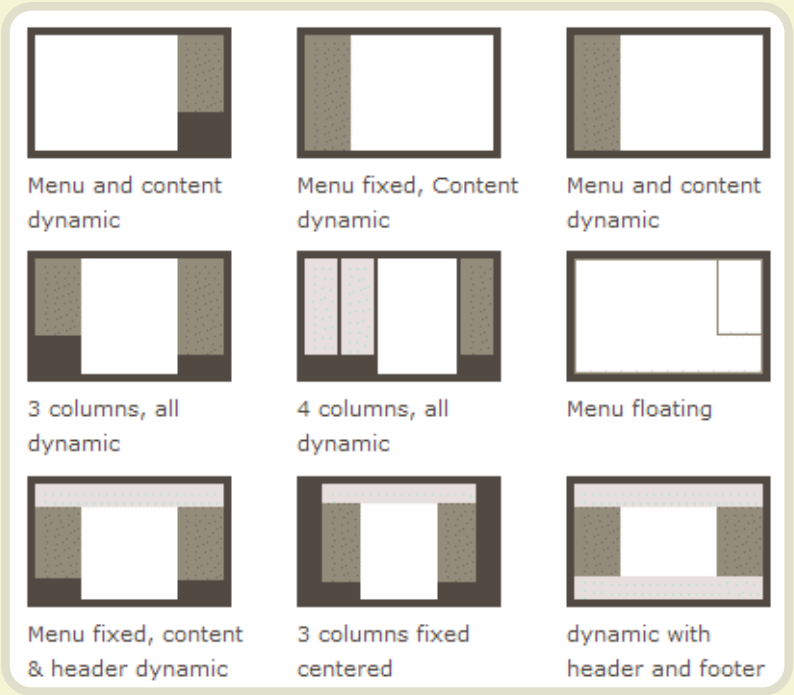


# Flex 布局教程：语法篇

作者： 阮一峰

日期： 2015年7月10日

网页布局（layout）是 CSS 的一个重点应用。



布局的传统解决方案，基于[盒状模型](#)，依赖 `display` 属性 + `position` 属性 + `float` 属性。它对于那些特殊布局非常不方便，比如，[垂直居中](#)就不容易实现。



2009年，W3C 提出了一种新的方案----Flex 布局，可以简便、完整、响应式地实现各种页面布局。目前，它已经得到了所有浏览器的支持，这意味着，现在就能很安全地使用这项功能。

## Browser Support



Chrome  
21+



Opera  
12.1+



Firefox  
22+



Safari  
6.1+



IE  
10+

Flex 布局将成为未来布局的首选方案。本文介绍它的语法，[下一篇文章](#)给出常见布局的 Flex 写法。网友 [JailBreak](#) 为本文的所有示例制作了 [Demo](#)，也可以参考。

以下内容主要参考了下面两篇文章：[A Complete Guide to Flexbox](#) 和 [A Visual Guide to CSS3 Flexbox Properties](#)。

## 一、Flex 布局是什么？

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。

```
.box{  
  display: flex;  
}
```

行内元素也可以使用 Flex 布局。

```
.box{  
  display: inline-flex;  
}
```

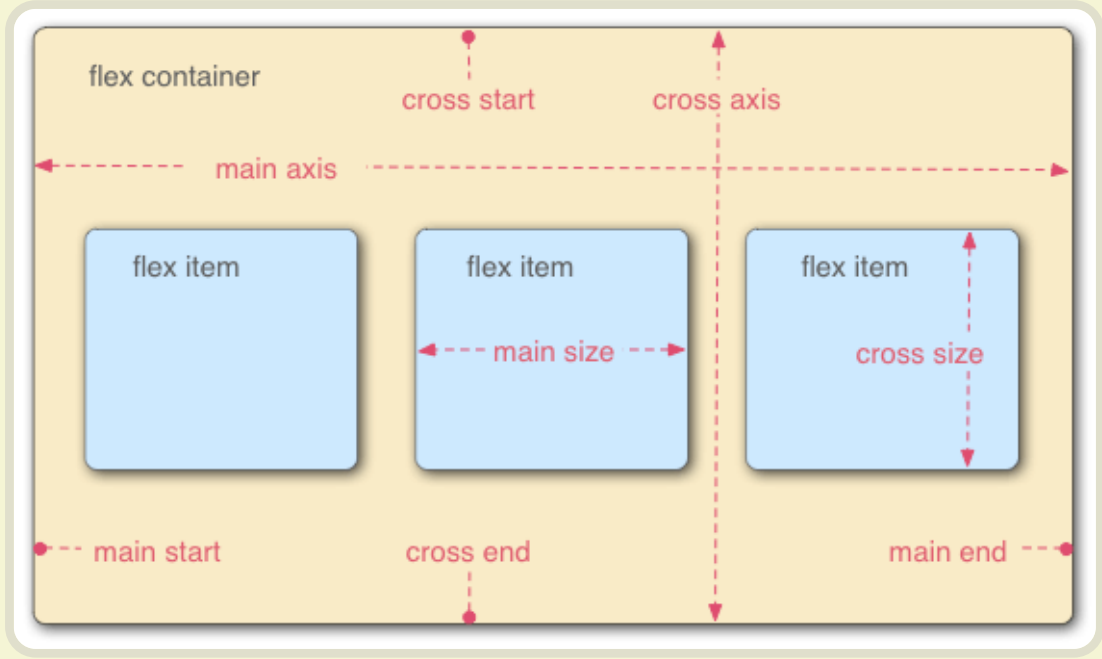
Webkit 内核的浏览器，必须加上 `-webkit` 前缀。

```
.box{  
  display: -webkit-flex; /* Safari */  
  display: flex;  
}
```

注意，设为 Flex 布局以后，子元素的 `float`、`clear` 和 `vertical-align` 属性将失效。

## 二、基本概念

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"。



容器默认存在两根轴：水平的主轴（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`。

### 三、容器的属性

以下6个属性设置在容器上。

- `flex-direction`
- `flex-wrap`
- `flex-flow`
- `justify-content`
- `align-items`
- `align-content`

#### 3.1 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;  
}
```

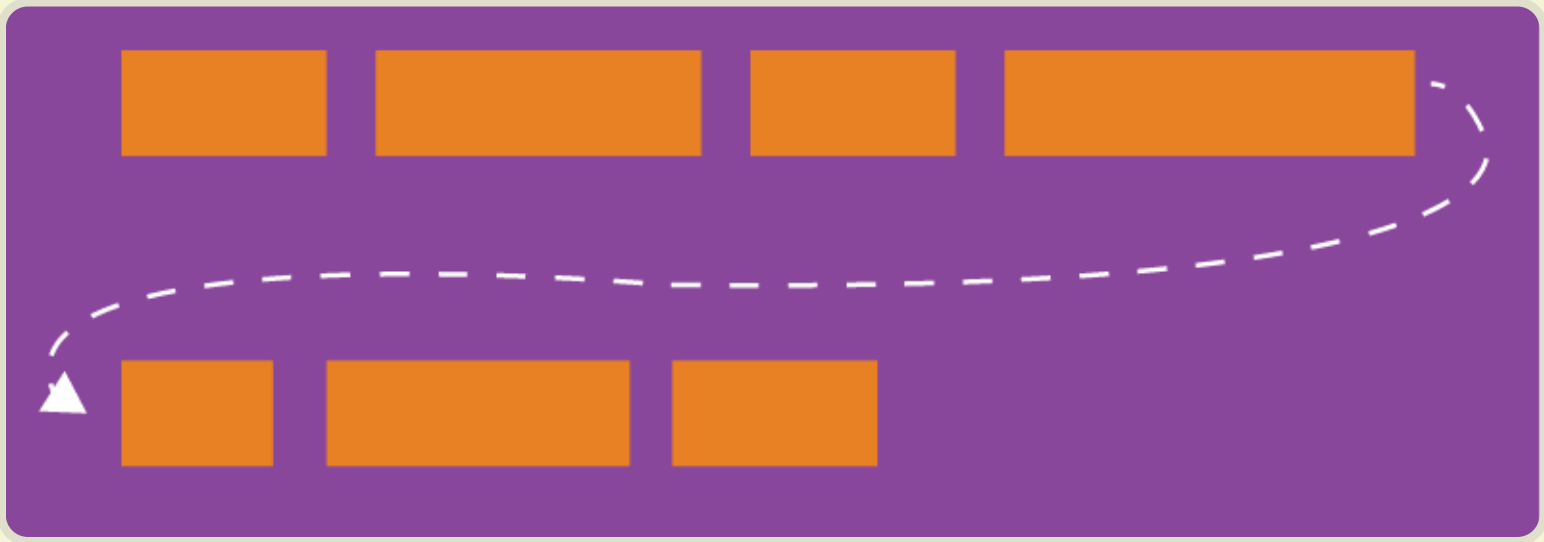


它可能有4个值。

- 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。

### 3.2 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。



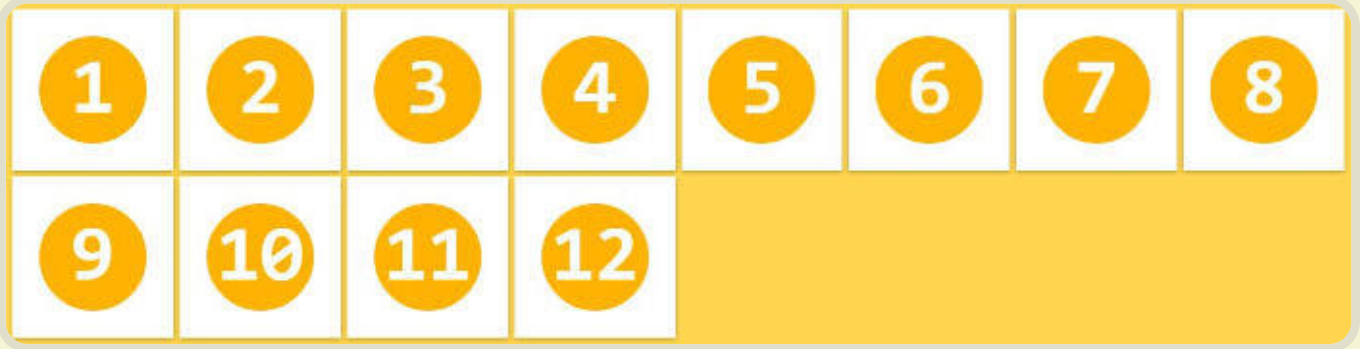
```
.box{  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

它可能取三个值。

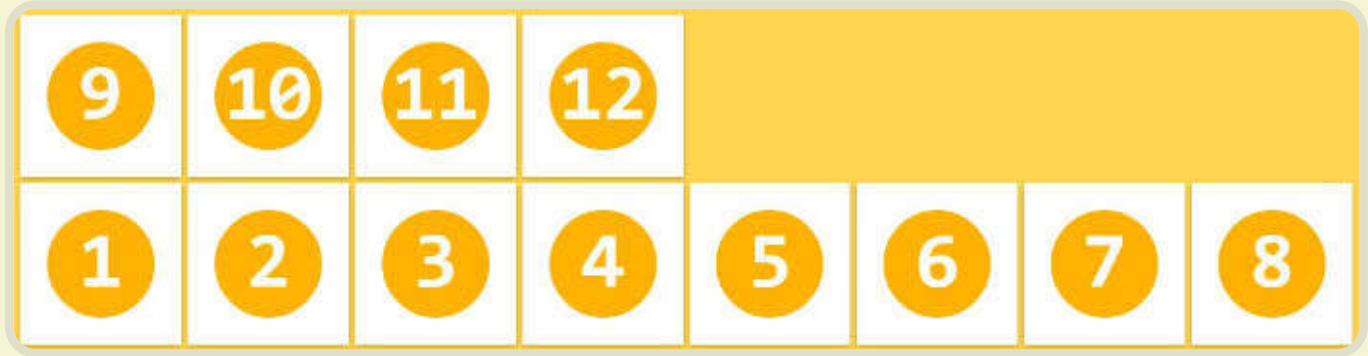
- (1) `nowrap`（默认）： 不换行。



- (2) `wrap`： 换行，第一行在上方。



- (3) `wrap-reverse`： 换行，第一行在下方。



### 3.3 flex-flow

`flex-flow` 属性是 `flex-direction` 属性和 `flex-wrap` 属性的简写形式，默认值为 `row nowrap`。

```
.box {  
  flex-flow: <flex-direction> || <flex-wrap>;  
}
```

### 3.4 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  
}
```

flex-start



flex-end



center



space-between



space-around



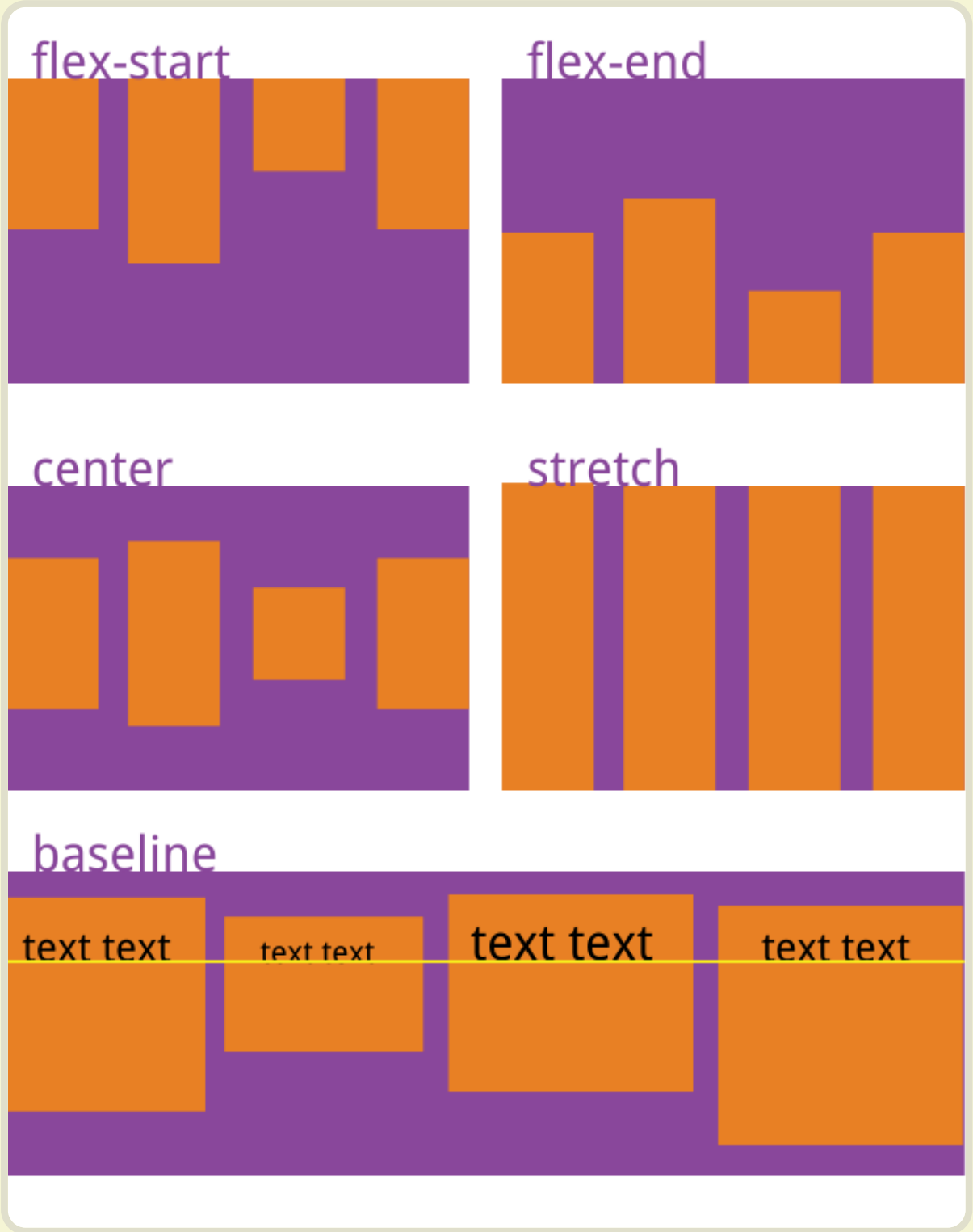
它可能取5个值，具体对齐方式与轴的方向有关。下面假设主轴为从左到右。

- 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。

### 3.5 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 | stre  
}
```



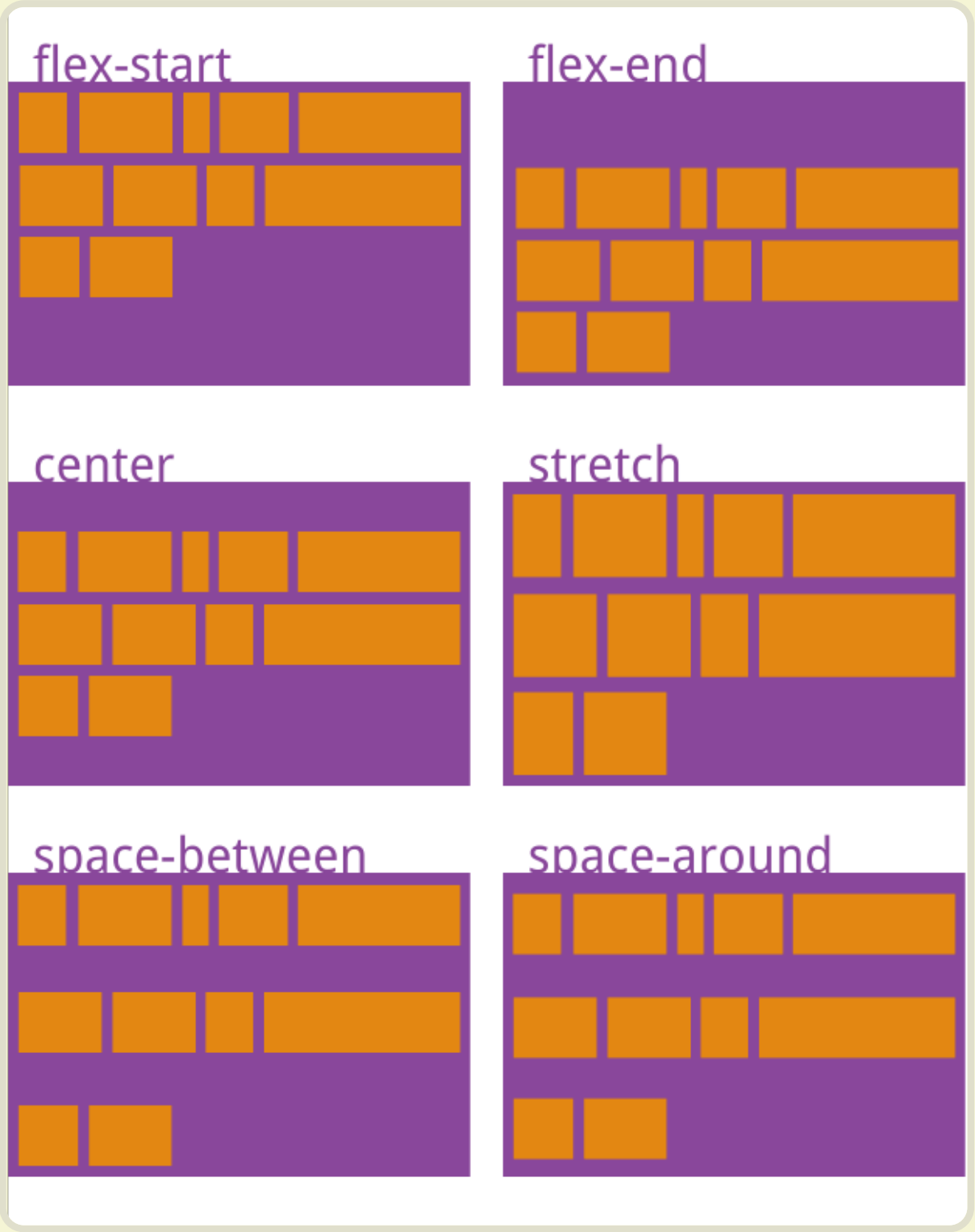
它可能取5个值。具体的对齐方式与交叉轴的方向有关，下面假设交叉轴从上到下。

- 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。

### 3.6 align-content属性

`align-content` 属性定义了对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  
}
```



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。

#### 四、项目的属性

以下6个属性设置在项目上。

- `order`
- `flex-grow`
- `flex-shrink`
- `flex-basis`
- `flex`

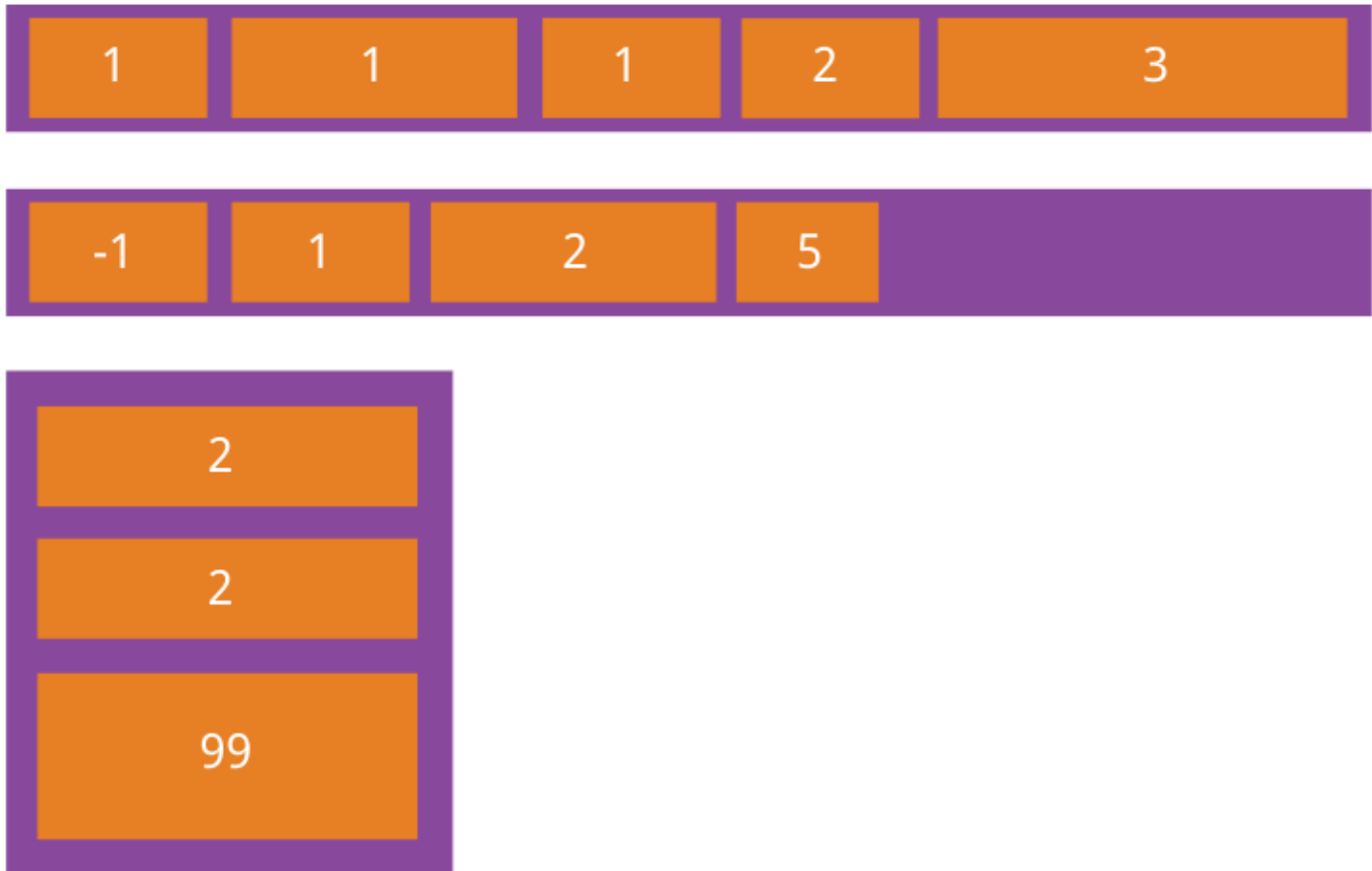


- align-self

#### 4.1 order属性

`order` 属性定义项目的排列顺序。数值越小，排列越靠前，默认为0。

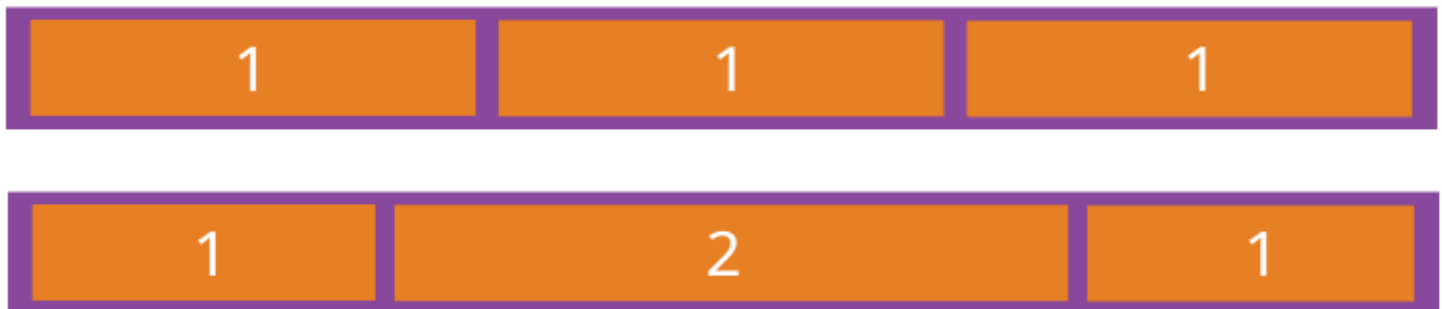
```
.item {  
  order: <integer>;  
}
```



#### 4.2 flex-grow属性

`flex-grow` 属性定义项目的放大比例，默认为 0，即如果存在剩余空间，也不放大。

```
.item {  
  flex-grow: <number>; /* default 0 */  
}
```

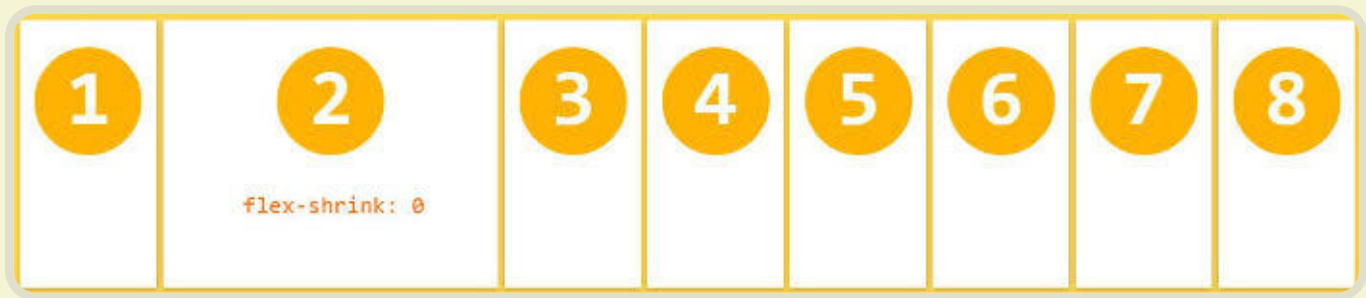


如果所有项目的 `flex-grow` 属性都为1，则它们将等分剩余空间（如果有的话）。如果一个项目的 `flex-grow` 属性为2，其他项目都为1，则前者占据的剩余空间将比其他项多一倍。

#### 4.3 flex-shrink属性

`flex-shrink` 属性定义了项目的缩小比例，默认为1，即如果空间不足，该项目将缩小。

```
.item {
  flex-shrink: <number>; /* default 1 */
}
```



如果所有项目的 `flex-shrink` 属性都为1，当空间不足时，都将等比例缩小。如果一个项目的 `flex-shrink` 属性为0，其他项目都为1，则空间不足时，前者不缩小。

负值对该属性无效。

#### 4.4 flex-basis属性

`flex-basis` 属性定义了分配多余空间之前，项目占据的主轴空间（main size）。浏览器根据这个属性，计算主轴是否有多余空间。它的默认值为 `auto`，即项目的本来大小。

```
.item {
  flex-basis: <length> | auto; /* default auto */
}
```

它可以设为跟 `width` 或 `height` 属性一样的值（比如350px），则项目将占据固定空间。

#### 4.5 flex属性

`flex` 属性是 `flex-grow`，`flex-shrink` 和 `flex-basis` 的简写，默认值为 `0 1 auto`。后两个属性可选。

```
.item {
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
}
```

该属性有两个快捷值： `auto`（`1 1 auto`）和 `none`（`0 0 auto`）。

建议优先使用这个属性，而不是单独写三个分离的属性，因为浏览器会推算相关值。

#### 4.6 align-self属性

`align-self` 属性允许单个项目有与其他项目不一样的对齐方式，可覆盖 `align-items` 属性。默认值为 `auto`，表示继承父元素的 `align-items` 属性，如果没有父元素，则等同于 `stretch`。

