专题三: 各种布局实战套路

table布局

float布局

inline-block布局

盒布局

flexbox布局

Grid布局

columns布局

Shapes布局

常见的CSS布局实战

- 水平居中布局
- 垂直居中布局
- 居中布局(水平+垂直)
- 两列布局
- 三列布局
- 圣杯布局
- 双飞翼布局
- 等分布局
- 等高分局
- 全屏布局
- CSS3多列布局

> 大纲目录
>
> - table布局
> - float布局
> - inline-block布局
> - 盒布局
> - flexbox布局
> - Grid布局
> - columns布局
> - Shapes布局

> 什么是布局?
> 简单来说就是HTML页面的整体结构或骨架，类似于传统的报纸或杂志中的排版；
> 布局不是某个技术内容，而是一种设计思想；
>
> - CSS知识体系的重中之重
> - 早期以table为主(简单)
> - 后来以技巧性布局为主(难)
> - 现在有flexbox/grid(偏简单)
> - 响应式布局是必备知识

table布局

> display:table

> display:table-cell，相当于td元素
> display:table-row，相当于tr元素
> table-layout:fixed | auto(默认)

float布局

> CSS清除浮动方法总结: https://juejin.im/post/582d98d5da2f600063e28f27
> BFC原理全面剖析: https://juejin.im/entry/59c3713a518825396f4f6969
> 特点:
> - 元素"浮动"
> - 脱离文档流
> - 但不脱离文本流
>
> 影响:
> - 对自身的影响: 形成"块"(BFC)、位置尽量靠上、位置尽量靠左(右)，无法满足会靠下
> - 对兄弟的影响: 上面贴非float元素、旁边贴float元素、不影响其它块级元素位置、影响其它块级元素内部文本
> - 对父级的影响: 从布局上"消失"、高度塌陷(overflow:hidden | clearfix)
>
> 实战: float + margin
> - 两列布局
> - 三列布局

inline-block布局

> 特点:
> - 像文本一样排block元素
> - 没有清除浮动等问题
> - 需要处理间隙(父元素:font-size:0)

盒布局

> https://www.html5rocks.com/en/tutorials/flexbox/quick/
> - box-orient：horizontal | vertical | inherit; 用来确定父容器里子容器的排列方式，是水平还是垂直
> - box-direction: normal | reverse　用来确定父容器里的子容器排列顺序
> - box-align：start | end | center | baseline | stretch; 表示父容器里面子容器的垂直对齐方式;
> - box-pack：start | end | center | justify; 表示父容器里面子容器的水平对齐方式
> - box-flex: 子元素之间比例

flexbox布局

> http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html
> https://www.zhangxinxu.com/wordpress/2018/10/display-flex-css3-css/
> https://css-tricks.com/snippets/css/a-guide-to-flexbox/

https://www.w3cplus.com/css3/understanding-flexbox-everything-you-need-to-know.html
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。
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，行内元素也可以使用Flex布局；webkit内核的浏览器，必须加上-webkit-前缀。

- 弹性盒子
- 盒子本来就是并列的
- 指定宽度即可

全兼容写法:

- display: -webkit-box; /* Chrome 4+, Safari 3.1, iOS Safari 3.2+ */
- display: -moz-box; /* Firefox 17- */
- display: -webkit-flex; /* Chrome 21+, Safari 6.1+, iOS Safari 7+, Opera 15/16 */
- display: -moz-flex; /* Firefox 18+ */
- display: -ms-flexbox; /* IE 10 */
- display: flex; /* Chrome 29+, Firefox 22+, IE 11+, Opera 12.1/17/18, Android 4.4+ */

Grid布局

http://www.ruanyifeng.com/blog/2019/03/grid-layout-tutorial.html
https://www.zhangxinxu.com/wordpress/2018/11/display-grid-css-css3/
https://zhuanlan.zhihu.com/p/33030746
网格布局（Grid）是最强大的 CSS 布局方案。
它将网页划分成一个个网格，可以任意组合不同的网格，做出各种各样的布局。以前，只能通过复杂的CSS框架达到的效果，现在浏览器内置了；
Grid 布局与 Flex 布局有一定的相似性，都可以指定容器内部多个项目的位置。但是，它们也存在重大区别。
Flex 布局是轴线布局，只能指定"项目"针对轴线的位置，可以看作是一维布局。Grid 布局则是将容器划分成"行"和"列"，产生单元格，然后指定"项目所在"的单元格，可以看作是二维布局。Grid 布局远比Flex布局强大。

columns布局

https://www.zhangxinxu.com/wordpress/2019/01/css-css3-columns-layout/
https://www.zhangxinxu.com/wordpress/2017/02/css3-multiple-column-layout-read-horizontal/
column-gap 列的间距

