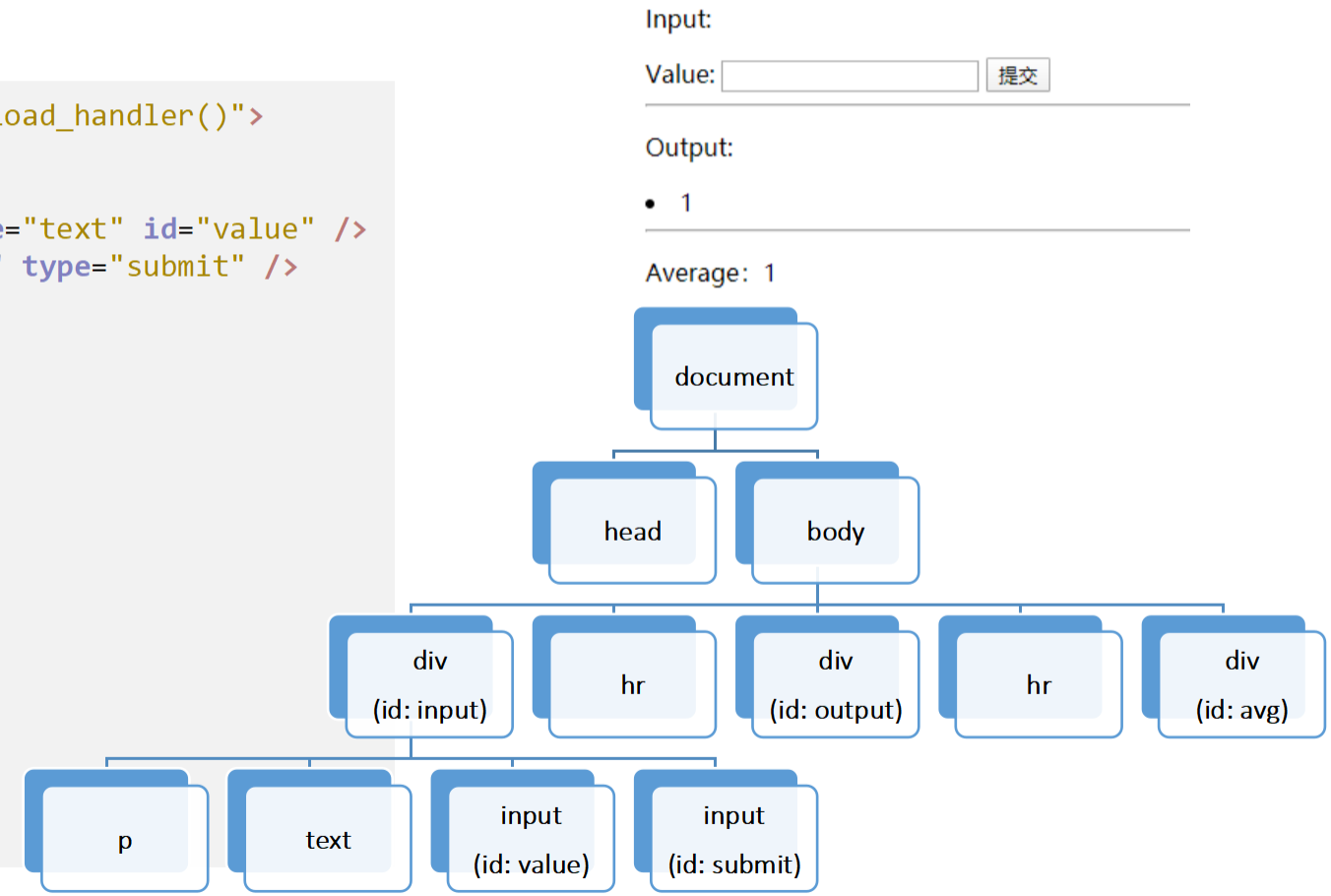
```
<body>
    <h1>委托事件（事件代理）</h1>
    <p>点击按钮，会弹出对话框，显示“Hello, world!”</p>
    <button id="test-button">点击我</button>
</body>
<script>
    document.body.addEventListener('click', function (e) {
        if (e.target.id === 'test-button') {
            alert('Hello, world!');
        }
    });
</script>
```

