CSS Positioning

Standards-Based Layout For The Web

In this module you will learn about the CSS positioning standard, which specifies the mechanism for laying out documents on the web.

Basic Positioning: Centering Text

Recall from your earlier practice with CSS that style sheets can align text elements on a page; for example:

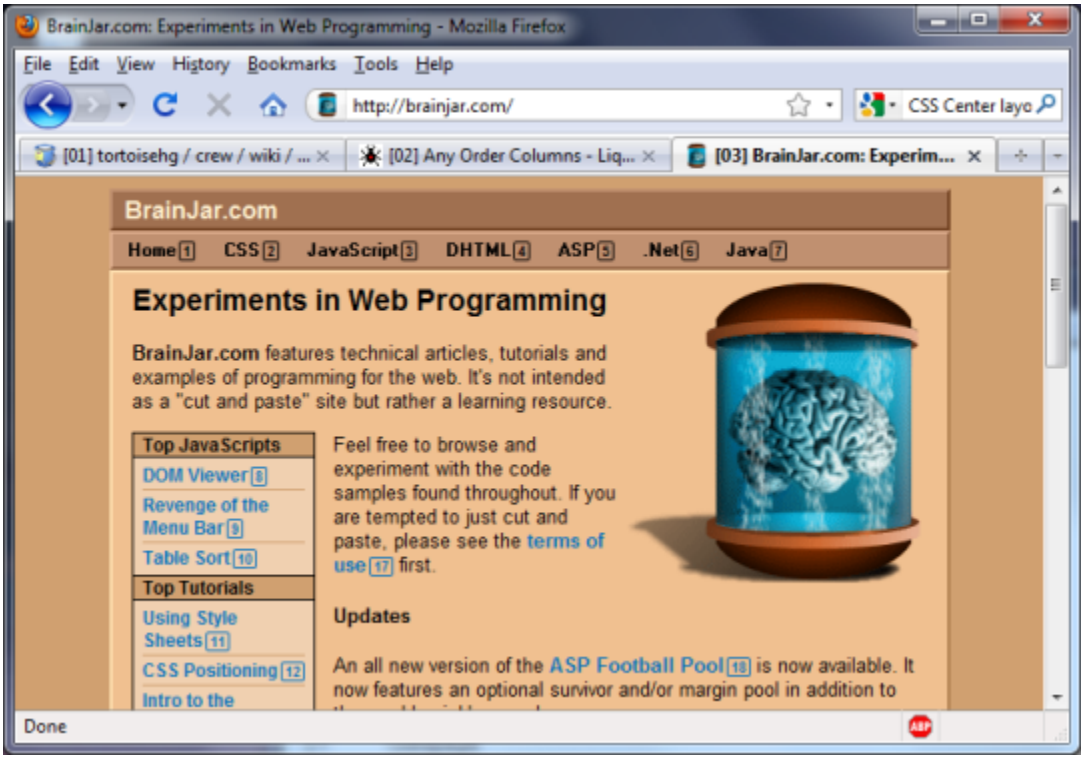
```
h1 {
    text-align: center;
}

p {
    text-align: justify;
}
```

This CSS would cause level 1 headings to be centered on the page and paragraphs to be justified—that is, stretched to fit the entire column, like a newspaper.

Centered Layout

What if you want to center the entire contents of your site on the page? You probably recognize the centered layout from the many sites on the Internet that use it:



Creating a centered layout gives your site a polished look and is fairly easy to do. However, before we look at the technique, we'll need to examine a couple new XHTML and CSS ideas.

The Tag and Element IDs

So far we've worked with XHTML elements as individual pieces. However, it is often helpful to think of a web page in terms of groups or blocks of elements. When designers speak about a web design, they will use terms like "header," "footer," and "navigation bar" to refer to the logical parts of the page. During design, it is helpful to think in terms of higher-level semantic blocks like these, rather than trying to define the site in terms of cumbersome syntax elements of XHTML.

