

Unit H: Implementing Page Layout with HTML and CSS

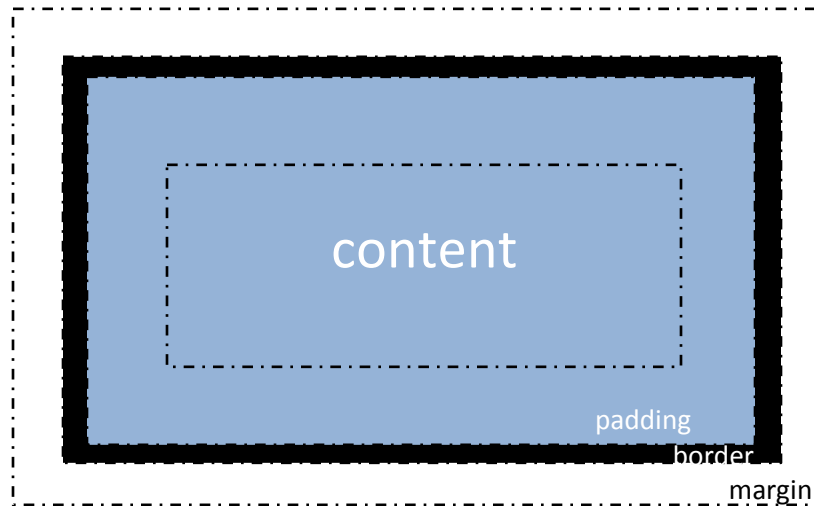
In Unit H, you will learn how to create professional Web page designs using multicolumn layouts. Before creating these fancier layouts, a Web page designer needs to understand (1) the CSS Box Model; (2) how the browser naturally flows HTML elements and content onto the page. After we learn about these two, then we'll create layouts using two different methods: Floats and Positioning.

Table of Contents

Box Model -----	1
Content -----	2
Padding -----	2
Border -----	3
Margins -----	3
Flow -----	4
Block-Level -----	4, 5
Inline -----	6, 7
Margins for Block-Level and Inline Elements -----	8, 9
Constructing a Multicolumn Layout -----	10 – 12
Float -----	13 – 18
Liquid, Frozen, and Jello Layouts -----	19
Liquid (Fluid) Layout -----	19
Frozen Layout -----	19
Jello Layout -----	20 - 25
Positioning -----	26
Absolute Positioning -----	26 - 30
Z-Indexes -----	27 – 29
Fixed Positioning -----	31 - 32
Relative Positioning -----	33 - 34

Box Model

It is important to understand the Box Model before creating multicolumn layouts. Pages 186 and 187 in your book provide good information about the box model. Read and study these pages. Below is a summary of the box model, along with some details in addition to what is in the book.

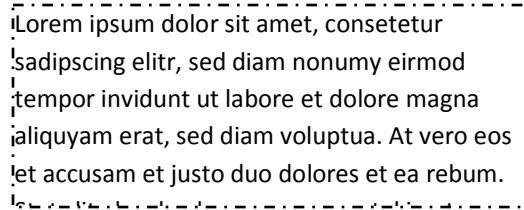


CSS treats every element on a Web page as a box; paragraphs, headings, lists, even links, and the inline `` element.

- Every box is made up of a content area, along with optional padding, border, and margins.
- The content area holds the content, such as text, an image, links, lists, etc.
- The padding is optional and transparent.
- The border is optional and can be seen.
- The margin is optional and transparent.

CONTENT

Since CSS treats every element as a box, the content is placed inside a box that is just big enough to hold it. The content has no extra spaces between the content and the edge of the box.

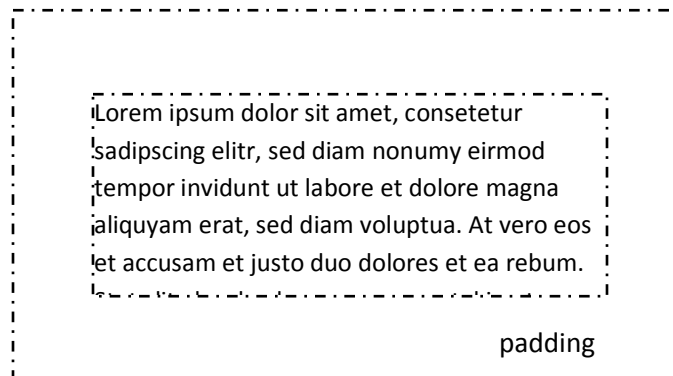


!Lorem ipsum dolor sit amet, consetetur
!s adipiscing elit, sed diam nonummy eirmod
!tempor invidunt ut labore et dolore magna
!aliquyam erat, sed diam voluptua. At vero eos
!et accusam et justo duo dolores et ea rebum.

In the browser, there is no visible edge. The dashed line shown above is so you can see the size of the content area.

PADDING

The Padding is optional and transparent. It has no color or decoration of its own. Padding can be used to add whitespace around the content. Whitespace is often used as part of design. The Padding has up to four values to specify spacing; top, right, bottom and left. See page 418, Appendix B, in the back of your book and page 187 for padding properties (names) and values. Note the shorthand examples in your book as it is easier and faster to write, however it may be a little more challenging to debug when problems occur.



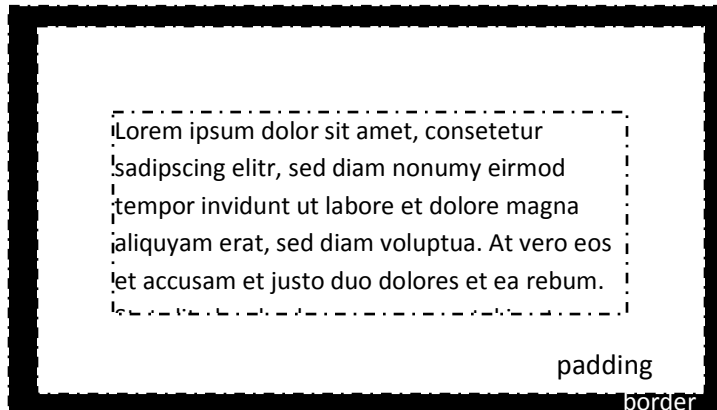
!Lorem ipsum dolor sit amet, consetetur
!s adipiscing elit, sed diam nonummy eirmod
!tempor invidunt ut labore et dolore magna
!aliquyam erat, sed diam voluptua. At vero eos
!et accusam et justo duo dolores et ea rebum.

padding

In the browser, there is no visible edge. The dashed line shown above is so you can see the size of the padding area.

