

Chapter 3: Designing a Page Layout: 3-15a The CSS Positioning Styles  
Book Title: New Perspectives on HTML5, CSS3, and JavaScript  
Printed By: Kirsten Markley (markleyk@otc.edu)  
© 2018 Cengage Learning, Cengage Learning

## 3-15a The CSS Positioning Styles

CSS supports several properties to place objects at specific coordinates within the page or within their container. To place an element at a specific position within its container, you use the following style properties

```
position: type;  
top: value;  
right: value;  
bottom: value;  
left: value;
```

where *type* indicates the kind of positioning applied to the element, and the *top*, *right*, *bottom*, and *left* properties indicate the coordinates of the top, right, bottom, and left edges of the element, respectively. The coordinates can be expressed in any of the CSS measuring units or as a percentage of the container's width or height.

CSS supports five kinds of positioning: *static* (the default), *relative*, *absolute*, *fixed*, and *inherit*. In **static positioning** ([A layout technique that places an element where it would have fallen naturally within the flow of the document.](#)), the element is placed where it would have fallen naturally within the flow of the document. This is essentially the same as not using any CSS positioning at all. Browsers ignore any values specified for the *top*, *left*, *bottom*, or *right* properties under static positioning.

Chapter 3: Designing a Page Layout: 3-15a The CSS Positioning Styles  
Book Title: New Perspectives on HTML5, CSS3, and JavaScript  
Printed By: Kirsten Markley (markleyk@otc.edu)  
© 2018 Cengage Learning, Cengage Learning

© 2018 Cengage Learning Inc. All rights reserved. No part of this work may be reproduced or used in any form or by any means - graphic, electronic, or mechanical, or in any other manner - without the written permission of the copyright holder.

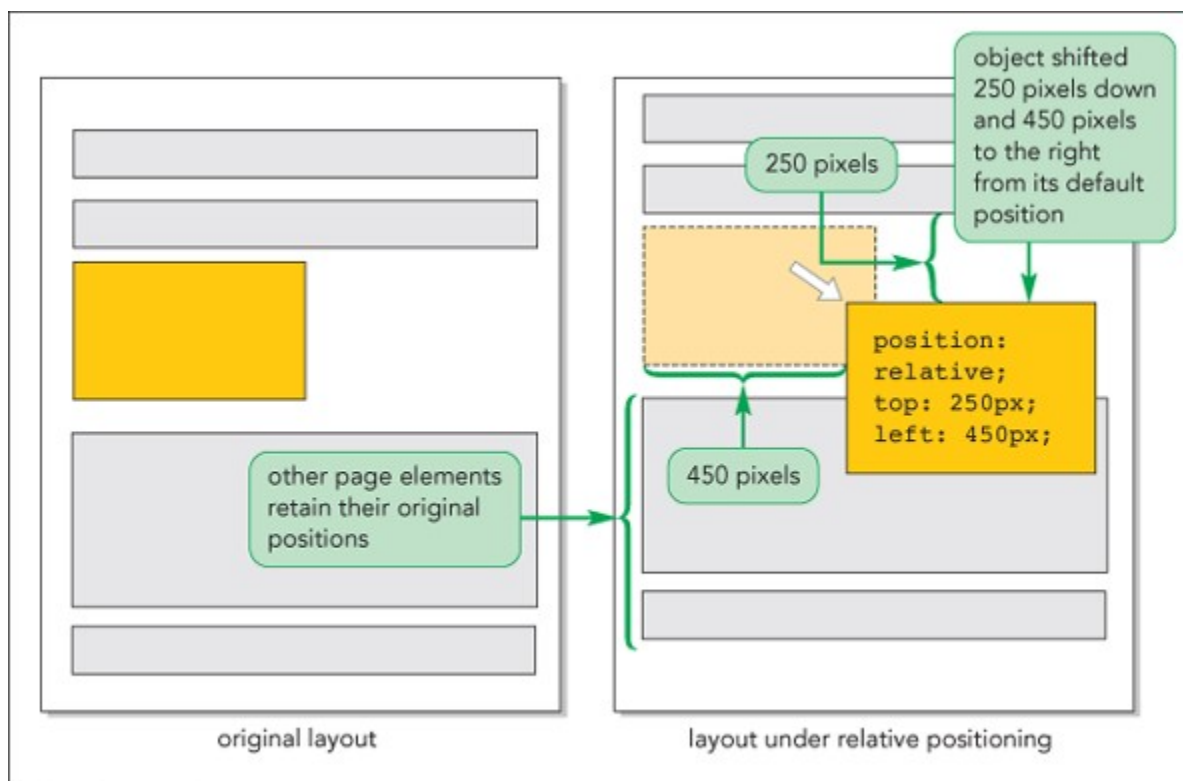
## 3-15b Relative Positioning

Relative positioning is used to nudge an element out of its normal position in the document flow. Under relative positioning, the `top`, `right`, `bottom`, and `left` properties indicate the extra space that is placed alongside the element as it is shifted into a new position. For example, the following style rule adds 250 pixels of space to the top of the element and 450 pixels to the left of the element, resulting in the element being shifted down and to the right (see [Figure 3-46](#)):

```
div {  
    position: relative;  
    top: 250px;  
    left: 450px;  
}
```

**Figure 3-46**

### Moving an object using relative positioning



© 2016 Cengage Learning

© 2016 Cengage Learning

Note that the layout of the other page elements are not affected by relative positioning; they will still occupy their original positions on the rendered page, just as if the object had never

been moved at all.

Relative positioning is sometimes used when the designer wants to “tweak” the page layout by slightly moving an object from its default location to a new location that fits the overall page design better. If no top, right, bottom, or left values are specified with relative positioning, their assumed values are 0 and the element will not be shifted at all.

Chapter 3: Designing a Page Layout: 3-15b Relative Positioning

Book Title: New Perspectives on HTML5, CSS3, and JavaScript


Printed By: Kirsten Markley (markleyk@otc.edu)

© 2018 Cengage Learning, Cengage Learning

© 2018 Cengage Learning Inc. All rights reserved. No part of this work may be reproduced or used in any form or by any means - graphic, electronic, or mechanical, or in any other manner - without the written permission of the copyright holder.