JS DOM操作回顾

2.4.2 DOM

- 将整个HTML文档描述成一棵树，文档中的每个标签对应DOM树的一个节点

```
<body onload="body_onload_handler()">
  <div id="input">
    <p>Input:</p>
    Value: <input type="text" id="value" />
    <input id="submit" type="submit" />
  </div>

  <hr />

  <div id="output">
    <p>Output:</p>
    <li>1</li>
  </div>

  <hr />

  <div id="avg">
    <p>Average: 1</p>
  </div>
</body>
```



学习更多：https://www.w3school.com.cn/js/js_htmldom.asp

DOM操作

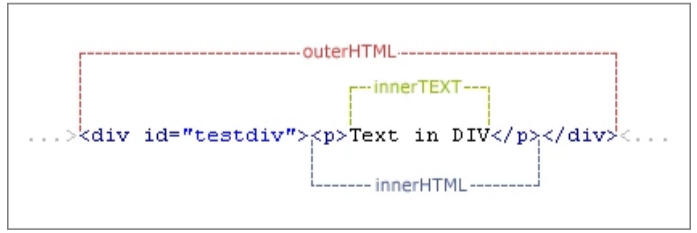
定位输入框：

- document.getElementById("value")
- document.body.childNodes[1].firstChild.nextSibling.nextSibling.nextSibling

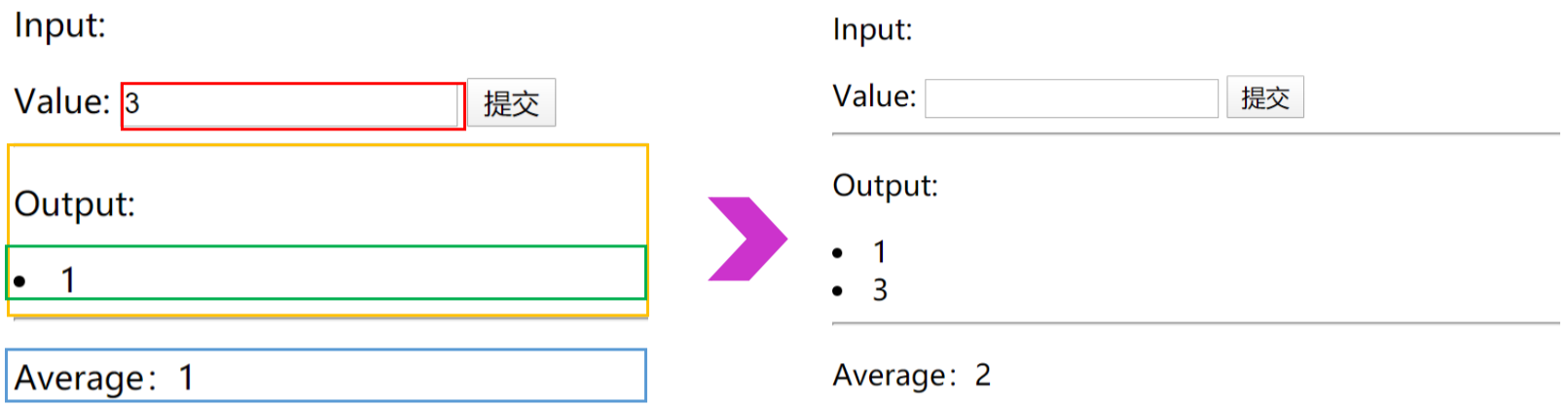
- 获取节点
  - 按id查询：node = document.getElementById(id)
  - 按类名查询：node = document.getElementsByClass(class)
  - 按标签名查询：node = document.getElementsByTagName(tag)
- 节点指针
  - 返回目标节点的所有子节点的列表：node.childNodes
  - 指向在childNodes列表中的第一个节点：node.firstChild
  - 指向在childNodes列表中的最后一个节点：node.lastChild
  - 指向父节点：node.parentNode
  - 指向前一个兄弟节点：node.previousSibling
  - 指向后一个兄弟节点：node.nextSibling
  - 指向这个节点所属的文档：node.ownerDocument

DOM操作

- 节点操作
  - 创建节点：document.createElement(tag)
  - 添加到指定位置：parentNode.appendChild(newNode);
parentNode.insertBefore(newNode, targetNode);
  - 删除节点：parentNode.removeChild(childNode)
  - 替换节点：parentNode.replaceChild(newNode, oldNode)
- 属性操作
  - 获取属性：node.getAttribute(attrName)
  - 设置属性：node.setAttribute(attrName, value)
  - 删除属性：node.removeAttribute(attrName)
- 文本操作
  - innerText：节点内部内容(包括目标元素标签)
  - innerHTML：节点内部内容(不含目标元素标签)
  - outerHTML：含目标元素标签的节点完整内容