- 表示每一栏之间的那个空白间隙大小；
- normal默认值。在多栏布局中为1em，在其它类型的布局中为0。
- <length>具体的长度值。不支持负数。

- <percentage>百分比值。和column-width不同，column-gap支持百分比值。同样，不能是负数。

column-width 列的宽度

- 表示每一栏/列的最佳宽度。如果我们只设定column-width，浏览器会自动根据现有容器宽度划分栏目的个数。
- <length>表示设定的最佳列宽值。实际呈现的每一栏的宽度可能与指定值不同;
- auto默认值。表示每一栏的宽度由其它CSS属性决定，例如column-count。
- 一些细节：
- column-width有时候会无效。例如容器宽度400像素，设定的每一栏宽度是300像素，不足以分栏，此时内容填充填充表现为充分利用可用空间，最终呈现的列宽比设定的更宽。又例如容器宽度400像素，column-width设置为500像素，则最终分栏宽度不会超过容器宽度，比设定的500像素要小。
- column-width不支持负值，也不支持百分比值。

column-rule 列的边框

- column-rule是column-rule-width，column-rule-style和column-rule-color这3个CSS属性的缩写。正如border是border-style，border-width和border-color的缩写一样。
- column-rule-width: 表示每个栏目中间分隔线的宽度大小。支持的属性值和border-width是一模一样的,thin：薄薄的，等同于1px; medium（默认值）：薄厚均匀，等同于3px；thick：厚厚的，等同于5px；
- column-rule-style: 表示每个栏目中间分隔线的类型。支持的属性值和border-style是一模一样的;
- column-rule-color: 表示每个栏目中间分隔线的颜色;

column-span 横跨多列

- 有点类似于表格布局中的colspan这个HTML属性，表示某一个内容是否跨多栏显示。
- none表示不横跨多栏，默认值。
- all表示横跨所有垂直列。

column-fill 列的填充

- 用是当内容分栏的时候，如何平衡每一栏填充的内容。
- auto 按顺序填充每一列。内容只占用它需要的空间。
- balance 默认值。尽可能在列之间平衡内容。在分隔断开的上下文中，只有最后一个片段是平衡的。举例来说就是有多个<p>元素，正好最后一个<p>换行了，那这个<p>元素的内容前后等分，保持平衡。这就会造成最后一栏内容较少的情况。
- balance-all（可忽略）尽可能在列之间平衡内容。在分隔断开的上下文中，所有片段都是平衡的。

Shapes布局

https://www.zhangxinxu.com/wordpress/2019/02/css-css3-shapes/

> CSS Shapes布局可以实现不规则的文字环绕效果，需要和浮动配合使用。

常见的CSS布局实战

水平居中布局

> 含义: 指当前元素在父级元素容器中，水平方向是居中显示的；
>
> 方案一: inline-block + text-align 属性配合使用
>
> 珠峰培训
>
> 通过以下CSS样式代码实现水平方向居中布局效果：
> .parent{text-align:center;}
> .child{display:inline-block;}
> 优点: 浏览器兼容性比较好;
> 缺点：text-align属性具有继承性，导致子元素的文本也是居中显示的;
>
> 方案二: table + margin 属性配合使用
>
> 珠峰培训
>
> 通过以下CSS样式代码实现水平方向居中布局效果：
> .child{display:table;margin:0 auto;}
> 优点: 只需要对子级元素进行设置就可以实现水平方向居中布局效果;
> 缺点：如果子级元素脱离文档流(float:left或right/position:absolute或fixed)，导致margin属性的值无效;
>
> 方案三：absolute + margin 属性配合使用
>
> 珠峰培训
>
> 通过以下CSS样式代码实现水平方向居中布局效果：
> .parent{position:relative;}
> .child{position:absolute;left:50%;margin-left:-父元素宽度/2}
>
> 方案四：absolute + transform属性配合使用
>
> 珠峰培训
>

通过以下CSS样式代码实现水平方向居中布局效果：
.parent{position:relative;}
.child{position:absolute;left:50%;transform:translateX(-50%)}
优点: 父级元素是否脱离文档流，不影响子级元素水平居中效果;
缺点: transform属性是CSS3中新增属性，浏览器支持情况不好;

方案五：flex + justify-content 属性配合使用

    珠峰培训

通过以下CSS样式代码实现水平方向居中布局效果：
.parent{display:flex;justify-content: center;}

总结:

- 文本/行内元素/行内块级元素 .parent{text-align:center}
- 单个块级元素 .son{width:1000px(定宽)，margin:0 auto}
- 多个块级元素 .parent{text-align:center} .son{display:inline-block}
- 使用绝对定位: 子绝父相，top、right、bottom、left的值是相对于父元素尺寸的，然后margin或者transform是相对于自身尺寸的，组合使用达到水平居中的目的;
- 任意个元素(flex): #parent{display: flex; justify-content: center; }

垂直居中布局

含义: 指当前元素在父级元素容器中，垂直方向是居中显示的;

方案一: table-cell + vertical-align 属性配合使用

    珠峰培训

通过以下CSS样式代码实现垂直方向居中布局效果：
.parent{display:table-cell;vertical-align:middle;}
优点: 浏览器的兼容性比较好;
缺点：vertical-align属性具有继承性，导致父元素的文本也是居中显示的;

方案二：absolute + transform属性配合使用

    珠峰培训

通过以下CSS样式代码实现垂直方向居中布局效果：
.parent{position:relative;}
.child{position:absolute;top:50%;transform:translateY(-50%)}

优点: 父级元素是否脱离文档流，不影响子级元素垂直居中效果;
缺点：transform属性是CSS3中新增属性，浏览器支持情况不好;

方案三：flex +align-items 属性配合使用

```
<div class="parent">
    <div class="child">珠峰培训</div>
</div>
```

通过以下CSS样式代码实现垂直方向居中布局效果：

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

总结:

- 文本/行内元素/行内块级元素 .parent{height:150px;line-height:150px;} 高度等于行高的值；
- 多行文本/行内元素/行内块级元素 .parent{height:150px;line-height:30px;} 行高等于height/行数；
- 图片元素： .parent{height:150px;line-height:150px;font-size:0;} .son{vertical-align:middle}
- 单个块级元素:
  - 使用tabel-cell实现： .parent{display:table-cell;vertical-align:middle}
  - 使用position实现： 子绝父相，top、right、bottom、left的值是相对于父元素尺寸的，然后margin或者transform是相对于自身尺寸的，组合使用达到垂直居中的目的；
  - 利用flex实现 .parent{display:flex; align-items: center;}
- 任意个元素：.parent{display:flex; align-items: center;} 或 .parent{display:flex;} .son{align-self: center;}
  或者 .parent{display:flex;flex-direction: column;justify-content: center;}

居中布局(水平+垂直)

含义: 居中布局实际上就是既要水平方向居中，也要垂直方向居中;
16种方法实现水平居中垂直居中 https://juejin.im/post/6844903474879004680

方案一: table + margin实现水平方向居中，table-cell + vertical-align实现垂直方向居中

```
<div class="parent">
    <div class="child">珠峰培训</div>
</div>
```

通过以下CSS样式代码实现水平垂直方向居中布局效果：

```
.parent{display:table-cell;vertical-align:middle;}
.child{display:table;margin:0 auto;}
```

优点: 浏览器的兼容性比较好;
缺点：vertical-align属性具有继承性，导致父元素的文本也是居中显示的;

方案二：absolute + transform/margin 实现水平方向和垂直方向居中

```
<div class="parent">
    <div class="child">珠峰培训</div>
</div>
```

通过以下CSS样式代码实现水平垂直方向居中布局效果：

```
.parent{position:relative;}
.child{position:absolute;top:50%;left:50%;transform:translate(-50%,-50%)}
```

优点: 父级元素是否脱离文档流，不影响子级元素垂直居中效果;

缺点：transform属性是CSS3中新增属性，浏览器支持情况不好;

方案三：flex+ justify-content/justify-content 实现水平方向和垂直方向居中

```
<div class="parent">
    <div class="child">珠峰培训</div>
</div>
```

通过以下CSS样式代码实现水平垂直方向居中布局效果：

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

优点: 简单灵活；功能强大;

缺点：PC端兼容性不好，移动端（Android4.0+）；

总结:

- 行内/行内块级/图片: 原理：text-align: center; 控制行内内容相对于块父元素水平居中,然后就是line-height和vertical-align的基友关系使其垂直居中，font-size: 0; 是为了消除近似居中的bug；

  ```
  .parent{height:150px;line-height:150px;text-align:center;font-size:0;}
  .son{vertical-align:middle;}
  ```
- table-cell: CSS Table，使表格内容垂直对齐方式为middle,然后根据是行内内容还是块级内容采取不同的方式达到水平居中;
- 绝对定位: 三种写法-子绝父相，top、right、bottom、left的值是相对于父元素尺寸的，然后margin或者transform是相对于自身尺寸的，组合使用达到几何上的水平垂直居中；
- 利用flex居中: .parent{display:flex;justify-content:center;align-items:center;}