BORDER

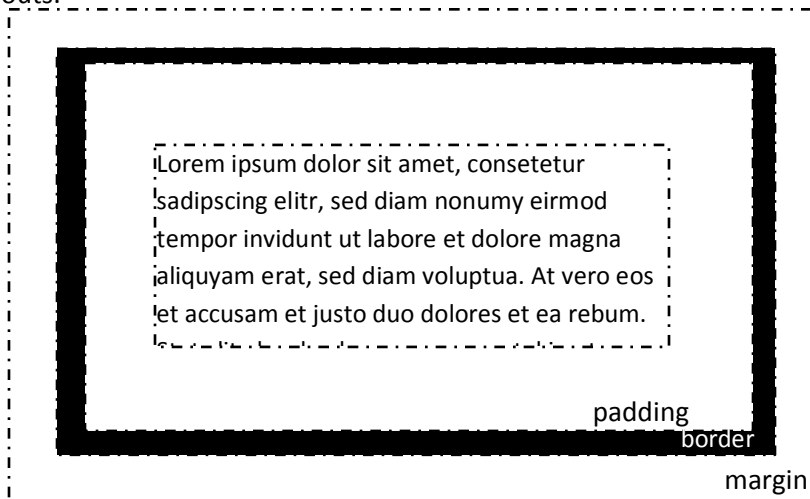
Borders are optional and visible. The border has three values: width, style, and color. See page 417, Appendix B, in the back of your book and page 187 for border properties (names) and values. Note the shorthand examples in your book as it is easier and faster to write, however it may be a little more challenging to debug when problems occur.



MARGINS

The Margin is optional and transparent. Like padding, it has no color or decoration of its own. Margins can be used to add space around elements on the same page. The Margin has up to four values to specify spacing; top, right, bottom and left. See page 418, Appendix B, in the back of your book and page 187 for margin properties (names) and values. Note the shorthand examples in your book as it is easier and faster to write, however it may be a little more challenging to debug when problems occur.

When the browser flows HTML elements, shared margins are collapsed when two block-level elements are placed on top of each other. The element with the largest margin is the only margin that is used. You'll learn more about this shortly (page 8), when learning about how the browser flow HTML elements. It is important to understand how the browser flows HTML elements, before you can effectively create layouts.



Elements

The following is a complete list of all HTML block level elements (although "block-level" is not technically defined for elements that are new in HTML5).

<code><address></code> Contact information.	Field set label.	supported or turned off.
<code><article></code> <small>HTML5</small> Article content.	<code><figcaption></code> <small>HTML5</small> Figure caption.	<code></code> Ordered list.
<code><aside></code> <small>HTML5</small> Aside content.	<code><figure></code> <small>HTML5</small> Groups media content with a caption (see <code><figcaption></code>).	<code><output></code> <small>HTML5</small> Form output.
<code><audio></code> <small>HTML5</small> Audio player.	<code><footer></code> <small>HTML5</small> Section or page footer.	<code><p></code> Paragraph.
<code><blockquote></code> Long ("block") quotation.	<code><form></code> Input form.	<code><pre></code> Preformatted text.
<code><canvas></code> <small>HTML5</small> Drawing canvas.	<code><h1></code> , <code><h2></code> , <code><h3></code> , <code><h4></code> , <code><h5></code> , <code><h6></code> Heading levels 1-6.	<code><section></code> <small>HTML5</small> Section of a web page.
<code><dd></code> Definition description.	<code><header></code> <small>HTML5</small> Section or page header.	<code><table></code> Table.
<code><div></code> Document division.	<code><hgroup></code> <small>HTML5</small> Groups header information.	<code><tfoot></code> Table footer.
<code><dl></code> Definition list.	<code><hr></code> Horizontal rule (dividing line).	<code></code> Unordered list.
<code><fieldset></code>	<code><noscript></code> Content to use if scripting is not	<code><video></code> <small>HTML5</small> Video player.

The above list of block-level elements obtained from:

https://developer.mozilla.org/en/HTML/Block-level_elements

Also see: <http://www.tutorialchip.com/tutorials/HTML5-block-level-elements-complete-list/>

Note: `div` is a generic block-level element that has no special meaning. It is used to mark general block-level content.

The following is a complete list of all HTML inline elements (although “inline” is not technically defined for elements that are new in HTML5).

* will represent Inline Elements new in HTML5

Selector	HTML Use	Selector	HTML Use
a	Anchored Link	label	Label for Form Element
abbr	Abbreviation	legend	Title in Fieldset
address	A Physical Address	link	Resource Reference
area	Area in Image Map	mark*	Marked Rext
audio*	Sound Content	meter*	Measurement Range
bm	Bold text	nav*	Navigation Links
cite	Short Citation	optgroup	Group of Form Options
code	Code Text	option	An Option in a Drop-down List
del	Deleted Text	q	Short Quote
details*	Details of an Element	small	Small Print
dfn	Defined Term	select	Selectable List
command*	Command Button	source*	Media resource
datalist*	Drop-down List	span	Localized Style Formatting
em	Emphasis	strong	Strong Emphasis
font	Font Appearance	sub	Subscript
i	Italic	summary*	Details Header
iframe	Inline sub-window	sup	Superscript
img	Image Embedding	tbody	Table Body
input	Input Field	td	Table Data
ins	Inserted Text	time*	Date/Time
kbd	Keyboard Text	var	Variable

The above list of inline elements obtained from:

<http://www.tutorialchip.com/tutorials/inline-elements-list-whats-new-in-HTML5/>

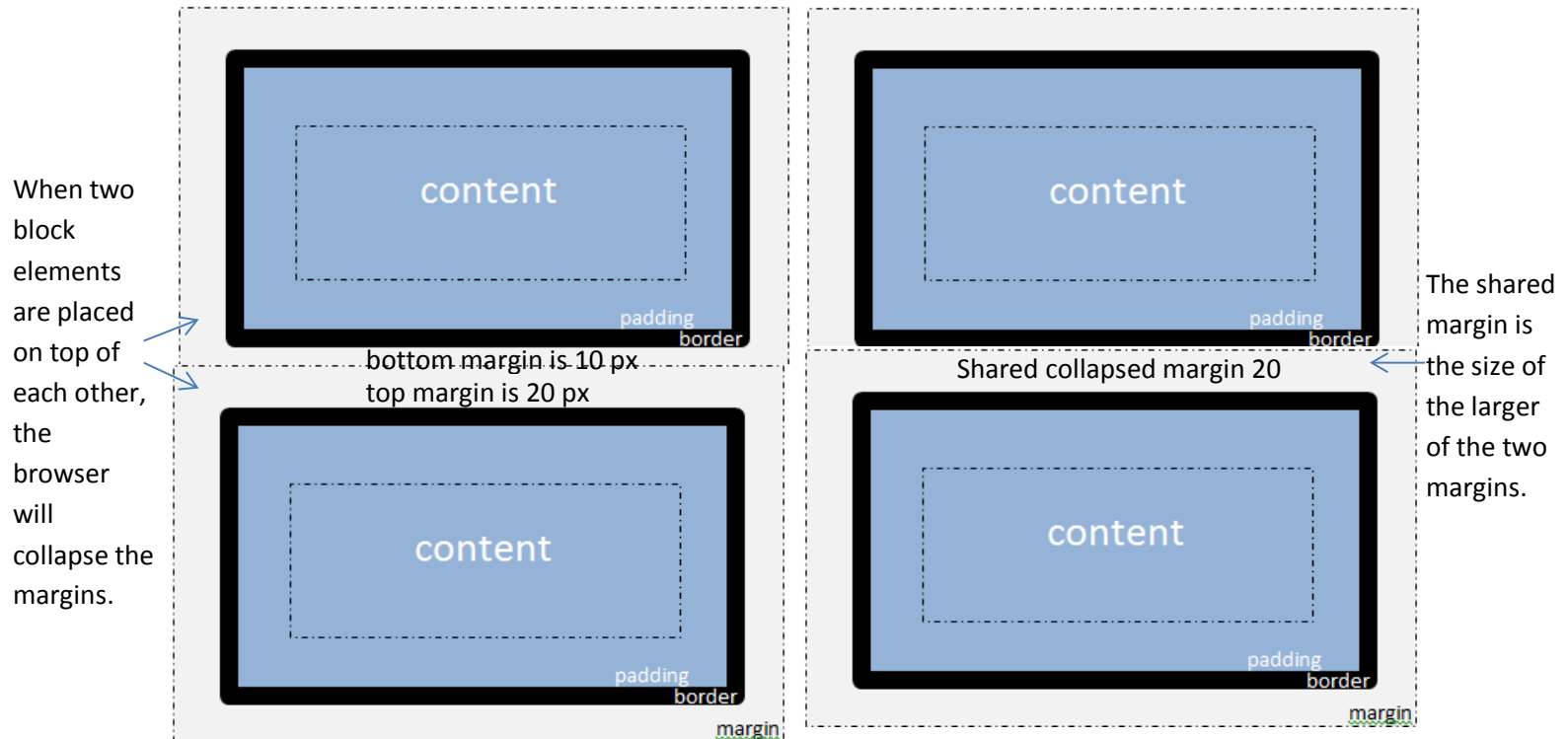
Note: span is a generic inline element that has no special meaning. It is used to mark general inline content.

MARGINS FOR BLOCK-LEVEL AND INLINE ELEMENTS

Back on page 3 you learned about margins in the Box Model . Margins are optional, transparent and can be used to add space around elements. The browser lays out block and inline elements differently when margins are involved. These differences are important to know as they will affect your layouts.

BLOCK ELEMENTS AND MARGINS

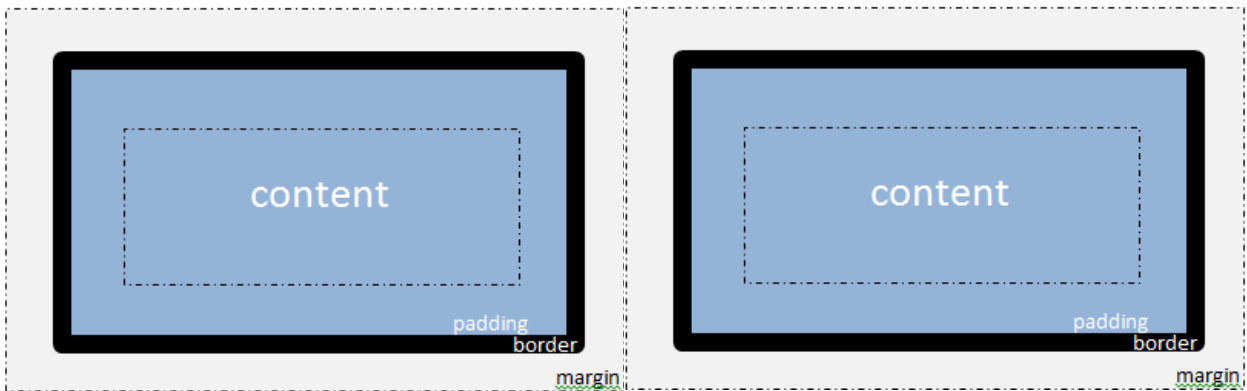
When the browser flows HTML elements, shared margins are collapsed when two block-level elements are placed on top of each other. The element with the largest margin is the only margin that is used.



For example, if the top element's bottom margin is 10px, and the bottom element's top margin is 20 px, then the collapsed margin will be 20 pixels.

INLINE ELEMENTS AND MARGINS

When the browser flows HTML elements, shared margins are left alone when inline elements are placed side by side. Of course the elements are side by side, because they are “inline.” Remember, inline flows horizontally, from top-left to bottom-right. The browser does not put a line break above and below the elements.



Note: You probably will not set the margins for inline elements often. The one exception is images. It is very common to set margins, borders and padding on images.

Constructing a Multicolumn Layout

With an understanding of the box model and flow, we now have the foundations needed to build multicolumn layouts. There are two methods to build multicolumn layouts. We will start with Floats first, and then move on to Positioning.

Download the provided files in Bb. Files include: index.html, layout.css and an image folder containing 5 images. Open the index.html file in your browser. Next, open the index.html and layout.css in your text editor.

Below is the Web page as it looks now. What we want to do is create a two column layout underneath the photo. Take a look at the Web page here and then the markup and CSS that's styling it on the next couple pages.

The nature project is just a gif image and the photo a jpg; both images are contained inside of a **header**.

This list of Sponsors is inside of an **aside** container with a background color.

The main content is inside of an **article** container with a background color.

The **footer** is inside of footer container with a background color.



HTML MARKUP

Each logical section is a descriptive HTML5 tag: header, aside, article and footer. These semantic elements help user agents understand the content of the page.

The nature project is just a gif image and the photo a jpg; both images are contained inside of **header** tags.

This list of Sponsors is inside of an **aside** tag with a background color.

The main content is inside of an **article** tag with a background color.