DOM操作示例




```
/* 当提交按钮被点击时触发执行onclick_handler()函数 */
function onclick_handler() {
  value = document.getElementById("value");          //找到value输入框对象
  output = document.getElementsByTagName("div")[1];  //找到output区块对象
  li = document.createElement("li");                 //创建li列表对象
  li.innerHTML = value.value;                        //设置li列表对象的内容为value输入框对象的值
  output.appendChild(li);                            //将新创建的li列表对象添加为output区块对象的儿子节点
  value.value = "";                                  //清空value输入框对象的值
}
/* 当output区块插入DOM节点时触发执onDOMNodeInserted_handler函数 */
function onDOMNodeInserted_handler() {
  avg = document.getElementById("avg");              //找到avg区块对象
  lis = document.getElementsByTagName("li");         //找到所有的li列表对象
  sum = 0;                                           //设置变量sum的初始值为0
  for (i = 0; i < lis.length; i++) {                 //遍历每一个li列表对象
    sum += Number(lis[i].innerText);                 //将li列表对象的内容显示转化为数值内容，累加到sum变量
  }
  avg.innerText = "Average: " + sum/lis.length;      //设置avg区块对象的内容为sum与li列表对象个数的商（平均值）
}
```

练习2——获取正确的DOM对象

练习要求

补全代码，使得代码能够在调试控制台输出“测试通过！”

```
<body>
    <div id="test-div">
        <div class="c-red">
            <p id="test-p">JavaScript</p>
            <p>Java</p>
        </div>
        <div class="c-red c-green">
            <p>Python</p>
            <p>Ruby</p>
            <p>Swift</p>
        </div>
        <div class="c-green">
            <p>Scheme</p>
            <p>Haskell</p>
        </div>
    </div>
</body>
```

```
<script>
    // 选择<p>JavaScript</p>:
    let js = ???;

    // 选择<p>Python</p>,<p>Ruby</p>,<p>Swift</p>:
    let arr = document.querySelector('.c-red.c-green').
    getElementsByTagName('p');

    // 选择<p>Haskell</p>:
    let haskell = ???;

    // 测试:
    if (!js || js.innerText !== 'JavaScript') {
        alert('选择JavaScript失败!');
    } else if (!arr || arr.length !== 3 || !arr[0] || !arr[1] || !arr
    [2] || arr[0].innerText !== 'Python' || arr[1].innerText !==
    'Ruby' || arr[2].innerText !== 'Swift') {
        console.log('选择Python,Ruby,Swift失败!');
    } else if (!haskell || haskell.innerText !== 'Haskell') {
        console.log('选择Haskell失败!');
    } else {
        console.log('测试通过!');
    }
</script>
```

练习参考代码

```
<body>
    <div id="test-div">
        <div class="c-red">
            <p id="test-p">JavaScript</p>
            <p>Java</p>
        </div>
        <div class="c-red c-green">
            <p>Python</p>
            <p>Ruby</p>
            <p>Swift</p>
        </div>
        <div class="c-green">
            <p>Scheme</p>
            <p>Haskell</p>
        </div>
    </div>
</body>
```

```
// 选择<p>JavaScript</p>:
let js = document.getElementById('test-p');
```

```
» document.getElementById('test-p')
← <p id="test-p">
```

练习参考代码

```
<body>
    <div id="test-div">
        <div class="c-red">
            <p id="test-p">JavaScript</p>
            <p>Java</p>
        </div>
        <div class="c-red c-green">
            <p>Python</p>
            <p>Ruby</p>
            <p>Swift</p>
        </div>
        <div class="c-green">
            <p>Scheme</p>
            <p>Haskell</p>
        </div>
    </div>
</body>
```

```
// 选择<p>Python</p>,<p>Ruby</p>,<p>Swift</p>:
let arr = document.querySelector('.c-red.c-green').
getElementsByTagName('p');
```

```
» document.querySelector('.c-red.c-green')
← ▶ <div class="c-red c-green">
» document.querySelector('.c-red.c-green').getElementsByTagName('p')
← ▼ HTMLCollection { 0: p , 1: p , 2: p , length: 3 }
    ▶ 0: <p>
    ▶ 1: <p>
    ▶ 2: <p>
      length: 3
```