## 3-15c Absolute Positioning

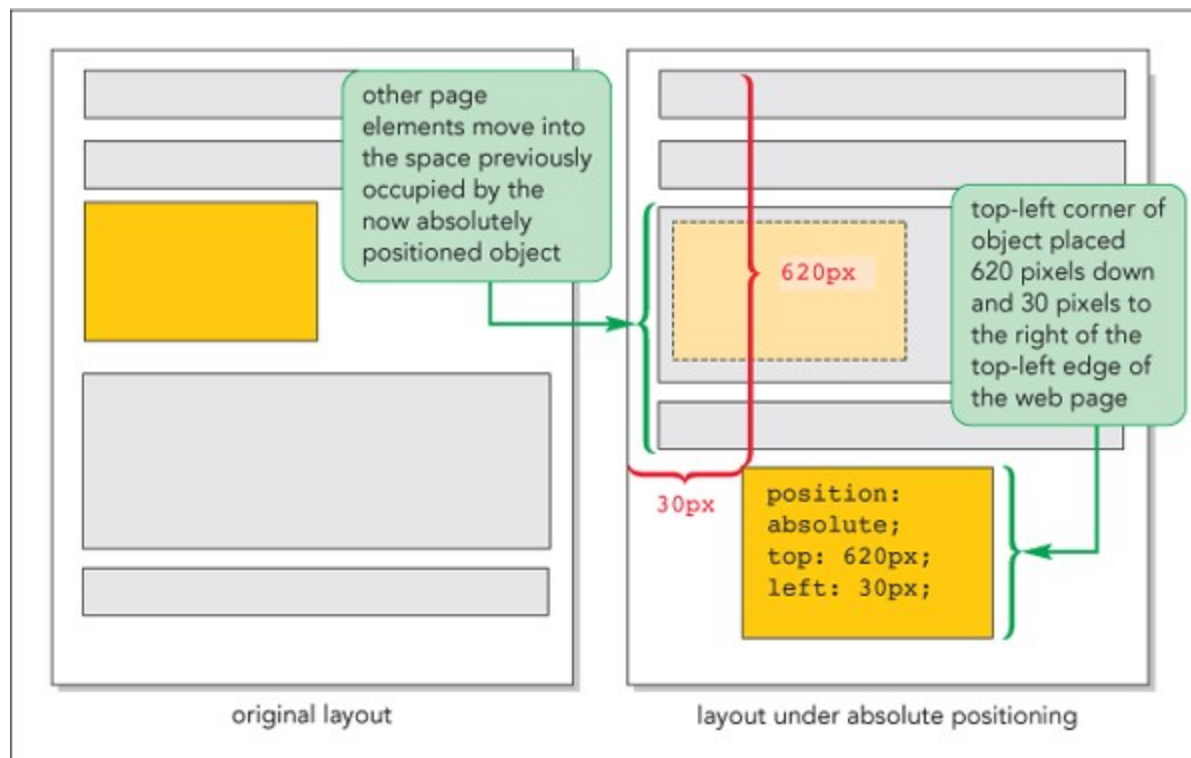
Absolute positioning places an element at specific coordinates within a container where the `top` property indicates the position of the element's top edge, the `right` property sets the position of the right edge, the `bottom` property sets the bottom edge position, and the `left` property sets the position of the left edge.

 For example, the following style rule places the `header` element 620 pixels from the top edge of its container and 30 pixels from the left edge (see [Figure 3-47](#)).

```
header {  
    position: absolute;  
    top: 620px;  
    left: 30px;  
}
```

**Figure 3-47**

### Moving an object using absolute positioning




© 2016 Cengage Learning

© 2016 Cengage Learning

To place an object with absolute positioning, you use either the top/left coordinates or the bottom/right coordinates, but you don't use all four coordinates at the same time because

that would confuse the browser. For example an object cannot be positioned along both the left and right edge of its container simultaneously.

As with floating an element, absolute positioning takes an element out of normal document flow with subsequent elements moving into the space previously occupied by the element. This can result in an absolutely positioned object overlapping other page elements.


 The interpretation of the coordinates of an absolutely positioned object are all based on the edges of the element's container. Thus the browser needs to “know” where the object's container is before it can absolutely position objects within it. If the container has been placed using a `position` property set to `relative` or `absolute`, the container's location is known and the coordinate values are based on the edges of the container. For example the following style rules place the `article` element at a coordinate that is 50 pixels from the top edge of the `section` element and 20 pixels from the left edge.

```
section {  
    position: relative;  
}  
section > article {  
    position: absolute;  
    top: 50px;  
    left: 20px;  
}
```

Note that you don't have to define coordinates for the `section` element as long as you've set its position to `relative`.

The difficulty starts when the container has not been set using `relative` or `absolute` positioning. In that case, the browser has no context for placing an object within the container using absolute positioning. As a result, the browser must go up a level in the hierarchy of page elements, that is, to the container's container. If that container has been placed with `absolute` or `relative` positioning, then any object nested within it can be placed with absolute positioning. For example, in the following style rule, the position of the `article` element is measured from the edges of the `body` element, not the `section` element:

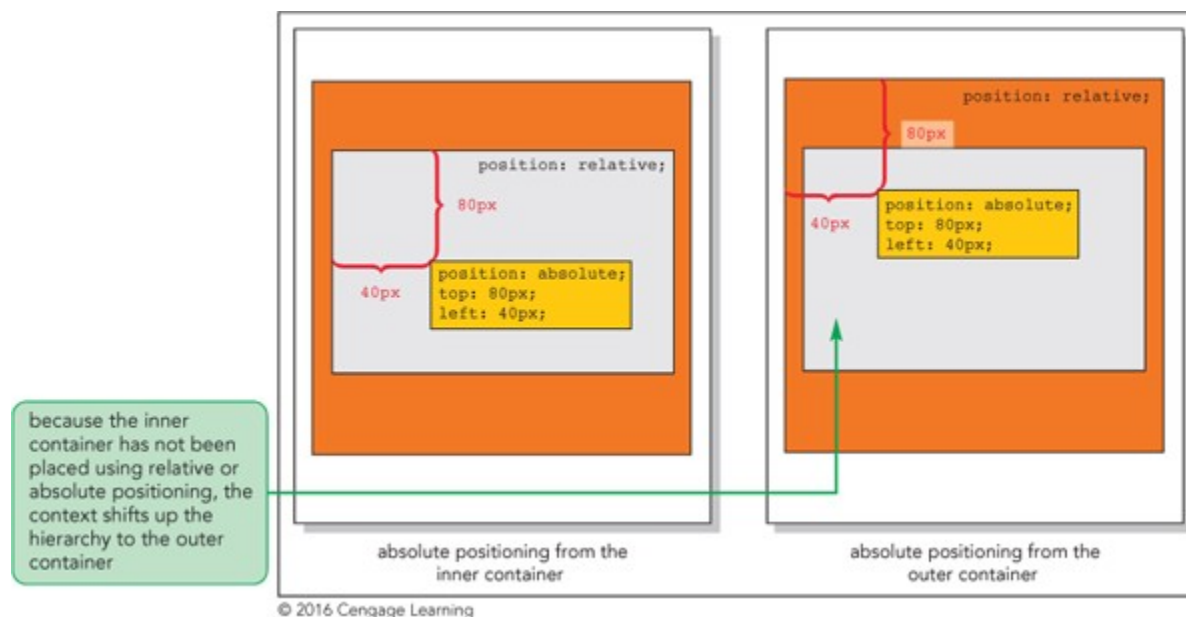
```
body {position: absolute;}  
  
body > section {position: static;}  
  
body > section > article {  
    position: absolute;  
    top: 50px;  
    left: 20px;  
}
```

 Proceeding in this fashion the browser will continue to go up the hierarchy of elements until it finds a container that has been placed with `absolute` or `relative` positioning or it

reaches the root `html` element. If it reaches the `html` element, the coordinates of any absolutely positioned object are measured from the edges of the browser window itself. [Figure 3-48](#) shows how the placement of the same object can differ based on which container supplies the context for the `top` and `left` values.

**Figure 3-48**

### Context of the top and left coordinates



© 2016 Cengage Learning

Coordinates can be expressed in percentages as well as pixels. Percentages are used for flexible layouts in which the object should be positioned in relation to the width or height of its container. Thus, the following style rule places the `article` element halfway down and 30% to the right of the top-left corner of its container.

```
article {
  position: absolute;
  top: 50%;
  left: 30%;
}
```

As the container of the `article` changes in width or height, the `article`'s position will automatically change to match.

