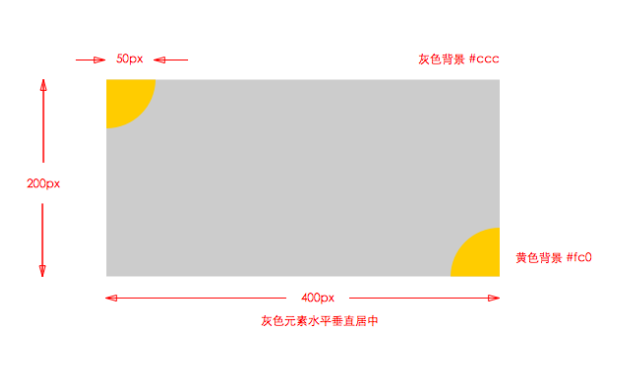
## 任务四：定位和居中问题

### 任务目标

* 实践HTML/CSS布局方式
* 深入了解position等CSS属性

### 任务描述

* 实现如 [示例图](http://7xrp04.com1.z0.glb.clouddn.com/task_1_4_1.png) 的效果
* 
* 灰色元素水平垂直居中，有两个四分之一圆位于其左上角和右下角。

### 任务注意事项

* 思考不同情况下（如灰色高度是根据内容动态变化的）水平垂直居中的解决方案。
* 动手试一试各种情况的组合，父元素和子元素分别取不同的 position 值。思考 position 属性各种取值的真正含义，尤其是 absolute 究竟是相对谁而言的。
* 注意测试不同情况，尤其是极端情况下的效果。
* 调节浏览器宽度，灰色元素始终水平居中。
* 调节浏览器高度，灰色元素始终垂直居中。
* 调节浏览器高度和宽度，黄色扇形的定位始终准确。
* 其他效果图中给出的标识均被正确地实现，错一项扣一分。

**我的笔记：**

CSS定位

浮动问题：虽然浮动相当的给力，但他们自己还是存在一定的问题。最典型的问题就是一个父元素包含了多个浮动的子元素。页面的内容设置了一个宽度，子元素的浮动确定了他们的位置，但浮动元素不会影响父元素的宽度。这样做会让父元素塌陷，从而使父元素的高度为“0”，以及忽略其他的属性。很多时候，这种现像都被忽略，特别是在父元素没有任何样式，以及其子元素看起来都正确的对齐。

所以我们有种解决方案：overflow，clearfix

1. Overflow

一种清除浮动的技巧是使用“overflow”属性。在具有浮动元素的父容器中设置“overflow”的属性值为“auto”，这样父容器就会有一个高度存在，在我们例子中的灰色背景也就能看得到了。



使用“overflow”技巧清除浮动，确实存在一些缺点。例如：当你添加样式，或者将嵌套在里面的“span”元素移动到父容器的外面，或者你想给元素添加一个盒子阴影和制作一个下拉菜单。在下面的演示例子中，你可以看到元素的盒子阴影被切断在父元素之内。

1. Clearfix

根据上下文，清除浮动更好的方法是clearfix技巧。“clearfix”清除浮动的技术是有点复杂，但有有比使用“overflow”技巧清除浮动更好的方法？

“clearfix”技巧是基于在父元素上使用“:before”和“:after”两个伪类。使用这些伪类，我们可以在浮动元素的父容器前面和后面创建隐藏元素。“:before”伪类是用来防止子元素顶部的外边距塌陷，使用“display: table”创建一个匿名的“table-cell”元素。这也确保在IE6和IE7下具有一致性。“:after”伪类是用来防止子元素的底部的外边距塌陷，以及用来清除元素的浮动。

在IE6和7的浏览器中，加上“\*zoom”属性来触发父元素的hasLayout的机制。决定了元素怎样渲染内容，以及元素与元素之间的相互影响。

例子：

.box-set:before,

.box-set:after {

content: "";

display: table;

}

.box-set:after {

clear: both;

}

.box-set {

\*zoom: 1;