```
.item {
  align-self: auto | flex-start | flex-end | center | baseline
}
```

# flex-start



# flex-end

该属性可能取6个值，除了auto，其他都与align-items属性完全一致。

(完)

## 文档信息

- 版权声明： 自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期： 2015年7月10日

## 相关文章

- **2021.05.10:** [软件工程的最大难题](#)
  - 一、引言 大学有一门课程《软件工程》，研究如何组织和管理软件项目。
- **2020.12.13:** [《SSH 入门教程》发布了](#)
  - SSH 是登录 Linux 服务器的必备工具，只要你在做互联网开发，多多少少都会用到它。
- **2020.11.02:** [微信小程序入门教程之四：API 使用](#)
  - 今天是这个系列教程的最后一篇。
- **2020.10.29:** [微信小程序入门教程之三：脚本编程](#)
  - 这个系列教程的前两篇，介绍了小程序的项目结构和页面样式。



Weibo | Twitter | GitHub

Email: [yifeng.ruan@gmail.com](mailto:yifeng.ruan@gmail.com)

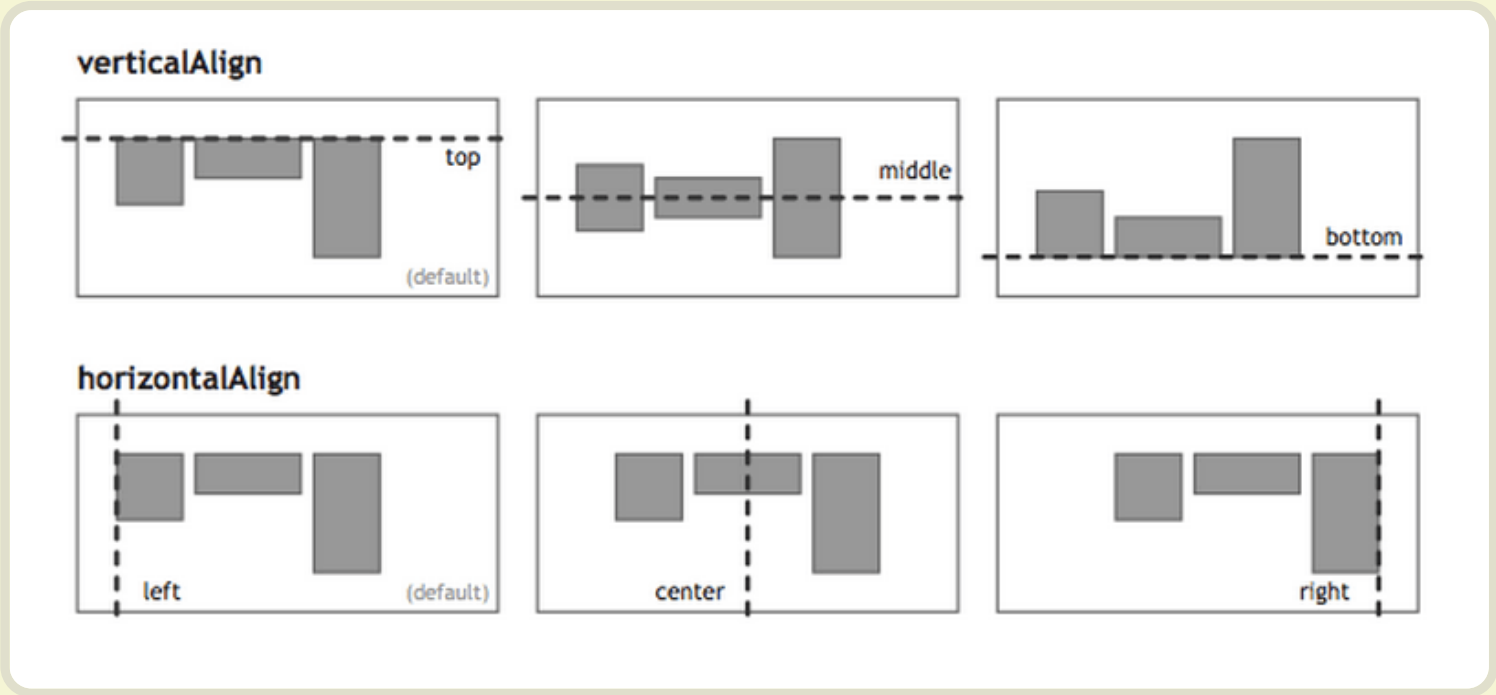
# Flex 布局教程：实例篇

作者： 阮一峰

日期： 2015年7月14日

[上一篇文章](#)介绍了Flex布局的语法，今天介绍常见布局的Flex写法。

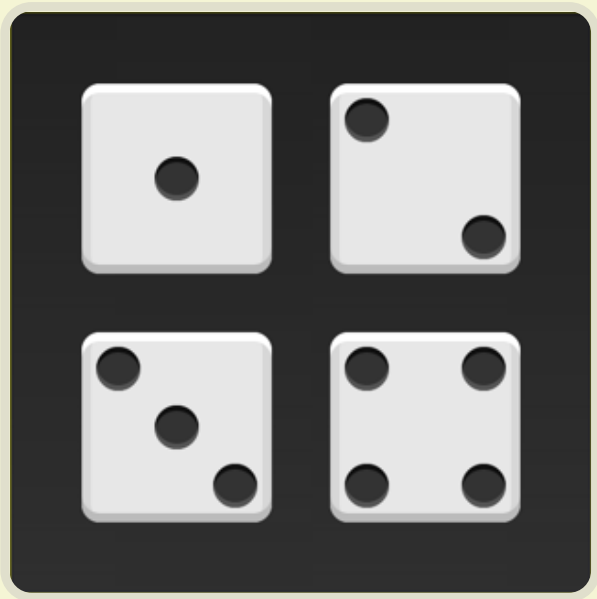
你会看到，不管是什么布局，Flex往往都可以几行命令搞定。



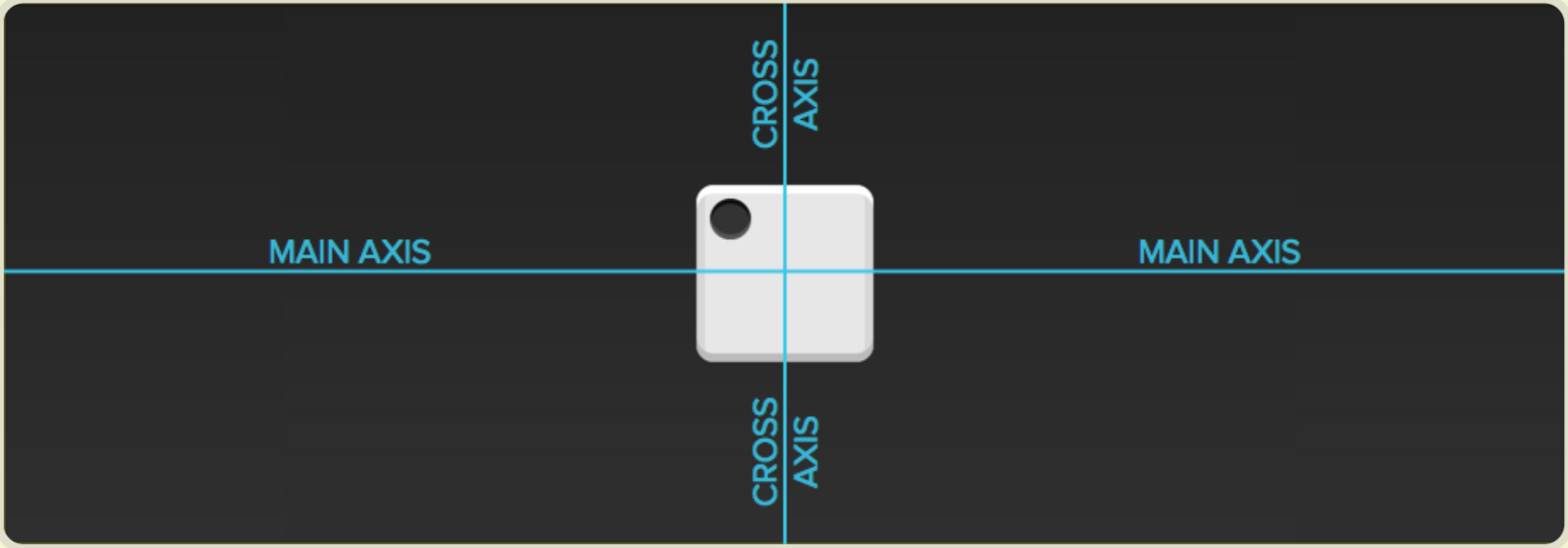
我只列出代码，详细的语法解释请查阅《[Flex布局教程：语法篇](#)》。我的主要参考资料是[Landon Schropp](#)的文章和[Solved by Flexbox](#)。

## 一、骰子的布局

骰子的一面，最多可以放置9个点。



下面，就来看看Flex如何实现，从1个点到9个点的布局。你可以到[codepen](#)查看Demo。



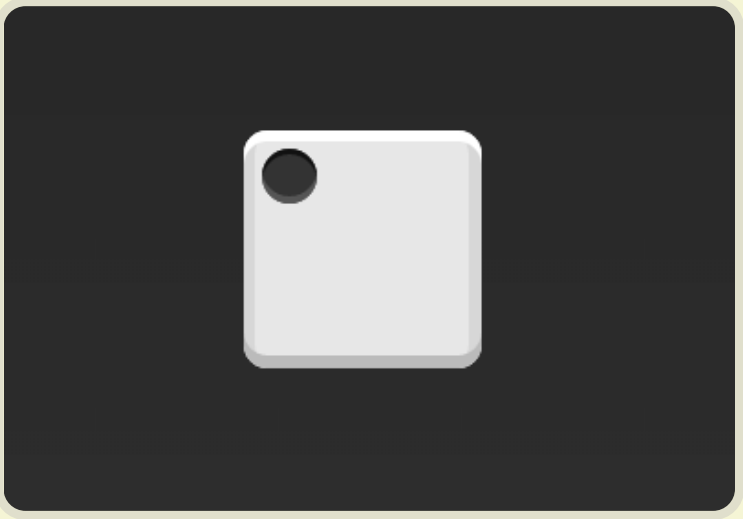
如果不加说明，本节的HTML模板一律如下。

```
<div class="box">
  <span class="item"></span>
</div>
```

上面代码中，div元素（代表骰子的一个面）是Flex容器，span元素（代表一个点）是Flex项目。如果有多个项目，就要添加多个span元素，以此类推。

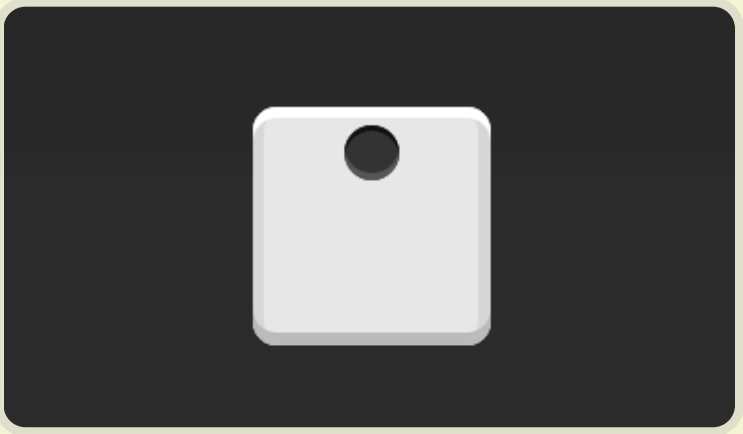
1.1 单项目

首先，只有左上角1个点的情况。Flex布局默认就是首行左对齐，所以一行代码就够了。



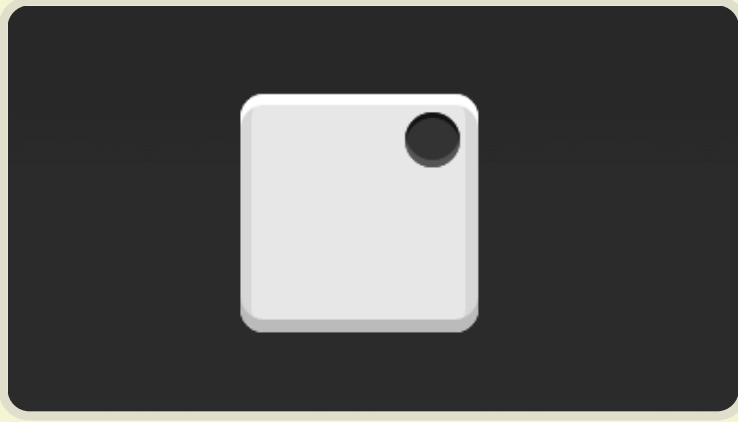
```
.box {
  display: flex;
}
```

设置项目的对齐方式，就能实现居中对齐和右对齐。

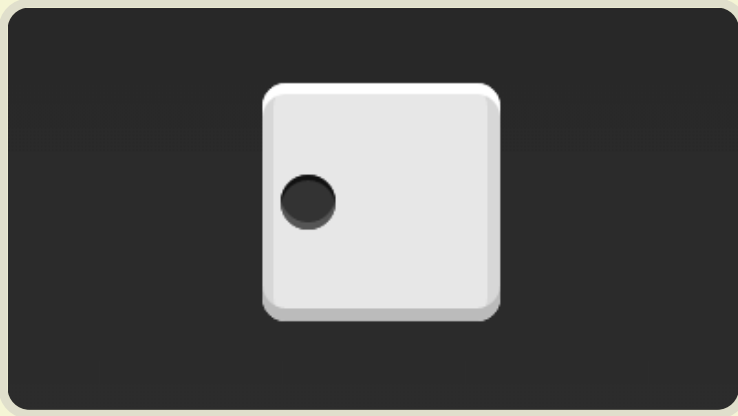


```
.box {
```

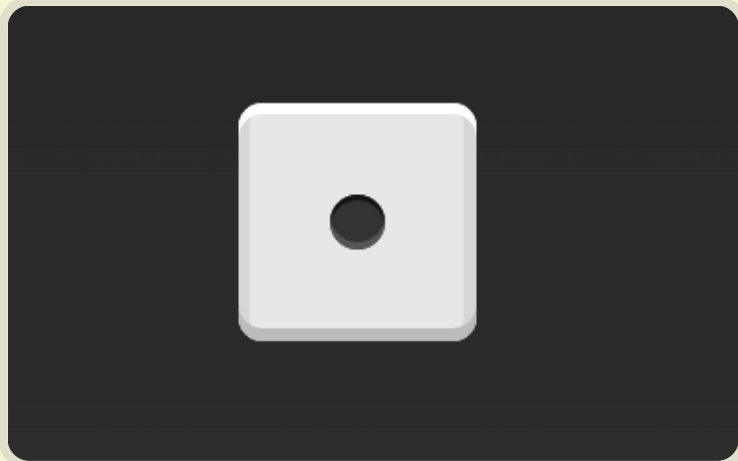
```
display: flex;
justify-content: center;
}
```



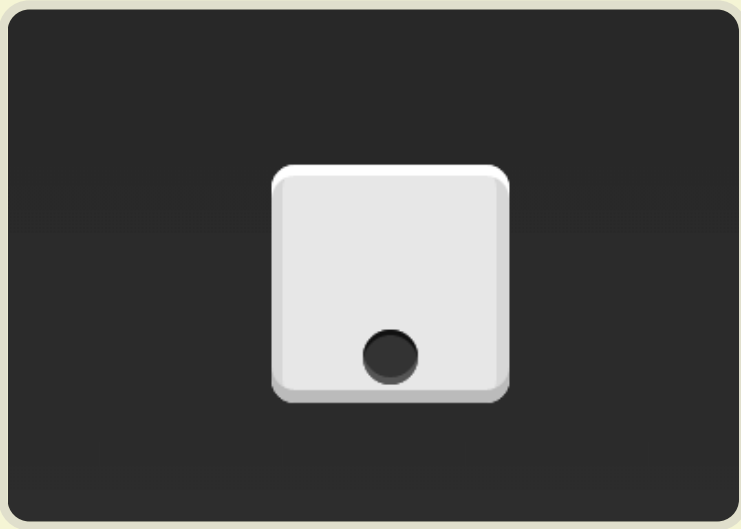
```
.box {
  display: flex;
  justify-content: flex-end;
}
```



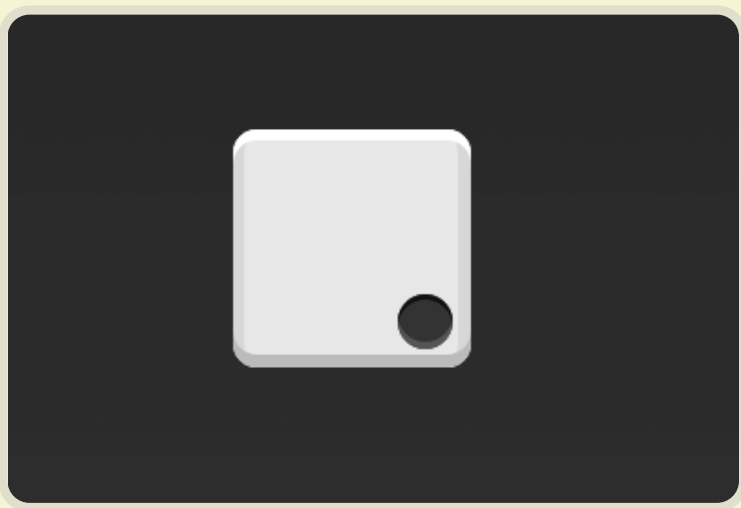
```
.box {
  display: flex;
  align-items: center;
}
```



```
.box {
  display: flex;
  justify-content: center;
  align-items: center;
}
```



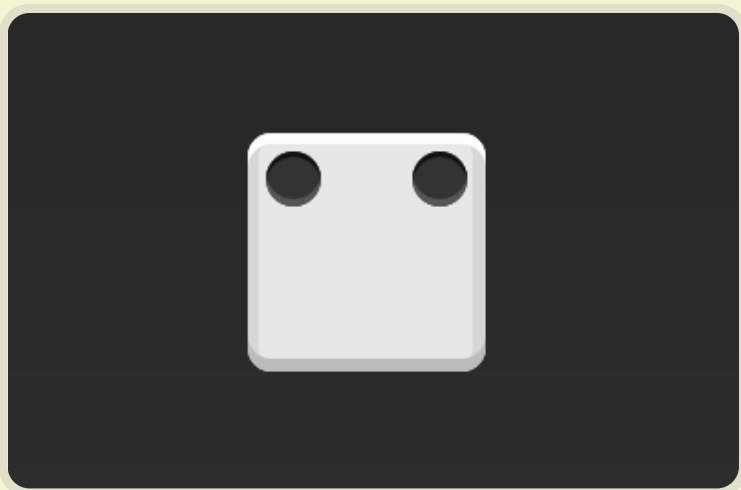
```
.box {  
  display: flex;  
  justify-content: center;  
  align-items: flex-end;  
}
```



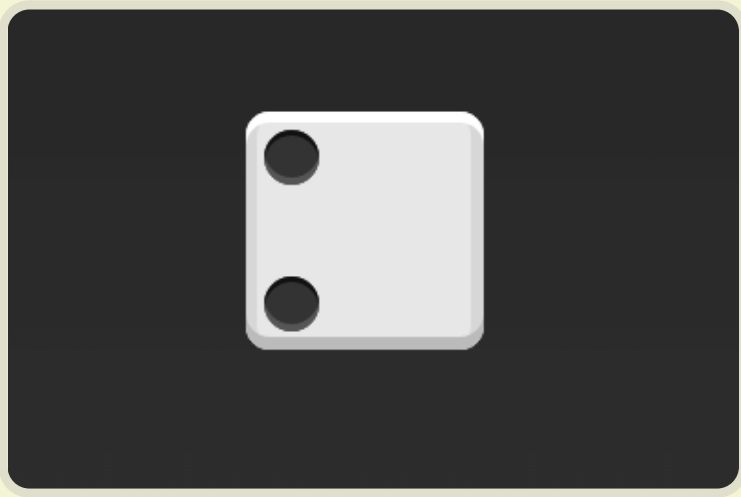
```
.box {  
  display: flex;  
  justify-content: flex-end;  
  align-items: flex-end;  
}
```

## 1.2 双项目

---



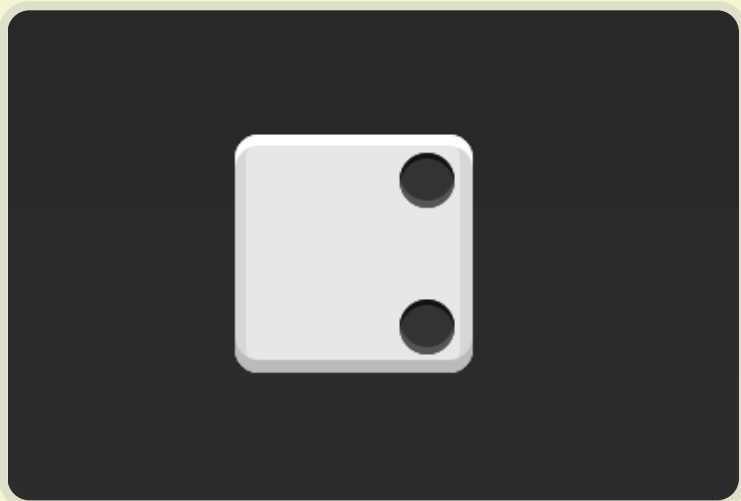
```
.box {  
  display: flex;  
  justify-content: space-between;  
}
```



```
.box {  
  display: flex;  
  flex-direction: column;  
  justify-content: space-between;  
}
```

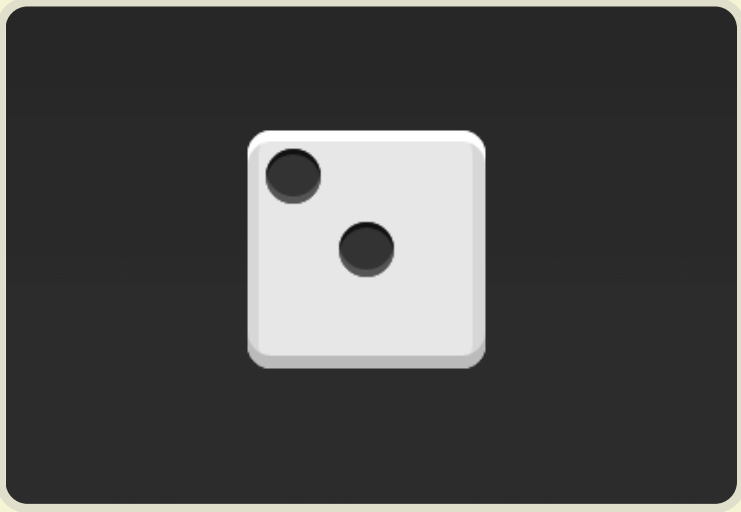


```
.box {  
  display: flex;  
  flex-direction: column;  
  justify-content: space-between;  
  align-items: center;  
}
```

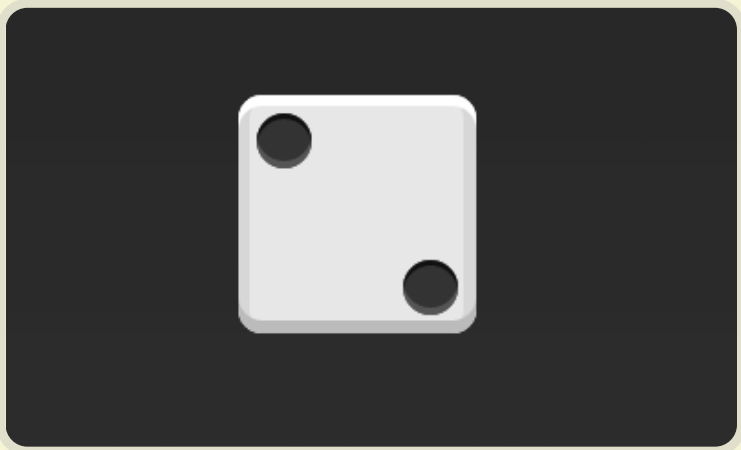


```
.box {  
  display: flex;  
  flex-direction: column;  
  justify-content: space-between;  
  align-items: flex-end;  
}
```





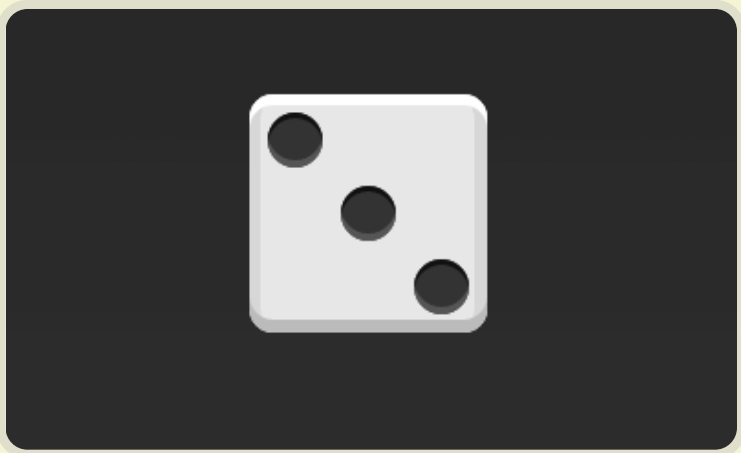
```
.box {  
  display: flex;  
}  
  
.item:nth-child(2) {  
  align-self: center;  
}
```



```
.box {  
  display: flex;  
  justify-content: space-between;  
}  
  
.item:nth-child(2) {  
  align-self: flex-end;  
}
```

### 1.3 三项目

---

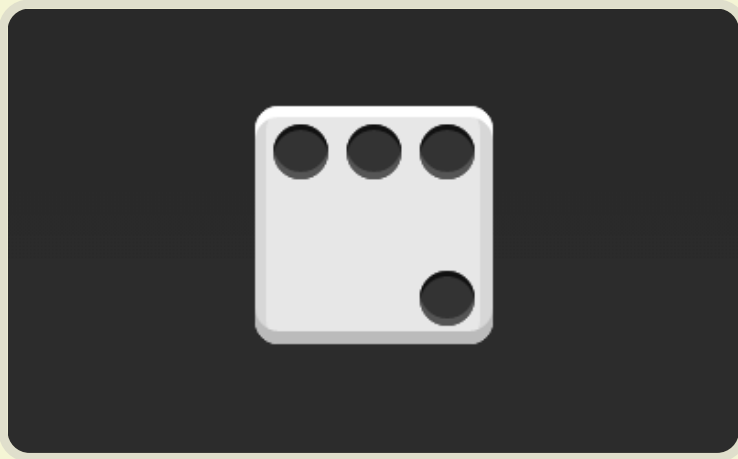


```
.box {  
  display: flex;  
}  
  
.item:nth-child(2) {
```

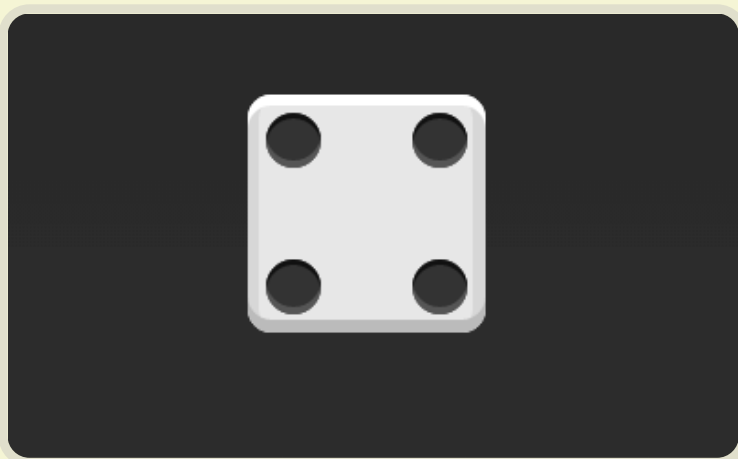
```
align-self: center;
}

.item:nth-child(3) {
  align-self: flex-end;
}
```

#### 1.4 四项目



```
.box {
  display: flex;
  flex-wrap: wrap;
  justify-content: flex-end;
  align-content: space-between;
}
```



HTML代码如下。

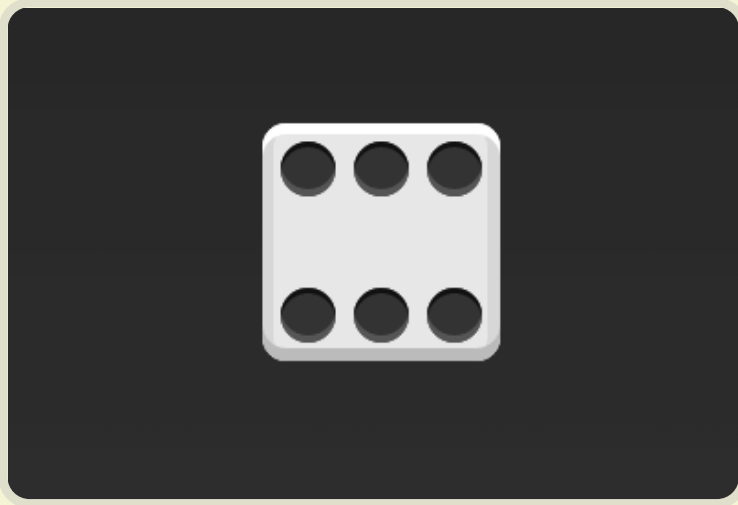
```
<div class="box">
  <div class="column">
    <span class="item"></span>
    <span class="item"></span>
  </div>
  <div class="column">
    <span class="item"></span>
    <span class="item"></span>
  </div>
</div>
```

CSS代码如下。

```
.box {
  display: flex;
  flex-wrap: wrap;
  align-content: space-between;
}
```

```
.column {  
  flex-basis: 100%;  
  display: flex;  
  justify-content: space-between;  
}
```

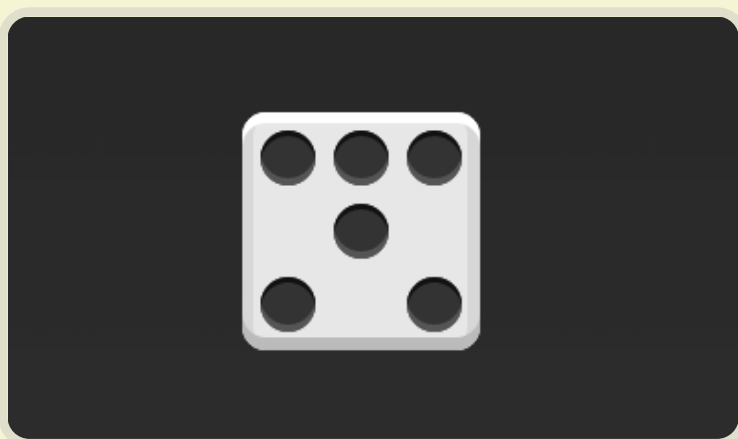
## 1.5 六项目



```
.box {  
  display: flex;  
  flex-wrap: wrap;  
  align-content: space-between;  
}
```



```
.box {  
  display: flex;  
  flex-direction: column;  
  flex-wrap: wrap;  
  align-content: space-between;  
}
```



HTML代码如下。

```
<div class="box">
  <div class="row">
    <span class="item"></span>
    <span class="item"></span>
    <span class="item"></span>
  </div>
  <div class="row">
    <span class="item"></span>
  </div>
  <div class="row">
    <span class="item"></span>
    <span class="item"></span>
  </div>
</div>
```

CSS代码如下。

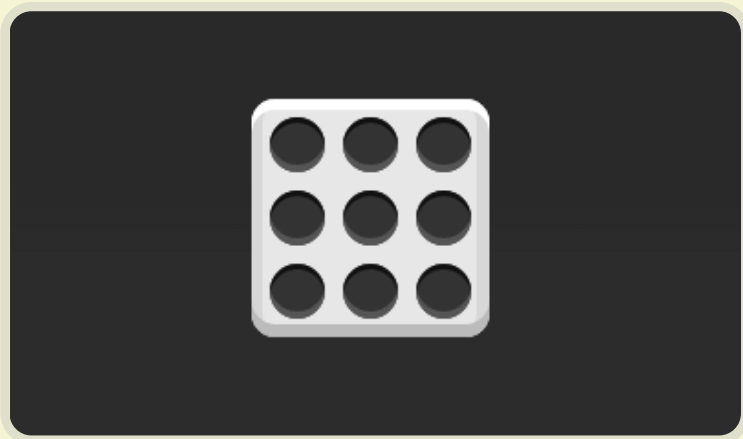
```
.box {
  display: flex;
  flex-wrap: wrap;
}

.row{
  flex-basis: 100%;
  display: flex;
}

.row:nth-child(2){
  justify-content: center;
}

.row:nth-child(3){
  justify-content: space-between;
}
```

## 1.6 九项目

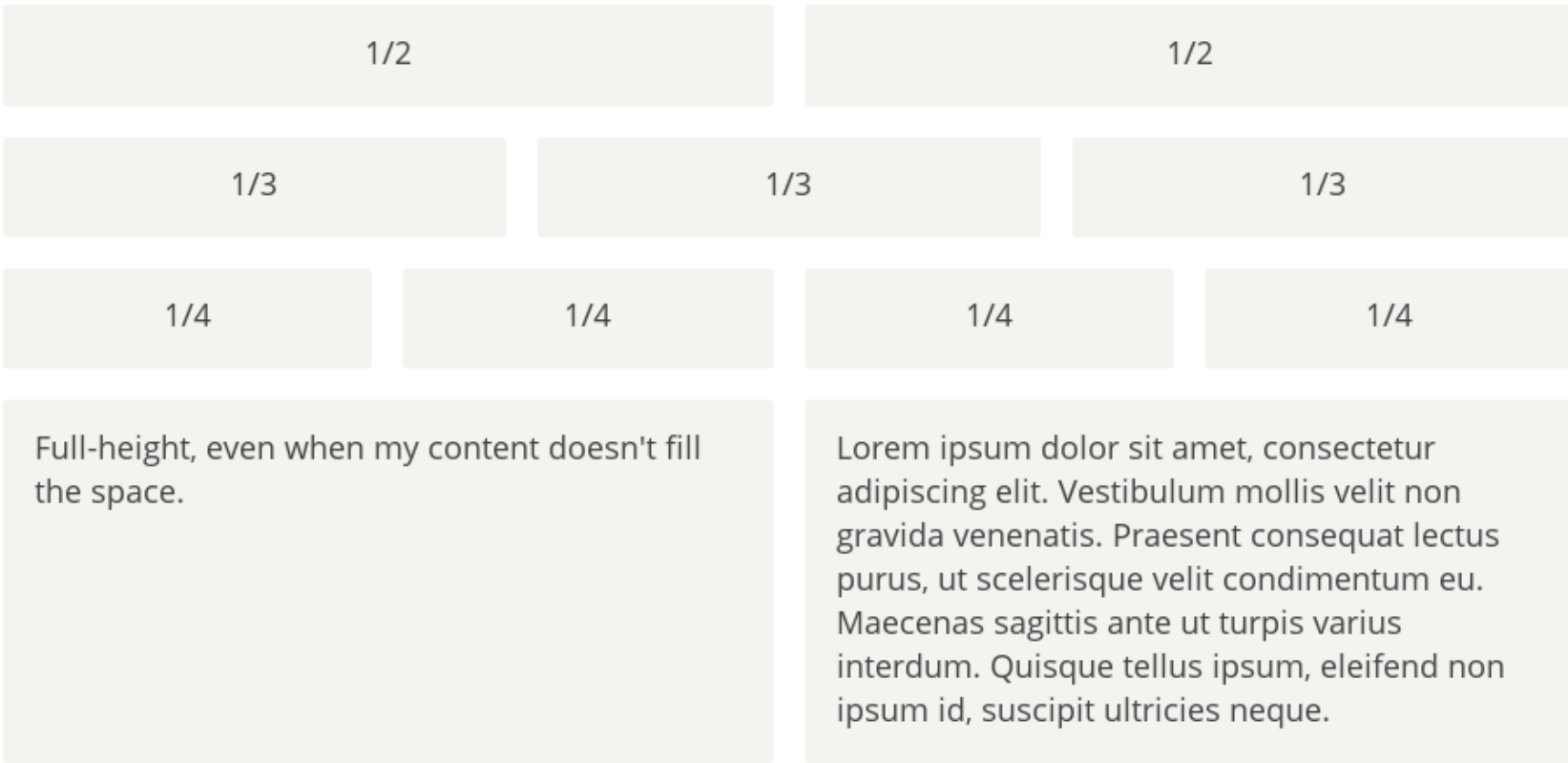


```
.box {
  display: flex;
  flex-wrap: wrap;
}
```

## 二、网格布局

### 2.1 基本网格布局

最简单的网格布局，就是平均分布。在容器里面平均分配空间，跟上面的骰子布局很像，但是需要设置项目的自动缩放。



HTML代码如下。

```
<div class="Grid">
  <div class="Grid-cell">...</div>
  <div class="Grid-cell">...</div>
  <div class="Grid-cell">...</div>
</div>
```

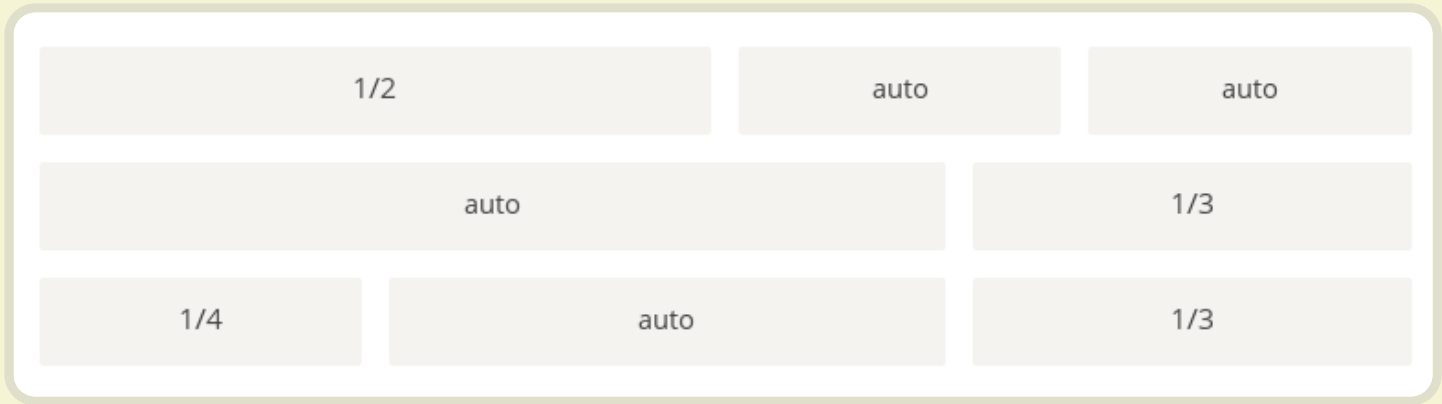
CSS代码如下。