To support this way of thinking, XHTML provides a mechanism for grouping other elements known as the `div` tag:

```
<div id="content">...
```

A div tag creates a block at of the elements inside it: whatever headings, paragraphs, text, lists, or other elements occur inside the div can be referenced and treated as a group.

To reference a div, you need to give it an id[*]: a unique identifier that can be used to unambiguously reference a single element. In the example div above, the ID for the div is `content`; you could, for example, cause all the text in the container to use justified alignment with this CSS:

```
#content {
    text-align: justify;
}
```

The CSS selector `#content` tells the browser to apply the specified style to the element with the ID `content`.

[*] Any XHTML element can be given an id, including paragraphs, images, hyperlinks, etc.

Keeping `id` and `class` straight

At this point, you may be itching to ask the question

What's the difference between id and class?

id: unique identifier for an element; used to apply style to exactly one element (a div container is one element that contains other elements), e.g.

XHTML:

```
<div id="footer">...
```

CSS:

```
#footer {
    ...
```

class: selector for a set of elements that are not necessarily related by tag name or structure.

The `highlight-yellow` example from your earlier CSS exercises provides a simple example of applying style information to an arbitrary set of elements. As a second, real-life example, I once developed a web site for a client who wanted to display hyperlinks to external sites differently from those internal to the site (Wikipedia uses this technique for links in their articles). To do this, internal links were created normally:

```
<a href="contact_us.html">Contact us</a>
```

But external links were specially identified as such using a class:

```
<a href="http://google.com" class="external-link">Google</a>
```

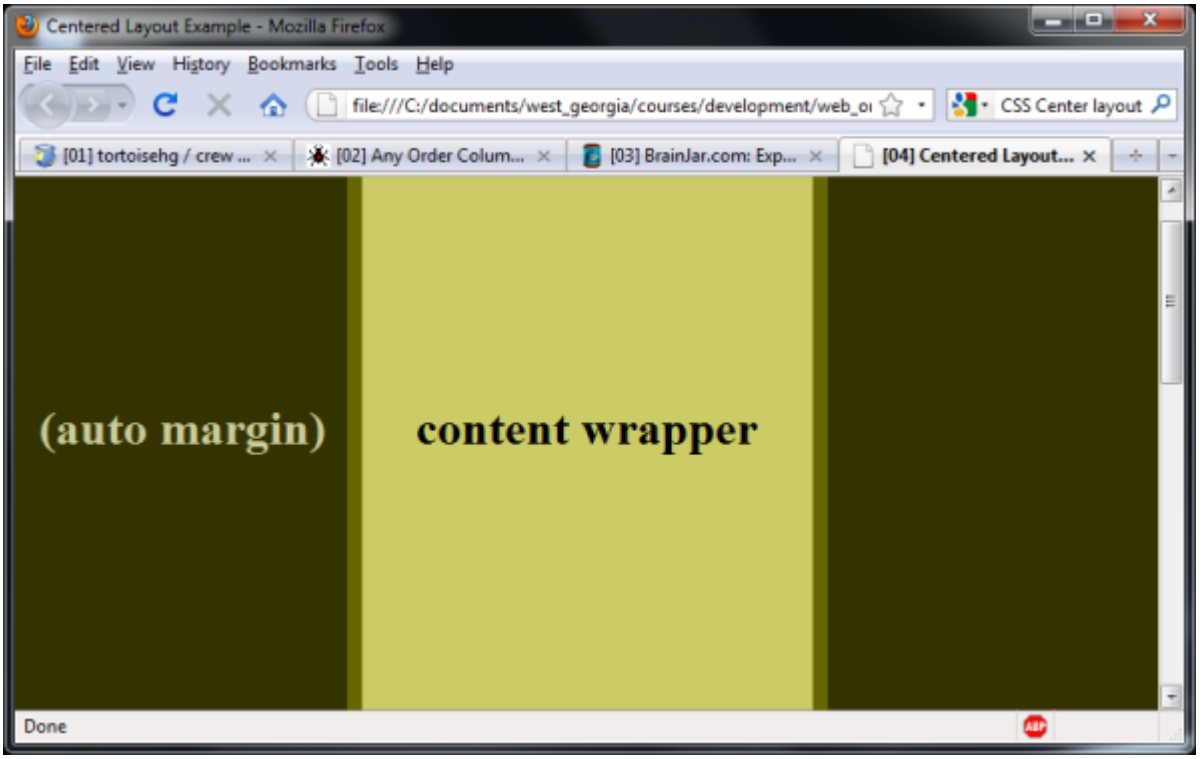
This provides a simple way of differentiating regular links from external ones using CSS styles.