练习参考代码

```html
<body>
    <div id="test-div">
        <div class="c-red">
            <p id="test-p">JavaScript</p>
            <p>Java</p>
        </div>
        <div class="c-red c-green">
            <p>Python</p>
            <p>Ruby</p>
            <p>Swift</p>
        </div>
        <div class="c-green">
            <p>Scheme</p>
            <p>Haskell</p>
        </div>
    </div>
</body>
```

```
// 选择<p>Haskell</p>:
let haskell = document.getElementsByClassName('c-green')[1].
getElementsByTagName('p')[1];
```

```
>> document.getElementsByClassName('c-green')
<- ▶ HTMLCollection { 0: div.c-red.c-green , 1: div.c-green , length: 2 }
>> document.getElementsByClassName('c-green')[1]
<- ▶ <div class="c-green">
>> document.getElementsByClassName('c-green')[1].getElementsByTagName('p')
<- ▶ HTMLCollection { 0: p , 1: p , length: 2 }
>> document.getElementsByClassName('c-green')[1].getElementsByTagName('p')[1]
<- ▶ <p>
>> document.getElementsByClassName('c-green')[1].getElementsByTagName('p')[1].innerHTML
<- "Haskell"
```

练习3——删除DOM元素

练习要求

补全代码，使得代码能够在调试控制台输出“测试通过！”

```html
<body>
    <p>把不属于人大的地点删除</p>
    <ul id="test-list">
        <li>一勺池</li>
        <li>世纪馆</li>
        <li>未名湖</li>
        <li>明德楼</li>
        <li>汇贤大厦</li>
        <li>乾清宫</li>
    </ul>
</body>
```

```html
<script>
    // TODO


    // 测试:
    let test_func = function () {
        let
            arr, i,
            t = document.getElementById('test-list');
        if (t && t.children && t.children.length === 3) {
            arr = [];
            for (i = 0; i < t.children.length; i++) {
                arr.push(t.children[i].innerText);
            }
            if (arr.toString() === ['一勺池', '世纪馆', '明德楼', '汇贤大厦'].toString()) {
                console.log('测试通过!');
            }
            else {
                console.log('测试失败: ' + arr.toString());
            }
        }
        else {
            console.log('测试失败!');
        }
    }
    test_func();

</script>
```

练习参考代码

```html
<body>
    <p>把不属于人大的地点删除</p>
    <ul id="test-list">
        <li>一勺池</li>
        <li>世纪馆</li>
        <li>未名湖</li>
        <li>明德楼</li>
        <li>汇贤大厦</li>
        <li>乾清宫</li>
    </ul>
</body>
```

```javascript
let list = document.getElementById('test-list');
let items = list.children;
// 从后向前遍历，因为删除元素会改变数组长度
for (let i = items.length - 1; i >= 0; i--) {
    let item = items[i];
    let text = item.innerText;
    // 如果地点不在人大地点列表中，则删除
    if (text !== '一勺池' && text !== '世纪馆' &&
    text !== '明德楼' && text !== '汇贤大厦') {
        list.removeChild(item);
    }
}
```

```
>> document.getElementById('test-list').children
← HTMLCollection { 0: li , 1: li , 2: li , 3: li , 4: li , 5: li , length: 6 }
    0: <li>
    1: <li>
    2: <li>
    3: <li>
    4: <li>
    5: <li>
    length: 6
```

练习参考代码

```html
<body>
    <p>把不属于人大的地点删除</p>
    <ul id="test-list">
        <li>一勺池</li>
        <li>世纪馆</li>
        <li>未名湖</li>
        <li>明德楼</li>
        <li>汇贤大厦</li>
        <li>乾清宫</li>
    </ul>
</body>
```

```javascript
let list = document.getElementById('test-list');
let items = list.children;
// 从后向前遍历，因为删除元素会改变数组长度
for (let i = items.length - 1; i >= 0; i--) {
    let item = items[i];
    let text = item.innerText;
    // 如果地点不在人大地点列表中，则删除
    if (text !== '一勺池' && text !== '世纪馆' &&
    text !== '明德楼' && text !== '汇贤大厦') {
        list.removeChild(item);
    }
}
```

```
>> document.getElementById('test-list').children[0].innerText
← "一勺池"
```