两列布局

两列布局一般情况下是指定宽与自适应布局，两列中左列是确定的宽度，右列是自动填满剩余所有空间的一种布局效果；

左列定宽，右列自适应

- float + margin属性实现；
  - 优点：实现方式简单

    - 缺点：自适应元素margin属性值需要与定宽元素的width属性值保持一致；
      定宽元素浮动与自适应元素不浮动导致浏览器兼容性不好；
      右列自适应元素中定义的了加clear:both的子级元素会出问题；

- float + margin(fix) 实现；
- float + overflow属性实现；
    - 优点: 简单易用 全兼容
    - 缺点: overflow属性不仅解决了两列布局问题，同时设置了内容溢出的情况；
- display属性的table相关值实现；
    - 优点: 浏览器兼容比较好
    - 缺点: 将所有元素的display属性设置为table相关值，受到相应制约；
- 使用绝对定位实现；
- 使用flex实现；
- 使用Grid实现；

左列自适应，右列定宽

- float + margin属性实现；
- float + overflow属性实现；
- display属性的table相关值实现；
- 使用绝对定位实现；
- 使用flex实现；
- 使用Grid实现；

一列不定宽，一列自适应

- float + margin属性实现；
- 使用flex实现；
- 使用Grid实现；

三列布局

三列布局一般情况下是指三列中左边两列是确定的宽度，右边一列是自动填满剩余所有空间的一种布局效果；

两列定宽,一列自适应(右边)

- float + margin属性实现；
- float + overflow属性实现；
- display属性的table相关值实现；
- 使用flex实现；
- 使用Grid实现；

两侧定宽,中间自适应

- 圣杯布局方法
- 双飞翼布局方法
- 使用Gird实现
- 使用table实现
- 使用flex实现
- 使用position实现

圣杯布局

圣杯布局是来源于该布局效果类似圣杯而得名。简单来说，就是指三行三列布局；
圣杯布局核心：主要是实现中间主体部分中的左右定宽+中间自适应的布局效果；

圣杯布局

```
// CSS
#parent {
    box-sizing: border-box;
    height: 500px;
    padding: 0 215px 0 115px;  /*为了使#center摆正,左右padding分别等于左右盒子的宽,可以结合左右盒子相对定位的left调整间距*/
}
#left {
    margin-left: -100%;  /*使#left上去一行*/
    position: relative;
    left: -115px;  /*相对定位调整#left的位置,正值大于或等于自身宽度*/
    float: left;
    width: 100px;
    height: 500px;
    background-color: #f00;
    opacity: 0.5;
}
#center {
    float: left;
    width: 100%;  /*由于#parent的padding,达到自适应的目的*/
    height: 500px;
```

```
    box-sizing: border-box;
    border: 1px solid #000;
    background-color: #eeff2b;
}
#right {
    position: relative;
    left: 215px; /*相对定位调整#right的位置,大于或等于自身宽度*/
    width: 200px;
    height: 500px;
    margin-left: -200px;  /*使#right上去一行*/
    float: left;
    background-color: #0f0;
    opacity: 0.5;
}

// HTML
<div id="parent">
    <!--#center需要放在前面-->
    <div id="center">中间自适应</div>
    <div id="left">左列定宽</div>
    <div id="right">右列定宽</div>
</div>
```

双飞翼布局

> 双飞翼布局最早是淘宝团队提出，是针对圣杯布局的优化解决方案。
> 主要是优化了圣杯布局中开启定位的问题。

双飞翼布局

```
// CSS
#left {
    float: left;
```

```
    width: 100px;
    height: 500px;
    margin-left: -100%; /*调整#left的位置,值等于自身宽度*/
    background-color: #f00;
    opacity: 0.5;
}
#center {
    height: 500px;
    float: left;
    width: 100%;
    background-color: #eeff2b;
}
#center_inbox{
    height: 480px;
    border: 1px solid #000;
    margin: 0 220px 0 120px;  /*关键!!!左右边界等于左右盒子的宽度,多出来的为盒子间隔*/
}
#right {
    float: left;
    width: 200px;
    height: 500px;
    margin-left: -200px;  /*使right到指定的位置,值等于自身宽度*/
    background-color: #0f0;
    opacity: 0.5;
}

<!--中间栏需要放在前面-->
<div id="parent">
    <div id="center">
        <div id="center_inbox">中间自适应</div>
        <hr>  <!--方便观察原理-->
    </div>
    <div id="left">左列定宽</div>
    <div id="right">右列定宽</div>
</div>
```