Exercise 1: Centering Content

Now that you understand the necessary CSS and XHTML constructs, let's see how to center of the content area of a page. Here is the basic idea:

- create a container (`div`) to hold the content area of the page
- set the width of the content container to be less than the width of the screen/browser window
- set the side margins of the container to automatically fill the remaining space

Here is an example of a centered layout. The light center portion is the content of the page and has a defined width. The dark sidebars are automatically computed to center the content.



1. To begin the CSS positioning exercise, download the start package, open the XHTML file, and add a `div` called `wrapper` around the contents of the body section:

```
...
<body>
<div id="wrapper">
    ...
</div>
</body>
</html>
```

   Be very careful and make sure that you wrap the entire contents of the page, as it is quite easy to overlook some elements or insert the closing tag in the wrong place.

2. Now, edit the CSS file. Find the selector for the `wrapper` id and set its width:

```
#wrapper {
    width: 700px;
}
```

3. Finally, set the side margins to automatically fill up the remaining space:

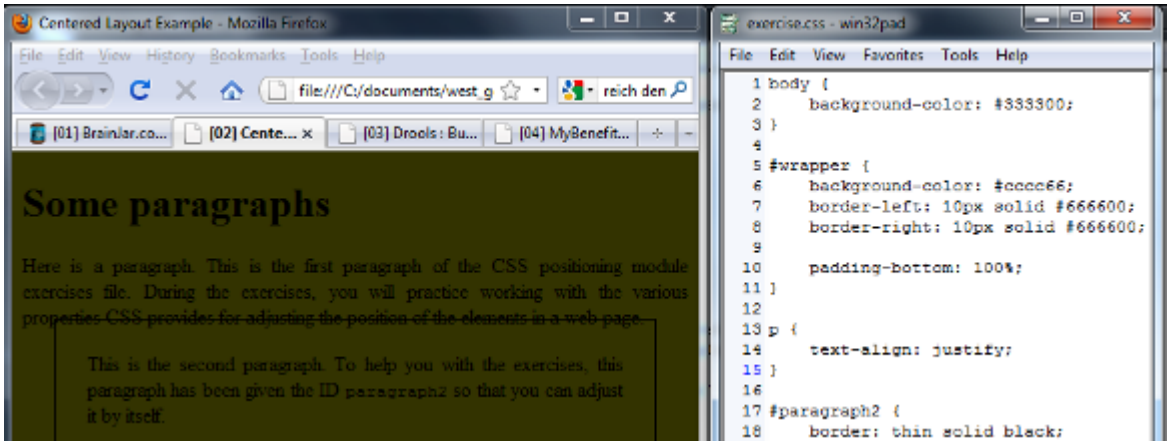
```
#wrapper {
    width: 700px;
    margin-left: auto;
    margin-right: auto;
}
```

CSS Positioning Fundamentals

CSS provides powerful tools for controlling the position of the elements that make up a web page. Let's review some of the basic concepts required and explore the details of CSS positioning.

Exercise 2: Box Model

1. Review the CSS box model: http://www.brainjar.com/css/positioning/default.asp
2. Review the `margin` and `padding` CSS properties:
   - http://www.w3schools.com/Css/css_margin.asp
   - http://www.w3schools.com/Css/css_padding.asp
3. Load the exercise webpage in your browser and open the CSS file in your text editor. It is helpful to position the windows next to each other on your screen so you can quickly switch between making changes and seeing the results of your work:



4. Adjust the margins and padding of the paragraph with ID `paragraph2`, which is surrounded by a black border:
   a. Change the padding on all four sides to 25 pixels
   b. Change the margin on all four sides to 25 pixels
   c. Experiment with other values for the margin and padding
   d. Challenge: what happens if you specify a negative margin?
5. On your own: modify the CSS so that there is some space between the content text and the borders on the sides of the content.