示例1——微人大登录界面（动态）

如何实现验证码每一次都不一样？如何实现登录验证功能？
这里我们需要用到JS来实现这一“动态”效果

```
// 页面加载完成时执行初始化函数
window.onload = body_onload_handler;
```

```
/**
 * 页面加载完成时的初始化函数
 * 注册按钮点击事件和验证码图片点击事件
 */
function body_onload_handler() { //页面加载时触发
    // 获取登录按钮并绑定点击事件
    button = document.getElementById("login-submit");
    button.onclick = login; //注册登录按钮的鼠标点击事件处理函数

    // 获取验证码图片元素并绑定点击事件
    img = document.getElementById("captcha-img").childNodes[0];
    img.onclick = getYzm; //注册验证码图片的鼠标点击事件处理函数
    img.click(); // 触发一次点击事件，生成初始验证码
}
```

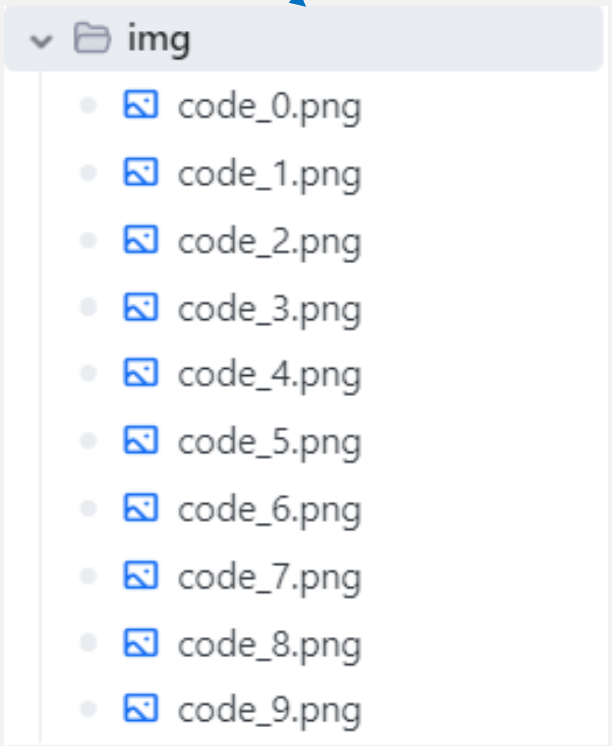
```
<div><button class="button" id="login-submit" type="submit"><span>登录</span></button></div>
```

```
<div class="input-group" id="captcha-img"><img/></div>
```

window.onload是一个回调函数，每当页面加载时都会调用这个函数，因此我们可以重载该函数来执行初始化操作。

body_onload_handler会进行事件的绑定，这里绑定了登录按钮点击事件login和验证码图片点击事件getYzm。

```javascript
/**
 * 生成随机验证码图片
 * @param {Event} e - 点击事件对象
 */
function getYzm(e) { //随机生成一个0-9的整数x，使用图片code_x.png作为验证码图片
    x = Math.floor(Math.random() * 10); // 生成0-9的随机数
    img = e.target; // 获取被点击的图片元素
    img.src = "./img/code_" + x + ".png"; // 设置验证码图片路径
}
```



```html
<div class="input-group" id="captcha-img"><img /></div>
<div class="alert" id="login-alert">
  <p id="captcha-info">验证码只包含字母,不区分大小写</p>
</div>
<div><button class="button" id="login-submit" type="submit"><span>登录</span></button></div>
```

```javascript
/**
 * 处理登录逻辑
 * 验证用户输入的验证码、用户名和密码是否正确
 */
function login(e) {
    // 阻止表单默认提交
    e.preventDefault();
    //登录时检验验证码/用户名/密码
    img = document.getElementById("captcha-img").childNodes[0];
    form = document.getElementById("login-form");
    info = document.getElementById("login-alert");

    //维护一个验证码正确值和验证码图片下标的映射
    checks = ["pupr", "khrb", "huks", "egwb", "ekdv", "khns", "wuxc", "tcyk", "nupf", "brpk"];
    // 根据图片路径最后的数字获取对应的正确验证码
    correct_code = checks[img.src.charAt(img.src.length - 5)];

    // 获取用户输入的值
    username = form.username.value;
    password = form.password.value;
    code = form.code.value;

    // 验证逻辑
    if (code != correct_code) { info.innerText = "验证码不正确或已失效,请重试! "; }
    else if (username != "abcd" || password != "1234") { info.innerText = "用户名或密码不正确,请重试! "; }
    else { info.innerText = "验证成功! "; }
}
```