The **footer** is inside of footer tag with a background color.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>The Nature Project</title>
    <link type="text/css" rel="stylesheet" href="layout-inprogress.css"
  </head>
  <body>
    <header>
      

    <aside id="rightcolumn">
      <p class="bold">Sponsors</p>
      <ul>
        <li>Helpers of America</li>
        <li>Natures Conversationalists</li>
        <li>LLT Limited</li>
        <li>Green of theWorld</li>
        <li>Future of Nature, Inc.</li>
        <li>IPM Limited</li>
      </ul>
    </aside>

    <article>
      <span id="tree">How you can help our cause</h1>
      <p>Its up to you to protect our earth Lorem ipsum dolor sit
      bibendum. Cras id urna. Morbi tincidunt, orci ac convallis aliquam, lectus t
      mattis, purus nec placerat bibendum, dui pede condimentum odio, ac blandit a
      pharetra condimentum, lorem tellus eleifend magna, eget fringilla velit magn
      Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec libero. Susp
      <section>&quot;Lorem ipsum dolor sit amet, consectetur adip
    </article>

    <footer>&copy; The Nature Project Home | Sitemap | RSS Feed<br
      All trademarks and registered trademarks appearing on th
    </footer>
  </body>
</html>
```

THE STYLE RULES

Below is the CSS that is use to style the page.

The body background color, font and margins are set here. Margin of 0 makes sure there is no extra space between the body of the page on the browser edge.

Each logical section for header, article, aside (already given a unique id of rightcolumn), and footer.

The word "Sponsor" in the aside container was given a class of bold and a couple of declarations to make it stand out and align closer to the top.

We'll get to the tree rule later.

```
body {
  background-color: #E5D6A3;
  font-family: Verdana, Geneva, sans-serif;
  font-size: .8em;
  margin: 0px;
}

header {
  margin: 10px;
  height: 294px;
}

article {
  background-color: #FFEEB6;
  padding: 15px;
  margin: 0px 10px 10px 10px;
}

#rightcolumn {
  background-color: #FFEEB6;
  padding: 15px;
  margin: 0px 10px 10px 10px;
}

footer {
  background-color: #403B2D;
  color: #E5D6A3;
  text-align: center;
  padding: 15px;
  margin: 10px;
  font-size: 90%;
}

.bold {
  font-weight: 600;
  font-size: 1.5em;
  margin-top: .5em;
}

#tree {
  float: right;
}
```

FLOAT

The way floats work are closely tied to how the browser flows elements on to the page. You learned about flow on the previous pages. The float property floats an element to the far left or right. It then flows all the content below or around the element. See page 418, Appendix B, in the back of your book for float property (name) and values.

Now that you have taken a look at the markup, lets create two columns and float the aside container, with the list of sponsors, to the right. Steps 1 and 2 below already have been done for you. Complete steps 3 and 4 in the CSS file, save, and then refresh your browser.

Four basic steps to create a float:

- HTML:
1. Give the element you're going to float a unique name using an id.
 2. Make sure the element's HTML is just below the element you want to float under; in this case, the header.
- CSS:
3. Set the width of the element. Any element you wish to float must have a width.
 4. Float the element to the right or left. The only values for a float are right or left

Step 1: Element is given a unique id of rightcolumn. I already did this and you saw the id on the previous pages.

Step 2: The aside is just below the element that we want to float under, in this case, header.

```
<header>
  
  
</header>

<aside id="rightcolumn">
  <p class="bold">Sponsors</p>
  <ul>
    <li>Helpers of America</li>
    <li>Natures Conversationalists</li>
    <li>LLT Limited</li>
    <li>Green of theWorld</li>
    <li>Future of Nature, Inc.</li>
    <li>IPM Limited</li>
  </ul>
</aside>
```

Step 3: Set the width.

Step 4: Float it.

```
#rightcolumn {
  background-color: #FFEEB6;
  padding: 15px;
  margin: 0px 10px 10px 10px;
  width: 250px;
  float: right;
}
```

THE RESULTS

Resize your browser so that it appears as shown below. After setting the width and float for the **rightcolumn** on page 13, the **rightcolumn** is moved as far as possible to the right below the header, and is also removed from the normal flow; it now floats (See below. I added a shadow to the screen capture to more visually demonstrate the float). Everything else below the **rightcolumn** in the HTML is going to move up and around it (compare to the screen capture on page 10).

The **article** (How you can help our cause) is a block element and moves underneath the **rightcolumn** (Sponsors). The text are inline elements and they respect the boundaries of the **rightcolumn**.

Content in **article** container extends underneath the floating **rightcolumn** container with the list of sponsors. Since the text in the **article** container are inline elements, they respect the boundaries of the **rightcolumn**



rightcolumn container is floating on the page. It is not in the normal flow.

Key points:

- Block elements ignore floating elements. The block element, **article**, ignores the floating **rightcolumn** and extends underneath the **rightcolumn**.
- An inline element, the text here, knows the floating **rightcolumn** is there and respects the **rightcolumn** boundaries.

Issues with floating the rightcolumn to the right:

- There is no separation between the two columns. You cannot see the darker background between the two columns as the **article** block-level element is extending all the way to the right and underneath the **rightcolumn**.
- When the browser window is resized smaller, the text in the **article** wraps around and under the **rightcolumn** as shown above.

- When the browser window is resized larger, the **article** and **rightcolumn** extend beyond the **header** above. We'll look at an alternative layout layer to fix this particular problem.

Solution to create the illusion of two columns

I revised the right margin of the **article** block element to be at least the size of the **rightcolumn** (250px), and then I added a little more to account for margins. For the heck of it, I also doubled the list of sponsors in the **rightcolumn** HTML to demonstrate another issue you're about to see. Revise the markup as shown in the yellow highlights below, save, and then refresh your browser.

```
article {  
    background-color: #FFEEB6;  
    padding:         15px;  
    margin:          0px 300px 10px 10px;  
}
```

```
<aside id="rightcolumn">  
    <p class="bold">Sponsors</p>  
    <ul>  
        <li>Helpers of America</li>  
        <li>Natures Conversationalists</li>  
        <li>LLT Limited</li>  
        <li>Green of theWorld</li>  
        <li>Future of Nature, Inc.</li>  
        <li>IPM Limited</li>  
        <li>Helpers of America</li>  
        <li>Natures Conversationalists</li>  
        <li>LLT Limited</li>  
        <li>Green of theWorld</li>  
        <li>Future of Nature, Inc.</li>  
        <li>IPM Limited</li>  
    </ul>  
</aside>
```

THE RESULTS #2

After revising the right margin width of the **article** and adding to the list of sponsors in the **rightcolumn**, below are the results.



Key points:

- The **article** is still extending underneath the **rightcolumn** and taking up the width of the browser window (because that's what block elements do), but the article now has a right margin at least as wide as the rightcolumn to reduce the width of the content within it.
- Margins are transparent (see page 3 for margins), so the darker background shows through, creating the illusion of two columns.
- The **rightcolumn** is still floating (I added a bit of a shadow on the screen capture to remind you) and is ignored by block elements.

Issues:

- The **footer**, a block element (see page 5), extends underneath the floating **rightcolumn** and takes up the width of the browser window. The floating **rightcolumn** appears on top of the footer. Remember, block elements ignore floating elements. The **footer** is ignoring the **rightcolumn**.
- When the browser window is resized larger, the **article** and **rightcolumn** extend beyond the **header** above. Again, we'll look at an alternative layout later to fix this particular problem.

Solution for the rightcolumn overlapping onto the footer

Since the **rightcolumn** has been removed from the flow and is floating, the browser lays out the **article** and **footer** block elements like it normally would, ignoring the **rightcolumn** (remember though that when the browser flows inline elements, it will respect the borders of the floating **rightcolumn** and wrap inline elements, in this case text and an image, around it).

The CSS **clear** property solves this issue. When this property is set on an element, no floating elements are allowed to be on the left, right or both sides of an element. We'll use the clear property on the **footer** to tell the browser to let no floating elements to the right of it. Revise the CSS as shown in the yellow highlights below, save, and then refresh your browser.