}

相比起在box-set中添加样式，另外一个常见的方法是新增一个类，再把这个类添加在需要清除浮动的容器中。

如：

.group:before,

.group:after {

content: "";

display: table;

}

.group:after {

clear: both;

}

.group {

\*zoom: 1;

}

**Position:static，relative，absolute，fixed**

**z-index**

1. Static

元素都有position属性，其默认值是“static”，这也意味着，他们没有也不接受位置属性设置（top、right、bottom、left属性值设置）。另外元素设置了position属性，将会覆盖元素的默认值“static”。

1. Relative

relative”是“position”的另一个属性值，他和“static”属性值非常的相似。主要的区别是“relative”可以给元素设置位移（offset）“top、right、bottom和left”属性。通过这些位移属性设置可以给元素进行精确的定位。对于相对定位元素，这些属性的设置让元素从默认位置移动。例如，top设置一个值“20px”在一个相对定位的元素上，这个元素会在原来位置向下移动“20px”。反之，“top”设置一个“-20px”，这个元素会在原来的位置向上移动“20px”。

（相对定位元素的默认位置还是被元素自身占用，别的元素是无法占用的。也就是说相对定位元素的位移是相对于元素自身的边缘进行位移）

优先级：

top>bottom

Left>right(英文页面)

Right>left(阿拉伯页面)

1. Absolute

绝对定位元素也具有盒子位移属性，然而，绝对定位元素会脱离文档流。绝对定位元素直接从文档流中移出，绝对定位元素的位置直接和父容器是否设置了相对定位（绝对定位）有直接关系。绝对定位元素需要至少一个祖先元素设置了相对定位（绝对定位），不然元素定位会相对于页面的主体进行定位。

使用绝对定位的元素可以指定垂直和水平的位移属性，使绝对定位元素相对于设置了相对定们的祖先元素边缘进行移位。例如，一个绝对定位的元素设置了“top”值为“50px”和一个“right”值为“100px”,绝对定位元素会相对于其设置了相对定位的父元素的顶边向下移动50px;向左移动100px。

然而，使用了绝对定位的元素并没有进行任何盒子位移属性设置，那么绝对定位元素的顶部和左部会和设置了相对定位的父元素的顶边和左边重合。如果设置了一个盒子位移属性，比如说“top”，那么绝对定位元素垂直方向会进行移动，而水平位置默认还是左边对齐。

同时设置top，bottom，left和right可以是元素居中，因为这样使整个元素的高度和宽度都跨越整个容器。

1. Fixed

固定定位和绝对定位很类似，但是他定位是相对于浏览器窗口，并且不会随滚动条进行滚动。

“position”属性值中，仅有“fixed”属性值不能在IE6浏览器下运行，如果你想在IE6正常使用固定定位，那么你就需要为他写一些Hacks。

固定定位元素的盒子位移属性的使用和绝对定位的一样。

保持前面盒子移位，可以看到盒子固定定位是相对于浏览器窗口而不是设置了相对定位的父元素。

固定页头和页脚

固定定位最常见的一种用途就是在页面中创建一个固定头部、或者脚部、或者固定页面的一个侧面。就算是用户移动浏览器的滚动条，还是固定在页现与用户交流。

下面的示例代码能实现。注意如何设置“left”和“right”两个盒子位移，这使得“页脚”跨越了页面的整个宽度，而不需使用margin、border和padding来破坏盒模形就做了收缩自如。

footer {

bottom: 0;

left: 0;

position: fixed;

right: 0;

}

Z-index:

给元素设置“z-index”属性，首先要在这个元素上设置了“position”属性值为“relatvie”、“absolute”或者“fixed”之一。同样的，你要使用盒子位移属性，你也要先确认元素设置了“positions”属性值为“relative”、“absolute”或者“fixed”之一。

文章来源：

http://www.w3cplus.com/css/advanced-html-css-lesson2-detailed-css-positioning.html

注意：实现四分之一圆可以用：

border-radius:50px 0 0 0 ;