示例2——RUC小喇叭界面（动态）

我们先只考虑实现RUC小喇叭的发帖子的功能，简单来说，页面中主要有3个需要绑定事件的按钮——添加图片，发表按钮，搜索按钮

为「添加图片」绑定事件

因为添加图片比较复杂，我们简单的小页面暂时还不需要支持。所以当用户点击添加图片按钮时，我们就弹出一个提示框。

为「添加图片」绑定事件

```
<div class="input_container">
  <form id="input_form">
    <textarea id="input_textarea"  placeholder="分享你的快乐,
    <div class="btn_row">
      <img id="add_img_btn" src="./img/add_images.svg" />
       <button type="submit"><span>发表</span></button>
    </div>
  </form>
</div>
```

```
// 未开发功能的提示函数 - 当用户点击未完成的功能时，显示提示信息
function undevelopAlert() {
    alert("该功能正在开发中，请耐心等待...");
}

// 获取添加图片按钮元素，并添加点击事件监听器
const addImgBtn = document.getElementById("add_img_btn")
addImgBtn.addEventListener("click", undevelopAlert)
```

我们使用document.getElementById这个函数获取到「添加图片」这个DOM元素，然后使用addEventListener函数为该DOM元素绑定「点击」事件。当我们点击该DOM元素之后，就会去调用undevelopAlert这个函数，函数会调用alert，弹出提示框。

为「搜索按钮」绑定事件

```html
<div class="search_container">
  <form id="search_form">
    <input type="text"
     placeholder="请输入搜索关键词" required>
    <button type="submit"><span>搜索</span></button>
  </form>
</div>
```

```javascript
// 获取搜索表单元素，并添加提交事件监听器
const searchForm = document.getElementById("search_form")
searchForm.addEventListener("submit", function(event) {
    // 阻止表单的默认提交行为，改为执行我们自定义的动作
    event.preventDefault();
    // 展示未开发的提示
    undevelopAlert();
})
```

搜索功能同样有点复杂，我们先不实现。我们为「表单提交」这一事件绑定一个响应函数，去调用undevelopAlert函数，弹出提示框。不过值得注意的是，搜索框默认是一个表单，表单默认会向action属性指向的地址发送一个表单，现在我们没有后端，所以需要调用event.preventDefault阻止它默认的行为，转而执行我们自定义的行为（即弹出提示框）。

「添加帖子」——构造DOM

```html
<div class="post_container">
  <div class="post_header">
    <div class="post_id"><span>#3335757</span></div>
    <div class="post_time"><span>10分钟前</span></div>
  </div>
  <div class="post_content">
    <h4>求助帖</h4>
    <p>问问大家去哪里补办学生卡？</p>
  </div>
  <div class="banner"></div>
  <div class="post_bottom">
    <div class="like"><img src="./img/like.png" /><span>10</span></div>
    <div class="message"><img src="./img/message.png" /><span>2</span></div>
  </div>
</div>
```

添加帖子功能需要我们使用JS创建DOM元素并插入。以上是一个「帖子卡片」的html代码。

「添加帖子」——构造DOM

```
// 创建最外层的帖子容器div
const postContainer = document.createElement('div');
postContainer.className = 'post_container';
```

```
<div class="post_container">
  <div class="post_header">
    <div class="post_id"><span>#3335757</span></div>
    <div class="post_time"><span>10分钟前</span></div>
  </div>
  <div class="post_content">
    <h4>求助帖</h4>
    <p>问问大家去哪里补办学生卡？</p>
  </div>
  <div class="banner"></div>
  <div class="post_bottom">
    <div class="like"><img src="./img/like.png" /><span>10</span></div>
    <div class="message"><img src="./img/message.png" /><span>2</span></div>
  </div>
</div>
```