```
footer {  
  background-color: #403B2D;  
  color: #E5D6A3;  
  text-align: center;  
  padding: 15px;  
  margin: 10px;  
  font-size: 90%;  
  clear: right;  
}
```

THE RESULTS #3

After clearing the float on the right, the floating **rightcolumn** is no longer allowed on the right side of the **footer**, below are the results.



Key point:

- When the browser places the elements on the page, it looks to see if there is a floating element to the right side of the **footer**, and if there is, it moves the footer down until there is nothing on its right. No matter how wide you open the browser, the footer will always be below the **rightcolumn**.

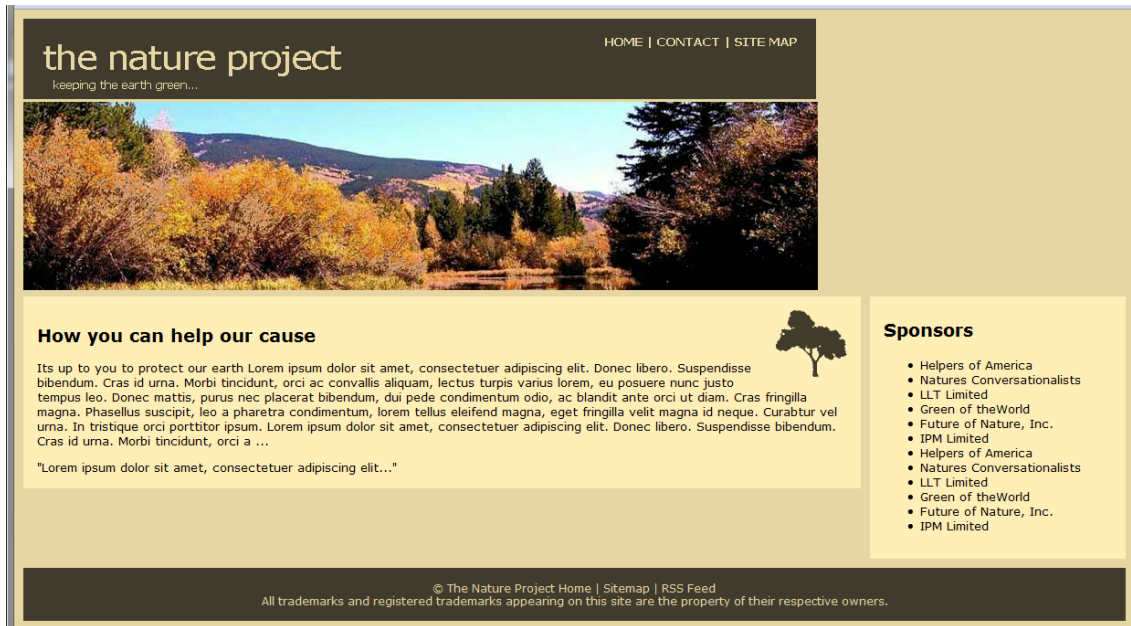
Issues:

- The column heights now look different. We could make each column extend down to meet the footer by adding a name value pair of height: 250 for each the article and rightcolumn. Sometimes it is just a matter of playing around with the CSS and HTML until you achieve the results you want.
- We still have the problem that when the browser window is resized larger, the **article** and **rightcolumn** extend beyond the **header** above them. Next, we'll look at an alternative layout to fix this particular problem.
- Another issue in the way the page was designed is the order in which the **rightcolumn** and article were placed in the HTML (refer back to page 11). Within the **body**, the **header** is first, then the **rightcolumn** and then the **article**. Anyone with a browser that has limited capabilities (PDAs, mobile phones, screen readers, etc.) will see the page in the order it is written in the HTML, with the list of sponsors in the **rightcolumn** first. The **article**, the much more important part, should be displayed first.

LIQUID, FROZEN, AND JELLO LAYOUTS

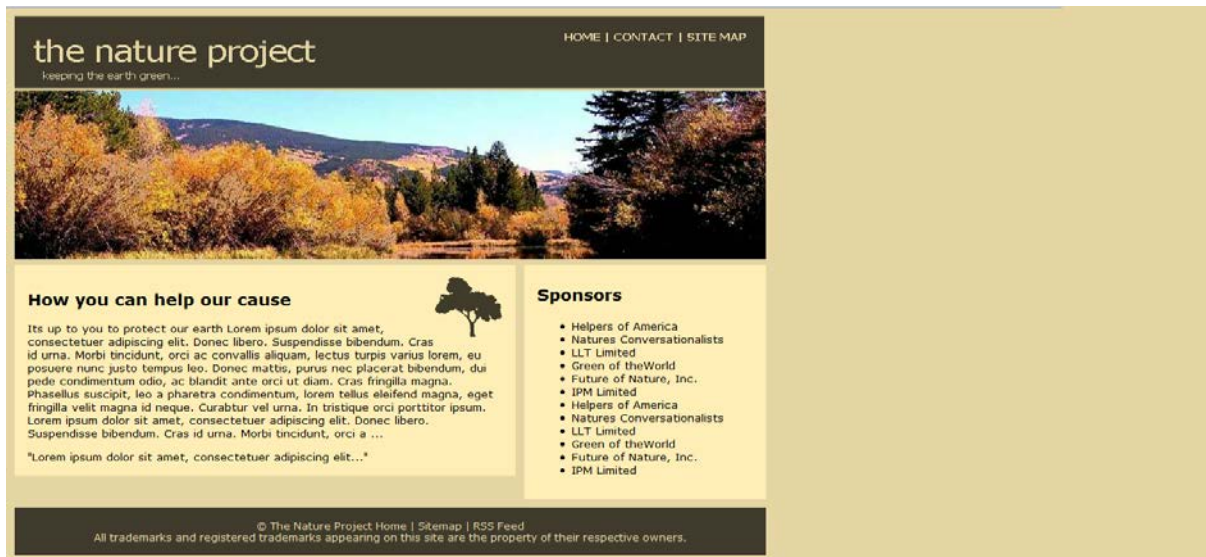
Liquid Layout (also called Fluid Layout)

On the previous pages, the design was a liquid layout (also see below). The content filled the browser window at any size. This design takes advantage of all the available space, but layout can be more of a challenge to control, especially for inexperienced Web designers. Look at amazon.com for another example of a liquid layout.



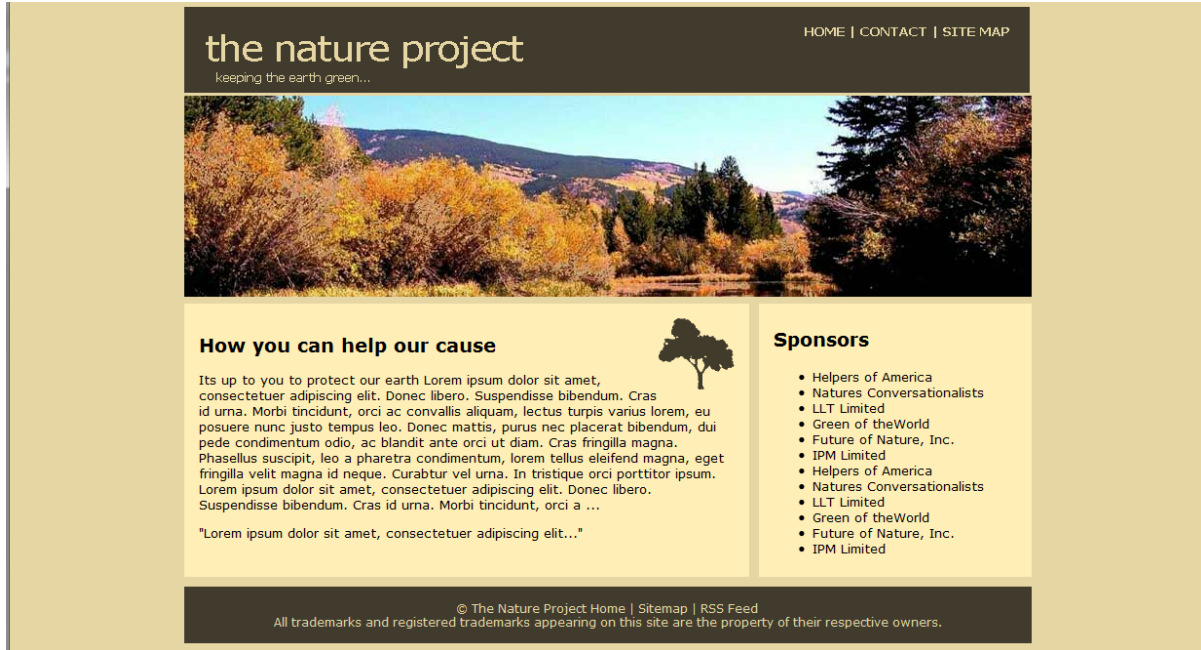
Frozen Layout

Everything is locked down on the page and when a user resizes the browser window; your design will look as it should (see below). However, there is a lot of empty space on the right that makes it look bad.



Jello Layout

Jello is between liquid and frozen (see below). It locks down everything on the page like frozen, but centers the content in the browser. The left and right margins, around the content expand to fill the browser window like liquid. Jello is a popular layout on the Web and easier to control than liquid. Look at craigslist.com or my site at ckautz.org for additional examples of a jello layout.



Next, we will create a Jello Layout, but first a couple of tweaks are necessary.

Jello Layout

Next we'll make the page a Jello layout so that it appears as shown on page 20. First a couple of additional revisions:

1. Remember the four basic steps to create a float on page 13? Step 2 stated: make sure the element's HTML is just below the element you want to float under; in this case, the **header**. After we moved our HTML in the correct order, the **article** is now directly below the **header**. This means we need to float the **article** to the left and remove the float for the **rightcolumn**. See below.
2. Step 1 on page 13 stated: give the element you need to float a unique name using an id. Therefore, we'll need to give the **article** a unique id. You can use any id that you want. I'll use **leftcolumn** as it is descriptive. We no longer need the id for the **rightcolumn** in the aside tag, and so we'll remove it. See below.

Revise your HTML markup as shown below. **rightcolumn** is removed. **leftcolumn** is added. You want your markup to look as shown on the right for "After revision." Don't refresh your browser yet as it still is not ready.

Before revision