Relative Positioning

The CSS standard specifies a normal set of layout rules that are used whenever no position information is given. In the normal scheme, block elements like paragraphs are stacked on top of each other vertically while inline elements like text flow horizontally.

CSS allows web pages to override the normal position rules with one of three alternate positioning schemes. The first and easiest to understand is *relative positioning*. With relative positioning, an element is placed in the document normally and occupies the same amount of space as it normally would. However, after the element has been placed, it is shifted based on the properties specified in the CSS. Relative positioning can be used to nudge an element without dramatically affecting the layout of the rest of the document.

Exercise 3: Relative Positioning

1. Review the explanation of relative positioning: http://www.brainjar.com/css/positioning/default2.asp
2. Edit the CSS and add a selector for the paragraph with ID `relative_paragraph`.
3. Set the element to be relatively positioned:

   ```
   position: relative;
   ```

   Note that this does not change the element's position, it merely indicates that we wish to activate relative positioning for this element.
4. Push the paragraph down 15 pixels using the `top` property:

   ```
   top: 15px;
   ```
5. Now, nudge the paragraph 35 pixels to the right using the `left` property. Save your work and check your progress.

Floats

The second of the CSS positioning schemes is floating elements, or floats. Floating elements are pushed to one side of their container, and any text or other inline content flows around the floated element. This technique is often used with images, which can be pushed to one side of the text, minimizing the amount of empty vertical space next to them. Floats are also commonly used to create column layouts or for navigation menus.

Exercise 4: Floats

1. Review the explanation of float positioning: http://www.brainjar.com/css/positioning/default2.asp
2. Edit the CSS and add a selector for the class `float-right`.
3. Set the class to be floated to the right. Save your work, then reload the page to see the effect.
4. On your own: modify the CSS so that the paragraphs in the section **Columns** are laid out in two columns side-by-side.

Absolute Positioning

The final CSS positioning scheme is absolute positioning, which allows you to specify precisely where you want an element to appear. Absolutely positioned elements are completely removed from the normal flow of the document. Unlike relatively positioned elements, no telltale opening remains in the original layout to show where the absolutely positioned element came from.

While absolute positioning gives you a high degree of control over the positioning of the elements on your page, be warned! Your meticulously constructed, absolutely positioned work of haute art will probably not adjust well for folks using different browsers, screen resolutions, or media (such as print). Use absolute positioning sparingly and with caution; if you do use it, be sure to test your page on as many platform combinations as you can manage.

Exercise 5: Absolute Positioning

1. Review the description of absolute positioning: http://www.brainjar.com/css/positioning/default4.asp
2. Edit the CSS and add a selector for the element with the ID `absolute`.
3. Add CSS properties to:
   - set the color of the text to white
   - make the elements 75 pixels wide
   - position the element on the right edge of the window, 250 pixels down from the top
4. Challenge: the last section of the page includes a copy of the columns paragraphs that have been narrowed to fit side-by-side. This time, use absolute positioning (instead of floats) to lay out the two paragraphs in column format.

Summary

In this module you have learned about CSS positioning, the web standard that specifies how the content elements of a page can be rearranged and positioned. Combined with CSS style adjustments, CSS positioning gives you complete control over the look and feel of your web pages while supporting the clean separation between the content and style of the page.

Key Points

- “div“s provide convenient containers for structural elements
  Inline literal start-string without end-string.
- element IDs allow adjustments to individual elements
- relative positioning can be used to nudge an element's position without affecting the document flow
- elements can be floated to one side, allowing text and other inline content to flow around them
- absolute positioning provides precise control over the position of an element

Additional Resources

- W3Schools CSS positioning reference: http://www.w3schools.com/Css/css_positioning.asp