```
.Grid {
  display: flex;
}

.Grid-cell {
  flex: 1;
}
```

## 2.2 百分比布局

某个网格的宽度为固定的百分比，其余网格平均分配剩余的空间。



HTML代码如下。

```
<div class="Grid">
  <div class="Grid-cell u-1of4">...</div>
  <div class="Grid-cell">...</div>
  <div class="Grid-cell u-1of3">...</div>
</div>
```

```
.Grid {
  display: flex;
}

.Grid-cell {
  flex: 1;
}

.Grid-cell.u-full {
  flex: 0 0 100%;
}

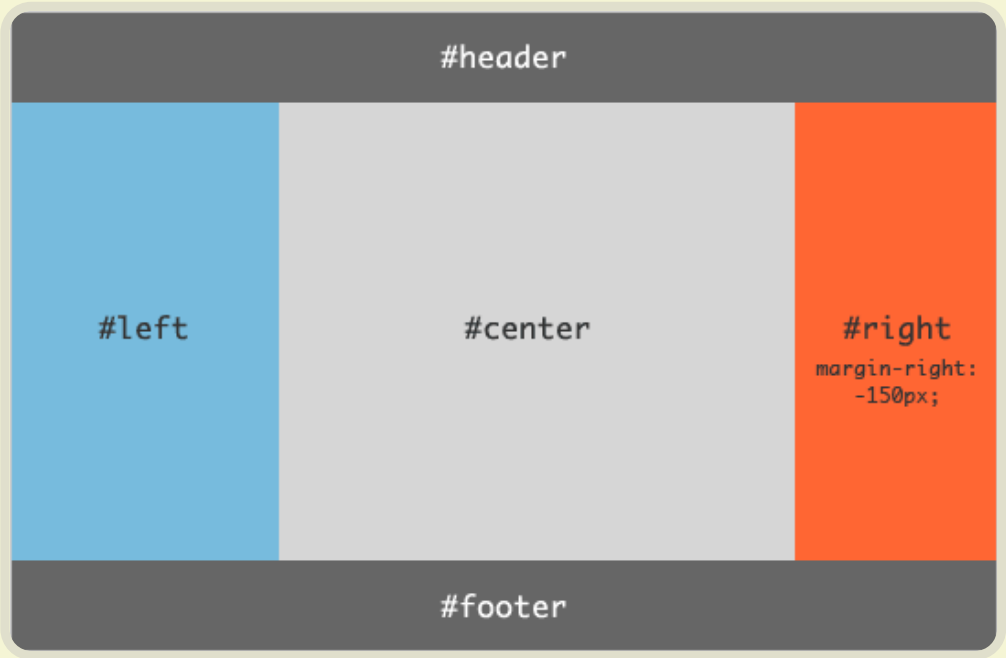
.Grid-cell.u-1of2 {
  flex: 0 0 50%;
}

.Grid-cell.u-1of3 {
  flex: 0 0 33.3333%;
}

.Grid-cell.u-1of4 {
  flex: 0 0 25%;
}
```

### 三、圣杯布局

[圣杯布局](#)（Holy Grail Layout）指的是一种最常见的网站布局。页面从上到下，分成三个部分：头部（header），躯干（body），尾部（footer）。其中躯干又水平分成三栏，从左到右为：导航、主栏、副栏。



HTML代码如下。

```
<body class="HolyGrail">
  <header>...</header>
  <div class="HolyGrail-body">
    <main class="HolyGrail-content">...</main>
    <nav class="HolyGrail-nav">...</nav>
    <aside class="HolyGrail-ads">...</aside>
  </div>
  <footer>...</footer>
</body>
```

CSS代码如下。

```

.HolyGrail {
  display: flex;
  min-height: 100vh;
  flex-direction: column;
}

header,
footer {
  flex: 1;
}

.HolyGrail-body {
  display: flex;
  flex: 1;
}

.HolyGrail-content {
  flex: 1;
}

.HolyGrail-nav, .HolyGrail-ads {
  /* 两个边栏的宽度设为12em */
  flex: 0 0 12em;
}

.HolyGrail-nav {
  /* 导航放到最左边 */
  order: -1;
}

```

如果是小屏幕，躯干的三栏自动变为垂直叠加。

```

@media (max-width: 768px) {
  .HolyGrail-body {
    flex-direction: column;
    flex: 1;
  }
  .HolyGrail-nav,
  .HolyGrail-ads,
  .HolyGrail-content {
    flex: auto;
  }
}

```

## 四、输入框的布局

我们常常需要在输入框的前方添加提示，后方添加按钮。

### Add-on Prepended

Amount

### Add-on Appended

Go

★

### Appended and Prepended Add-ons

Example One

Example One

HTML代码如下。

```
<div class="InputAddOn">
  <span class="InputAddOn-item">...</span>
  <input class="InputAddOn-field">
  <button class="InputAddOn-item">...</button>
</div>
```

CSS代码如下。


```
.InputAddOn {
  display: flex;
}

.InputAddOn-field {
  flex: 1;
}
```

## 五、悬挂式布局


有时，主栏的左侧或右侧，需要添加一个图片栏。

### Basic Examples



#### Standard Media Object

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ac nisl quis massa vulputate adipiscing. Vivamus sit amet risus ligula. Nunc eu pulvinar augue.




#### Standard Media Object


Donec imperdiet sem leo, id rutrum risus aliquam vitae. Cras tincidunt porta mauris, vel feugiat mauris accumsan eget.

#### Media Object Reversed

Phasellus vel felis purus. Aliquam consequat pellentesque dui, non mollis erat dictum sit amet. Curabitur non quam dictum, consectetur arcu in, vehicula justo. Donec tortor massa, eleifend nec viverra in, aliquet at eros. Mauris laoreet condimentum mauris, non tempor massa fermentum ut. Integer gravida pharetra cursus. Nunc in suscipit nunc.




### Non-images



#### Using Icons

Donec imperdiet sem leo, id rutrum risus aliquam vitae. Vestibulum ac turpis non lacus dignissim dignissim eu sed dui.



#### Vertically Centering the Figure

Nunc nec fermentum dolor. Duis at iaculis turpis. Sed rutrum elit ac egestas dapibus. Duis nec consequat enim.

HTML代码如下。

```
<div class="Media">
  <img class="Media-figure" src="" alt="">
  <p class="Media-body">...</p>
</div>
```

CSS代码如下。



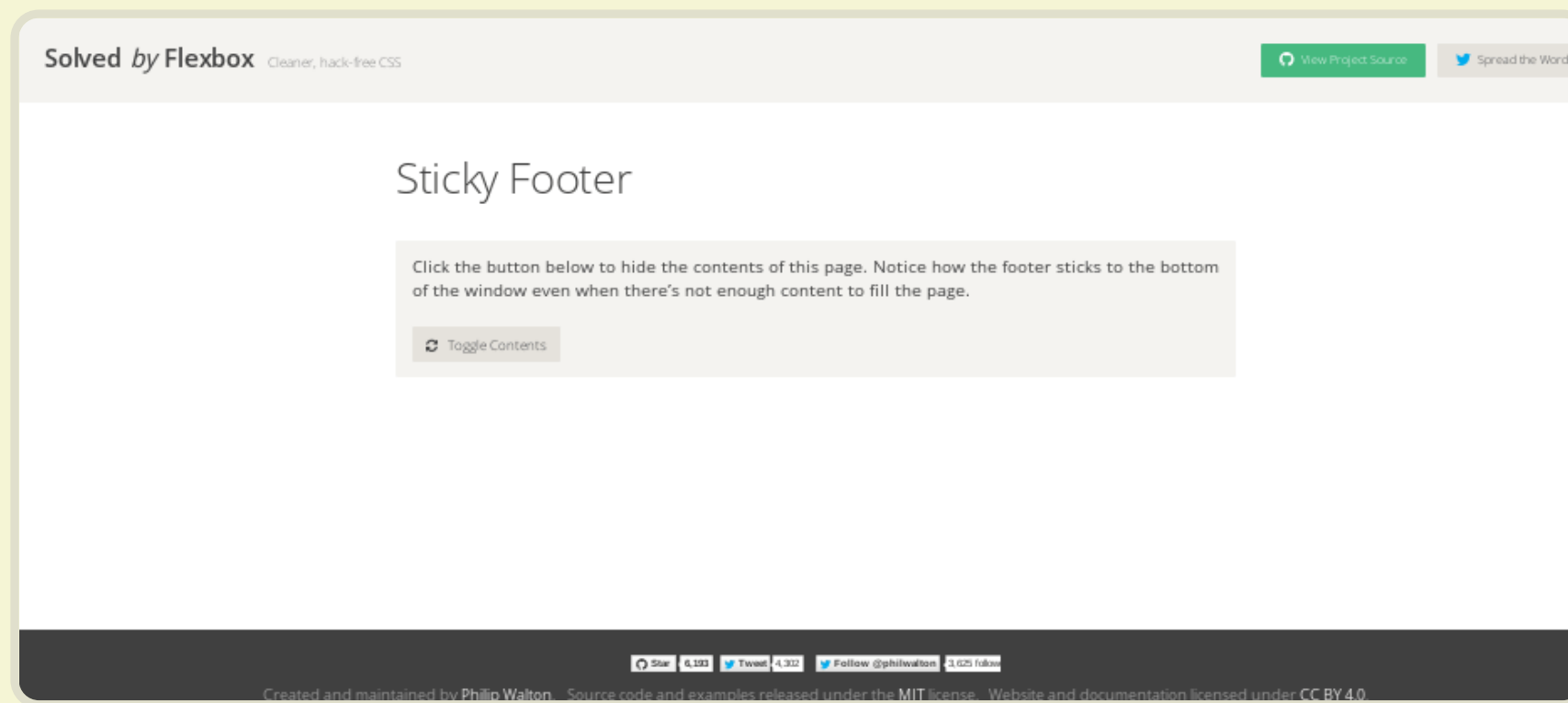
```
.Media {
  display: flex;
  align-items: flex-start;
}

.Media-figure {
  margin-right: 1em;
}

.Media-body {
  flex: 1;
}
```

## 六、固定的底栏

有时，页面内容太少，无法占满一屏的高度，底栏就会抬高到页面的中间。这时可以采用Flex布局，让底栏总是出现在页面的底部。



HTML代码如下。

```
<body class="Site">
  <header>...</header>
  <main class="Site-content">...</main>
  <footer>...</footer>
</body>
```

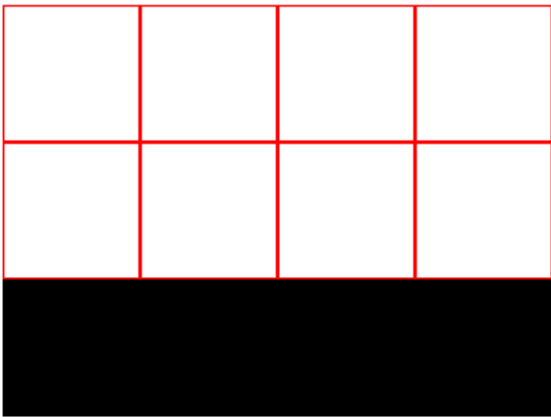
CSS代码如下。

```
.Site {
  display: flex;
  min-height: 100vh;
  flex-direction: column;
}

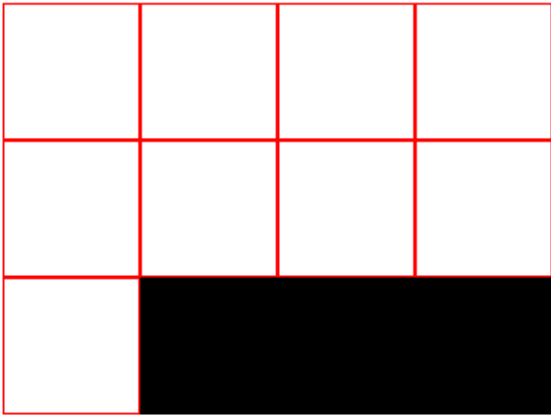
.Site-content {
  flex: 1;
}
```

## 七、流式布局

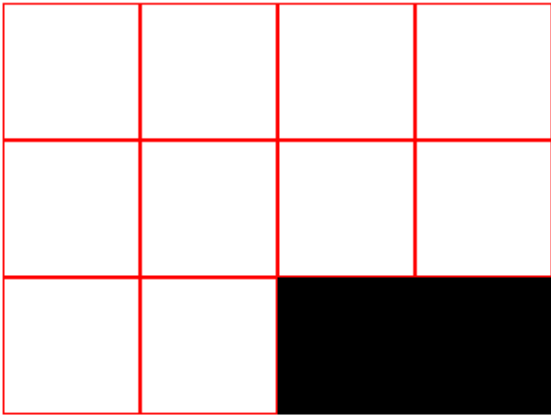
每行的项目数固定，会自动分行。



8个



9个



10个

CSS的写法。

```
.parent {
  width: 200px;
  height: 150px;
  background-color: black;
  display: flex;
  flex-flow: row wrap;
  align-content: flex-start;
}

.child {
  box-sizing: border-box;
  background-color: white;
  flex: 0 0 25%;
  height: 50px;
  border: 1px solid red;
}
```

(完)

- 版权声明： 自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期： 2015年7月14日

---

## 相关文章

---

- **2021.05.10:** [软件工程的最大难题](#)  
一、引言 大学有一门课程《软件工程》，研究如何组织和管理软件项目。
  - **2020.12.13:** [《SSH 入门教程》发布了](#)  
SSH 是登录 Linux 服务器的必备工具，只要你在做互联网开发，多多少少都会用到它。
  - **2020.11.02:** [微信小程序入门教程之四：API 使用](#)  
今天是这个系列教程的最后一篇。
  - **2020.10.29:** [微信小程序入门教程之三：脚本编程](#)  
这个系列教程的前两篇，介绍了小程序的项目结构和页面样式。
- 



[Weibo](#) | [Twitter](#) | [GitHub](#)  
[Email: yifeng.ruan@gmail.com](mailto:yifeng.ruan@gmail.com)

# Flexbox 布局的最简单表单

作者： 阮一峰

日期： 2018年10月18日

弹性布局（Flexbox）逐渐流行，越来越多人使用，因为它写 CSS 布局真是太方便了。

三年前，我写过 Flexbox 的介绍（[上](#)，[下](#)），但是有些地方写得不清楚。今天，我看到一篇[教程](#)，才意识到一个最简单的表单，就可以解释 Flexbox，而且内容还很实用。

下面，你只需要10分钟，就可以学会简单的表单布局。

## 一、<form> 元素

表单使用 `<form>` 元素。

```
<form></form>
```

上面是一个空表单。根据 HTML 标准，它是一个块级元素，默认将占据全部宽度，但是高度为0，因为没有任何内容。

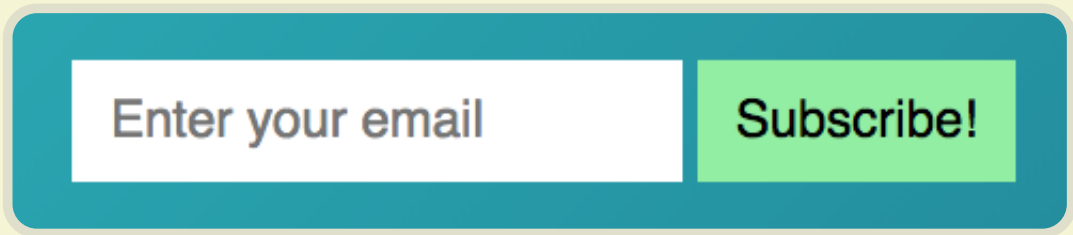
## 二、表单控件

现在，加入两个最常用的表单控件。

```
<form>
  <input type="email" name="email">
  <button type="submit">Send</button>
</form>
```

上面代码中，表单包含一个输入框（`<input>`）和一个按钮（`<button>`）。

根据标准，这两个控件都是行内块级元素（inline-block），也就是说，它们默认并排在一行上。

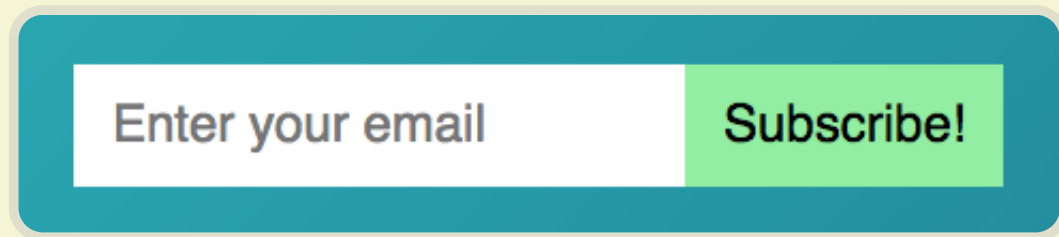


上图是浏览器对这个表单的默认渲染（颜色除外），可以看到，这两个控件之间有3像素~4像素的间隔，这是浏览器的内置样式指定的。

## 三、指定 Flexbox 布局

接着，指定表单使用 Flexbox 布局。

```
form {  
  display: flex;  
}
```



可以看到，两个控件之间的间隔消失了，因为弹性布局的项目（item）默认没有间隔。

## 四、flex-grow 属性

两个地方值得注意。

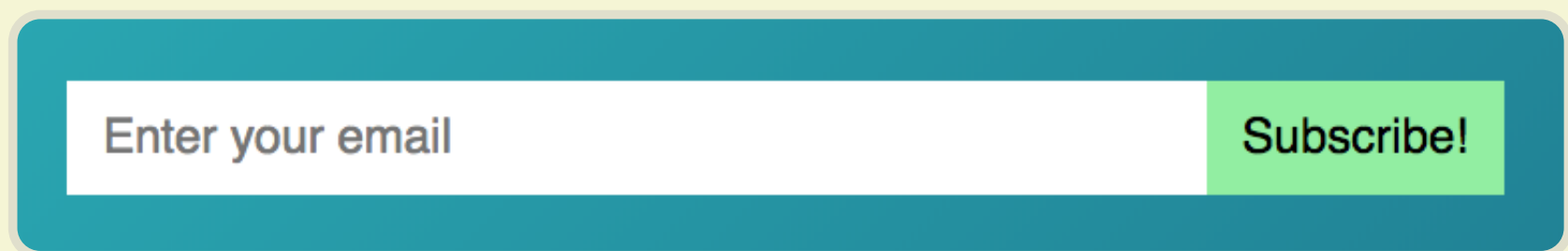
（1）两个控件元素的宽度没有发生变化，因为弹性布局默认不改变项目的宽度。

（2）弹性布局默认左对齐，所以两个控件会从行首开始排列。

如果我们希望，输入框占据当前行的所有剩余宽度，只需要指定输入框的

`flex-grow` 属性为 `1`。

```
input {  
  flex-grow: 1;  
}
```



上图中，按钮的宽度没变，但是输入框变宽了，等于当前行的宽度减去按钮的宽度。

`flex-grow` 属性默认等于 `0`，即使用本来的宽度，不拉伸。等于 `1` 时，就表示该项目宽度拉伸，占据当前行的所有剩余宽度。

## 五、align-self 属性和 align-items 属性

我们做一点改变，在按钮里面插入一张图片。

```
<form action="#">
  <input type="email" placeholder="Enter your email">
  <button type="button"><svg>  <!-- a smiley icon -->  </svg>
</form>
```

按钮插入图片后，它的高度变了，变得更高了。这时，就发生了一件很奇妙的事情。

上图中，按钮变高了，输入框也自动变得一样高了！

前面说过，弹性布局默认不改变项目的宽度，但是它默认改变项目的高度。如果项目没有显式指定高度，就将占据容器的所有高度。本例中，按钮变高了，导致表单元素也变高了，使得输入框的高度自动拉伸了。

`align-self` 属性可以改变这种行为。

```
input {
  flex-grow: 1;
  align-self: center;
}
```

## Flex-start

## Flex-end

## Center

## Stretch

`align-self` 属性可以取四个值。

- `flex-start`: 顶边对齐，高度不拉伸

- flex-end: 底边对齐，高度不拉伸
- center: 居中，高度不拉伸
- stretch: 默认值，高度自动拉伸

如果项目很多，一个个地设置 align-self 属性就很麻烦。这时，可以在容器元素（本例为表单）设置 align-items 属性，它的值被所有子项目的 align-self 属性继承。

```
form {
  display: flex;
  align-items: center;
}
```

上面代码中，<form> 元素设置了 align-items 以后，就不用控件上设置 align-self，除非希望两者的值不一样。

(完)

### 文档信息

- 版权声明： 自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期： 2018年10月18日

## 相关文章

- **2020.12.13:** [《SSH 入门教程》发布了](#)  
SSH 是登录 Linux 服务器的必备工具，只要你在做互联网开发，多多少少都会用到它。
- **2020.11.02:** [微信小程序入门教程之四：API 使用](#)  
今天是这个系列教程的最后一篇。
- **2020.10.29:** [微信小程序入门教程之三：脚本编程](#)  
这个系列教程的前两篇，介绍了小程序的项目结构和页面样式。
- **2020.10.27:** [微信小程序入门教程之二：页面样式](#)  
这个系列的上一篇教程，教大家写了一个最简单的 Hello world 微信小程序。



[Weibo](#) | [Twitter](#) | [GitHub](#)

Email: [yifeng.ruan@gmail.com](mailto:yifeng.ruan@gmail.com)



# CSS 定位详解

作者： 阮一峰

日期： 2019年11月19日

CSS 有两个最重要的基本属性，前端开发必须掌握：`display` 和 `position`。

`display` 属性指定网页的布局。两个重要的布局，我已经介绍过了：[弹性布局 flex](#) 和[网格布局 grid](#)。

本文介绍非常有用的 `position` 属性。我希望通过10分钟的阅读，帮助大家轻松掌握网页定位，说清楚浏览器如何计算网页元素的位置，尤其是新引入的 `sticky` 定位。



本文由国内最大的在线教育平台之一["腾讯课堂"](#)赞助。他们现在启动了["腾讯课堂101计划"](#)，推广平台上的课程资源，有不少优质内容。希望提高前端技术水平的同学，可以留意一下本文结尾的免费课程信息。

## 一、position 属性的作用

`position` 属性用来指定一个元素在网页上的位置，一共有5种定位方式，即 `position` 属性主要有五个值。

- `static`
- `relative`

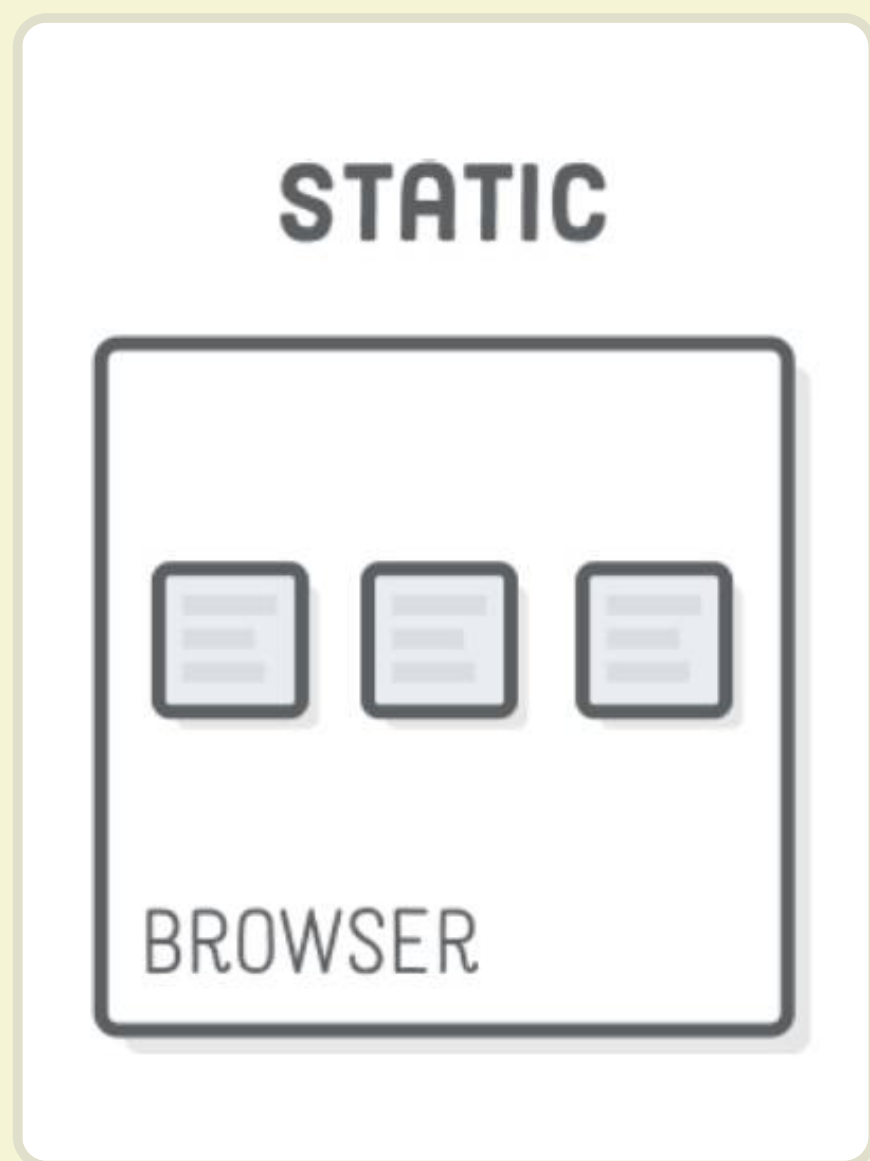
- fixed
- absolute
- sticky

下面就依次介绍这五个值。最后一个 `sticky` 是2017年浏览器才支持的，本文将重点介绍。

## 二、static 属性值

`static` 是 `position` 属性的默认值。如果省略 `position` 属性，浏览器就认为该元素是 `static` 定位。

这时，浏览器会按照源码的顺序，决定每个元素的位置，这称为"正常的页面流"（normal flow）。每个块级元素占据自己的区块（block），元素与元素之间不产生重叠，这个位置就是元素的默认位置。



注意，`static` 定位所导致的元素位置，是浏览器自主决定的，所以这时 `top`、`bottom`、`left`、`right` 这四个属性无效。

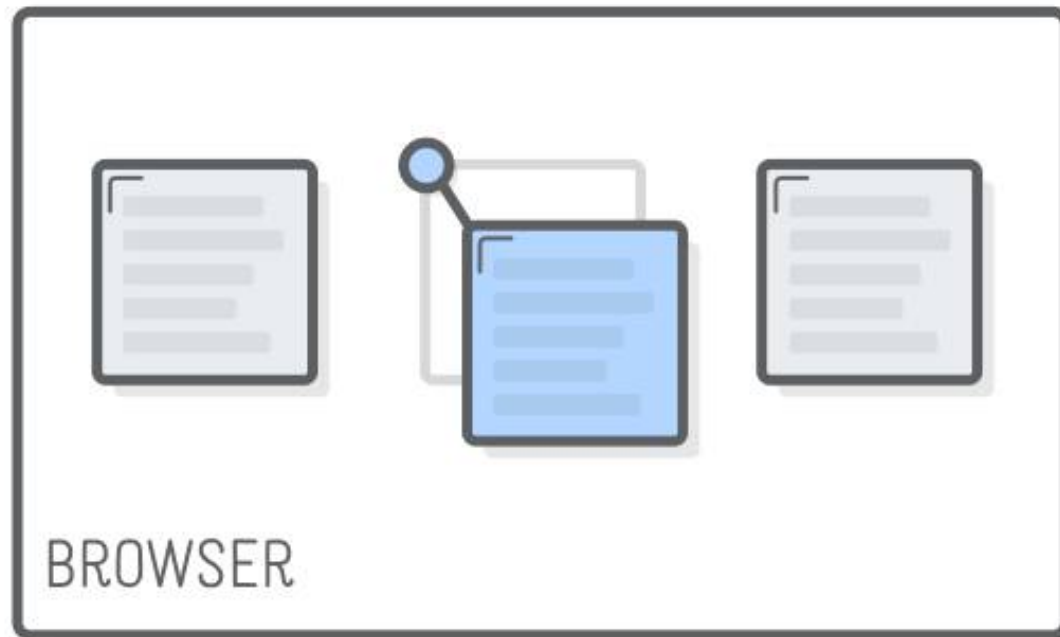
## 三、relative, absolute, fixed

`relative`、`absolute`、`fixed` 这三个属性值有一个共同点，都是相对于某个基点的定位，不同之处仅仅在于基点不同。所以，只要理解了它们的基点是什么，就很容易掌握这三个属性值。

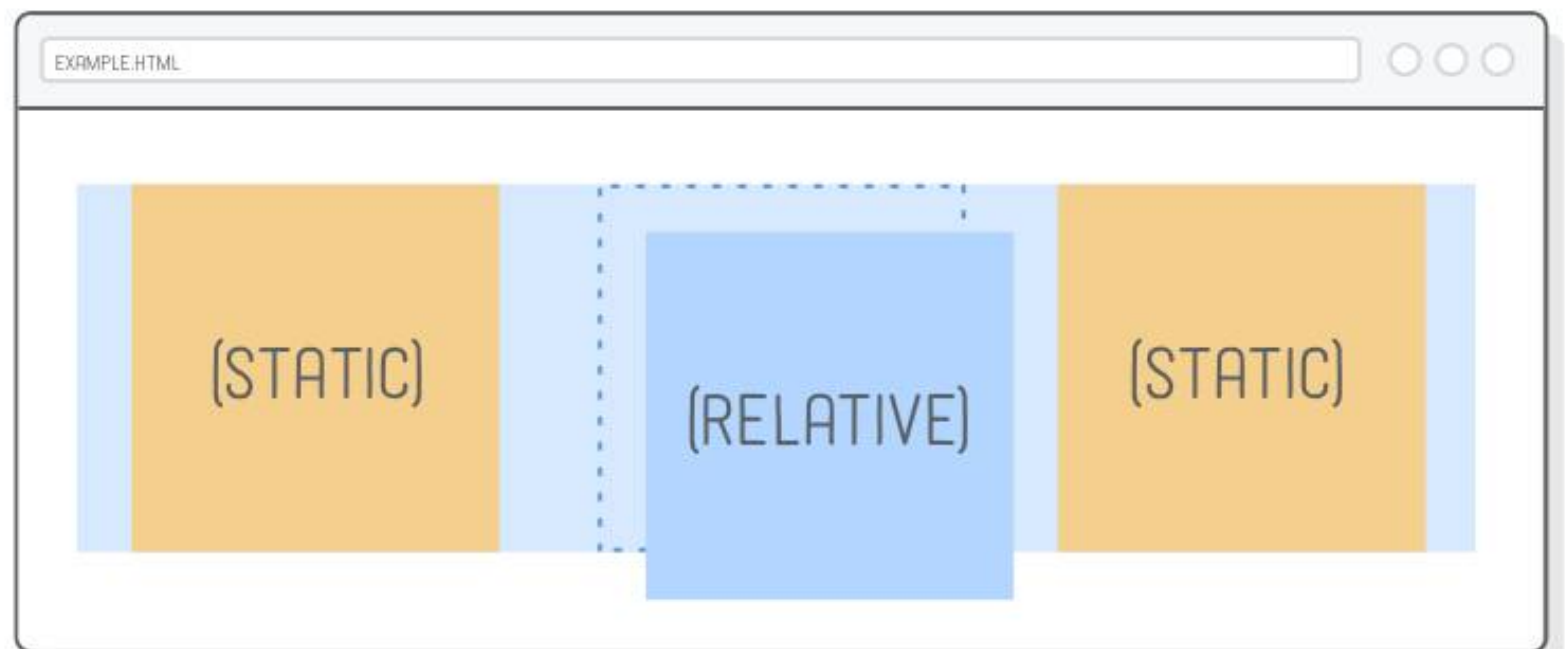
这三种定位都不会对其他元素的位置产生影响，因此元素之间可能产生重叠。

### 3.1 relative 属性值

`relative` 表示，相对于默认位置（即 `static` 时的位置）进行偏移，即定位基点是元素的默认位置。

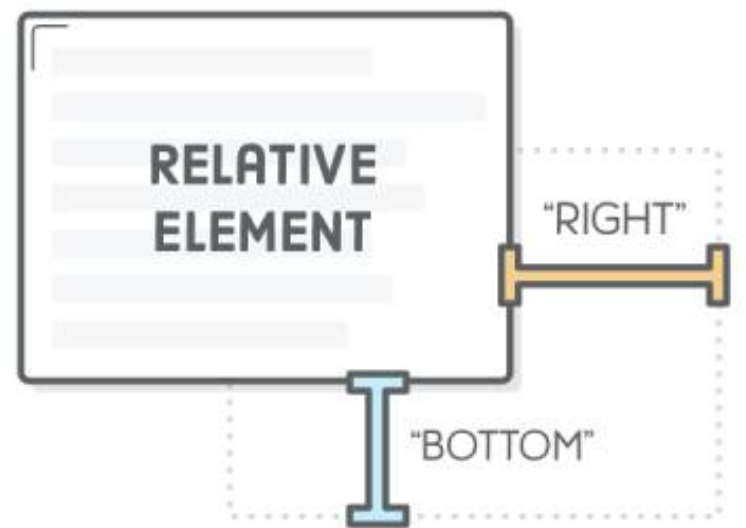
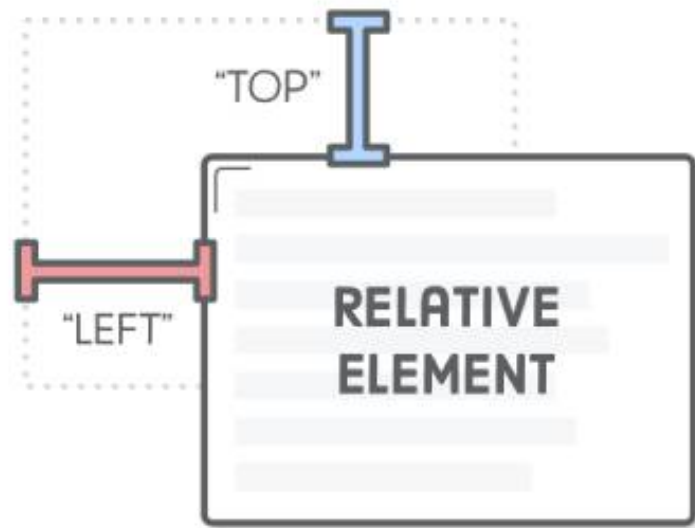


## RELATIVE POSITIONING



它必须搭配 `top`、`bottom`、`left`、`right` 这四个属性一起使用，用来指定偏移的方向和距离。

## ORIGINAL POSITION



## ORIGINAL POSITION

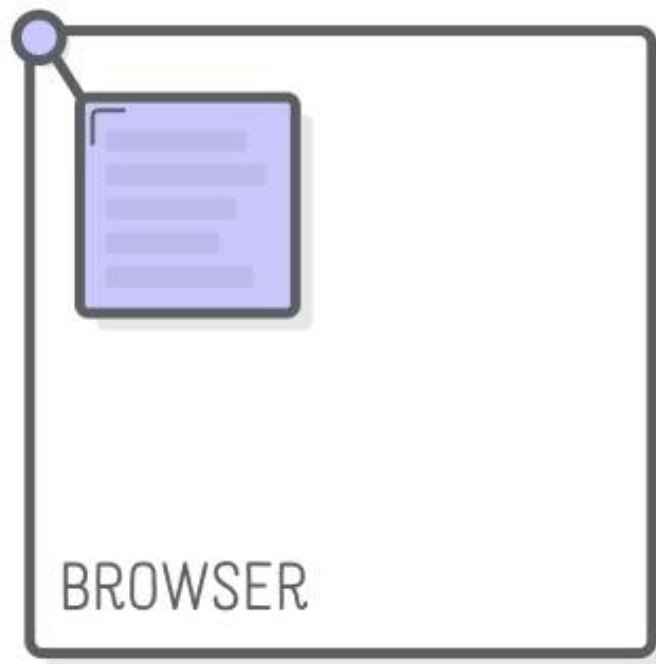
```
div {  
  position: relative;  
  top: 20px;  
}
```

上面代码中，`div` 元素从默认位置向下偏移 `20px`（即距离顶部 `20px`）。

### 3.2 absolute 属性值

`absolute` 表示，相对于上级元素（一般是父元素）进行偏移，即定位基点是父元素。

它有一个重要的限制条件：定位基点（一般是父元素）不能是 `static` 定位，否则定位基点就会变成整个网页的根元素 `html`。另外，`absolute` 定位也必须搭配 `top`、`bottom`、`left`、`right` 这四个属性一起使用。



## ABSOLUTE POSITIONING

