CSS Positioning

Absolute and relative positioning are not difficult to work with, once you have had the chance to experiment with them. In this first exercise you will create and position several coloured blocks or boxes, just to get a feel for working with positioning.

Exercise 1

1. Create an HTML document and save it as absolute_and_relative.htm.

2. Create five class selectors and name them: redbox, greenbox, bluebox, yellowbox, and cyanbox. Remember that a class selector must have the crosshatch character, (the full stop sign), preceding the name, as in: .redbox.

3. Assign each of the class selectors a height and width of 100 pixels.

4. Set the background colour for each of the boxes to match its name. For example, the background colour for .redbox should be red, for .greenbox green, and so on.

5. Set the text colour for each box to a contrasting colour, so the text will stand out. It doesn't matter which colour you use, as long as it is visible against the coloured background.

6. In the <body> portion of your HTML document, add five sets of elements, and assign each a class corresponding to one of the ones created in Step 2. For example, one should be .

7. In between each set of tags, add some content. In Figure 1-1, the contents simply reflect the box's colour. You can add whatever content you would like.

8. Save the page and display it in your browser. It should resemble Figure 1-1. This is the "static" positioning of these elements.

Figure 1-1 Five different-coloured boxes in a "static" arrangement

9. Modify the style rules for each of the boxes by adding the position property, with a value set to "relative."

10. Position the boxes (see figure 1-2) by using the top, right, bottom, and left offset properties with different combinations of values. Try using lengths, percentages, and combinations of both, along with different property combinations. Be sure to note how the various boxes reposition when you set the coordinates with these properties.

11. Save the page and view it in your browser.

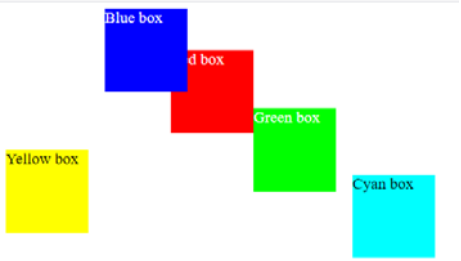


Figure 1-2 Relatively positioned boxes

12. Experiment with the boxes by changing the value of the position property from "relative" to "absolute." The results of that change from the original settings are reflected in Figure 1-3. Two boxes have moved completely off the screen, one is halfway off, and the remaining two have moved closer to the top of the screen. The same values have radically different effects. Alter the values to bring the boxes all back onto the page.

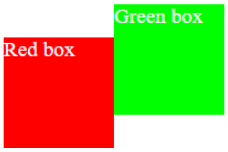


Figure 1-3 Absolutely positioned boxes

Exercise 2:

1. Try experimenting further with the absolute and relative positioning properties by trying to position a box in each corner of the browser window.
2. Create a layout in which you stack or overlap the boxes.
3. Try positioning the boxes in a stair-step arrangement.
4. Write a style rule for another element that will set its dimensions at 250 pixels by 250 pixels, give it a background colour of navy and text colour of yellow, and position it with absolute positioning so that the box's upper-left corner is in the centre of the page.
5. Try and set the above box so that it is horizontally and vertically centred in the browser window.

Exercise 3

- Create a folder called **layering_and_sound** on your disk space.
- Copy the contents of **resource** from the teachers drive into your new folder
- Create an html document and include a style for the body tag with:
  font-family: verdana, arial, Helvetica, sans-serif;
  color: #ffffff;
  font-size: 35px;
  background-color: #ff9900;
- note: there are no heading tags (eg. h1, h2) in this exercise.
- develop the page content to display the text content shown in the diagram to the right – the large text size of 35px is automatically applied from the body style
- add the code for the sunset.jpg image
- remember to include height, width and alt attribute for the image
- save the document as layereing1.html in the new folder
- view the document in different browsers – the text and the image are displayed in the top left of the browser window.
- in the next stage, you will alter the position of the image by applying a style.

Absolute positioning of elements

- add a custom class style to the document **.one** {position:absolute; top:80px; left:200px;}
- place the cursor immediately before the image coding for sunset.jpg and insert a new code
- insert the closing tag immediately after the image code
- save the file and view the updated document in the browser window
- the image has moved and is now placed 80 pixels from the top of the browser window and 200 pixels from the left
- alter the custom class .one so that the position of the sunset.jpg image is 30px from the top of the browser window.
- view the document in the browser – the image now overlaps the text
- the absolute positioning has placed this image at a precise position in the browser window, regardless of the placement of other elements in this document.

- reset the custom class .one so that the image is 80 pixels from the top and 200 from the left.
- place the cursor immediately before the closing </body> tag and insert the code for another image called ozsunset.gif
- view the document and note the default position of this second image in the browser window.
- create another custom class called .oz with an absolute position of 360px from the top and 190px from the left.