Chapter 3: Designing a Page Layout: 3-15c Absolute Positioning  
 Book Title: New Perspectives on HTML5, CSS3, and JavaScript  
 Printed By: Kirsten Markley (markleyk@otc.edu)  
 © 2018 Cengage Learning, Cengage Learning

© 2018 Cengage Learning Inc. All rights reserved. No part of this work may be reproduced or used in any form or by any means - graphic, electronic, or mechanical, or in any other manner - without the written permission of the copyright holder.

## 3-15d Fixed and Inherited Positioning

When you scroll through a document in the browser window, the page content scrolls along. If you want to fix an object within the browser window so that it doesn't scroll, you can set its `position` property to **fixed**. For example, the following style rule keeps the `footer` element at a fixed location, 10 pixels up from the bottom of the browser window:

```
footer {  
    position: fixed;  
    bottom: 10px;  
}
```

Note that a fixed object might cover up other page content, so you should use it with care in your page design.

Finally, you can set the `position` property to **inherit** so that an element inherits the position value of its parent element.

### Reference

#### Positioning Objects with CSS

- To shift an object from its default position, use the properties

```
position: relative;  
top: value;  
left: value;  
bottom: value;  
right: value;
```

where *value* is the distance in one of the CSS units of measure that the object should be shifted from the corresponding edge of its container.

- To place an object at a specified coordinate within its container, use the properties

```
position: absolute;  
top: value;  
left: value;  
bottom: value;  
right: value;
```

where *value* is a distance in one of the CSS units of measure or a

percentage of the container's width or height.

- To fix an object within the browser window so that it does not scroll with the rest of the document content, use the property

```
position: fixed;
```

Chapter 3: Designing a Page Layout: 3-15d Fixed and Inherited Positioning

Book Title: New Perspectives on HTML5, CSS3, and JavaScript

Printed By: Kirsten Markley (markleyk@otc.edu)

© 2018 Cengage Learning, Cengage Learning

© 2018 Cengage Learning Inc. All rights reserved. No part of this work may be reproduced or used in any form or by any means - graphic, electronic, or mechanical, or in any other manner - without the written permission of the copyright holder.

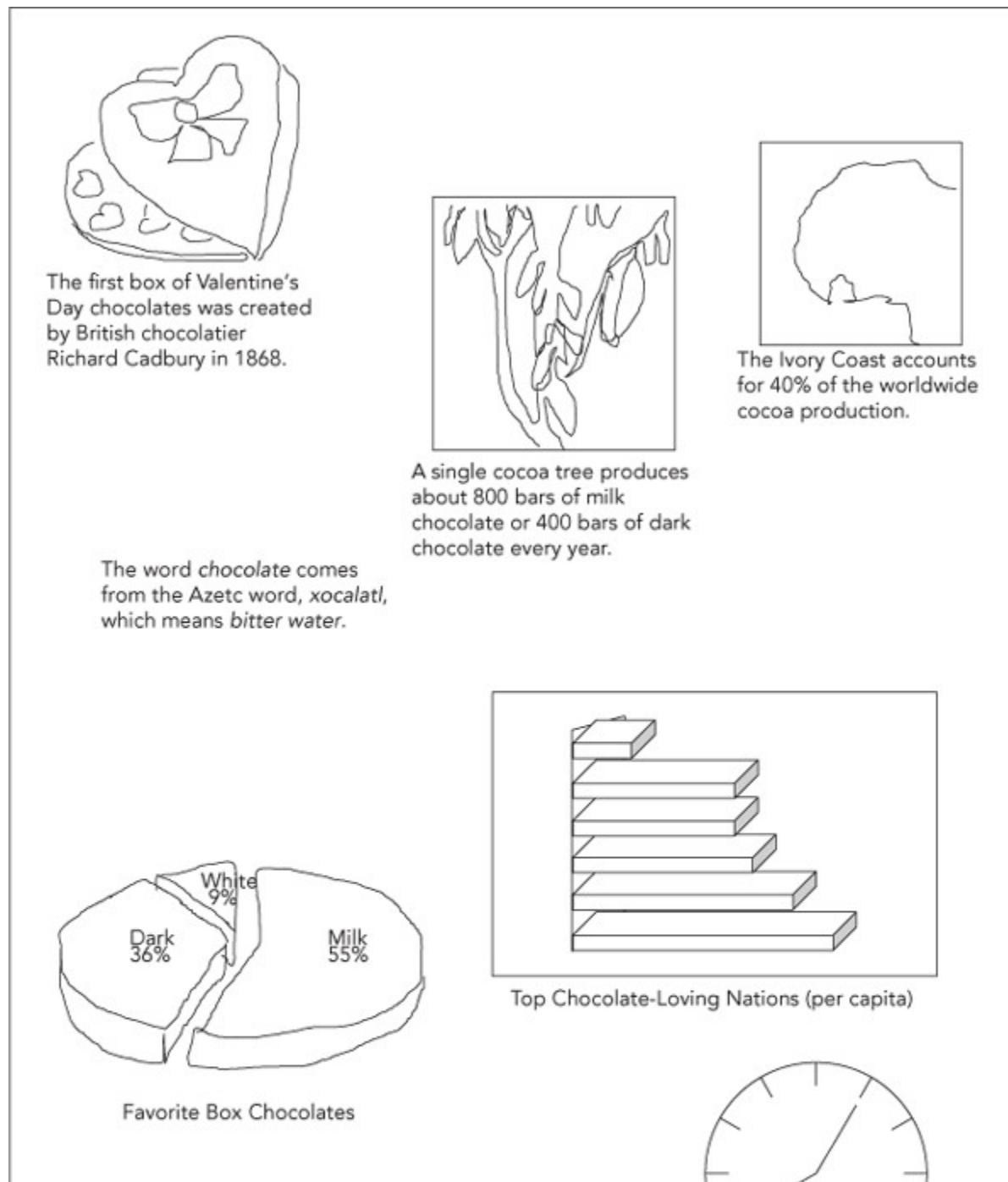


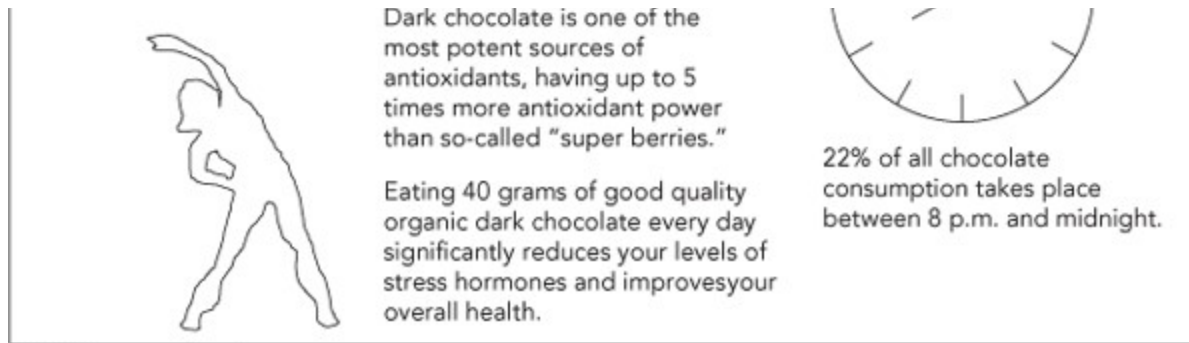
## 3-15e Using the Positioning Styles

Anne wants you to work on the layout for a page that contains an infographic on chocolate. She sketched the layout of the infographic page, as shown in [Figure 3-49](#).