```
/*
HTML 代码如下
<div id="father">
  <div id="son"></div>
</div>
*/

#father {
  position: relative;
}
#son {
  position: absolute;
  top: 20px;
}
```

上面代码中，父元素是 `relative` 定位，子元素是 `absolute` 定位，所以子元素的定位基点是父元素，相对于父元素的顶部向下偏移 `20px`。如果父元素是 `static` 定位，上例的子元素就是距离网页的顶部向下偏移 `20px`。

注意，`absolute` 定位的元素会被"正常页面流"忽略，即在"正常页面流"中，该元素所占空间为零，周边元素不受影响。

### 3.3 fixed 属性值

`fixed` 表示，相对于视口（viewport，浏览器窗口）进行偏移，即定位基点是浏览器窗口。这会导致元素的位置不随页面滚动而变化，好像固定在网页上一样。



## FIXED POSITIONING

它如果搭配 `top`、`bottom`、`left`、`right` 这四个属性一起使用，表示元素的初始位置是基于视口计算的，否则初始位置就是元素的默认位置。

```
div {  
  position: fixed;  
  top: 0;  
}
```

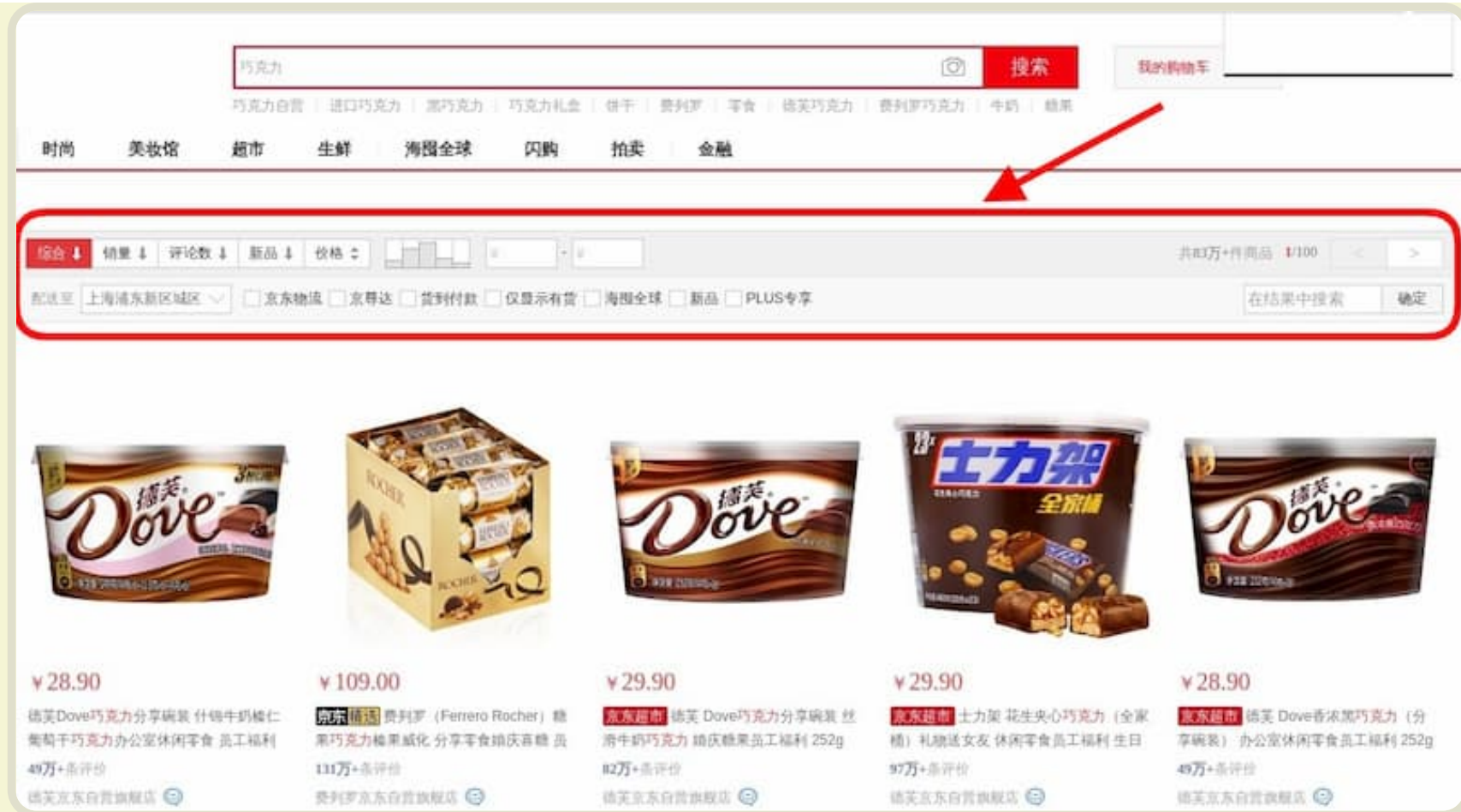
上面代码中，`div` 元素始终在视口顶部，不随网页滚动而变化。

## 四、sticky 属性值

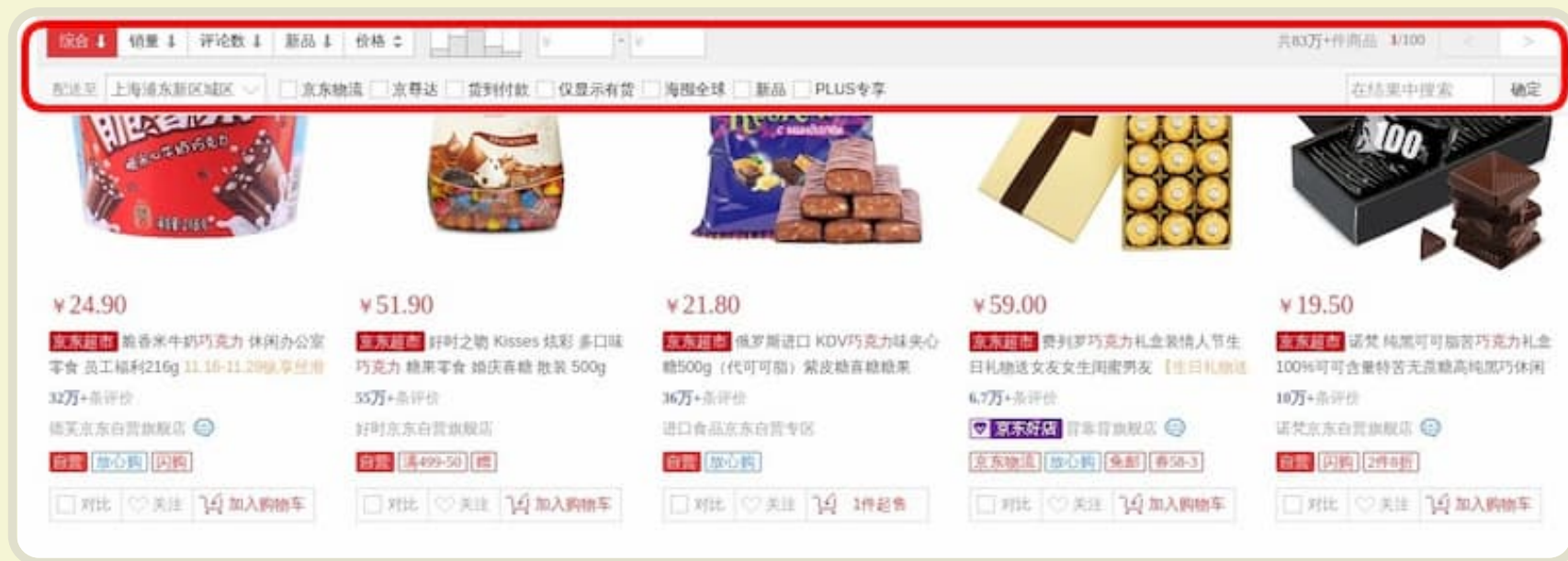
`sticky` 跟前面四个属性值都不一样，它会产生动态效果，很像 `relative` 和 `fixed` 的结合：一些时候是 `relative` 定位（定位基点是自身默认位置），另一些时候自动变成 `fixed` 定位（定位基点是视口）。

因此，它能够形成"动态固定"的效果。比如，网页的搜索工具栏，初始加载时在自己的默认位置（`relative` 定位）。





页面向下滚动时，工具栏变成固定位置，始终停留在页面头部（`fixed` 定位）。



等到页面重新向上滚动回到原位，工具栏也会回到默认位置。

`sticky` 生效的前提是，必须搭配 `top`、`bottom`、`left`、`right` 这四个属性一起使用，不能省略，否则等同于 `relative` 定位，不产生"动态固定"的效果。原因是这四个属性用来定义"偏移距离"，浏览器把它当作 `sticky` 的生效门槛。

它的具体规则是，当页面滚动，父元素开始脱离视口时（即部分不可见），只要与 `sticky` 元素的距离达到生效门槛，`relative` 定位自动切换为 `fixed` 定位；等到父元素完全脱离视口时（即完全不可见），`fixed` 定位自动切换回 `relative` 定位。

请看下面的示例代码。（注意，除了已被淘汰的 IE 以外，其他浏览器目前都支持 `sticky`。但是，Safari 浏览器需要加上浏览器前缀 `-webkit-`。）

```
#toolbar {
  position: -webkit-sticky; /* safari 浏览器 */
  position: sticky; /* 其他浏览器 */
  top: 20px;
}
```

上面代码中，页面向下滚动时，`#toolbar` 的父元素开始脱离视口，一旦视口的顶部与 `#toolbar` 的距离小于 `20px`（门槛值），`#toolbar` 就自动变为 `fixed` 定位，保持与视口顶部 `20px` 的距离。页面继续向下滚动，父元素彻底离开视口（即整个父元素完全不可见），`#toolbar` 恢复成 `relative` 定位。

## 五、sticky 的应用

`sticky` 定位可以实现一些很有用的效果。除了上面提到"动态固定"效果，这里再介绍两个。

### 5.1 堆叠效果

堆叠效果（stacking）指的是页面滚动时，下方的元素覆盖上方的元素。下面是一个图片堆叠的例子，下方的图片会随着页面滚动，覆盖上方的图片（查看 [demo](#)）。



HTML 代码就是几张图片。

```
<div></div>
<div></div>
<div></div>
```

CSS 代码极其简单，只要两行。

```
div {
  position: sticky;
  top: 0;
}
```

它的原理是页面向下滚动时，每张图片都会变成 `fixed` 定位，导致后一张图片重叠在



前一张图片上面。详细解释可以看[这里](#)。

## 5.2 表格的表头锁定

大型表格滚动的时候，表头始终固定，也可以用 `sticky` 实现（查看 [demo](#)）。

Some data		Some
Lorem	Ipsum	Lorem
Vestibulum rhoncus	Aliquam sed	Donec lobortis pharetra
Vestibulum rhoncus	Aliquam sed	Donec lobortis pharetra
Vestibulum rhoncus	Aliquam sed	Donec lobortis pharetra
Vestibulum rhoncus	Aliquam sed	Donec lobortis pharetra
Vestibulum rhoncus	Aliquam sed	Donec lobortis pharetra
Vestibulum rhoncus	Aliquam sed	Donec lobortis pharetra
Vestibulum rhoncus	Aliquam sed	Donec lobortis pharetra

CSS 代码也很简单。

```
th {
  position: sticky;
  top: 0;
}
```

需要注意的是，`sticky` 必须设在 `<th>` 元素上面，不能设在 `<thead>` 和 `<tr>` 元素，因为这两个元素没有 `relative` 定位，也就无法产生 `sticky` 效果。详细解释可以看[这里](#)。

（正文完）

## 免费前端全栈课程

初学者刚接触前端，往往会被一大堆技术名词、框架和工具，搞得眼花缭乱。

到底哪些技术是目前的主流技术栈，既能用于公司的开发实务，又能为自己的简历增添亮点？



下面就是一套目前主流的前端技术栈。

- (1) Node.js：服务器端的 JavaScript 运行环境，不管哪种前端开发，都必不可少的底层环境。
- (2) Webpack：语法转换工具，把 ES6/TypeScript/JSX 语法转成浏览器可以运行的代码。
- (3) Koa2：一个非常流行、简洁强大的 Node.js 后端的 Web 开发框架。
- (4) MongoDB：目前应用最广泛的非关系数据库之一，功能丰富，用法较简单。
- (5) Vue 全家桶：

- Vue：前端基础框架
- Vuex：配套的前端状态管理库。
- Vue Router：官方的路由插件，构建单页面应用必不可少。
- Vue CLI：脚手架工具，帮你快速上手 Vue 开发，无需再花多余时间去实现项目架构。
- Vant：有赞前端团队开发的轻量级移动端 Vue 组件库，让你快速使用已经封装好的各种页面组件。

看到这个名单，你是不是感到有点头大，全部掌握它们需要多少时间啊？

现在，腾讯课堂就有一门这样的课程，内容包含了所有这些工具，教你怎么用它们从头完成一个全栈项目，亲手做出一个手机端的移动商城，是由 **慕课网的精英讲师--谢成老师** 讲授。

这个课程原价98元， **活动期间，只要1块钱哦！** 微信扫描下面的二维码，就可以领取优惠券，享受1元听课的福利。



该课程的制作单位是青盟科技。它是《腾讯课堂101计划》重点推广的优质机构，已有7年 IT 行业教学经验，培养收费学员2000+，有超过72%的学员都进入到名企大厂。如果你想了解课程的详细内容，获取课程大纲，或者想接受系统的前端培训，可以登录腾讯课堂查看"青盟科技"。

(完)

### 文档信息

- 版权声明： 自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期： 2019年11月19日

## 相关文章

- **2021.05.10:** [软件工程的最大难题](#)  
一、引言 大学有一门课程《软件工程》，研究如何组织和管理软件项目。
- **2020.12.13:** [《SSH 入门教程》发布了](#)  
SSH 是登录 Linux 服务器的必备工具，只要你在做互联网开发，多多少少都会用到它。
- **2020.11.02:** [微信小程序入门教程之四：API 使用](#)  
今天是这个系列教程的最后一篇。
- **2020.10.29:** [微信小程序入门教程之三：脚本编程](#)  
这个系列教程的前两篇，介绍了小程序的项目结构和页面样式。



[Weibo](#) | [Twitter](#) | [GitHub](#)

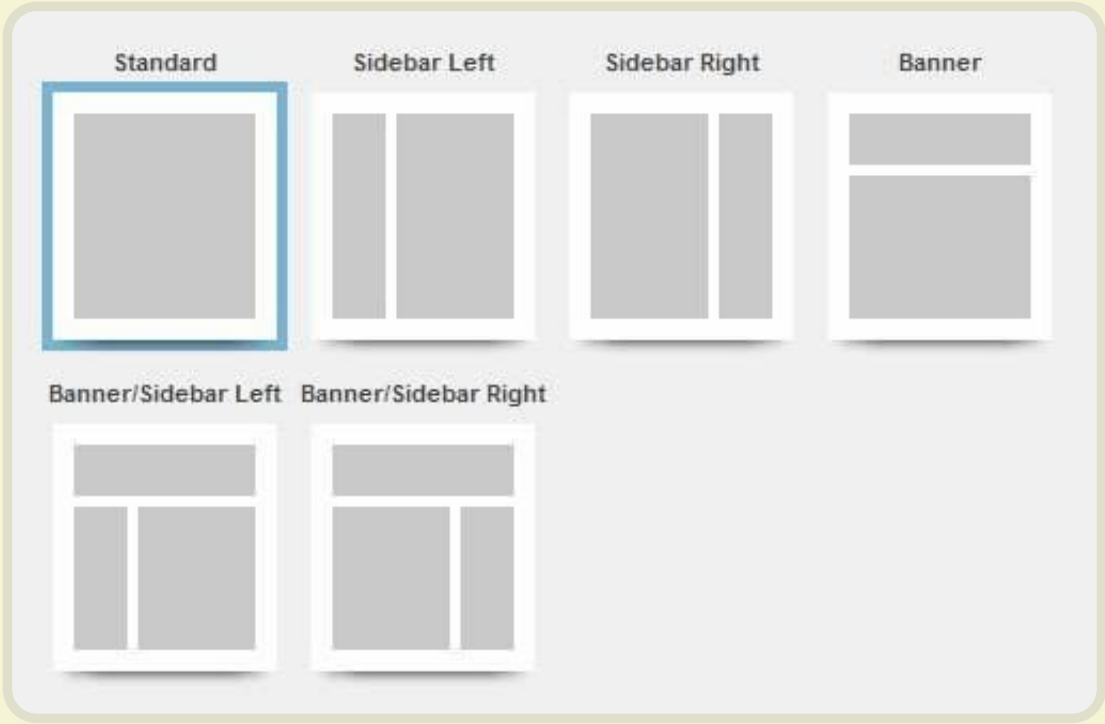
Email: [yifeng.ruan@gmail.com](mailto:yifeng.ruan@gmail.com)

# 只要一行代码，实现五种 CSS 经典布局

作者： 阮一峰

日期： 2020年8月10日

页面布局是样式开发的第一步，也是 CSS 最重要的功能之一。



常用的页面布局，其实就那么几个。下面我会介绍5个经典布局，只要掌握了它们，就能应对绝大多数常规页面。

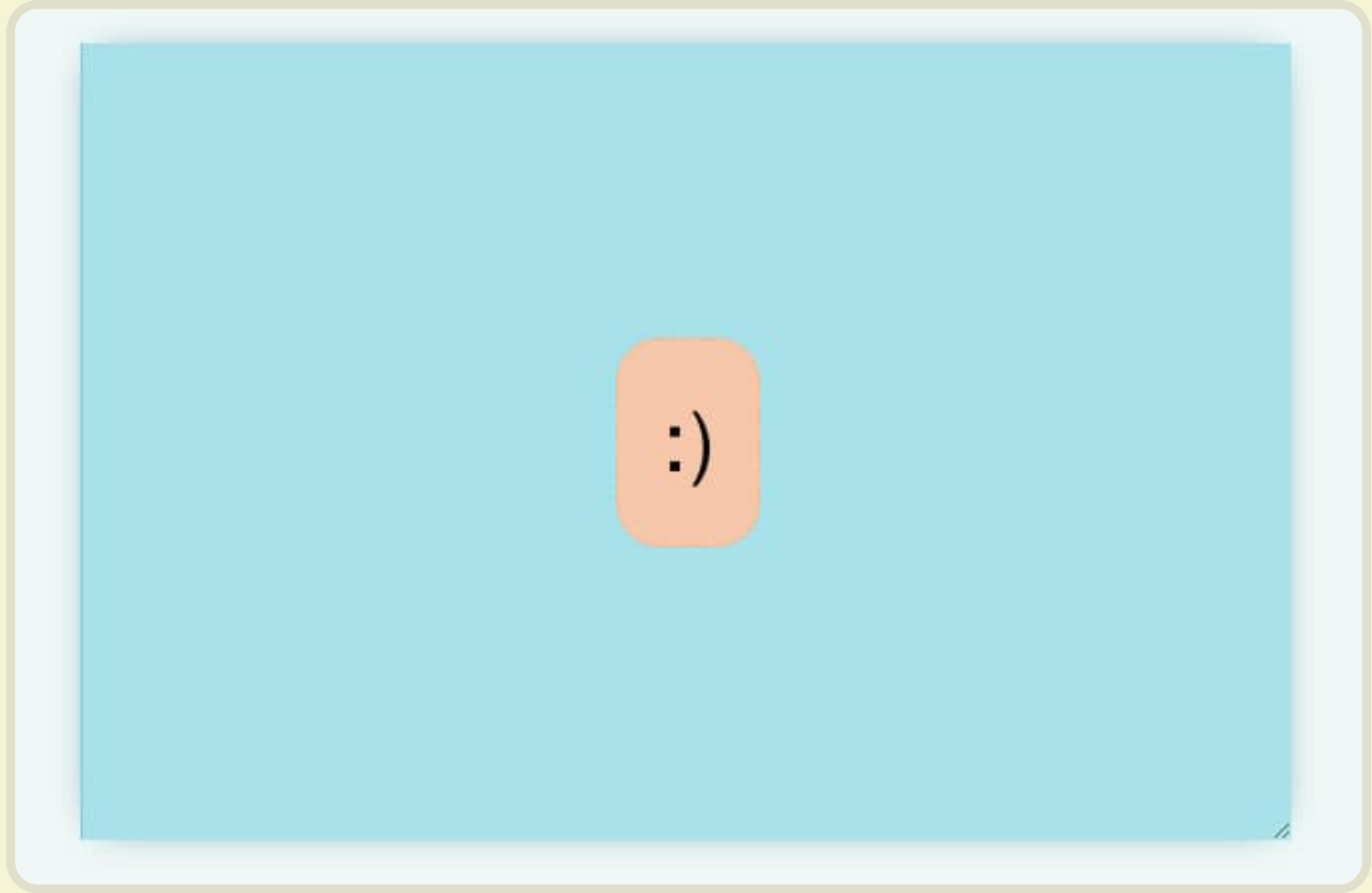
这几个布局都是自适应的，自动适配桌面设备和移动设备。代码实现很简单，核心代码只有一行，有很大的学习价值，内容也很实用。

我会用到 CSS 的 [Flex 语法](#)和 [Grid 语法](#)，不过只用到一点点，不熟悉的朋友可以先看看教程链接，熟悉一下基本概念。每一个布局都带有 CodePen 示例，也可以到[这个网页](#)统一查看。

本文是跟极客大学合作的前端学习讲座的一部分，详见文末说明。

## 一、空间居中布局

空间居中布局指的是，不管容器的大小，项目总是占据中心点。



CSS 代码如下（[CodePen](#) 示例）。

```
.container {  
  display: grid;  
  place-items: center;  
}
```

上面代码需要写在容器上，指定为 Grid 布局。核心代码是 `place-items` 属性那一行，它是一个简写形式。

```
place-items: <align-items> <justify-items>;
```

`align-items` 属性控制垂直位置，`justify-items` 属性控制水平位置。这两个属性的值一致时，就可以合并写成一个值。所以，`place-items: center;` 等同于 `place-items: center center;`。

同理，左上角布局可以写成下面这样。

```
place-items: start;
```



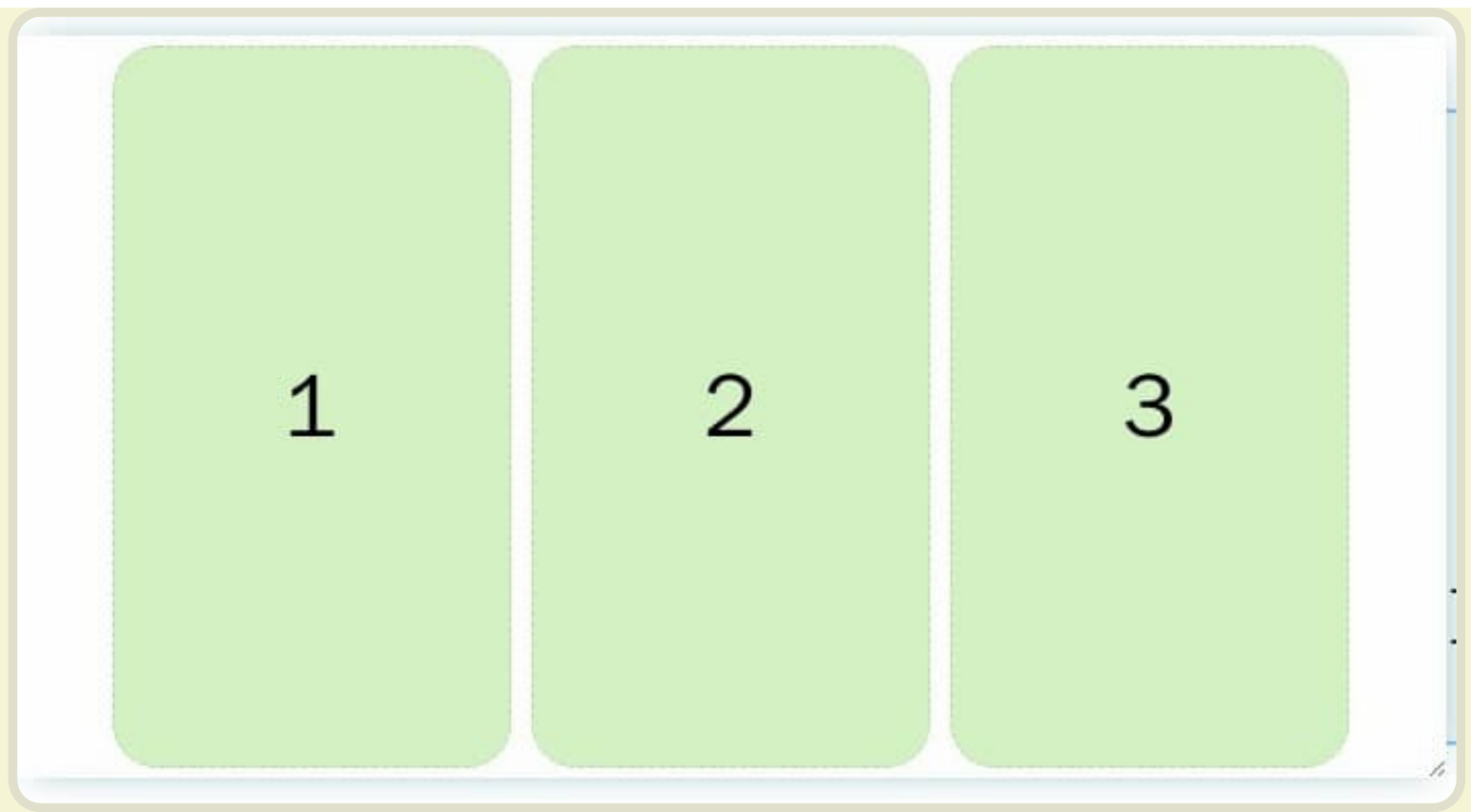
右下角布局。

```
place-items: end;
```

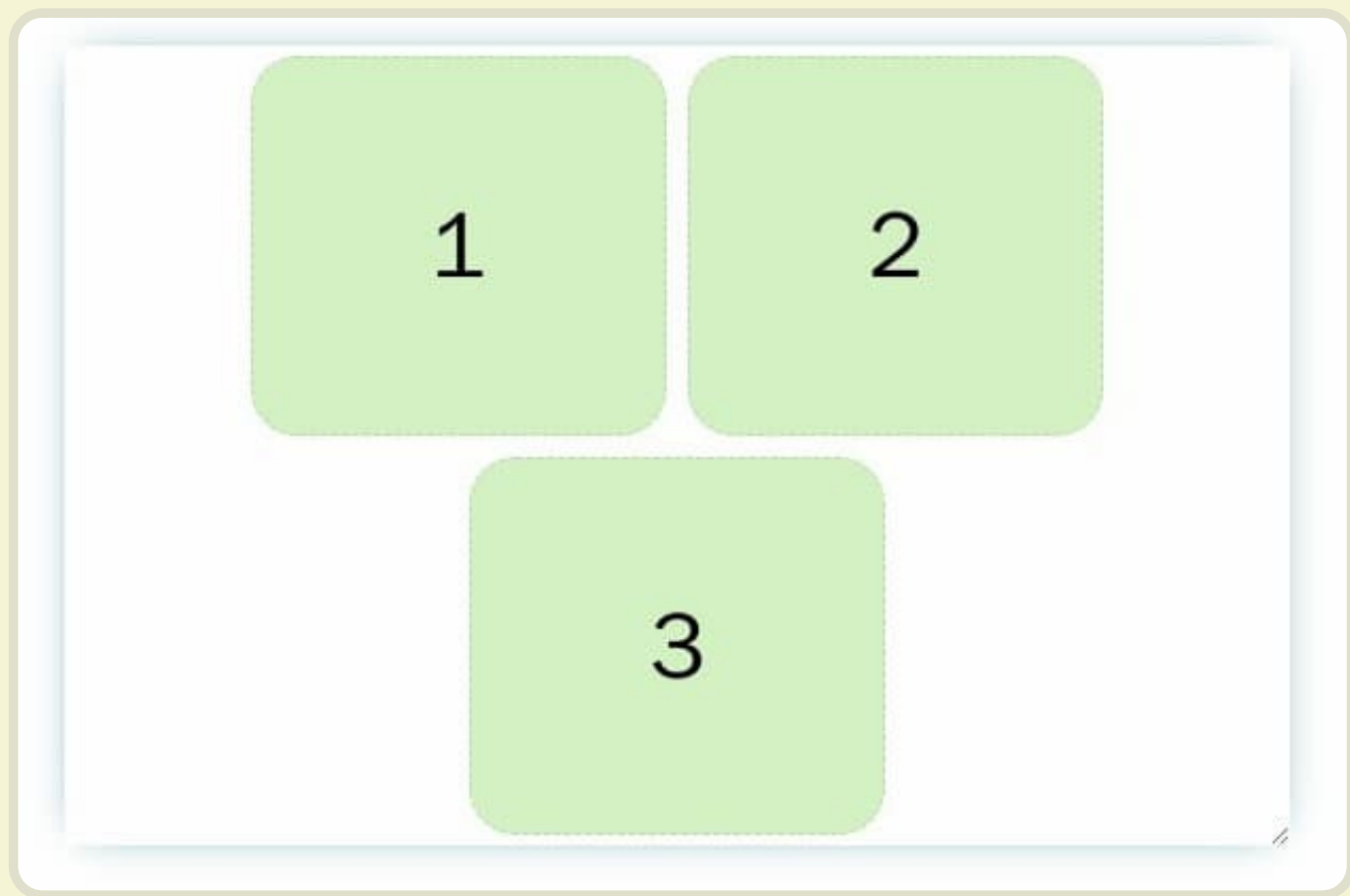


## 二、并列式布局

并列式布局就是多个项目并列。



如果宽度不够，放不下的项目就自动折行。







它的实现也很简单。首先，容器设置成 Flex 布局，内容居中（`justify-content`）可换行（`flex-wrap`）。

```
.container {  
  display: flex;  
  flex-wrap: wrap;  
  justify-content: center;  
}
```

然后，项目上面只用一行 `flex` 属性就够了（[CodePen 示例](#)）。