```
<article>
...
</article>
```

```
<aside id="rightcolumn">
...
</aside>
```

After revision

```
<article id="leftcolumn">
...
</article>
```

```
<aside>
...
</aside>
```


Next, revise the CSS to reflect the changes in the HTML. Revise the selectors and properties as highlighted below. `article` is changed to `#leftcolumn`, `#rightcolumn` is changed to `aside`. You want your CSS to look as shown on the right for "After revisions." Save your work, but still not ready yet to refresh the browser.

*We don't need the wide right margin anymore. It is changed back to 10.

Float the **leftcolumn** to the left. 500 is an arbitrary number I picked. I just want the **leftcolumn** bigger than the **aside** with the Sponsors.

Before revisions

```

article {
  background-color: #FFEEB6;
  padding: 15px;
  margin: 0px 300px 10px 10px;
  height: 250px;
}

```

After revisions

```

#leftcolumn {
  background-color: #FFEEB6;
  padding: 15px;
  margin: 0px 10px 10px 10px;
  height: 250px;
  width: 500px;
  float: left;
}

```

```

#rightcolumn {
  background-color: #FFEEB6;
  padding: 15px;
  margin: 0px 10px 10px 10px;
  width: 250px;
  float: right;
  height: 250px;
}

```

```

aside {
  background-color: #FFEEB6;
  padding: 15px;
  margin: 0px 10px 10px 550px;
  width: 250px;
  float: right;
  height: 250px;
}

```

The fixed width and float will now be on the **leftcolumn**, and so we need to remove it from the **aside**.

*The **aside** will now flow under the **leftcolumn** content, and so we need to move the large left margin to the **aside**. The total width of the **leftcolumn** is 500. The **leftcolumn** also has 10px of margin on each side + 15px of padding on each side = 550px.

*If you wondering how you know which value is the top, right, bottom, or left, refer to the shorthand examples on page 187 in your book.

Jello layout continued . . .

Since the leftcolumn is now going to float to the left, we need to adjust the footer to clear everything to the left, rather than the right.