**Figure 3-49**

### Proposed layout of the chocolate infographic





© 2016 Cengage Learning

© 2016 Cengage Learning

Because the placement of the text and figures do not line up nicely within a grid, you'll position each graphic and text box using the CSS positioning styles. Anne has already created the content for this page and written the style sheets to format the appearance of the infographic. You will write the style sheet to layout the infographic contents using the CSS positioning styles.

#### To open the infographic file:

1

Use your editor to open the **pc\_info\_txt.html** file from the html03 ► tutorial folder. Enter ***your name*** and ***the date*** in the comment section of the file and save the document as **pc\_info.html**.

2

Directly after the `title` element, insert the following `link` elements to attach the file to the `pc_reset.css`, `pc_styles3.css`, and `pc_info.css` style sheets.

```
<link href="pc_reset.css" rel="stylesheet" />
<link href="pc_styles3.css" rel="stylesheet" />
<link href="pc_info.css" rel="stylesheet" />
```

3

Take some time to study the structure and content of the `pc_info.html` document. Note that Anne has placed eight information graphics, each within a separate `div` element with a class name of `infobox` and an id name ranging from `info1` to `info8`.

4

Close the file, saving your changes.

Next, you'll start working on the `pc_info.css` file, which will contain the positioning and other design styles for the objects in the infographic. You will begin by formatting the `main` element, which contains the infographics. Because you'll want the position of each infographic to be measured from the top-left corner of this container, you will place the `main` element with relative positioning and extend the height of the container to 1400 pixels so that it can contain all eight of the graphic elements.

#### To format the main element:

1

Use your editor to open the **`pc_info_txt.css`** file from the `html03 ► tutorial` folder. Enter ***your name*** and ***the date*** in the comment section of the file and save the document as **`pc_info.css`**.

2

Go to the Main Styles section and insert the following style rule to format the appearance of the `main` element:

```
main {  
    position: relative;  
    height: 1400px;  
    width: 100%;  
}
```

When you want to position objects in an exact or absolute position within a container, set the `position` property of the container to `relative`.

It will be easier to see the effect of placing the different `div` elements if they are not displayed until you are ready to position them. Add a rule to hide the `div` elements, then as you position each element, you can add a style rule to redisplay it.

3

Directly before the Main Styles section, insert the following style rule to hide all of the infoboxes:

```
div.infobox {display:none;}
```

Figure 3-50 highlights the newly added code in the style sheet.

#### Figure 3-50

#### Setting the display styles of the main element



4

Save your changes to the file and then open the **pc\_info.html** file in your browser. Verify that the browser shows an empty box, about 1400 pixels high, where the infographic will be placed.

Next, you will add a style rule for all of the information boxes so that they are placed within the `main` element using absolute positioning.

### To position the information boxes:

1

Return to the **pc\_info.css** file in your editor and scroll down to the Infographic Styles section.

2

Add the following style rule to set the position type of all of the information boxes.

```
div.infobox {
  position: absolute;
}
```

3

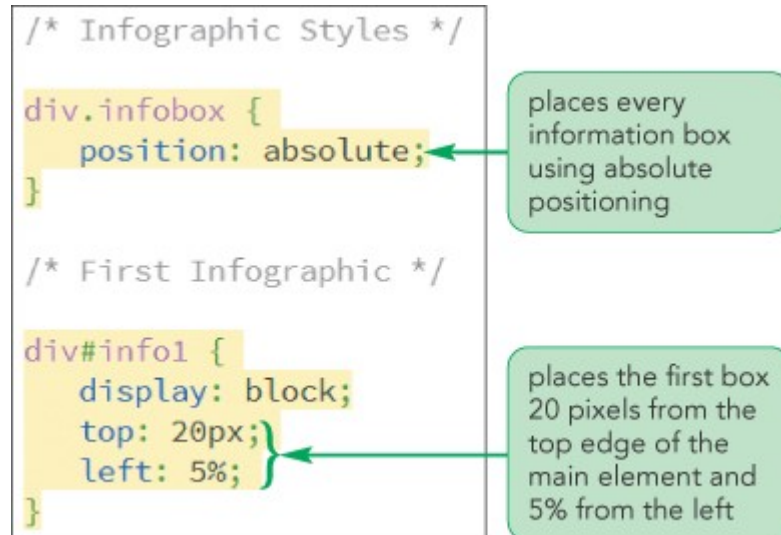
Position the first information box 20 pixels from the top edge of its container and 5% from the left edge.

```
div#infol {
  display: block;
  top: 20px;
  left: 5%;
}
```

Note that we set the `display` property to `block` so that the first information box is no longer hidden on the page. [Figure 3-51](#) highlights the style rules for all of the information boxes and the placement of the first information box.

**Figure 3-51**

### Placing the first information box

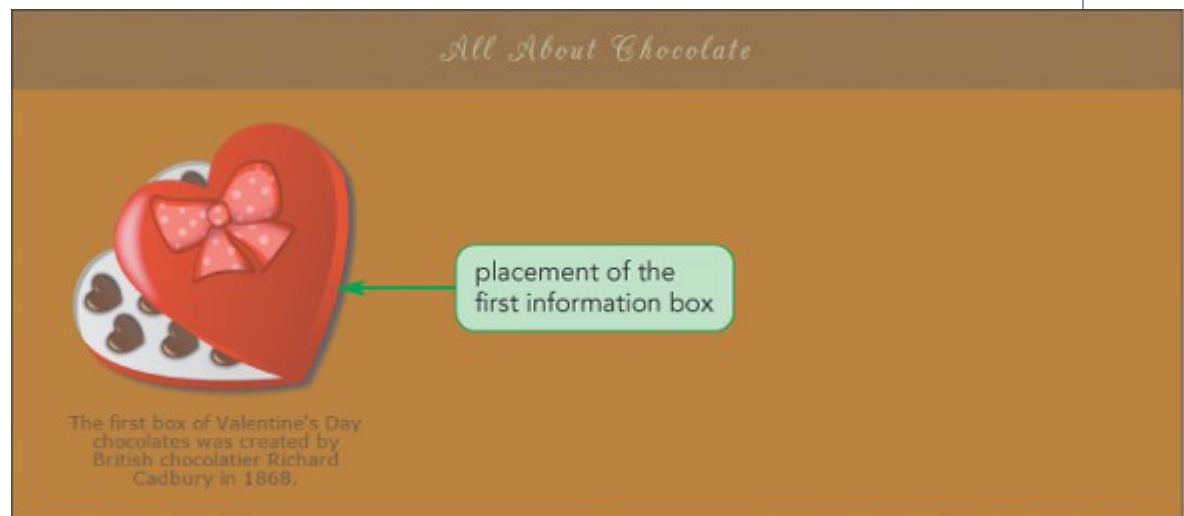


4

Save your changes to the file and then reload the **pc\_info.html** file in your browser. [Figure 3-52](#) shows the placement of the first information box.

**Figure 3-52**

### Appearance of the first information box



Now place the second and third information boxes.

### To place the next two boxes:

1

Return to the **pc\_info.css** file in your editor and go to the Second Infographic section.