```
.item{  
  flex: 0 1 150px;  
  margin: 5px;  
}
```

`flex` 属性是 `flex-grow`、`flex-shrink`、`flex-basis` 这三个属性的简写形式。

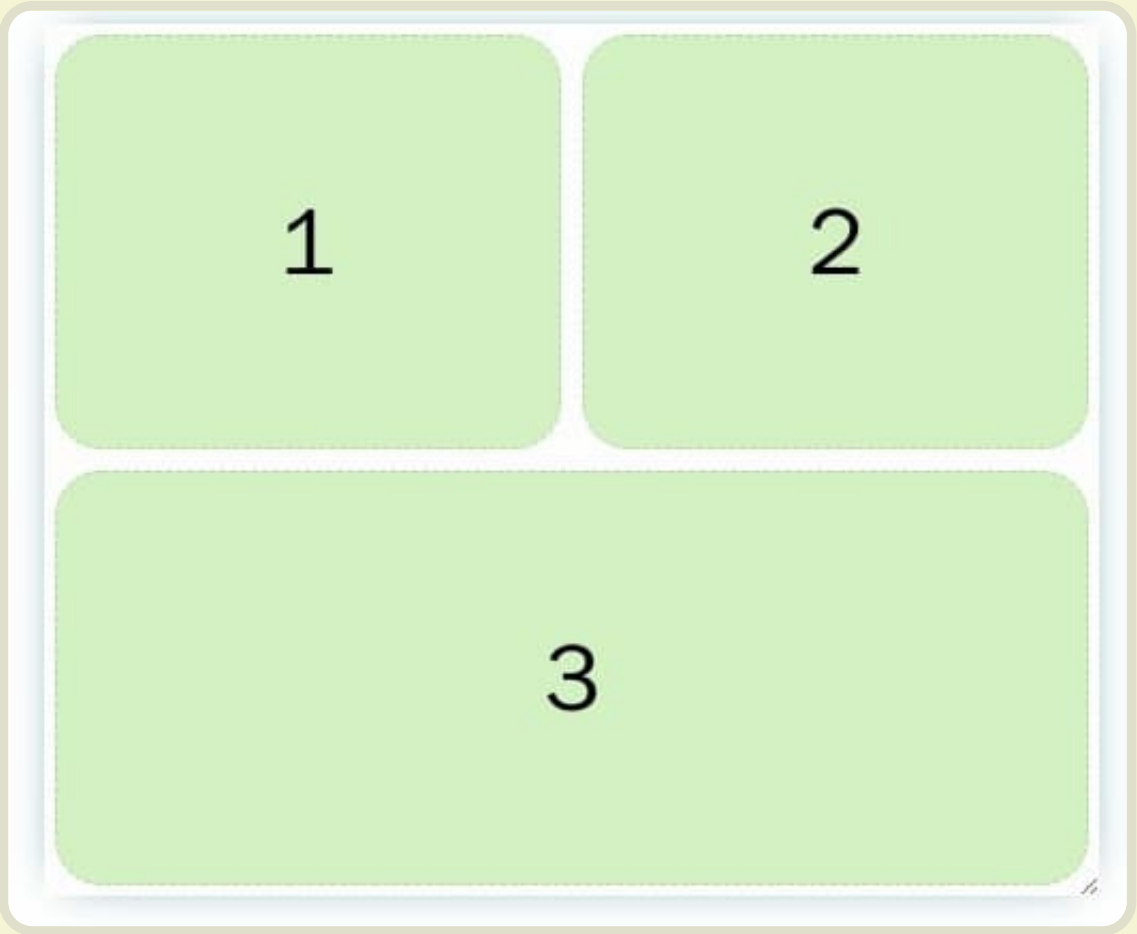
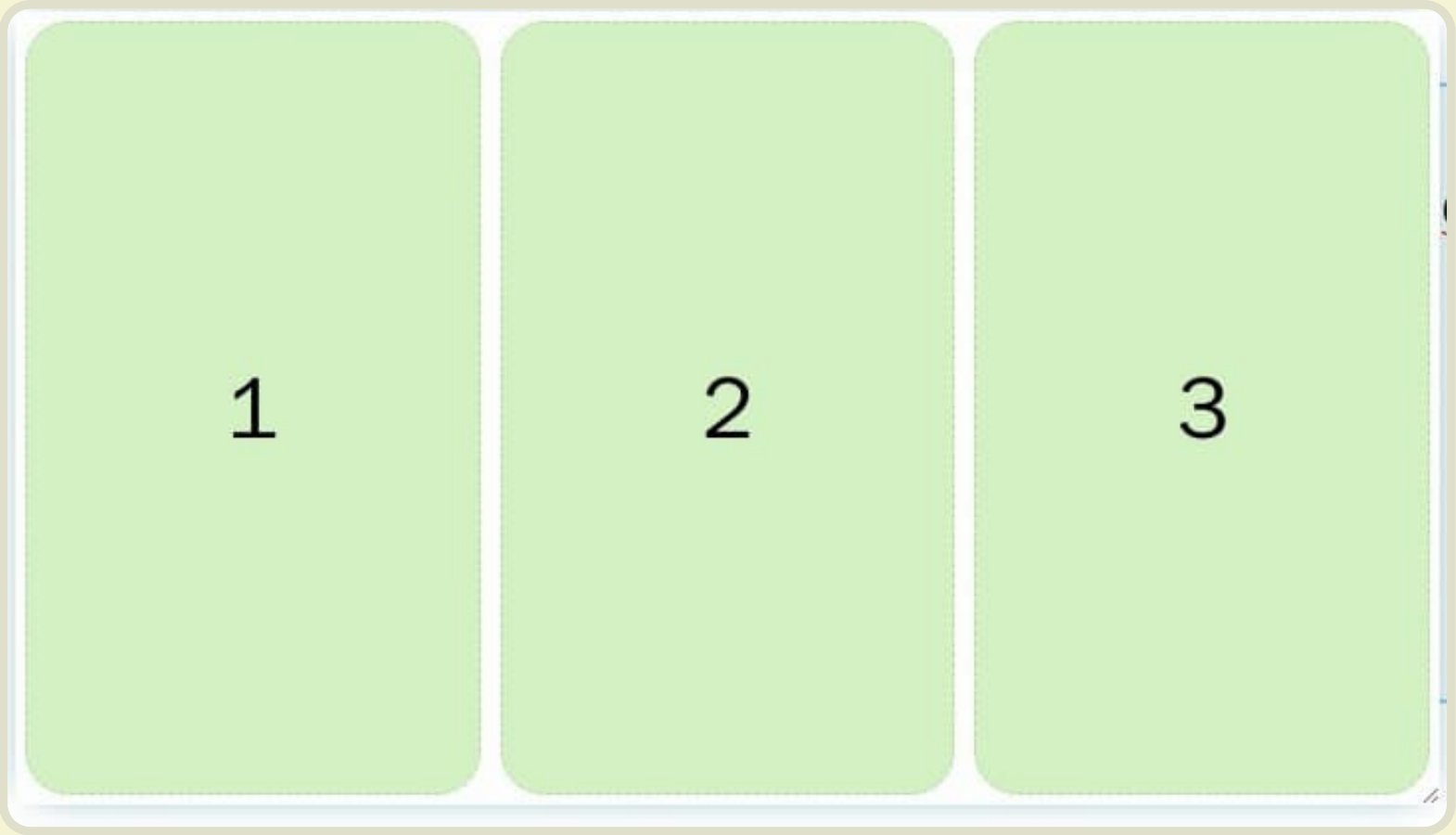
```
flex: <flex-grow> <flex-shrink> <flex-basis>;
```

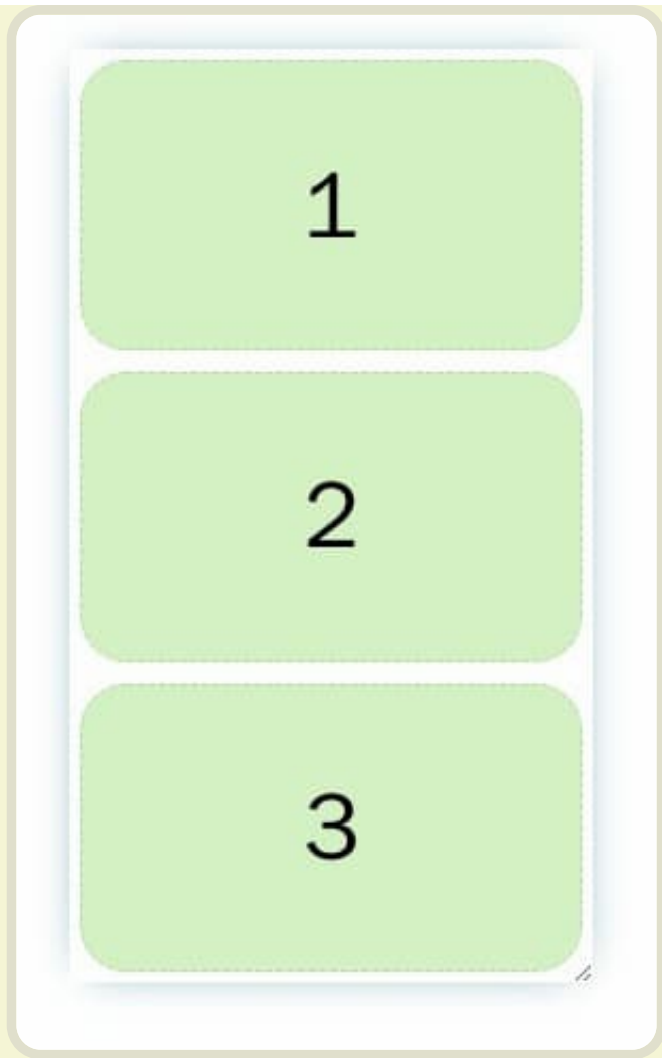
- `flex-basis`：项目的初始宽度。
- `flex-grow`：指定如果有多余宽度，项目是否可以扩大。
- `flex-shrink`：指定如果宽度不足，项目是否可以缩小。

`flex: 0 1 150px;` 的意思就是，项目的初始宽度是150px，且不可以扩大，但是当容器宽度不足150px时，项目可以缩小。

如果写成 `flex: 1 1 150px;`，就表示项目始终会占满所有宽度。



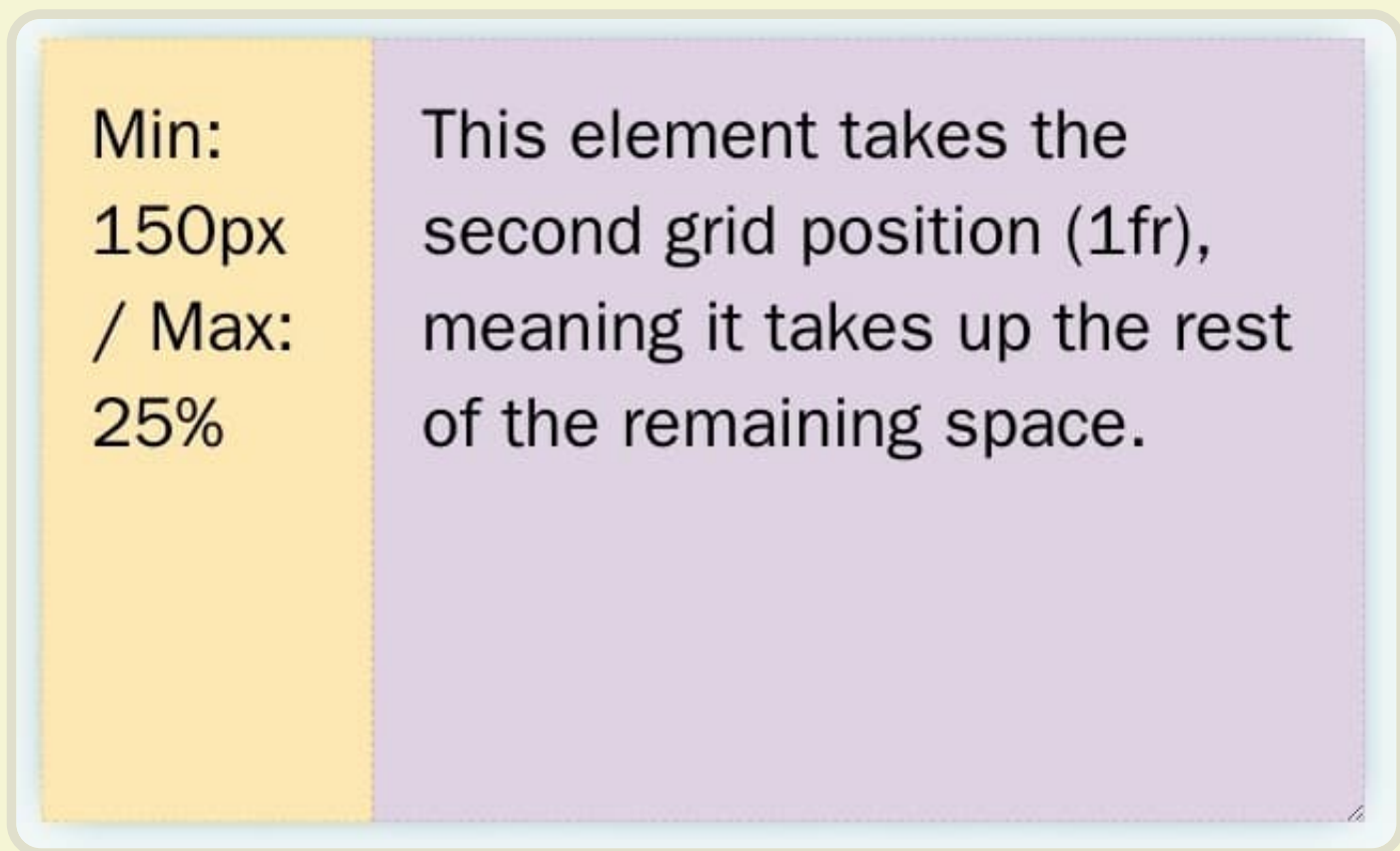




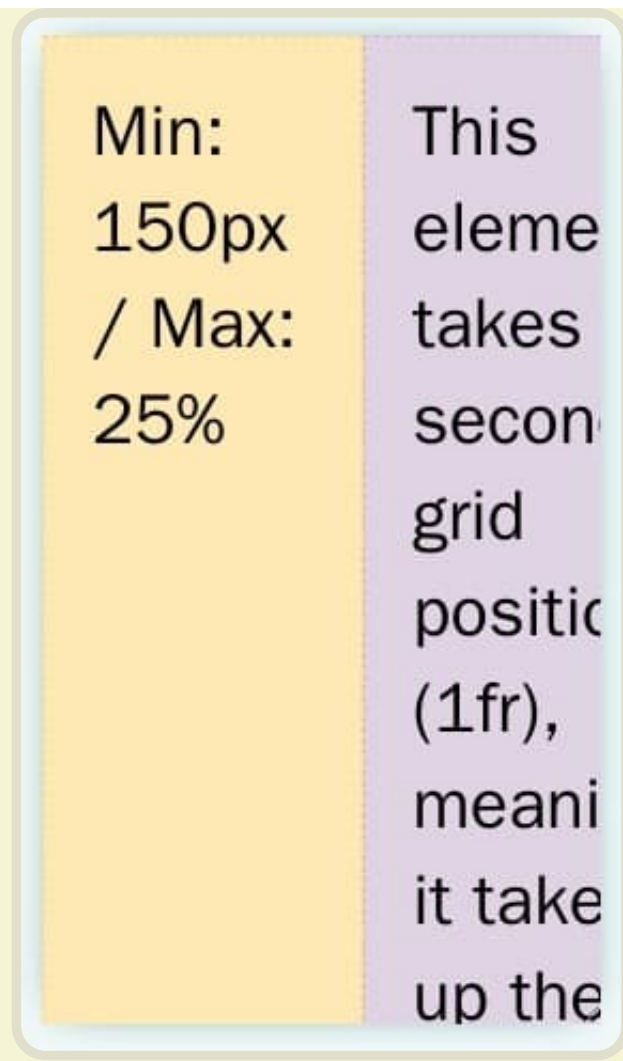
### 三、两栏式布局

---

两栏式布局就是一个边栏，一个主栏。



下面的实现是，边栏始终存在，主栏根据设备宽度，变宽或者变窄。如果希望主栏自动换到下一行，可以参考上面的"并列式布局"。



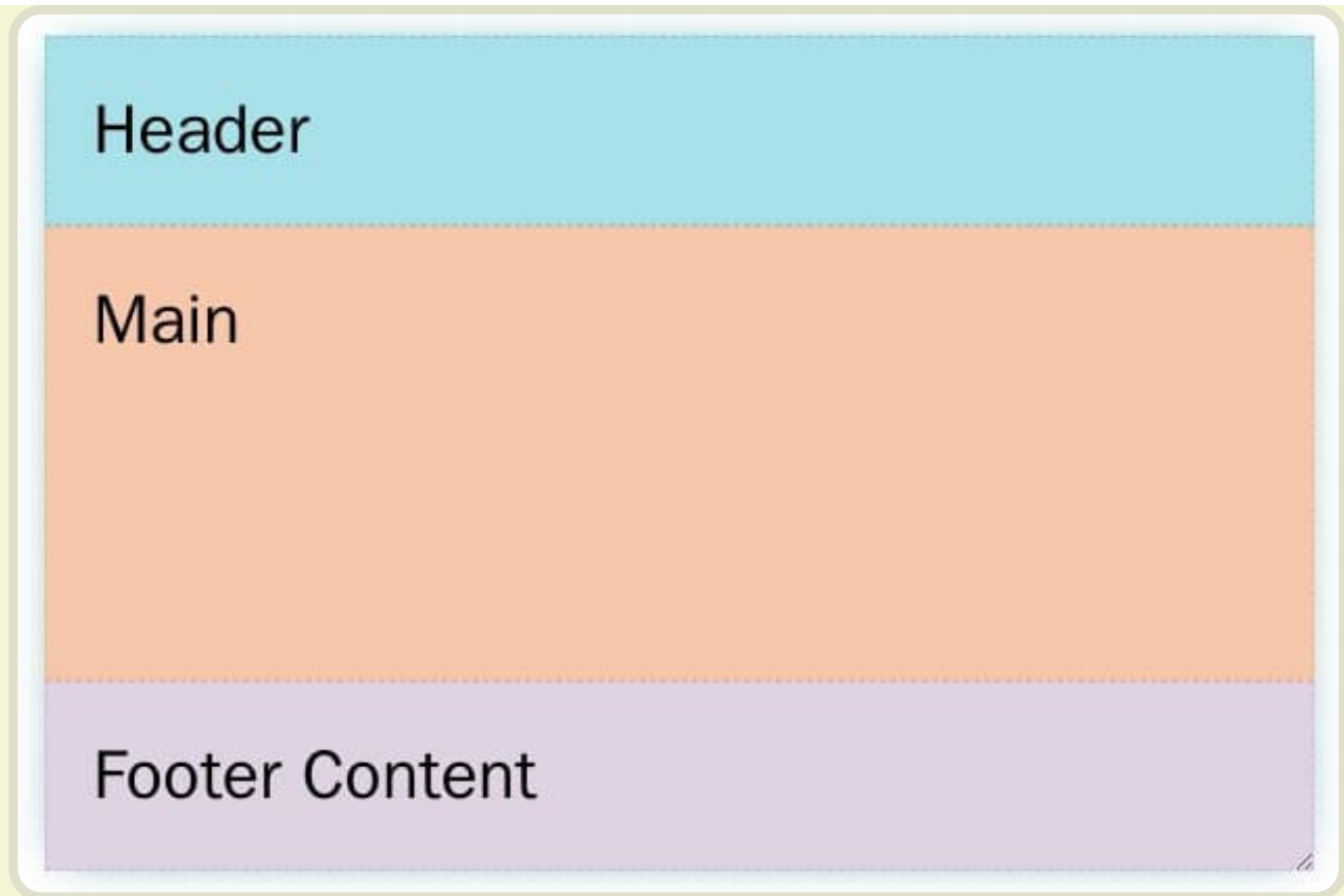
使用 Grid，实现很容易（[CodePen 示例](#)）。

```
.container {  
  display: grid;  
  grid-template-columns: minmax(150px, 25%) 1fr;  
}
```

上面代码中，`grid-template-columns` 指定页面分成两列。第一列的宽度是 `minmax(150px, 25%)`，即最小宽度为 `150px`，最大宽度为总宽度的25%；第二列为 `1fr`，即所有剩余宽度。

## 四、三明治布局

三明治布局指的是，页面在垂直方向上，分成三部分：页眉、内容区、页脚。



这个布局会根据设备宽度，自动适应，并且不管内容区有多少内容，页脚始终在容器底部（粘性页脚）。也就是说，这个布局总是会占满整个页面高度。



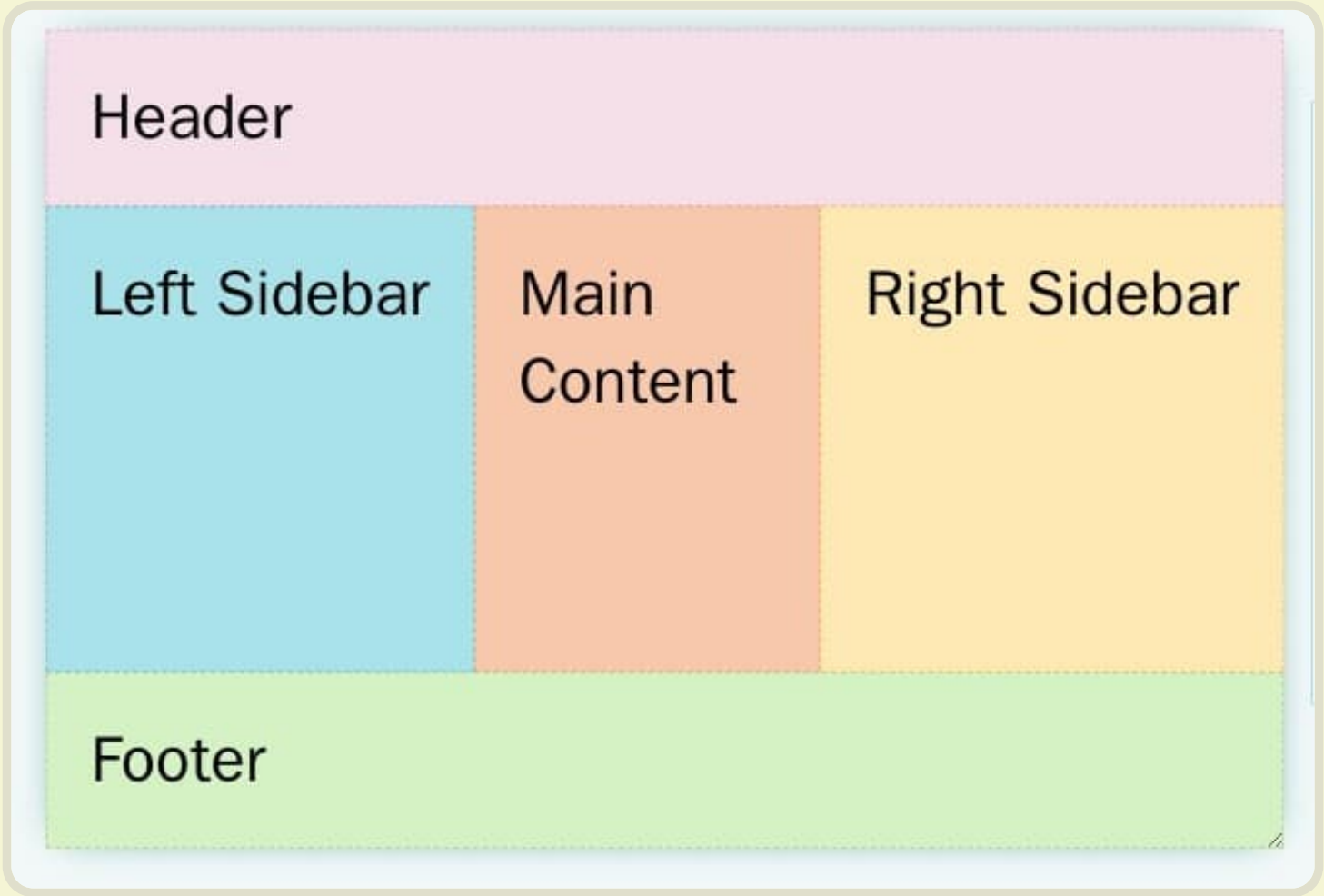
CSS 代码如下（[CodePen 示例](#)）。

```
.container {  
  display: grid;  
  grid-template-rows: auto 1fr auto;  
}
```

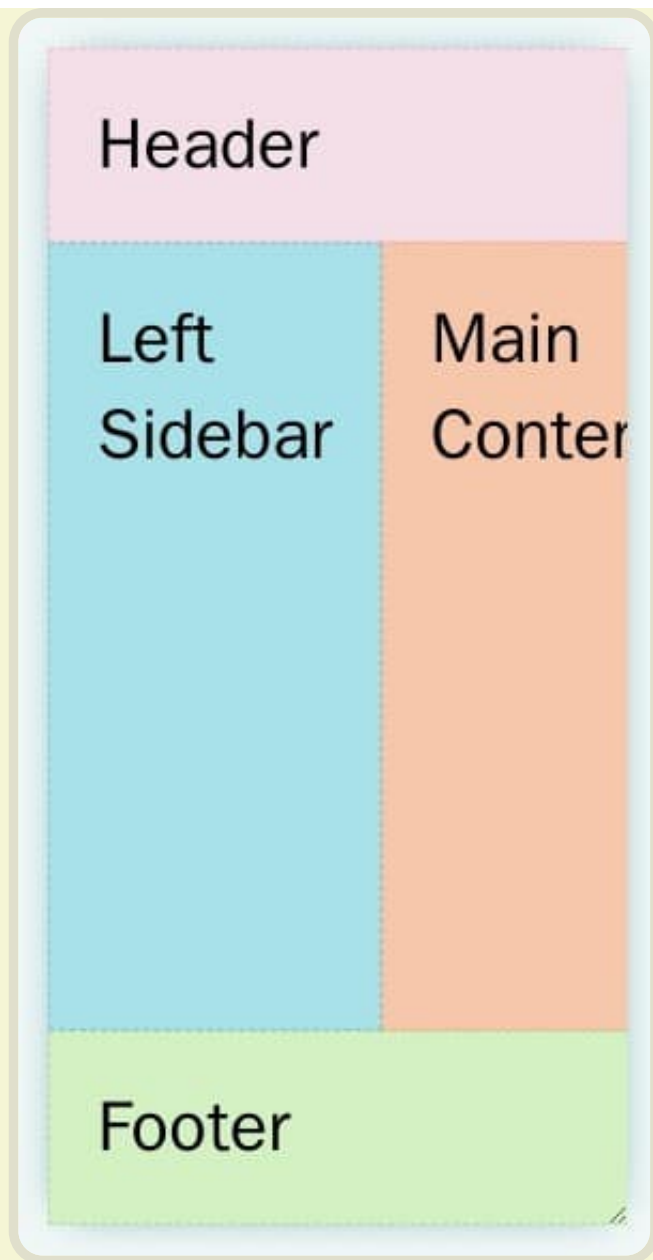
上面代码写在容器上面，指定采用 Grid 布局。核心代码是 `grid-template-rows` 那一行，指定垂直高度怎么划分，这里是从上到下分成三部分。第一部分（页眉）和第三部分（页脚）的高度都为 `auto`，即本来的内容高度；第二部分（内容区）的高度为 `1fr`，即剩余的所有高度，这可以保证页脚始终在容器的底部。

## 五、圣杯布局

圣杯布局是最常用的布局，所以被比喻为圣杯。它将页面分成五个部分，除了页眉和页脚，内容区分成左边栏、主栏、右边栏。



这里的实现是，不管页面宽度，内容区始终分成三栏。如果宽度太窄，主栏和右边栏会看不到。如果想将这三栏改成小屏幕自动堆叠，可以参考并列式布局。



HTML 代码如下。

```
<div class="container">
  <header/>
  <div/>
  <main/>
  <div/>
  <footer/>
</div>
```

CSS 代码如下（[CodePen 示例](#)）。

```
.container {
  display: grid;
  grid-template: auto 1fr auto / auto 1fr auto;
}
```

上面代码要写在容器上面，指定采用 Grid 布局。核心代码是 `grid-template` 属性那一行，它是两个属性 `grid-template-rows`（垂直方向）和 `grid-template-columns`（水平方向）的简写形式。