```
footer {  
background-color: #403B2D;  
color:           #E5D6A3;  
text-align:      center;  
padding:         15px;  
margin:          10px;  
font-size:       90%;  
clear:           left;  
}
```

Now that all the revisions are complete, we need to add a new, generic <div> element with the id of “wrapper”. You can use any id you wish. I choose this one because it “wraps” all the content. Place the <div> tag after the opening <body> tag and before the closing </body> tag.

```
<body>  
  <div id="wrapper">  
    . . . all HTML here  
  </div>  
</body>
```

Next, we’ll limit the <div> to 890px to constrain the size of all the elements and content in the “wrapper”. Add the CSS rule:

```
#wrapper {  
  width:      890px;  
  margin-left: auto;  
  margin-right: auto;  
}
```

I used 890px fixed width here, but many designers like to use a popular 960px width. See page 198 in your book for “Determining page widths.”

The “auto” value for margins lets the browser figure out the correct margins, making sure they are the same so that the content is centered.

Okay, save all your work and NOW refresh your browser. Your page should now look as shown on page 25. Trouble? Go back and check your markup. One missed semi-colon or # or closing bracket, etc. could affect the entire layout.

Completed Jello Layout

Below is the completed Jello layout, the same as shown on page 20. I just added a shadow on the screen capture of the **leftcolumn** here to be sure you understand that this is the area that is now floating. The **aside** with the Sponsors is now extending underneath the **leftcolumn**. The text in the **aside** are inline elements, which respect the boundaries of the floating **leftcolumn**. Therefore, the text does not go underneath the floating **leftcolumn**.



What if you wanted to create a three column layout? For the left column, float it to the left. For the right column, float it to the right. For the middle column, add left and right margins that sit under the two floating elements. For the footer at the bottom of the page, be sure to add the **clear: both** name value pair so that no elements will be allowed to the left or to the right of the footer. Keep this in mind as an idea for your Final at the end of the semester.

POSITIONING

The Positioning property can be used to create the same layout as was done with floats. We'll create the Jello layout as shown on page 25, except use the Position property instead. Not much editing is needed. We will start with Absolute Positioning (along with z-index), and then take a look at Fixed and Relative Positioning.

ABSOLUTE POSITIONING

For the leftcolumn add the position and top properties as shown below. Remove the float property as we are no longer using it. Revise the markup as shown in the yellow highlights below, save, and then refresh your browser. Everything should look the same.

```
#leftcolumn {  
  background-color: #FFEEB6;  
  padding: 15px;  
  margin: 0px 10px 10px 10px;  
  height: 250px;  
  width: 500px;  
  position: absolute;  
  top: 314px;  
}
```

Since we removed the float property from the **leftcolumn**, we also remove the clear from the footer. Flowed elements do not know about absolute elements, therefore **clear** means nothing for positioning. Bottom line: when using **position** instead of **float** for layout, you do not need **clear**.

```
footer {  
  background-color: #403B2D;  
  color: #E5D6A3;  
  text-align: center;  
  padding: 15px;  
  margin: 10px;  
  font-size: 90%;  
}
```

clear: left
removed from footer rule.

The result should look the same as the completed Jello Layout created with floats on page 25. The **leftcolumn** is floating with Positioning, the same as it was with Floats.

Key points:

- The **aside** column with the Sponsors has a left margin of 550px to force it further to the right, otherwise, it would appear underneath the **leftcolumn** as the **leftcolumn** is floating. Try it. Change the left margin of the **aside** from 550px to 10px and see what happens, then change it back again.
- The **leftcolumn** is 314px from the top of the browser window (header height of 294px + 10px of margin on the top and bottom = 314px).

- The default for positioning is static, which places the element in the normal flow. The browser decides where to put it, not you.
- You can absolutely position any block or inline element; however it is much more common to position block elements. One exception for inline is the `img` element.
- Width for absolutely positioned elements does not have to be specified, but you will likely want to do so for greater control.
- Pixels were used to position elements in the above example, but you may use percentages instead. With percentages, the positions of your elements may appear to change as you change the width of the browser window. For example, if your browser is 800 pixels wide, and your element's left position is set to 10%, then your element will be 80 pixels from the left of the browser window. If your browser is instead resized to 400 pixels wide then the width will be reduced to 10% of 400 pixels, or 40 pixels from the left of the browser window.

Z-INDEXES

Z-indexes are often used with Absolute Positioning. It is used to determine the stacking order of absolutely positioned elements on a Web page.

We'll make a couple of changes to the tree image to see how this works. The tree image has been floating to the right side of the **leftcolumn**. We'll pull it out of the **leftcolumn** and then position it absolutely to demonstrate how z-indexes work.

1. First, in the HTML page, move the inline span element, with the id `tree`, from its current position below the **leftcolumn**, to above the **leftcolumn** as shown below. Add paragraph tags around the span element (the `</p>` tag is out of view here, but be sure to add it).

```

<body>
  <div id="wrapper">
    <header>
      
      
    </header>
    <p><span id="tree"></p>
    <article id="leftcolumn">
      <h1>How you can help our cause</h1>
      <p>Its up to you to protect our earth Lorem ipsum dolor sit amet, consectetur
      ero. Suspendisse bibendum. Cras id urna. Morbi tincidunt, orci ac convallis aliquam, lect
      uere nunc justo tempus leo. Donec mattis, purus nec placerat bibendum, dui pede condiment
      i ut diam. Cras fringilla magna. Phasellus suscipit, leo a pharetra condimentum, lorem te
  
```

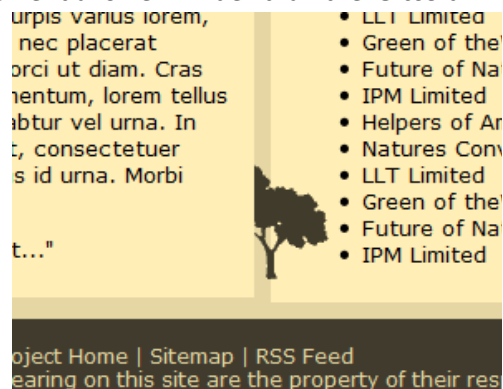
Note: according to W3C, inline elements (**span** in this case) cannot be placed directly inside the **body** element; they must be wholly nested within block-level elements. The `<p>` tag is a block-level element and that is why we added the `<p>` element around the **span** element.

- Next, in the CSS, remove the current name value pair for the tree rule, and replace with the following:

```
#tree {  
    position: absolute;  
    top: 510px;  
    left: 605px;  
}
```

The tree img will be positioned 510px from the top edge of the page and 605px from the left edge of the page. After revising the tree rule, save, and then refresh your browser.

- Resize your browser just right so that the tree img element looks as shown below. The absolutely positioned img appears underneath the absolutely positioned **leftcolumn**. The img element has a lower stacking order and appears underneath the **leftcolumn**. Some browsers will give the img element a lower z-index than the **leftcolumn**.



- To fix this problem, add the z-index property as shown below:

```
#tree {  
    position: absolute;  
    top: 510px;  
    left: 605px;  
    z-index: 99;  
}
```

The number 99 was added to make sure the tree is always stacked on top. It doesn't matter what number you use, as long as it is higher than any other absolutely positioned element on the page. The element with the higher number is always positioned on top of the element with a lower number (or no number). In this case, we're making sure the tree element is stacked above the absolutely positioned **leftcolumn**.

- Save your work and refresh your browser. Below are the results after the z-index was added to the tree rule:



What would happen if the tree image was INSIDE of the absolutely positioned **leftcolumn**? Let's find out.

- First, in the HTML page, move the inline span element, with the id tree, back to its original position; below the **leftcolumn**. Remove the paragraph tags around the span element. <p> tags are no longer needed as the inline span element is now inside the block-level **article**. Revise the markup as shown in the yellow highlights below.

```

<body>
  <div id="wrapper">
    <header>
      
      
    </header>
    <article id="leftcolumn">
      <span id="tree"></span>
      <h1>How you can help our cause</h1>
      <p>Its up to you to protect our earth Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec mattis. nunc justo tempus leo. Donec mattis. nunc nec placerat bibendum. dui pede condimentum odio. ac blandit ante
    </article>
  </div>

```

- After saving your revision, refresh your browser. Where did the tree image go? When you position an element, you're specifying the position relative to the closest ancestor element that is also positioned. Since we moved the absolutely positioned tree image inside of the absolutely positioned **leftcolumn**, the tree image element is positioned relative to the **leftcolumn** element (the ancestor). The tree image is positioned 510px from the top of the **leftcolumn** and 605px from the left of the **leftcolumn**, completely out of view!

Note: when the tree image was outside of the **leftcolumn** element, the tree was positioned relative to the <HTML> element, which was the top and left of the browser window.

- To solve the problem, revise the top and left values in the tree rule. The values are adjusted from the top and left of the absolutely positioned **leftcolumn**, NOT the edge of the page. I determined the values below by simple trial and error until I had the position I preferred. Revise the tree rule as shown below in the yellow highlights, save, and then refresh your browser.

```
#tree {
  position: absolute;
  top: 5px;
  left: 465px;
  z-index: 99;
}
```

- Below are the results after the values have been revised:



left: 465px
Measured from the left of the leftcolumn.



FIXED POSITIONING

Fixed Positioning allows you to place an element on the page and the element won't move, even if you scroll the page. To see how it works, we will add a donate image to the page that a user can click to donate money to the nature project.

1. First, add the following markup just above the footer in the HTML page. It is inserted above the footer here, but it does not really matter where in the HTML it is placed, unless the browser does not support positioning. The element is not the most important on the page, and so it will be placed near the bottom.

```
</aside>
```

```
<div id="donate">
  <a href="index.html" title="Click here to donate to the cause">
    
  </a>
</div>
```

```
<footer>&copy; The Nature Project Home | Sitemap | RSS Feed<br />
  All trademarks and registered trademarks appearing on this site are the
```

Note: according to W3C, inline elements (**img** in this case) cannot be placed directly inside the **body** element; they must be wholly nested within block-level elements. The generic **<div>** is a block-level element and that is why we added the **<div>** around the **img** element. We also needed to give the element a unique id.

2. Next, add the following rule to the CSS:

```
#donate {
  position: fixed;
  top: 300px;
  left: 0px;
}
```

After revising the markup as shown above, save, and then refresh your browser.

3. The fixed positioned donate image appears in the same place, regardless of the browser size or if the user scrolls. Resize the browser to a smaller window and scroll. Note the fixed position of the donate image.



4. Use a negative left property value to make it look like the coupon is partially off the left side of the screen. Change the left value from 0px to -30 and see what happens!

RELATIVE POSITIONING

Relative Positioning is not used as often as the others, but is still worth mentioning. Unlike absolute and fixed positioning, an element that is relatively positioned is still part of the flow of the page, but at the last moment, just before the element is displayed, the browser offsets its position.

So, the element still takes up the same spot on the page, it's just displayed in a different location. The spot that it is actually taking up looks empty. In other words, the original location is preserved.

1. To see this first hand, add the following to your HTML page. Also, remove the last 6 sponsors in the unordered list `` as they are duplicates that were used in a previous example. Save your file and refresh your browser window.

```
</article>

<aside>
  <p class="bold">Sponsors</p>
  <div id="thankyou">
    
  </div>
  <ul>
    <li>Helpers of America</li>
    <li>Natures Conversationalists</li>
    <li>LLT Limited</li>
    <li>Green of theWorld</li>
    <li>Future of Nature, Inc.</li>
    <li>IPM Limited</li>
  </ul>
</aside>
```

Delete the duplicate list of sponsors, the last 6.

2. Look at how the image appears in the browser, before the CSS is applied. This is its original location before we add the offset.



3. Next, add the CSS rule as shown below, save the file, and then refresh your browser again.

```
#thankyou {  
  position: relative;  
  left: 220px  
}
```

4. Look at how the page appears, after the CSS is applied. We specified that the thank you image should be displayed **220px from the left of where it sits in the flow of the document**.

The element still takes up the same spot on the page, it's just displayed in a different location. The original location looks empty and it's space is preserved.



Note: you can use right, top and bottom properties too; not just left. On page 190 in your book, the only properties listed are top and left.

Your completed work should look as shown on the next page.

Save your work, then submit files as instructed in Bb for Unit H.

the nature project

keeping the earth green...

HOME | CONTACT | SITE MAP



Donate Now

How you can help our cause

Its up to you to protect our earth Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec libero. Suspendisse bibendum. Cras id urna. Morbi tincidunt, orci ac convallis aliquam, lectus turpis varius lorem, eu posuere nunc justo tempus leo. Donec mattis, purus nec placerat bibendum, dui pede condimentum odio, ac blandit ante orci ut diam. Cras fringilla magna. Phasellus suscipit, leo a pharetra condimentum, lorem tellus eleifend magna, eget fringilla velit magna id neque. Curabitur vel urna. In tristique orci porttitor ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec libero. Suspendisse bibendum. Cras id urna. Morbi tincidunt, orci a ...

"Lorem ipsum dolor sit amet, consectetur adipiscing elit..."



Sponsors

- Helpers of America
- Natures Conversationalists
- LLT Limited
- Green of theWorld
- Future of Nature, Inc.
- IPM Limited

Thank you

© The Nature Project Home | Sitemap | RSS Feed
All trademarks and registered trademarks appearing on this site are the property of their respective owners.