2

Add the following style rule to place the second box 185 pixels down from the top of the container and 42% from the left edge.

```
div#info2 {  
    display: block;  
    top: 185px;  
    left: 42%;  
}
```

3

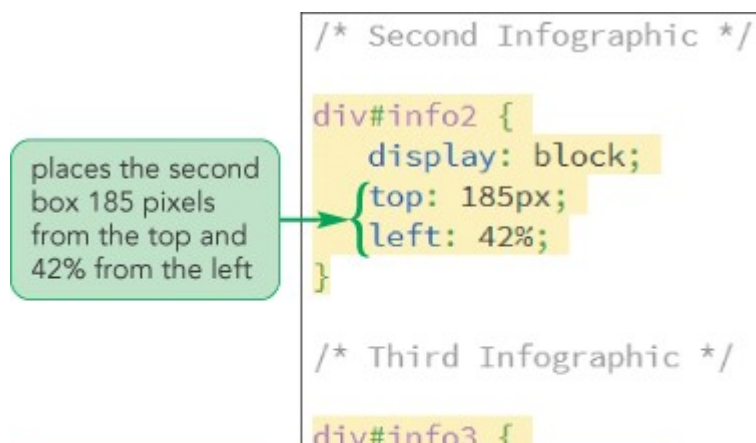
Within the Third Infographic section insert the following style rule to place the third box 135 pixels from the top edge and 75% of the width of its container from the left edge.

```
div#info3 {  
    display: block;  
    top: 135px;  
    left: 75%;  
}
```

Figure 3-53 highlights the style rules to position the second and third information boxes.

**Figure 3-53**

### Positions of the second and third boxes



places the third box 135 pixels from the top and 75% from the left

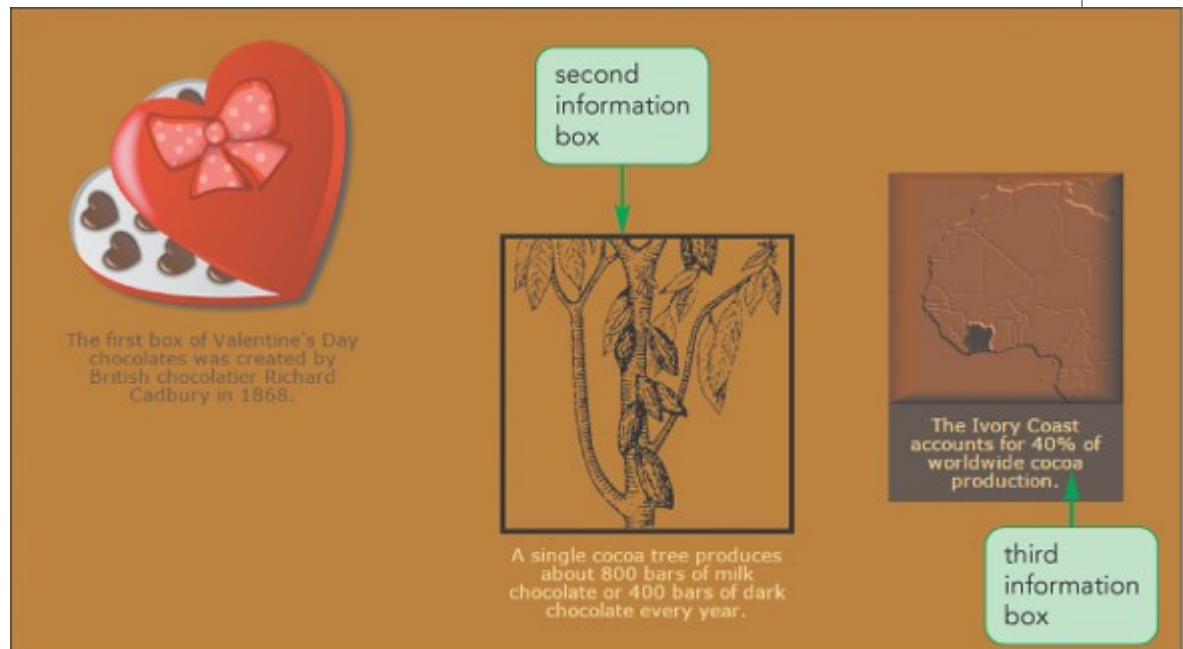
```
display: block;
{
  top: 135px;
  left: 75%;
}
```

4

Save your changes to the file and reload **pc\_info.html** in your browser. [Figure 3-54](#) shows the placement of the first three information boxes.

**Figure 3-54**

### Placement of the first three boxes



Place the next three information boxes.

### To place the next three boxes:

1

Return to the **pc\_info.css** file in your editor, go to the Fourth Infographic section and place the fourth box 510 pixels from the top edge and 8% from the left edge.

```
div#info4 {
  display: block;
  top: 510px;
  left: 8%;
}
```



2

Add the following style rule to the Fifth Infographic section to position the fifth box:

```
div#info5 {  
    display: block;  
    top: 800px;  
    left: 3%;  
}
```

3

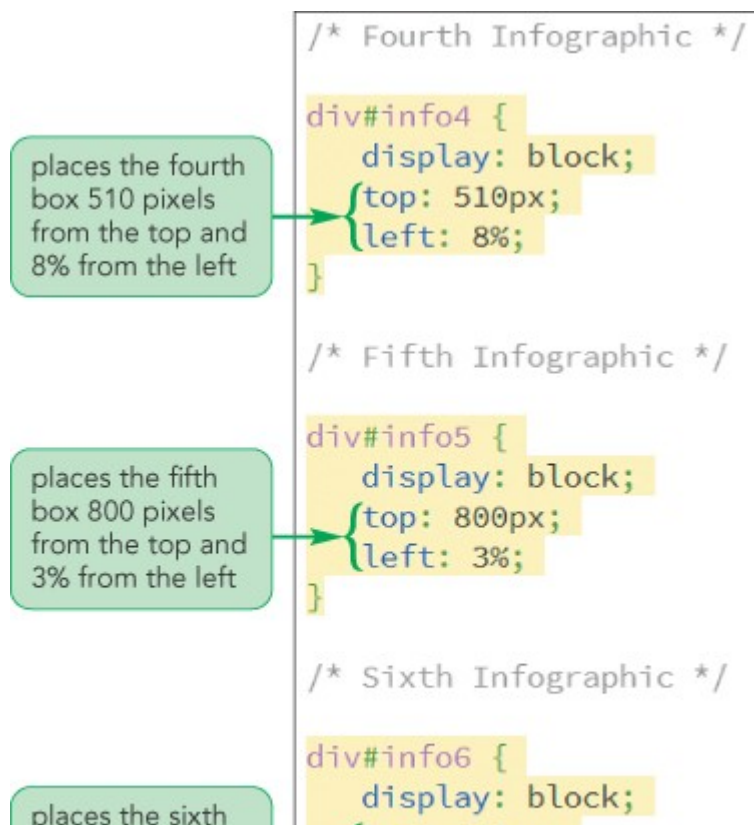
Add the following style rule to the Sixth Infographic section to position the sixth box:

```
div#info6 {  
    display: block;  
    top: 600px;  
    left: 48%;  
}
```

Figure 3-55 highlights the positioning styles for the fourth, fifth, and sixth information boxes.