等分布局

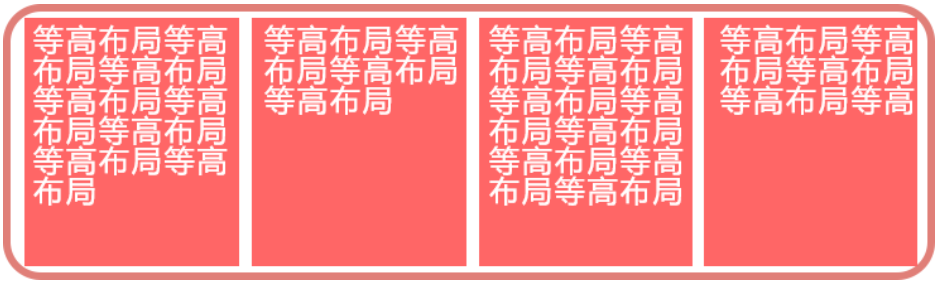
> 等分布局就是指一行被分为若干列，每一列的宽度是相同的值；

- float属性实现等分布局效果；
- display属性的值有关table实现等分布局效果；
- flex属性实现等分布局效果；
- Grid属性实现等分布局效果；

等高分局

> https://www.cnblogs.com/xiaohuochai/p/5457127.html

> 等高布局就是一行被划分成若干列，每一列的高度是相同的值；

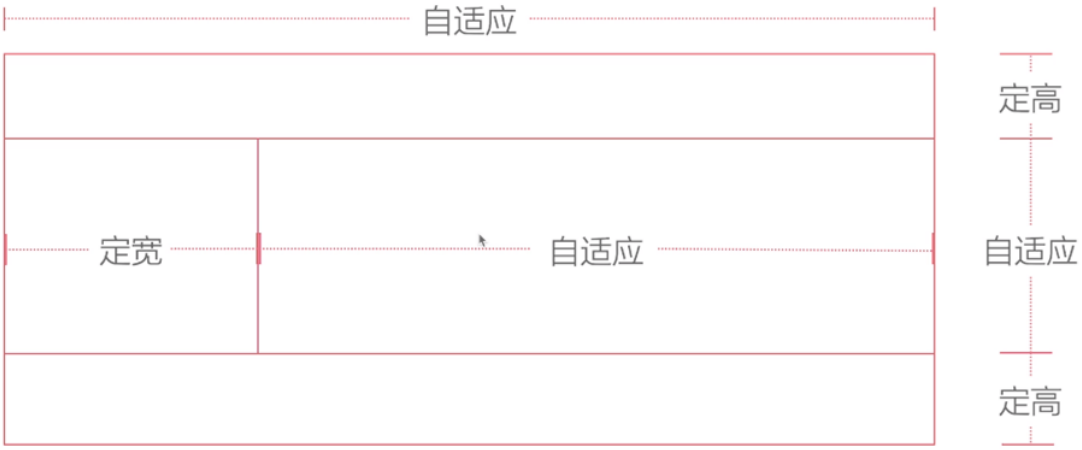


- display属性的值有关table实现；
- padding + margin属性实现等高布局效果；

全屏布局

> https://www.cnblogs.com/xiaohuochai/p/5458068.html

> 全屏布局就是指HTML页面铺满整个浏览器窗口，并且没有滚动条。而且还可以跟着浏览器的大小变化而变化；



- 利用绝对定位实现；
- 利用flex实现；

CSS3多列布局

- columns: 他是column-width和column-count属性的缩写；
- column-width表示每一栏/列的最佳宽度,如果我们只设定column-width，浏览器会自动根据现有容器宽度划分栏目的个数；
- column-count表示理想的分栏数目；
- column-rule-color表示每个栏目中间分隔线的颜色；
- column-rule-style表示每个栏目中间分隔线的类型。支持的属性值和border-style是一模一样的；
- column-rule-width表示每个栏目中间分隔线的宽度大小。支持的属性值和border-width是一模一样的；
- column-rule:column-rule是column-rule-width，column-rule-style和column-rule-color这3个CSS属性的缩写。正如border是border-style，border-width和border-color的缩写一样；
- column-span有点类似于表格布局中的colspan这个HTML属性，表示某一个内容是否跨多栏显示。
- column-fill作用是当内容分栏的时候，如何平衡每一栏填充的内容；
- column-gap表示每一栏之间的那个空白间隙大小；