```
grid-template: <grid-template-rows> / <grid-template-columns>
```

`grid-template-rows` 和 `grid-template-columns` 都是 `auto 1fr auto`，就表示页面在垂直方向和水平方向上，都分成三个部分。第一部分（页眉和左边栏）和第三部分（页脚和右边栏）都是本来的内容高度（或宽度），第二部分（内容区和主

栏) 占满剩余的高度 (或宽度) 。

## 六、参考链接

---

- [Ten modern layouts in one line of CSS](#), Una Kravets
- [Flex 布局教程](#)
- [Grid 布局教程](#)
- [grid-template 属性](#), MDN

## 前端小课

---

看了上面的内容，如果你还想进一步学习更多前端知识，欢迎关注 **极客大学的前端小课**。

前端小课除了 CSS，还讲授 JavaScript，尤其是生产中用得最多的几个框架。这次的授课老师是阿里巴巴前手淘前端团队负责人winter，他可能是目前市场上最大牌的前端讲师。

他结合自己的经验，手把手教大家，如何从零开始自己实现一个类似 React 的简单框架，教你领悟前端框架的原理。它是如何将所有功能封装在一起，暴露接口，给开发者使用，并且还能支持组件。



7天集训营

# winter 手把手带你实现 ToyReact 框架

讲师

程劭非 (winter)

前手机淘宝前端负责人  
前淘宝终端架构组 Leader



你将获得

- 1 了解一个 ToyReact 框架搭建的全过程
- 2 掌握 React 框架背后的原理及实现方式
- 3 掌握 React 中的组件化思想
- 4 亲自实现一个 ToyReact 框架

 **限时优惠 ¥9.8**  
原价 ¥399

优惠名额仅限 **200** 人  
立即扫码报名>>>



整个前端小课是 4 天视频课程 + 3 天实践训练，还有助教随时答疑辅导、班主任每天督促 + 配套实战作业提交，只需要 **¥9.8**！

点击[这里](#)，或者手机扫描下面的二维码就可报名。





(完)

## 文档信息

- 版权声明： 自由转载-非商用-非衍生-保持署名（创意共享3.0许可证）
- 发表日期： 2020年8月10日

## 相关文章

- **2020.12.13:** [《SSH 入门教程》发布了](#)

SSH 是登录 Linux 服务器的必备工具，只要你在做互联网开发，多多少少都会用到它。

- **2020.11.02:** [微信小程序入门教程之四：API 使用](#)

今天是这个系列教程的最后一篇。

- **2020.10.29:** [微信小程序入门教程之三：脚本编程](#)

这个系列教程的前两篇，介绍了小程序的项目结构和页面样式。

- **2020.10.27:** [微信小程序入门教程之二：页面样式](#)

这个系列的上一篇教程，教大家写了一个最简单的 Hello world 微信小程序。



Weibo | Twitter | GitHub

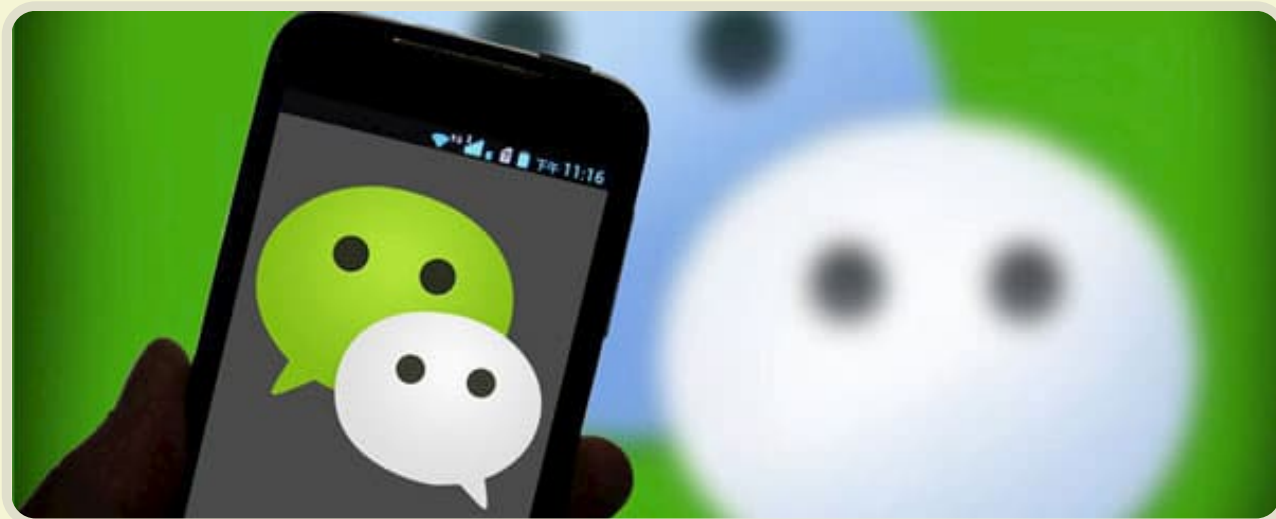
Email: yifeng.ruan@gmail.com

# 微信小程序入门教程之一：初次上手

作者： 阮一峰

日期： 2020年10月26日

微信是中国使用量最大的手机 App 之一，日活跃用户超过3亿，月活跃用户超过11亿（[2019年底统计](#)），市场极大。



2017年，微信正式推出了小程序，允许外部开发者在微信内部运行自己的代码，开展业务。这引发了热烈反响，截止2020年6月，小程序数量已经超过了[550万个](#)。



小程序已经成为国内前端的一个重要业务，跟 Web 和手机 App 有着同等的重要性。小程序开发者供不应求，市场招聘需求极其旺盛，企业都抢着要。

尽管如此，小程序的教程却很缺，要么是不够系统，要么就是跳跃性太大，很多关键的地方寥寥数语，初学者摸不着头脑。我自己学的时候，就苦于找不到好一点的教程。

本文就是我的小程序学习笔记，整理成教程的形式，希望对于初学者有用。需要学会的主要知识点，我都会讲到，我的目标是你读完这个教程，就能学会怎么写小程序。

考虑到很多同学并没有开发经验，小程序是他们接触的第一个开发领域。我会讲得比较

细，希望新人也能没有困难地阅读这个教程。由于内容比较多，这个教程将分成四次连载。



所有示例的完整代码，都可以从 GitHub 的[代码仓库](#)下载。

## 一、小程序是什么？

---

学习小程序之前，先简单说一下，它到底是什么。

字面上讲，小程序就是微信里面的应用程序，外部代码通过小程序这种形式，在微信这个手机 App 里面运行。

但是，更准确的说法是，小程序可以视为只能用微信打开和浏览的网站。小程序和网页的技术模型是一样的，用到的 JavaScript 语言和 CSS 样式也是一样的，只是网页的 HTML 标签被稍微修改成了 WXML 标签。所以，小程序页面本质上就是网页。

小程序的特殊之处在于，虽然是网页，但是它不支持浏览器，所有浏览器的 API 都不能使用，只能用微信提供的 API。这也是为什么小程序只能用微信打开的原因，因为底层全变了。

## 二、小程序的优势

---

小程序最大的优势，就是它基于微信。

微信 App 的功能（比如拍照、扫描、支付等等），小程序大部分都能使用。微信提供了各种封装好的 API，开发者不用自己实现，也不用考虑 iOS 和安卓的平台差异，只要一行代码就可以调用。

而且，开发者也不用考虑用户的注册和登录，直接使用微信的注册和登录，微信的用户自动成为你的用户。

### 三、知识准备

由于小程序基于网页技术，所以学习之前，最好懂一点网页开发。具体来说，下面两方面的知识是必需的。

- (1) JavaScript 语言：懂基本语法，会写简单的 JS 脚本程序。
- (2) CSS 样式：理解如何使用 CSS 控制网页元素的外观。

此外，虽然 HTML 标签和浏览器 API 不是必备知识，但是了解浏览器怎么渲染网页，对于理解小程序模型有很大的帮助。

总的来说，先学网页开发，再学小程序，是比较合理的学习途径，而且网页开发的资料比较多，遇到问题容易查到解决方法。但是，网页开发要学的东西太多，不是短期能掌握的，如果想快速上手，先学小程序，遇到不懂的地方再去查资料，也未尝不可。

### 四、开发准备

小程序开发的第一步，是去[微信公众平台](#)注册，申请一个 AppID，这是免费的。





## 设置

[基本设置](#)[开发设置](#)[第三方授权管理](#)[接口设置](#)[开发者工具](#)

### 开发者ID

开发者ID

操作

AppID(小程序ID)

wxf3563c

AppSecret(小程序密钥)

重置 ?

申请完成以后，你会得到一个 AppID（小程序编号） 和 AppSecret（小程序密钥），后面都会用到。

然后，下载微信提供的[小程序开发工具](#)。这个工具是必需的，因为只有它才能运行和调试小程序源码。

开发者工具支持 Windows 和 MacOS 两个平台。我装的是 Windows（64位）的版本，这个教程的内容也是基于该版本的，但是 MacOS 版本的操作应该是完全一样的。



安装好打开这个软件，会要求你使用微信扫描二维码登录。



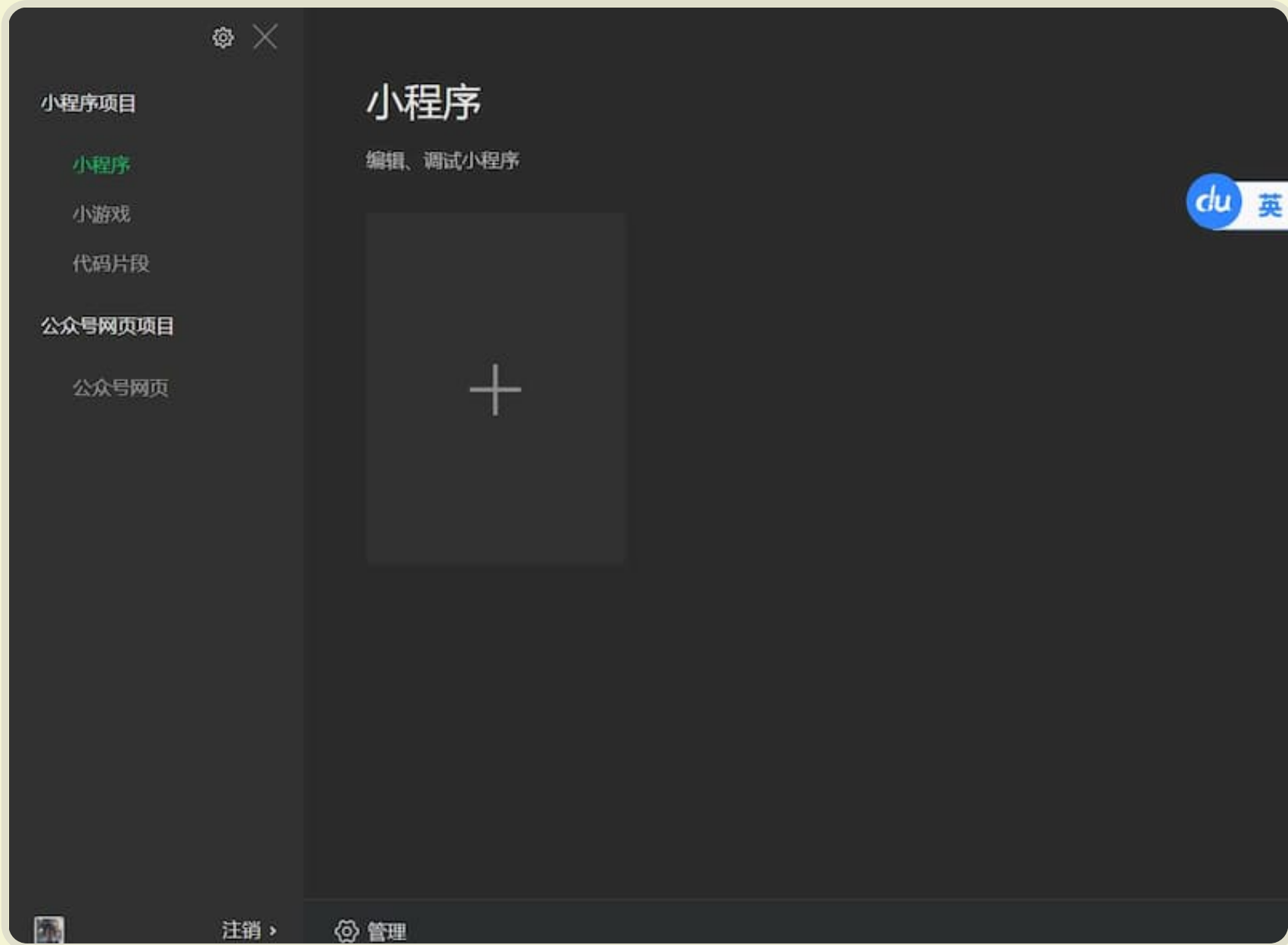
# 微信开发者工具

v1.03.2009140



欢迎使用微信开发者工具

登录后，进入新建项目的页面，可以新建不同的项目，默认是新建小程序项目。



点击右侧的 **+** 号，就跳出了新建小程序的页面。

新建项目

导入项目

项目名称

目录

AppID

若无 AppID 可 [注册](#)  
或使用 [测试号](#) ?

开发模式

小程序

后端服务

☒ 小程序·云开发

小程序·云开发为开发者提供数据库、存储和云函数等完整的云端支持。无需搭建服务器，使用平台提供的 API 进行核心业务开发，即可实现小程序快速上线和迭代。 [了解详情](#)

☐ 不使用云服务

如果直接新建小程序，会生成一个完整的项目脚手架。对于初学者来说，这样反而不利于掌握各个文件的作用。更好的学习方法是，自己从头手写每一行代码，然后切换

到"导入项目"的选项，将其导入到开发者工具。



导入时，需要给小程序起一个名字，并且填写项目代码所在的目录，以及前面申请的 AppID。

## 五、hello world 示例

下面，就请大家动手，跟着写一个最简单的小程序，只要五分钟就能完成。

第一步，新建一个小程序的项目目录。名字可以随便起，这里称为 `wechat-miniprogram-demo`。

你可以在资源管理器里面，新建目录。如果熟悉命令行操作，也可以打开 Windows Terminal（没有的话，需要安装），在里面执行下面的命令，新建并进入该目录。

```
> mkdir wechat-miniprogram-demo
> cd wechat-miniprogram-demo
```

第二步，在该目录里面，新建一个脚本文件 `app.js`。这个脚本用于对整个小程序进行初始化。

`app.js` 内容只有一行代码。

```
App({});
```

上面代码中，`App()` 由小程序原生提供，它是一个函数，表示新建一个小程序实例。它的参数是一个配置对象，用于设置小程序实例的行为属性。这个例子不需要任何配置，所以使用空对象即可。

第三步，新建一个配置文件 `app.json`，记录项目的一些静态配置。

`app.json` 采用 JSON 格式。JSON 是基于 JavaScript 语言的一种数据交换格式，只有五条语法规则，非常简单，不熟悉 JSON 的同学可以参考[这篇教程](#)。



`app.json` 文件的内容，至少必须有一个 `pages` 属性，指明小程序包含哪些页面。

```
{
  "pages": [
    "pages/home/home"
  ]
}
```

上面代码中，`pages` 属性是一个数组，数组的每一项就是一个页面。这个示例中，小程序只有一个页面，所以数组只有一项 `pages/home/home`。

`pages/home/home` 是一个三层的文件路径。

1. 所有页面都放在`pages`子目录里面。
2. 每个页面有一个自己的目录，这里是`pages`下面的`home`子目录，表示这个页面叫做`home`。页面的名字可以随便起，只要对应的目录确实存在即可。
3. 小程序会加载页面目录`pages/home`里面的`home.js`文件，`.js`后缀名可以省略，所以完整的加载路径为`pages/home/home`。`home.js`这个脚本的文件名也可以随便起，但是习惯上跟页面目录同名。

第四步，新建 `pages/home` 子目录。

```
$ mkdir -p pages/home
```

然后，在这个目录里面新建一个脚本文件 `home.js`。该文件的内容如下。

```
Page({});
```

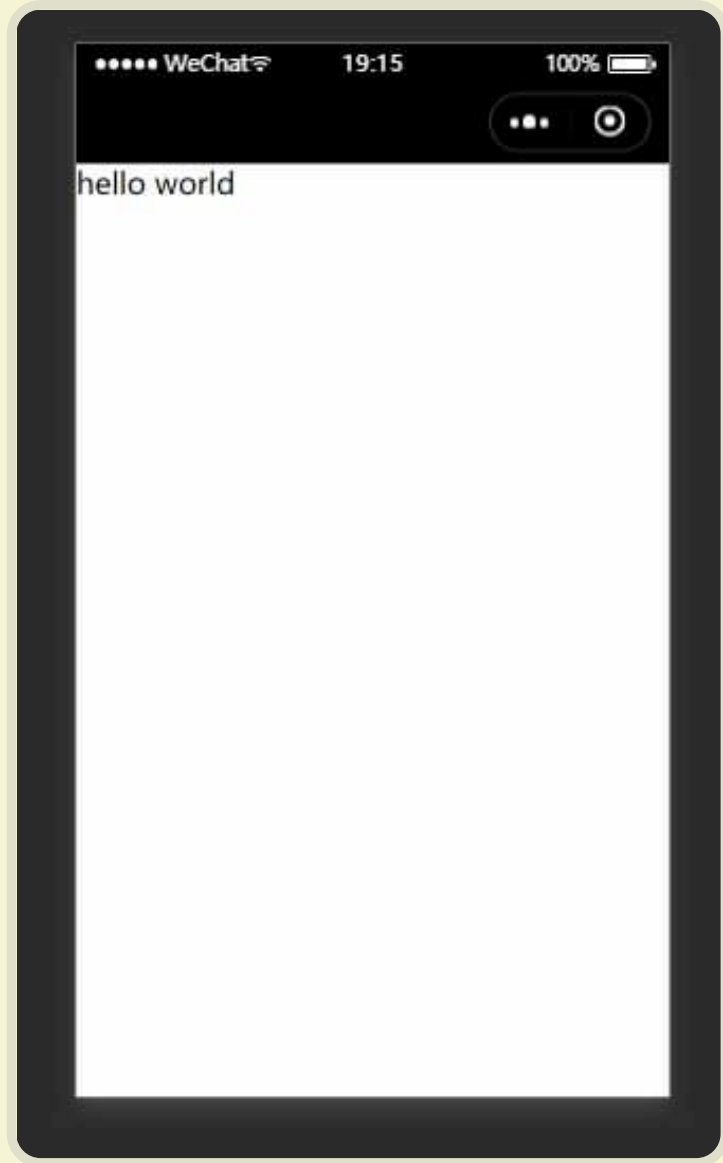
上面代码中，`Page()` 由小程序原生提供，它是一个函数，用于初始化一个页面实例。它的参数是一个配置对象，用于设置当前页面的行为属性。这里是一个空对象，表示不设置任何属性。

第五步，在 `pages/home` 目录新建一个 `home.wxml` 文件。WXML 是微信页面标签语言，类似于 HTML 语言，用于描述小程序的页面。

`home.wxml` 的内容很简单，就写一行 `hello world`。

```
hello world
```

到这一步，就算基本完成了。现在，打开小程序开发工具，导入项目目录 `wechat-miniprogram-demo`。如果一切正常，就可以在开发者工具里面，看到运行结果了。



点击工具栏的"预览"或"真机调试"按钮，还可以在你的手机上面，查看真机运行结果。



这个示例的完整代码，可以到[代码仓库](#)查看。

## 六、WXML 标签语言

上一节的 `home.wxml` 文件，只写了最简单的一行 `hello world`。实际开发中，不会这样写，而是要加上各种标签，以便后面添加样式和效果。

小程序的 WXML 语言提供各种页面标签。下面，对 `home.wxml` 改造一下，加上两个最常用的标签。

```
<view>
  <text>hello world</text>
</view>
```

上面的代码用到了两个标签：`<view>` 和 `<text>`。

`<view>` 标签表示一个区块，用于跟其他区块分隔，类似 HTML 语言的 `<div>` 标签。`<text>` 表示一段行内文本，类似于 HTML 语言的 `<span>` 标签，多个 `<text>` 标签之间不会产生分行。

注意，每个标签都是成对使用，需要有闭合标记，即标签名前加斜杠表示闭合，比如

`<view>` 的闭合标记是 `</view>`。如果缺少闭合标记，小程序编译时会报错。

由于我们还没有为页面添加任何样式，所以页面的渲染效果跟上一节是一样的。后面添加样式时，大家就可以看到标签的巨大作用。

## 七、小程序的项目结构

总结一下，这个示例一共有4个文件，项目结构如下。

```
| - app.json
| - app.js
| - pages
|   | - home
|     | - home.wxml
|     | - home.js
```

这就是最简单、最基本的小程序结构。所有的小程序项目都是这个结构，在上面不断添加其他内容。

这个结构分成两层：描述整体程序的顶层 app 脚本，以及描述各个页面的 page 脚本。

## 八、项目配置文件 app.json

顶层的 `app.json` 文件用于整个项目的配置，对于所有页面都有效。

除了前面提到的必需的 `pages` 属性，`app.json` 文件还有一个 `window` 属性，用来设置小程序的窗口。`window` 属性的值是一个对象，其中有三个属性很常用。

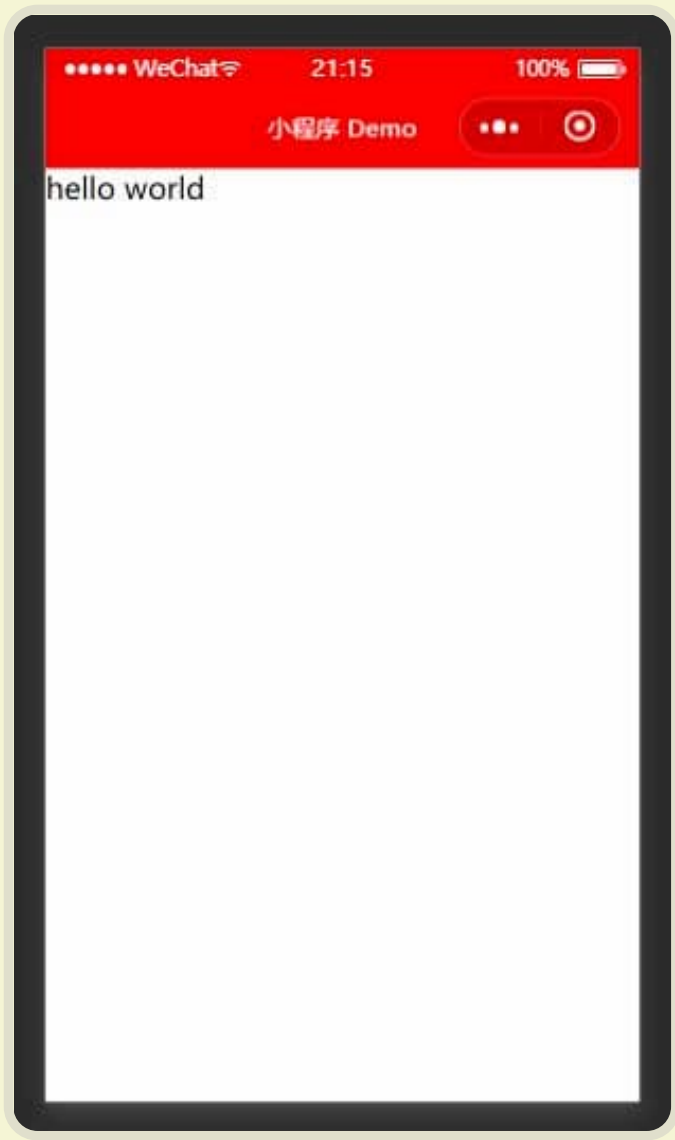
- `navigationBarBackgroundColor`：导航栏的颜色，默认为#000000（黑色）。
- `navigationBarTextStyle`：导航栏的文字颜色，只支持black（黑色）或white（白色），默认为white。
- `navigationBarTitleText`：导航栏的文字，默认为空。

下面，改一下前面的 `app.json`，加入 `window` 属性。

```
{
  "pages": [
    "pages/home/home"
  ],
  "window": {
    "navigationBarBackgroundColor": "#ff0000",
    "navigationBarTextStyle": "white",
    "navigationBarTitleText": "小程序 Demo"
  }
}
```

上面代码中，`window` 属性设置导航栏的背景颜色为红色（`#ff0000`），文本颜色为白色（`white`），标题文字为"小程序 Demo"。

开发者工具导入项目代码，就可以看到导航栏变掉了。



这个示例的完整代码，可以到[代码仓库](#)查看。

除了窗口的样式，很多小程序的顶部或尾部，还有选项栏，可以切换到不同的选项卡。



这个选项栏，也是在 `app.json` 里面设置，使用 `tabBar` 属性，这里就不展开了。

如果你看到了结尾，说明真的对小程序开发非常感兴趣。今天就讲到这里，[下一篇教程](#)将讲解如何设置基本的页面样式，做出用户界面 UI。

(完)

### 文档信息

- 版权声明： 自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期： 2020年10月26日

## 相关文章

---

- **2020.12.13:** [《SSH 入门教程》发布了](#)

SSH 是登录 Linux 服务器的必备工具，只要你在做互联网开发，多多少少都会用到它。

- **2020.11.02:** [微信小程序入门教程之四：API 使用](#)

今天是这个系列教程的最后一篇。

- **2020.10.29:** [微信小程序入门教程之三：脚本编程](#)

这个系列教程的前两篇，介绍了小程序的项目结构和页面样式。

- **2020.10.27:** [微信小程序入门教程之二：页面样式](#)

这个系列的上一篇教程，教大家写了一个最简单的 Hello world 微信小程序。



Weibo | Twitter | GitHub

Email: [yifeng.ruan@gmail.com](mailto:yifeng.ruan@gmail.com)

# 微信小程序入门教程之二：页面样式

作者： 阮一峰

日期： 2020年10月27日

这个系列的[上一篇教程](#)，教大家写了一个最简单的 Hello world 微信小程序。

但是，那只是一个裸页面，并不好看。今天接着往下讲，如何为这个页面添加样式，使它看上去更美观，教大家写出实际可以使用的页面。

所有示例的完整代码，都可以从 GitHub 的[代码仓库](#)下载。



## 一、总体样式

微信小程序允许在顶层放置一个 `app.wxss` 文件，里面采用 CSS 语法设置页面样式。这个文件的设置，对所有页面都有效。

注意，小程序虽然使用 CSS 样式，但是样式文件的后缀名一律要写成 `.wxss`。

打开上一篇教程的示例，在项目顶层新建一个 `app.wxss` 文件，内容如下。

```
page {  
  background-color: pink;  
}  
  
text {  
  font-size: 24pt;  
  color: blue;  
}
```

```
}
```

上面代码将整个页面的背景色设为粉红，然后将 `<text>` 标签的字体大小设为 24 磅，字体颜色设为蓝色。

开发者工具导入代码之后，得到了下面的渲染结果。



可以看到，页面的背景色变成粉红，文本字体变大了，字体颜色变成了蓝色。

实际开发中，直接对 `<text>` 标签设置样式，会影响到所有的文本。一般不这样用，而是通过 `class` 属性区分不同类型的文本，然后再对每种 `class` 设置样式。

打开 `pages/home/home.wxml` 文件，把页面代码改成下面这样。

```
<view>
  <text class="title">hello world</text>
</view>
```

上面代码中，我们为 `<text>` 标签加上了一个 `class` 属性，值为 `title`。

然后，将顶层的 `app.wxss` 文件改掉，不再直接对 `<text>` 设置样式，改成对 `class` 设置样式。

```
page {
  background-color: pink;
}

.title {
  font-size: 24pt;
  color: blue;
}
```



上面代码中，样式设置在 class 上面（`.title`），这样就可以让不同的 `class` 呈现不同的样式。修改之后，页面的渲染结果并不会有变化。

这个示例的完整代码，可以到[代码仓库](#)查看。

## 二、Flex 布局

各种页面元素的位置关系，称为布局（layout），小程序官方推荐使用 Flex 布局。不熟悉这种布局的同学，可以看看我写的[《Flex 布局教程》](#)。

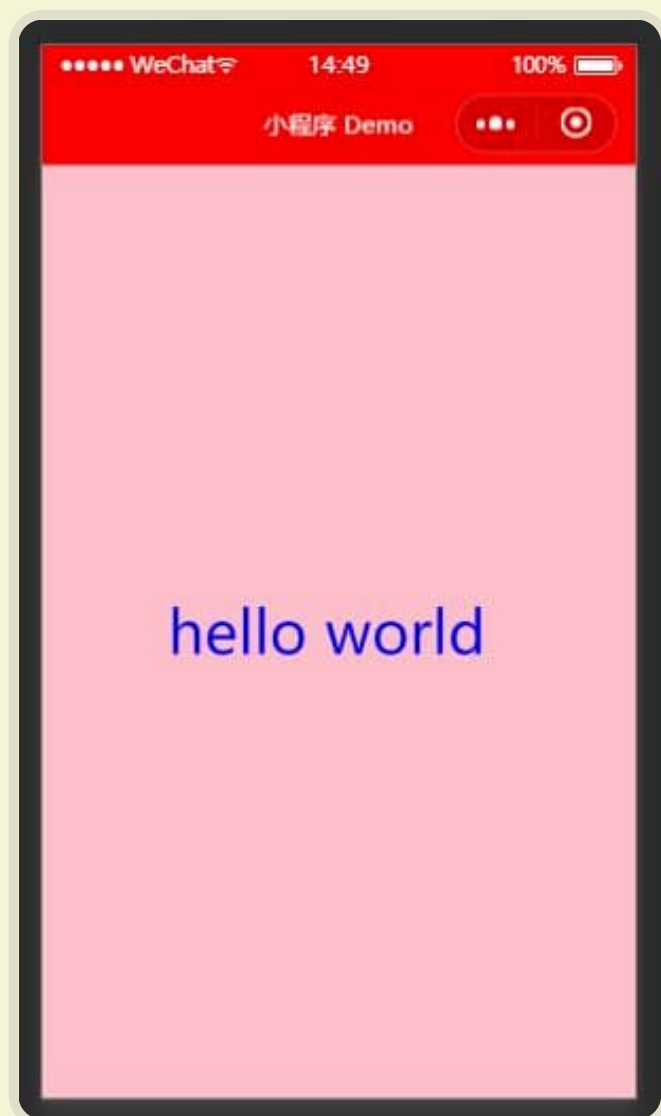
下面演示如何通过 Flex 布局，将上面示例的文本放置到页面中央。

首先，在 `pages/home` 目录里面，新建一个 `home.wxss` 文件，这个文件设置的样式，只对 home 页面生效。这是因为每个页面通常有不一样的布局，所以页面布局一般不写在全局的 `app.wxss` 里面。

然后，`home.wxss` 文件写入下面的内容。

```
page {  
  height: 100%;  
  width: 750rpx;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
}
```

开发者工具导入项目代码，页面渲染结果如下。



下面解释一下上面这段 WXSS 代码，还是很简单的。



(1) `height: 100%;` : 页面高度为整个屏幕高度。

(2) `width: 750rpx;` : 页面宽度为整个屏幕宽度。

注意，这里单位是 `rpx`，而不是 `px`。`rpx` 是小程序为适应不同宽度的手机屏幕，而发明的一种长度单位。不管什么手机屏幕，宽度一律为 `750rpx`。它的好处是换算简单，如果一个元素的宽度是页面的一半，只要写成 `width: 375rpx;` 即可。

(3) `display: flex;` : 整个页面 (page) 采用 Flex 布局。

(4) `justify-content: center;` : 页面的一级子元素 (这个示例是 `<view>`) 水平居中。

(5) `align-items: center;` : 页面的一级子元素 (这个示例是 `<view>`) 垂直居中。一个元素同时水平居中和垂直中央，就相当于处在页面的中央了。

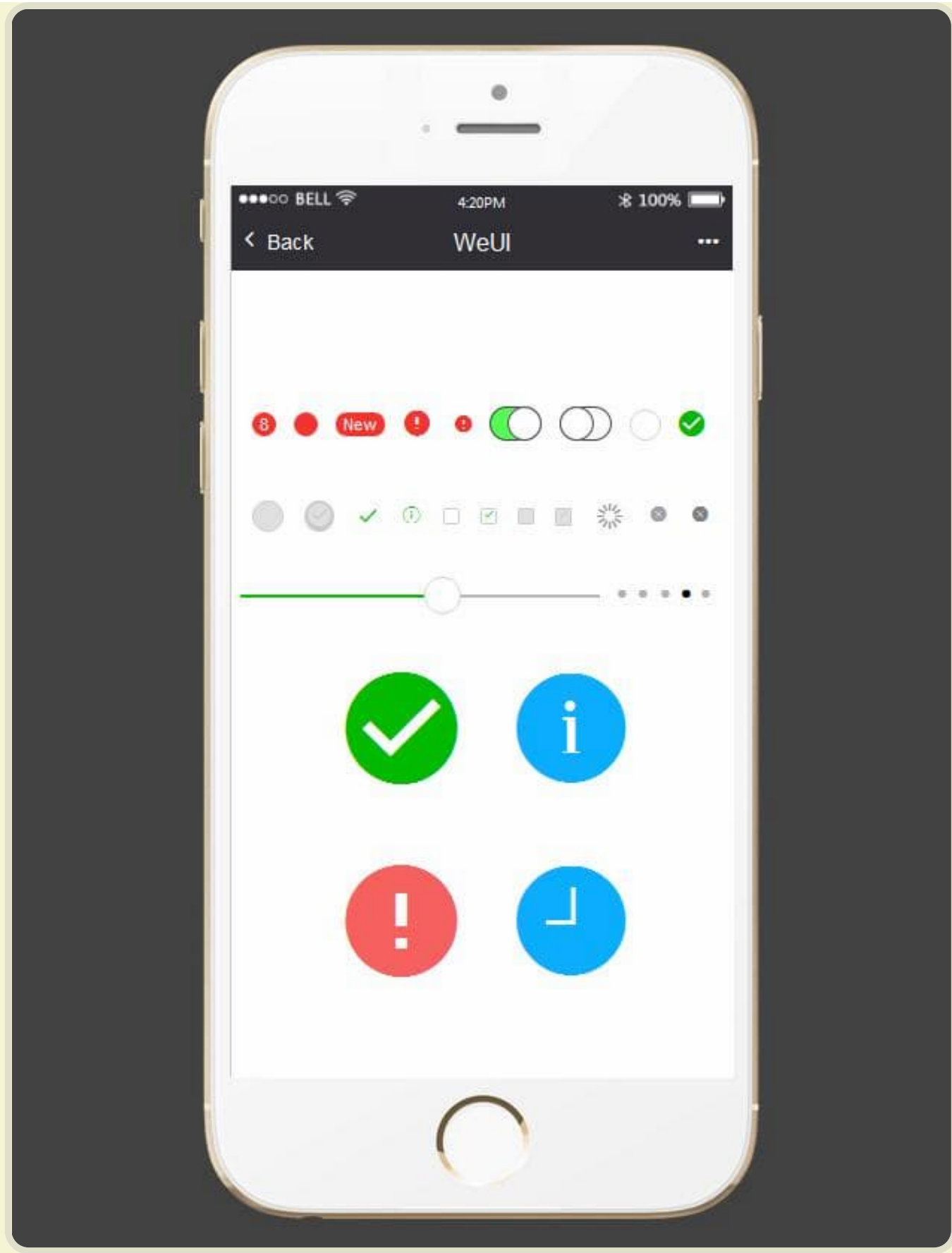
这个示例的完整代码，可以到[代码仓库](#)查看。

## 三、WeUI

---

如果页面的所有样式都自己写，还是挺麻烦的，也没有这个必要。腾讯封装了一套 UI 框架 [WeUI](#)，可以拿来用。

手机访问 [weui.io](#)，可以看到这套 UI 框架的效果。



这一节就来看看，怎么使用这个框架的小程序版本 [WeUI-WXSS](#)，为我们的页面加上官方的样式。

首先，进入它的 [GitHub 仓库](#)，在 `dist/style` 目录下面，找到 `weui.wxss` 这个文件，将[源码](#)全部复制到你的 `app.wxss` 文件的头部。

然后，将 `page/home/home.wxml` 文件改成下面这样。

```
<view>
  <button class="weui-btn weui-btn_primary">
    主操作
  </button>
  <button class="weui-btn weui-btn_primary weui-btn_loading">
    <i class="weui-loading"></i>正在加载
  </button>
  <button class="weui-btn weui-btn_primary weui-btn_disabled">
    禁止点击
  </button>
</view>
```

开发者工具导入项目代码，页面渲染结果如下。



可以看到，加入 WeUI 框架以后，只要为按钮添加不同的 class，就能自动出现框架提供的样式。你可以根据需要，为页面选择不同的按钮。

这个示例中，`<button>` 元素使用了下面的 class 。

- weui-btn：按钮样式的基类
- weui-btn\_primary：主按钮的样式。如果是次要按钮，就使用weui-btn\_default。
- weui-btn\_loading：按钮点击后，操作正在进行中的样式。该类内部需要用*<i>*元素，加上表示正在加载的图标。
- weui-btn\_disabled：按钮禁止点击的样式。

WeUI 提供了大量的元素样式，完整的清单可以查看[这里](#)。

这个示例的完整代码，可以到[代码仓库](#)查看。

## 四、加入图片

美观的页面不能光有文字，还必须有图片。小程序的 `<image>` 组件就用来加载图片。

打开 `home.wxml` 文件，将其改为如下代码。

```
<view>
```

```
<image src="https://picsum.photos/200"></image>
</view>
```

开发者工具加载项目代码，页面的渲染结果如下，可以显示图片了。



`<image>` 组件有[很多属性](#)，比如可以通过 `style` 属性指定样式。

```
<view>
  <image
    src="https://picsum.photos/200"
    style="height: 375rpx; width: 375rpx;"
  ></image>
</view>
```

默认情况下，图片会占满整个容器的宽度（这个例子是 `<view>` 的宽度），上面代码通过 `style` 属性指定图片的高度和宽度（占据页面宽度的一半），渲染结果如下。

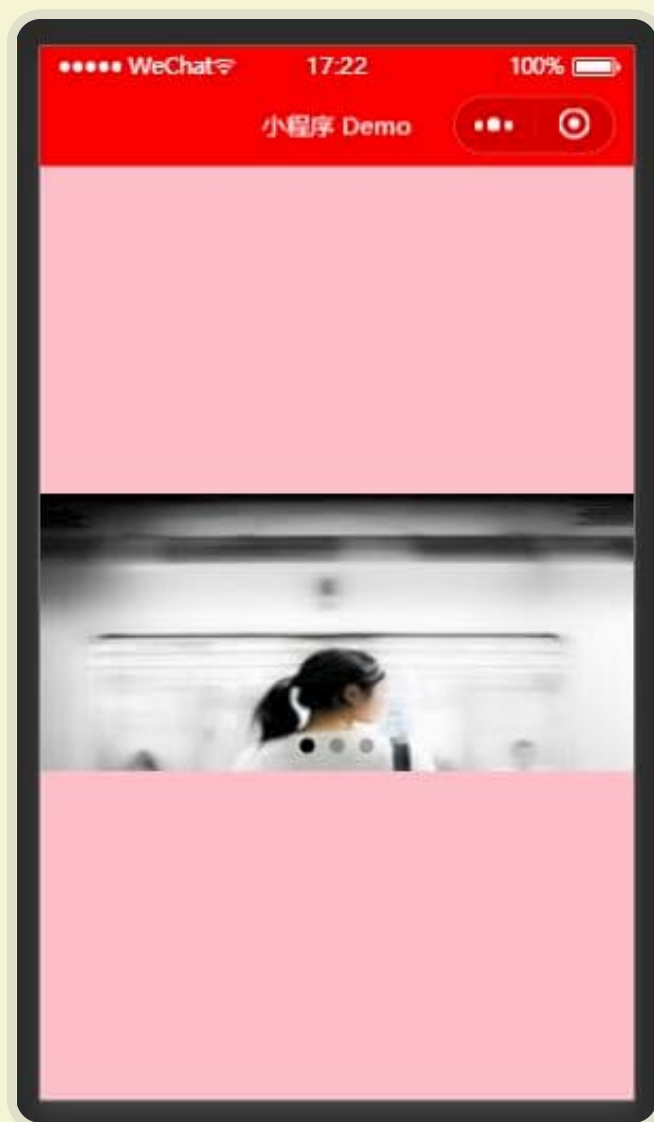


当然，图片样式不一定写在 `<image>` 组件里面，也可以写在 WXSS 样式文件里面。

这个示例的完整代码，可以到[代码仓库](#)查看。

## 五、图片轮播

小程序原生的 `<swiper>` 组件可以提供图片轮播效果。



上面页面的图片上面，有三个提示点，表示一共有三张图片，可以切换显示。

它的代码很简单，只需要改一下 `home.wxml` 文件即可。