- insert a tag with the new custom class included before the ozsunset.gif image code
- view the document – now both images overlap

Layering elements with z-index

The z-index places elements in a 3-D stacking order on the z axis

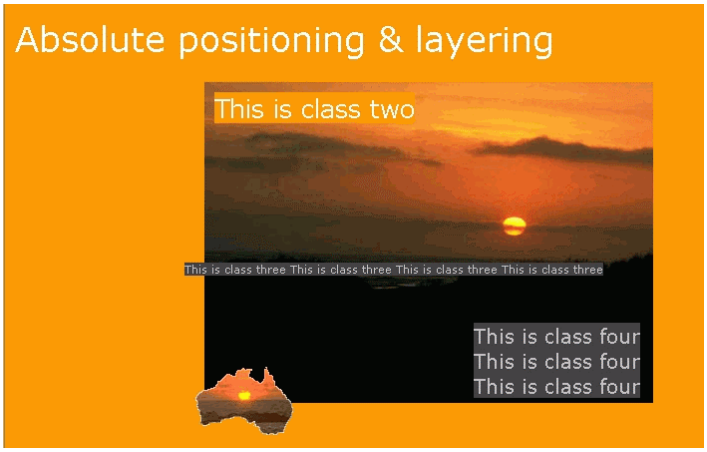
- alter the custom class **.one** to {position:absolute; top:80px; left:200px; z-index:2;}
- the z-index can also have a negative value eg. z-index:-1;
- view the document – the ozsunset.gif is now placed behind the other image

Develop the page to include the items listed below. Use the following styles as a guide – again, there are no heading tags in this exercise.

This is class twofont – verdana, arial, Helvetica, sans-serif
text color - #ffffff
text size – 25px
background colour - #ff9900
plus absolute positions and z-index

This is class three font – verdana, arial, Helvetica, sans-serif
text color - #c0c0c0
text size – 11px
background colour - #404040
plus absolute positions and z-index

This is class four font – verdana, arial, Helvetica, sans-serif
text color - #c0c0c0
text size – 20px
background colour - #404040
plus absolute positions and z-index



Exercise 4

Download and make the following changes to the file posit.css. You only need to edit the CSS - don't make any changes to the html.

- Change the size of the browser window and try scrolling down the page. The sidebar (blue) div has *fixed positioning* on it. Move it from where it is to the bottom left corner of the browser window.
- Keeping it attached to the bottom of the browser window, extend the width of the sidebar (blue) div to the full width of the browser. The bar must stay full-width no matter how wide the window is made.
- Without using margins, target the h2 inside the sidebar (i.e. the word *Sidebar* and pull it 100 pixels *up*, out of the blue box. Confirm by scrolling and resizing the browser window that is still attached to the blue sidebar div.

Exercise 5

Using some of the techniques above, you will create a header that sticks to the top of the viewport, but so that it doesn't interfere with the content, we're going to minimize it when the user scrolls down the page. Download the sticky_header.html and link up the jquery file in this folder. Add the following custom javascript to the bottom of the page and then insert some css and test.

CSS transitions are the best way of handling the animation portion of our sticky header. All we're using jQuery for is detecting the scroll position of the window.

When the scroll position of the window is greater than 1—meaning that the user has scrolled downwards—then we want to add the class 'sticky' to the *header*; otherwise we want to remove it. This means we'll be able to style the header based on whether the 'sticky' class is applied.

```
$(window).scroll(function() {
if ($(this).scrollTop() > 1){
   $('header').addClass("sticky");
   $('article').addClass("sticky");
  }
  else{
   $('header').removeClass("sticky");
   $('article').removeClass("sticky");
  }
});
```

The important thing to note is that using jQuery in this way degrades gracefully; if JavaScript is disabled, the navigation will still work, the header will simply be styled in the non-sticky default state.

Our CSS is used to style the two different states, the default state, and the 'sticky' state; and to transition between the two states.

To start with, let's add some simple styles that improve the look of the header and content:

```
* {
 margin: 0;
padding: 0;
border: 0;
font-size: 100%;
font-family: calibri, sans-serif;
vertical-align: baseline;
}

header{
  text-align: center;
  font-size: 72px;
  line-height: 108px;
  height: 108px;
  background: #335C7D;
  color: #fff;
  transition: all 0.4s ease;
}
p{
            margin-top:20px;
}
```

When the user scrolls down, the 'sticky' class will be applied, and we can now style the header differently to reflect that new priority on the page. We also set the position to fixed, so that we're not changing positioning mid-scroll.

There are several things we want to do: first, we want to change the size so that it uses up less screen space; we also want to change the colour and align to the left so that visually it doesn't interfere too much:

```
header.sticky {
  position: fixed;
  font-size: 24px;
  line-height: 48px;
  height: 48px;
  width: 100%;
  background: #efc47D;
  text-align: left;
  padding-left: 20px;
}

article.sticky {
  display: inline-block;
  margin-top: 110px;
}
```

Naturally, what you do here will depend on the design you're trying to achieve. You can do just about anything you like.

If you test this now, you'll see that the header changes as soon as we scroll down.