**Figure 3-55**

### Positions of the fourth, fifth, and sixth boxes





box 600 pixels  
from the top and  
48% from the left

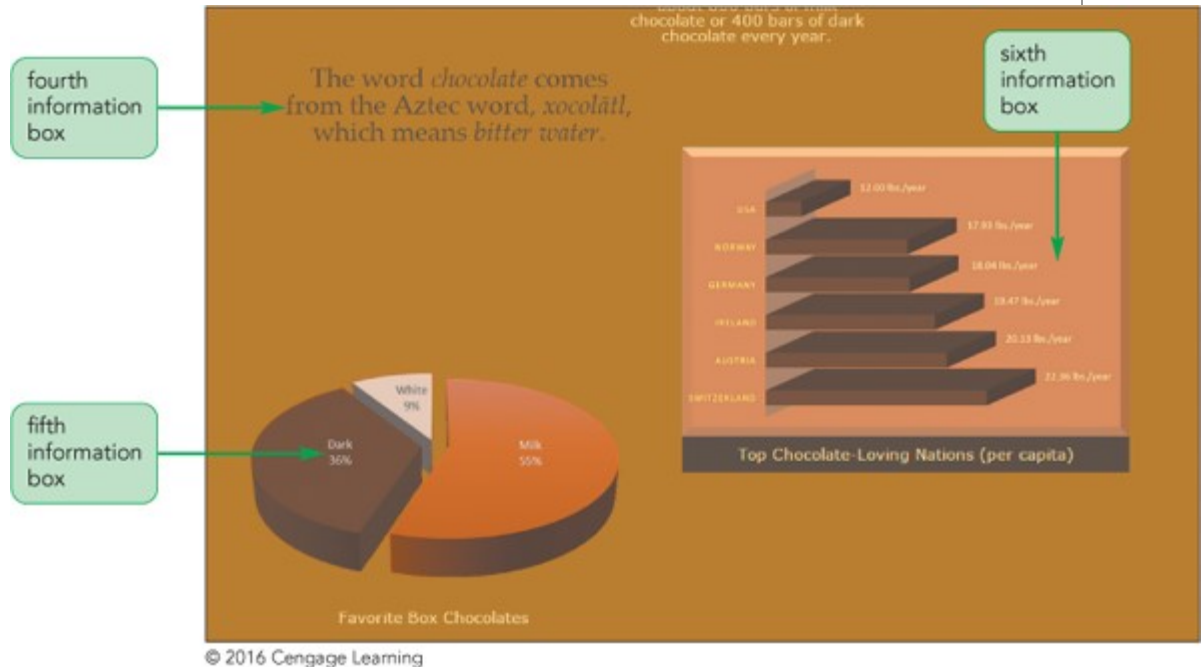
```
{
  top: 600px;
  left: 48%;
}
```

4

Save your changes to the file and reload **pc\_info.html** in your browser. [Figure 3-56](#) shows the revised layout of the infographic.

**Figure 3-56**

### Placement of the next three boxes



© 2016 Cengage Learning

Complete the layout of the infographic by placing the final two boxes on the page.

### To place the last two boxes:

1

Return to the **pc\_info.css** file in your editor, go to the Seventh Infographic section and insert the following style rules:

```
div#info7 {
  display: block;
  top: 1000px;
  left: 68%;
}
```

2

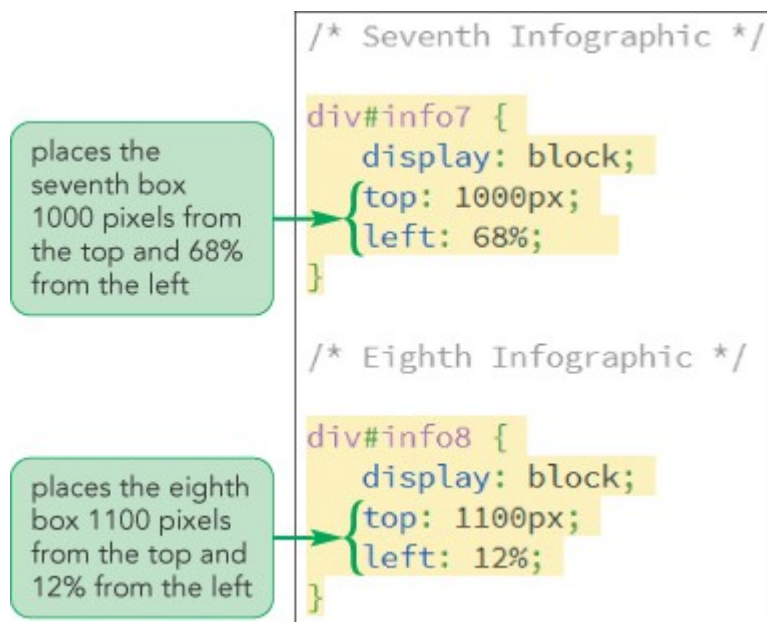
Add the following style rules to the Eighth Infographic section:

```
div#info8 {  
    display: block;  
    top: 1100px;  
    left: 12%;  
}
```

Figure 3-57 highlights the style rules for the seventh and eighth information boxes.

**Figure 3-57**

### Positioning the seventh and eighth boxes



3

Scroll up to before the Main Styles section and delete the style rule `div.infobox {display: none;}` because you no longer need to hide any information boxes.

4

Save your changes to the file and reload **pc\_info.html** in your browser. Figure 3-58 show the complete layout of the eight boxes in the infographic.

**Figure 3-58**

### Final layout of the infographic



The first box of Valentine's Day chocolates was created by British chocolatier Richard Cadbury in 1868.

A single cocoa tree produces about 800 bars of milk chocolate or 400 bars of dark chocolate every year.

The Ivory Coast accounts for 40% of worldwide cocoa production.

The word *chocolate* comes from the Aztec word, *xocolatl*, which means *bitter water*.

**Favorite Box Chocolates**

Chocolate Type	Percentage
Milk	55%
Dark	36%
White	9%

**Top Chocolate-Loving Nations (per capita)**

Nation	Per Capita Consumption (kg/year)
USA	12.00
NORWAY	17.00
GERMANY	18.04
IRELAND	23.47
AUSTRIA	20.55
SWITZERLAND	22.36

**seventh information box**

Dark chocolate is one of the most potent sources of antioxidants, having up to 5 times more antioxidant power than so-called "super berries".

Eating 40 grams of good quality organic dark chocolate every day significantly reduces your levels of stress hormone and improves your overall health.

**eighth information box**

22% of all chocolate consumption takes place between 8pm and midnight.

Pandaisia Chocolate © 2017 All Rights Reserved

© 2016 Cengage Learning

Anne likes the appearance of the infographic, but she is concerned about its length. She would like you to reduce the height of the infographic so that it appears within the boundaries of the browser window. This change will create overflow because the content is longer than the new height. You will read more about overflow and how to handle it now.

Insight

**Creating an Irregular Line Wrap**

Many desktop publishing and word-processing programs allow designers to create irregular line wraps in which the text appears to flow tightly around an image. This is not easily done in a web page layout because all images appear as rectangles rather than as irregularly shaped objects. However, with the aid of a graphics package, you can simulate an irregularly shaped image.

The trick is to use your graphics package to slice the image horizontally into several pieces and then crop the individual slices to match the edge of the image you want to display. Once you've edited all of the slices, you can use CSS to stack the separate slices by floating them on the left or right margin, displaying each slice only after the previous slice has been cleared. For example, the following style rule stacks all inline images that belong to the "slice" class on the right margin:

```
img.slice {  
    clear: right;  
    float: right;  
    margin-top: 0px;  
    margin-bottom: 0px;  
}
```

Now any text surrounding the stack of images will tightly match the image's boundary, creating the illusion of an irregular line wrap. Note that you should always set the top and bottom margins to 0 pixels so that the slices join together seamlessly.