```
<view>
  <swiper
    indicator-dots="{{true}}"
    autoplay="{{true}}"
    style="width: 750rpx;">
    <swiper-item>
      <image src="https://picsum.photos/200"></image>
    </swiper-item>
    <swiper-item>
      <image src="https://picsum.photos/250"></image>
    </swiper-item>
    <swiper-item>
      <image src="https://picsum.photos/300"></image>
    </swiper-item>
  </swiper>
</view>
```

上面代码中，`<swiper>` 组件就是轮播组件，里面放置了三个 `<swiper-item>` 组件，表示有三个轮播项目，每个项目就是一个 `<image>` 组件。

`<swiper>` 组件的 `indicator-dots` 属性设置是否显示轮播点，`autoplay` 属性设置是否自动播放轮播。它们的属性值都是一个布尔值，这里要写成 `{{true}}`。这种 `{{...}}` 的语法，表示里面放置的是 JavaScript 代码，这个放在下一次讲解。

这个示例的完整代码，可以到[代码仓库](#)查看。

页面样式就讲到这里，[下一篇教程](#)讲解怎么在微信小程序里面加入 JavaScript 脚本，跟用户互动。

(完)

### 文档信息

- 版权声明： 自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期： 2020年10月27日

## 相关文章

- **2020.12.13:** [《SSH 入门教程》发布了](#)

SSH 是登录 Linux 服务器的必备工具，只要你在做互联网开发，多多少少都会用到它。

- **2020.11.02:** [微信小程序入门教程之四：API 使用](#)

今天是这个系列教程的最后一篇。

- **2020.10.29:** [微信小程序入门教程之三：脚本编程](#)

这个系列教程的前两篇，介绍了小程序的项目结构和页面样式。

■ **2020.10.26:** [微信小程序入门教程之一：初次上手](#)

微信是中国使用量最大的手机 App 之一，日活跃用户超过3亿，月活跃用户超过11亿（2019年底统计），市场极大。

---



[Weibo](#) | [Twitter](#) | [GitHub](#)

Email: [yifeng.ruan@gmail.com](mailto:yifeng.ruan@gmail.com)



# 微信小程序入门教程之三：脚本编程

---

作者： 阮一峰

日期： 2020年10月29日

这个系列教程的前两篇，介绍了小程序的[项目结构](#)和[页面样式](#)。

今天，接着往下讲，教大家为小程序加入 JavaScript 脚本，做出动态效果，以及如何跟用户互动。学会了脚本，就能做出复杂的页面了。

本篇的难度要大于前两篇，如果觉得不好理解，可以先跟着例子，动手做一遍，然后再读文字说明，可能就容易理解了。

所有示例的完整代码，都可以从 GitHub 的[代码仓库](#)下载。



## 一、数据绑定

---

前面的所有示例，小程序的页面都是写死的，也就是页面内容不会变。但是，页面数据其实可以通过脚本传入，通过脚本改变页面，实现动态效果。

小程序提供了一种特别的方法，让页面可以更方便地使用脚本数据，叫做"数据绑定"（data binding）。

所谓"数据绑定"，指的是脚本里面的某些数据，会自动成为页面可以读取的全局变量，两者会同步变动。也就是说，脚本里面修改这个变量的值，页面会随之变化；反过来，页面上修改了这段内容，对应的脚本变量也会随之变化。这也叫做 MVVM 模式。



下面看一个例子。打开 `home.js` 文件，改成下面这样。

```
Page({
  data: {
    name: '张三'
  }
});
```

上面代码中，`Page()` 方法的配置对象有一个 `data` 属性。这个属性的值也是一个对象，有一个 `name` 属性。数据绑定机制规定，`data` 对象的所有属性，自动成为当前页面可以读取的全局变量。也就是说，`home` 页面可以自动读取 `name` 变量。

接着，修改 `home.wxml` 文件，加入 `name` 变量。

```
<view>
  <text class="title">hello {{name}}</text>
</view>
```

上面代码中，`name` 变量写在 `{{...}}` 里面。这是小程序特有的语法，两重大括号表示，内部不是文本，而是 JavaScript 代码，它的执行结果会写入页面。因此，`{{name}}` 表示读取全局变量 `name` 的值，将这个值写入网页。

注意，变量名区分大小写，`name` 和 `Name` 是两个不同的变量名。

开发者工具导入项目代码，页面渲染结果如下。



可以看到，`name` 变量已经自动替换成了变量值“张三”。

这个示例的完整代码，可以查看[代码仓库](#)。

页面和脚本对于变量 `name` 是数据绑定关系，无论哪一方改变了 `name` 的值，另一方也会自动跟着改变。后面讲解到事件时，会有双方联动的例子。

## 二、全局数据

数据绑定只对当前页面有效，如果某些数据要在多个页面共享，就需要写到全局配置对象里面。

打开 `app.js`，改写如下。

```
App({
  globalData: {
    now: (new Date()).toLocaleString()
  }
});
```

上面代码中，`App()` 方法的参数配置对象有一个 `globalData` 属性，这个属性就是我们要在多个页面之间分享的值。事实上，配置对象的任何一个属性都可以共享，这里起名为 `globalData` 只是为了便于识别。

然后，打开 `home.js`，改成下面的内容，在页面脚本里面获取全局对象。

```
const app = getApp();

Page({
  data: {
    now: app.globalData.now
  }
});
```

上面代码中，`getApp()` 函数是小程序原生提供的函数方法，用于从页面获取 `App` 实例对象。拿到实例对象以后，就能从它上面拿到全局配置对象的 `globalData` 属性，从而把 `app.globalData.now` 赋值给页面脚本的 `now` 属性，进而通过数据绑定机制，变成页面的全局变量 `now`。

最后，修改一下页面代码 `home.wxml`。

```
<view>
  <text class="title">现在是 {{now}}</text>
</view>
```

开发者工具导入项目代码，页面渲染结果如下。



可以看到，页面读到了全局配置对象 `app.js` 里面的数据。

这个示例的完整代码，可以查看[代码仓库](#)。

## 三、事件

事件是小程序跟用户互动的主要手段。小程序通过接收各种用户事件，执行回调函数，做出反应。

小程序的[常见事件](#)有下面这些。

- `tap`：触摸后马上离开。
- `longpress`：触摸后，超过 350ms 再离开。如果指定了该事件的回调函数并触发了该事件，`tap`事件将不被触发。
- `touchstart`：触摸开始。
- `touchmove`：触摸后移动。
- `touchcancel`：触摸动作被打断，如来电提醒，弹窗等。
- `touchend`：触摸结束。

上面这些事件，在传播上分成两个阶段：先是捕获阶段（由上层元素向下层元素传播），然后是冒泡阶段（由下层元素向上层元素传播）。所以，同一个事件在同一个元素上面其实会触发两次：捕获阶段一次，冒泡阶段一次。详细的介绍，请参考我写的[事件模型解释](#)。

小程序允许页面元素，通过属性指定各种事件的回调函数，并且还能够指定是哪个阶段触发回调函数。具体方法是为事件属性名加上不同的前缀。小程序提供四种前缀。

- capture-bind: 捕获阶段触发。
- capture-catch: 捕获阶段触发，并中断事件，不再向下传播，即中断捕获阶段，并取消随后的冒泡阶段。
- bind: 冒泡阶段触发。
- catch: 冒泡阶段触发，并取消事件进一步向上冒泡。

下面通过一个例子，来看如何为事件指定回调函数。打开 `home.wxml` 文件，改成下面的代码。

```
<view>
  <text class="title">hello {{name}}</text>
  <button bind:tap="buttonHandler">点击</button>
</view>
```

上面代码中，我们为页面加上了一个按钮，并为这个按钮指定了触摸事件（`tap`）的回调函数 `buttonHandler`，`bind:` 前缀表示这个回调函数会在冒泡阶段触发（前缀里面的冒号可以省略，即写成 `bindtap` 也可以）。

回调函数必须在页面脚本中定义。打开 `home.js` 文件，改成下面的代码。

```
Page({
  data: {
    name: '张三'
  },
  buttonHandler(event) {
    this.setData({
      name: '李四'
    });
  }
});
```

上面代码中，`Page()` 方法的参数配置对象里面，定义了 `buttonHandler()`，这就是 `<button>` 元素的回调函数。它有几个地方需要注意。

(1) 事件回调函数的参数是事件对象 `event`，可以从它上面获取[事件信息](#)，比如事件类型、发生时间、发生节点、当前节点等等。

(2) 事件回调函数内部的 `this`，指向页面实例。

(3) 页面实例的 `this.setData()` 方法，可以更改配置对象的 `data` 属性，进而通过数据绑定机制，导致页面上的全局变量发生变化。

开发者工具导入项目代码，点击按钮后，页面渲染结果如下。



可以看到，点击按钮以后，页面的文字从"hello 张三"变成了"hello 李四"。

这个示例的完整代码，可以查看[代码仓库](#)。

这里要强调一下，修改页面配置对象的 `data` 属性时，不要直接修改 `this.data`，这不仅无法触发事件绑定机制去变更页面，还会造成数据不一致，所以一定要通过 `this.setData()` 去修改。`this.setData()` 是一个很重要的方法，有很多细节，详细介绍可以读一下[官方文档](#)。

## 四、动态提示 Toast

小程序提供了很多组件和方法，用来增强互动效果。比如，每次操作后，都显示一个动态提示，告诉用户操作的结果，这种效果叫做 Toast。

打开 `home.js` 文件，为 `this.setData()` 加上第二个参数。

```
Page({
  data: {
    name: '张三'
  },
  buttonHandler(event) {
    this.setData({
      name: '李四'
    }, function () {
      wx.showToast({
        title: '操作完成',
        duration: 700
      });
    });
  },
})
```

上面代码中，`this.setData()` 方法加入了第二个参数，这是一个函数，它会在页面变更完毕后（即 `this.setData()` 执行完）自动调用。

这个参数函数内部，调用了 `wx.showToast()` 方法，`wx` 是小程序提供的原生对象，所有客户端 API 都定义在这个对象上面，`wx.showToast()` 会展示微信内置的动态提示框，它的参数对象的 `title` 属性指定提示内容，`duration` 属性指定提示框的展示时间，单位为毫秒。

开发者工具导入项目代码，点击按钮后，页面渲染结果如下。



过了700毫秒，提示框就会自动消失。

这个示例的完整代码，可以查看[代码仓库](#)。

## 五、对话框 Modal

下面，我们再用小程序提供的 `wx.showModal()` 方法，制作一个对话框，即用户可以选择"确定"或"取消"。

打开 `home.js` 文件，修改如下。

```
Page({
  data: {
    name: '张三'
  },
  buttonHandler(event) {
    const that = this;
    wx.showModal({
      title: '操作确认',
      content: '你确认要修改吗?',
      success(res) {
        if (res.confirm) {
```

```

        that.setData({
          name: '李四'
        }, function () {
          wx.showToast({
            title: '操作完成',
            duration: 700
          });
        });
      } else if (res.cancel) {
        console.log('用户点击取消');
      }
    }
  });
}
});
});

```

上面代码中，用户点击按钮以后，回调函数 `buttonHandler()` 里面会调用 `wx.showModal()` 方法，显示一个对话框，效果如下。



`wx.showModal()` 方法的参数是一个配置对象。`title` 属性表示对话框的标题（本例为"操作确认"），`content` 属性表示对话框的内容（本例为"你确认要修改吗？"），`success` 属性指定对话框成功显示后的回调函数，`fail` 属性指定显示失败时的回调函数。

`success` 回调函数里面，需要判断一下用户到底点击的是哪一个按钮。如果参数对象的 `confirm` 属性为 `true`，点击的就是"确定"按钮，`cancel` 属性为 `true`，点击的就是"取消"按钮。

这个例子中，用户点击"取消"按钮后，对话框会消失，控制台会输出一行提示信息。点击"确定"按钮后，对话框也会消失，并且还会去调用 `that.setData()` 那些逻辑。

注意，上面代码写的是 `that.setData()`，而不是 `this.setData()`。这是因为 `setData()` 方法定义在页面实例上面，但是由于 `success()` 回调函数不是直接定义在 `Page()` 的配置对象下面，`this` 不会指向页面实例，导致



`this.setData()` 会报错。解决方法就是在 `buttonHandler()` 的开头，将 `this` 赋值给变量 `that`，然后在 `success()` 回调函数里面使用 `that.setData()` 去调用。关于 `this` 更详细的解释，可以参考[这篇教程](#)。

这个示例的完整代码，可以查看[代码仓库](#)。

今天的教程就到这里，对于初学者来说，已经讲了很多东西，可能需要慢慢消化。如果对网页开发和 JavaScript 语言不熟悉，你也许会觉得不容易完全理解，不用担心，初学者只需要知道加入脚本的方法，以及脚本可以达到的效果就可以了，后面做到实际的项目，慢慢就会加深理解。

有了脚本以后，就可以通过小程序 API，去调用微信的各种内置能力。[下一篇文章](#)将重点讲解如何使用小程序 API。

(完)

## 文档信息

- 版权声明：自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期：2020年10月29日

## 相关文章

- **2020.12.13:** [《SSH 入门教程》发布了](#)

SSH 是登录 Linux 服务器的必备工具，只要你在做互联网开发，多多少少都会用到它。

- **2020.11.02:** [微信小程序入门教程之四：API 使用](#)

今天是这个系列教程的最后一篇。

- **2020.10.27:** [微信小程序入门教程之二：页面样式](#)

这个系列的上一篇文章，教大家写了一个最简单的 Hello world 微信小程序。

- **2020.10.26:** [微信小程序入门教程之一：初次上手](#)

微信是中国使用量最大的手机 App 之一，日活跃用户超过3亿，月活跃用户超过11亿（2019年底统计），市场极大。



Weibo | Twitter | GitHub

Email: [yifeng.ruan@gmail.com](mailto:yifeng.ruan@gmail.com)



# 微信小程序入门教程之四：API 使用

作者： 阮一峰

日期： 2020年11月 2日

今天是这个系列教程的最后一篇。

[上一篇教程](#)介绍了，小程序页面如何使用 JavaScript 脚本。有了脚本以后，就可以调用微信提供的各种能力（即微信 API），从而做出千变万化的页面。本篇就介绍怎么使用 API。

所有示例的完整代码，都可以从 GitHub 的[代码仓库](#)下载。



## 一、WXML 渲染语法

前面说过，小程序的页面结构使用 WXML 语言进行描述。

WXML 的全称是微信页面标签语言（Weixin Markup Language），它不仅提供了许多功能标签，还有一套自己的语法，可以设置页面渲染的生效条件，以及进行循环处理。

微信 API 提供的数据，就通过 WXML 的渲染语法展现在页面上。比如，`home.js` 里面的数据源是一个数组。

```
Page({  
  data: {
```

```
      items: [ '事项 A', '事项 B', '事项 C' ]  
    }  
  });
```

上面代码中，`Page()` 的参数配置对象的 `data.items` 属性是一个数组。通过数据绑定机制，页面可以读取全局变量 `items`，拿到这个数组。

拿到数组以后，怎样将每一个数组成员展现在页面上呢？WXML 的数组循环语法，就是一个很简便的方法。

打开 `home.wxml`，改成下面的代码。

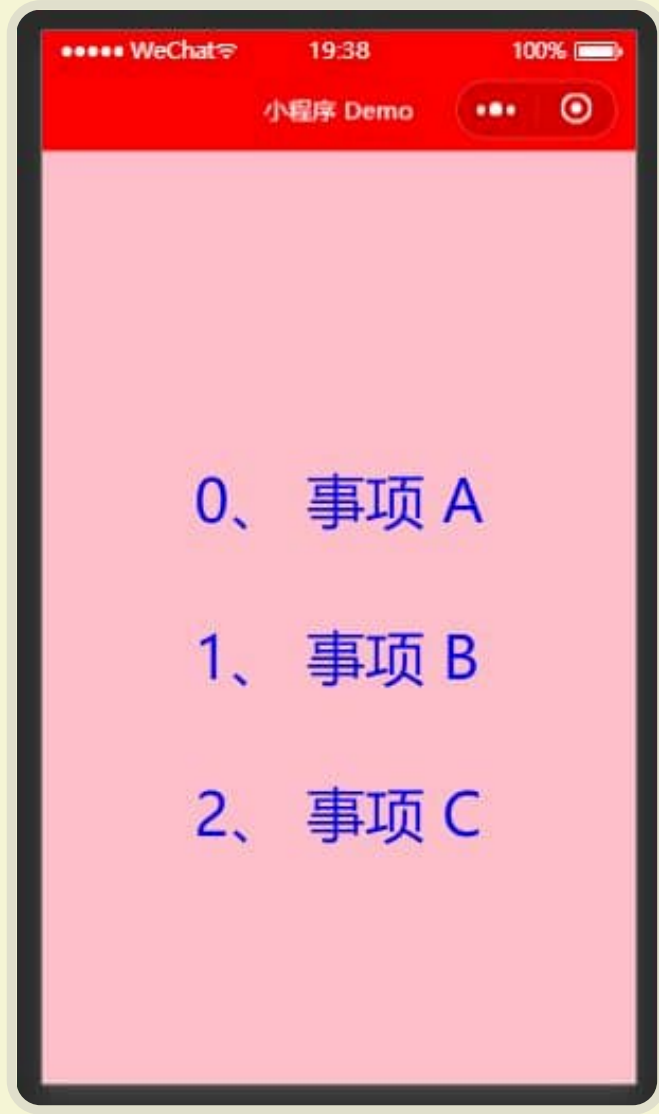
```
<view>  
  <text class="title" wx:for="{{items}}">  
    {{index}}、 {{item}}  
  </text>  
</view>
```

上面代码中，`<text>` 标签的 `wx:for` 属性，表示当前标签（`<text>`）启用数组循环，处理 `items` 数组。数组有多少个成员，就会生成多少个 `<text>`。渲染后的页面结构如下。

```
<view>  
  <text>...</text>  
  <text>...</text>  
  <text>...</text>  
</view>
```

在循环体内，当前数组成员的位置序号（从 `0` 开始）绑定变量 `index`，成员的值绑定变量 `item`。

开发者工具导入项目代码，页面渲染结果如下。



这个示例的完整代码，可以参考[代码仓库](#)。

WXML 的其他渲染语法（主要是条件判断和对象处理），请查看[官方文档](#)。

## 二、客户端数据储存

页面渲染用到的外部数据，如果每次都从服务器或 API 获取，有时可能会比较慢，用户体验不好。

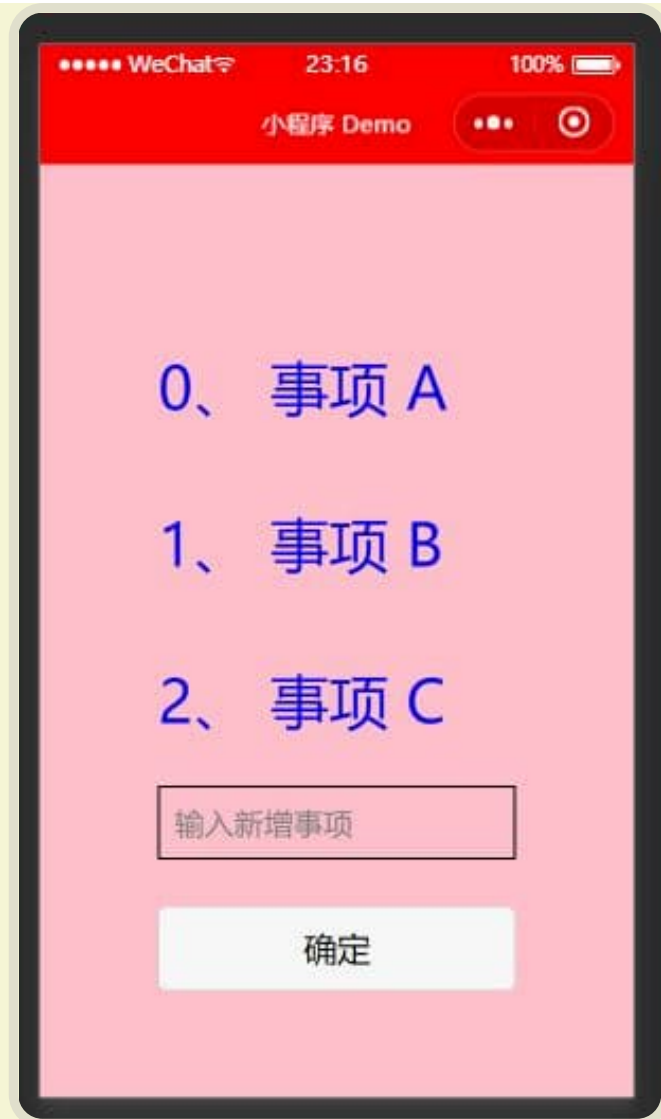
小程序允许将一部分数据保存在客户端（即微信 App）的本地储存里面（其实就是自定义的缓存）。下次需要用到这些数据的时候，就直接从本地读取，这样就大大加快了渲染。本节介绍怎么使用客户端数据储存。

打开 `home.wxml`，改成下面的代码。

```
<view>
  <text class="title" wx:for="{{items}}">
    {{index}}、 {{item}}
  </text>
  <input placeholder="输入新增事项" bind:input="inputHandler"/>
  <button bind:tap="buttonHandler">确定</button>
</view>
```

上面代码除了展示数组 `items`，还新增了一个输入框和一个按钮，用来接受用户的输入。背后的意图是，用户通过输入框，为 `items` 数组加入新成员。

开发者工具导入项目代码，页面渲染结果如下。



注意，输入框有一个 `input` 事件的监听函数 `inputHandler`（输入内容改变时触发），按钮有一个 `tap` 事件的监听函数 `buttonHandler`（点击按钮时触发）。这两个监听函数负责处理用户的输入。

然后，打开 `home.js`，代码修改如下。

```
Page({
  data: {
    items: [],
    inputValue: ''
  },
  inputHandler(event) {
    this.setData({
      inputValue: event.detail.value || ''
    });
  },
  buttonHandler(event) {
    const newItem = this.data.inputValue.trim();
    if (!newItem) return;
    const itemArr = [...this.data.items, newItem];
    wx.setStorageSync('items', itemArr);
    this.setData({ items: itemArr });
  },
  onLoad() {
    const itemArr = wx.getStorageSync('items') || [];
    this.setData({ items: itemArr });
  }
});
```

上面代码中，输入框监听函数 `inputHandler()` 只做了一件事，就是每当用户的输入发生变化时，先从事件对象 `event` 的 `detail.value` 属性上拿到输入的内容，然后将其写入全局变量 `inputValue`。如果用户删除了输入框里面的内容，`inputValue` 就设为空字符串。

按钮监听函数 `buttonHandler()` 是每当用户点击提交按钮，就会执行。它先从

`inputValue` 拿到用户输入的内容，确定非空以后，就将其加入 `items` 数组。然后，使用微信提供的 `wx.setStorageSync()` 方法，将 `items` 数组存储在客户端。最后使用 `this.setData()` 方法更新一下全局变量 `items`，进而触发页面的重新渲染。

`wx.setStorageSync()` 方法属于小程序的客户端数据储存 API，用于将数据写入客户端储存。它接受两个参数，分别是键名和键值。与之配套的，还有一个 `wx.getStorageSync()` 方法，用于读取客户端储存的数据。它只有一个参数，就是键名。这两个方法都是同步的，小程序也提供异步版本，请参考[官方文档](#)。

最后，上面代码中，`Page()` 的参数配置对象还有一个 `onLoad()` 方法。该方法属于页面的生命周期方法，页面加载后会自动执行该方法。它只执行一次，用于页面初始化，这里的意图是每次用户打开页面，都通过 `wx.getStorageSync()` 方法，从客户端取出以前存储的数据，显示在页面上。

这个示例的完整代码，可以参考[代码仓库](#)。

必须牢记的是，客户端储存是不可靠的，随时可能消失（比如用户清理缓存）。用户换了一台手机，或者本机重装微信，原来的数据就丢失了。所以，它只适合保存一些不重要的临时数据，最常见的用途一般就是作为缓存，加快页面显示。

## 三、远程数据请求

---

小程序可以从外部服务器读取数据，也可以向服务器发送数据。本节就来看看怎么使用小程序的网络能力。

微信规定，只有后台登记过的服务器域名，才可以进行通信。不过，开发者工具允许开发时放松这个限制。





按照上图，点击开发者工具右上角的三条横线（“详情”），选中“不校验合法域名、web-view（业务域名）、TLS 版本以及 HTTPS 证书”。这样的话，小程序在开发时，就可以跟服务器进行通信了。

下面，我们在本地启动一个开发服务器。为了简单起见，我选用了 [json-server](#) 作为本地服务器，它的好处是只要有一个 JSON 数据文件，就能自动生成 RESTful 接口。

首先，新建一个数据文件 `db.json`，内容如下。

```
{
  "items": ["事项 A", "事项 B", "事项 C"]
}
```

然后，确认本机安装了 Node.js 以后，进入 `db.json` 所在的目录，在命令行执行下面命令，启动服务器。

```
npx json-server db.json
```

正常情况下，这时你打开浏览器访问 `localhost:3000/items` 这个网址，就能看到返回了一个数组 `["事项 A", "事项 B", "事项 C"]`。

接着，打开 `home.js`，代码修改如下。

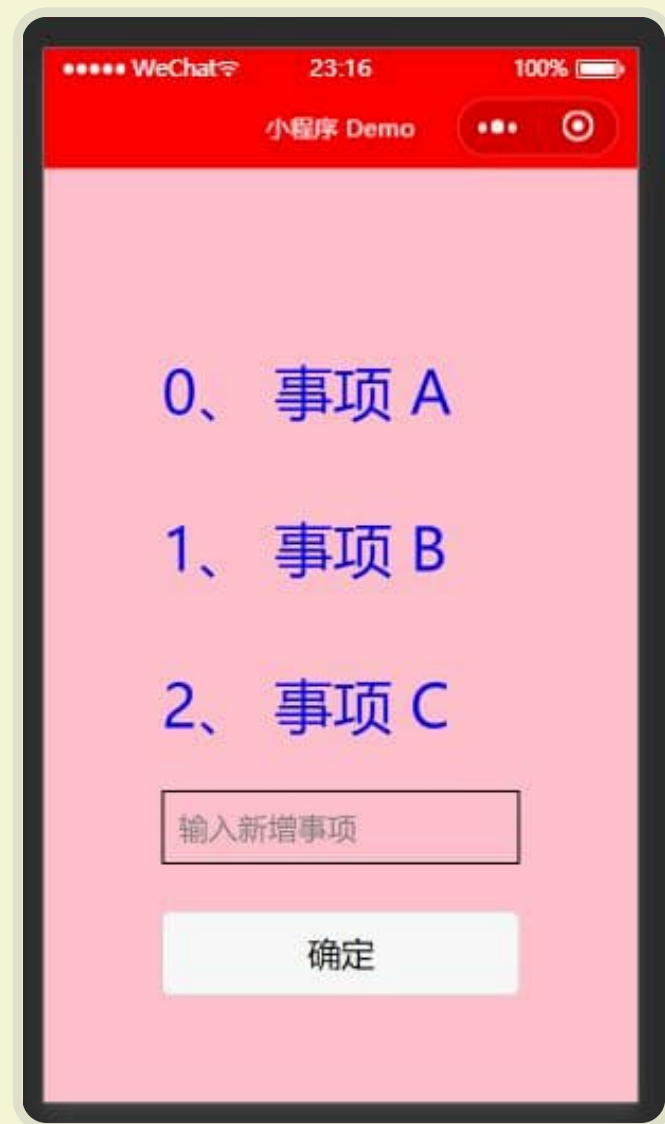
```
Page({
  data: { items: [] },
```

```
onLoad() {
  const that = this;
  wx.request({
    url: 'http://localhost:3000/items',
    success(res) {
      that.setData({ items: res.data });
    }
  });
}
```

上面代码中，生命周期方法 `onLoad()` 会在页面加载后自动执行，这时就会执行 `wx.request()` 方法去请求远程数据。如果请求成功，就会执行回调函数 `success()`，更新页面全局变量 `items`，从而让远程数据显示在页面上。

`wx.request()` 方法就是小程序的网络请求 API，通过它可以发送 HTTP 请求。它的参数配置对象最少需要指定 `url` 属性（请求的网址）和 `success()` 方法（服务器返回数据的处理函数）。其他参数请参考[官方文档](#)。

开发者工具导入项目代码，页面渲染结果如下。它的初始数据是从服务器拿到的。



这个示例的完整代码，可以参考[代码仓库](#)。

这个例子只实现了远程数据获取，json-server 实际上还支持数据的新增和删改，大家可以作为练习，自己来实现。

## 四、<open-data>组件

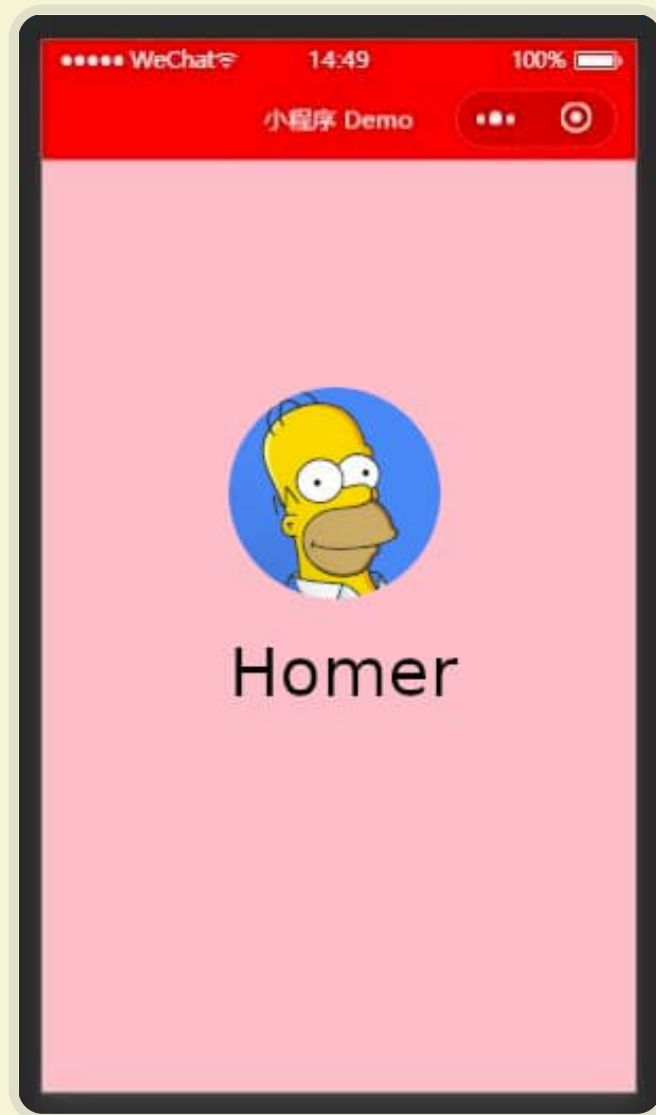
如果要在页面上展示当前用户的身份信息，可以使用小程序提供的 `<open-data>` 组件。

打开 `home.wxml` 文件，代码修改如下。

```
<view>
  <open-data type="userAvatarUrl"></open-data>
  <open-data type="userNickName"></open-data>
</view>
```

上面代码中，`<open-data>` 组件的 `type` 属性指定所要展示的信息类型，`userAvatarUrl` 表示展示用户头像，`userNickName` 表示用户昵称。

开发者工具导入项目代码，页面渲染结果如下，显示你的头像和用户昵称。



`<open-data>` 支持的用户信息如下。

- `userNickName`: 用户昵称
- `userAvatarUrl`: 用户头像
- `userGender`: 用户性别
- `userCity`: 用户所在城市
- `userProvince`: 用户所在省份
- `userCountry`: 用户所在国家
- `userLanguage`: 用户的语言

这个示例的完整代码，可以参考[代码仓库](#)。

`<open-data>` 不需要用户授权，也不需要登录，所以用起来很方便。但也是因为这个原因，小程序不允许用户脚本读取 `<open-data>` 返回的信息。



## 五、获取用户个人信息

如果想拿到用户的个人信息，必须得到授权。官方建议，通过按钮方式获取授权。

打开 `home.wxml` 文件，代码修改如下。

```
<view>
  <text class="title">hello {{name}}</text>
  <button open-type="getUserInfo" bind:getuserinfo="buttonHandler">
    授权获取用户个人信息
  </button>
</view>
```

上面代码中，`<button>` 标签的 `open-type` 属性，指定按钮用于获取用户信息，`bind:getuserinfo` 属性表示点击按钮会触发 `getUserInfo` 事件，即跳出对话框，询问用户是否同意授权。



用户点击"允许"，脚本就可以得到用户信息。

`home.js` 文件的脚本代码如下。

```
Page({
  data: { name: '' },
  buttonHandler(event) {
    if (!event.detail.userInfo) return;
    this.setData({
      name: event.detail.userInfo.nickName
    });
  }
});
```

上面代码中，`buttonHandler()` 是按钮点击的监听函数，不管用户点击"拒绝"或"允许"，都会执行这个函数。我们可以通过事件对象 `event` 有没有 `detail.userInfo` 属性，来判断用户点击了哪个按钮。如果能拿到 `event.detail.userInfo` 属性，就表示用户允许读取个人信息。这个属性是一个对象，里面就是各种用户信息，比如头像、昵称等等。

这个示例的完整代码，可以参考[代码仓库](#)。

实际开发中，可以先用 `wx.getSetting()` 方法判断一下，用户是否已经授权过。如果已经授权过，就不用再次请求授权，而是直接用 `wx.getUserInfo()` 方法获取用户信息。

注意，这种方法返回的用户信息之中，不包括能够真正识别唯一用户的 `openid` 属性。这个属性需要用到保密的小程序密钥去请求，所以不能放在前端获取，而要放在后端。这里就不涉及了。

## 六、多页面的跳转

真正的小程序不会只有一个页面，而是多个页面，所以必须能在页面之间实现跳转。

`app.json` 配置文件的 `pages` 属性就用来指定小程序有多少个页面。