我们先调用函数createElement创建最外层的div，并设置其class为post_container

「添加帖子」——构造DOM

```
// 创建帖子头部区域（包含帖子ID和发布时间）
const postHeader = document.createElement('div');
postHeader.className = 'post_header';
postHeader.innerHTML = `
    <div class="post_id"><span>#${postId}</span></div>
    <div class="post_time"><span>${timeText}</span></div>
`;
```

```
// 创建帖子内容区域
const postContent = document.createElement('div');
postContent.className = 'post_content';
postContent.innerHTML = `
    ${content}
`;
```

```
<div class="post_container">
  <div class="post_header">
    <div class="post_id"><span>#3335757</span></div>
    <div class="post_time"><span>10分钟前</span></div>
  </div>
  <div class="post_content">
    <h4>求助帖</h4>
    <p>问问大家去哪里补办学生卡？ </p>
  </div>
  <div class="banner"></div>
  <div class="post_bottom">
    <div class="like"><img src="./img/like.png" /><span>10</span></div>
    <div class="message"><img src="./img/message.png" /><span>2</span></div>
  </div>
</div>
```

调用函数createElement创建头部区域的div，并设置其class为post_header，同时设置该div内部的html代码。`${postID}`是JS中的格式化字符串语法。
帖子内容区域类似

「添加帖子」——构造DOM

```
// 创建分隔线（用于分隔内容和底部信息）
const banner = document.createElement('div');
banner.className = 'banner';
```

```
// 创建帖子底部区域（包含点赞和评论信息）
const postBottom = document.createElement('div');
postBottom.className = 'post_bottom';
postBottom.innerHTML = `
    <div class="like"><img src="./img/like.png" /><span>${likes}</span></div>
    <div class="message"><img src="./img/message.png" /><span>${messages}</span></div>
`;
```

```
<div class="post_container">
  <div class="post_header">
    <div class="post_id"><span>#3335757</span></div>
    <div class="post_time"><span>10分钟前</span></div>
  </div>
  <div class="post_content">
    <h4>求助帖</h4>
    <p>问问大家去哪里补办学生卡？</p>
  </div>
  <div class="banner"></div>
  <div class="post_bottom">
    <div class="like"><img src="./img/like.png" /><span>10</span></div>
    <div class="message"><img src="./img/message.png" /><span>2</span></div>
  </div>
</div>
```

「添加帖子」——构造DOM

```
// 将所有创建的元素按顺序添加到帖子容器中
postContainer.appendChild(postHeader);
postContainer.appendChild(postContent);
postContainer.appendChild(banner);
postContainer.appendChild(postBottom);

// 获取帖子列表容器并将新帖子插入到列表最前面
const postList = document.querySelector('.post_list');
postList.insertBefore(postContainer, postList.firstChild);
```

```
<div class="post_container">
  <div class="post_header">
    <div class="post_id"><span>#3335757</span></div>
    <div class="post_time"><span>10分钟前</span></div>
  </div>
  <div class="post_content">
    <h4>求助帖</h4>
    <p>问问大家去哪里补办学生卡？ </p>
  </div>
  <div class="banner"></div>
  <div class="post_bottom">
    <div class="like"><img src="./img/like.png" /><span>10</span></div>
    <div class="message"><img src="./img/message.png" /><span>2</span></div>
  </div>
</div>
```

最后将我们新建的DOM元素组织起来，放到postContainer中，postContainer再插入到页面中

「添加帖子」——绑定事件

```
// 添加表单提交事件监听器
intputForm.addEventListener("submit", function(event) {
    event.preventDefault(); // 阻止表单的默认提交行为

    // 获取并清理输入内容
    const post_content = inputTextarea.value.trim();
    // 如果内容为空，直接返回
    if(post_content.length === 0 || post_content === null){
        return;
    }

    // 生成新帖子的ID（当前帖子数量 + 1）
    const posts = document.querySelectorAll('.post_container');
    let post_id = posts.length + 1;

    // 创建新帖子并添加到列表中
    createPostContainer(post_id, "刚刚", post_content, 0, 0);
    // 清空输入框
    inputTextarea.value = "";
})
```

```
// ===== 发帖功能相关 =====
// 获取发帖表单和输入框元素
const intputForm = document.getElementById("input_form")
const inputTextarea = document.getElementById("input_textarea")
```

```
<form id="input_form">
  <textarea id="input_textarea"  placeholder="分享你的快乐，分担你的忧愁" required></textarea>
  <div class="btn_row">
    <img id="add_img_btn" src="./img/add_images.svg" />
    <button type="submit"><span>发表</span></button>
  </div>
</form>
```

我们再通过getElementById这个函数获取表单和textarea对象，为表单的提交事件绑定一个响应函数。
函数中会从textarea对象中获取用户输入，接着判空，然后添加到帖子列表中，最后再清空输入框。