Chapter 3: Designing a Page Layout: 3-15e Using the Positioning Styles

Book Title: New Perspectives on HTML5, CSS3, and JavaScript

Printed By: Kirsten Markley (markleyk@otc.edu)

© 2018 Cengage Learning, Cengage Learning

© 2018 Cengage Learning Inc. All rights reserved. No part of this work may be reproduced or used in any form or by any means - graphic, electronic, or mechanical, or in any other manner - without the written permission of the copyright holder.

## 3-16 Handling Overflow

The infographic is long because it displays several information boxes. If you reduce the height of the infographic you run the risk of cutting off several of the boxes that will no longer fit within the reduced infographic. However you can control how your browser handles this excess content using the following `overflow` property

```
overflow: type;
```

where `type` is `visible` (the default), `hidden`, `scroll`, or `auto`. A value of `visible` instructs browsers to increase the height of an element to fit the overflow content. The `hidden` value keeps the element at the specified height and width, but cuts off excess content. The `scroll` value keeps the element at the specified dimensions, but adds horizontal and vertical scroll bars to allow users to scroll through the overflowed content. Finally, the `auto` value keeps the element at the specified size, adding scroll bars only as they are needed. [Figure 3-59](#) shows examples of the effects of each overflow value on content that is too large for its space.

**Figure 3-59**

### Values of the overflow property



CSS3 also provides the `overflow-x` and `overflow-y` properties to handle overflow specifically in the horizontal and vertical directions.

## Working with Overflow

- To specify how the browser should handle content that overflows the element's boundaries, use the property

`overflow: type;`

where `type` is `visible` (the default), `hidden`, `scroll`, or `auto`.

You decide to limit the height of the infographic to 450 pixels and to set the `overflow` property to `auto` so that browsers displays scroll bars as needed for the excess content.

### To apply the `overflow` property:

1

Return to the `pc_info.css` file in your editor and go to the Main Styles section.

2

Within the style rule for the `main` selector, insert the property `overflow: auto;`.

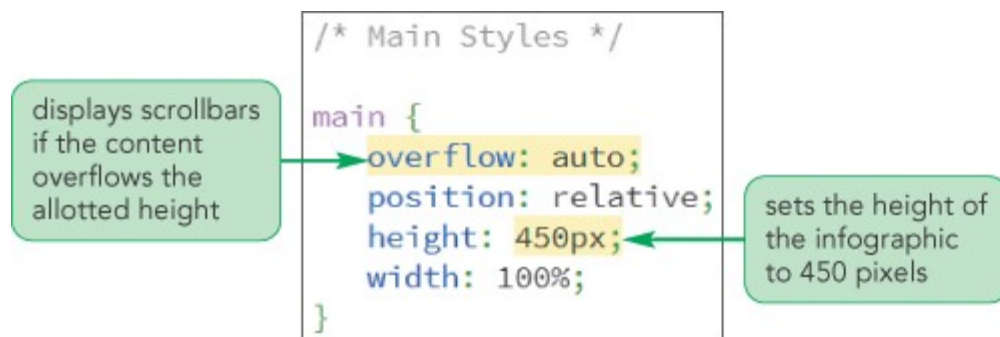
3

Reduce the height of the element from 1400px to **450px**.

Figure 3-60 highlights the revised code in the style rule.

**Figure 3-60**

### Setting the overflow property



4

Close the file, saving your changes.



5

Reload the **pc\_info.html** file in your browser. As shown in [Figure 3-61](#), the height of the infographic has been reduced to 450 pixels and scrollbars have been added that you can use to view the entire infographic.

**Figure 3-61****Final layout of the infographic page**

© 2016 Cengage Learning

6

Close any open files now.

**Insight****Managing White Space with CSS**

Scroll bars for overflow content are usually placed vertically so that you scroll down to view the extra content. In some page layouts, however, you may want to view content in a horizontal rather than a vertical direction. You can accomplish this by adding the following style properties to the element:

```
overflow: auto;  
white-space: nowrap;
```

The `white-space` property defines how browsers should handle white space in the rendered document. The default is to collapse consecutive occurrences of white space into a single blank space and to automatically wrap text to a new line if it extends beyond the width of the container. However, you can set the `white-space` property of the element to `nowrap` to keep inline content on a single line, preventing line wrapping. With the content thus confined to a single line, browsers will display only horizontal scroll bars for the overflow content. Other values of the `white-space` property include `normal` (for default handling of white space), `pre` (to preserve all white space from the HTML file), and `pre-wrap` (to preserve white space but to wrap excess content to a new line).

Chapter 3: Designing a Page Layout: 3-16 Handling Overflow  
Book Title: New Perspectives on HTML5, CSS3, and JavaScript  
Printed By: Kirsten Markley (markleyk@otc.edu)  
© 2018 Cengage Learning, Cengage Learning

© 2018 Cengage Learning Inc. All rights reserved. No part of this work may be reproduced or used in any form or by any means - graphic, electronic, or mechanical, or in any other manner - without the written permission of the copyright holder.



## 3-17 Clipping an Element

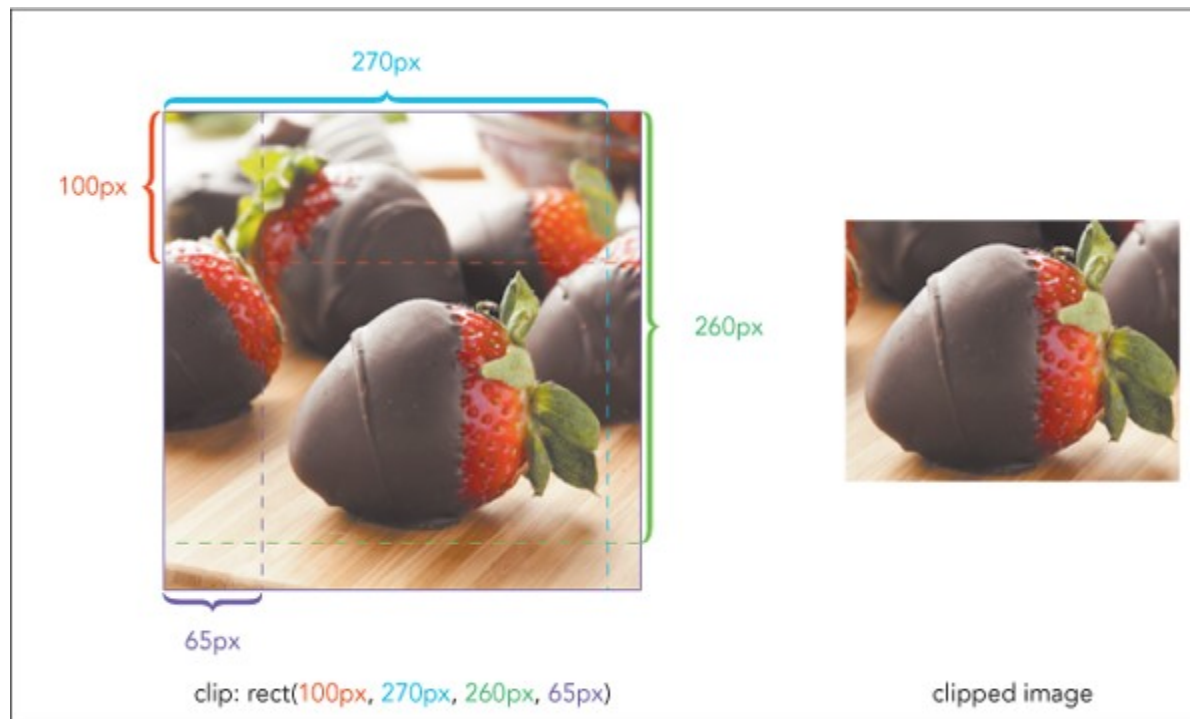
Closely related to the `overflow` property is the `clip` property, which defines a rectangular region through which an element's content can be viewed. Anything that lies outside the boundary of the rectangle is hidden. The syntax of the `clip` property is

```
clip: rect(top, right, bottom, left);
```

where *top*, *right*, *bottom*, and *left* define the coordinates of the clipping rectangle. For example, a clip value of `rect(100px, 270px, 260px, 65px)` defines a clip region whose top and bottom boundaries are 100 and 260 pixels from the top edge of the element, and whose right and left boundaries are 270 and 65 pixels from the element's left edge. See [Figure 3-62](#).