```
{
  "pages": [
    "pages/home/home",
    "pages/second/second"
  ],
  "window": ...
}
```

上面代码中，`pages` 数组包含两个页面。以后每新增一个页面，都必须把页面路径写在 `pages` 数组里面，否则就是无效页面。排在第一位的页面，就是小程序打开时，默认展示的页面。

新建第二个页面的步骤如下。

第一步，新建 `pages/second` 目录。

第二步，在该目录里面，新建文件 `second.js`，代码如下。

```
Page({});
```

第三步，新建第二页的页面文件 `second.wxml`，代码如下。

```
<view>
  <text class="title">这是第二页</text>
  <navigator url="../home/home">前往首页</navigator>
</view>
```

上面代码中，`<navigator>` 就是链接标签，相当于网页标签 `<a>`，只要用户点击就可以跳转到 `url` 属性指定的页面（这里是第一页的位置）。

第四步，修改第一页的页面文件 `home.wxml`，让用户能够点击进入第二页。

```
<view>
  <text class="title">这是首页</text>
  <navigator url="../second/second">前往第二页</navigator>
</view>
```

开发者工具导入项目代码，页面渲染结果如下。



用户点击"前往第二页"，就会看到第二个页面。

这个示例的完整代码，可以参考[代码仓库](#)。

## 七、wx.navigateTo()

除了使用 `<navigator>` 组件进行页面跳转，小程序也提供了页面跳转的脚本方法 `wx.navigateTo()`。

首先，打开 `home.wxml` 文件，代码修改如下。

```
<view>
  <text class="title">这是首页</text>
  <button bind:tap="buttonHandler">前往第二页</button>
</view>
```

开发者工具导入项目代码，页面渲染结果如下。



然后，打开 `home.js` 文件，代码修改如下。

```
Page({
  buttonHandler(event) {
    wx.navigateTo({
      url: '../second/second'
    });
  }
});
```

上面代码中，`buttonHandler()` 是按钮点击的监听函数，只要用户点击按钮，就会调用 `wx.navigateTo()` 方法。该方法的参数是一个配置对象，该对象的 `url` 属性指定了跳转目标的位置，自动跳转到那个页面。

这个示例的完整代码，可以参考[代码仓库](#)。

写到这里，这个小程序入门教程就告一段落了，入门知识基本上都涉及了。下一步，大家可以阅读小程序的[官方教程](#)和[使用文档](#)，争取对小程序 API 有一个整体的把握，然后再去看看各种实际项目的源码，应该就可以动手开发了。以后，我还会写小程序的进阶教程，包括云开发，介绍如何写小程序的后端，下次再见。

(完)

## 文档信息

- 版权声明：自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期：2020年11月 2日

# 相关文章

---

- **2021.05.10:** [软件工程的最大难题](#)

一、引言 大学有一门课程《软件工程》，研究如何组织和管理软件项目。

- **2020.12.13:** [《SSH 入门教程》发布了](#)

SSH 是登录 Linux 服务器的必备工具，只要你在做互联网开发，多多少少都会用到它。

- **2020.10.29:** [微信小程序入门教程之三：脚本编程](#)

这个系列教程的前两篇，介绍了小程序的项目结构和页面样式。

- **2020.10.27:** [微信小程序入门教程之二：页面样式](#)

这个系列的上一篇教程，教大家写了一个最简单的 Hello world 微信小程序。

---



[Weibo](#) | [Twitter](#) | [GitHub](#)

Email: [yifeng.ruan@gmail.com](mailto:yifeng.ruan@gmail.com)

# CSS使用技巧

---

作者： 阮一峰

日期： 2010年3月31日

最近，我开始升级网志了。

在修改模板的过程中，需要重写CSS样式表。正好看到[instantshift.com](http://instantshift.com)有一篇CSS常用技巧的总结文章，我就把它整理出来，供自己参考，也希望对大家有用。

未来，本文将持续更新。



## 1. 文字的水平居中

将一段文字置于容器的水平中点，只要设置text-align属性即可：

```
text-align:center;
```

## 2. 容器的水平居中

先为该容器设置一个明确宽度，然后将margin的水平值设为auto即可。

```
div#container {  
    width:760px;  
    margin:0 auto;  
}
```

## 3. 文字的垂直居中

单行文字的垂直居中，只要将行高与容器高设为相等即可。

比如，容器中有一行数字。

```
<div id="container">1234567890</div>
```

然后CSS这样写：

```
div#container {height: 35px; line-height: 35px;}
```

如果有n行文字，那么将行高设为容器高度的n分之一即可。

#### 4. 容器的垂直居中

比如，有一大一小两个容器，请问如何将小容器[垂直居中](#)？

```
<div id="big">
  <div id="small">
  </div>
</div>
```

首先，将大容器的定位为relative。

```
div#big{
  position:relative;
  height:480px;
}
```

然后，将小容器定位为absolute，再将它的左上角沿y轴下移50%，最后将它margin-top上移本身高度的50%即可。

```
div#small {
  position: absolute;
  top: 50%;
  height: 240px;
  margin-top: -120px;
}
```

使用同样的思路，也可以做出水平居中的效果。

#### 5. 图片宽度的自适应

如何使得较大的图片，能够自动适应小容器的宽度？CSS可以这样写：

```
img {max-width: 100%}
```

但是IE6不支持max-width，所以遇到IE6时，使用[IE条件注释](#)，将语句改写为：

```
img {width: 100%}
```

## 6. 3D按钮

要使按钮具有[3D效果](#)，只要将它的左上部边框设为浅色，右下部边框设为深色即可。

```
div#button {  
    background: #888;  
    border: 1px solid;  
    border-color: #999 #777 #777 #999;  
}
```

## 7. font属性的快捷写法

font快捷写法的格式为：

```
body {  
    font: font-style font-variant font-weight font-size line-height font-family;  
}
```

所以，

```
body {  
    font-family: Arial, Helvetica, sans-serif;  
    font-size: 13px;  
    font-weight: normal;  
    font-variant: small-caps;  
    font-style: italic;  
    line-height: 150%;  
}
```

可以被写成：

```
body {  
    font: italic small-caps normal 13px/150% Arial, Helvetica, sans-serif;  
}
```

## 8. link状态的设置顺序

link的四种状态，需要按照下面的前后顺序进行设置：



```
a:link
a:visited
a:hover
a:active
```

## 9. IE条件注释

你可以利用条件注释，设置只对IE产生作用的语句：

```
<!--[if IE]>
    <link rel="stylesheet" type="text/css" href="ie-
stylesheet.css" />
< ![endif]-->
```

还可以区分各种不同的IE版本：

```
<!--[if IE 6]> - targets IE6 only -->
<!--[if gt IE 6]> - targets IE7 and above -->
<!--[if lt IE 6]> - targets IE5.5 and below -->
<!--[if gte IE 6]> - targets IE6 and above -->
<!--[if lte IE 6]> - targets IE6 and below -->
```

## 10. IE6专用语句：方法一

由于IE6不把html视为文档的根元素，所以利用这一点，可以写出只有IE6才能读到的语句：

```
/* the following rules apply only to IE6 */

* html{

}

* html body{

}

* html .foo{

}
```

IE7专用语句则要写成

```
/* the following rules apply only to IE7 */

*+html .foo{

}
```

## 11. IE专用语句：方法二

除了IE6以外，所有浏览器都不能识别属性前的下划线。而除了IE7之外，所有浏览器都不能识别属性前的\*号，因此可以写出只有这两个浏览器才能读到的语句：

```
.element {  
    background: red; /* modern browsers */  
    *background: green; /* IE 7 and below */  
    _background: blue; /* IE6 exclusively */  
}
```

## 12. CSS的优先性

如果同一个容器被多条CSS语句定义，那么哪一个定义[优先](#)呢？

基本规则是：

行内样式 > id样式 > class样式 > 标签名样式

比如，有一个元素：

```
<div id="ID" class="CLASS" style="color:black;"></div>
```

行内样式是最优先的，然后其他设置的优先性，从低到高依次为：

```
div < .class < div.class < #id < div#id < #id.class  
< div#id.class
```

## 13. IE6的min-height

IE6不支持min-height，有两种方法可以解决这个问题：

方法一：

```
.element {  
    min-height: 500px;  
    height: auto !important;  
    height: 500px;  
}
```

共有三条CSS语句，第一句是针对其他浏览器设置最小高度，第三句是针对IE设置最小高度，第二句则是让其他浏览器覆盖第三句的设置。

方法二：

```
.element {
    min-height: 500px
    _height: 500px
}
```

\_height只有IE6能读取。

#### 14. font-size基准

浏览器的缺省字体大小是16px，你可以先将基准字体大小设为10px：

```
body {font-size:62.5%;}
```

后面统一采用em作为字体单位，2.4em就表示24px。

```
h1 {font-size: 2.4 em}
```

#### 15. Text-transform和Font Variant

Text-transform用于将所有字母变成小写字母、大写字母或首字母大写：

```
p {text-transform: uppercase}
p {text-transform: lowercase}
p {text-transform: capitalize}
```

Font Variant用于将字体变成小型的大写字母（即与小写字母等高的大写字母）。

```
p {font-variant: small-caps}
```

#### 16. CSS重置

CSS重置用于取消浏览器的内置样式，请参考[YUI](#)和[Eric Meyer](#)的样式表。

#### 17. 用图片充当列表标志

默认情况下，浏览器使用一个黑圆圈作为列表标志，可以用图片取代它：

```
ul {list-style: none}

ul li {
    background-image: url("path-to-your-image");
    background-repeat: none;
    background-position: 0 0.5em;
}
```

## 18. 透明

将一个容器设为透明，可以使用下面的代码：

```
.element {  
    filter:alpha(opacity=50);  
    -moz-opacity:0.5;  
    -khtml-opacity: 0.5;  
    opacity: 0.5;  
}
```

在这四行CSS语句中，第一行是IE专用的，第二行用于Firefox，第三行用于webkit核心的浏览器，第四行用于Opera。

## 19. CSS三角形

如何使用CSS生成一个三角形？

先编写一个空元素

```
<div class="triangle"></div>
```

然后，将它四个边框中的三个边框设为透明，剩下一个设为可见，就可以生成三角形效果：

```
.triangle {  
    border-color: transparent transparent green  
transparent;  
    border-style: solid;  
    border-width: 0px 300px 300px 300px;  
    height: 0px;  
    width: 0px;  
}
```

## 20. 禁止自动换行

如果你希望文字在一行中显示完成，不要自动换行，CSS命令如下：

```
h1 { white-space:nowrap; }
```

## 21. 用图片替换文字

有时我们需要在标题栏中使用图片，但是又必须保证搜索引擎能够读到标题，CSS语句可以这样写：

```
h1 {
    text-indent:-9999px;
    background:url("h1-image.jpg") no-repeat;
    width:200px;
    height:50px;
}
```

## 22. 获得焦点的表单元素

当一个表单元素获得焦点时，可以将其突出显示：

```
input:focus { border: 2px solid green; }
```

## 23. !important规则

多条CSS语句互相冲突时，具有!important的语句将覆盖其他语句。由于IE不支持!important，所以也可以利用它区分不同的浏览器。

```
h1 {
    color: red !important;
    color: blue;
}
```

上面这段语句的结果是，其他浏览器都显示红色标题，只有IE显示蓝色标题。

## 24. CSS提示框

当鼠标移动到链接上方，会自动出现一个提示框。

```
<a class="tooltip" href="#">链接文字 <span>提示文字
</span></a>
```

CSS这样写：

```
a.tooltip {position: relative}
a.tooltip span {display:none; padding:5px;
width:200px;}
a:hover {background:#fff;} /*background-color is a
must for IE6*/
a.tooltip:hover span{display:inline;
position:absolute;}
```

## 25. 固定位置的页首

当页面滚动时，有时需要页首在位置固定不变，CSS语句可以这样写，效果参见

<http://limpid.nl/lab/css/fixed/header>:

```
body{ margin:0;padding:100px 0 0 0;}

div#header{
    position:absolute;
    top:0;
    left:0;
    width:100%;
    height:<length>;
}

@media screen{
    body>div#header{position: fixed;}
}

* html body{overflow:hidden;}

* html div#content{height:100%;overflow:auto;}
```

IE6的另一种写法（用于固定位置的页脚）：

```
* html #footer {
    position:absolute;
    top:expression((0-(footer.offsetHeight)+
(document.documentElement.clientHeight ?
document.documentElement.clientHeight :
document.body.clientHeight)+(ignoreMe =
document.documentElement.scrollTop ?
document.documentElement.scrollTop :
document.body.scrollTop))+ 'px' );
}
```

## 26. 在IE6中设置PNG图片的透明效果

```
.classname {

    background: url(image.png);

    _background: none;

    _filter:progid:DXImageTransform.Microsoft.AlphaImageLoader
        (src='image.png', sizingMethod='crop');

}
```

## 27. 各类浏览器的专用语句

```
/* IE6 and below */
* html #uno { color: red }

/* IE7 */
*:first-child+html #dos { color: red }

/* IE7, FF, Saf, Opera */
html>body #tres { color: red }

/* IE8, FF, Saf, Opera (Everything but IE 6,7) */
html>/**/body #cuatro { color: red }

/* Opera 9.27 and below, safari 2 */
html:first-child #cinco { color: red }

/* Safari 2-3 */
html[xmlns*=""] body:last-child #seis { color: red }

/* safari 3+, chrome 1+, opera9+, ff 3.5+ */
body:nth-of-type(1) #siete { color: red }

/* safari 3+, chrome 1+, opera9+, ff 3.5+ */
body:first-of-type #ocho { color: red }

/* saf3+, chromel+ */
@media screen and (-webkit-min-device-pixel-ratio:0)
{
    #diez { color: red }
}

/* iPhone / mobile webkit */
@media screen and (max-device-width: 480px) {
    #veintiseis { color: red }
}

/* Safari 2 - 3.1 */
html[xmlns*=""]:root #trece { color: red }

/* Safari 2 - 3.1, Opera 9.25 */
*|html[xmlns*=""] #catorce { color: red }

/* Everything but IE6-8 */
:root *> #quince { color: red }
```

```

/* IE7 */
*+html #dieciocho { color: red }

/* Firefox only. 1+ */
#veinticuatro, x:-moz-any-link { color: red }

/* Firefox 3.0+ */
#veinticinco, x:-moz-any-link, x:default { color:
red }

/***** Attribute Hacks *****/

/* IE6 */
#once { _color: blue }

/* IE6, IE7 */
#doce { *color: blue; /* or #color: blue */ }

/* Everything but IE6 */
#diecisiete { color/**/: blue }

/* IE6, IE7, IE8 */
#diecinueve { color: blue\9; }

/* IE7, IE8 */
#veinte { color/*\**/: blue\9; }

/* IE6, IE7 -- acts as an !important */
#veintesiete { color: blue !ie; } /* string after !
can be anything */

```

## 28. 容器的水平和垂直居中

HTML代码如下：

```

<figure class='logo'>

    <span></span>

    <img class='photo' />

</figure>

```

CSS代码如下：

```

.logo {
    display: block;

```



```

        text-align: center;
        display: block;
        text-align: center;
        vertical-align: middle;
        border: 4px solid #dddddd;
        padding: 4px;
        height: 74px;
        width: 74px; }

.logo * {
    display: inline-block;
    height: 100%;
    vertical-align: middle; }

.logo .photo {
    height: auto;
    width: auto;
    max-width: 100%;
    max-height: 100%; }

```

## 29. CSS阴影

外阴影：

```

.shadow {
    -moz-box-shadow: 5px 5px 5px #ccc;
    -webkit-box-shadow: 5px 5px 5px #ccc;
    box-shadow: 5px 5px 5px #ccc;
}

```

内阴影：

```

.shadow {
    -moz-box-shadow:inset 0 0 10px #000000;
    -webkit-box-shadow:inset 0 0 10px #000000;
    box-shadow:inset 0 0 10px #000000;
}

```

## 30. 取消IE文本框的滚动条

```

textarea { overflow: auto; }

```

## 31. 图片预加载

请参考[3 Ways to Preload Images with CSS, JavaScript, or Ajax](#)。

### 32. CSS重置

请参考[Should You Reset Your CSS?](#)。

(完)

#### 文档信息

- 版权声明： 自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期： 2010年3月31日

## 相关文章

- **2020.12.13:** [《SSH 入门教程》发布了](#)

SSH 是登录 Linux 服务器的必备工具，只要你在做互联网开发，多多少少都会用到它。

- **2020.11.02:** [微信小程序入门教程之四：API 使用](#)

今天是这个系列教程的最后一篇。

- **2020.10.29:** [微信小程序入门教程之三：脚本编程](#)

这个系列教程的前两篇，介绍了小程序的项目结构和页面样式。

- **2020.10.27:** [微信小程序入门教程之二：页面样式](#)

这个系列的上一篇教程，教大家写了一个最简单的 Hello world 微信小程序。



[Weibo](#) | [Twitter](#) | [GitHub](#)

Email: [yifeng.ruan@gmail.com](mailto:yifeng.ruan@gmail.com)

# CSS3常用功能的写法

---

作者： 阮一峰

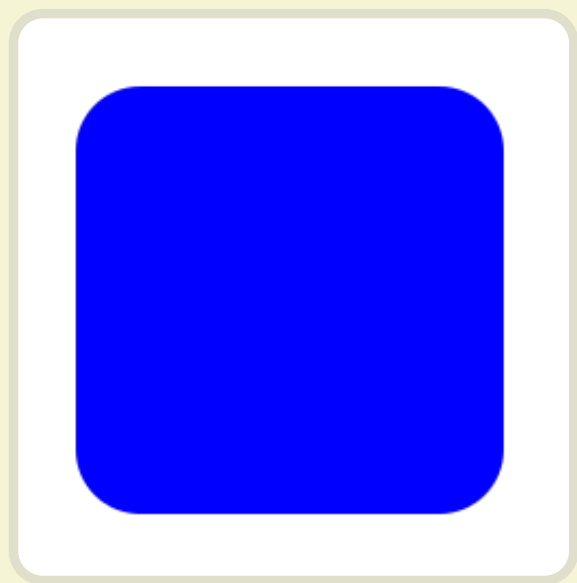
日期： 2010年3月15日

随着浏览器的升级，CSS3已经可以投入实际应用了。

但是，不同的浏览器有不同的CSS3实现，兼容性是一个大问题。上周的[YDN](#)介绍了[CSS3 Please](#)网站，该网站总结了一些常用功能的写法。

以下就是这些写法的详细介绍。所有代码都经过了Firefox 3.6和IE 8.o的验证，原文的错误之处也已得到改正。

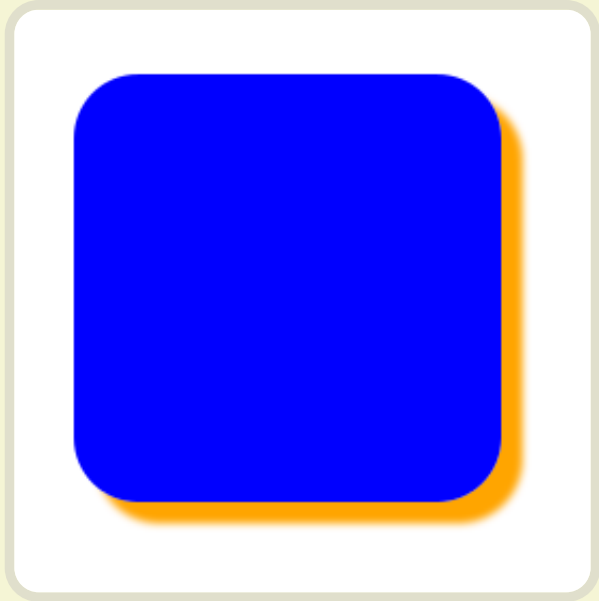
## 一、圆角（Rounded Corner）



```
.box_round {  
  
    -moz-border-radius: 30px; /* FF1+ */  
  
    -webkit-border-radius: 30px; /* Saf3+, Chrome */  
  
    border-radius: 30px; /* Opera 10.5, IE 9 */  
  
}
```

圆角的实现比较简单，只要设好一个半径值就可以了。遗憾的是，目前所有的IE都不支持CSS圆角，要等到IE 9才行。

## 二、盒状阴影（Box Shadow）



```
.box_shadow {  
  
    -moz-box-shadow: 3px 3px 4px #ffffff; /* FF3.5+ */  
  
    -webkit-box-shadow: 3px 3px 4px #ffffff; /* Saf3.0+,  
Chrome */  
  
    box-shadow: 3px 3px 4px #ffffff; /* Opera 10.5, IE  
9.0 */  
  
    filter:  
progid:DXImageTransform.Microsoft.dropshadow(OffX=3px,  
OffY=3px, Color='#ffffff'); /* IE6,IE7 */  
  
    -ms-filter:  
"progid:DXImageTransform.Microsoft.dropshadow(OffX=3px,  
OffY=3px, Color='#ffffff')"; /* IE8 */  
}
```

-moz-box-shadow、-webkit-box-shadow和box-shadow的设置是一样的，都有4个参数，含义分别为：x轴偏移值、y轴偏移值、阴影的模糊度、以及阴影颜色。

IE 6~8使用其独有的滤镜，需要设置三个参数：offX（X轴偏移值）、offY（Y轴偏移值）、Color（阴影颜色）。

### 三、线性渐变（Gradient）



```
.box_gradient {

    background-image: -moz-linear-gradient(top, #444444,
#999999); /* FF3.6 */

    background-image: -webkit-gradient(linear,left top,
left bottom, color-stop(0, #444444),color-stop(1,
#999999)); /* Saf4+, Chrome */

    filter:
progid:DXImageTransform.Microsoft.gradient(startColorstr='
endColorstr='#999999', GradientType='0'); /* IE6,IE7 */

    -ms-filter:
"progid:DXImageTransform.Microsoft.gradient(startColorstr=
endColorstr='#999999',GradientType='0')"; /* IE8 */

}
```

先看Firefox。

```
-moz-linear-gradient(top, #444444, #999999);
```

-moz-linear-gradient有三个参数。第一个参数表示线性渐变的方向，top是从上到下、left是从左到右，如果定义成left top，那就是从左上角到右下角。第二个和第三个参数分别是起点颜色和终点颜色。你还可以在它们之间插入更多的参数，表示多种颜色的渐变。

```
-webkit-gradient(linear,left top, left bottom, color-
stop(0, #444444),color-stop(1, #999999));
```

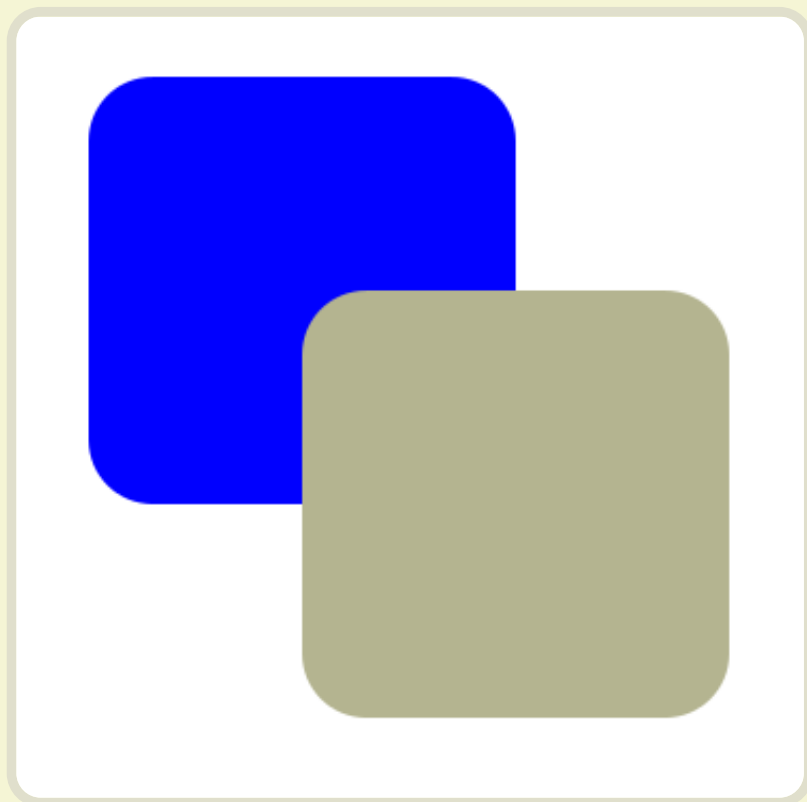
-webkit-gradient是webkit引擎对渐变的实现，一共有五个参数。第一个参数表示渐变类型（type），可以是linear（线性渐变）或者radial（辐射渐变）。第二个参数和第三个参数，都是一对值，分别表示渐变起点和终点。这对值可以用坐标形式表示，也可以用关键值表示，比如left top（左上角）和left bottom（左下角）。第四个和第五个参数，分别是两个color-stop函数。color-stop函数接受两个参数，第一个表示渐变的位置，0为起点，0.5为中点，1为结束点；第二个表示该点的颜色。

```
DXImageTransform.Microsoft.gradient(startColorstr='#444444
endColorstr='#999999', GradientType='0');
```

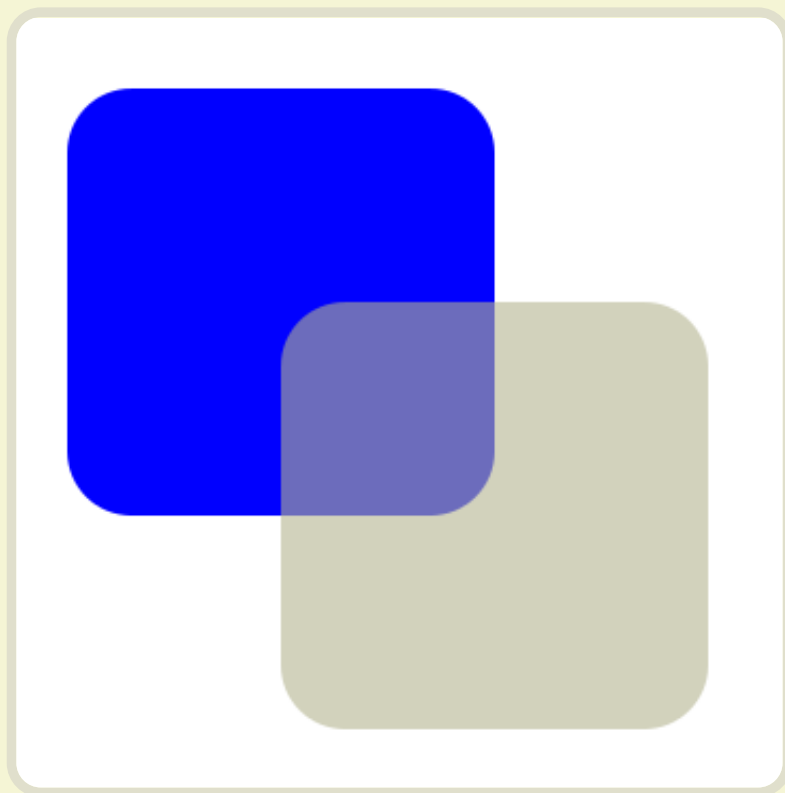
IE依靠滤镜实现渐变。startColorstr表示起点的颜色，endColorstr表示终点颜色。GradientType表示渐变类型，0为缺省值，表示垂直渐变，1表示水平渐变。

#### 四、透明（opacity）

正常情况下，上层的对象会覆盖下层的对象。



但是，如果将上层对象的颜色变为透明，就可以透过它看到下层对象。



```
.box_rgba {  
  
    background-color: #B4B490;  
  
    background:transparent;  
  
    filter:  
progid:DXImageTransform.Microsoft.gradient(startColorstr='  
/* IE6,IE7 */  
  
    -ms-filter:  
"progid:DXImageTransform.Microsoft.gradient(startColorstr=  
/* IE8 */  
  
    zoom: 1;  
  
    background-color: rgba(180, 180, 144, 0.6); /* FF3+,
```

```
Saf3+, Opera 10.10+, Chrome */  
  
}
```

先看第一行。

```
background-color: #B4B490;
```

这是设置对象的预备色，也就是不透明时的颜色。如果浏览器不支持透明，就将显示这个颜色。

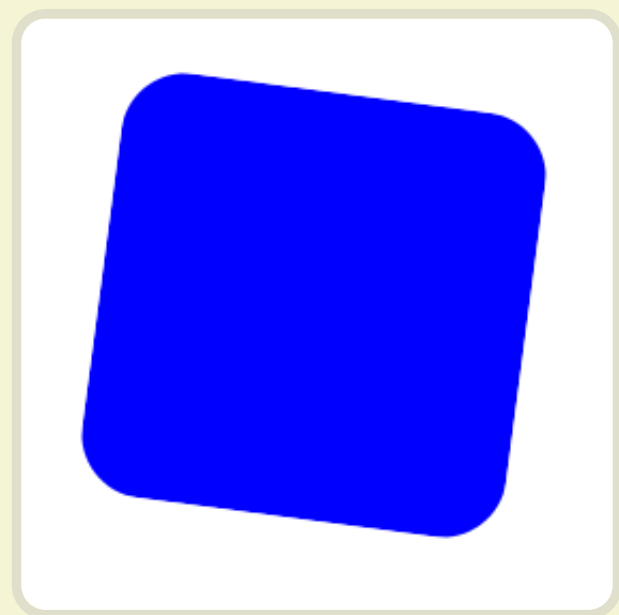
```
background:transparent;  
  
filter:  
progid:DXImageTransform.Microsoft.gradient(startColorstr='  
/* IE6,IE7 */  
  
-ms-filter:  
"progid:DXImageTransform.Microsoft.gradient(startColorstr=  
/* IE8 */  
  
zoom: 1;
```

这几行是专门为IE写的，其中主要用到 DXImageTransform.Microsoft.gradient滤镜。我们要为它设置起点色（startColorstr）和终点色（endColorstr）。在单色透明的情况下，这两个值是相同的。需要注意的是，它们的取值是一个八位的十六进制值，前两位表示alpha通道值，00表示完全透明，FF表示完全不透明；后六位则是这个颜色的RGB值。

```
background-color: rgba(180, 180, 144, 0.6);
```

除了IE，其他浏览器几乎都支持rgba函数。它有四个参数，前三个为一种颜色的RGB值，第四个为透明度，这里设为0.6。

## 五、旋转（rotation）





```

.box_rotate {

    -moz-transform: rotate(7.5deg); /* FF3.5+ */

    -o-transform: rotate(7.5deg); /* Opera 10.5 */

    -webkit-transform: rotate(7.5deg); /* Saf3.1+,
Chrome */

    filter:
progid:DXImageTransform.Microsoft.Matrix(M11=0.9914,M12=-0
expand');

    -ms-filter:
"progid:DXImageTransform.Microsoft.Matrix(M11=0.9914,M12=-
expand)"; /* IE8 */

}

```

除了IE以外，其他浏览器都是用rotate函数，实现某个对象的旋转。比如rotate(7.5deg)就表示顺时针旋转7.5度（degree）。

IE则需要用到一个复杂的滤镜DXImageTransform.Microsoft.Matrix。它一共接受五个参数，前四个参数需要自行计算三角函数，然后分别写成M11 = cos(rotation),M12 = -sin(rotation),M21 = sin(rotation),M22 = cos(rotation)，其中的rotation表示旋转角度，如果顺时针旋转7.5度，则rotation就为7.5；第五个参数SizingMethod表示重绘方式，'auto expand'代表自动扩展到新的边界。

除了这个滤镜，IE还有一个稍微简单一点的滤镜

DXImageTransform.Microsoft.BasicImage(rotation=x)。其中的x只能取值为1, 2, 3, 0，分别表示顺时针选择90度、180度、270度和360度。

## 六、服务器端字体（font-face）

设计网页的时候，可能会用到某种特殊的字体。如果用户的机器中没有安装，文字只能以普通字体显示。



这时可以让用户的浏览器自行下载服务器端字体，然后就能呈现出设计者想要的效果。



```
@font-face {  
  
    font-family: 'MyFont';  
  
    src: url('myfont.eot'); /* IE6+ */  
  
    src: local('myfont.ttf'),  
  
        url('myfont.woff') format('woff'), /* FF3.6 */  
  
        url('myfont.ttf') format('truetype'); /* FF3.5+,  
Saf3+,Chrome,Opera10+ */  
  
}
```

第一行代码：

```
font-family: 'MyFont';
```

表示为这种字体起一个名称，可以随意设置，我这里用的是MyFont。

```
src: url('myfont.eot');
```

这一行表示字体位置，由于ie只支持服务器端的[eot字体](#)，所以这一行是ie专用的。

```
src: local('myfont.ttf'),  
  
        url('myfont.woff') format('woff'),  
  
        url('myfont.ttf') format('truetype');
```

local()表示在本机（客户端）查找该字体，如果本机已经安装了，就不用下载了。url()表示字体在服务器上的位置，format()用来说明字体格式。Firefox 3.5支持TrueType和OpenType字体，Firefox 3.6又增加了WOFF字体。其他基于Webkit引擎的浏览器（safari, opera、chrome），目前好像只支持truetype。

然后，使用的时候这样写就可以了。

```
h2{ font-family: "MyFont"; }
```

需要注意的是，字体文件必须与网页文件来自同一个域名，符合浏览器的"同源政策"。另外，由于中文字体文件太大，服务器端字体显然只适用于英文字体。

## 七、其他

利用css3，还可以完成transform（变形），包括skew（扭曲）和scale（缩放），以及css transitions（动态变换）。这些内容待以后再补充。

（完）

### 文档信息

- 版权声明： 自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期： 2010年3月15日

## 相关文章

- **2020.12.13:** [《SSH 入门教程》发布了](#)

SSH 是登录 Linux 服务器的必备工具，只要你在做互联网开发，多多少少都会用到它。

- **2020.11.02:** [微信小程序入门教程之四：API 使用](#)

今天是这个系列教程的最后一篇。

- **2020.10.29:** [微信小程序入门教程之三：脚本编程](#)

这个系列教程的前两篇，介绍了小程序的项目结构和页面样式。

- **2020.10.27:** [微信小程序入门教程之二：页面样式](#)

这个系列的上一篇教程，教大家写了一个最简单的 Hello world 微信小程序。



[Weibo](#) | [Twitter](#) | [GitHub](#)

Email: [yifeng.ruan@gmail.com](mailto:yifeng.ruan@gmail.com)