**Figure 3-62**

### Clipping an image



© Brent Hofacker/Shutterstock.com

© Brent Hofacker/Shutterstock.com

The top, right, bottom, and left values also can be set to `auto`, which matches the specified edge of the clipping region to the edge of the parent element. A clip value of `rect(10, auto, 125, 75)` creates a clipping rectangle whose right edge matches the right edge of the parent element. To remove clipping completely, apply the style `clip: auto`. Clipping can only be

applied when the object is placed using absolute positioning.

## Reference

### Clipping Content

- To clip an element's content, use the property

```
clip: rect(top, right, bottom, left);
```

where *top*, *right*, *bottom*, and *left* define the coordinates of the clipping rectangle.

- To remove clipping for a clipped object, use

```
clip: auto;
```

Chapter 3: Designing a Page Layout: 3-17 Clipping an Element  
Book Title: New Perspectives on HTML5, CSS3, and JavaScript  
Printed By: Kirsten Markley (markleyk@otc.edu)  
© 2018 Cengage Learning, Cengage Learning

© 2018 Cengage Learning Inc. All rights reserved. No part of this work may be reproduced or used in any form or by any means - graphic, electronic, or mechanical, or in any other manner - without the written permission of the copyright holder.

## 3-18 Stacking Elements

Positioning elements can sometimes lead to objects that overlap each other. By default, elements that are loaded later by the browser are displayed on top of elements that are loaded earlier. In addition, elements placed using CSS positioning are stacked on top of elements that are not. To specify a different stacking order, use the following `z-index` property:

```
z-index: value;
```

where *value* is a positive or negative integer, or the keyword `auto`. As shown in [Figure 3-63](#), objects with the highest `z-index` values are placed on top of other page objects. A value of `auto` stacks the object using the default rules.

**Figure 3-63**

**Using the `z-index` property to stack elements**



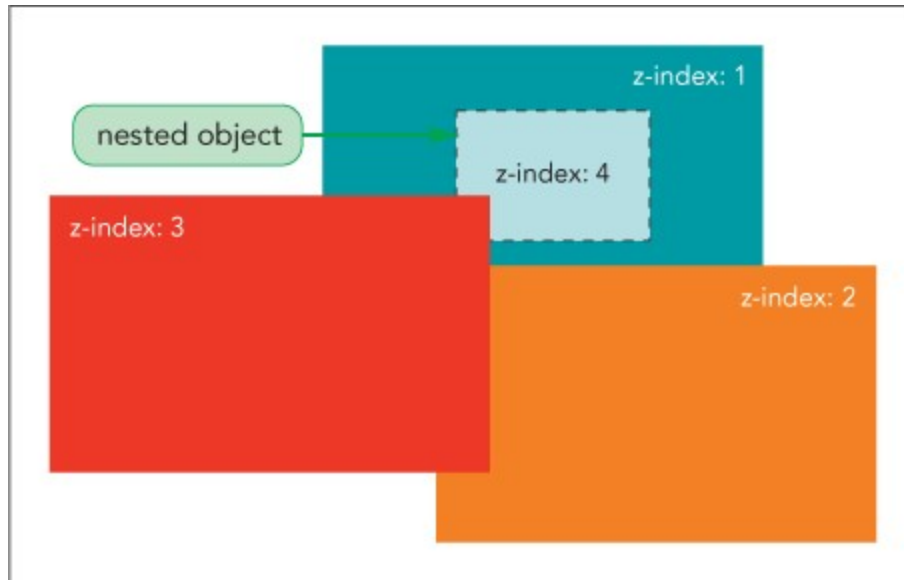
© 2016 Cengage Learning

© 2016 Cengage Learning

The `z-index` property works only for elements that are placed with absolute positioning. Also, an element's `z-index` value determines its position relative only to other elements that share a common parent; the style has no impact when applied to elements with different parents. [Figure 3-64](#) shows a layout in which the object with a high `z-index` value of 4 is still covered because it is nested within another object that has a low `z-index` value of 1.

**Figure 3-64**

## Stacking nested objects



© 2016 Cengage Learning

© 2016 Cengage Learning

You do not need to include the `z-index` property in your style sheet because none of the elements in the infographic page are stacked upon another.

### Proskills

#### Problem Solving: Principles of Design

Good web page design is based on the same common principles found in other areas of art, which include balance, unity, contrast, rhythm, and emphasis. A pleasing layout involves the application of most, if not all, of these principles, which are detailed below:

- **Balance** involves the distribution of elements. It's common to think of balance in terms of **symmetrical balance**, in which similar objects offset each other like items on a balance scale; but you often can achieve more interesting layouts through asymmetrical balance, in which one large page object is balanced against two or more smaller objects.
- **Unity** is the ability to combine different design elements into a cohesive whole. This is accomplished by having different elements share common colors, font styles, and sizes. One way to achieve unity in a layout is to place different objects close to each other, forcing your viewers' eyes to see these items as belonging to a single unified object.
- **Contrast** consists of the differences among all of the page elements. To create an effective design, you need to vary the placement, size, color, and

general appearance of the objects in the page so that your viewers' eyes aren't bored by the constant repetition of a single theme.

- **Rhythm** is the repetition or alteration of a design element in order to provide a sense of movement, flow, and progress. You can create rhythm by tiling the same image horizontally or vertically across the page, by repeating a series of elements that progressively increase or decrease in size or spacing, or by using elements with background colors of the same hue but that gradually vary in saturation or lightness.
- **Emphasis** involves working with the focal point of a design. Your readers need a few key areas to focus on. It's a common design mistake to assign equal emphasis to all page elements. Without a focal point, there is nothing for your viewers' eyes to latch onto. You can give a page element emphasis by increasing its size, by giving it a contrasting color, or by assigning it a prominent position in the page.

Designers usually have an intuitive sense of what works and what doesn't in page design, though often they can't say why. These design principles are important because they provide a context in which to discuss and compare designs. If your page design doesn't feel like it's working, evaluate it in light of these principles to identify where it might be lacking.

Anne is pleased with the final design of the infographic page and all of the other pages you've worked on. She'll continue to develop the website and test her page layouts under different browsers and screen resolutions. She'll get back to you with future projects as she continues the redesign of the Pandaisia Chocolates website.

Chapter 3: Designing a Page Layout: 3-18 Stacking Elements  
Book Title: New Perspectives on HTML5, CSS3, and JavaScript  
Printed By: Kirsten Markley (markleyk@otc.edu)  
© 2018 Cengage Learning, Cengage Learning

© 2018 Cengage Learning Inc. All rights reserved. No part of this work may be reproduced or used in any form or by any means - graphic, electronic, or mechanical, or in any other manner - without the written permission of the copyright holder.